

THE EFFECTS OF CLUSTERING ON THE MEDIUM AND LARGE-SCALE CAPACITATED LOCATION-ROUTING PROBLEM

Jacoba Hendrina Bührmann

Supervisor: Professor Montaz Ali
Co-supervisor: Doctor Ian Campbell

A thesis submitted to the Faculty of Engineering and the Built Environment,
University of the Witwatersrand, Johannesburg, in fulfilment of the
requirements for the degree of Doctor of Philosophy.

February 23, 2016



Declaration

I declare that this thesis is my own unaided work. It is being submitted for the Degree of Doctor of Philosophy to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

(Signature of Candidate)

_____ day of _____ 20_____ in _____

Abstract

This work investigates the effectiveness of using clustering methods in solving various capacitated location-routing problems (CLRP) for medium- and large-scale datasets, with up to 20 000 datapoints. Different clustering methods as well as hybrid clustering methods are tested and compared.

A new problem called the planar CLRP (plCLRP) is introduced. Based on the results from the clustering methods, cluster-based approaches are suggested to solve variants of the CLRP. These include the Hamiltonian p -median problem (HpMP), the planar CLRP (plCLRP), the concentrator discrete CLRP (cdCLRP) and the standard discrete CLRP (sdCLRP). A new method called the two-phased proportional regret ordering based unconstrained to constrained (PROBUC) method is also proposed to create capacitated clusters.

The focus falls on finding effective non-exponential time algorithms that can be used to solve large-scale problems with good results. A full set of results for each problem are presented and comparisons are made with known results from the literature where possible.

The PLRP (periodic location-routing problem) introduced by Prodhon and Prins (2008), is also investigated. A change in the current problem formulation, as provided by Prodhon (2011), is proposed to enforce single-source constraints across time horizon and limit the maximum number of vehicles.

An approach to solve the PLRP, based on the cluster-based approaches to solve the discrete CLRPs, is suggested. The results of the cluster-based approach are compared to best-known solutions for existing PLRP instances given by Prodhon (2009a). A set of large scale PLRP instances are introduced, based on instances generated by Harks et al. (2013) for the sdCLRP.

Samevatting

In hierdie studie word die gebruik van groeperingsmetodes ondersoek om verskeie ligging-roeteringsprobleme met kapasiteitbeperkings (LRPK) op te los. Dit sluit die oplos van middelmatige en grootskaalse datastelle tot 'n maksimum van 20 000 datapunte in. Verskillende groeperingsmetodes, asook hibriede groeperingsmetodes, word met mekaar vergelyk en getoets.

Gebaseer op die bostaande ondersoek, word algoritmes wat van groeperingsmetodes gebruik maak voorgestel om vier variasies van die LRPK mee op te los. Dit sluit onder andere die Hamiltoniaanse p-mediaanprobleem, die kontinue LRPK en twee diskrete LRPK variasies, naamlik die konsentrator-diskrete - en die standaard-diskrete probleem in. 'n Nuwe metode, genaamd die twee-fase proporsionele-berou gesorteerde onbeperk-tot-beperk-metode word bekendgestel om gegroepeerde oplossings volgens kapasiteitbeperkings aan te pas.

Die fokus val op nie-eksponensiële-tyd algoritmes wat gebruik kan word om grootskaalse probleme met goeie resultate te kan oplos. Die voorgestelde algoritmes is vir al vier variasies op groot datastelle getoets en met bestaande resultate vergelyk waar moontlik.

Die periodiese LRPK, voorgestel deur Prodhon en Prins (2008), word ook ondersoek. 'n Aanpassing op die huidige wiskundige model, soos gegee deur Prodhon (2011), word voorgestel. Beperkings om te voorkom dat kliënte deur meer as een fasiliteit bedien word, asook 'n maksimum limiet op die aantal voertuie, word voorgestel.

'n Algoritme gebaseer op die groeperingsgebaseerde algoritmes wat vir die diskrete LRPKs voorgestel is, word vir die periodiese LRPK voorgestel. Die algoritme word vergelyk met akademiese datastelle van Prodhon (2009a). Drie grootskaalse datastelle, van Harks et al. (2013) vir die diskrete LRPK, word aangepas vir die periodiese LRPK en die resultate word hier bekend gestel.

Aan my ouers. Julle staan altyd by my. Dankie vir alles wat julle vir my doen en julle onophoudelike liefde en ondersteuning.

Aan Jan-Rudolph. Dankie vir jou liefde, jou ondersteuning en die wonderlike lewe wat ons saam het.

Aan Sunelle en Johann. Julle is baie spesiaal. Dankie dat julle my lewe elke dag opnuut ophelder.

And to God. What an awesome God we serve. Eph. 3:20 - 21

Acknowledgements

I would like to acknowledge the following people for their contributions towards this thesis:

- Professor Montaz Ali and Doctor Ian Campbell from the University of the Witwatersrand, as supervisor and co-supervisor, thank you for your feedback and encouragement in completing this study.
- An anonymous source for the medium size case study data.
- Professor David Lubinsky and colleagues for opening up the world of logistics.
- Ms Jean le Roux, Operations Research senior lecturer UNISA, for your encouragement, advice and guidance.

Contents

Declaration	i
Abstract	ii
Samevatting	iii
Dedication	iv
Acknowledgements	v
List of Figures	xi
List of Tables	xiv
List of Acronyms	xviii
1 Introduction	1
1.1 Background and motivation	1
1.2 Problem statement	2
1.2.1 Limitations and scope	3
1.2.2 Thesis objectives	3
1.3 Methodology	4
1.3.1 Data preparation and experiments	4
1.3.2 Approach	5
1.4 Thesis contribution	7
1.5 Thesis overview	7
2 Literature review - clustering methods	8
2.1 SAHN clustering methods	8
2.1.1 The single linkage method	10
2.1.2 The complete linkage method	10
2.1.3 The average linkage method	10
2.1.4 The weighted average-linkage method	11
2.1.5 The centroid linkage method	11
2.1.6 The weighted centroid (median) linkage method	12
2.1.7 Ward's method	12

2.1.8	The Lance-Williams formula for SAHN methods	13
2.1.9	CPU times of the SAHN methods	14
2.1.10	Stopping rules in SAHN clustering	15
2.2	Iterative partitioning methods	16
2.2.1	The k-means method	18
2.2.2	The h-means method	19
2.2.3	The j-means (k-medoid) method	19
2.2.4	The PAM (h-medoid) method	20
2.2.5	The k-median method	20
2.2.6	The h-median method	21
2.2.7	Starting solutions for iterative partitioning methods	21
2.2.8	Methods to prevent the number of clusters from declining	22
2.2.9	The CLUSTER partitioning methods	22
2.2.10	The AS 136 method	23
2.3	Graph-based methods	24
2.3.1	MST graph-based clustering method	25
2.3.2	RNG graph-based clustering method	26
2.3.3	GG graph-based clustering method	26
2.4	Nearest neighbour methods	27
2.4.1	The k-near neighbours method	27
2.4.2	The mutual neighbourhood value (MNV) method	27
2.5	Density-based methods	28
2.5.1	The k_{th} nearest neighbour clustering method	29
2.5.2	The hybrid clustering density-based method	30
2.6	Hybrid clustering methods	31
2.7	Clustering criteria	31
2.8	Distance calculations	34
2.9	Summary	34
3	Literature review - distribution network problems	36
3.1	The facility location problem (FLP)	38
3.1.1	The discrete CFLP	39
3.1.2	Solution approaches for the discrete CFLP	44
3.1.3	The continuous CFLP	55
3.1.4	Solution approaches for the continuous CFLP	59
3.2	The vehicle routing problem (VRP)	63
3.2.1	The CVRP	63
3.2.2	Solution approaches for the CVRP	65
3.2.3	The PVRP	77
3.2.4	Solution approaches for the PVRP	79
3.3	The location-routing problem (LRP)	81
3.3.1	The Hamiltonian p-median problem (HpMP)	82

3.3.2	Solution approaches for the HpMP	85
3.3.3	The planar or continuous location-routing problem (pILRP) .	86
3.3.4	Solution approaches for the pILRP	88
3.3.5	The discrete capacitated location-routing problem (CLRP) .	91
3.3.6	Solution approaches for the CLRP	94
3.4	The periodic location-routing problem (PLRP)	106
3.4.1	Solution approaches for the PLRP	109
3.5	Summary	113
4	An analysis of clustering methods	115
4.1	Methodology	115
4.1.1	Data preparation	116
4.1.2	Evaluation criteria	116
4.1.3	Method	117
4.2	Computational results	124
4.3	Discussion	125
4.3.1	SAHN clustering	132
4.3.2	Iterative partitioning clustering	139
4.3.3	Graph-based clustering	153
4.3.4	Nearest neighbour clustering	154
4.3.5	Density-based clustering	157
4.3.6	Hybrid clustering methods	160
4.4	Conclusions	169
5	Cluster-based approaches to solve variants of the CLRP	170
5.1	Solving the HpMP	171
5.1.1	Proposed new problem formulation	171
5.1.2	A new cluster-based method to solve the HpMP	172
5.1.3	Methodology and data preparation	175
5.1.4	Computational results	177
5.1.5	Discussion	177
5.1.6	Conclusions - HpMP	181
5.2	Solving the pILRP	182
5.2.1	Proposed new problem formulation	182
5.2.2	Determining an initial number for depots and vehicles	184
5.2.3	Creating capacitated depot and route clusters	186
5.2.4	The two-phase PROBUC method	189
5.2.5	A new cluster-based method to solve the pILRP	206
5.2.6	Methodology and data preparation	209
5.2.7	Computational results	211
5.2.8	Discussion	212
5.2.9	Conclusions - pILRP	213

5.3	Solving the cdCLRP	215
5.3.1	A new cluster-based method to solve the cdCLRP	215
5.3.2	Methodology and data preparation	218
5.3.3	Computational results	219
5.3.4	Discussion	219
5.3.5	Conclusions - cdCLRP	219
5.4	Solving the sdCLRP	221
5.4.1	The effect of using the cdCLRP location phase in the sdCLRP	221
5.4.2	A new cluster-based method to solve the sdCLRP	222
5.4.3	Methodology and data preparation	226
5.4.4	Computational results	227
5.4.5	Discussion	228
5.4.6	Conclusions - sdCLRP	229
5.5	Conclusions	229
6	A cluster-based approach to solve the PLRP	231
6.1	Proposed adapted problem formulation	231
6.2	A new cluster-based method to solve the PLRP	234
6.3	Methodology and data preparation	235
6.4	Computational results	237
6.5	Discussion	237
6.6	Conclusions	240
7	Conclusions	241
7.1	Contribution	241
7.1.1	Clustering methods	241
7.1.2	The HpMP	242
7.1.3	The plCLRP	243
7.1.4	The cdCLRP	243
7.1.5	The sdCLRP	244
7.1.6	The PLRP	244
7.1.7	Test instances	245
7.2	Conclusive remarks	245
7.2.1	Observations regarding clustering methods	246
7.2.2	Visualisation of data points	247
7.3	Areas for future research	247
	References	249
	Appendix A Clustering method results	259
A.1	Clustering method results	260
A.1.1	Clustering method results - DataS1	260
A.1.2	Clustering method results - DataS2	272

A.1.3	Clustering method results - DataS3	284
A.1.4	Clustering method results - USA13509	296
A.2	Variations of the hybrid density method	299
A.3	Hybrid clustering methods	300
A.3.1	Hybrid SAHN - SAHN clustering methods - DataS3	300
A.3.2	Hybrid SAHN - partitioning clustering methods - DataS3	301
A.3.3	Stepwise aggregation partitioning clustering methods - DataS3	307
A.3.4	Hybrid partitioning clustering - DataS3	308
Appendix B Displaying points in Google maps		311
Appendix C Program optimisation techniques		313
C.1	The use of arrays versus recalculations	313
C.2	Methods to store a big distance matrix	314
C.3	Sorting methods	315
C.4	Keeping track of a cluster's nearest neighbours	316
C.5	Using coordinates instead of distance calculations for comparisons	316
C.6	Identifying adjacent clusters	317
C.7	Arrays versus text string variables	317
C.8	Linked lists	318
C.9	Summary	318

List of Figures

2.1	A dendrogram representation of SAHN clustering.	9
2.2	An illustration of the hierarchical clustering distance calculation used by the Lance-Williams parameters.	14
2.3	An illustration of the k-means method.	17
2.4	The regions of influence for the RNG and GG, (Jain and Dubes, 1998, p. 125).	26
3.1	Typical costs associated with the FLP problem.	42
3.2	An illustration of the discrete FLP problem.	43
3.3	The tree algorithm as illustrated by Papadimitriou and Steiglitz (1998).	71
3.4	Christofides algorithm as illustrated by Papadimitriou and Steiglitz (1998).	71
3.5	The four basic neighbourhood exchanges of the Granular Tabu Search, (Toth and Vigo, 2003).	76
3.6	An illustration of the discrete LRP problem.	81
3.7	An example of the impact of outlier points on the placement of a depot in the LRP.	82
3.8	An example of customer assignments based on shortest distance in the CLRP.	92
3.9	The ECWA method as illustrated by Prins et al. (2006).	96
4.1	A summary of the results of the SSR measure for the various clustering methods using DataS3.	126
4.2	A summary of the results of the MSR measure for the various clustering methods using DataS3.	127
4.3	A summary of the results of the PD measure for the various clustering methods using DataS3.	128
4.4	A summary of the results of the ADM measure for the various clustering methods using DataS3.	129
4.5	A summary of the results of the AWCD measure for the various clustering methods using DataS3.	130
4.6	The CPU times for the various clustering methods using DataS3, up to 5 000 seconds.	131
4.7	An illustration of the single linkage method using DataS3.	135
4.8	An illustration of the complete linkage method using DataS3.	136
4.9	An illustration of the average linkage method using DataS3.	136
4.10	An illustration of the weighted average linkage method using DataS3.	137

4.11	An illustration of the centroid linkage method using DataS3.	137
4.12	An illustration of the weighted centroid linkage method using DataS3. . .	138
4.13	An illustration of the Ward's method using DataS3.	138
4.14	The results of the PAM clustering method using DataS2.	141
4.15	An illustration of the k-means partitioning method using DataS3.	145
4.16	An illustration of the h-means partitioning method using DataS3.	145
4.17	An illustration of the j-means partitioning method using DataS3.	146
4.18	An illustration of the PAM (h-medoid) partitioning method using DataS3.	146
4.19	An illustration of the k-median partitioning method using DataS3.	147
4.20	An illustration of the h-median partitioning method using DataS3.	147
4.21	An illustration of the best h-median starting solution using DataS3. . . .	148
4.22	An illustration of the CLUSTER k-means partitioning method using DataS3.	149
4.23	An illustration of the CLUSTER h-means partitioning method using DataS3.	149
4.24	An illustration of the CLUSTER j-means partitioning method using DataS3.	150
4.25	An illustration of the CLUSTER PAM partitioning method using DataS3.	150
4.26	An illustration of the CLUSTER k-median partitioning method using DataS3.	151
4.27	An illustration of the CLUSTER h-median partitioning method using DataS3.	151
4.28	A zoomed in view of the solution created by the CLUSTER PAM method using DataS3.	152
4.29	An illustration of the AS 136 algorithm using DataS3.	152
4.30	An illustration of the MST graph-based method using DataS3.	154
4.31	An illustration of the GG graph-based method using DataS3.	155
4.32	An illustration of the RNG graph-based method using DataS1.	155
4.33	An illustration of the k-near neighbours clustering method using DataS3.	156
4.34	An illustration of the density-based k_{th} nearest neighbour clustering method using DataS3.	158
4.35	An illustration of the CLUSTER h-median density-based variant 1 method using DataS3.	159
4.36	An illustration of the CLUSTER h-median density-based variant 3 method using DataS3.	159
4.37	The SSR values for the hybrid single and complete linkage - Ward's method.	160
4.38	The SSR values for the hybrid single and complete linkage - Ward's method with 3 000 – 4 000 aggregated clusters.	161
4.39	The CPU times for the hybrid single and complete linkage - Ward's method.	161
4.40	The SSR values for the hybrid single linkage - partitioning methods. . . .	162
4.41	The SSR values for the hybrid complete linkage - partitioning methods. .	163
4.42	The SSR values for the best SAHN - partitioning methods with 2 500 – 7 500 aggregated clusters.	163
4.43	The CPU times of the hybrid SAHN - partitioning methods.	164
4.44	The CPU times of the hybrid SAHN - partitioning methods with best SSR values.	165

4.45	The hybrid single linkage - h-median method with 6 500 aggregated clusters.	166
4.46	The SSR values for the aggregated stepwise- and hybrid SAHN - h-median methods.	166
4.47	The hybrid partitioning method with 70 clusters created with the h-means method in phase one followed by the h-median method.	168
5.1	The proposed cluster-based method to solve the HpMP.	173
5.2	The routing costs for various clustering methods with 50 clusters using DataS1.	178
5.3	The results from the complete linkage clustering method where capacitated clusters are ignored using DataS3.	188
5.4	The work flow diagram of the suggested two-phase PROBUC method. . .	190
5.5	The re-assignment phase of the suggested two-phased PROBUC method.	192
5.6	Different regret functions for the single linkage method using DataS3. . .	197
5.7	Different regret functions for the single linkage method using DataS3 (cont. a).	198
5.8	Different regret functions for the single linkage method using DataS3 (cont. b).	199
5.9	Different regret functions for the single linkage method using DataS3 (cont. c).	200
5.10	Different regret functions for the complete linkage method using DataS3.	201
5.11	Different regret functions for the complete linkage method using DataS3 (cont. a).	202
5.12	Different regret functions for the complete linkage method using DataS3 (cont. b).	203
5.13	Different regret functions for the complete linkage method using DataS3 (cont. c).	204
5.14	The proposed cluster-based method to solve the plCLRP.	207
5.15	The calculation of the depot locations for the plCLRP.	209
5.16	The proposed cluster-based method to solve the cdCLRP.	216
5.17	The difference between using the closest two customers as endpoints versus lollipop routes to calculate the distance of a route.	218
5.18	An example of using the location phase method suggested for the cdCLRP to solve the sdCLRP.	223
5.19	The proposed cluster-based method to solve the sdCLRP.	224
6.1	The proposed solution to solve the PLRP.	235

List of Tables

2.1	The Lance-Williams parameters for different SAHN methods.	14
3.1	An overview of the distribution network problem definitions.	37
4.1	The three derived datasets from a food distribution company used to compare the clustering methods.	116
5.1	The routing costs for the HpMP and clustering measures when creating 50 clusters using DataS1.	179
5.2	The routing cost for the HpMP using three TSPLIB instances given by Gollowitzer et al. (2014).	180
5.3	The results for the HpMP using the USA13509 TSPLIB instance.	181
5.4	The different proximity measures in the two-phase PROBUC method using DataS3.	205
5.5	A list of instances tested using the TSP13509 dataset.	210
5.6	A list of instances tested using the WTSP17237 dataset.	211
5.7	The results for the plCLRP using the USA13509 TSPLIB instances.	214
5.8	The results for the plCLRP using the WTSP17237 instances.	214
5.9	The results for the cdCLRP using the USA13509 TSPLIB instances.	220
5.10	The results for the cdCLRP using the WTSP17237 instances.	220
5.11	A list of instances tested from Harks et al. (2013) for the sdCLRP.	227
5.12	The results for the sdCLRP using the CLRLib instances generated by Harks et al. (2013).	228
6.1	A list of instances tested from Harks et al. (2013) adapted for the PLRP.	237
6.2	The results for the PLRP for instances given by Prodhon (2009a).	238
6.3	The results for the PLRP using the CLRLib instances generated by Harks et al. (2013).	238
6.4	The results for the PLRP using XL size instances generated by Harks et al. (2013), excluding the branch-and-bound method in the MCA.	239
6.5	A comparison of the results for the PLRP, between including and excluding the branch-and-bound method in the MCA.	239
A.1	The results of the SSR measure for the various clustering methods using DataS1.	260

A.2	The results of the SSR measure for the various clustering methods using DataS1 (continue).	261
A.3	The results of the MSR measure for the various clustering methods using DataS1.	262
A.4	The results of the MSR measure for the various clustering methods using DataS1 (continue).	263
A.5	The results of the PD measure for the various clustering methods using DataS1.	264
A.6	The results of the PD measure for the various clustering methods using DataS1 (continue).	265
A.7	The results of the ADM measure for the various clustering methods using DataS1.	266
A.8	The results of the ADM measure for the various clustering methods using DataS1 (continue).	267
A.9	The results of the AWCD measure for the various clustering methods using DataS1.	268
A.10	The results of the AWCD measure for the various clustering methods using DataS1 (continue).	269
A.11	The CPU times for the various clustering methods using DataS1.	270
A.12	The CPU times for the various clustering methods using DataS1 (continue).	271
A.13	The results of the SSR measure for the various clustering methods using DataS2.	272
A.14	The results of the SSR measure for the various clustering methods using DataS2 (continue).	273
A.15	The results of the MSR measure for the various clustering methods using DataS2.	274
A.16	The results of the MSR measure for the various clustering methods using DataS2 (continue).	275
A.17	The results of the PD measure for the various clustering methods using DataS2.	276
A.18	The results of the PD measure for the various clustering methods using DataS2 (continue).	277
A.19	The results of the ADM measure for the various clustering methods using DataS2.	278
A.20	The results of the ADM measure for the various clustering methods using DataS2 (continue).	279
A.21	The results of the AWCD measure for the various clustering methods using DataS2.	280
A.22	The results of the AWCD measure for the various clustering methods using DataS2 (continue).	281
A.23	The CPU times for the various clustering methods using DataS2.	282

A.24 The CPU times for the various clustering methods using DataS2 (continue).	283
A.25 The results of the SSR measure for the various clustering methods using DataS3.	284
A.26 The results of the SSR measure for the various clustering methods using DataS3 (continue).	285
A.27 The results of the MSR measure for the various clustering methods using DataS3.	286
A.28 The results of the MSR measure for the various clustering methods using DataS3 (continue).	287
A.29 The results of the PD measure for the various clustering methods using DataS3.	288
A.30 The results of the PD measure for the various clustering methods using DataS3 (continue).	289
A.31 The results of the ADM measure for the various clustering methods using DataS3.	290
A.32 The results of the ADM measure for the various clustering methods using DataS3 (continue).	291
A.33 The results of the AWCD measure for the various clustering methods using DataS3.	292
A.34 The results of the AWCD measure for the various clustering methods using DataS3 (continue).	293
A.35 The CPU times for the various clustering methods using DataS3.	294
A.36 The CPU times for the various clustering methods using DataS3 (continue).	295
A.37 The results of the SSR and MSR measures for various clustering methods using USA13509.	296
A.38 The results of the PD and ADM measures for various clustering methods using USA13509.	297
A.39 The results of the AWCD measure and CPU times for various clustering methods using USA13509.	298
A.40 The results for the three variants of the hybrid density method using DataS3.	299
A.41 The results for the hybrid single linkage Ward's and complete linkage Ward's methods using DataS3.	300
A.42 The results of the SSR and MSR measures for the hybrid single linkage - partitioning clustering methods using DataS3.	301
A.43 The results of the PD and ADM measures for the hybrid single linkage - partitioning clustering methods using DataS3.	302
A.44 The results of the AWCD measure and CPU times for the hybrid single linkage - partitioning clustering methods using DataS3.	303
A.45 The results of the SSR and MSR measures for the hybrid complete linkage - partitioning clustering methods using DataS3.	304

A.46	The results of the PD and ADM measures for the hybrid complete linkage - partitioning clustering methods using DataS3.	305
A.47	The results of the AWCD measure and CPU times for the hybrid complete linkage - partitioning clustering methods using DataS3.	306
A.48	The results for the stepwise aggregated partitioning methods using DataS3.	307
A.49	The results of the SSR and MSR measures for the hybrid partitioning methods using DataS3.	308
A.50	The results of the PD and ADM measures for the hybrid partitioning methods using DataS3.	309
A.51	The results of the AWCD measure and CPU times for the hybrid partitioning methods using DataS3.	310

List of Acronyms

Acronym	Meaning	Page
ADM	Average distance measure	p 33
AGS	Active guided search	p 103
ACO	Ant colony optimisation	p 53
ATL	Alternating transportation-location heuristic	p 59
BKS	Best-known solution	p 210
CCP	Capacitated clustering problem	p 41
CCCP	Capacitated centred clustering problem	p 56
CFLP	Capacitated facility location problem	p 43
CCLP	Capacitated concentrator location problem	p 40
cdCLRP	Concentrator discrete capacitated location–routing problem	p 215
CLRGTS	Cooperative Lagrangian relaxation - granular tabu search	p 104
CLRP	Capacitated location–routing problem	p 91
CLS	Composite local search	p 102
CpMP	Capacitated p–median problem	p 40
CVRP	Capacitated vehicle routing problem	p 63
CWA	Clarke and Wright algorithm	p 67
DVRP	Distance vehicle routing problem	p 65
ECWA	Extended Clarke and Wright algorithm	p 95
ELS	Evolutionary Local Search	p 110
FDR	Furthest distance rule	p 59
FIFO	First in first out	p 73
FLP	Facility location problem	p 38

GA	Genetic algorithm	p 61
GAP	Generalised assignment problem	p 69
GG	Gabriel graph	p 26
GIS	Geographic information system	p 1
GLS	Guided local search	p 102
GRASP	Greedy randomised adaptive search procedure	p 96
HES	Hybrid extended savings method	p 102
HpMP	Hamiltonian p-median problem	p 82
ILP	Integer linear programming	p 40
LH-ACS	Lagrangian heuristic - ant colony system hybrid method	p 54
LTL	Less-than-truckload shipments	p 81
LP	Linear programming	p 46
LR	Lagrangian relaxation	p 50
LRP	Location-routing problem	p 81
MACS	Multiple ant colony system	p 53
MA PM	Memetic algorithm with population management	p 109
MCA	Modified Christofides algorithm	p 174
MDVRP	Multi-depot vehicle routing problem	p 78
MILP	Mixed integer linear programming	p 46
MNV	Mutual neighbourhood value	p 27
MSR	Mean of Squared Residuals	p 33
MST	Minimum spanning tree	p 25
MSWP	Multi-source Weber problem	p 56
mTSP	Multiple travelling salesman problem	p 63
NN	Nearest neighbours	p 15
PAM	Partitioning around medoids method	p 20
PD	Partition Diameter	p 33
plCLRP	planar CLRP	p 182

plLRP	planar LRP (uncapacitated)	p 86
PLRP	Periodic location–routing problem	p 106
POPMUSIC	Partial optimization metaheuristic under special intensification conditions	p 104
PROBUC	Proportional regret ordering based unconstrained to constrained	p 189
pTSP	Periodic travelling salesman problem	p 77
PVN	Penalty variable neighbourhood	p 103
PVRP	Periodic vehicle routing problem	p 77
RCL	Restricted candidate list	p 97
RECWA	Randomised extended Clarke and Wright algorithm	p 97
RNG	Relative neighbourhood graph	p 26
RNN	Reciprocal nearest neighbours	p 15
RVNS	Reduced variable neighbourhood search	p 62
SAHN	Sequential agglomerative hierarchical non-overlapping clustering	p 8
sdCLRP	Standard discrete capacitated location–routing problem	p 221
SNC	Subtour number constraints	p 83
SOM	Self organised map	p 89
SSR	Sum of squared residuals	p 32
TL	Truckload shipments	p 81
TP	Transportation problem	p 58
TSP	Travelling salesman problem	p 63
VLSN	Very large scale neighbourhood search	p 112
VNS	Variable neighbourhood search	p 61
VRP	Vehicle routing problem	p 63
VRPTW	Vehicle routing problem with time windows	p 65
UFLP	Uncapacitated facility location problem	p 38

Chapter 1

Introduction

The design of a distribution network for a big company has never been a task for the faint at heart. Many companies spend hours searching for the optimal placement of depots in their distribution network in the light of their individual service delivery footprint. According to Langevin and Riopel (2005) the location of depots or facilities is one of the most difficult network decisions to make because of the huge costs involved.

With the rapid growth of available data and technological advances, GIS (geographic information system) applications are becoming commonly available. The ability to create visualisations and maps from GIS data is becoming more accessible to everyone*. Because of this, GIS data is also becoming more useful and therefore more reliable. With this comes a motivation for companies to utilise the GIS data in their day-to-day operations while solving distribution problems like the vehicle routing problem.

Many goods distribution companies have customer footprints of over tens of thousands customers. For example, Klose (1990) refers to actual distribution network case studies with several thousand customers and Alvim and Taillard (2013) refers to a case study with 32 000 customers. The size of GIS datasets that companies have to deal with are therefore becoming larger with time.

1.1 Background and motivation

The capacitated location-routing problem (CLRP) is a well-known twofold optimisation problem. It addresses the placement of depots while at the same time constructs vehicle routes to ensure all customers' demands are met. The main objective is to minimise both the depot and vehicle costs. While the depots and vehicles can have capacity limitations, the customers can have varying demands that need to be met.

*An example is the free GoogleAPI tool that can be used to display geographical points on GoogleMaps used in this thesis, described in Appendix B.

Being derived from both the facility location problem (FLP) and vehicle routing problem (VRP), the CLRP is known to be NP-hard (Min, 1996; Lopes, 2011) and cannot be solved optimally within polynomial time complexity.

Numerous heuristic and metaheuristic solution approaches for the CLRP have been suggested. Some are listed in review papers like Nagy and Salhi (2007) and Drexler et al. (2014). Up to 2013, the solution approaches have mostly been tested on small instances, like those given by Prins (2004) where the biggest academic dataset with benchmarks had 200 customers and 10 potential depots. Unfortunately, these approaches often become impractical and too time consuming to solve bigger case studies.

Recently the focus has turned to solving location-routing problems with bigger datasets. Harks et al. (2013) proposed an approximate algorithm[†] to solve the standard discrete CLRP (sdCLRP). Their largest instance is a randomly generated dataset with 10 000 customers and 1 000 potential depots. Alvim and Taillard (2013) proposed a solution approach to solve a variant of the CLRP, called the concentrator discrete CLRP (cdCLRP), where all customers locations are considered potential depots. This case study does not cater for depots capacity constraints and allow an unlimited number of vehicles per depot. Their largest instance has over 1.9 million customers and is derived from the World TSP instance, (Cook, 2013).

From papers like Toth and Vigo (2003) and Lam et al. (2009) it is clear that customers within close proximity of each other should be grouped on the same routes and assigned to the same depots in order to create good CLRP solutions. Identifying customers within close proximity can easily be done with the help of clustering methods. Some clustering methods can quite effectively create clusters for large instances. It is therefore worth investigating how these clustering methods can be used to solve the CLRP for big case studies as well.

Min (1996), Barreto et al. (2007) and Lam et al. (2009) have investigated the use of clustering methods in the CLRP in the past. These studies were limited to hierarchical clustering methods. The solution approaches were also not tested on big instances.

1.2 Problem statement

The purpose of this thesis is to evaluate and compare the results from different clustering methods for large case studies with more than ten thousand data points. Based on the above investigation, cluster-based approaches are proposed to solve four variants of the CLRP and the PLRP (periodic location-routing problem). These

[†]**Approximate algorithms** find solutions guaranteed to be within a bounded optimality gap in polynomial time.

solution approaches are then evaluated to determine if they are useful to solve bigger case studies.

1.2.1 Limitations and scope

The study is limited to the following five clustering methods:

- Sequential agglomerative hierarchical non-overlapping (SAHN) clustering methods;
- Iterative partitioning methods;
- Graph-based methods;
- Nearest neighbour methods; and
- Density-based methods.

Combinations of the above mentioned methods are also tested as hybrid clustering methods. The methods are described in Chapter 2.

The following variants of the CLRP are studied:

- HpMP (Hamiltonian p-Median problem);
- plCLRP (planar capacitated location-routing problem);
- cdCLRP (concentrator discrete capacitated location-routing problem); and
- sdCLRP (standard discrete capacitated location-routing problem).

The PLRP (periodic location-routing problem) is also studied. An overview of the various distribution network problems and their characteristics are given in Table 3.1.

A number of case studies are considered with instances containing between 1 000 and 20 000 customer points and up to 1 000 potential depots in the case of the discrete CLRP. This is in line with realistic case study sizes and big enough to give an indication of the effectiveness of the solution approaches while at the same time considering software limitations such as variable declarations.

The study is limited to single-source problems where all customer demands are to be met in full, from a single depot. Each customer is therefore assigned to one and only one depot in all cases. In the case of the PLRP, customers are assigned to a single depot across the complete time horizon.

1.2.2 Thesis objectives

The objectives are as follows:

- Investigate, code and compare various clustering methods with regards to CPU times and quality of clustering. This includes:
 - Investigate methods to create capacitated clusters using clustering methods.
 - Investigate the impact of starting solutions on the various iterative partitioning methods.
 - Investigate the combining of the most effective clustering methods investigated above into hybrid clustering methods.
- Propose solution approaches using clustering methods to solve variants of the CLRP. This includes:
 - Propose a method to create capacitated clusters from clustering methods.
 - Propose a simple and quick heuristic to create routes once the customers have been clustered.
- Adapt the newly proposed solution approaches for the discrete CLRP to also solve the PLRP.

1.3 Methodology

The following methodology with regards to the data preparation and approach have been used in this thesis:

1.3.1 Data preparation and experiments

All clustering methods and suggested solution approaches in this thesis were coded and tested by the author using Visual Basic for MS Excel 2013. Excel was chosen because it is an application that leans itself towards the storing and analysis of big datasets. It can also be used as a data input tool and the code can easily be accessed and modified.

The experiments were carried out on an Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz computer with 8GB RAM and using a Windows 7 operating system. Instead of using standard functions or software packages, each of the clustering methods were coded so that different variations and modifications in the algorithms could also be tested.

Three types of datasets are used to test the proposed cluster-based methods suggested in this thesis. These include:

- a dataset of 7 681 geographical data points from a South African food distribution company. The dataset represents the delivery locations of customer

points in the distribution network of the company. The customer locations are data points used in the clustering methods with decimal Euclidean coordinates (x, y) used in GIS applications to determine positioning. This data was used in Chapter 4 to test the relevant clustering methods[‡];

- academic instances found in the literature for the HpMP, sdCLRP, cdCLRP and PLRP (Gollowitzer et al., 2014, Alvim and Taillard, 2013, Harks et al., 2013 and Prodhon, 2009a); and
- a large-scale TSP by Reinelt (2015) with 13 509 customers from the USA.

Not all of the problems have benchmarked datasets and some only have small instances. For these problems, the instances were adapted to test the proposed solutions for big case studies.

1.3.2 Approach

A couple of challenges have to be addressed in order to use clustering methods in solving the CLRP, these include:

- The methods have to be modified to cater for capacity constraints and variations in demand size.
- While the number of depots and vehicles to use are unknown in the CLRP, they are needed as inputs for the clustering methods.
- Some iterative partitioning clustering methods rely heavily on good starting positions to generate good results, which means that another method must first be identified to generate good starting solutions before these methods will become useful.
- Clustering methods determine their own cluster centres and do not inherently keep the centres of clusters at fixed potential depot locations. For the discrete CLRP, these cluster centres have to be moved back to the depot locations afterwards.

Based on these challenges, the research was divided into six steps. These are as follows:

1.3.2.1 An analysis of clustering methods

The effectiveness of the five different clustering methods are analysed and compared in Chapter 4. Five criteria measures are used to determine the quality of a clustering

[‡]This is a privately owned dataset from a food distribution company. Approval has been given to use the data in this thesis. The data is not made publicly available but visualisations can be seen in Chapter 4.

solution. The SSR (sum of squared residuals) measure is used the most in comparisons because it is the closest to the objective function value of the facility location phase of the CLRP. A critical look is given at the impact of starting solutions on the different iterative partitioning methods.

Some of the SAHN and iterative partitioning methods are then combined in hybrid clustering methods. Here the data is first aggregated with one method before being clustered with a second method.

1.3.2.2 Solution approach for the HpMP

Section 5.1 deal with the creation of Hamiltonian cycles using clustering methods. A heuristic algorithm called the Modified Christofides Algorithm (MCA), is introduced to create the route cycles. This method is based on the existing Christofides algorithm. A modified problem formulation for the HpMP is also proposed.

1.3.2.3 Solution approach for the plCLRP

A new problem called the plCLRP is introduced in Section 5.2. Here the multi-depot planar LRP (plLRP) is adapted to a fixed charge problem and depot capacity constraints are added. Based on the current plLRP problem formulation, a new capacitated fixed charge problem formulation for the plCLRP is introduced.

A new algorithm to create capacitated clusters from the clustering methods is proposed. The method is called the two-phased PROBUC (proportional regret ordering based unconstrained to constrained) method. This is described in Section 5.2.

1.3.2.4 Solution approach for the cdCLRP

The solution approach used in the plCLRP is adapted for the concentrator case in the cdCLRP. Different continuous to discrete methods are discussed. The concept of lollipop routes to approximate route costs is discussed. With the use of lollipop routes the depot locations can be determined without changing the routes. If any route has a round trip cost larger than the cost of opening a depot, the number of depots is increased. Once a decision is made on the number of depots to use, the final distribution costs can be calculated.

1.3.2.5 Solution approach for the sdCLRP

The solution approach for the cdCLRP will be ineffective for the sdCLRP if the depot clusters are not close to potential depots. This solution approach also do not cater for big variations in depot supplies and costs. An alternative heuristic is

proposed for the location phase of the solution approach to solve the sdCLRP.

1.3.2.6 Solution approach for the PLRP

A heuristic to cater for the periodic component of the PLRP is proposed. This heuristic makes use of the solution approaches for the discrete CLRP to solve the PLRP. A modified version of the problem formulation for the PLRP is also introduced to enforce the single-source constraint across multiple visits and limit the maximum number of vehicles. These topics are discussed in Chapter 6.

1.4 Thesis contribution

The contributions of this study are as follows:

- Various clustering methods and hybrid combinations are coded, tested and compared using the same instances. The comparisons are based on five measuring criteria, CPU times and illustrations of the clustering solutions in maps.
- A new problem called the plCLRP is introduced and a new problem formulation based on the plLRP and CVRP is introduced.
- Modified problem formulations are provided for the HpMP and PLRP.
- Cluster-based solution approaches to solve the HpMP, plCLRP, cdCLRP, sdCLRP and PLRP are suggested.
- New large scale instances are introduced for the HpMP, plCLRP and PLRP and benchmarks are provided.
- Results are compared to benchmarked instances for the cdCLRP and sdCLRP and a couple of the smaller instances for the HpMP and PLRP.

1.5 Thesis overview

The remainder of this research can be summarised as follows. Chapter 2 contains a literature review of clustering methods while Chapter 3 discusses the relevant distribution network problems. In Chapter 4 the effectiveness of the different clustering methods are analysed and compared. Based on these findings Chapter 5 proposes new cluster-based solution approaches to solve four variants of the CLRP. Chapter 6 proposes a cluster-based solution approach to solve the PLRP. Chapter 7 provides conclusions and identifies future areas of research.

Chapter 2

Literature review - clustering methods

The goal of clustering methods is to group items together to minimise the total dissimilarity between the points in a cluster. This goes hand-in-hand with maximising the dissimilarity between points from different clusters, (Everitt et al., 2011). The idea of clustering items together to simplify a problem is very intuitive and for most people, it is the first thought that comes to mind when looking at a map of customers to be serviced.

Multiple clustering methods have been developed over the years; the most well-known are discussed in books like Hartigan (1975), Jain and Dubes (1998) and Everitt et al. (2011). Five different types of clustering methods for geographical data are discussed; these include hierarchical, iterative partitioning, graph-based, nearest neighbour and density-based clustering methods.

2.1 SAHN clustering methods

Hierarchical clustering methods use a nested sequence of partitions. There are two types of hierarchical clustering methods: agglomerative and divisive. The *agglomerative* hierarchical methods start with each customer or point placed in its own cluster. The clusters are then merged together based on inter-cluster distances, (Jain and Dubes, 1998). In contrast, *divisive* methods start with all points in one cluster and incrementally divide the points into an increasing number of clusters.

According to Everitt et al. (2011, p 84), divisive methods are computationally more demanding than agglomerative methods. If used with an exhaustive search, the time complexity of the methods are $O(2^n)$, where n is the number of points to be clustered. This is higher than the maximum estimated time complexity of $O(n^3)$, for

2.1 SAHN clustering methods

agglomerative methods, (Müllner, 2013). For these reasons the focus in this work will be on agglomerative hierarchical methods. These methods are referred to as SAHN (sequential agglomerative hierarchical non-overlapping) methods, (Jain and Dubes, 1998, p 79).

SAHN clustering is often depicted with dendrograms, (Jain and Dubes, 1998, p 65). A dendrogram is a tree diagram that graphically represents the merging of the clusters. The horizontal axis represents the points and the vertical axis represents the critical distance measure. Horizontal lines are drawn to represent the merging of two clusters or points. An example of a dendrogram is given in Figure 2.1.

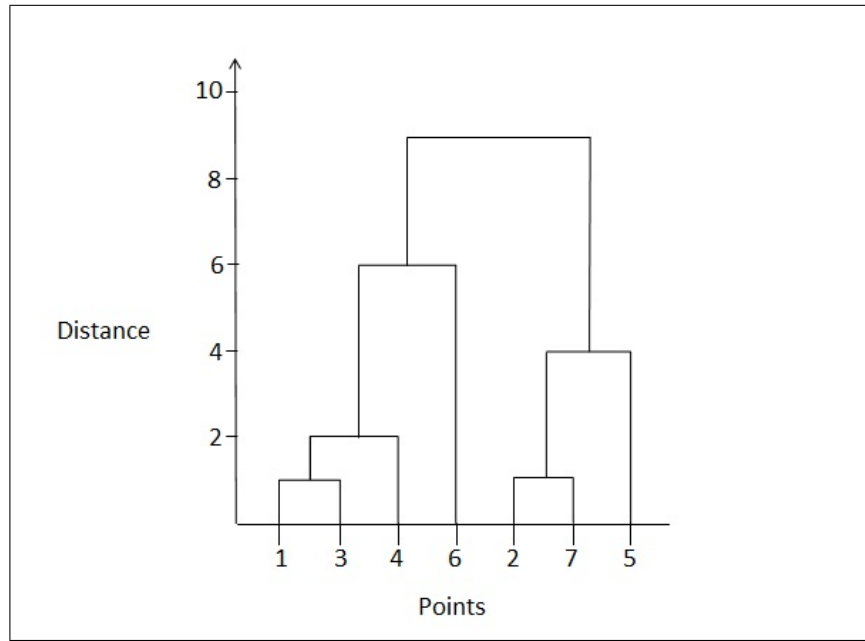


Figure 2.1: A dendrogram representation of SAHN clustering.

In this figure, the dendrogram illustrates that SAHN methods can continue clustering until all customers are in one cluster. This can be prevented by the use of stopping rules as discussed in Section 2.1.10. A stopping rule can be represented by one or more vertical lines through the dendrogram and is called a "cut", (Everitt et al., 2011, p 95).

SAHN methods iteratively calculate the inter-cluster distances and then merge the next shortest distance. Different inter-cluster distances can be used for example: shortest distance, furthest distance, average distance, distance between centroids and the sum of intra-cluster variance, also known as Ward's method (Everitt et al., 2011). The different SAHN methods are now discussed in more detail.

2.1.1 The single linkage method

The single linkage method is also referred to as the "nearest-neighbour" or minimum spanning tree (MST) method, (Han and Kamber, 2006). The distance between two clusters is defined as the distance between their closest points to each other. The two clusters with the smallest distance between them are then merged. The process is repeated until all points belong to one cluster. A stopping rule can also be used to stop earlier, for example when a specified number of clusters are reached.

According to Lam (2008), this method tends to produce string-like clusters, known as chaining. The chain effect could be useful when creating routes, but it is not efficient for the location phase, (Lam et al., 2009; Barreto et al., 2007).

2.1.2 The complete linkage method

A second method called the complete linkage method, uses the furthest inter-cluster distance to determine which clusters to merge. Similar to the single linkage method, the distances between all points in the different clusters are determined. The maximum distance between any two points in two different clusters is identified as the furthest inter-cluster distance for the cluster pair. All furthest inter-cluster distances are ordered based on size and the cluster pair with the shortest furthest inter-cluster distance are merged. All furthest inter-cluster distances for the new merged cluster are calculated and the process is repeated to find the next two clusters to merge.

Han and Kamber (2006) call this method the furthest-neighbour clustering algorithm. Everitt et al. (2011) state that this method tends to create compact clusters with equal diameters, but does not consider different cluster structures.

2.1.3 The average linkage method

This method merges clusters based on the average distance between all inter-cluster pairs, (Everitt et al., 2011). Similar to the other SAHN methods, the distances are ordered and the cluster pair with the shortest average distance are merged. The process is repeated until a stopping rule is reached or all points are in one cluster.

The method is also referred to as the unweighted pair-group method using averages (UPGMA). It is called "unweighted" because all inter-cluster pair distances weigh the same when calculating the average distance. Weighting based on previous merges is not considered.

2.1.4 The weighted average-linkage method

The weighted average-linkage method is also called the weighted pair-group method using averages (WPGMA). The merging of the clusters is based on the shortest weighted average inter-cluster distance. It is called "weighted" because the distances used to calculate the average, are weighted based on the number of points in each of the two clusters from the previous cluster merge, (Everitt et al., 2011).

The weighted average distance can be described based on the illustration given in Figure 2.2. Consider two mutually exclusive clusters i and j , each with n_i and n_j number of points respectively. In the figure, cluster i and j have merged to form cluster k . The total number of points in cluster k is then $n_k = n_i + n_j$. Let the sets I, J and H be the set of points in clusters i, j and h respectively. Then $I \cup J = K$ is the set of points in cluster k .

The weighted average inter-cluster distance between an arbitrary cluster h and cluster k (Everitt et al., 2011) can now be calculated as

$$\overline{d_{hk}} = \frac{1}{n_i} \left(\sum_{x=1}^{n_i} \sum_{y=1}^{n_h} d(xy) \right) + \frac{1}{n_j} \left(\sum_{z=1}^{n_j} \sum_{y=1}^{n_h} d(zy) \right) \quad (2.1)$$

$$x \in I, y \in J, z \in H \text{ and } I \cup J = K, \quad (2.2)$$

where $d(xy)$ is the distance between points x and y and $\overline{d_{hk}}$ is the weighted average distance between clusters h and k . The weighting is therefore based on the number of points in clusters i and j and each inter-cluster pair does not contribute the same weight towards the average. Instead, this method ensures that the inter-cluster distances from both clusters i and j contributes equally to the new average.

If the clusters have an equal number of points, the average would be the same as for the average linkage method. If one of the clusters is smaller, the points in the smaller cluster will weigh more than in the big cluster in order to get the clusters to weigh the same. The rationale behind this is to prevent small clusters from being dominated by bigger ones.

2.1.5 The centroid linkage method

In the centroid linkage method, the average of the longitudes and latitudes of all points in a cluster is calculated. This is referred to as the centroid of each cluster, (Everitt et al., 2011). Clusters are then merged based on closest distance between the centroids of the clusters. The centroid linkage method calculates the distances between all centroids.

If point i is represented by the coordinate vector as a tuple rather than a column

vector in Euclidean space $\mathbf{x}_i = (x_i^1, x_i^2, \dots, x_i^l)$, for $\mathbf{x}_i \in \mathfrak{R}^l$, then the centroids are determined by the mean of the coordinates, (Everitt et al., 2011). The mean of the r_{th} coordinate for cluster k is represented by

$$\bar{x}_k^r = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i^r, \quad (2.3)$$

where x_i^r is the value of the r_{th} coordinate of point i assigned to cluster k and n_k is the number of points in cluster k . The centroid is the vector with all mean values as coordinates, $\bar{\mathbf{x}}_k = (\bar{x}_k^1, \bar{x}_k^2, \dots, \bar{x}_k^l)$.

The centroid method is also referred to as the unweighted pair-group method using the centroid approach (UPGMC). Similar to the average linkage method, it is called unweighted because each point in a cluster is equally weighted when determining the centroid.

2.1.6 The weighted centroid (median) linkage method

The weighted centroid linkage method is also called the weighted pair-group method using the centroid approach (WPGMC). The method also uses the distances between the centroids of the clusters to determine the inter-cluster distances, similar to the centroid linkage method. The difference between the centroid and weighted centroid linkage methods lies in how the centroid of cluster k is calculated, (Everitt et al., 2011). Similar to the weighted average method, this method also gives the contribution of the two clusters that just merged an equal amount when calculating the distances to the centroid of the new merged cluster. If the two merged clusters differ in size, points in the smaller cluster are given a bigger weight compared to the points in the bigger cluster to ensure an equal contribution per cluster.

This is called the median of the two merged clusters and the method is also referred to as the median SAHN method, (Everitt et al., 2011).

2.1.7 Ward's method

Ward (1963) suggested a method to minimise the incremental sum of square variance within clusters. The sum of squares, E_k , within a cluster k is determined by the sum of the squared distances between the set K containing all the points in cluster k and its mean. If the points are defined by more than one Euclidean space variable, the sum of square variance of all the variables needs to be added together to determine E_k .

When merging, the two clusters that will result in the minimum E_k value are merged

in every iteration. E_k can be calculated as

$$E_k = \sum_{i=1}^{n_k} \|x_i - \bar{x}_k\|_2^2 \quad i \in K \quad (2.4)$$

$$= \sum_{r=1}^l \sum_{i=1}^{n_k} (x_i^r - \bar{x}_k^r)^2 \quad i \in K, \quad (2.5)$$

where $\bar{x}_k^r$ is the mean of the r_{th} variable in cluster k , x_i^r is the value of the r_{th} variable for point i in cluster k , n_k is the number of points in cluster k and l is the total number of variables defined in the Euclidean space. The total sum of square variance for all the clusters is then

$$E = \sum_{k=1}^p E_k \quad (2.6)$$

where p is the total number of clusters. Everitt et al. (2011, p. 77) refers to this as the within-cluster sum of square errors, while Jain and Dubes (1998) call it the sum of intra-cluster variance.

According to Everitt et al. (2011), this method tends to create equal sized clusters with a spherical form, but it is sensitive to outliers. Barreto et al. (2007) and Lam et al. (2009) found this method to be the most effective in the routing phase of the LRP.

2.1.8 The Lance-Williams formula for SAHN methods

The Lance-Williams formula can be used as an alternative to calculate the new inter-cluster distances after every merge, based on the previous inter-cluster distances, (Lance and Williams, 1967). The advantage of the formula is that the inter-cluster distance calculations are hugely simplified. The authors define a general recurring distance formula to calculate the new inter-cluster distances between the clusters after every merge.

The method starts by calculating the distances between all customers and store the values in a matrix, called a distance matrix (Li et al., 2010). This is used at the start because all customers are in their own clusters. These distances can be calculated in different ways, as discussed in Section 2.8. After every merge the inter-cluster distances are updated based on the previous values. In Figure 2.2, the inter-cluster distance between cluster h and cluster k is represented by $d_{hk} = d_{h(ij)}$ and can be determined as

$$d_{hk} = \alpha_i d_{hi} + \alpha_j d_{hj} + \beta d_{ij} + \lambda |d_{hi} - d_{hj}|, \quad (2.7)$$

where the Lance-Williams parameters $\alpha_i, \alpha_j, \beta$ and λ differ per method.

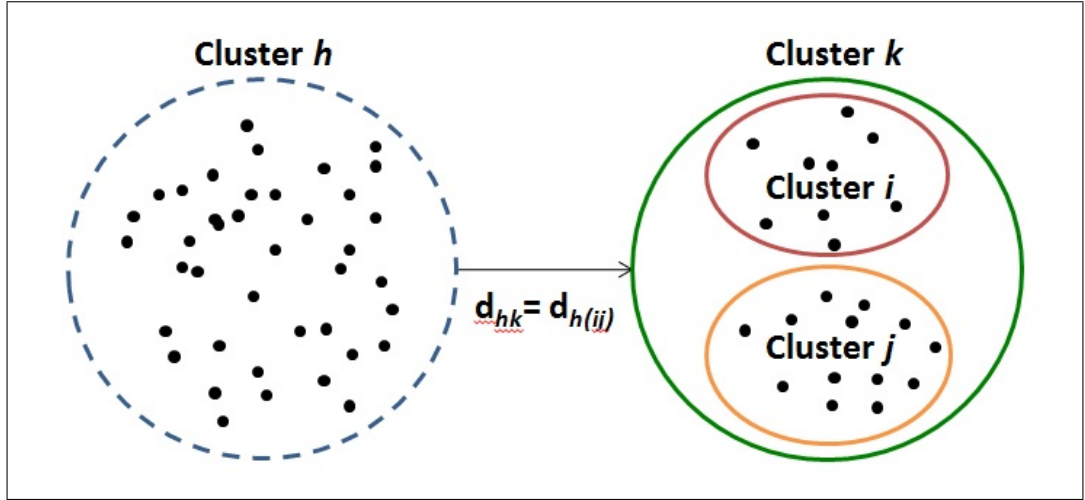


Figure 2.2: An illustration of the hierarchical clustering distance calculation used by the Lance-Williams parameters.

A summary of the Lance-Williams parameters for the different SAHN methods are given in Table 2.1.

Method	α_i	α_j	β	λ
Single linkage	$\frac{1}{2}$	$\frac{1}{2}$		$-\frac{1}{2}$
Complete linkage	$\frac{1}{2}$	$\frac{1}{2}$		$\frac{1}{2}$
Average linkage	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$		
Weighted average linkage	$\frac{1}{2}$	$\frac{1}{2}$		
Centroid linkage	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	$\frac{n_i n_j}{(n_i + n_j)^2}$	
Weighted centroid linkage	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	
Ward's method	$\frac{n_h + n_i}{n_h + n_i + n_j}$	$\frac{n_h + n_j}{n_h + n_i + n_j}$	$-\frac{n_h}{n_h + n_i + n_j}$	

Table 2.1: The Lance-Williams parameters for different SAHN methods.

2.1.9 CPU times of the SAHN methods

The time complexity of the SAHN methods gives an indication of the order of the CPU times. Both the single linkage and complete linkage methods only need the calculation of a distance matrix between all points once. This means that the time complexity for these two methods is $O(n^2)$, (Murtagh, 1983). When symmetrical distances are assumed, as we do in this thesis, the distance between points i and j is the same in both directions i.e. $d_{ij} = d_{ji}$. Even in symmetrical cases where only $n(n-1)/2 = (n^2 - n)/2$ calculations are needed to calculate the distance matrix, the order of the time complexity remains $O(n^2)$, (Murtagh, 1983).

The other SAHN methods have higher time complexities when the Lance-Williams formula are not used, because recalculations are needed after the merge of every cluster pair. The advantage on the other hand of not using the Lance-Williams formula is that the methods can work without a distance matrix which can save a lot of memory space. When the Lance-Williams formula are used to calculate the inter-cluster distances, the time complexity is $O(n^2 \log(n))$, Everitt et al. (2011, p 80), Hansen and Jaumard (1997).

Murtagh (1983) describes another method to lower the CPU time with the use of nearest neighbours (NN). If a point (or cluster) q is the nearest neighbour of point (cluster) p then $q = NN(p)$. The author also defines the concept of mutual or reciprocal nearest neighbours (RNN) as $q = NN(p)$ and $p = NN(q)$. So the two points (clusters) are RNN if both are the nearest neighbour of the other.

According to Murtagh (1983) any RNN found can immediately be merged in the SAHN methods with the exception of the centroid and median methods. The author claims that even though the order of the merging will depend on the order in which the check is performed, the resulting hierarchy will be unique and provide the same results.

2.1.10 Stopping rules in SAHN clustering

Since SAHN clustering methods can continue to merge clusters until all points are in one cluster, a rule is needed to determine when to stop clustering. This is referred to as a stopping rule by Lam et al. (2009). The simplest rule is to stop when the required number of clusters is reached, effectively cutting the dendrogram, illustrated in Figure 2.1, at a certain height.

Care must be taken when using this stopping rule as-is, because the clusters are unconstrained and can have completely different sizes. This is particularly true when dealing with outliers, the outliers can create single-point clusters while all the other points are grouped together in the one cluster. When using the SAHN methods to solve the CLRP, the number of clusters needed in the CLRP is not always known beforehand. If this is the case, the method will have to be repeated over a range of different numbers of clusters to identify the best solution.

Everitt et al. (2011) suggest using a distance measure as cut-off to determine the best number of clusters to use. The dendrogram is then cut at this distance and the number of clusters is determined. This is defined as the *Best Cut*. The same problem as above can occur with outliers forming their own clusters, which will in return result in expensive routes or extra fixed depot opening costs when used in the LRP. Another type of cut to overcome this problem is the *dynamic tree cut*, where

clusters are cut at different levels of the dendrogram.

Min (1987) and Barreto et al. (2007) explore the use of vehicle capacity as a stopping rule to ensure that clusters do not exceed capacity limits. Barreto et al. (2007) suggest excluding a cluster if it becomes fully capacitated, and continue with the merging of clusters that still have capacity available. Their conclusion is that this can result in two far-off sub-clusters being merged together, giving suboptimal routing solutions.

To prevent points too far from each other from being clustered together, Lam (2008) suggests using the ratio of inter-cluster distance variation and between-cluster distance variation as a stopping rule. This is called the pseudo F-statistic.

2.2 Iterative partitioning methods

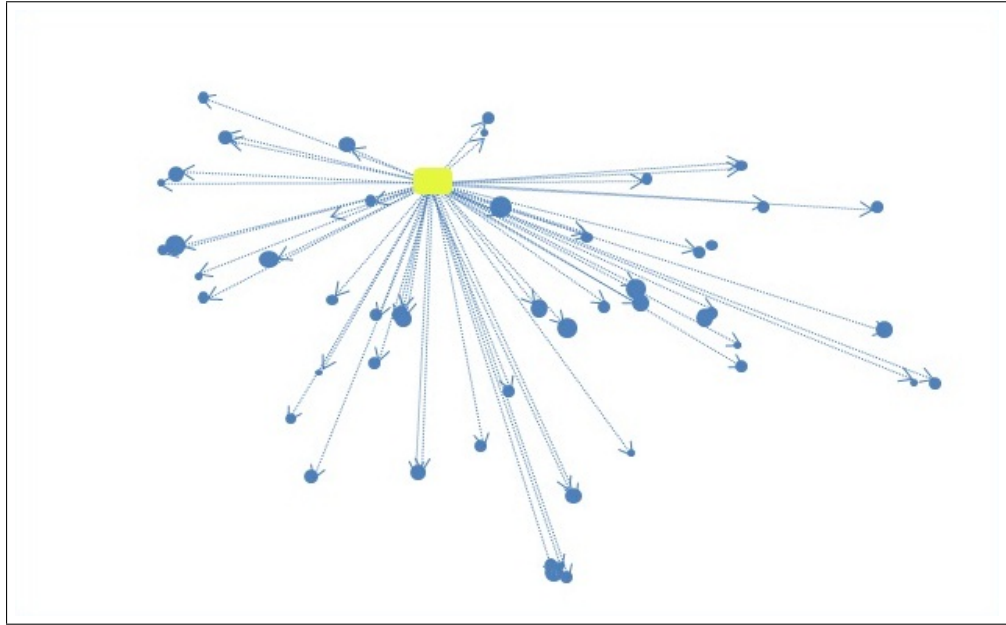
The iterative partitioning methods assume that all data points, also referred to as customers, need to be assigned to p mutually exclusive clusters. The methods start by selecting p customers as seed points. The seed points will serve as the centres of the clusters and all remaining customers are assigned to the closest seed points. This is referred to as a starting solution. The objective is to minimise the sum of all the distances between the customers and their assigned cluster centres.

Unlike SAHN clustering, iterative partitioning clustering methods therefore start off with the data points already in p different clusters. After a starting solution has been created, the methods iteratively cycle between re-assigning points to cluster centres and recalculating the cluster centres. An example of an iteration that illustrates the movement of the cluster centres can be seen in Figures 2.3(a) and 2.3(b). An iteration consists out of two steps:

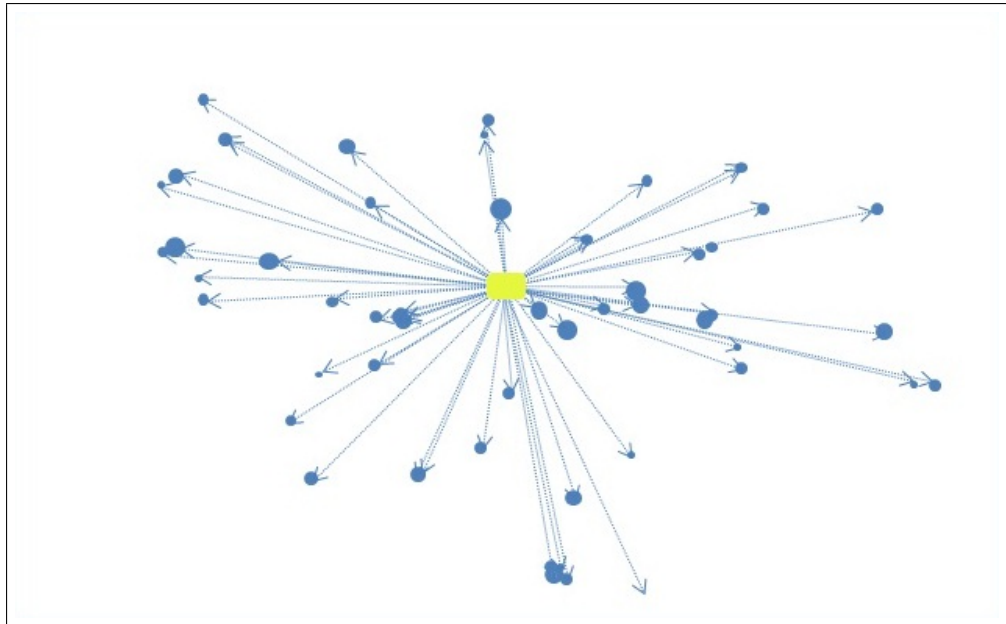
1. Assign customers to the closest cluster centre; and
2. Recalculate the cluster centres.

A popular criterion to use as the objective function during clustering is to minimise the sum of square residuals (SSR) or variances, of the distances between all the points and their cluster centres. Although minimising the SSR is only effective for spherical and equal sized clusters, (Everitt et al., 2011), it is still the most commonly used criterion for clustering methods.

When recalculating the cluster centres, three different calculations are often referred to; the means, medoid and median calculations. All three calculations methods will form the basis of the different types of iterative partitioning methods discussed below.



(a) Step 1: Customers are assigned to the closest cluster centres.



(b) Step 2: The cluster centres are recalculated based on the assigned customers.

Figure 2.3: An illustration of the k-means method.

One can further distinguish the methods between "k-" and "h-"variants. Both variants start off with all data points already assigned to p different clusters, but the distinction is based on the number of customer points re-assigned during every iteration. There is much confusion in the literature regarding which variant of the method is referred to under which name. In this thesis the naming convention suggested by Hansen and Mladenović (2001) is used. This is described as follows:

- The "k-" variant refers to the re-assignment of only one customer point to the closest centre at a time before recalculating the cluster centres. The point to be re-assigned is the one that will give the biggest distance saving.
- The "h-" variant refers to re-assignment of all customer points to their closest centres during every cycle, before recalculating the cluster centres.

On the basis of different centre calculations and variants, six different iterative partitioning methods are now described as follows:

2.2.1 The k-means method

In the k-means method, the centres of the clusters are iteratively recalculated as the means of the coordinates of the customer points assigned to the cluster. This is called the centroids of the clusters, (Geetha et al., 2009). In $\mathbf{x}_i \in \mathbb{R}^2$, the coordinates (x_j, y_j) of the j_{th} centroid are therefore calculated as follows:

$$x_j = \frac{\sum_{i=1}^{n_j} x_i}{n_j} \quad y_j = \frac{\sum_{i=1}^{n_j} y_i}{n_j}, \quad (2.8)$$

where n_j is the number of points in cluster j and $j = (1, 2, \dots, p)$ for p clusters in total and $\sum_{j=1}^p n_j = n$.

The distance savings for re-assigning every customer to the closest centre is calculated. In the k-variant, only the customer with the best distance saving is re-assigned, (Hansen and Mladenović, 2001). After every re-assignment the cluster centres of the involved clusters are recalculated. Iterations, consisting of assigning customers and recalculating cluster centres, are repeated until no re-assignments will result in a cost saving or a predefined number of iterations is reached. The method is more sensitive to outliers than other partitioning methods, (Han and Kamber, 2006).

According to Negreiros and Palhano (2006) the time complexity of the method is $O(n)$ per iteration, while Han and Kamber (2006) goes into more detail and place the total computation time at $O(npt)$, where t is the number of iterations. The k-means method is not very scalable to large datasets, because of the large number of iterations needed before the method converges, (Negreiros and Palhano, 2006).

To make the k-means easier to use on large datasets, a suggestion was made by Han and Kamber (2006) to first group points into "micro clusters", before performing the k-means. The customer points in the micro clusters are then treated as a single point in the k-means. This is also known as the aggregation of data. The single linkage

SAHN method is often used with a stopping rule based on a specified distance to create the micro clusters.

2.2.2 The h-means method

The above k-means method can be quite time-consuming because only one re-assignment is done per iteration. This makes the method unsuitable for large datasets. To overcome this problem, all customer points can be re-assigned before recalculating the cluster centres in every iteration. This is called the h-means method, (Hansen and Mladenović, 2001).

Care should be taken which variant of the means method is referred to in literature; the h-means variant is often meant when referring to the k-means method. The method has the same time complexity of $O(npt)$ as the k-means method, (Han and Kamber, 2006), but the number of iterations are far less than the k-means method.

Han and Kamber (2006) also claims that the method is not as effective in large datasets because it often finds a local minimum instead of the global minimum, especially if the number of clusters or the number of iterations are far less than the number of points. Negreiros and Palhano (2006) refer to the method as the *Forgy method*.

Hansen and Mladenović (2001) state that the k-means method cannot be improved by the h-means, because the k-means is more accurate. They suggest using the h-means followed by the k-means in large datasets. They referred to this as the **hk-means** method.

2.2.3 The j-means (k-medoid) method

The k-medoid method is very similar to the k-means, but is not as sensitive to the impact of outliers. Instead of calculating the means as the cluster centres, this method calculates the medoids. The medoid of a cluster is the most centrally located cluster point. To calculate the most central points, the absolute error criterion is used. The objective criterion is to minimise E , where E is the absolute error value for all clusters summed together. It is formulated and described by Han and Kamber (2006), as follows:

$$E = \sum_{j=1}^p \sum_{i=1}^{n_j} (\|x_i - o_j\|_l) \quad (2.9)$$

$$= \sum_{j=1}^p \sum_{i=1}^{n_j} \sum_{r=1}^l (|x_i^r - o_j^r|), \quad (2.10)$$

where x_i^r is the r_{th} variable of the point x_i in cluster j with n_j points. The current medoid of cluster j is o_j (with the r_{th} coordinate o_j^r) and l is the total number of dimensions in the Euclidean space.

As in the k-means method, an initial set of medoids is chosen as seed points for a starting solution. Points are re-assigned one at a time based on the best distance saving. To calculate the medoid, the method iteratively replaces the medoid with another random point in the cluster if E becomes less, (Han and Kamber, 2006). Every time E changes, the process is repeated. Hansen and Mladenović (2001) call this method the **j-means method**. They suggest calculating E using every point in the cluster as the medoid and replacing the medoid with the point that returned the minimum E .

Although Hansen and Mladenović (2001) state that this method is quite effective for large datasets, it is clear that the amount of calculations needed to determine the best swap between all possible pairs of chosen - not chosen points can be quite time consuming. This is especially true if a distance matrix of size $n \times n$ becomes too big to be stored, in which case distances between all points and the centres need to be recalculated in every iteration to determine the best possible swap. Negreiros and Palhano (2006) also deem the method unsuitable for larger datasets because of the large number of calculations needed.

2.2.4 The PAM (h-medoid) method

A variation of the j-means method is referred to as the **partitioning around medoids (PAM) method**. Here the best possible medoid for every cluster is calculated in every iteration. According to Han and Kamber (2006), the PAM method uses more computation time than the k-means, with a complexity of $O(p(n-p)^2)$ per iteration. If the number of iterations is t , then the total time complexity is $O(pt(n-p)^2)$. Although it works very effectively in small datasets, it is not scalable to large datasets.

2.2.5 The k-median method

The k-median method described by Han and Kamber (2006), uses the median of each dimension or attribute to determine the cluster centre. The median does not necessarily have to be a point in the cluster but is the combination of the middle values of the different dimensions. This method is even less sensitive to outliers than the k-medoids method. This is the type of clustering needed for the LRP, since outliers can be grouped into single trips. The effect of outliers on the current depot location is discussed in more detail in Section 3.3 and Figure 3.7.

2.2.6 The h-median method

Similar to the k-median method, the h-median method also calculates the median but in this case all points are re-assigned at the same time before new medians are calculated.

2.2.7 Starting solutions for iterative partitioning methods

It is clear that different starting solutions may result in different solutions. For this reason it is normal to repeat the partitioning method using different starting solutions in search of a good solution. According to Negreiros and Palhano (2006), Everitt et al. (2011) and Hartigan and Wong (1979), the quality of the results of the partitioning methods depend greatly on the selected starting solution. A couple of suggestions for initial starting solutions made by Jain and Dubes (1998) are listed below:

- Random starting points – p random points are selected as starting points for the solution, where p is the number of clusters required.
- The first p points – this method is biased towards the order in which data is provided. This method cannot be used more than once, since the same starting solution will always give the same final solution when using k-means.
- Choose points well separated from each other. The centroid of the data is taken as the first point, other points selected must be at least a specified distance from the points already selected.
- Use another clustering method to generate a starting solution, for example SAHN clustering, the k_{th} -nearest neighbours or density-based clustering. Grid based clustering can also be used to divide the data into p equal sized blocks and selecting one random point per block.
- Hartigan and Wong (1979) suggested the **stepwise function**. The data points are ordered based on their distances to the centroid of the data from smallest to largest. Next the points are divided into intervals and every $\left(1 + (j - 1) \times \frac{n}{p}\right)_{th}$ point is selected as the next cluster centre j . This method is used to generate starting solutions for the iterative partitioning methods in Chapter 4.

In Chapter 4 we also investigate the use of aggregation (the creation of micro clusters) in the stepwise function. Here the number of points used to generate starting solutions for the partitioning method are made less, by aggregating points close to each other. This ensures that the starting solutions are more diverse and the number of starting solutions is less.

The different suggestions are investigated further in Section 4.3.2.7.

2.2.8 Methods to prevent the number of clusters from declining

The iterative partitioning methods start with p clusters, but as customers are re-assigned to the closest cluster during the iterations, it is possible for one or more clusters to have no customers assigned to them and become empty. This will cause the number of clusters in the solution to decline. It is common practice to add precautionary measures to prevent this from happening. Two methods to prevent this are as follows:

1. Restrict the minimum number of points assigned to any cluster.
In an effort to preserve the cluster, the last couple of points are not allowed to be assigned to other clusters. This method is called constrained partitioning clustering, (Hartigan and Wong, 1979).
2. Introduce new cluster centres.
New cluster centres are added to the solution at the customer points the furthest distance away from any of the current clusters. Hansen and Mladenović (2001) refer to this as the h-means+ method. This method is similar to the **FDR (furthest distance rule)** used by Zainuddin and Salhi (2007) to create starting solutions to solve the MSWP (Multi-source Weber problem).

2.2.9 The CLUSTER partitioning methods

Jain and Dubes (1998) suggested a variation of the h-means method, called the CLUSTER h-means method. This method also uses the h-means method, but instead of finding different starting solutions by selecting cluster centres, this method searches for good solutions by changing the number of clusters to use. It cycles through a number of h-means repetitions, each time changing the number of clusters by dividing and merging clusters in the previous solution. The results of the different solutions found for p number of clusters are compared based on an objective function like Eq. (2.10) and the best solution is selected.

The method consists of the following steps:

1. Start with two clusters. Calculate the centroid of the data as the first cluster centre. The furthest point from the centroid is the centre of the second cluster. Use a partitioning method like the h-means method to create clusters.
2. For $3 - p$ clusters: Add an extra cluster to the cluster centres found in the previous solution at the furthest point from these centres, where the initial

solution was created in Step 1. Use the cluster centres to create a starting solution as input for the partitioning method using the new number of clusters. Calculate the objective function for each solution and note the number of clusters associated with these.

3. For each of the $3 - p$ cluster solutions found in Step 2, merge the two clusters with the smallest between-cluster distance and calculate a new cluster centre. Use the h-means method to cluster again. Calculate the objective function values as before.
4. Compare the objective functions of the solutions calculated in Steps 2 and 3 with the same number of clusters. Select the best solution for each number of clusters. Use the cluster centres of these solutions as seed points and repeat Step 2 and 3.

The process is continued until the difference in the objective function found between consecutive solutions with p clusters becomes insignificant or the objective function value starts to deteriorate compared to previous best found values. Return the best solution for p clusters. Although Jain and Dubes (1998) only refer to the h-means method, the CLUSTER partitioning method can be used to generate starting solutions for any of the iterative partitioning variants.

2.2.10 The AS 136 method

Hartigan and Wong (1979) created a partitioning algorithm, named the AS 136 k-means algorithm. Similar to the k-means method, this method calculates the means of the coordinates of the assigned points as the cluster centres and the centres are also recalculated after each iteration. This algorithm consists of two customer assignment phases and cycles between the two phases until no more improvements can be found. A *live set* is introduced to remember the clusters that have recently changed.

An initial solution is created by selecting p random seedpoints as the cluster centres. For each point $\mathbf{x}_i \in \mathbb{R}^d$, store the cluster number of the closest cluster centre in variable $I_1(x_i)$, and the cluster number of the second closest centre in variable $I_2(x_i)$, respectively. Assign each point to $I_1(x_i)$, the closest centre, and calculate the new cluster centres using the means. All clusters are initially added to the *live set*. The method cycles through two phases, starting with the optimal-transfer phase, it continues until no more improvements can be made. The two phases of the algorithm are as follows:

Optimal-transfer phase:

For each point x_i , calculate $R_1(x_i)$ using

$$R_1(x_i) = \frac{n_j d(x_i, \bar{x}_j)^2}{(n_j + 1)} \quad (2.11)$$

where point x_i is currently assigned to cluster j , n_j is the number of points in cluster j , $\bar{x}_j$ is the centre of cluster j and $d(x_i, \bar{x}_j)$ is the distance between point x_i and $\bar{x}_j$. Assign $I_1(x_i) = \text{cluster } j$.

Calculate $R_j(x_i)$ in a similar fashion for a subset of the other clusters, using Eq. (2.11). If cluster j is part of the *live set*, calculate $R_j(x_i)$ for all clusters. If cluster j is not part of the *live set*, only include the clusters in the *live set*. Assign the minimum value of $R_j(x_i)$ to $R_2(x_i)$ and note the associated cluster as cluster k .

If $R_2(x_i) < R_1(x_i)$, then re-assign point x_i to cluster k and change $I_1(x_i) = k$ and $I_2(x_i) = j$. Assign both clusters k and j to the *live set*. Recalculate the new cluster means for them, because there was a change in the points assigned to them.

If $R_2(x_i) \geq R_1(x_i)$, then $I_2(x_i) = k$.

Continue until $R_1(x_i)$ and $R_2(x_i)$ have been calculated for all points. Clusters that have been changed in the last n optimal-phase steps are part of the *live set*, where n is the total number of points. Recalculate the cluster centres. If the *live set* is empty, continue with the quick-transfer phase, otherwise repeat the optimal-transfer phase.

Quick-transfer phase:

For each point x_i , calculate $R_1(x_i)$ and $R_2(x_i)$ using the clusters from $I_1(x_i)$ and $I_2(x_i)$ respectively with the new calculated cluster centres. If $R_2(x_i) < R_1(x_i)$ then swap the values of $R_1(x_i)$ and $R_2(x_i)$ and of $I_1(x_i)$ and $I_2(x_i)$ respectively. If the values were swapped, re-assign the point to the new $I_1(x_i)$ cluster and add both $I_1(x_i)$ and $I_2(x_i)$ to the *live set*. After all points have been checked and re-assigned where needed, recalculate the cluster centres.

Continue with the quick-transfer phase until no points have been re-assigned in the last n steps. When complete, repeat the optimal-transfer phase until the *live set* is empty.

2.3 Graph-based methods

In the graph-based clustering methods, customer points are clustered using a graph. The graph can be constructed by letting the points represent nodes and the relationship (distance) between points be represented by edges that connect the nodes. Where the nodes are connected, the customer points belong to the same cluster.

At the start all edges are added to the graph and all customer points belong to the same cluster. Next, edges are evaluated and "inconsistent" edges are removed from the graph. This will result in groups of customer points being disconnected from each other, effectively forming a number of clusters.

An edge between points i and j is considered inconsistent if either

$$\frac{l_{ij}}{\bar{l}_i} \geq l_T \quad \text{and} \quad \frac{l_{ij}}{\bar{l}_j} \geq l_T, \quad \text{or} \quad (2.12)$$

$$\frac{l_{ij} - \bar{l}_i}{\sigma_i} \geq \sigma_T \quad \text{and} \quad \frac{l_{ij} - \bar{l}_j}{\sigma_j} \geq \sigma_T. \quad (2.13)$$

Here l_{ij} is the length of the edge between points i and j . $\bar{l}_i$ and $\bar{l}_j$ are the averages of the length of edges around points i and j respectively for a sub-tree of depth d . σ_i and σ_j are the sample standard deviations of the lengths of all the edges at points i and j for a sub-tree of depth d . σ_T and l_T are predefined cut-off values.

Eq. (2.12) is called the **ratio of edge lengths** for points i and j respectively. The **z-score** of points i and j respectively are given in Eq. (2.13), (Jain and Dubes, 1998, p. 122).

Zahn (1971) recommends using $\sigma_T = 3$ and $l_T = 2$ as a first pass to eliminate oversized edges from the statistics. The depth d used in identifying which edge to include in the statistics, is also referred to as the local neighbourhood depth. Zahn (1971) seems to favour $d = 3$.

Once the inconsistent edges have been removed from the constructed graph, the connected customer points are considered part of the same cluster. The number of clusters cannot be guaranteed to be an exact number, but depends on the values chosen for l_T and σ_i . Three types of graphs can be constructed based on the graph-based clustering. These are as follows:

2.3.1 MST graph-based clustering method

Zahn (1971) suggested a graph-based method where a minimum spanning tree (MST) is constructed. This is also referred to as Zahn's method. The minimum spanning tree includes all the edges needed to ensure all points are connected. Although the MST graph-based method works well with well-separated clusters, it seems to produce less good results if clusters have varying inter-cluster distances. Zahn (1971) suggests first identifying and removing the denser clusters before analysing the rest of the data.

2.3.2 RNG graph-based clustering method

Jain and Dubes (1998) suggest another graph-based method, using the relative neighbourhood graph (RNG) instead of the MST. In order to construct a RNG; points i and j are declared connected if and only if

$$d(x_i, x_j) \leq \max \left\{ d(x_i, x_k), d(x_j, x_k) \right\} \quad \forall k, k \neq i \text{ and } k \neq j, \quad (2.14)$$

where k represents all points in the graph excluding points i and j and $d(x_i, x_j)$ is the Euclidean distance between points i and j . Points x_i and x_j are therefore only connected if no other points fall in the region of influence. Here the region of influence is the overlapping area of the two circles with a radius of $d(x_i, x_j)$ originating at points x_i and x_j , (Jain and Dubes, 1998), illustrated in Figure 2.4.a.

Inconsistent edges are determined using Eq. (2.12) and Eq. (2.13). According to Jain and Dubes (1998) the RNG graph-based clustering is more effective than the MST graph-based method.

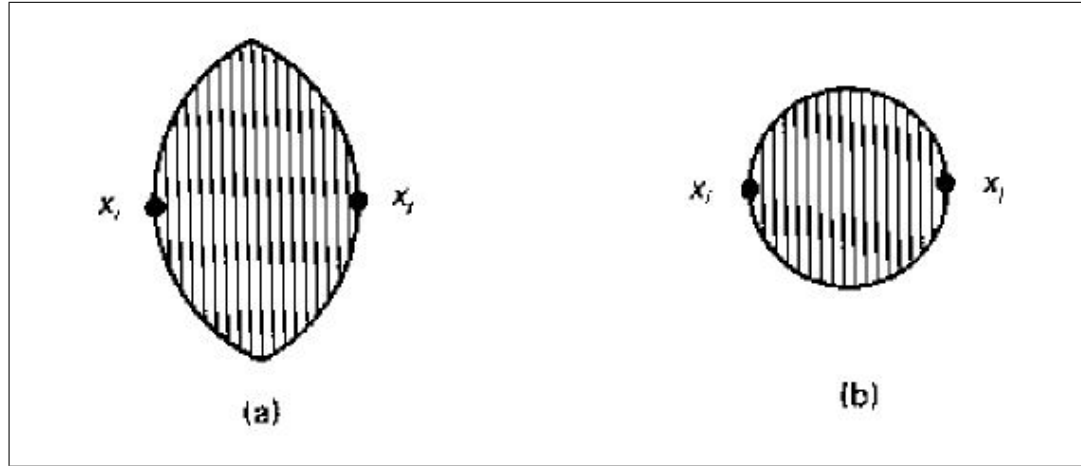


Figure 2.4: The regions of influence for the RNG and GG, (Jain and Dubes, 1998, p. 125).

2.3.3 GG graph-based clustering method

A third method suggested by Jain and Dubes (1998) is to construct a Gabriel graph (GG). For this type of graph points are connected if and only if

$$d^2(x_i, x_j) < d^2(x_i, x_k) + d^2(x_j, x_k) \quad \forall k, k \neq i \text{ and } k \neq j. \quad (2.15)$$

The region of influence for the GG is illustrated in Figure 2.4.b. Points x_i and x_j are connected if the circle associated with the diameter having x_i and x_j as endpoints does not have any other points within its region of influence. Jain and Dubes (1998)

state that the use of the RNG and GG graph-based methods are less sensitive to changes in pattern densities.

2.4 Nearest neighbour methods

Clustering can easily be based on the nearest neighbours. Here the nearest neighbours of each customer point are identified and customer points are placed in the same clusters as these neighbours, (Jain and Dubes, 1998). Two nearest neighbour methods discussed in more detail are as follows:

2.4.1 The k–near neighbours method

In the k–near neighbours method, a point x_i is assigned to the same cluster as x_j if both are within the nearest k neighbours of each other, called k–near neighbours, and both share at least k_t nearest neighbours, (Jain and Dubes, 1998). The user has to specify the values k and k_t and the best values are determined iteratively. This method is very attractive because of its computational ease, but it can be difficult to find the correct parameter values to match the number of clusters required.

The method starts by assigning each customer point to its own cluster. For each point identify the k nearest neighbours. For each customer pair, check if they share at least k_t nearest neighbours. If they do, merge the clusters the two points are assigned to. Continue until all the neighbours of all the customers have been checked.

2.4.2 The mutual neighbourhood value (MNV) method

Jain and Dubes (1998) refer to another method, called the MNV method. If x_j is the q_{th}^1 nearest neighbour to x_i and x_i is the q_{th}^2 nearest neighbour to x_j , then the MNV is $q^1 + q^2$. The smaller the MNV of two points, the closer they are to each other. Points are clustered together in a similar manner to the single linkage method, but based on the MNV.

The MNV of customer point pairs are calculated. All pairs with a MNV below a predefined threshold value s are assigned to the same cluster, having $q^1 + q^2 \leq s$. To eliminate computation of points far from each other, only combinations with the s nearest neighbours of each point are considered, i.e. where $q^1 \leq s$ and $q^2 \leq s$. The smaller the value of s , the more clusters will be formed with strongly defined clusters where points are closer to one another than with a large value of s . As a rule of thumb the value of $s = 5$ is used by Jain and Dubes (1998).

The algorithm is described as follows:

1. Assign each customer to its own cluster.
2. Identify the nearest s neighbours of each customer point. Order the neighbours from closest to furthest.
3. Calculate the MNV of each customer pair. If the one is not within the s nearest neighbours or the MNV is bigger than s , set the MNV to an arbitrary large value.
4. Identify all pairs with a MNV of 2. Merge the clusters of the two points into one cluster.
5. Repeat Step 4 for MNV threshold values of $3, 4, \dots, s$. Every time merging the clusters of the associated pairs.

The method is said to return the same results as the graph-based RNG method, (Jain and Dubes, 1998, p. 91–92).

2.5 Density-based methods

Density-based clustering divides the points into regions and base the cluster centres in the regions with the highest number of points. The closer the points are to each other, the higher the density of the region. These high density centres are also referred to as modes, (Jain and Dubes, 1998).

In Euclidean space, the simplest method to calculate p high-density modes is to divide the area where the customer points lie into a number *larger than* p of equally sized non-overlapping regions, (Jain and Dubes, 1998). For each region k , count the number of points, n_k . The centres of regions with the p highest number of points then becomes the highest density.

Formally, the probability density estimate $\hat{p}_n(x_i)$ can be calculated as

$$\hat{p}_n(x_i) = \frac{n_k}{nV_k(x_i)}, \quad (2.16)$$

where n_k is the number of points in region k , n is the total number of points and $V_k(x_i)$ is the volume of region k around point x_i , (Jain and Dubes, 1998). The size of $V_k(x_i)$ is critical for a good solution. If $V_k(x_i)$ is too small, there will not be a clear indication of the areas with high densities. If $V_k(x_i)$ is too large, the regions will be overly smooth and modes might be overlooked.

Jain and Dubes (1998) suggest calculating the volume of the regions in such a way that each region represents a unimodal graph, i.e. there is one density peak per region. They suggest two methods of determining $V_k(x_i)$, either by using the nearest

neighbour, or the *parzen window* approach. In the first, the volume is a function of a fixed number k of nearest neighbours around a point, x_i . The volume will differ for each point, depending on the distance to the nearest neighbours. In the latter method, the volume within a predefined window, called the *parzen window*, is fixed and the number of points in the window will vary. In both methods it is suggested that the term $\frac{n_k}{V_k(x_i)}$ should be proportional to $\sqrt{n}$.

The following two clustering methods use the above mentioned approaches:

2.5.1 The k_{th} nearest neighbour clustering method

This method was introduced by Wong and Lane (1981) and should not be confused with the k -near neighbours method discussed previously. This method uses the fixed number of nearest neighbours method discussed above. Two points, x_i and x_j , are called neighbours of each other if x_i is one of the closest k points to x_j and x_j is also one of the closest k points to x_i .

A computed density-distance matrix is calculated with the density-distance between points x_i and x_j as inversely proportional to the estimated density of the point halfway between them. The density-distance between x_i and x_j , $\hat{d}(x_i, x_j)$, is formulated as

$$\hat{d}(x_i, x_j) = \begin{cases} \frac{1}{2\hat{p}_n(x_i)} + \frac{1}{2\hat{p}_n(x_j)} & \text{if } x_i, x_j \text{ are neighbours} \\ \infty & \text{if } x_i, x_j \text{ are not neighbours.} \end{cases} \quad (2.17)$$

$$= \begin{cases} \frac{n}{2k} (V_k(x_i) + V_k(x_j)) & \text{if } x_i, x_j \text{ are neighbours} \\ \infty & \text{if } x_i, x_j \text{ are not neighbours.} \end{cases} \quad (2.18)$$

Wong and Lane (1981) suggest replacing the term $\frac{nV_k(x_i)}{k}$ with $d_k(x_i)$ in the computational algorithm. Here $d_k(x_i)$ is the Euclidean distance for point x_i to its k_{th} nearest neighbour. The computed density-distance can then be calculated as

$$\hat{d}(x_i, x_j) = \begin{cases} \frac{1}{2}(d_k(x_i) + d_k(x_j)) & \text{if } d(x_i, x_j) \leq d_k(x_i) \text{ or } d_k(x_j) \\ \infty & \text{otherwise,} \end{cases} \quad (2.19)$$

where $d(x_i, x_j)$ is the Euclidean distance between points x_i and x_j . All pairs of points with density-distances $\hat{d}(x_i, x_j) < \infty$ are then merged into one cluster

Wong (1982) states that the method is not very practical for large datasets and has a computation time of $O(n^2)$.

2.5.2 The hybrid clustering density-based method

This method uses the *parzen window* density-based approach to create clusters and consists of two phases (Wong, 1982):

1. **Phase 1:** a value c , bigger than the required number of clusters k , is calculated and the data set is divided into c clusters using the k-means method. The suggested formula for determining c is

$$c = 7 \left(\frac{n}{\log(n)^{1/3}} \right). \quad (2.20)$$

2. **Phase 2:** the c clusters are merged together using the single linkage method. The distance between two clusters is inversely proportional to the density of the point halfway between the two cluster means, $\bar{x}_i$ and $\bar{x}_j$. This can be calculated as

$$\hat{d}(\bar{x}_i, \bar{x}_j) = \begin{cases} \frac{1}{\hat{p}_n(x_{ij})} & \text{if clusters } i \text{ and } j \text{ are connected} \\ \infty & \text{otherwise,} \end{cases} \quad (2.21)$$

where clusters i and j are connected if the midpoint x_{ij} , between $\bar{x}_i$ and $\bar{x}_j$, is closer to $\bar{x}_i$ or $\bar{x}_j$ than to any other cluster means. $\hat{p}_n(x_{ij})$ is the probability density estimate at the midpoint x_{ij} for neighbouring clusters i and j . In two-dimensional Euclidean space $\hat{p}_n(x_{ij})$ can be calculated as

$$\hat{p}_n(x_{ij}) \propto \frac{(n_i + n_j)^{\frac{3}{2}}}{(E_i + E_j + \frac{1}{4}(n_i + n_j)d^2(\bar{x}_i, \bar{x}_j))^{\frac{1}{2}}},$$

where E_i and E_j are the within cluster sum of squares for clusters i and j respectively and $d(\bar{x}_i, \bar{x}_j)$ is the Euclidean distance between the centres of cluster i and cluster j .

After constructing a distance matrix for the c clusters, the connected clusters closest to each other are merged using the single linkage method until all clusters with distances smaller than infinity have been merged. Unconnected clusters will not be merged, because the distances between them are infinite. Wong (1982) gives the time complexity as $O(nc)$.

It is clear that the number of clusters in the final solution depends on the value chosen for c in phase 1 and the distance between the clusters created in the k-means method. It will not necessarily be the same as p , the predetermined number of clusters wanted.

2.6 Hybrid clustering methods

Beside the hybrid clustering density-based method, other hybrid clustering methods have also been suggested in the past. Min (1987) proposes a hybrid heuristic, where the clusters centres from Ward's method are used as an initial starting solution for the k-means method. The k-means method is discussed in more detail in Section 2.2.1. Similar methods combining SAHN and iterative partitioning methods are also referred to by Han and Kamber (2006) as multiple-phase clustering, while Everitt et al. (2011) refers to this as adding a preclustering or sampling phase. Hybrid clustering methods including the above mentioned are tested in Section 4.3.6.

2.7 Clustering criteria

The clustering criterion is used as the objective function in optimisation methods to guide the method towards good clustering solutions. An example of a clustering criterion is to minimise the maximum distance between any two points in the same cluster. In iterative partitioning methods, a clustering criterion can be used in the assignment of points to clusters phase instead of assigning a point to the closest cluster centre, (Jain and Dubes, 1998).

According to Everitt et al. (2011), the clustering criterion can also be used to quantify the quality of a clustering solution compared to other solutions. It is important to note that different criteria can have different desired outcomes and do not always rate solutions the same.

Everitt et al. (2011, Ch. 5) discuss two types of goals concerning clustering criteria. The criteria can either minimise the lack of homogeneity within clusters or maximise the separation of the clusters. The authors note that the objective criteria to minimising the lack of homogeneity within clusters is equivalent to maximising the separation of the clusters. For this reason the lack of homogeneity measures are used for the remainder of this work.

Three measures per cluster (also known as clustering indices) are defined for the lack of homogeneity:

1. The sum of all the squared distances between all pairs of points within a cluster, called $h_1(m)$. This is also called the within-cluster sums of dissimilarity measure by Brusco and Stahl (2005). Here m is the m_{th} cluster, $m \in \{1, 2, \dots, p\}$.
2. The maximum value found during calculation of the first index, called $h_2(m)$. This is also known as the cluster diameter, (Brusco and Stahl, 2005, p 17).

3. The minimum sum of squared differences between all points in a cluster and the medoid, called $h_3(m)$. Here the minimum value is found at the most central value. In the case of the discrete clustering problem, this point is called the *medoid* of the cluster, and in the continuous case, it is the *means* of the cluster, (Everitt et al., 2011). Brusco and Stahl (2005) refer to this as the within-cluster sum of squared deviation from the means. Salhi and Nagy (2009) refer to this, in a paper on the pLRP, as the radial distance.

The formulations for $h_1(m)$, $h_2(m)$ and $h_3(m)$, (Everitt et al., 2011), are as follows:

$$h_1(m) = \sum_{i=1}^{n_m} \sum_{j=1}^{i-1} (d_{ij})^2, \quad i \neq j \quad i, j \in \{1, 2, \dots, n_m\} \quad (2.22)$$

$$= \frac{1}{2} \left[\sum_{l=1}^{n_m} \sum_{v=1, v \neq l}^{n_m} (d_{lv})^2 \right] \quad l, v \in \{1, 2, \dots, n_m\}, \quad (2.23)$$

$$h_2(m) = \max \left[(d_{lv})^2 \right] \quad l, v \in \{1, 2, \dots, n_m\}, \quad (2.24)$$

$$h_3(m) = \min_{v \in \{1, \dots, n_m\}} \left[\sum_{l=1}^{n_m} (d_{lv})^2 \right] \quad l \neq v, \quad (2.25)$$

where n_m is the number of points in cluster m and d_{lv} is the distance between points l and v within cluster m .

Everitt et al. (2011) and Brusco and Stahl (2005) state that in order to create a clustering criterion for a complete clustering solution, the measures per cluster need to be calculated across all clusters in the solution. Four types of summary calculations are suggested: summation, averaging, minimum and maximum.

We identify five cluster criteria to measure the quality of a clustering solution, based on the above summary calculations, in this thesis. These criteria will be used to measure and compare the quality of the clustering solutions obtained by the different methods. They are as follows:

1. The most popular criterion used in clustering problems is to minimise the sum of the squared distances of points within clusters to their respective cluster centres, the sum of $h_3(m)$. This is also called the within cluster variation criterion, (Jain and Dubes, 1998) or the star index, (Everitt et al., 2011). The authors also state that this criterion is equivalent to maximising the between-cluster variation. In this work this measure is referred to as the **sum of squared residuals (SSR)**. Here the term "residual" refers to the dissimilarity between the actual point and the respective cluster centres.

2. If the sum of the within-cluster sums of all dissimilarities, $\sum h_1(m)$, is calculated there is a possibility that the sum can become very large if there are a lot of points in one cluster. Everitt et al. (2011) suggested that one should rather use the average aggregation by dividing each measure per cluster by the number of points in the cluster first, before summing the total. Similar to this, one could also calculate the average aggregation of the within-cluster sum of dissimilarities to the mean (thereby using index $h_3(m)$ instead of $h_1(m)$). This criterion is referred to as the **mean sum of square residuals (MSR)** criterion. The disadvantage of using the MSR criterion is that it does not consider uneven sized clusters.
3. Brusco and Stahl (2005) suggest determining the maximum cluster diameter across all clusters, called the **partition diameter (PD)**.
4. The average lack of homogeneity using index $h_1(m)$ is given by Everitt et al. (2011) as the most appropriate measure. They call this measure the **average distance measure (ADM)**.
5. Brusco and Stahl (2005) mention using the sum of within cluster distances divided by the number of pairs. The averages per cluster are then summed together. This measure is called the **average within cluster distance (AWCD)** measure.

The criteria is to find the solution with the minimum of the above measures. The measures can be formulated as follows (Everitt et al., 2011 and Brusco and Stahl, 2005).

$$SSR = \sum_{m=1}^p h_3(m) \quad (2.26)$$

$$MSR = \sum_{m=1}^p \frac{h_3(m)}{n_m} \quad (2.27)$$

$$PD = \max_{m=1, \dots, p} \left[h_2(m) \right] \quad (2.28)$$

$$ADM = \frac{1}{n} \sum_{m=1}^p h_1(m) \quad (2.29)$$

$$AWCD = \sum_{m=1}^p \frac{h_1(m)}{\frac{n_m}{2}(n_m - 1)}, \quad (2.30)$$

where $h_1(m)$, $h_2(m)$ and $h_3(m)$ have been defined in Eq. (2.23) – (2.25), p is the number of clusters, n is the number of points and n_m is the number of points in cluster m .

2.8 Distance calculations

Han and Kamber (2006) provide an explanation for the different distance measures that can be used, as follows. Distances between points can be calculated with different distance measures, like Euclidean or absolute distances. The general formula for calculating distance is:

$$d(x_i, x_j) = \|x_i - x_j\|_q = \left(\sum_{r=1}^l (|x_i^r - x_j^r|^q) \right)^{1/q} \quad (2.31)$$

where l is the total number of variables defined in the Euclidean space, x_i^r and x_j^r are the values of the variables at points x_i and x_j in dimension r and q is the type of distance measure to be used.

The generalised distance formula, Eq. (2.31), is also known as the Minkowski distance formula, where the value of q needs to be decided beforehand. If $q = 2$ and there are in total l variables defined in the Euclidean space, $\mathcal{R}^l$, then the squared Euclidean distance between clusters h and k is

$$d_{hk} = \sqrt{\sum_{r=1}^l (x_h^r - x_k^r)^2} . \quad (2.32)$$

In the case where each point k consists of two coordinates (longitude and latitude) in $\mathcal{R}^2$ Euclidean space, the coordinate vector can be represented as $\mathbf{x}_k = (x_k^1, x_k^2)$. Here the Euclidean distances between points h and k can be calculated as

$$d_{hk} = \sqrt{(x_h^1 - x_k^1)^2 + (x_h^2 - x_k^2)^2} . \quad (2.33)$$

For Manhattan distances, absolute values are used and the value of q is one. Other values of q can also be used. In the k-means method, the Euclidean distance formula is used, while the k-medoids method uses Manhattan distances to minimise the absolute distance error.

2.9 Summary

In this chapter different clustering methods to cluster geographical data are discussed. The SAHN methods continue merging until a stopping rule criterion such as a specified number of clusters, has been reached. Iterative partitioning methods do not use stopping rules and are the most popular methods to use, but a known number of clusters is needed at the start. These methods also need a starting solution as input and some methods can only create good solutions if a good starting solution is used. The graph-based, nearest neighbour and density-based methods return the number of clusters as a decision variable, depending on the predefined

input parameters.

Points can be aggregated into smaller groups and treated as a single customer by a second type of clustering method, referred to as a hybrid clustering method. There is not much literature comparing different types of hybrid clustering combinations nor on the cut-off point between the first and second clustering method.

A number of criteria have been suggested in the past to measure the quality of clusters. Five of these measures are used to compare clustering methods in Chapter 4. Different distance calculations are also discussed. In Chapter 4, Euclidean distance as given by Eq. (2.33) is used in the distance calculations.

The next chapter is a literature review of different distribution network problems including the facility location problem (FLP), the vehicle routing problem (VRP) and the location routing problem (LRP).

Chapter 3

Literature review - distribution network problems

This chapter describes the relevant distribution network problems used in this study, namely:

- Section 3.1 – the facility location problem (FLP)
- Section 3.2 – the vehicle routing problem (VRP)
- Section 3.3 – the location-routing problem (LRP)
- Section 3.4 – the periodic location-routing problem (PLRP)

Problem formulations and solution approaches found in the literature for the above mentioned problems are included. A summary of the chapter is given in Section 3.5.

Table 3.1 gives a summary of the different characteristics of the distribution network problems to be discussed. The LRP is a combination of the FLP and VRP, whereas the PLRP introduces a periodic component as well.

Problem acronymn	Descriptive name	Multi- source	Type of problem		Depot selection to be made			Depot locations			Capacitated		Perio- dic
			Vehicle routing	Depot / cluster selection	Given	Select p depots / clusters	Based on costs	Conti- nuous, on plane	Discrete, separate list provided	Discrete, concen- trator problem	Clusters depots	Vehicles routes	
Facility location / Capacitated clustering problem													
p-median	p-median problem			✓		✓				✓			
CpMP	Capacitated p-median problem			✓		✓				✓	✓		
CFLP	Fixed charge capacitated facility location problem			✓			✓		✓		✓		
p-CCCP	p-capacitated centred clustering problem			✓		✓		✓			✓		
generic CCCP	Generic capacitated centred clustering problem			✓			✓	✓			✓		
CCP	Capacitated clustering problem			✓		✓			✓		✓		
CCLP	Capacitated concentrator location problem			✓			✓			✓	✓		
MSWP	Multi-source Weber problem	✓		✓		✓		✓					
TP	Transportation problem	✓			✓				✓		✓		
Vehicle routing problem													
MDVRP	Multi-depot vehicle routing problem		✓		✓				✓			✓	
PVRP	Periodic vehicle routing problem		✓		✓				✓			✓	✓
Location-routing problem													
HpMP	Hamiltonian p-median problem		✓	✓		✓		✓					
pILRP	Planar location-routing problem		✓	✓			✓	✓				✓	
plCLRP	Planar / continuous capacitated location-routing problem		✓	✓			✓	✓			✓	✓	
cdCLRP	Concentrator discrete capacitated location-routing problem		✓	✓			✓			✓	✓	✓	
sdCLRP	Standard discrete capacitated location-routing problem		✓	✓			✓		✓		✓	✓	
PLRP	Periodic location-routing problem		✓	✓			✓		✓		✓	✓	✓

Table 3.1: An overview of the distribution network problem definitions.

3.1 The facility location problem (FLP)

Facility location is defined by Drezner and Hamacher (2001) and Ceselli et al. (2009) as a collective term for problems with the goal of finding the best locations to place facilities given a company's distribution network. Facilities are sometimes referred to as depots, warehouses or distribution centres. These problems include assigning sites (also called customers or demand points) to the depots, based on a given criteria such as shortest distance, minimising costs or maximising coverage.

The different location problems are known in the literature under many names, such as the Weber problem, spatial median, p -median, centre of gravity, network location and location-allocation, to name but a few, see for example, Drezner and Hamacher (2001) and Bischoff and Dächert (2009).

The FLP assumes that all customers are to be serviced separately using single trips between the depots and back. These are also known as truckload (TL) shipments, (Min, 1996). The demand of each customer determines the importance of each contribution to the optimal location placement. The frequency of services as well as the relative size of the demand per service trip can be used as the weight to bias the solution towards customers with a higher demand. A customer's demand can be met by various depots or restricted to a singular depot. The latter is called the **single-source FLP**. The problem can have no depot capacity supply restrictions (called the uncapacitated facility location problem or UFLP) or have depot capacity constraints, referred to as the capacitated facility location problem (CFLP).

The FLP can be a continuous or a discrete location problem, (Mladenović et al., 2007). In the continuous case, the depots are centred anywhere on a plane and calculations are needed to determine the best positions to place them based on the customer points assigned to them.

Sometimes borders are used to prevent depots from being placed in unwanted areas. These are referred to as forbidden regions, (Salhi and Nagy, 2009). Luis et al. (2009) refer to a continuous FLP where the number of depots to use are known beforehand as the capacitated multi-source Weber problem (MSWP). Continuous problems can also be network-based, where depots can be placed anywhere within a finite union of linear continuous sets. For discrete problems a set of potential depots is given and the most appropriate positions must be selected.

In continuous location models, customers are grouped together and the centroid of each group is determined. These centroids then become the optimal locations for each group. The goal is to minimise the within-clustered distances, also called radial distances by Salhi and Nagy (2009). One method to calculate the placement of the centroids is to use the Weiszfeld equation, discussed in Section 3.1.4.1.

3.1 The facility location problem (FLP)

The continuous problem is often used, in high-level strategic decisions, to identify possible areas for further investigations. When possible locations have been identified and the estimated costs and capacity constraints are known, the discrete model becomes more practical.

Discrete problems assign customer points to a subset of the potential depots and there are therefore no calculations needed to determine the depot positions within the plane. The starting subset selection is a well discussed topic of interest in iterative partitioning methods. Ideas vary from random selection, to basing the selection on the highest customer demand (in the case of the p -median) to distributing depots evenly across the search space. A logical approach to solve discrete location problems is k -medoids iterative partitioning methods, because the centres of the clusters are actual customer points. Different starting solutions as well as the k -medoids methods, referred to as the j -means and PAM methods, are explained in Section 2.2.3.

Rationales behind providing a set of possible depots include a preference for industrial areas. The number of skilled workers available in a given area also makes established towns more preferable to the countryside. Geographic areas and access to main roads can also play a role. Rand (1976) discusses the pros and cons for using a discrete set of depots versus the continuous case. The author mentions that distribution costs must be a monotonic function of distance in the continuous case which is not necessarily the case for the discrete problem.

Yet another factor is the position of competitors. While some companies might prefer to be further away from similar companies, others would prefer to be close to their competitors. In the placement of new ATMs, for example, it is a common strategy to place these ATMs close to other banks' ATMs. Not only does this enhance security, but also make them easier to find. Other security issues also come to mind, such as avoiding high-risk areas.

If single-source constraints are assumed, the mathematical formulation becomes a zero-one integer programming model (0-1 IP), because all decision variables are binary. A customer is either assigned to a depot, represented by 1, or not, represented by 0. This study will only address the single-source FLP.

3.1.1 The discrete CFLP

Drezner and Hamacher (2001, Ch. 3) identified eight different discrete facility location models: set covering, maximal covering, p -center, p -dispersion, p -median, fixed charge, hub and maxisum. The problem formulations of the p -median and fixed charge FLP are relevant to the CLRPP and discussed in further detail. The capacitated clustering problem (CCP) by Mulvey and Beck (1984) is another variant

of the CFLP discussed below.

3.1.1.1 The capacitated p-median problem (CpMP)

In a p-median problem, the goal is to find the median(s) and minimise the sum of the within-clustered Euclidean distances between customers and the median(s), (Drezner and Hamacher, 2001). It is a discrete problem where the list of potential depots is the same as the list of customers. Any customer can serve as a potential depot and the selected depots are called the medians or concentrators of each cluster. If only a limited capacity can be assigned to each of the p centres, then the problem is referred to as the CpMP. This can also be referred to as a capacitated p-concentrator location problem, the CCLP (Ceselli et al., 2009).

The number of depots to select is restricted to p . The objective is to minimise the distance from the chosen p depots to the customers assigned to them. Each customer must be assigned to a depot and can only be assigned to one depot to enforce the single-source constraints.

Problem formulation

The problem can be represented by a graph $G(V, A)$, where V is the set of all points in the graph and A is the set of directed edges between the points, used to represent customer assignments. The CpMP can now be formulated as the following integer linear programming (ILP) model:

$$\text{Minimise} \quad \sum_{j \in V} \sum_{i \in V} d_{ij} x_{ij} \quad (3.1)$$

$$\text{subject to} \quad \sum_{j \in V} x_{ij} = 1, \quad \forall i \in V, \quad (3.2)$$

$$\sum_{j \in V} x_{jj} = p, \quad (3.3)$$

$$\sum_{i \in V} w_i x_{ij} \leq W x_{jj}, \quad \forall j \in V, \quad (3.4)$$

$$x_{ij} \leq x_{jj}, \quad \forall i \in V, \forall j \in V, \quad (3.5)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in V, \forall j \in V, \quad (3.6)$$

where V is the set of n customers to be clustered. The decision variables $(x_{ij} \forall i \in V, \forall j \in V)$ are binary with $x_{ij} = 1$ if customer i is assigned to depot j . If customer j is a depot, then $x_{jj} = 1$. The number of depots allowed is p . The distance used in the objective function, d_{ij} , is the Euclidean distance between point i and point j

as given in Eq. (2.32). The demand of the customer i is w_i and W is the maximum capacity of any depot.

3.1.1.2 The capacitated clustering problem (CCP)

The capacitated clustering problem (CCP) is a discrete clustering problem with a set of candidate centres or depots to select from, where exactly p capacitated clusters needs to be created. Similar to the CpMP it assumes that each customer has a demand and each facility/depot has a limited supply. According to Negreiros and Palhano (2006), the CpMP is a specific case of the CCP where, the capacity constraints for all the depots are homogeneous and the coefficients of the objective function are distances.

In general, for p -median problems, the list of potential cluster centres is the same as the list of points, whereas for the CCP and discrete CFLP the list can differ from the list of points. According to Mulvey and Beck (1984), the CCP is NP-hard.

Problem formulation

Given a set I of n customer points with an associated w_i weight (or demand) per point, group the points into p clusters in order to minimise the total dissimilarity between the points in a cluster and the cluster centre, given a maximum capacity constraint per cluster, W_j . The set J represents the list of m potential cluster centres and the set of all points in the graph is then $V = I \cup J$. The formulation for the CCP given by Mulvey and Beck (1984) as follows:

$$\text{Minimise} \quad \sum_{j \in J} \sum_{i \in I} d_{ij} x_{ij} \quad (3.7)$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I, \quad (3.8)$$

$$\sum_{j \in J} y_j = p \quad , \quad (3.9)$$

$$x_{ij} \leq y_j \quad \forall i \in I, \forall j \in J, \quad (3.10)$$

$$\sum_{i \in I} w_i x_{ij} \leq W_j \quad \forall j \in J, \quad (3.11)$$

$$x_{ij}, y_j \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \quad (3.12)$$

where w_i is the demand of the customer i , W_j the maximum capacity of cluster j , p is the specified number of clusters to select so that $m \geq p$. $x_{ij} = 1$ if point x_i is assigned to cluster j and the binary variable $y_j = 1$ if point j is chosen as a cluster

3.1 The facility location problem (FLP)

centre. The dissimilarity d_{ij} is calculated between a point and the median.

Constraints (3.8) ensure that each point is assigned to only one cluster centre, while constraints (3.9) limit the number of clusters to exactly p . Constraints (3.10) allow points to only be assigned to the selected open cluster centres and constraints (3.11) enforce the capacity constraint per cluster. The last constraint ensures all variables are binary.

3.1.1.3 The fixed charge capacitated facility location problem (CFLP)

In the fixed charge variation of the FLP, the goal moves from minimising distances to minimising costs that also include fixed depot costs. The constraint on the number of depots is replaced with a fixed cost per depot. The number of depots then becomes a trade-off between the fixed costs and distribution costs.

In the literature, reference to the FLP assumes the fixed charge variant. See, for example, Erlenkotter (1978), Ceselli et al. (2009), Klose and Görtz (2007). In this study when referring to the FLP, the fixed charge variant is meant, except where explicitly stated otherwise.

Figure 3.1 illustrates an example of the costs versus the number of depots used. It is similar to the generalised cost graphs for the FLP as illustrated by Ballou (2004, p. 574) and Rand (1976). It shows how the distribution costs decrease exponentially as more depots are added.

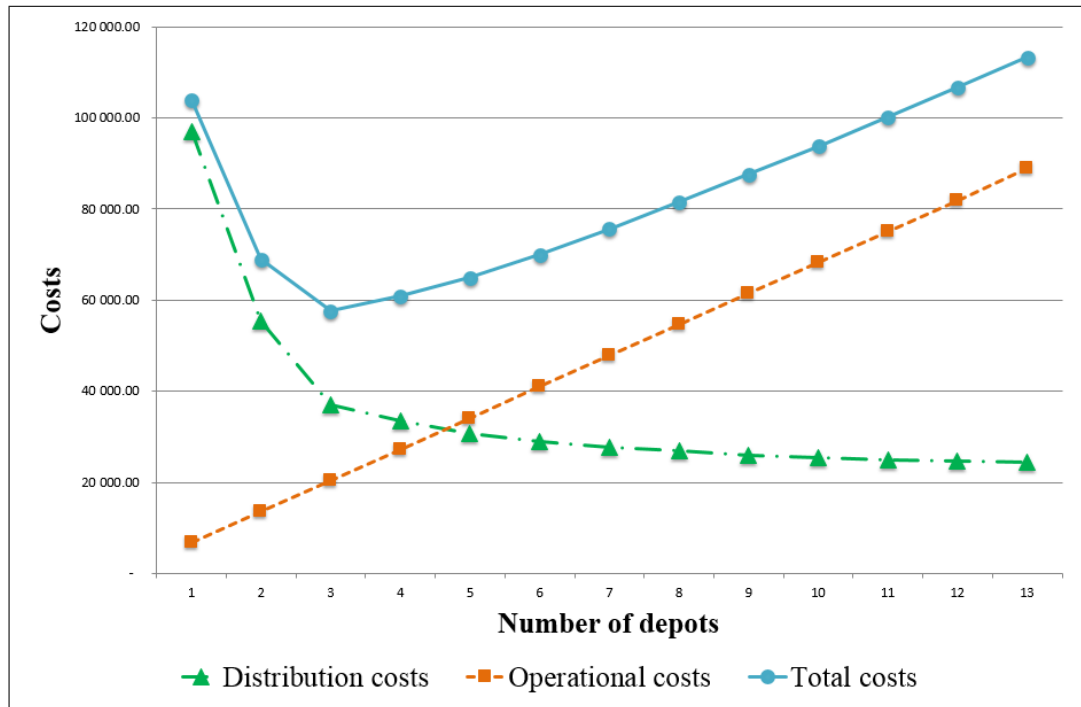


Figure 3.1: Typical costs associated with the FLP problem.

3.1 The facility location problem (FLP)

In practical situations the fixed costs per depot will typically not include the complete once off set-up costs for opening the depots in the objective function. The reason for this is that they are too high and impractical. Rather, the depot costs are divided by the expected pay-off term to calculate a periodic instalment cost. This cost together with the expected operating cost over the same time period is then used to create a fixed cost per depot.

In practice, if the fixed depot costs are not available and the number of depots to open is unknown, another popular approach is to determine the number of depots for which the decrease in distribution costs is still significant. The acceptable level of significance will then be determined by the decision maker.

In the FLP, the set of candidate depots does not have to be the same as the list of customers and a completely different set of possible depot locations can be used. An example of the discrete FLP is illustrated in Figure 3.2. The points denote the customers to be visited, the squares denote the candidate depots and the filled squares denote the chosen depots.

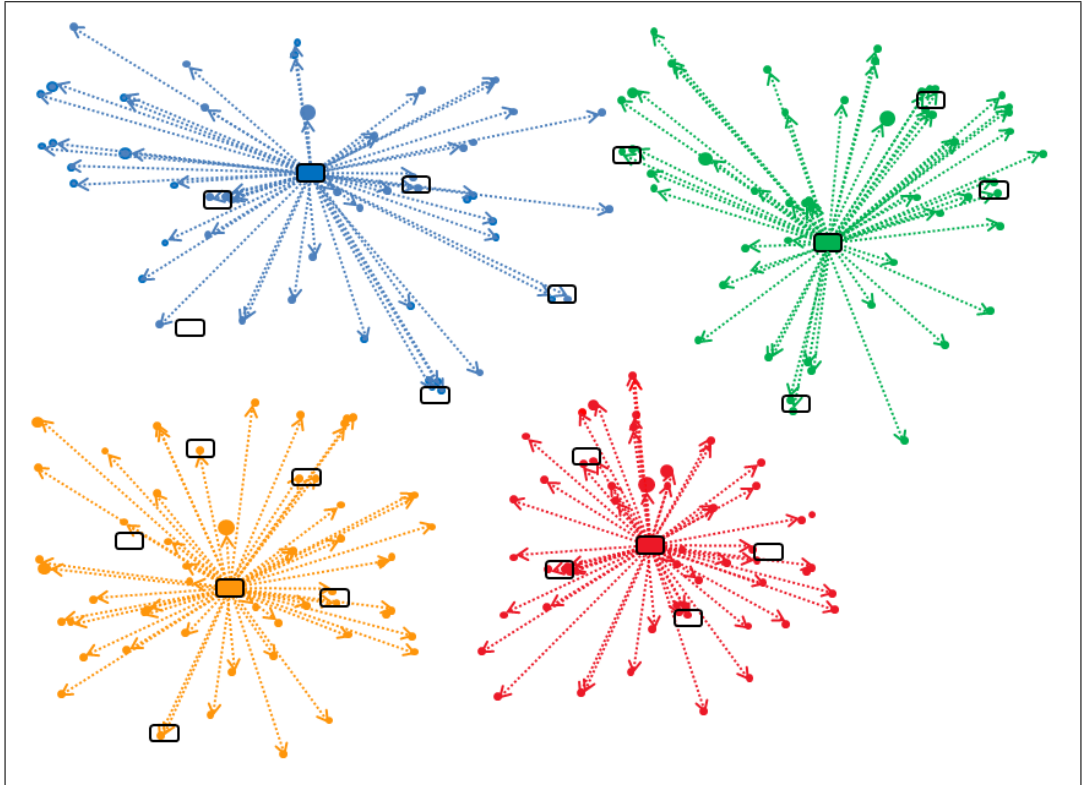


Figure 3.2: An illustration of the discrete FLP problem.

If there are no capacity constraints, customers should be assigned to the closest open depot in order to minimise distribution costs. If the depots are restricted by a supply capacity, the problem is called the capacitated FLP (CFLP) and if customers can only be assigned to one depot, the problem is called the **single-source capacitated facility location problem (SS-CFLP)**. Most researchers refer to the single-source

3.1 The facility location problem (FLP)

variant when referring to the CFLP, therefore in this work the single-source variant is assumed. In cases where the set of potential depots is equal to the list of customers, the problem is referred to as the capacitated concentrator location problem, the CCLP (Simchi-Levi and Bramel, 1997, Ch. 12 and Ceselli et al., 2009).

Problem formulation

The mathematical formulation for the discrete CFLP is as follows:

$$\text{Minimise} \quad \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} + \sum_{j \in J} f_j y_j \quad (3.13)$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I, \quad (3.14)$$

$$x_{ij} \leq y_j \quad \forall i \in I, \forall j \in J, \quad (3.15)$$

$$\sum_{i \in I} w_i x_{ij} \leq W_j y_j \quad \forall j \in J, \quad (3.16)$$

$$x_{ij}, y_j \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \quad (3.17)$$

where I is the set of all n customers, J is the set of all m candidate depots and f_j is the fixed cost for opening depot j . The distribution cost to deliver to customer i from depot j is c_{ij} . If depot j is open, $y_j = 1$ and if customer i is assigned to depot j , $x_{ij} = 1$.

The capacity can differ per depot. This is referred to as W_j while w_i is the demand of the customer i . The total demand of customers assigned to a depot j should be equal or less than W_j as shown in constraints (3.16).

The problem formulation differs from the p -median problem in the following ways: A new variable y_j , is introduced to determine whether depot j is open or not and the constraint on the number of open depots equal to p has been removed.

3.1.2 Solution approaches for the discrete CFLP

Various methods have been suggested in literature to solve the CFLP, see for example: Klose and Görtz (2007), Simchi-Levi and Bramel (1997, Chapter 12) and Drezner and Hamacher (2001). The problem is known to be NP-hard, (Simchi-Levi and Bramel, 1997, Chapter 6). The solution approaches provided in the literature can be divided into three main categories: exact, heuristic and metaheuristic approaches.

Although exact methods provide the optimal solution, they become too time consuming to be used for bigger case studies of NP-hard problems. Exact methods are

sometimes used in hybrid methods to solve parts of the bigger problem. They can also be used as incomplete optimisation methods where the exact method terminates prematurely based on a predefined time variable, (Simchi-Levi and Bramel, 1997, Chapter 6).

In order to find solutions at least close to optimality within a reasonable time, heuristic and metaheuristic approaches are used. Mladenović et al. (2007), Hansen and Mladenovic (1997), and Drezner and Hamacher (2001) provide a list of heuristic approaches used to solve the p -median and fixed charge FLP problems.

Whereas heuristic approaches are said to be more problem specific, metaheuristic approaches are seen as general frameworks for solving any combinatorial problem. Blum and Roli (2003) defines these approaches as not problem specific, but rather they can be adapted for each problem. The methods can make use of problem specific heuristic methods in their search. Metaheuristics are designed to avoid local optimal solutions by allowing inferior solutions from other parts of the solution search space, in order to expand the search in areas that would otherwise have been overlooked, (Mladenović et al., 2007). Metaheuristic approaches are normally non-deterministic and use some kind of guided search approach, which can include memory to find improved solutions.

Joubert (2006) adds that these approaches in general use three phases; the initialisation, diversification and intensification phases. In the initialisation phase a heuristic method can be used to create a starting solution. Some metaheuristics use randomly generated starting solutions. During the diversification phase the solution is adapted by searching through the different neighbourhoods of the search space. While improvements are in general accepted, solutions that are worse than the current best are also accepted based on a guide criteria to prevent the method from converging to a local optima. During the intensification phase, the metaheuristic will zoom in on the best found solution(s) and intensify the search in that specific search space. See Blum and Roli (2003) for a more in-depth description of metaheuristics.

Different metaheuristic approaches include simulated annealing, tabu search, genetic algorithms, ant colony optimisation and other approaches as mentioned by Blum and Roli (2003) and Gonzalez (2007, Part II). These methods are also often combined with other approaches and then referred to as hybrid methods. For example a Genetic Algorithm, followed by a local search, etc.

Although solution approaches of the CpMP can be used to solve the CFLP, knowing the right number of depots to use is not solved by these methods because it is a given constraint. Rand (1976) states that even when using an approximate algorithm that can find near optimal solutions in polynomial time to solve the p -median, the number of depots to use can only be determined through enumeration. This means

that the algorithm would have to be repeated for every number of depots to be tested. If there are m potential depots, and the algorithm is repeated for all cases from 1 to m number of depots, it means the time complexity will also become m times more.

A number of exact, heuristic and metaheuristic methods are now described to solve the CpMP and CFLP.

3.1.2.1 Exact methods

i) The exhaustive search method

The easiest exact method is the exhaustive search method, where every possible solution is evaluated and the best solution is returned. This is also known as explicit enumeration, (Winston, 2004, p 966).

ii) The branch-and-bound method

Another approach is called the branch-and-bound method. This method is referred to as intelligent enumeration, (Papadimitriou and Steiglitz, 1998, Ch. 18) and used to solve any mixed integer linear problems (MILP). An initial best feasible solution with integer values is used to find an upper bound for the problem. This is found by either solving the problem with a heuristic method, (Brusco and Stahl, 2005), or using the results from the first complete branch of the search, (Winston, 2004, Ch 9). The upper bound is updated every time a better feasible solution is found.

The method now starts by solving the linear programming (LP) relaxation of the problem, where all the integer constraints are ignored, optimally (Winston, 2004, Ch 9). The solution of the LP relaxation is called the lower bound of the MILP problem. The method branches on the non-integer values found in the LP relaxation solution. For the 0-1 MILP this means fixing $x_{ij} = 0$ in the first subproblem and $x_{ij} = 1$ in the second subproblem for a non-integer variable x_{ij} . The problem is again solved optimally and branching continues until all variables return integer variables.

A key part of the branch-and-bound method is to know when a subproblem is not worth investigating. A branch cannot better the value of the LP relaxation solution it originally split from, called the branch's lower bound. When it becomes clear that the lower bound of a particular branch is higher than the upper bound, the branch will not be investigated further. The speed of the branch-and-bound is therefore highly dependent on finding an effective heuristic that will produce a good initial upper bound to eliminate as many branches as possible.

iii) **Column generation**

The column generation method can be used to solve set covering or set partitioning problems, (Minoux, 1987), like the single-source CFLP. A set of customers assigned to the same depot is called a subset or column.

The problem is decomposed into a master problem and various subproblems, (Winston, 2004, Ch. 10.3). The integer constraints in the master problem are relaxed and solved optimally, using the simplex method (Neebe and Rao, 1983). At each step, column generation is used to determine the best non-basic variable (column) to enter the basis of the master problem.

Instead of listing all possible combinations of customers assigned to a depot as columns, the columns are generated. This is done by decomposing the problem into m binary knapsack subproblems (one for each depot). Each knapsack problem is written as a dual problem used to determine the maximum cost saving per depot. The column associated with the maximum cost saving of all the depots is selected to enter the basis of the master problem. The process continues until the master problem is solved. If the LP relaxation finds a non-integer solution, the problem branches, similar to the branch and bound method.

According to Lorena and Senne (2004), a straightforward application of column generation can result in slow convergence. The authors suggest using Lagrangian multipliers instead of the simplex multipliers. They also allow multiple columns to be introduced to the problem at the same time.

iv) **The branch-and-price method**

Ceselli et al. (2009) explain this method as a combination of the branch-and-bound and column generation methods. A master problem similar to that of the column generation method is generated. Instead of generating columns, the master problem is here restricted to a limited number of columns. Similar to the column generation method, the restricted master problem is solved exactly by means of the simplex method.

The solution to the restricted master problem is used as the lower bound. If the solution contains only integer values, the best feasible solution for the given subset of columns has been found. If the solution contains non-integer values, the problem is split into subproblems, using the branch-and-bound method. The best feasible solution from the subproblems gives the optimal solution.

Examples of facility location problems solved by the branch-and-price method can be found in Ceselli et al. (2009) and Klose and Görtz (2007).

v) **The dynamic programming method**

The dynamic programming method solves a problem in stages using different states in each stage. To solve the p -median, (Winston, 2004, p. 982–983), the stages are the number of depots to open and the states are the set of depots

used to open the number of depots in each stage. The method starts by solving the problem for one depot (stage one). The cost functions for opening each of the candidate depots are calculated and the minimum cost is noted as $g(1)$ and the set $x(1)$ is the depot associated with the best cost function.

The method now moves to stage two - opening two depots. There are two ways to open two depots using the results of the previous stages: Either depot $x(1)$ plus one other depot is opened, or $x(1)$ is not used and two other depots are opened. Both scenarios are enumerated and the best costs for stage two are noted as $g(2)$. If $x(1)$ is used, then the objective function is $g(1)$ plus the extra costs incurred from adding the second depot. $x(2)$ is then noted as the set containing the two depots used to calculate $g(2)$.

Stage three continues in a similar way. There are three methods to open three depots; use $x(2)$ and open one other depot, use $x(1)$ and open two other depots excluding the solution set $x(2)$ or open three other depots, where the depots exclude the sets $x(1)$ or $x(2)$.

In the case of the p -median, the stages continue until stage p , the desired number of depots is reached. Every time the enumerations exclude the depot sets that have already been determined in the previous stages. The method is therefore not equivalent to the exhaustive search method.

For the FLP, repeat the stages from 1 to m , where m is the number of potential depots or until it becomes apparent that

$$g(k) < g(j) \quad \forall \quad j \in (k + 1, m).$$

3.1.2.2 Greedy and greedy-drop heuristic methods

These well-known heuristic methods can be used to solve any general MILP, (Winston, 2004). The greedy method starts by finding one depot to open with the minimum best cost for servicing all customers. Depots are then added one by one, based on the best cost impact until the required p depots are opened. This is followed by a local search improvement algorithm where depots are replaced one at a time by a customer assigned to it, until a better solution is found. The search is repeated until no further improvements can be made. The method is also referred to as a construction heuristic.

The extended greedy construction method was introduced by Hribar and Daskin (1997). It is similar to the greedy method, but instead of only considering the best depot at each iteration, a set of H best solutions is used. For every new stage, the previous set of H solutions is appended with one more depot and a new best set is obtained. Care must be taken not to have duplicate sets in H .

3.1 The facility location problem (FLP)

Unfortunately, Hribar and Daskin (1997) are not able to provide guidelines for choosing a good H value, but their results suggest that the value has to grow non-linearly as p becomes larger in order to find optimal solutions.

The greedy-drop method is also called the simple elimination method and works the opposite way of the greedy method. In the case of the FLP the method starts with all depots open and depots are then eliminated one by one based on the smallest negative impact on the objective function until only p depots are left in the starting solution. A local search procedure can then also be performed to better the solution. Feldman et. al. (1966) referred to this method as the Stingy heuristic.

Salhi and Atkinson (1995) introduced the concept of using random subsets instead of the complete list of candidate depots as input for the drop heuristic. They referred to this method as the subdrop heuristic.

3.1.2.3 The alternate method

The alternate or improvement p -median method was introduced by Maranzana (1964). A random solution is selected and customers are assigned to the closest depot. For each set of customers assigned to the same depot, a new median is calculated (called a 1-median problem) and the closest potential depot to the median is chosen.

The method alternates between selecting depots and assigning customers to the closest depot until no more changes occur. This method can be repeated numerous times with different starting solutions, keeping the best overall solution.

This method differs from the j -means and PAM methods, because the closest point to the calculated median is selected as the depot, while the j -means and PAM methods test all points assigned to the cluster to determine the best depot.

3.1.2.4 The interchange method

This method is also known as the T&B (Drezner and Hamacher, 2001, p 182) method, named after its authors Teitz and Bart in 1968. As with the alternate method, a random starting solution is selected. Depots are then swapped one by one with candidate depots that have not been selected using enumeration.

If a better solution is found, the depot is replaced and the search continues until no more improvements can be made. The method can be repeated with different starting solutions. The interchange method is often used to compare results from other methods.

3.1 The facility location problem (FLP)

Mladenović et al. (2007) suggest the use of two new developments for the use of the interchange method. The first is to use a formula given by Resende and Werneck (2003) to determine the best interchange:

$$\text{Maximise } w_{hj} = \sum_{i \in I} \max \{0, d_1(i) - d(i, h)\} - \sum_{i \in I} [d_2(i) - d_1(i)] + e_{hj}, \quad (3.18)$$

where w_{hj} represents the gain from replacing depot j with depot h and I is the set of all customers and $j = m_1(i)$ is the closest depot to customer i . The distance from customer i to depot h is $d(i, h)$ and $d_1(i)$ and $d_2(i)$ are the distances to the closest and second closest depots for customer i respectively, for the current open depot set.

The first term represents the reduction in distances of customers that can be re-assigned to depot h . The second term represents the increase in distances for customers assigned to depot j that now needs to be re-assigned to their second closest open depot. The last term e_{hj} is the decrease in distances for customers currently assigned to depot j that should rather be assigned to depot h than to the second closest open depot. e_{hj} can be calculated with the formula:

$$e_{hj} = \sum_{i \in I} [d_2(i) - \max \{d(i, h), d_1(i)\}] \text{ where } d(i, h) < d_2(i). \quad (3.19)$$

The thought behind the introduction of the term e_{hj} is that in large datasets, the computational time of w_{hj} will be smaller if only a few customers move assignments from depot j to h .

Mladenović et al. (2007) mention a second improvement to the interchange method. Here a neighbourhood depth parameter, k , is used. Depots that have been dropped in the last k moves cannot enter the solution. This is known as the Lin-Kernighan $LK(k)$ neighbourhood.

3.1.2.5 The Lagrangian relaxation method (LR)

The LR method for the p-median is explained in Drezner and Hamacher (2001, p. 104–107). A constraint set of the original problem is "relaxed" (i.e. eliminated) from the problem. The relaxed constraint set is then multiplied by Lagrange multipliers and added to the objective function. For example, in the problem formulation of the CpMP, constraint set (3.2) are relaxed. This is the constraint set that ensures all customers are assigned to one and only one depot.

3.1 The facility location problem (FLP)

The LR problem can now be formulated as follows:

$$\max_{\lambda_i} \min_{x_{ij}} \quad \sum_{j \in I} \sum_{i \in I} d_{ij} x_{ij} + \sum_{i \in I} \lambda_i \left(1 - \sum_{j \in I} x_{ij} \right), \quad (3.20)$$

$$= \sum_{j \in I} \sum_{i \in I} (d_{ij} - \lambda_i) x_{ij} + \sum_{i=1}^n \lambda_i \quad (3.21)$$

$$\text{subject to} \quad \sum_{j \in I} x_{jj} = p, \quad (3.22)$$

$$x_{ij} \leq x_{jj} \quad \forall i \in I, \forall j \in I, \quad (3.23)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in I, \quad (3.24)$$

where I is the set of customers. By solving the LR problem, a lower bound to the original problem can be found. An upper bound can be found by making the lower bound solution feasible with the use of a heuristic.

If the values of the multipliers are given, the relaxed constraints should make the problem easy to solve. In Eq. (3.21), the last term will be a constant and can be ignored. To minimise the first term, the x_{ij} values should be set to one if $(d_{ij} - \lambda_i) < 0$. The solution procedure is:

Calculate $V_j = \sum_{i=1}^n \min(0, d_{ij} - \lambda_i)$. Rank the values of V_j from smallest to largest and set corresponding $x_{jj} = 1$ for the first p smallest V_j values. This will give the minimum objective function value for a set of λ_i .

To narrow the gap between the upper and lower bounds, the Lagrange multipliers are changed and the new LR is solved. This can in turn be made feasible again with the chosen heuristic and replaces the current upper bound if it is better.

Different methods can be used to change the multipliers, the best-known method is the subgradient optimisation method. The formula for updating the multipliers is:

$$\lambda_i^{t+1} = \max \left\{ 0, \lambda_i^t - T^t \left(\sum_{j=1}^n x_{ij}^t - 1 \right) \right\} \quad \forall i \in I, \quad (3.25)$$

where

$$T^t = \Delta \left[\frac{\bar{Z} - Z_L^t}{\sum_{i=1}^n \left(\sum_{j=1}^n x_{ij}^t - 1 \right)^2} \right], \quad (3.26)$$

and t is the index number of the current iteration, $\bar{Z}$ is the best feasible solution

and Z_L^t is the value of the current Lagrangian iteration.

The method is repeated iteratively until the gap between the upper and lower bound does not differ with more than a predetermined percentage.

Mulvey and Crowder (1979), and Mulvey and Beck (1984) mention that the number of decision variables is a function of n^2 for the CpMP. For the CFLP, where the number of customers n , is not equal to the number of candidate depots m , this guideline applies to m . The authors therefore recommend not using the method for $m \geq 100$. See also Section 3.1.2.9, for an example of Lagrangian relaxation used together with the ant colony system metaheuristic to solve the CFLP.

3.1.2.6 Aggregation

Mladenović et al. (2007) discuss using aggregation, where customer points are clustered together based on a commonality, as a method to make the problem smaller. According to the authors the goal is to reduce the computation time involved. However they continue to state that the method also introduces errors due to the replacement of the aggregated points. In the end they conclude that the complexity of the problem depends more on the number of depots m than the number of customers n and doubt whether the aggregation method will pay-off, since computers can now store very large datasets easily. Using aggregation might still be useful for the p-median, where the list of customers is equal to the list of candidate depots with $m = n$. Aggregation is also discussed in Drezner and Hamacher (2001, Ch. 7).

3.1.2.7 Perturbation heuristic methods

Different combinations of the above methods have also been proposed in the literature and are called hybrid heuristics. Salhi (1997) suggested for example the perturbation heuristic, where the greedy and greedy-drop methods are executed iteratively, each for a given number of steps. The method is described as follows:

Allow the number of depots p , solved by a heuristic method to change by a value of q , with

$$q = \max(2, \left\lceil \frac{p}{3} \right\rceil), \quad (3.27)$$

where $\lceil x \rceil$ is the smallest integer value larger than or equal to x .

An initial solution is created by some heuristic method. Salhi (1997) suggest using the Lagrangian relaxation method. The solution is then changed by removing q depots one at a time from a solution with p open depots, using the greedy-drop method. A total of q depots are then added again one by one using the greedy

method.

This is followed by the interchange method (called a swap method), where one open depot is replaced with another candidate depot one at a time, until no further improvements can be made. The steps are then repeated by adding and removing q depots, followed by the interchange method to check for a local optimum.

After a given number of iterations, or when the solution set did not change, another perturbation is introduced. In this modification, the method removes the q most frequently used depots. The method then uses the greedy method to add q depots again, where the depots just removed are allowed back in the solution.

They suggest allowing the method to run for $5n$ iterations, where n is the number of customers. With every iteration a different solution for opening p depots is provided. The method can also be adapted for the FLP, by making p the number of open depots in the current best solution (which they refer to as $\hat{p}$). This can then be updated at any moment when a better objective function is found.

3.1.2.8 The multiple ant colony system (MACS) metaheuristic

The MACS is described by Chen and Ting (2008) to solve the CFLP. Solutions are constructed with the ant colony system, also referred to as ant colony optimisation (ACO). Two construction rules are used; one to select the depot locations and another to assign customers to depots. Both phases make use of ACO to find good solutions. In order to construct solutions for N iterations, b number of ants are used.

The method makes use of the concept of *pheromones*, which are chemical substances that the ants leave behind for other ants to detect food. At the end of an iteration, the best solution is improved with two local search procedures. The first improvement is called an insert move. A customer is moved to another depot, open or closed, that will cause the best cost savings for the objective function. If a customer is moved to a closed depot, the addition fixed cost charges should be added to the costs incurred. The most likely scenario where a closed depot will be opened, is where the fixed costs are lower than the penalty costs in the objective function. The second improvement is a swap move. Here two customers, assigned to two different depots are swapped if the costs are beneficial. This type of move excludes closed depots.

At the end of each iteration, the global best solution and global pheromone parameters are updated. Chen and Ting (2008) state that the number of ants (b) to use per iteration influences the quality of solutions significantly and recommend making $b = 50$ for large scale problems with 100 depots and 1 000 customers. Unfortunately,

they do not mention how many iterations were used.

3.1.2.9 The Lagrangian heuristic – ant colony system hybrid (LH–ACS) metaheuristic

This method uses a LR heuristic to select the depots to open, followed by the ant colony system heuristic to assign customers to the depots. Similar to the MACS method, it was also developed by Chen and Ting (2008). The LR heuristic is used to find a good lower bound to the relaxed problem. This establishes which depots to open.

For the customer assignments, an ant colony search, similar to the MACS method is used to find initial solutions with b ants. The best solution for the iteration is then further improved with the local search procedures from the MACS method. This is used as the upper bound to the problem. The current upper and lower bounds of the problem are then used to update the Lagrangian multipliers. This is used as input for the next LR heuristic to determine depots. Similar to the MACS method, the method is also run for a predefined maximum of N iterations. Chen and Ting (2008) state that the LH–ACS method is more powerful and robust than the MACS method.

3.1.2.10 Primal heuristic method

This method was introduced by Mulvey and Beck (1984) to solve the CCP and is similar to the PAM method described in Section 2.2.3. It uses the most central depots as cluster centres. The method starts by selecting p random depots as the cluster centres. Customer points are ordered according to a regret value. The regret value is the difference in the objective function value between assigning the point to its closest depot and the second closest depot. Customers are assigned to clusters in decreasing order of regret, without exceeding a cluster's capacity constraint.

After all points have been assigned, the method calculates the most central depot from the candidate set for each cluster. The cycle is repeated if a new depot has been selected and the objective function decreases by more than a predefined percentage. To prevent an infinite loop, the cycles stop after a predefined number of iterations have been completed.

To improve the solution, all pairwise interchanges are examined to swap chosen depots with non chosen depots. If the objective function improves without violating the capacity constraints, the improvement is made. After all possible improvements have been made, the solution is recorded. The complete process is repeated a predefined number of times, each time starting with a different set of random depots

and the solution with the best objective function is returned.

3.1.2.11 The hybrid heuristic–subgradient method to solve the CCP

The hybrid heuristic–subgradient method uses the primal heuristic discussed above to find an upper bound and the Lagrangian relaxation (LR) method to find a lower bound for the problem, (Mulvey and Beck, 1984). The primal heuristic is used to search for the best feasible solution found so far. It is therefore known that the optimal solution cannot be bigger than this value, hence it is called an upper bound.

The LR method provides the optimal solution for the problem where a constraint set is ignored. The optimal feasible solution can at best be equal to this solution if all constraints are met, hence it is called the lower bound. Constraints (3.8) are removed from the model and the objective function is replaced with Eq. (3.21). The primal heuristic and subgradient methods are used iteratively, each time the Lagrange multipliers u_i are changed according to the subgradient optimisation formula, given in Eq. (3.25).

A feasible solution is then calculated for each subgradient solution by replacing the random start of the primal heuristic method with the open depots set from the LR solution. The process continues until the upper and lower bounds are within a predefined percentage of each other, after which the best feasible solution is returned.

Geetha et al. (2009) make three suggestions with regards to solving the CCP:

1. In the case of the CpMP, base the initial p medians on the points with the highest demand (w_i).
2. Assign the points in descending order of demand; assigning the point with the biggest demand first.
3. As an alternative, assign the points in descending order according to $r = \text{demand} / \text{distance}$.

3.1.3 The continuous CFLP

Two capacitated continuous CFLP problems are discussed, the CCCP (capacitated centred clustering problem) and the capacitated MSWP (multisource Weber problem). Unlike the CCCP, customers are not limited to single-source depot assignments in the capacitated MSWP. The capacitated MSWP also has a fixed number of depots to open and a constant capacity is assumed for all depots, whereas in the CCCP this is not necessarily the case.

3.1.3.1 The capacitated multi-source Weber problem (MSWP)

The capacitated MSWP is the continuous variant of the CpMP, because the locations of the depots are not limited to fixed points. Similar to the CpMP, the number of depots, p to select is known beforehand. Different from the CpMP the depot supply capacities W_j are replaced with a fixed depot capacity, W . If the transportation cost is a linear function of the distances between points, the problem can be formulated as follows (Zainuddin and Salhi, 2007):

$$\text{Minimise} \quad \sum_{j \in J} \sum_{i \in I} d(\mathbf{X}_j, \mathbf{a}_i) x_{ij} \quad (3.28)$$

$$\text{subject to} \quad \sum_{i \in I} x_{ij} \leq W \quad \forall j \in J, \quad (3.29)$$

$$\sum_{j \in J} x_{ij} = w_i \quad \forall i \in I, \quad (3.30)$$

$$x_{ij} \geq 0 \quad \forall i \in I, \forall j \in J, \quad (3.31)$$

$$\mathbf{X}_j \in \mathbb{R}^2 \quad \forall j \in J, \quad (3.32)$$

where I is the set of customers, J is the set of clusters with $|J| = p$ and $\mathbf{a}_i = (a_i^1, a_i^2)$ is the coordinates of customer i with $\mathbf{a}_i \in \mathbb{R}^2$. The coordinates of depot j are represented by the additional decision variables $\mathbf{X}_j = (X_j^1, X_j^2)$ with $\mathbf{X}_j \in \mathbb{R}^2$. These coordinates need to be determined in such a way that the distances, $d(\mathbf{X}_j, \mathbf{a}_i)$, associated with the transportation costs, are minimised. Normally Euclidean distances, calculated with Eq. (2.33) are assumed and the distance can also be written as the Euclidean norm: $d(\mathbf{X}_j, \mathbf{a}_i) = \|\mathbf{X}_j - \mathbf{a}_i\|_2$ (Brimberg et al., 2000). The decision variable x_{ij} is the amount to be delivered from depot j to customer i .

Constraints (3.29) ensure the constant depot capacity is adhered to while constraints (3.30) ensure all customer demands are met. Constraints (3.31) limit the decision variables to positive values, while Constraints (3.32) limits the depot locations to the Euclidean plane.

3.1.3.2 The capacitated centred clustering problem (CCCP)

The CCCP is similar to the CCP but unlike the CCP, the CCCP does not have a list of potential cluster centres (or depots) because the problem is a continuous FLP. The centre of a cluster, is called the depot of the cluster and is calculated as the point with the minimum total distance from all points assigned to the cluster and itself. It can be located anywhere within the Euclidean space where the points

3.1 The facility location problem (FLP)

are located. According to Negreiros and Palhano (2006), the CCCP is similar to the normal clustering problem of assigning n points into p clusters, but it assumes that each cluster has its own capacity constraint, making it an unique problem.

Negreiros and Palhano (2006) define two different versions of the CCCP: the p-CCCP and the generic CCCP. In the p-CCCP the number of clusters is predefined and the problem can be formulated as follows:

$$\text{Minimise} \quad \sum_{j \in J} \sum_{i \in I} d(\mathbf{X}_j, \mathbf{a}_i) x_{ij} \quad (3.33)$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I, \quad (3.34)$$

$$\sum_{i \in I} x_{ij} = n_j \quad \forall j \in J, \quad (3.35)$$

$$\sum_{i \in I} \mathbf{a}_i x_{ij} = n_j \mathbf{X}_j \quad \forall j \in J, \quad (3.36)$$

$$\sum_{i \in I} w_i x_{ij} \leq W_j y_j \quad \forall j \in J, \quad (3.37)$$

$$\mathbf{X}_j \in \mathbb{R}^2, \quad n_j \leq n, \quad x_{ij} \in \{0, 1\}, \quad \forall i \in I, \quad \forall j \in J, \quad (3.38)$$

where I is the set of customers, J is the set of clusters with $|J| = p$. $\mathbf{a}_i = (a_i^1, a_i^2)$ is the Euclidean coordinates of point i and the means vector of the points in cluster j is $\mathbf{X}_j$. The number of customers assigned to cluster j is n_j and W_j is the capacity constraint of cluster j .

The problem is single-source and because x_{ij} is binary, constraints (3.34) ensure each point is assigned to only one centre, while constraints (3.35) restrict the amount of points in each cluster to n_j . Constraints (3.36) locate the depots as the means vector $\mathbf{X}_j$, while constraints (3.37) enforce capacity constraints. The depots can be placed anywhere on the Euclidean plane while the assignment variables must be binary, enforced by Constraints (3.38).

In the generic CCCP, the objective is to minimise the costs associated with the dissimilarity between points and depots and the depot opening costs, f_j , given the cluster capacity, W_j per depot j . This is similar to the objective function of the fixed charge CFLP. It allows for different depot capacity constraints and depot opening costs and assumes the transportation costs are the same as the squared Euclidean distances in the objective function. The generic CCCP is formulated by Negreiros and Palhano (2006) as follows:

$$\text{Minimise} \quad f_j z_j + \sum_{j \in J} z_j \left(\sum_{i \in I} d(\mathbf{X}_j, \mathbf{a}_i) x_{ij} \right) \quad (3.39)$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I, \quad (3.40)$$

$$\sum_{i \in I} \mathbf{a}_i x_{ij} = \mathbf{X}_j \left(\sum_{i \in I} x_{ij} \right) \quad \forall j \in J, \quad (3.41)$$

$$\sum_{i \in I} w_i x_{ij} \leq W_j z_j \quad \forall j \in J, \quad (3.42)$$

$$\mathbf{X}_j \in \mathbb{R}^2, \quad z_j, x_{ij} \in \{0, 1\}, \quad \forall i \in I, \forall j \in J, \quad (3.43)$$

where z_j and x_{ij} are the decision variables associated with opening depot j and assigning point i to depot j respectively, I is the set of customers and J is the set of chosen depots. Similar to the p-CCCP, $\mathbf{a}_i = (a_i^1, a_i^2)$ is the Euclidean coordinates of point i and the means vector of the points in cluster j is $\mathbf{X}_j$.

Constraints (3.40) ensure each point is assigned to one and only one depot, while Constraints (3.41) are used to calculate the coordinates of the depots. Constraints (3.42) enforce the depot capacity constraints and constraints (3.43) limit the decision variables.

In order to lower the travelling costs, the best solution will be to create as many depot clusters as possible. To let the model choose to open less clusters, there must be an incentive. This is achieved by restricting the capacity per cluster W_j and by associating a cost f_j to the opening of cluster j in the generic CCCP.

A similar problem is called the transportation problem (TP). The TP can be described as follows: given a set of depots with given supply capacity and associated transportation costs, assign a set of customers each with a given demand to minimise the total transportation cost. The TP can be solved optimally in polynomial time. See Winston (2004, Ch. 7) for an example of how to solve the TP optimally using the simplex method. The difference between the TP and the CCCP problem is that the single-source constraints are not enforced in the TP. Unlike the capacitated MSWP and CCCP, the depot locations are already known and all depots are assumed open in the TP, implying that customers can be assigned to any depot without an additional depot opening cost charge.

3.1.4 Solution approaches for the continuous CFLP

A couple of solution approaches to solve the continuous CFLP are discussed next. Two methods are given to solve the capacitated MSWP as well as two methods to solve the CCCP. The methods are as follows:

3.1.4.1 A perturbation-based heuristic

Zainuddin and Salhi (2007) suggest an approach called the perturbation-based heuristic to solve the capacitated MSWP. The method is based on Cooper's alternating transportation-location heuristic (ATL). The ATL starts by selecting p random customer points as the cluster centres or depots of the starting solution. The TP of this solution is solved assigning customers to each depot based on the depot capacity constraint. Because the problem is multi-source, a customer's demand can be met by more than one depot. New cluster centres are now calculated using Weiszfeld's algorithm. For each cluster, this algorithm iteratively solves the Weiszfeld equation until the change in coordinates is significantly small. The Weiszfeld equation is given as follows:

$$\mathbf{X}_{j_{t+1}} = \left(\frac{\sum_{i \in I} \frac{w_i a_i^1}{d(\mathbf{a}_i, \mathbf{X}_{j_t})}}{\sum_{i \in I} \frac{w_i}{d(\mathbf{a}_i, \mathbf{X}_{j_t})}}, \frac{\sum_{i \in I} \frac{w_i a_i^2}{d(\mathbf{a}_i, \mathbf{X}_{j_t})}}{\sum_{i \in I} \frac{w_i}{d(\mathbf{a}_i, \mathbf{X}_{j_t})}} \right), \quad (3.44)$$

where w_i is the demand of customer i allocated to depot j . The coordinates of customer i are $\mathbf{a}_i = (a_i^1, a_i^2)$ with $\mathbf{a}_i \in \mathbb{R}^2$. The coordinates of depot j at iteration t are $\mathbf{X}_{j_t} = (X_{j_t}^1, X_{j_t}^2)$ with $\mathbf{X}_{j_t} \in \mathbb{R}^2$. The distance between customer i and depot j at iteration t is $d(\mathbf{a}_i, \mathbf{X}_{j_t})$. Initial coordinates for the depot, $\mathbf{X}_{j_0} = (X_{j_0}^1, X_{j_0}^2)$, are needed as input variables.

The ATL heuristic continues to alternate between solving the TP and using Weiszfeld's algorithm to calculate depot centres until the improvement in cost is below a pre-determined threshold value.

Instead of using random starting points, Zainuddin and Salhi (2007) introduce the furthest distance rule (FDR) to select depots. The first depot is still selected randomly, from there onwards the method determines the customer point with the furthest distance from all depots in the current depot set and adds this customer to the depot set. The method continues in this manner until p depots have been selected.

Zainuddin and Salhi (2007) propose a perturbation-based method that first solves the uncapacitated problem for different K starting solutions. A subset of the K solutions

that differ a substantial enough cost amount from each other are selected to solve the capacitated problem. The ATL heuristic is used to solve the capacitated problem, but two extra steps are added to this heuristic. Here all customers are assigned to their closest depot after Weiszfeld's algorithm has been used to determine the depot centres. The demand of the customers are therefore moved to a single depot and Weiszfeld's algorithm is again used to determine the depot centres.

A post optimisation search is used to find better solutions by clustering "borderline" customers and assigning them to their closest depot. A customer is seen as a borderline customer if the distance from the customer to the closest and second closest depots are almost the same. In these cases the ratio of the closest to second closest distances will be very small. A predefined threshold ratio parameter can be used to identify them. These customers are excluded from the TP if their demand cannot fully be met by the nearest depot due to capacity constraints. After the TP has been solved the customers are introduced back into the method. The reason behind this removal is to move depot centres further away from each other in order to reduce the number of borderline customers.

Zainuddin and Salhi (2007) also note that solving the TP for the different sets of depots can become quite time consuming. They suggest using a subset of depots at a time. Forbidden regions are also used to prevent starting solutions from selecting previously chosen depots. The authors also mention that selecting the best uncapacitated starting solution does not necessarily result in the optimal capacitated solution, which is why they use multiple starting solutions.

3.1.4.2 A region-rejection based heuristic

This method was suggested by Luis et al. (2009) to solve the capacitated MSWP. The heuristic forbids new depots from being placed too close to previously selected depots by placing a forbidden radius around these depots. The radius is then adjusted dynamically. The heuristic generates initial starting solutions which are then solved using the ATL heuristic discussed above.

A similar approach is followed by the same authors in Luis et al. (2011). They make use of restricted regions when constructing a restricted candidate list in the guided reactive GRASP metaheuristic. The GRASP method is discussed in more detail in Section 3.3.6.2. The construction of the restricted candidate list (RCL) is guided with a parameter, α and the search is enhanced with a learning process to define the bounds of α . The restricted regions are also used to prevent new solutions from being generated too close to already tested ones.

3.1.4.3 The cluster search GA metaheuristic to solve the CCCP

Chaves and Lorena (2010) suggest a cluster search algorithm to solve the CCCP. This algorithm consists of four phases: The first phase generates random CCCP solutions. The second phase groups solutions together based on similarity and the cluster centres and number of similar solutions found are updated. The third phase analyses the information gathered about the grouped solutions. If a certain grouped solution has been identified in a predefined number of solutions, this is identified as a candidate solution to explore further. In the fourth phase, a local search heuristic is used to explore the candidate solution.

Chaves and Lorena (2011) refined the cluster search heuristic to use a genetic algorithm (GA) to generate solutions for the CpMP. Their reasoning behind this is that it is less time consuming to evaluate the CpMP objective function than the CCCP objective function. These solutions are then used in phase one to identify possible candidate solutions. If a candidate solution is detected, it is then evaluated with the CCCP objective function in the local search algorithm.

An additional parameter, called the inefficiency rate, is introduced in the second phase. The parameter keeps track of the number of consecutive times the local search algorithm could not improve the objective function of a candidate solution. This parameter prevents the heuristic from repeating a local search in an area that has already proven inefficient.

3.1.4.4 A two-phase heuristic method to solve the CCCP

Negreiros and Palhano (2006) suggest a two-phase heuristic method to solve the CCCP, the first phase is a constructive phase and the second is a refinement phase using a variable neighbourhood search (VNS) as follows.

Constructive phase

Two methods can be used in the constructive phase depending on the problem. The log-polynomial geometric tree search method was used to solve the generic CCCP. The unconstrained to constrained method was used in conjunction with an unconstrained solution to solve the p-CCCP.

- **The log-polynomial geometric tree search method**

This method constructs clusters from subtrees of the minimum spanning tree (MST) graph. The MST is redrawn to balance points evenly throughout the graph, from a centre point. This graph is referred to as a balanced q-tree. It is unconstrained, since there are no limits on the capacities of the subtrees. The only focus is to balance the demand equally.

Once the unconstrained balanced q-tree has been constructed, a subtree is chosen and if the demand of the customers in the subtree is less than the capacity of a cluster, the whole subtree is assigned to one cluster. The method then looks at the next subtree, to see if it can add it to the created cluster or if it is too big, assign it to a new cluster or find the closest cluster centre with capacity. This is repeated until all the points have been assigned to clusters.

The above tree search methods can be used for the generic CCCP where the number of clusters is unknown beforehand.

- **The unconstrained to constrained method**

For the p-CCCP, an unconstrained to constrained method was suggested. The method can be described as follows:

- 1) An initial unconstrained solution for p clusters is created with either the h-means or j-means methods.
- 2) The points in the clusters are then ordered by decreasing demand and infeasible points are moved to feasible clusters. The ordering can also be based on distance or the demand / distance ratio as discussed in the CCP solution approach above.
- 3) If no feasible cluster could be found, the point is moved to a new cluster and $p = p + 1$. (No mention is made about the merging of clusters to bring p back to its original value.)
- 4) Once a capacitated feasible initial solution has been created, either the h-means or the j-means methods can be called to better the solution. Here the methods are restricted and only feasible exchanges to other clusters are allowed.

Refinement phase

The refinement phase is based on a Reduced Variable Neighbourhood Search (RVNS), (Burke and Kendall, 2005). The refinement phase is an iterative process with the following steps:

1. A fixed number of points in a local neighbourhood are selected randomly to exchange between clusters.
2. The points can either be exchanged with their closest centres found in the constructive phase or a new set of random clusters are selected to exchange the points with. The randomly chosen clusters do not have to be distinct.
3. Once the exchanges have been made, the objective function is evaluated.
4. If the objective function has improved, the solution is updated to reflect the new assignments to clusters. If not, the solution defaults back to the original solution.

3.2 The vehicle routing problem (VRP)

Unlike the VNS method, the RVNS does not refine the solution further by looking for more improvements in the area of the preferred exchange, but rather start afresh with a new random selection. According to Burke and Kendall (2005), this makes the RVNS method more suitable for large-scale problem instances.

The iterative process is repeated until a fixed amount of time has expired. According to Negreiros and Palhano (2006), using a fixed number of points and clusters makes the method $O(1)$ time consuming. The method is a type of Monte Carlo method, integrated with a neighbourhood search algorithm.

3.2 The vehicle routing problem (VRP)

The routing component of the LRP is another well-known problem in literature, called the vehicle routing problem (VRP). The VRP has been a popular research topic and is known to be NP-hard, (Simchi-Levi and Bramel, 1997, Chapter 6; Joubert, 2006). The main goal of the basic VRP is to minimise distribution costs, given a single depot or facility, a fleet of homogeneous vehicles and a set of customers, each with an associated demand.

The VRP can also be seen as an extension of the travelling salesman problem (TSP). In the TSP, a salesman seeks the shortest closed path that visits every node in a graph exactly once and returns to the point he started from. The TSP is known to be NP-hard, (Papadimitriou and Steiglitz, 1998, p 3). The multiple travelling salesman problem (mTSP) allows for multiple paths, all beginning and ending at the same point. See Bektas (2006) for more details about the mTSP. The mTSP is seen as a relaxation of the VRP, (Laporte, 1992b, p 346) and the same methods are often used to solve both problems.

3.2.1 The CVRP

In the VRP, vehicles may have capacity limits and these problems are referred to as capacitated vehicle routing problems (CVRP). The CVRP has a set of K vehicles each with a capacity constraint Q for a homogeneous fleet, or a variable Q_k associated with vehicle k for a heterogeneous fleet. Each vehicle is allowed one route and must start and end at the depot. Each customer has an associated demand w_i and should only be visited once.

Problem formulation

The CVRP can be defined as a weighted, directed graph $G = (V, A)$, where V includes the single depot at node 0 and the set I of $i = 1$ to n customers to visit. The arc $(i, j) \in A$, is given a weight, c_{ij} , representing the cost (or distance) associated

3.2 The vehicle routing problem (VRP)

with travelling from node i to node j . The CVRP is said to be symmetrical if $c_{ij} = c_{ji}$.

Different variations of the CVRP formulation have been provided. In order to keep the formulation between the CVRP and the LRP similar, the three-index vehicle flow formulation will be used, (Laporte, 1992b; Joubert, 2006). Here the binary decision variable $x_{ijk} = 1$ if the arc (i, j) is part of the route of vehicle k . If S is a subset of nodes with $S \subseteq I$ then $|S|$ is the number of nodes in the subset.

Based on the formulations from Laporte (1992b, p 353), the single depot CVRP can be formulated as follows:

$$\text{Minimise} \quad \sum_{\substack{i,j \in V \\ i \neq j}} \sum_{k \in K} c_{ij} x_{ijk} \quad (3.45)$$

$$\text{subject to} \quad \sum_{j \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in I, \quad (3.46)$$

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall j \in I, \quad (3.47)$$

$$\sum_{i \in I} \sum_{j \in V} w_i x_{ijk} \leq Q \quad \forall k \in K, \quad (3.48)$$

$$\sum_{i \in V} x_{ijk} - \sum_{i \in V} x_{jik} = 0 \quad \forall j \in V, \forall k \in K, \quad (3.49)$$

$$\sum_{j \in I} x_{0jk} \leq 1 \quad \forall k \in K, \quad (3.50)$$

$$\sum_{i \in I} x_{i0k} \leq 1 \quad \forall k \in K, \quad (3.51)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1 \quad \forall S \subseteq I, |S| \geq 2, \forall k \in K, \quad (3.52)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K. \quad (3.53)$$

The objective function is to minimise the distribution costs associated with the routes of the $|K|$ vehicles. Constraints (3.46) and (3.47) ensure that there is only one delivery per customer and that every customer is assigned to only one route.

Constraints (3.48) ensure that no route overloads a vehicle. Constraints (3.49) guar-

3.2 The vehicle routing problem (VRP)

antee continuity of the routes. Constraints (3.50) and (3.51) ensure that each route starts and ends at the depot and that every vehicle has only one route. Constraints (3.52) are used to avoid cycles or subtours, by ensuring that the number of arcs between customers on a route is one less than the number of customers. Constraints (3.53) enforce binary decision variables. A time constraint can also be placed on the duration of routes, but has been left out since it is irrelevant to the routing phase of the LRP.

The constraint sets (3.47) and (3.51) are automatically implied in Constraints (3.46) and Constraints (3.50) through the continuity of routes enforced by Constraints (3.49) and are sometimes left out of the problem formulation.

There are many variations of the VRP that address different constraints and limitations. For example the VRP with time windows (VRPTW) restricts the time when a vehicle may visit a customer. The distance VRP (DVRP) restricts the maximum travel distance of the routes by adding Constraints (3.54) to the above problem formulation, where D is the maximum distance allowed,

$$\sum_{j \in V} \sum_{i \in V} c_{ij} x_{ijk} \leq D \quad \forall k \in K. \quad (3.54)$$

Laporte (1992b) also mentions that the DVRP can be used to restrict time. Other restrictions that originated from real-world case studies are listed by Poot et al. (1999) and Kant et al. (2008). The periodic vehicle routing problem (PVRP) is discussed in Section 3.2.3.

3.2.2 Solution approaches for the CVRP

Various exact, classic heuristics and metaheuristic methods concerning the CVRP, as formulated in Eq. (3.45) – (3.53), have been suggested and there is a growing amount of literature on the subject. The methods can also be used to solve the routing phase of the CLRP. In the CLRP, the routing phase is used to estimate distribution costs that will, together with the depot costs from the location phase, give an indication of how many depots are to be used. Some exact methods to solve the CVRP are discussed first. After this, heuristic and metaheuristic methods that can find good estimate distribution costs quickly, are discussed.

3.2.2.1 Exact methods

1. Branch-and-bound methods

Many exact methods mentioned in the FLP have been considered. The best-known methods include branch-and-bound methods, where the constraints in

Eq. (3.52) are normally relaxed and subtours are allowed, (Laporte, 1992b). If the solution to the relaxed problem contains a subtour, the problem branches into two subproblems. Each subproblem excludes one of the arcs in the subtour.

2. Dynamic programming

Dynamic programming can also be used to solve the VRP exactly. This is described in Laporte (1992b) and Papadimitriou and Steiglitz (1998, p 450). Start by calculating the costs to travel from the depot to any given one customer by listing the costs from the depot to all customers. Then add a second customer to the route and calculate the minimum cost per possible customer combination by adding to the costs listed before. Continue to add customers in this manner, every time adding the customer demand until the vehicle capacity is reached. The customers already in the route are defined by the subset U and the minimum cost is determined for any given subset U . Each time an extra customer is added to the stage, the best solution found from the previous stage for that specific subset of customers is used.

To continue solving the CVRP, calculate all the cost possibilities of adding another route, also using the dynamic programming technique of the TSP. Iteratively continue to add routes until $|K|$ vehicles are added, each time ensuring that the subset of customers to add exclude the ones already on routes for that specific state. It is clear that the number of computations required to solve even small sized VRPs can be excessive. The dynamic programming method is therefore not often used to solve VRPs.

3. Column generation and set partitioning

Foster and Ryan (1976), and later Laporte (1992b) discuss a column generation and set partitioning approach to solve the VRP. Campbell and Wilson (2014) reported that using this method to solve the VRP is slow to converge, therefore making it unattractive for larger datasets. The method has been described on page 46 for the FLP. For the VRP, the method remains the same, here customers on the same route are used to represent the columns. Simchi-Levi and Bramel (1997, Chapter 8) give a discussion on the analogies between the FLP and VRP when using the column generation method.

Fisher and Jaikumar (1981) and Cordeau et al. (2007) divided heuristic methods into four main categories: incomplete optimisation methods, tour building, tour improvement and two-phase methods. These heuristic methods are often referred to as "classic heuristics" in the literature. These methods are discussed next.

3.2.2.2 Incomplete optimisation methods

The incomplete optimisation methods refer to exact methods that are stopped before running to completion. Simchi-Levi and Bramel (1997) state that the biggest disadvantage with incomplete optimisation methods is that they still require large processing times and they do not recommend using the methods for problems with more than 100 customers.

3.2.2.3 Tour building methods

Tour building or tour construction methods construct routes by sequentially adding links between customers. Two types of tour building heuristics exist: parallel methods and sequential methods. Parallel methods build all the routes at the same time, while sequential methods build only one route at a time before moving to the next, (Cordeau et al., 2007).

1. Savings based or Clarke and Wright algorithm (CWA)

The most popular tour building heuristic is called the savings based method. It is also sometimes referred to as the Clarke and Wright algorithm (CWA), since it was first introduced by Clarke and Wright (1964). The savings based method starts by placing each customer on a single route and then merges routes together based on the biggest savings found.

The cost saving is calculated as the costs for adding an arc between two customers, minus the costs of the arcs from and to the depot that can be removed by the merge. Prins et al. (2006) state that the Clarke and Wright algorithm is most effective when used together with an improvement method as described below. Laporte et al. (2000) compare different variants of the savings based method. They recommend using the parallel version with a 3-opt improvement method. The 3-opt method involves neighbourhood moves where 3 arcs are exchanged.

The savings based method was also used by Prins et al. (2006) in the Extended Clarke and Wright algorithm (ECWA) to solve the LRP, described in Section 3.3.6.1.

2. Nearest neighbour tour building method

The nearest neighbour method also starts with single customer routes and then merges customers based on the nearest distance to one another, similar to the single linkage method. It is described by Laporte (1992b).

3. Insertion tour building method

The insertion method (Cordeau et al., 2007) starts with $|K|$ single customer routes and then adds unassigned customers one by one, based on a cost savings

calculation. If customers i and j are already on a route, then the customer k with the highest cost saving: $c_{ik} + c_{kj} - \lambda c_{ij}$, is to be added, where λ is a predefined parameter.

Cordeau et al. (2007) continue to describe a more effective insertion heuristic in detail. Here the insertion heuristic is used to create routes as above. A customer is then selected per route, which is used in the parallel insertion method. Instead of using a cost savings calculation directly to determine the next insertion, they suggest using a regret calculation based on the difference between the savings of best and the second-best insertion. This is followed by a 3-opt improvement method before continuing with the next insertion.

The tour construction methods are generally followed up by tour improvement methods to better the results, (Cordeau et al., 2007).

3.2.2.4 Tour improvement methods

A tour improvement method starts with a constructed tour and tries to improve it by exchanging a combination of arcs. If the exchange is feasible and provides a cost saving, the exchange is made, (Fisher and Jaikumar, 1981).

1. k-opt tour improvement method

The most popular tour improvement method is called the k-opt. It was first suggested in 1965 for the TSP, where k represents the number of arcs or links to be exchanged. The method was later improved to allow k to change dynamically, (Laporte et al., 2000).

In order to save time, it has been suggested that the heuristic only search for the first improvement and then change the routes accordingly. Laporte et al. (2000) suggest that using the best move rather than the first improvement found, has empirically given better results.

2. Or's algorithm

The Or's algorithm is a tour improvement method described by Laporte (1992a) for the TSP as follows:

- 1) Start with a created tour. Set $t = 1$ and $s = 3$.
- 2) Remove s consecutive arcs (chains) from a tour, starting with the vertex in position t . Insert the removed vertices between the other points on the tour.
- 3) If a cost savings could be found, make the exchange, set $t = 1$ and repeat Step 2.
- 4) If a cost savings could not be found, update $t = t + 1$. If $t = n + 1$, where n is the number of customers on the tour, then $t = 1$ and $s = s - 1$.

5) If $s > 0$ then repeat Step 2 else stop.

In the case of the VRP, Laporte et al. (2000) suggest that Or's algorithm can be used as a restricted 3-opt to interchange three consecutive customers to other positions, by keeping $s = 3$.

3.2.2.5 Two-phase methods

These methods divide the problem into two phases, the clustering phase and the routing phase. According to Simchi-Levi and Bramel (1997) there are two types of two-phase methods, the cluster-first, route-second and the route-first, cluster-second.

1. Fisher and Jaikumar algorithm

A popular cluster-first, route-second two-phase heuristic method was introduced by Fisher and Jaikumar (1981). The method was named after its authors as the Fisher and Jaikumar algorithm. In the clustering phase, k seed customers are selected, where k is the number of clusters or vehicles to be used. The remaining customers are then assigned to these seed customers using the generalised assignment problem (GAP). In the second phase, the routing for each cluster of customers is solved using any TSP method.

Fisher and Jaikumar (1981) provide a list of considerations that could be taken into account when selecting the seed customers. These include:

- Customers along major roads.
- The most distant customers among these major roads and thoroughfares.
- Customers with a demand bigger than half the vehicle capacity size - the logic is that no pair of these customers can be clustered together because of the vehicle capacity constraints.
- Partition the customers into k "cones" or areas running with straight-lines from the depot position in the middle at different angles. Select only one customer per cone as the seed customer.

After the customers have been clustered into routes, the route sequencing is determined. This can be done with any known TSP method. Where the number of customers are limited, exact methods can be used. Barreto et al. (2007) recommend solving the TSP problem exactly for 40 customers or less. For more customers, heuristic methods can be used. Some of the most well-known heuristic methods to solve the TSP include:

2. The tree algorithm

The tree algorithm guarantees that the distance of any tour found using this

method, will not be more than double the distance of the optimal solution, (Papadimitriou and Steiglitz, 1998, p 414). The method is described as follows:

- (a) Find the minimum spanning tree (MST), the shortest path that will connect all the vertices in the graph G , representing the VRP.
- (b) Double every edge in the MST, effectively allowing bidirectional traffic to and from every vertex in G .
- (c) Construct an Eulerian walk. This is a route through the graph G that goes through every edge exactly once.
- (d) Delete all duplicate vertices in the Eulerian walk. The remaining sequence of vertices is a solution to the TSP. This creates a Hamiltonian path - a path that visits each vertex in graph G exactly once, (Srooker, 1988).

An Eulerian walk or trial is a route through the graph G that goes through every edge exactly once. An Eulerian cycle is also an Eulerian walk that begins and ends at the same vertex, (Simchi-Levi and Bramel, 1997). Any algorithm to construct an Eulerian cycle can therefore also be used to create an Eulerian walk.

The method can therefore use Hierholzer's algorithm (Gould, 2012) to construct an Eulerian cycle in Step 2c. This algorithm has the following steps:

- (a) Choose a vertex v to start at and follow a tour until ending at v .
- (b) Find a new vertex v on the constructed tour that has an adjacent edge that is not part of the tour. Follow the unused edges and return to v . Insert this new tour into the current tour at vertex v .
- (c) Repeat the previous step until all edges form part of the tour.

Figure 3.3 illustrates the tree algorithm, (Papadimitriou and Steiglitz, 1998). Figure 3.3.a shows the MST, Figure 3.3.b shows the Eulerian spanning graph, Figure 3.3.c gives an Eulerian walk with the tour to be used underlined, Figure 3.3.d depicts the created tour. This can be compared to Figure 3.3.e, the shortest tour for this example.

3. The Christofides algorithm

The Christofides algorithm guarantees that the distance for the solutions found is at maximum $3/2$ times as much as the distance in the optimal solution, (Papadimitriou and Steiglitz, 1998, p 416–418). The algorithm is depicted in Figures 3.4.a - e. by Papadimitriou and Steiglitz (1998) and works as follows:

- (a) Find the minimum spanning tree (MST), the shortest path that will connect all the vertices in the graph G , representing the VRP.

3.2 The vehicle routing problem (VRP)

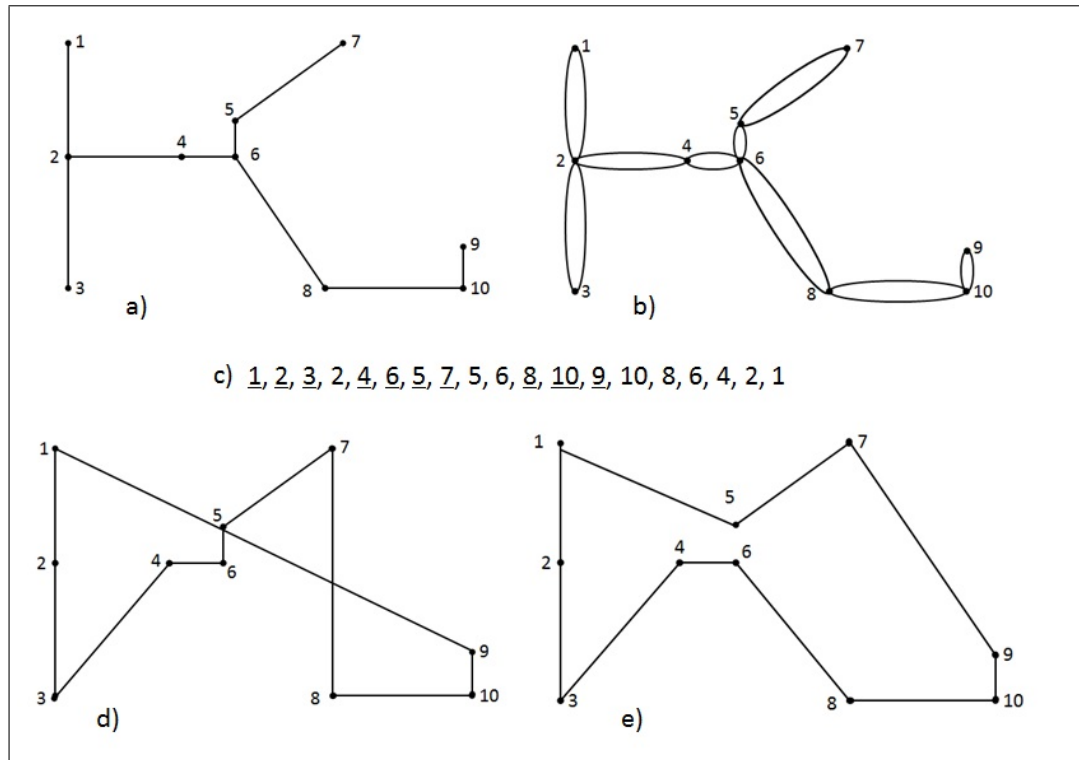


Figure 3.3: The tree algorithm as illustrated by Papadimitriou and Steiglitz (1998).

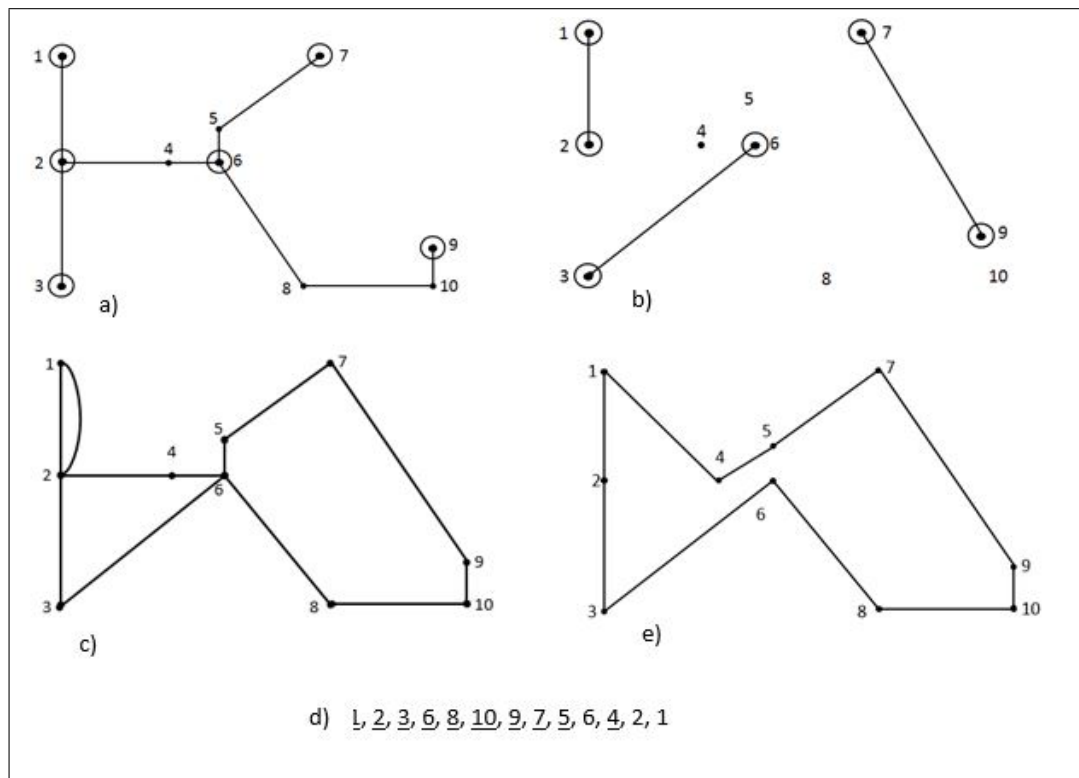


Figure 3.4: Christofides algorithm as illustrated by Papadimitriou and Steiglitz (1998).

- (b) Identify all vertices with an uneven number of edges in G , in Figure 3.4.a the nodes with uneven edges have been circled, these are 1; 2; 3; 6; 7 and 9.
- (c) Find the shortest set of edges to match all these vertices in pairs. In Figure 3.4.b edges 1-2; 3-6 and 7-9 were matched.
- (d) Add these edges to the graph G . Where an edge is in both the MST and the matching edges, the edge is doubled. This can be seen in Figure 3.4.c.
- (e) Construct an Eulerian walk, going through every edge exactly once. Figure 3.4.d gives the Eulerian walk for the example.
- (f) Delete all duplicate vertices in the Eulerian walk. The remaining sequence of vertices is a Hamiltonian path that visits each vertex once and is a solution to the TSP, (Figure 3.4.e).

4. The location based heuristic

Another cluster-first, route-second heuristic described by Simchi-Levi and Bramel (1997) is called the location based heuristic. Their approach is to reformulate the CVRP as a single-source CFLP, solve the CFLP and then solve the routing part for each cluster of customers. The distribution costs of the CVRP are divided into two parts:

- 1) The cost to travel from the depot to the suburb or area where a group of customers are located and back (round trip costs) and
- 2) the travelling cost to visit each individual customer.

The problem is reformulated as a CFLP by making the round trip costs the fixed costs for placing depots in the CFLP. The CFLP can now be solved to find the clusters of customers to route together. The authors suggest two exact methods to solve the CFLP, the cutting plane algorithm or the Lagrangian relaxation. Obviously these methods will only work for smaller case studies, but it is also possible to replace this with one of the heuristic CFLP methods described in Section 3.1 for bigger case studies. The routing costs can next be solved using a TSP method or tour construction method as described above.

5. Route-first, cluster-second algorithms

The second type of two-phase heuristics are called route-first, cluster-second methods. Simchi-Levi and Bramel (1997) describe an algorithm where a constructed tour with all customers is created first and then the customers are clustered into routes based on the sequence and vehicle capacity constraints. The authors mention that route-first, cluster-second methods are not as effective as cluster-first, route-second methods. One of the biggest problems with this approach is that varying customer demand are not taken into consideration during the initial phase and can result in bad routes in the second phase.

This problem can be overcome by using a tour building method afterwards to improve the results. They also suggest solving a TSP problem for the customers clustered together on one route instead of using the original routing sequence.

Simchi-Levi and Bramel (1997) mention another route-first, cluster-second method called the sweep algorithm, (Laporte et al., 2000, Fisher and Jaikumar, 1981, Simchi-Levi and Bramel, 1997). When depicting the customers and depot on a graph, draw a straight-line from the depot to the end of the area where the customers are located at an angle. Now start rotating the line in one direction, keeping the one endpoint at the depot. As customers are touched by the line, they are included on a route. Once a route has reached its capacity, a new route is started. The line is rotated until it has completed the 360-degree circle.

Routes can also be constructed afterwards using a TSP construction method; such as Christofides algorithm. This method is sometimes classified as a tour construction method and is known to be less effective than the savings based method.

Simchi-Levi and Bramel (1997) state explicitly that the essential part of any VRP problem is the clustering of customers and that routing is secondary to this. The authors therefore recommend rather using a cluster-first, route-second than a route-first cluster-second method.

3.2.2.6 Tabu search

Various metaheuristic methods have been implemented to solve the CVRP, see for example Joubert (2006) and Hosny (2010). One of the most successful metaheuristics is the tabu search method, described by Blum and Roli (2003). It was first introduced by Glover in 1986. (Laporte et al., 2000) give an example of how to use the tabu search method to solve the VRP.

The basic concept is to start with an initial solution and then perform a local search for better solutions. A short-term memory, called the tabu list, keeps a list of most recently visited solutions that the method is forbidden to move to. The tabu list is used to escape from local minima and prevent cycling. Every time a new solution is introduced, this solution is added to the tabu list and another solution is removed from the list, using the FIFO order. The method remembers the best found solution and stops after a predefined number of iterations or when all possible solutions are forbidden.

The size or length of the tabu list, called the tabu tenure, controls the behaviour of the search process. If the tabu tenure is small, the search will focus on smaller areas

of the search space. If the tabu tenure is large, the search is forced to cover larger areas of the search space. Normally the tabu tenure will vary during the course of the method. An effective method is to let the tenure become larger if solutions tend to repeat. This will increase diversification of the search. To intensify the search around a certain solution, the tenure can be decreased.

Normally complete solutions cannot be kept in the tabu list. Instead, the method keeps track of the attributes of the solution. In the case of the VRP, this can be the last edge change made or the route a customer was removed from as well as the route he was re-assigned to. The storing of attributes can cause the exclusion of other unvisited solutions as well. To overcome this, an aspiration criterion is set in place, allowing a move to be included even if it is in the tabu list when certain conditions are met. The most common aspiration criterion is if a solution is found that will better the current best solution found.

Other long term statistics can help to guide the method towards better solutions. These are called recency, frequency, quality and influence statistics. The *recency* statistics will remember how many iterations ago the same attribute was involved in a move. *Frequency* statistics are stored as counters. These statistics keep track of often made attributes. *Quality* statistics are used to identify good solution components. Finally the *influence* statistics keep track of the impact a certain move had on the value of the solutions in order to identify the most critical decisions made.

1. Granular tabu search

The Granular tabu search method is said to produce quality solutions quickly, (Laporte et al., 2000). The method works on the principle that large edges are generally not in good solutions. By introducing a granular threshold, the method filters out all edges larger than this threshold and only considers the rest of the edges when establishing the solutions' neighbourhood, (Toth and Vigo, 2003). The goal is to only look at 10% - 20% of all edges in the complete graph. Focussing on the area where good solutions are, makes the search space around the solution much smaller. It also intensifies the search to find good quality solutions faster.

Consider a complete graph $G(V, A)$ for the VRP, where V is the set of vertices consisting of all customers and the depot and A is the set of all arcs connecting all vertices. Toth and Vigo (2003) define a new graph $G'(V, A')$, where $|A'| \ll n^2$ and A' is the set of arcs in the graph to be considered when creating a new solution, called the sparse graph. A granular threshold φ is defined as:

$$\varphi = \beta \frac{z'}{n + l}, \quad (3.55)$$

where z' is the objective function value of the current solution, n is the number

of customers, l is the number of vehicles and β is a predefined parameter value between 1 and 2. A' can then be defined as:

$$A' = \left\{ (i, j) \in A \mid c_{ij} \leq \varphi \text{ or } c_{ij} \in \hat{I} \right\}, \quad (3.56)$$

where $\hat{I}$ is defined as the set of *other important edges* and includes the arcs in the current solution as well as all arcs to and from the depot to each customer and the arcs in the best found solution. The set A' therefore includes all directed edges (arcs) in the complete graph with a cost smaller than or equal to φ as well as all directed edges in $\hat{I}$.

The method starts by using the savings based method to create an initial solution. Since the savings based method can use more than $|K|$ vehicles, some routes can be eliminated by moving customers from the smallest demand routes to the best possible position on other routes.

This could create infeasible solutions, but these solutions are allowed - the method caters for this by adding a penalty term to the objective function. The term consists of a penalty variable α_C , multiplied by the total value that the vehicles are over capacity. If the solutions have remained feasible for a maximum of n_i iterations (n_i is recommended to be 10), then α_C is updated to $\alpha_C/2$.

If all solutions in the last n_i iterations have remained infeasible, then α_C is updated to $2 \times \alpha_C$. α_C is also defined between a given predefined minimum and maximum range. Toth and Vigo (2003) suggest the values 1 and 6400 for the minimum and maximum α values with an initial value of α of 100.

Four basic neighbourhood exchanges are considered; the 2-opt, two special cases of the 3-opt and the customer swap 4-opt. Figure 3.5 illustrates these neighbourhoods, (Toth and Vigo, 2003). π_a and σ_a represent the customers before and after customer a respectively. Figure 3.5.a shows a normal 2-opt exchange, where two new arcs replace two arcs in the current solution.

Figures 3.5.b and 3.5.c show the two types of 3-opt exchanges. The first is called a customer insertion. An arc (a, b) is picked and customer a is then inserted into the route before customer b . The second type of 3-opt is called an Or exchange, where two consecutive customers are moved together. In Figure 3.5.c, the arc (a, b) is inserted in the route after customer a , then customers b and σ_b are inserted, before the route returns to customer σ_a . Lastly, Figure 3.5.d shows a particular 4-opt exchange called the swap exchange, where two pairs of consecutive arcs are removed. Arc (a, b) is inserted into the route customer a is in and customer σ_a replaces customer b in the second route.

The four neighbourhoods have the unique characteristic that only one arc (or an edge and the direction of the edge) needs to be chosen. Once the arc is

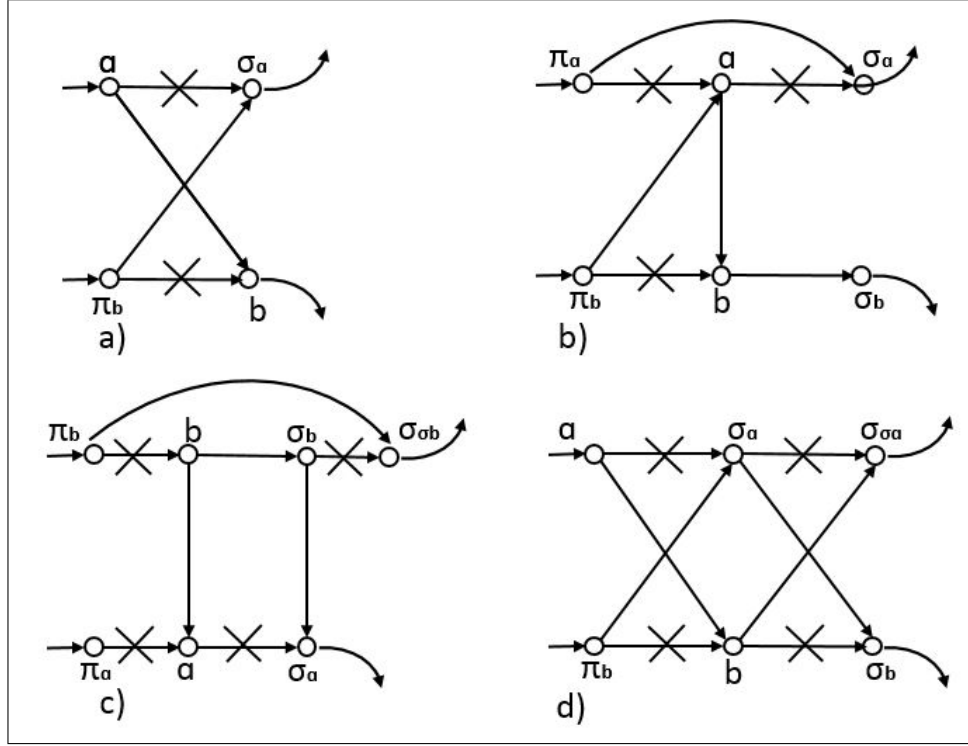


Figure 3.5: The four basic neighbourhood exchanges of the Granular Tabu Search, (Toth and Vigo, 2003).

chosen, the others are implicitly defined and do not necessarily belong to A' , the edges of the sparse graph. To decide which arc to choose, all arcs in A' in the sparse graph G' are considered and the best move is chosen.

After a move, the edges in the new solution are added to the sparse graph G' . G' is reset every $2n$ iterations. After every move, the arcs that have been removed from the new solution are added to the tabu list. These are seen as arcs forbidden to reinsert into the solution for a number of iterations called the tabu tenure. Toth and Vigo (2003) suggest selecting a random value between 5 and 10 for the tabu tenure.

If no improvements are found in n_d consecutive iterations, the method uses a diversification search to find new solutions. The method discards the current solution and continues the search with the current best found solution. The parameter β in Eq. 3.55 is increased, this increases the number of arcs to include in A' and the sparse graph G' is recalculated. The method continues using the increased value of β for n_h iterations after which β is restored to its normal value and the search continues. Toth and Vigo (2003) recommend using a normal β value of 1.25 and an increased β value of 1.75. β is increased after $n_d = 15n$ iterations if no improvements are found and returned to the normal value after $n_h = n$ iterations.

3.2.3 The PVRP

Another variant of the VRP which is of more interest when investigating the PLRP, is the periodic VRP (PVRP). The PVRP allows for multiple deliveries to the same customer multiple times over a time horizon at a given visit frequency. The periodic travelling salesman problem (pTSP) allows for only one vehicle per day. Two excellent summaries of solution approaches to solve the PVRP are given by Francis et al. (2008) and Campbell and Wilson (2014).

Problem formulation

Similar to the CVRP, the PVRP is defined as a weighted directed graph $G = (V, A)$, where V includes the single depot at node 0 and the set I of customers to visit. The arc $(i, j) \in A$, is given a weight, c_{ijkl} , representing the cost (or distance) associated with travelling from node i to node j with vehicle k on day l . It is assumed that the problem is symmetrical with $c_{ijkl} = c_{jikl}$. The PVRP can be formulated as follows, (Cordeau et al., 1997):

$$\text{Minimise } \sum_{j \in V} \sum_{i \in V} \sum_{l \in H} \sum_{k \in K} c_{ijkl} x_{ijkl} \quad (3.57)$$

$$\text{subject to } \sum_{r \in \text{Comb}_i} y_{ir} = 1 \quad \forall i \in V, \quad (3.58)$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijkl} - \sum_{r \in C_i} a_{rl} = 0 \quad \forall i \in I, \forall l \in H, \quad (3.59)$$

$$\sum_{i \in I} \sum_{j \in V} w_{ilr} x_{ijkl} \leq Q \quad \forall k \in K, \forall l \in H, \forall r \in \text{Comb}_i, \quad (3.60)$$

$$\sum_{i \in V} x_{ijkl} - \sum_{i \in V} x_{jikl} = 0 \quad \forall j \in V, \forall k \in K, \forall l \in H, \quad (3.61)$$

$$\sum_{j \in I} x_{0jkl} \leq 1 \quad \forall k \in K, \forall l \in H, \quad (3.62)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijkl} \leq |S| - 1 \quad \forall S \subseteq I, |S| \geq 2, \forall k \in K, \forall l \in H, \quad (3.63)$$

$$x_{ijkl} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K, \forall l \in H, \quad (3.64)$$

$$y_{ir} \in \{0, 1\} \quad \forall i \in V, \forall r \in \text{Comb}_i. \quad (3.65)$$

The objective function is to minimise the distribution costs over a time horizon H , consisting of l periods (normally days) with $|K|$ vehicles allowed one route per day.

3.2 The vehicle routing problem (VRP)

Each customer, i , is given a set of possible combinations of visit days, called $Comb_i$. Constraints (3.58) allow one set of visit days per customer. Constraints (3.59) ensure that every customer is only allowed one visit on the days corresponding to the chosen combination set. The visit days for the different combination sets $Comb_i$ are represented by $a_{rl} = 1$ if day l is included in a combination r , otherwise the value is zero.

The demand for a customer i can now be defined as w_{ilr} , where l specifies the day of the visit and r specifies the combination of visit days from $Comb_i$ that is used. Constraints (3.60) ensure that the routes adhere to the vehicle capacity constraints. Constraints (3.61) guarantee continuity of the routes. Constraints (3.62) ensure that each route ends at the depot and the vehicles are used only once per day.

Constraints (3.63) are used to avoid cycles or subtours, similar to the CVRP. Constraints (3.64) and (3.65) enforce binary decision variables. A time constraint or distance constraint, similar to Constraints (3.54) can optionally be added if required. Solution approaches to solve the periodic component of the PVRP are discussed in Section 3.2.4.

The VRP usually assumes a single depot, but multi-depot vehicle routing problems (MDVRP) have also been introduced in the past, (Pisinger and Ropke, 2007). Customers can be either assigned to multiple depots simultaneously (multi-sourced) or to one dedicated depot (single-sourced). The goal of the MDVRP is to meet the demand of customers given the depots' capacity constraints while minimising distribution costs.

According to Cordeau et al. (1997), the problem formulation for the PVRP can easily be adapted for the MDVRP by associating depots with days with a set of $|H|$ depots. The visit day combinations are then reduced to a set of single visits, each representing a depot. Every customer is restricted to only one visit day combination, resulting in the allocating the customer to a single depot. The travelling costs to and from the depot are the values c_{0jkl} and c_{i0kl} for depot l . The vehicle set, $|K|$, is also sometimes modified to vehicle sets, K_l , associated to the set of vehicles available at depot l , (Wang, 2013).

The difference between the MDVRP and the discrete CLRP is that the MDVRP is given a list of operational depots as input. The MDVRP focuses on solving the routing component, while the CLRP selects the depots to become operational from the provided candidate depots as well.

3.2.4 Solution approaches for the PVRP

Francis et al. (2008) and Campbell and Wilson (2014) provide comprehensive summaries of the PVRP. The PVRP was first introduced by Beltrami and Bodin in 1974. Although it is possible to use different time units, the time horizon usually spans over multiple days. Customers are assigned to different visit days within the horizon or allowed visit day combinations based on the visit frequency, (Francis et al., 2008). It is in general assumed that the demand of a customer should be evenly spread across all visit days, although this can vary for problem specific case studies.

3.2.4.1 Exact methods

Exact solution methods to solve the VRP were also attempted in the past, notably one by Foster and Ryan (1976), who discuss column generation techniques. Two interesting points made by Campbell and Wilson (2014) regarding this study, are that the formulation is exponential in size and column generation is slow to converge.

Francis et al. (2006) use Lagrangian relaxation to decompose the problem into a capacitated assignment problem and a number of TSPs. This is followed by the branch-and-bound method to search for the optimal solution.

3.2.4.2 Two-phased heuristics: construction and improvement

Besides exact methods, the first generation of solution approaches focussed on the two-phased construction and improvement heuristics. These were normally either methods to first create routes and then assign the routes to visit days or to first determine visit days for customers and then create routes per days.

Many cluster-first route-second approaches were also suggested. Two variants of this approach were given by Russell and Igo (1979). In the first, customers in close proximity to each other and with the same visit frequency are aggregated together to reduce the problem size. The centroid of the aggregated customers is used to represent the group. Next, daily customers are assigned to each day of the week and remaining customers are ordered from highest to lowest visit frequency and assigned to routes in a greedy manner. Three measures are used to determine the next best customer to assign: the average distance to the centroid of a route on a given day, the variance in the average and distance to the closest point on each route.

The second method also aggregates the customers and then uses the savings-based heuristic to generate routes. Customers need to be assigned to days beforehand and only certain combinations of delivery days are allowed to prevent visits too close to each other. Russell and Igo (1979) also suggest an improvement heuristic to better the results of these two methods by exchanging customers on routes.

3.2.4.3 Order based heuristic

Christofides and Beasley (1984) gave the first problem formulation for the PVRP and suggested decomposing the problem by first determining visit days before calculating routes. To assign visit days, they used an order based heuristic and ordered customers by highest demand and fewest visit day combinations first, therefore addressing the customers posing the biggest challenge first. Customers are then added to routes based on a greedy insertion algorithm, by adding the customer on the route with the smallest change in objective function. This is followed by an improvement phase customers are exchanged between feasible routes. They compared solving the p-median problem followed by the pTSP to solving the VRP problems in the improvement phase and conclude that the pTSP method gave better results.

3.2.4.4 Seed customer route construction

A simpler problem formulation was given by Tan and Beasley (1984). Here, the routing constraints of the problem are replaced with a general cost measure representing a visit to a customer without considering route ordering. They overcame complications with calculating the cost measure by finding seed customers and determining the costs based on the distance between the customers and their seed customers. The number of seed customers is equal to the maximum allowed number of vehicles per day $|K|$ times the number of days $|D|$.

3.2.4.5 Metaheuristic methods

The next generation of solution approaches for the PVRP focused on metaheuristics designed to escape local optima. Chao et al. (1995), for example, suggested a linear relaxation of the assignment problem and then try to make the problem feasible in an improvement stage. Their method also allows for infeasible moves below a certain threshold. The threshold gradually decreases to find a feasible solution at the end.

A tabu search method by Cordeau et al. (1997) was deemed as one of the most influential solution approaches by Campbell and Wilson (2014). They also allowed for infeasible solutions and compensate by adding an overcapacity penalty to the objective function. A greedy method was designed to determine which customer to add or remove from a route. Their tabu search method has a diversification phase but do not make use of an intensification phase.

Other metaheuristic approaches include a genetic algorithm with local search, first introduced by Drummond et al. (2001). According to Francis et al. (2008) this method is computationally intensive, but with the help of parallel computing it converges rapidly. When looking at the results they provide for academic test cases, it can be seen that this method becomes much slower than other methods in the

bigger case studies with more constrained capacities. See for example, Francis et al., 2008, p 85, instances 12 and 20 in Table 1. Three other metaheuristic approaches listed by Campbell and Wilson (2014) are the VNS, ACO and GRASP.

3.3 The location-routing problem (LRP)

One shortfall of the FLP is that it assumes that all customers are serviced on single trips or truckload (TL) shipments (Min, 1996). The LRP combines customer deliveries in multiple visits per trip, also known as less-than-truckload (LTL) shipments (Lam, 2008). Figure 3.6 gives an example of the discrete LRP. When compared to the illustration of the FLP in Figure 3.2, the difference in travelling distance becomes clear.

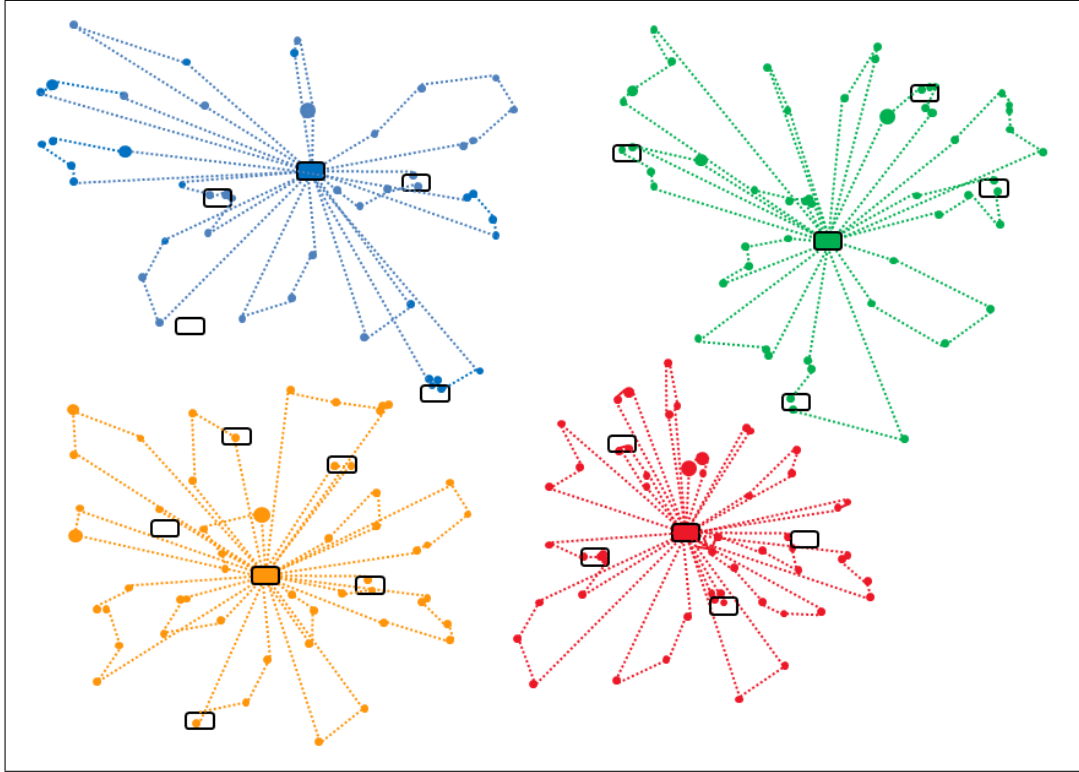


Figure 3.6: An illustration of the discrete LRP problem.

Salhi and Rand (1989) demonstrated the effect of neglecting routes when determining depot (facility) positions. An example shown in Figure 3.7, is a group of customers that can be serviced together in an outlier town. Because the distances between these outlier customers and the candidate depots are much further than the distances to the rest of the customers assigned to the depot, the outlier customers will have a bigger impact on the position of the depot than they should. This is depicted by the dotted square where the depot was originally located and the yellow square, where the depot is located after the outliers were included. If the outlier customers are

3.3 The location-routing problem (LRP)

grouped together on one route, the distance to the outlier town and back is added only once and the impact of the outlier customers to the depot position becomes more realistic.

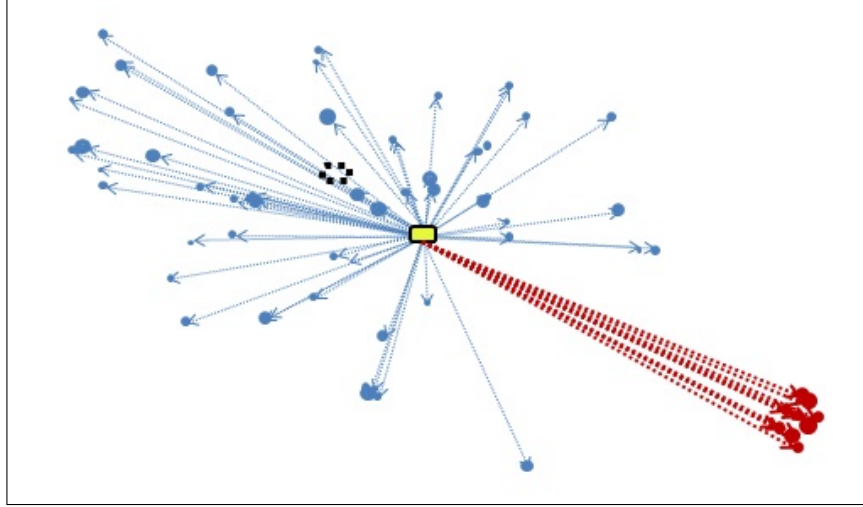


Figure 3.7: An example of the impact of outlier points on the placement of a depot in the LRP.

The LRP is therefore an extension of the FLP, but includes the creation of multiple routes similar to the MDVRP. The difference between the MDVRP and LRP is that all depots are in use in the case of the MDVRP and no decision about depot placements or number of depots to open are needed. Being derived from the FLP and VRP, which are both NP-complete problems, the LRP is also NP-complete, (Min, 1996, Prins et al., 2007, Lopes, 2011). Three types of LRP problems are described:

- The Hamiltonian p -median problem (HpMP)
- The planar or continuous location-routing problem (pLRP)
- The standard discrete capacitated location-routing problem (sdCLRP)

3.3.1 The Hamiltonian p -median problem (HpMP)

The HpMP was first introduced by Branco and Coelho (1990). They define the problem as a "routing location problem embedding the p -median and the travelling salesman problems". The problem divides a set of customer points into p mutually exclusive Hamiltonian cycles.

A Hamiltonian cycle in a graph $G(V, A)$ is a cycle that visits all vertices V and returns to the original vertex, (Simchi-Levi and Bramel, 1997). The HpMP is considered a location-routing problem where the objective is to minimise distribution costs while assigning n customers to p disjoint cycles, (Branco and Coelho, 1990).

3.3 The location-routing problem (LRP)

There are no capacity constraints for the depots or vehicles and the number of cycles to create, p , is known beforehand. The costs are a linear function of the distance between the customers in each cycle.

Problem formulation

Branco and Coelho (1990) formulated the problem in two ways, the first as a set partitioning problem, the second is based on the VRP. The VRP formulation is relevant to the p -median problem previously described. If the problem is depicted as a weighted directed graph $G(V, A)$, where V is the set of n customers (vertices / nodes) in the graph and A is the set of directed arcs, then the objective is to find exactly p cycles where every node belongs to one and only one cycle. Every node has exactly two arcs connected to it, one entering and the other exiting the node. The objective is to minimise the cost associated with these arcs. If an arc is connected, it is depicted as $x_{ijk} = 1$, meaning customer i precedes customer j in cycle k .

The formulation from Branco and Coelho (1990) identified p nodes as depots and allows a maximum of $|S| - 1$ arcs between S customer vertices in the same tour. This is similar to the subtour constraints, Eq. (3.52), from the VRP formulation. The variables y_{ik} are used to determine whether node i is a depot vertex for cycle k .

The biggest challenge according to Glaab and Pott (2000), with the HpMP formulation is to find a polyhedral formulation that ensures the cycles are not connected i.e. there are no other arcs with a value of 1, except the arcs forming the cycles. The authors refer to these constraints as the subtour number constraints (SNC).

Glaab and Pott (2000) rewrite the formulation by Branco and Coelho (1990) to exclude the depots - the y_{ik} variables. This is possible because, as noted by Branco and Coelho (1990) and Zohrehbandian and Hamidnia (2010), the depots do not have a fixed cost and the selection of depots has no impact on the distribution costs. Any customer in a cycle can therefore randomly be selected as the depot.

Glaab and Pott (2000) also point out that there are case studies where one and two nodes in a cycle are allowed and others where a cycle should require a minimum of three nodes, causing the assumption $V \geq 3p$ to appear in some problem definitions.

Other formulations are also provided by Zohrehbandian (2007) and Gollowitzer et al. (2014). The following definitions are given by Glaab and Pott (2000).

Define decision variables, x_{ijk} , as

$$x_{ijk} = \begin{cases} 1 & : & ij \in C_k \\ 0 & : & ij \notin C_k, \end{cases} \quad (3.66)$$

where $x_{ijk} = 1$ if customer i precedes customer j in cycle C_k for $k \in K$, with the

3.3 The location-routing problem (LRP)

number of vehicles $|K| = p$, the number of cycles or clusters. The vertices $V = I$ the set of customers. This can be divided into m pairwise disjoint subsets, called partitions P_m , with

$$P_m := \left\{ (S_1, \dots, S_m) \mid S_i \subset I, S_i \cap S_j = \emptyset \text{ for } i \neq j, \bigcup_{i=1}^m S_i = I \right\}.$$

For each element $(S_1, \dots, S_m) \in P_m$, let the direct m-cut associated with $(S_1, \dots, S_m)$ be denoted by:

$$A_m = A(S_1, \dots, S_m) := \{(i, j) \in A \mid i \in S_k, j \in S_l, 1 \leq k \leq l \leq m\}$$

If there are at least $p + 1$ cuts in P_m , the solution has at most p subtours.

The mathematical formulation of the HpMP (Glaab and Pott, 2000) is as follows:

$$\text{Minimise} \quad \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk} \quad i \neq j, \quad (3.67)$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \quad \forall i \in V, \quad (3.68)$$

$$\sum_{k \in K} \sum_{i \in V} x_{ijk} = 1 \quad \forall j \in V, \quad (3.69)$$

$$\sum_{i \in V} x_{ijk} - \sum_{l \in V} x_{jlk} = 0 \quad \forall j \in V, \forall k \in K, \quad (3.70)$$

$$\sum_{k \in K} \sum_{i, j \in A_{p+1}} x_{ijk} \geq 1 \quad (S_1, \dots, S_{p+1}) \in P_{p+1}, \quad (3.71)$$

$$\sum_{i \in V} \sum_{j \in V} x_{ijk} \geq 2 \quad \forall k \in K, \quad (3.72)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i, j \in A, \forall k \in K, \quad (3.73)$$

where c_{ij} is the distribution costs associated with travelling from customer i to j . The problem is restricted to p cycles because $|K| = p$, for the set of cycles K . The set V contains all n customer points. Constraints (3.68) and (3.69) ensure that each node has only one arc entering and one arc leaving the node respectively. Constraints (3.70) ensure that each node is assigned to one cycle. Constraints (3.71) are the SNC, preventing the creation of more than p cycles and subtours. Finally, Constraints (3.72) ensure that each route has 2 arcs or more and Constraints (3.73) are binary variable constraints.

3.3.2 Solution approaches for the HpMP

3.3.2.1 A clustering heuristic to solve the HpMP

Branco and Coelho (1990) solved the problem by creating p clusters using seed customers picked with the FDR (furthest distance rule). The seed customer of cluster k is referred to as k_1 . For each seed customer, the closest neighbour, k_2 , is found. To assign the rest of the customer points to clusters, the distance between each customer point i and seed k is calculated with the formula: $C_{ik} = d_{ik_1} + d_{ik_2}$, where d_{ik_1} is the distance between point i and k_1 .

The customers are then assigned to the cluster with the closest distance. The constant term, $d_{k_1k_2}$, can be deducted from the distance C_{ik} . Branco and Coelho (1990) suggest three other heuristics to also solve the HpMP, these are:

- The 3-optimal method, this is an improvement method. Once the customers have been clustered, any TSP heuristic can be used and the 3-opt heuristic can be used to create TSP routes.
- The shrinking heuristic, this method solves a 2-matching problem. The components of the 2-matching problem are then "shrunk" until a feasible HpMP solution is found, by merging the circuits of the 2-matching problem into Hamiltonian chains. The solution can be improved with the 3-optimal method.
- The spanning walk heuristic, this method starts by dividing the area where the customer points lie into small rectangles. The TSP of the customers in each rectangle is solved. Next the method merges the TSP subtours into p spanning walks until all customers are connected with an even number of edges of two or higher. The authors also state that this method did not return costs lower than the other two suggested heuristics.

3.3.2.2 An ACO algorithm to solve the HpMP

An ant colony optimisation (ACO) approach is suggested by Zohrehbandian and Hamidnia (2010). In this approach the ACO algorithm for the VRP is used by adding a virtual depot with zero distribution costs associated with it. Each node is given the demand of one, while the supply capacity is set to n , the number of customers in the problem. A VRP problem is then solved with the usual ACO steps, but when calculating the costs of a tour, the cost of connecting the two endpoints that go to the depot is also included. The tours are improved using a local exchange like the 2-opt heuristic.

Zohrehbandian and Hamidnia (2010) also included one case study, but because the instance only contained a distance matrix without any Euclidean coordinates, the case study could not be tested in this thesis.

3.3.2.3 A primal heuristic to solve the HpMP

Gollowitzer et al. (2014) mention a method that creates a solution for the relaxed problem using a primal heuristic. This is followed by solving the branch and cut search tree at every node using CPLEX. As soon as a constraint violation is picked up, the constraint is added to the problem. The primal heuristic consists of the following steps:

1. Create an MST graph.
2. Remove the longest edges, leaving k connected components.
3. For every component, C , with less than three nodes, search through all components with more than three nodes and find the closest edge to any node in C . Add this node to C .
4. Assign each node to its nearest neighbour.

The authors included test results for randomly generated instances as well as three TSPLIB instances given by Reinelt (2015). Unfortunately, the results only included the LP percentage gap and CPU times for the different models and not costs for the solutions.

3.3.3 The planar or continuous location-routing problem (pILRP)

The planar location-routing problem (pILRP) is similar to the HpMP because customers are also divided into mutually exclusive clusters to create routes cycles. The difference is that the routes are capacitated. The routes are also clustered together to originate from p depot positions, where p is known beforehand and the depots form part of the route cycles. The location of the depots are allowed to be anywhere on the Euclidean plane and calculated in such a way as to minimise the travelling distances. There are no supply capacity constraints placed on the depots, nor are there opening costs associated with the depots.

Problem formulation

Only a few authors have addressed the single-source planar location-routing problem in the past, most of them restrict the problem to a single depot. According to two review papers on the LRP, Nagy and Salhi (2007), and Drexler and Schneider (2015),

3.3 The location-routing problem (LRP)

the only authors to address the multi-depot variant of the pILRP problem were Salhi and Nagy (2009).

The problem formulation for the multiple depot single-source planar LRP is given by Salhi and Nagy (2009) as follows:

$$\text{Minimise} \quad \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} d_{ij} x_{ijk} \quad (3.74)$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \quad \forall i \in I, \quad (3.75)$$

$$\sum_{i \in I} \sum_{j \in V} w_i x_{ijk} \leq Q \quad \forall k \in K, \quad (3.76)$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} \quad \forall i \in V, \forall k \in K, \quad (3.77)$$

$$\sum_{i \in J} \sum_{j \in V} x_{ijk} \leq 1 \quad \forall k \in K, \quad (3.78)$$

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in V \setminus S} x_{ijk} \geq 1 \quad \forall S \subseteq I, \quad (3.79)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i, j \in I \cup J, \forall k \in K, \quad (3.80)$$

$$(X_j^1, X_j^2) \in \mathbb{R}^2 \quad \forall j \in J, \quad (3.81)$$

where the problem is defined as a weighted, directed graph $G = (V, A)$. Let I be the set of n customers to be clustered into routes, J represent the set of maximum p depot locations that can be placed anywhere on the Euclidean plane and $V = I \cup J$ is the set of all points in the graph. K is the set of vehicles or routes to be used, with $|K|$ the maximum number of vehicles. The edge $x_{ijk} \in A$, represent travelling from point i to point j using vehicle k , where $i, j \in V$. The number of vehicles and number of depots are decision variables that impact the objective function, based on the values of the x_{ijk} variables. The edge is given a weight, d_{ij} , representing the Euclidean distance (and associated costs) of travelling from point i to j if $x_{ijk} = 1$.

Each vehicle has a capacity constraint of Q and each customer has a demand of w_i . It is assumed that every vehicle makes a maximum of one trip and therefore represents exactly one route. The objective function is to minimise the distance of all connected edges, including the edges connected to the depots and back. Although the problem defined by Salhi and Nagy (2009) has vehicle capacity constraints, there are no depot capacity constraints. A maximum number of depots p is assumed although it is not part of the constraints given.

3.3 The location-routing problem (LRP)

Similar to the continuous FLP, the coordinates of depots need to be determined in such a way that the travelling distances between the first and last customers on the routes and the depots are minimised. Euclidean distances can be calculated using Eq. (2.33) but other distance calculations, for example Manhattan distances discussed in Section 2.8, are also possible.

Constraints (3.75) ensure that each customer is only assigned to one route. Constraints (3.76) stipulate that the total demand assigned to a vehicle should not be more than the vehicle's capacity. Constraints (3.77) enforce continuity of the routes by ensuring that there is an equal number of entry and exit edges per point on the same route. Constraints (3.78) denote that each route may leave from only one depot. If the vehicle is not used, the value will be zero. Constraints (3.79) eliminate subtours by ensuring that for any subset of customers, there is at least one edge on a route leaving the subset. Constraints (3.80) define binary variables x_{ijk} , while Constraints (3.81) allow the coordinates of the depots to be anywhere in Euclidean space.

Salhi and Nagy (2009) propose another constraint set to prevent routes from exceeding a maximum time limit. The constraint set is shown in Eq. (3.82). The delivery times are assumed to be a constant value of δ per customer. In this equation it is assumed that the travelling time is proportional to the travelling distance. The delivery times and maximum time limit are converted to a distance unit value. The constraint set is as follows:

$$\sum_{i \in V} \sum_{j \in V} x_{ijk}(d_{ij} + \delta) = D + \delta, \quad \forall k \in K. \quad (3.82)$$

Schwardt and Fischer (2009) use a different objective function for the pLLRP:

$$\begin{aligned} \text{Minimise} \quad & \sum_{i \in I} \sum_{j \in I} \sum_{k \in K} c_{ij} x_{ijk} + \sum_{i \in I} \sum_{k \in K} c_{0i}(x_0, y_0) x_{0ik} + \sum_{j \in I} \sum_{k \in K} c_{j0}(x_0, y_0) x_{j0k}. \end{aligned} \quad (3.83)$$

The objective function is split between the connected edges on the routes and the start and end edges to and from the depot. The cost function c_{ij} is proportional to the Euclidean distance between points i and j . This problem does not have any depot capacity constraints and is a special case of the problem given by Salhi and Nagy (2009) where $p = 1$.

3.3.4 Solution approaches for the pLLRP

Schwardt and Dethloff (2005) introduce a neural network solution approach for the pLLRP with a single depot. This was followed by Salhi and Nagy (2009) who de-

scribed a local improvement method to solve the pILRP with multiple depots. According to a survey paper by Nagy and Salhi (2007), these were the only two pILRP papers available at the time. Drexler and Schneider (2015) recently published a technical report in which the authors could only find two more papers addressing the single depot pILRP by Schwardt and Fischer (2009) and Manzour-al-Ajdad et al. (2012).

The location decision of the pILRP is similar to the continuous FLP, because depots are not restricted to specific locations. In order to overcome the placement of depots in areas where it is not possible, like for example a lake, Salhi and Nagy (2009) introduce the concept of forbidden regions.

3.3.4.1 Neural network approach on a self organised map (SOM)

Schwardt and Fischer (2009) suggest solving the single depot pILRP with a neural network approach on a self organised map (SOM). In this approach, the authors had adapted the use of a SOM to solve the mTSP and VRP because the depot locations are not available.

The authors illustrate the use of neuron rings to determine routes. Each neuron ring will represent a route at the end of a cycle and multiple neuron rings are connected at one point, called the depot neuron. A customer is inserted into a neuron ring based on its coordinates. The insertion could cause more than one neuron ring to get updated if the centre of the ring is close to the depot neuron.

In order to keep vehicle routes feasible a weighted capacity utilisation (γu_v) is added to the squared Euclidean distance when determining the strength of a neuron's adaptation. The weighted square distance is calculated as:

$$d'^2(x(t), w_j(t)) = d^2(x(t), w_j(t)) + \gamma u_v, \quad (3.84)$$

where u_v is the current demand of the route divided by the vehicle capacity and $d^2(x(t), w_j(t))$ is the squared Euclidean distance between input vector $x(t)$ and neuron j . The weighted square distance is only used in 50% of the cycles.

In order to create feasible solutions, customers assigned to overcapacitated routes, are re-allocated to other routes based on closest distance. A tabu counter is used to ensure customers cannot be re-assigned to the original routes. A non-elimination rule is also established to prevent a customer that has just been assigned to a route from being re-assigned to a different route, until the route becomes feasible.

The depot location can be improved using the well-known Weiszfeld algorithm. The starting coordinates (x_0, y_0) come from the feasible solution and the coordinates for the next iteration $t + 1$ are calculated using the Weiszfeld equation, given in Eq.

(3.44). The authors suggest using the values of $w_i = 1$, since the demand is not relevant to the distribution costs. For routes with only one customer, the authors suggest using $w_i = 2$, because the distance from the depot to the customer must be travelled forwards and backwards.

The authors concluded that the parameters have to be adjusted for every instance and can therefore not be used "off-the-shelf". They also suggest that their approach should only be used as a first approximation and can be improved further through a local search.

3.3.4.2 An iterative algorithm with the savings based method

Manzour-al-Ajdad et al. (2012) suggested an approach that embeds the routing problem as a subproblem in the location problem. The steps for this approach are as follows:

1. Calculate the centre of all the points with the use of the Weiszfeld equation, given in Eq. (3.44). Make the demand for each customer one.
2. Construct an ellipsoid with the calculated centre as the middle point. The horizontal and vertical radii r_1 and r'_1 are the standard deviation between the x and y coordinates of the customers respectively. Set $p = 1$ for the first iteration.
3. Generate N_p potential depot locations within the ellipsoid, where $N_p = l \times N_{p-1}$ and $0.2 \leq l \leq 1.5$. It is assumed that the points are randomly generated.
4. For each of the N_p potential locations, construct routes using the savings algorithm, adhering to the vehicle capacity constraints.
5. Improve the routes using five local search tour improvement methods.
6. Calculate the total distribution costs for each of the N_p potential depots and pick the potential depot with the best found solution as the centre of the next iteration.
7. Update the iteration to $p = p + 1$ and adapt the parameters r_p , r'_p and N_p for the new iteration.
8. Repeat the process from Step 3 onwards.
9. Continue until the improvement in distribution costs between two iterations is less than a predefined ϵ value.

3.3.4.3 An iterative algorithm that considers route endpoints

Salhi and Nagy (2009) propose an iterative algorithm where the location phase is solved initially, followed by alternating the routing and location phases until no more improvements can be found. The location phase is solved initially with the perturbation heuristic suggested by Salhi (1997), described in Section 3.1.2.7. Here the perturbation heuristic starts by opening all potential depots. Because this is a continuous problem and the authors made no mention how they identified potential depots, it is assumed that all customers points are the potential depot locations.

In the routing phase, the MDVRP is solved using the savings based heuristic. This is followed by a couple of refinement procedures which are assumed to be similar to the tour improvement methods described in Section 3.2.2.4. The customers first and last on every route travelling to and from the depots are identified as the endpoints of every route. These points are used to better the location of the depots with the use of the Weiszfeld equation, Eq. (3.44). Once again, the demand of the points is kept at unit demand, i.e. $w_i = 1 \ \forall \ i \in I$. It is assumed that routes remain allocated to the initial depots and depot allocations cannot change.

After the Weiszfeld equation is used to better the depot locations, the maximum time constraint set given by Eq. (3.82), is checked for violations. If this is the case, a weighted Weiszfeld equation is used. The route endpoints are weighted according to the percentage that the route exceeds the time capacity. This makes the endpoints of the routes in violation weigh more than the other route endpoints when calculating the depots. The new calculated depots will therefore be biased to be closer to these route endpoints. The total route cost of the solution is calculated and the method iterates between solving the routing and location problem until the improvements are below a given threshold or no improvements can be made to the current best solution.

Salhi and Nagy (2009) provide a list of results for instances tested. These instances include 18 sets of randomly generated data points with one depot and a repeat of 14 of the same instances for two depots. The number of customers vary between 50 and 199.

3.3.5 The discrete capacitated location-routing problem (CLRP)

The discrete capacitated location-routing problem (CLRP) has a capacity constraint for each candidate depot as well as homogeneous vehicle capacity constraints, (Nagy and Salhi, 2007). With the uncapacitated FLP, it is implicit that customers should be assigned to the closest open depot for the biggest saving. This is also to a great extent true for the CFLP, as long as the capacity constraints are adhered to. With the CLRP, this is not always the case. The assignment of other customers near the

3.3 The location-routing problem (LRP)

customer should also be taken into consideration.

An example is illustrated in Figure 3.8. Customers, represented by the coloured dots, are assigned according to shortest distance, to four depots represented by the coloured rectangles. The encircled customers in circle **a** are at an almost equal distance to the two depots. If vehicle capacity allows, it is logical that the yellow customer should be serviced together with the other blue customers. The same can be said of the customers in circle **b**. As long as vehicle capacity allows, distance will be saved by servicing customers close to each other together even if they are further from the assigned depot than from another open depot.

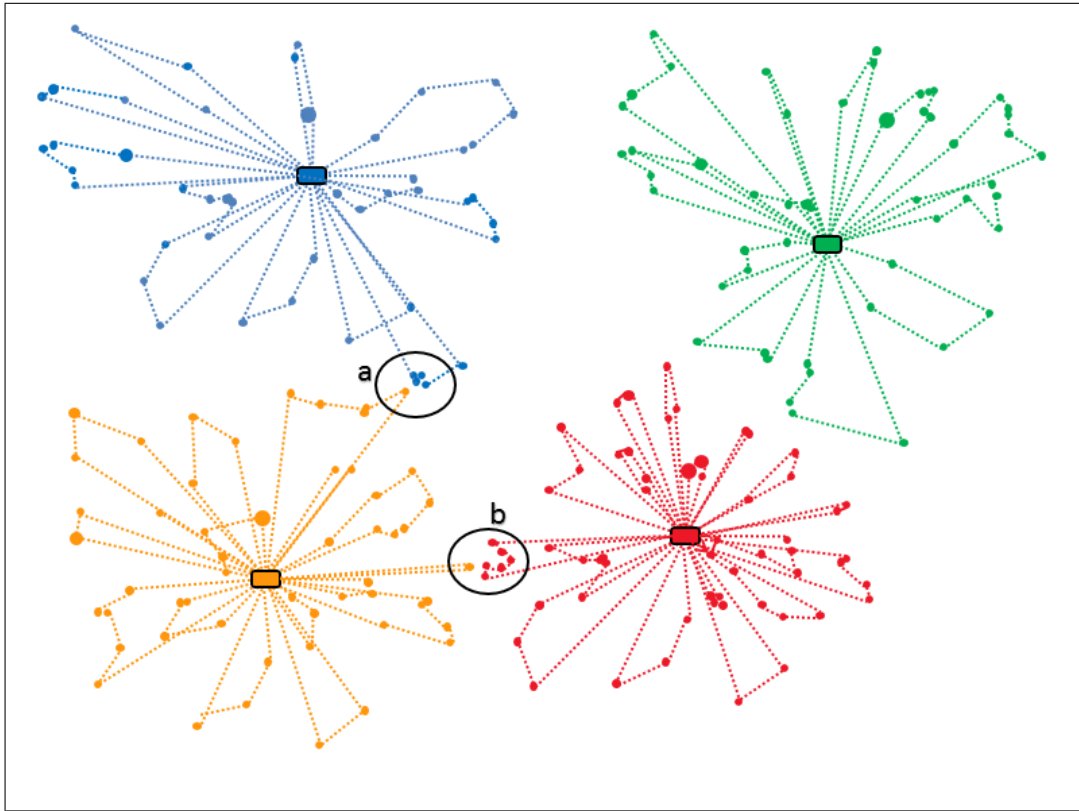


Figure 3.8: An example of customer assignments based on shortest distance in the CLRP.

The network location CLRP as described by Drexl and Schneider (2015), allows potential depots at any customer point or any place on the road network connecting two nodes. The authors report that they could not find any papers regarding the network location CLRP published. Lopes (2011) considered this problem classification the same as the discrete CLRP, because there are no changes in the demand calculation. The concentrator case, where the list of potential depots is the same as the list of customers, can also be considered a network location CLRP. In the FLP problem, a similar variant of the CFLP is referred to as the capacitated concentrator location problem (CCLP), see Section 3.1.1.1. In the concentrator CLRP, the problem formulation given for the discrete CLRP can be used as-is with only slight

modifications.

Problem formulation

The CLRP can be defined as a weighted and directed graph $G = (V, A)$, where V is the set of nodes. $V = I \cup J$ with I the list of all n customers and J the list of all m candidate depots. The boolean decision variable $z_{ijk} = 1$ if the edge (i, j) is part of the route of vehicle k and $i \in V$ and $j \in V$. The vehicles are represented by set K , containing $|K|$ homogeneous vehicles with capacity Q . If S is a subset of nodes with $S \subseteq I$ then $|S|$ is the number of nodes in the subset.

The mathematical formulation for the discrete CLRP is as follows, (Prins et al., 2007):

$$\text{Minimise} \quad \sum_{j \in V} \sum_{i \in V} \sum_{k \in K} c_{ij} z_{ijk} + \sum_{j \in J} f_j y_j + \sum_{j \in J} \sum_{i \in I} \sum_{k \in K} F z_{ijk} \quad (3.85)$$

$$\text{subject to} \quad \sum_{j \in V} \sum_{k \in K} z_{ijk} = 1 \quad \forall i \in I, \quad (3.86)$$

$$\sum_{i \in I} \sum_{j \in V} w_i z_{ijk} \leq Q \quad \forall k \in K, \quad (3.87)$$

$$\sum_{i \in V} z_{ijk} - \sum_{i \in V} z_{jik} = 0 \quad \forall k \in K, \forall j \in V, \quad (3.88)$$

$$\sum_{i \in I} \sum_{j \in J} z_{ijk} \leq 1 \quad \forall k \in K, \quad (3.89)$$

$$\sum_{i \in S} \sum_{j \in S} z_{ijk} \leq |S| - 1 \quad \forall S \subseteq I, |S| \geq 2, \forall k \in K, \quad (3.90)$$

$$\sum_{u \in I} z_{iuk} + \sum_{v \in V \cap \{j\}} z_{vjk} \leq 1 + x_{ij} \quad \forall i \in I, \forall j \in J, \forall k \in K, \quad (3.91)$$

$$\sum_{i \in I} w_i x_{ij} \leq W_j y_j \quad \forall j \in J, \quad (3.92)$$

$$\sum_{i \in I} z_{ijk} \leq y_j \quad \forall j \in J, \forall k \in K, \quad (3.93)$$

$$z_{ijk} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K, \quad (3.94)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \quad (3.95)$$

$$y_j \in \{0, 1\} \quad \forall j \in J. \quad (3.96)$$

3.3 The location-routing problem (LRP)

The objective function has two fixed cost variables, f_j is the fixed cost for opening depot j and F is the fixed cost for adding a vehicle to the model. The distribution costs from the CFLP are replaced with a cost per edge, where c_{ij} is the cost associated with transporting goods from site i to site j . Similar to the CFLP, if $y_j = 1$ then depot j is open and if $x_{ij} = 1$ then customer i is assigned to depot j . The demand of the customer i is defined as w_i and W_j is the maximum capacity of depot j .

The objective function Eq. (3.85) calculates the total costs. Constraints (3.86) ensure that there is only one delivery per customer and that every customer is assigned to only one route. Constraints (3.87) restrict the customer demand allowed on any given route to the vehicle capacity Q . Constraints (3.88) guarantee continuity of the routes and that each route returns to the depot of origin. Constraints (3.89) ensure that each vehicle has only one starting point and thus that every vehicle has only one route. Constraints (3.90) are used to avoid cycles or subtours, and Constraints (3.91) ensure that when a customer is assigned to a depot, it is also assigned to a route that belongs to this depot.

The capacity constraints (3.92) limit the total demand of customers assigned to an open depot to the supply capacity. Constraints (3.93) guarantee that vehicles are only assigned to open depots. These constraints are sometimes left out of the model because they are indirectly enforced by Eq. (3.91) and Eq. (3.92), but can help an algorithm find the solution quicker if it is part of the model. Lastly, constraints (3.94), (3.95) and (3.96) restrict the decision variables to binary values.

In the concentrator case, the only change to the problem formulation will be that each depot variable y_j can now be replaced by a customer variable x_{jj} and the number of potential depots given is equal to the number of customers, i.e. $y_j = x_{jj}, m = n$ and $J = I$.

3.3.6 Solution approaches for the CLRP

Drexler et al. (2014) provide a comprehensive summary of recent advances in the CLRP including a list of benchmark instances as well as different exact and metaheuristic solution approaches.

The common approach to solving the CLRP problem is to split it into a two-phase approach, (Tuzun and Burke, 1999): a location-allocation phase (or simply called the location phase) and a routing phase. The former determines the number of depots to use, the positions of these depots and the assignment of customers to the depots. The latter focuses on selecting the customers to be serviced on the same route based on vehicle capacity.

3.3 The location-routing problem (LRP)

There are different approaches for solving the CLRP. According to Nagy and Salhi (2007) and Lopes (2011) the following approaches could be followed:

1. The sequential approach - first the location and then the routing phase are solved or vice versa (Nagy and Salhi, 2007; Lopes, 2011). Since there is no feedback of information between the two phases this is often seen as solving two separate problems: the CFLP and the CVRP.
2. The cluster-based approach - customers are clustered into routes and depots are located using the medians or endpoints of the routes (Nagy and Salhi, 2007; Lopes, 2011). The customers can also be clustered according to depot locations, the routes for each depot are then determined by solving a single depot VRP or a travelling salesman problem (TSP).
3. The iterative approach - the two phases are iteratively solved with feedback between them and stop after a certain criterion has been met (Nagy and Salhi, 2007; Lopes, 2011).
4. The hierarchical or nested approach - the location phase is the superior decision, the routing phase is supportive and only used to calculate the associated distribution costs (Nagy and Salhi, 2007; Lopes, 2011).

3.3.6.1 The extended Clarke and Wright algorithm (ECWA)

Prins et al. (2006) adapted the savings based method by testing the merging of routes to be assigned to all m depots. Instead of four possible mergers per pair of routes, $4m$ possible mergers per pair must now be evaluated. They continue to state that the method can be reduced to $O(mn^2 \log n)$ time by using a preliminary heapsort method. The heapsort method is explained by Cormen et al. (2009, Ch. 6).

Figure 3.9 illustrates the ECWA, as provided by Prins et al. (2006). There are two routes with endpoints i, j and k, l respectively, assigned to depots r and s . The saving σ of the merger of points j and k and assignment to depot t can be calculated as:

$$\sigma = F + c_{ri} + c_{jr} + c_{sk} + c_{ls} - c_{jk} - c_{ti} - c_{lt} + f_r \lambda_r + f_s \lambda_s - f_t(1 - y_t) \quad (3.97)$$

where F is the cost saving of removing a vehicle. The cost of the edges to the current two depots r and s for each of the two routes is $c_{ri} + c_{jr} + c_{sk} + c_{ls}$. The cost to connect the two routes is c_{jk} . The cost to connect the new route to its new depot is $c_{ti} - c_{lt}$. The possible cost savings if a depot can be closed from either of the two routes is $f_r \lambda_r + f_s \lambda_s$. The extra cost incurred if the new depot has not been opened before is $f_t(1 - y_t)$.

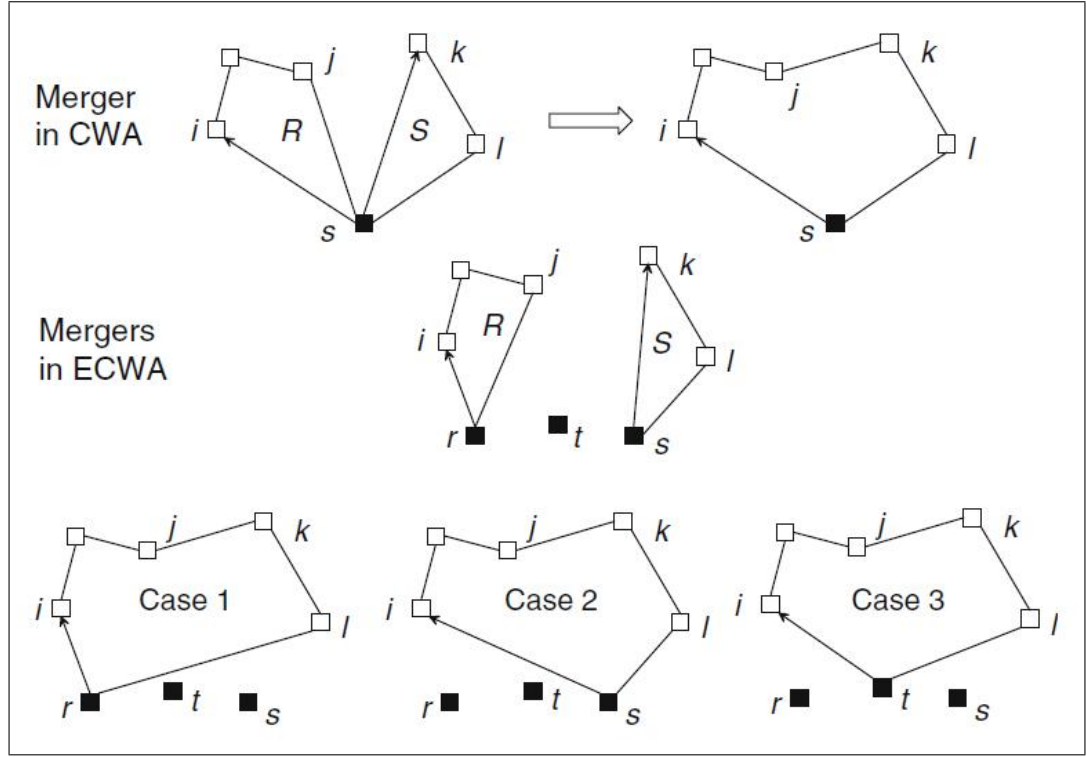


Figure 3.9: The ECWA method as illustrated by Prins et al. (2006).

In Eq. (3.97), λ_s is a boolean value to indicate if depot s has no more routes assigned to it and y_t is a boolean value to indicate whether depot t was already open beforehand.

3.3.6.2 GRASP

The greedy randomised adaptive search procedure (GRASP) metaheuristic is described by Blum and Roli (2003). It is an iterative process that consists of three steps:

1. Construct a greedy randomised solution
2. Perform a local search to better the solution
3. Compare the solution to the best found solution and update if needed.

In the first step, a greedy algorithm is used to construct the solution. An example of a greedy algorithm used to solve the FLP is explained in Section 3.1.2.2. Instead of only selecting the next best step to construct the solution, the GRASP method creates a restricted candidate list consisting of the best α choices that can be made. One of the choices is then selected randomly. It is clear that the choice of α is critical to the success of the method.

3.3 The location-routing problem (LRP)

In the GRASP method suggested by Prins et al. (2006) for the CLRP, the ECWA is used to construct an initial solution. When deciding which routes should be merged next, a restricted candidate list (RCL) of the α best possible mergers is calculated. A merger is then selected randomly and the routes are combined. This is referred to as the randomised extended Clarke and Wright algorithm (RECWA). The authors also suggest using the RECWA method to solve the MDVRP.

Since the value of α is critical, Prins et al. (2006) suggest selecting an α value randomly between 1 and a predefined maximum value each time. Once the routes have been constructed, the solution is improved with the use of a local search heuristic. This includes three possible moves:

- Move: a customer is moved to another route or a different place in the same route.
- Swap: two customers exchange their positions in the same or two different routes.
- 2-opt: two non-consecutive edges on the same or two different routes are removed and the routes are reconstructed. If the two new customer endpoints of a route do not end at the same depot, the routes are changed to end at the same depot. It is assumed that the depot that will incur the minimum route cost is selected.

The local search stops when no more moves can be made and the best found solution is updated if needed.

Prins et al. (2006) continue to enhance their GRASP heuristic with an extra element, which they call a learning process. The process divides the construction step into two modes. In the first mode, called the diversification mode, the method tries to vary the set of opening depots throughout the solution search space. In the second mode, the intensification mode, the method restricts the solutions to the set of opening depots from the best solution found in the diversification mode. The modes are explained below.

In the first iteration, each customer is placed on its own route assigned to its closest depot. All depots with customers closest to them are opened. During the rest of the diversification mode, only two depots are opened at the start. One depot is chosen randomly and the other iteratively to ensure that all depots are opened at least once.

If it becomes clear that a customer cannot be assigned to one of the two depots because of capacity constraints, the depot closest to this customer is also opened. Once an initial solution has been created, the process continues as described above.

3.3 The location-routing problem (LRP)

The routes are merged in the usual way and all depots are checked when deciding which depot a merged route should be assigned to.

After a fixed number of iterations, the method switches to intensification mode. The best found solution from the diversification mode is copied and also saved in the variable named *BSolInDiv*. Solutions constructed in the intensification mode are restricted to the subset of depots used in *BSolInDiv*. The merging of routes only considers the open depots from *BSolInDiv*. If, during the initial assignment of customers to depots, customers cannot be assigned because of capacity constraints, additional depots can be opened and included.

After every iteration, the solution is compared to the best solution found so far in both the diversification and intensification modes, but the solution kept in *BSolInDiv* will only depend on the diversification mode.

A post-optimisation step, named path relinking, is also proposed by Prins et al. (2006). For this step, a group of elite solutions that are not only the best found, but also the most distant from each other, are compared. The objective is to try to find a better solution by changing some attributes from a good solution with that of another solution distant enough from the first.

To calculate the distance between two solutions T and U , for each pair of consecutive customers i and j in T , Prins et al. (2006) provide the following rules:

- If i and j are next to each other on a route in U , then add 0 to the distance.
- If i and j are on the same route, but not consecutive, add 1.
- If i and j are not on the same route, but both assigned to the same depot in U , add 3.
- If i and j are assigned to two different depots in U , add 10.
- For customers at the beginning of a route in T that are assigned to a different depot in U , add 10.

The best solutions from the diversification mode are placed in a set, named *NBestSet*. To determine the set of solutions, named *DistSet* for the path relinking, add the best solution from *NBestSet*. Calculate the distances from the best solution to all other solutions in *NBestSet* and add the solution with the maximum distance. Each time the solution furthest from all the other solutions in *DistSet* is added until a predefined maximum number is reached.

To change a solution U to the guiding solution T , identify all the pairs (i, j) of customers that are consecutive in T but not in U . The routes in U are broken after

each i and shifted in the route sequence until all pairs (i, j) are also consecutive in U . If a capacity constraint for a vehicle or depot is broken, chains of customers are moved to repair the constraint as above, while keeping i and j in the order they were in solution U . After path relinking, the solution is refined with a local search and if a better solution is found, the best found solution is updated.

3.3.6.3 The four-step sequential clustering heuristic

Barreto et al. (2007) introduced a four-step sequential clustering heuristic to solve the CLRP. In the first step, customers are grouped based on vehicle capacity. The next step is to construct TSP routes within the groups. The third step is to improve the routes using a 3-optimal local search procedure. The last step is to collapse each route into an aggregated customer, determine the best depots to open and locate the routes to them.

In the second step, each TSP is solved by either an exact algorithm, if the number of customers in the group is less than 40, or by a heuristic method otherwise. The exact algorithm first solves the LP relaxation of the problem without the subtour constraints, Eq. (3.91). It then checks if a constraint was broken and if so, introduces it for the specific route iteratively until the problem is solved. For more than 40 customers, a two stage heuristic is used. An initial solution is obtained by selecting the furthest customer and inserting it in the route based on the savings heuristic. The second stage is to better the solution using the 3-optimal local search procedure.

Four grouping methods were considered in step one. In each one, six different distance measures were used, providing six different variants of the same method. Five of the six distance measures correspond with the SAHN methods for calculating the inter-cluster distances, (Section 2.1). These five are single linkage, complete linkage, group average linkage, centroid linkage and Ward's method.

A sixth distance measure is called the savings distance measure. Here a pair of customers from both groups are selected. (For example, select i and j in group A and k and l in group B. The distance is then

$$d(A, B) = \min_{i,j \in A; k,l \in B} [\min(\alpha, \beta)] \quad (3.98)$$

$$\text{with } \alpha = d(i, j) + d(k, l) - d(i, k) - d(j, l), \quad (3.99)$$

$$\beta = d(i, j) + d(k, l) - d(i, l) - d(j, k). \quad (3.100)$$

The four grouping methods are:

1. One-phase hierarchical method

This is the same as the SAHN methods described in Section 2.1. Customers start in groups of their own and groups are merged iteratively, based on the nearest distance. Once the vehicle capacity is reached, the group is excluded from the problem, but other groups can still be merged. This can lead to far away groups being merged.

2. Two-phase hierarchical method

The number of groups (r) to create is based on the formula:

$$r = \left\lceil \frac{\sum_{i \in I} w_i}{Q} \right\rceil \quad (3.101)$$

where w_i is the demand of customer i and Q is the vehicle capacity.

SAHN clustering methods are then used to create r groups, ignoring the capacity constraints. The method then attempts to fix the solution by moving customers to other groups with excess capacity based on a shortest distance coefficient. The coefficient of a customer is the ratio between the distance from the customer to the potential new depot and the distance from the customer to the current depot. The coefficients are ordered and the customer with the smallest coefficient is moved iteratively, until all capacities are met. If the method cannot move customers to meet the capacity constraints, an extra group is added and the method is restarted.

3. Direct assignment non-hierarchical method

Using the FDR (furthest distance rule), r seed customers are chosen where the calculation of r is given in Eq. (3.101). Customers are grouped with the chosen r customers based on closest distance. Only the distances between the chosen r customers and the individual customers are used and group distances are not taken into consideration. No mention is made of how the vehicle capacity constraints are enforced. In this method, all distance measures return the same clustering sequence as the single linkage method, because all compare single customers to the cluster groups forming around the r customers. The only exception is the distance measure calculation for Ward's method because the method of distance calculation, Eq. (2.5), uses the mean of the cluster.

4. Sequential assignment non-hierarchical method

The FDR is again used to choose r customers. The rest of the customers are assigned to groups sequentially. When determining the distance between a customer and a group, the complete group of customers is used. Unfortunately, Barreto et al. (2007) do not explain how the saving distance measure should be calculated between one individual customer and a group.

The conclusion from Barreto et al. (2007) was that the one-phase hierarchical method had a slightly better performance than the other grouping methods and that the group average method performed better on average compared to the other distance measures. Although the current best-known solutions were found for some instances, there were no new best-known solutions found by this method.

3.3.6.4 The capacitated hierarchical clustering heuristic

Lam and Mittenthal (2013) propose a hierarchical clustering based heuristic, similar to the one-phase hierarchical method with single linkage distance measure, (Barreto et al., 2007). The heuristic starts by creating route clusters. Instead of determining the number of vehicles to use beforehand, the authors propose two stopping rules to determine when to stop clustering.

The first stopping rule is called the minimum number of clusters stopping rule. Here the method continues clustering as long as the capacity constraint is not violated. If a merge will result in infeasible routes exceeding the capacity, the merge is ignored and the method moves to the next possible merge. This process continues until no more mergers are possible. The second stopping rule is called the change in variation stopping rule. This rule will only merge two clusters if the resulting cluster is a feasible route, but it also only merge clusters if the variation explained by the merge is significant compared to the variation from the previous merge. If this is not the case, the merge is ignored and the method moves to the next possible merge. A distance matrix is kept up to date to assist with the clustering.

After all customers have been allocated to route clusters, TSPs are created for each route, based on a variation of the k-opt method. The routes are aggregated into single customers and the location problem is solved. The method used to solve the location phase is not specified. Once initial solutions have been created, the routes are improved by the use of a descent heuristic. A descent heuristic is a method that searches for improvements and does not allow any changes that will not better the solution. The descent heuristic includes within and between route swaps of maximum 3 customers at the same time. The first improvement found is implemented, unlike the best found improvement used in some heuristics.

Although Lam and Mittenthal (2013) did not find better solutions than the current best-known, the authors do claim that their approach, with the change in variance stopping rule, returned solutions within 5% of the best-known solutions in 91.7% of the instances used by Prins et al. (2006). The authors also report that their approach returned on average better solutions than the solution approach proposed by Barreto et al. (2007). Fast CPU times were also reported, with a maximum of 48 seconds for an instance with 150 customers and 20 potential depots. The authors

therefore recommend using this method to create initial solutions with values close to the best found solution.

3.3.6.5 The hybrid extended savings method (HES)

Lopes (2011) introduced the HES method for the CLRP as a constructive heuristic. The method can be succeeded by an improvement heuristic as explained in the AGS method, (Section 3.3.6.6).

The method first solves the location phase using the tabu search metaheuristic. The location phase solution provides the set of open depots as well as the allocation of customers to depots. Single customer routes are constructed and the routes are merged using ECWA. After $0.2 \times n$ mergers, two local search methods are used to refine the solution, before continuing with the merging of routes.

The two local searches are as follows:

1. Composite local search (CLS)

This search comprises of three steps; the tabu search to solve the CFLP, a relocation method and the 2-opt local search. The location phase is again solved with the tabu search method, similar to the one at the start of the construction phase. Since the current solution consists not only of customer assignments to depots, but also of routes, Lopes (2011) suggests collapsing all routes into single customers. The author does not specify the method to collapse the route, but it is assumed to be the average of the coordinates of all customers on the same route.

Once the depot location and allocation of routes to depots have been determined, the search continues with a relocate method. This method iteratively relocates a single customer to another position in either its current route or on another route that will induce a cost saving. The method is prohibited from allowing a move that will break the vehicle or depot capacity constraints. It concludes with the 2-opt local search as described in the GRASP method, (Prins et al., 2006).

2. Guided local search (GLS)

After the CLS is completed, the GLS method is used. The GLS consists only of the two steps; relocate and 2-opt local search. The method "guides" the solution towards a near optimal solution by penalising long route edges. The method also tries to escape from local optima by penalising the number of times the same edge is chosen.

Lopes (2011) suggests changing the cost associated with an edge as follows:

$$c'_{ij} = c_{ij} + p_{ij}\lambda L, \quad (3.102)$$

where L is the average length of all the route edges in the solution, λ is a parameter which Lopes (2011) has empirically set to 0.05 and p_{ij} is the number of times the edge has already been penalised. With each iteration, only one edge is penalised. This is the edge with the highest utility value U . The utility value is calculated as follows:

$$U = \frac{c_{ij}}{1 + p_{ij}}. \quad (3.103)$$

The local search is only performed on the penalty variable neighbourhood (PVN), which consists of the routes closest to the route edge that was penalised. The following routes are added to PVN:

- The route containing the penalised edge.
- All routes assigned to the same depot as the route of the penalised edge.
- If the routes are collapsed into single customer points, determine the closest routes to the route with the penalised edge.

The size of the PVN is calculated as a random value between a predefined minimum number of routes and a maximum of 75% of the total number of routes in the solution. Lopes (2011) suggests using 250 iterations before resetting the penalised values and restarting at most 5 times.

After completion of the two local searches, the construction phase continues merging routes until either another $0.2n$ mergers are completed or no more mergers with a cost saving are possible. The two local searches are then performed a final time before the method stops.

3.3.6.6 The active guided search method (AGS)

The AGS method is also introduced by Lopes (2011). The method consists of two phases, a construction phase (using the HES method described above) and an intensification phase.

The intensification phase consists of two steps. It starts with a repeat of the GLS, but this time all depots are included in the problem. A more thorough search with 500 iterations is used. The second step is to repeat the construction phase once more, but this time excluding the depots that were closed in the current solution.

Lopes (2011) concludes that the intensification phase can be done iteratively, but was only executed once in the results he obtained.

3.3.6.7 The cooperative Lagrangian relaxation LR - granular tabu search heuristic (CLRGTS)

Prins et al. (2007) introduced this two-phase heuristic approach, which alternates between the location phase and the routing phase, to solve the CLRP. The method is called cooperative because there is no hierarchy between the two phases and each phase serves as input to the other. While the location phase provides the set of open depots to use in the routing phase, the routing phase provides the list of customers, aggregated by routes to use in the location phase.

The method starts with the LR to solve the location phase. For the initial iteration, it starts with a CLRP solution created with any constructive heuristic. The routes from this solution are aggregated into "supercustomers" representing the total demand for the route. In order to create a "supercustomer" for a route k , the edges to the depots are removed from the route and the endpoint customers are connected to each other to form a cycle. To calculate the cost (distance) between route k and each open depot j is the minimum insertion cost to add this depot between two consecutive customers i and l . The cost for route k to depot j , c_{kj} is calculated as $c_{kj} = c_{ij} + c_{jl} - c_{il}$.

The location phase provides the set of open depots for the routing phase. The routing phase is solved with the granular tabu search method for the VRP as explained in Section 3.2.2.6. The method continues to alternate between the location - and routing phases until the solution stops changing.

In order to prevent the method from converging to a local optimum, it keeps track of the frequency that edges are introduced in the solution. With each routing phase, the most frequently used edges within a predefined number, L , are automatically part of the solution, as long as each customer is visited only once and vehicle capacity is adhered to. L is made larger with every iteration. The method can restart a given number of times. With every restart, the edges in L are added to the new starting solution with a given probability to allow for diversification.

3.3.6.8 The partial optimization metaheuristic under special intensification conditions (POPMUSIC) algorithm

Alvim and Taillard (2013) proposed using the POPMUSIC algorithm to solve the LRP that has a time complexity of $O(n^{3/2})$ and can solve instances with more than 10^6 customers. The algorithm depends on a low time complexity starting solution

3.3 The location-routing problem (LRP)

for the CpMP which is then improved by the POPMUSIC algorithm with quasi-linear time complexity. The problem is actually a pLLRP, but this is simplified to the concentrator problem by only allowing depots to be placed at customer points. Depots do not have capacity constraints and although there is a vehicle capacity constraint, an unlimited number of vehicles are available.

The steps to create a CpMP solution are used numerous times in the algorithm and work as follows:

1. Select $\sqrt{n}$ random centres.
2. All customers are assigned to their closest centre.
3. The cluster centres are updated for 150 iterations using the LR method with a gradient method.

This solution can still contain some infeasibilities and is referred to this as a relaxation of the CpMP.

In order to create an initial starting solution, $20\sqrt{n}$ customers are sampled and the CpMP with $\sqrt{n}$ clusters are solved. This is referred to as superclusters. After the centres of the new $\sqrt{n}$ superclusters have been calculated, the complete dataset of n customers are assigned to the nearest supercluster centre. For each of the $\sqrt{n}$ superclusters, the number of vehicles needed is calculated as the total cluster demand divided by the vehicle capacity.

The CpMP per cluster is calculated as given above to create route clusters. For all route clusters a TSP tour is determined in $O(n)$ time complexity, where n is the number of customers in the tour. If the distribution cost of the tour is more than the depot cost, the cluster is decomposed into two and TSP tours are created for each. This process continues until all tours have a distribution cost lower than the depot costs. To avoid degrading the LRP, the authors suggest using a vehicle capacity $T \leq Q$.

The centres of the TSP tours are calculated as the depots to open so that each depot has exactly one route. Next, all tours where the distance between the depots will result in a lower distribution cost than the depot cost, are merged, ignoring the capacity constraint. Merging is done in a greedy manner and depot centres are calculated. The process continues until no more improvements can be made through the merging of tours. The depots for the solution have now been determined. In order to determine the routes, cut all TSP tours which are overcapacity into two routes until all routes meet the vehicle capacity constraint.

Once the initial solution has been created, the solution can be improved using the POPMUSIC algorithm. The solution is decomposed into subproblems, where every

seedpoint is the centre of a route. All routes assigned to the six closest depots to this route centre are part of the subproblem. The subproblem is a MDVRP which is improved using the tabu search algorithm. The algorithm uses two types of moves to improve tours: the swapping of two customers and the move of one customer at a time from one route to another. It allows only one iteration per customer. The subproblem with this route centre is calculated again and the improve search repeated until there are no more improvements. The algorithm then turns to the next subproblem until no subproblems can be improved anymore.

Alvim and Taillard (2013) created instances based on six large TSP datasets from Cook (2013). These datasets do not have depot capacities nor maximum number of vehicle constraints.

3.3.6.9 An approximate algorithm for the CLRP

Approximate algorithms are algorithms that can find near optimal solutions in polynomial time. A guaranteed optimality gap is given. This provides a ratio factor of how far out the solution determined by this algorithm can differ from the optimal solution in the worst case. Harks et al. (2013) suggested an approximate algorithm based on a method suggested by Ravi and Sinha (2006) for the UFLP without single-source constraints. The algorithm merges the solutions of the UFLP and the MST. The guaranteed optimality gap is 4.38%, but the authors remark that for the instances tested with best-known solutions, the solutions were on average within 10% – 20% of the best-known solution.

The problem defined by Harks et al. (2013) did not have any depot capacity constraints, only homogeneous vehicle capacity constraints. Fixed depot and vehicle costs were defined and the multi-source variant of the problem was used. In order to calculate the algorithm's CPU time for large instances, random datasets were generated with between 1 000 – 10 000 customers and 100 – 1 000 potential depots. The authors commented on the enormous growth in real-life problem sizes and the need to find "fast heuristics" with polynomial time complexity.

3.4 The periodic location-routing problem (PLRP)

The PLRP, recently introduced by Prodhon and Prins (2008), is similar to the CLRP, but instead of allowing only one visit per customer, different visit frequencies are allowed. In the CLRP, customers are clustered into capacitated routes, while in the PLRP visits are clustered into capacitated routes. The authors also added extra constraints to ensure visits to the same customer cannot be placed on the same route. Each customer must be assigned to the same number of routes as the number of required visits, this is referred to as the visiting frequency. Although a customer

base with varying visit frequencies are normal in a real-life environment, this problem has not received much attention in the LRP literature up to date.

Problem formulation

Given a time horizon H , consisting of l periods (normally days), each customer, i , has a set of possible combinations of visit days allowed, called $Comb_i$. The demand for a customer i can now be defined as w_{ilr} , where l specifies the day of the visit and r specifies the combination of visit days from $Comb_i$ that is used.

The maximum number of vehicles used on any given day within the j_{th} depot is called T_j . This is also referred to as the fleet of the depot, (Fu and Ishkhanov, 2004). The fixed cost associated with having a vehicle in the fleet is F . A set of K vehicles are available each day, with the maximum number of vehicles available as $|K|$. The routes have a capacity restriction of Q per day. Subsets I and J represent the set of customers and depots respectively with $V = I \cup J$. The depots each have a supply capacity of W_j that cannot be exceeded on any given day in H .

The problem formulation for the PLRP is similar to the CLRP in many ways. It is given by Prodhon (2011) as follows:

$$\text{Minimise } \sum_{j \in V} \sum_{i \in V} \sum_{k \in K} \sum_{l \in H} c_{ij} z_{ijkl} + \sum_{j \in J} f_j y_j + \sum_{j \in J} F T_j \quad (3.104)$$

subject to:

$$\sum_{i \in I} \sum_{k \in K} z_{ijkl} \leq T_j \quad \forall j \in J, \forall l \in H, \quad (3.105)$$

$$\sum_{i \in I} \sum_{j \in V} w_{ilr} z_{ijkl} \leq Q \quad \forall k \in K, \forall l \in H, \forall r \in Comb_i, \quad (3.106)$$

$$\sum_{i \in V} z_{ijkl} - \sum_{i \in V} z_{jikl} = 0 \quad \forall k \in K, \forall j \in V, \forall l \in H, \quad (3.107)$$

$$\sum_{i \in I} \sum_{j \in J} z_{ijkl} \leq 1 \quad \forall k \in K, \forall l \in H, \quad (3.108)$$

$$\sum_{i \in S} \sum_{j \in S} z_{ijk} \leq |S| - 1 \quad \forall S \subseteq I, |S| \geq 2, \forall k \in K, \forall l \in H, \quad (3.109)$$

$$\sum_{u \in I} z_{iukl} + \sum_{v \in V \cap \{j\}} z_{vukl} \leq 1 + x_{ij} \quad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in H, \quad (3.110)$$

$$\sum_{i \in I}^n w_{ilr} x_{ij} \leq W_j y_j \quad \forall j \in J, \forall l \in H, \forall r \in Comb_i, \quad (3.111)$$

$$\sum_{r \in Comb_i} b_{ir} = 1 \quad \forall i \in I, \quad (3.112)$$

$$\sum_{j \in V} \sum_{k \in K} z_{ijkl} - \sum_{r \in Comb_i} b_{ir} a_{rl} = 0 \quad \forall i \in I, \forall l \in H, \quad (3.113)$$

$$z_{ijkl} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K, \forall l \in H, \quad (3.114)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \quad (3.115)$$

$$y_j \in \{0, 1\} \quad \forall j \in J, \quad (3.116)$$

$$b_{ir} \in \{0, 1\} \quad \forall i \in I, \forall r \in Comb_i, \quad (3.117)$$

$$T_j \in \{1, 2, \dots, N\} \quad \forall j \in J. \quad (3.118)$$

The cost to travel from a customer or depot point i to another point j is c_{ij} and z_{ijkl} is a binary variable to indicate travel between points i and j on day l using vehicle k . The fixed depot cost f_j is only incurred if the j_{th} depot is open, indicated by the binary variable y_j . The fixed vehicle cost F is multiplied by T_j , the maximum number of vehicles used on any given day by depot j . This forces the model to balance the number of vehicles used per day. x_{ij} is a binary variable to indicate whether customer i is assigned to depot j .

Eq. (3.104) is the objective function value of the PLRP. Similar to the sdCLRP, the objective function value is composed of three terms: the distribution costs, the depot opening costs and the fixed vehicle costs. Constraints (3.105) ensure that T_j is the maximum number of vehicles used on any given day by the j_{th} depot. Constraints (3.106) restrict the customer demand allowed on any given route to the vehicle capacity Q . Constraints (3.107) guarantee continuity of the routes and that each route returns to the depot of origin. Constraints (3.108) ensure that each vehicle has only one starting point and thus that every vehicle has only one route per day.

Constraints (3.109) are used to avoid cycles or subtours, while constraints (3.110) ensure that a customer is assigned to the same depot as the depot of the route he belongs to. The capacity constraints (3.111) limit the total demand of customers assigned to an open depot to the supply capacity. Constraints (3.112) ensure that only one combination of visit days per customer is selected, represented by the binary decision variable b_{ij} .

Constraints (3.113) restrict the routes to one and only one visit per customer per day. These visits are only allowed on the selected set of visit days r , referred to as a_{rl} . Constraints (3.114) - (3.117) restrict the decision variables to boolean values, while Constraints (3.118) restrict T_j to positive integer values.

3.4.1 Solution approaches for the PLRP

A few methods have been suggested to solve the PLRP. In most cases these methods were adapted from either PVRP or CLRP methods. The methods are as follows.

3.4.1.1 An iterative metaheuristic for the PLRP

The PLRP was first introduced by Prodhon in 2007 in a conference proceeding and the method introduced to solve the PLRP was discussed in Prodhon and Prins (2008) and Prodhon (2011). It is called an iterative metaheuristic for the PLRP and is described as follows.

The method starts each iteration by determining a set of depots to use. When determining the depot locations, the entire set of customers to be visited during the time horizon are considered and a CLRP is solved. Next, the method assigns combinations of visit days to customers. Customers that were frequently on the same routes when solving the CLRP, are assigned to the same set of visit days.

Once the customer visit days have been determined, the problem is decomposed into one multi-depot vehicle routing problem (MDVRP) per day. Prodhon and Prins (2008) suggest using the Randomized Extended Clarke and Wright Algorithm (RECWA) method to solve these. A local search is performed on exchanging customer visit days, each time solving the associated MDVRP problems. The iteration is completed when it converges and a new iteration is started with a different set of opening depots.

3.4.1.2 A memetic algorithm with population management (MA|PM)

A memetic algorithm is an iterative genetic algorithm (GA) hybridized with a local search, (Prodhon and Prins, 2008). It starts by assigning visit days to customers. It then solves the CLRP for each day using a GA. Solutions too close to each other are not allowed in the population, this is referred to as population management. A threshold parameter Δ is introduced. Offspring solutions with a distance closer than Δ to the solutions in the population set are either mutated or discarded. Distances can be calculated similar to the GRASP method for the CLRP, (Prins et al., 2006).

For each child solution, a local search on the visit days is performed. Three types of local searches are considered: move, swap and opt. These are similar to the local search methods performed in the GRASP method for the CLRP. The results from the local search are recorded and used to bias the next global iteration.

3.4.1.3 An ELS x path relinking hybrid algorithm

This method starts by assigning visit days to customers, but the assignments are generated randomly, (Prodhon, 2009b). The solution is then evaluated and improved with an evolutionary local search (ELS) to better the visit day assignments. The evaluation is done by using the RECWA method to solve a MDVRP for each day. In the final stage, the method uses path relinking to intensify the search for better solutions as a post-optimisation process.

An ELS is an extension of an iterated local search. The iterated local search uses one solution to create an offspring using mutations and local search techniques at each iteration. The evolutionary local search also uses a one parent solution, but creates a number of offspring solutions. The best offspring is then selected for the next iteration. Prodhon (2011) also suggests using a multi-start variant of the ELS. The local search techniques are similar to the GRASP method of the CLRP.

3.4.1.4 A hybrid evolutionary algorithm

The method introduced by Prodhon (2011) is similar to the ELS x path relinking hybrid algorithm. The method excludes the path relinking algorithm and uses local search methods to better the routes, depots and visit days respectively. The local searches can be described as follows:

1. For the route local search, an improvement heuristic similar to the 2-opt tour improvement method is suggested.
2. The depot local search is based on the selection of depots per day. The RECWA method is applied for the solution to find the most suitable depots to use every day. In order to limit the amount of depots opened across the time horizon, the depots are limited to a subset. Two ratios are calculated per depot:
 - The maximal percentage of the daily use of the depot capacity on any given day.
 - The percentage of the total demand supplied by the depot over the complete time period.

The product of the two ratios are used to rank the depots, the depots with the highest ranks are opened to ensure that the total demand per day can be met on any given day.

3. In the visit day local search, customers' visit combinations are changed one at a time. If the new combination gives a better cost, the change is applied.

3.4.1.5 A VNS coupled with ILP-based very large scale neighbourhood searches

Pirkwieser and Raidl (2010) introduced a variant of the variable neighbourhood search (VNS) metaheuristic for solving the PLRP. The method relaxes both the depot and vehicle capacities when creating new solutions. Pirkwieser and Raidl (2010) recommend introducing a penalty cost relative to the size of the violation in the objective function.

It starts by creating an initial solution as follows:

1. Visit day combinations for each customer are selected randomly.
2. The maximum customer demand needed on any given day is determined. This is equal to the minimum required depot capacity needed.
3. Depots to open are selected randomly until the depots' accumulated capacity reaches the minimum required depot capacity needed.
4. For each day, every customer is assigned to the closest depot. Routes are constructed using CWA and adhering to the vehicle capacity constraints. If there are more than $|K|$ routes, the routes with the least customer assignments are dropped and these customers are re-assigned to their closest neighbour's route, allowing vehicle overloading with penalised costs.

Pirkwieser and Raidl (2010) also remark that the initial solution processes do not have to be very sophisticated. From preliminary tests, good quality initial solutions are often not beneficial to the overall optimisation process.

Once an initial solution has been constructed, neighbourhoods are constructed during the shaking phase. Pirkwieser and Raidl (2010) list five neighbourhood structures:

1. Exchanges between route segments from the same depot. Here sequential customers on one route can be swapped with a route segment from another route. The sizes of the route segments to be considered for exchanges can differ. The sizes within an exchange between the two routes can also differ.
2. Exchanges between route segments from two different depots.
3. Swap two depots. A depot currently open in the current solution can be swapped with a closed depot as long as the minimum required depot capacity remains.
4. A single depot can change status in the current solution, meaning an open depot can be closed or vice versa, as long as the minimum required depot capacity is still adhered to.

5. Changing visit day combinations. Pirkwieser and Raidl (2010) recommend randomly removing visit day combinations and determining the best days in a greedy manner.

The shaking phase is followed by an intensification phase. Here only the routes changed in the shaking phase are considered. A 3-opt intra-route exchange is used to improve the routes. This is re-applied until no more improvements can be made to the current solution. Next a 2-opt inter-route improvement exchange is repeatedly applied for all routes to the same depot on the same day.

Pirkwieser and Raidl (2010) suggest using a hybrid version of the VNS, by introducing a cooling scheme similar to the simulated annealing metaheuristic. Solutions with a worse objective function value are sometimes accepted. The acceptance rate linearly decreases to nearly zero as the method reaches the last iterations.

The VNS is combined with three very large scale neighbourhoods (VLSN) searches based on linear programming. The VLNS tries to improve solutions by bettering a part of the solution using (in this case) linear programming techniques. The searches are as follows:

1. Solving a CFLP model with customers on the same route aggregated into single "supercustomers" as described in Prins et al. (2007). The integer linear programming (ILP) model is solved optimally using CPLEX. This will assign complete routes to depots and determine the depots to open.
2. This search is a more sophisticated version of the previous search. In this search whole routes are also kept as-is, but customers are not aggregated. The model is formulated as a set covering model and again solved optimally using CPLEX.
3. Solving the possible exchange of the sequence of customers in a given route optimally. Several methods were used to determine which customer sequences to pick, including picking customers with the greatest detour and customers furthest away from a seed customer on the route. Pirkwieser and Raidl (2010) settle for picking customers independently of the actual routes. The routes are again solved optimally using CPLEX.

Pirkwieser and Raidl (2010) gave a set of results based on the datasets provided by Prodhon (2009a). According to these results, they were able to better the best-known solutions for 29 out of the 30 datasets.

3.4.1.6 Solving the periodic component of the PLRP

Various methods have been proposed to solve the periodic component of the PLRP. For the iterative metaheuristic method, (Prodhon and Prins, 2008 and Prodhon, 2011), the problem is first solved as a CLRP. Routes that span over all the days in the time horizon are created. Next, the method tries to group customers that were frequently on the same routes on the same visit days. In the MA|PM, (Prodhon and Prins, 2008), customers are assigned to a fixed combination of visit days, based on their visit frequency. The method focusses on optimising the CLRP problem that results from these fixed days. Afterwards a diversification phase is used to move individual customers to a better combination of visit days if the capacity constraint is met and the objective function costs associated with the move are beneficial.

Prodhon expressed a need to deepen the search of the periodic aspect and suggested encoding the visit days in the chromosomes of an evolutionary algorithm, (Prodhon, 2009b). Building on this method, the author also suggests using an evolutionary local search (ELS) algorithm to better the visit day assignments in the ELS x path relinking hybrid algorithm. Path relinking is used to intensify the search with regards to visit day assignments. The same suggestion is made in the hybrid evolutionary algorithm by Prodhon (2011), where the ELS is used to better visit day combinations, followed by a local search similar to the diversification phase in the MA|PM method. Visit day combinations are selected randomly in the variable neighbourhood search coupled with ILP-based very large scale neighbourhood search method by Pirkwieser and Raidl (2010). During the shaking phase, they suggest randomly removing customers visit day combinations and determining the best days in a greedy manner.

3.5 Summary

A vast amount of methods have been suggested to solve the different variations of the CFLP. These methods include exact, heuristic and metaheuristic methods. Exact methods are not able to solve large scale instances, because it becomes too time consuming. Metaheuristic methods do not always provide fast answers for bigger datasets either, because of the diversification and intensification phases used to escape local minima. A few heuristic methods were also discussed. In the case of the p-median, the methods will have to be adapted by iterating through various numbers of clusters to find the best CFLP solution.

The CVRP is also a well-studied topic. Numerous methods have been suggested to solve the variations of the CVRP. The savings based and k-opt tour improvement are still very popular heuristics to solve the CVRP and routing component of the CLRP. The granular tabu search is also particularly interesting because it uses the shortest edges and only looks at 10% - 20% of all the edges in a complete graph.

Two quick TSP methods are also considered; the tree algorithm and the Christofides algorithm. These two methods are described by Papadimitriou and Steiglitz (1998) and guarantee solutions that are at most double and $3/2$ times the optimal solution respectively.

Although the discrete CLRP is also a well versed topic, there is very little work on variants like the HpMP, pILRP and sdCLRP in publish literature. A tendency in suggested CLRP methods is to break the problem into smaller subproblems and solve these subproblems either exactly or close to optimal by using techniques like the Lagrangian Relaxation and the saving based method with tour improvement.

Most of the literature addressing the PLRP were written by the same authors. The suggested methods are adapted from methods to solve the CVRP or CLRP. Methods addressing the periodic component of the PLRP and PVRP are limited to basic heuristic techniques and not a lot of work has been done in this area either.

With the latest drive to solve bigger problem instances, a lot of instances are based on adapted TSP library data and random generated instances.

Chapter 4

An analysis of clustering methods

In this chapter, the different clustering methods discussed in Chapter 2 are evaluated. The chapter starts by describing the datasets used and five measures that have been identified to measure the quality of clustering solutions. These measures as well as CPU time are used throughout the chapter to compare the different clustering methods that were coded and tested.

The discussion of the results starts with the SAHN clustering methods. This is followed by a discussion about the various iterative partitioning methods as well as their dependency on good starting solutions. Different graph-based, nearest neighbour and density-based methods with promising results and CPU times were tested as well. Hybrid methods have also been suggested in the past and a couple of promising methods are discussed further.

4.1 Methodology

Different clustering methods were coded and tested in this chapter. The solutions provided by the different clustering methods are plotted on a geographical map of South Africa using GoogleMaps (2005)* and illustrated throughout the chapter. In these figures, the points are represented by different symbols. Each symbol and colour combination represent a different cluster. In order to compare the clustering methods, four datasets were used. The computing environment is described in Section 1.3.1. The details of the datasets are discussed below.

*An example of how to display points on a map using the free GoogleAPI (2014) tool have been added in Appendix B.

4.1.1 Data preparation

Four instances were used to test the clustering methods. These can be described as follows:

- **DataS1 - DataS3**

A privately owned dataset of 7 681 geographical data points from a food distribution company in South Africa was used to conduct extensive tests and compare the clustering methods. The data points consist of two dimensional decimal Euclidean coordinates. In order to investigate the effects of outliers and the growth in CPU times, the dataset was divided into three instances (DataS1 - DataS3) of different sizes.

The datasets are listed in Table 4.1. For the different datasets the number of clusters to divide the data into was decided beforehand; this is also given in the table. All clustering methods used the same number of clusters per dataset so that the results are comparable. In all the figures, the clustering results of DataS3 using ten clusters were depicted, unless specified otherwise.

Dataset name	Number of points	Description	Number of clusters
DataS1	1 000	randomly selected points	5
DataS2	3 000	random points, including DataS1	5
DataS3	7 681	complete dataset	10

Table 4.1: The three derived datasets from a food distribution company used to compare the clustering methods.

- **USA13509**

A TSP dataset from the TSP library given by Reinelt (2015) containing 13 509 customers from the USA, (USA13509), was also used. After all the tests were completed the most effective methods with the best CPU times were chosen to cluster the USA13509 dataset. The dataset was clustered into 50 clusters.

4.1.2 Evaluation criteria

In order to compare clustering solutions, a measure of the quality of the solutions is needed. This can for example, measure the tightness of the clusters and can be used to compare a solution to other solutions quantitatively. The clustering criterion is often to minimise or maximise a measure of a clustering solution.

In Section 2.7, five different clustering criteria are discussed to measure the quality of a clustering solution. These five measures are used in this chapter to determine the best clustering methods. In all the cases the objective is to minimise the values. The measures are as follows:

- **SSR** - the sum of the squared residuals – Eq. (2.26)
- **MSR** - the mean of the squared residual – Eq. (2.27)
- **PD** - the partition diameter – Eq. (2.28)
- **ADM** - the average distance measure – Eq. (2.29)
- **AWCD** - the average within cluster distance – Eq. (2.30)

Two extra clustering performance measures are added to the list that will also be used. These are as follows:

1. CPU times

This refers to the computer run time needed for the clustering program to run to completion. Although the CPU times will differ per computing environment, this measure can be used to compare clustering methods to each other when ran on the same or similar environment. It can also be used as an indication of the time growth involved with bigger datasets by comparing the growth in CPU times from DataS1 to DataS3.

2. Visual attractiveness

The visual attractiveness of a solution is a concept defined by Poot et al. (1999). With the advances in visualisation of geographical data came the need for a solution to also look acceptable, (Kant et al., 2008). Typical acceptability rules for depots include that there should be clear customer to depot assignment regions with no overlap. Route attractiveness includes that customers should not be closer to the centre of gravity of another route than the one they are assigned to, and routes are not allowed to cross.

Visual attractiveness is not traditionally measured in the objective function and extra constraints are often built into programs to flag these violations and add penalty costs to the objective function to avoid these. This means that a solution with a lower objective function cost might be discarded in favour of a more acceptable one with higher costs.

4.1.3 Method

The different clustering methods described in Chapter 2 are tested in this chapter. In this section, the methodology used to code and test each of these methods are

described in more detail. Various techniques that are used to overcome slow CPU time and memory issues are discussed in Appendix C.

4.1.3.1 SAHN clustering

During agglomerative hierarchical clustering, each point starts in its own cluster and the clusters are merged until there is only one cluster. In order to create multiple clusters, stopping rules need to be implemented. To view the properties of each type of clustering, the stopping rule used in this section was to stop when the specified number of clusters were reached. This means that the clusters will grow naturally and will not adhere to any capacity constraints or limitations.

The SAHN methods were coded twice. The first time the clustering descriptions provided in Sections 2.1.1 – 2.1.7 were used. The second time the Lance-Williams parameters provided in Table 2.1 were used.

Where the Lance-Williams parameters are used, a distance matrix (Li et al., 2010) is used to store the calculated distances. Every customer is given a unique index number, $\{1, 2, \dots, n\}$. The initial distances between all customers are calculated beforehand and stored in this matrix. The index numbers are used to identify which distance is referred to. For example $d(x_1, x_2)$ returns the distance between customers x_1 and x_2 .

Every time two clusters are merged, the new cluster is allocated to the minimum index number of the two clusters being merged and the distance matrix is updated with this value. For example, if cluster 1 and 3 are to merge, the new cluster will be indexed as cluster 1. All distances between cluster 1 and all other clusters are recalculated. The distance calculation is based on the method's specific formulations, as given in Table 2.1. The initial calculation of the distance matrix between all points takes $O(n^2)$ time complexity, making the methods polynomial in time and much more acceptable than other clustering methods.

SAHN clustering is also used in hybrid clustering methods. Here the single or complete linkage methods are used to first create aggregated clusters (by clustering the data into small micro clusters) which are then clustered further using a second method. These hybrid methods are discussed in Section 4.3.6.

4.1.3.2 Iterative partitioning

The iterative partitioning methods have a rule to stop clustering if the cluster centroids converge, i.e. the centroid values stop changing more than a predefined parameter ϵ . In the case of the "k-" variants, this rule is difficult to implement as-is

because only one point is changed at a time and the impact is not always clear after a single iteration. It was therefore decided to rather store a copy of the cluster centroid values in memory and compare it after a predetermined number of assignments. In order to prevent cycling, the number of iterations are given a maximum limit.

In the first set of results the stepwise function, described in Section 2.2.7, is used to generate starting solutions. This function steps through the list of customers with even steps to generate starting solutions. All partitioning methods tested in this thesis were therefore run several times, each time with a different starting solution generated by the stepwise function. Each result is compared to the previous and the best solution is returned as the final solution. The same number of runs and starting solutions were used for the different partitioning methods, making the results of the methods comparable.

Because the stepwise function can generate a lot of starting solutions in the case of a big dataset and cannot guarantee that any one will result in a good solution, using the other clustering methods as starting solutions were also tested. Next the CLUSTER partitioning variants, which makes use of the furthest distance rule (FDR) to generate starting solutions, were also tested. These variants were described in Sections 2.2.9.

The CLUSTER partitioning method given by Jain and Dubes (1998) uses the h-means method to create clusters. Here the method was adapted so that it could use any of the partitioning variants to create clusters. The method was also modified to check two criteria when determining which clusters to merge:

- 1) The clusters with the shortest between-cluster distance were merged.
- 2) The cluster with the fewest points was merged to its closest cluster.

The method was repeated twice, using both merging criteria, and returned the best solution afterwards. The method also checked merging from two to p clusters. For all solutions, the method then incrementally adds a new cluster using the furthest distance point until p clusters are created again.

For the partitioning methods with long CPU times, the construction phase of the CLUSTER partitioning methods were modified to shorten the times. Here only one starting solution with the correct number of clusters were used instead of introducing more clusters one by one, as described in Section 2.2.9. The solution used the centroid as the first centre and the FDR to find the other seed points.

The CLUSTER partitioning method was also modified to look at other starting solutions. Here the construction part of the method, building a solution with two

to p clusters at a time, is replaced by the given starting solution. The method continues from this point in the same way as before by merging and adding centres of the furthest distance back in again.

Finally the AS 136 method, described in Section 2.2.10, was tested. This method can also use the stepwise function to generate starting solutions instead of random seedpoints. Similar to the other iterative methods the method was also tested with other clustering methods as starting solutions.

4.1.3.3 Graph-based clustering

For the three variants of the graph-based method, different combinations for the depth (d), standard deviation cut-off (σ_T) and edge length cut-off (l_T) were experimented with to get the correct number of clusters, but this proved to be a difficult task. All edges with values greater than $\sigma_T = 3$ and $l_T = 2$ were removed as suggested by Zahn (1971, p 33). Depending on the values of the parameters, the number of clusters varied and it was not always possible to get the correct number of clusters.

Because of the inconsistency in reaching the right number of clusters, it was decided to rather create a sorted list and merge edges, ordered by size until the desired number of clusters were achieved. Because it is not possible to use both the edge length ratios, Eq. (2.12) and z-scores, Eq. (2.13) when sorting, it was decided to use the edge length ratios, as this consistently provided better solutions.

A distance cut-off of all edges larger than the edge average was introduced because of the long CPU time and huge number of edges in the RNG graph-based method. This is similar to the granular threshold suggested by Toth and Vigo (2003). The cut-off was not implemented for the MST and GG graph-based methods because it either returned worse results or prevented the methods from reaching the desired number of clusters.

4.1.3.4 Nearest neighbour clustering

The nearest neighbour method was first coded using the k-near neighbours method as explained in Section 2.4. In this method, all customers closer than the k closest neighbour to a customer and where both share at least k_t near neighbours, are placed in the same cluster. It is not possible to specify the number of clusters beforehand and the value of k is chosen by trial and error to give results as close as possible to the required number of clusters.

The disadvantage of using the k-near neighbours method as-is, is that there can be

big jumps in the number of clusters created with consecutive k values. When using DataS1 for example, with $k_t = 2$, a k value of 4 created 6 clusters while a k value of 3 created 20 clusters.

The method was next adapted instead to increment k and continue clustering until the correct number of clusters have been created and was set to $k_t = 1$. The adapted method can be described as follows.

- 1) Start at depth $k = 1$.
- 2) Calculate all Euclidean distances between all customers and their k_{th} closest neighbour.
- 3) Sort the distances.
- 4) Merge customers from the closest neighbour distance to the largest.
- 5) Increment k .
- 6) Repeat Steps 2) - 5) until the predefined number of clusters has been reached.

The advantage of the adapted k-near neighbours method is that the value of k does not have to be specified for any sized dataset. The results are not the same as the single linkage method because not all inter-customer distances are considered simultaneously, but only the distances of the neighbours at one depth at a time.

Because the results were a bit disappointing compared to other methods, another change was tested; to leave out merging customers further than a specified cut-off distance, when placing them in the same cluster. This change did cause a slight improvement if only a small number of distances were eliminated, but as the number of exclusions became bigger, the solution deteriorated. It also caused a trial and error situation again, where different cut-off distances had to be tested until an improvement was found. Because the change did not show a significant improvement if at all and caused a trial and error situation, it was decided not to make this change to the method.

The MNV nearest neighbour method, discussed in Section 2.4.2, was also tested and it did indeed return the same results as the graph-based RNG method as mentioned by Jain and Dubes (1998, p. 91–92). This method also needs trial and error iterations and the specified number of clusters is unknown beforehand. Similar to the k-near neighbours, the method can also be adapted to find the exact number of clusters by incrementally increasing the depth parameter s until the number of clusters is reached. Because the MNV allows for two depth parameters q_1 and q_2 with $q_1 + q_2 \leq s$, the adapted method needs to cycle through all combinations of q_1 and q_2 for every s value. This meant that the number of iterations in order to let s

increase incrementally grows exponentially. This causes the CPU time complexity to also be exponential. Since the results are the same as the graph-based RNG, (Jain and Dubes, 1998, p. 91–92) and the method could also not return results in an acceptable amount of time, it was left out of the results.

4.1.3.5 Density-based clustering

Using the density-based k_{th} nearest neighbour clustering method as described in Section 2.5.1, points are clustered together if both are within the nearest k points of each other. Similar to the k-near neighbours clustering method, this method also has to be adapted to allow exactly k clusters to be created.

It was adapted by calculating all the distances of customer pairs that are within the nearest k points of each other. All distances were then sorted and merging started with the shortest distance until the desired number of clusters have been reached. This is similar to the single linkage method. The difference between this method and the single linkage method is that density distances are used, as defined by Eq. (2.19) instead of Euclidean distances.

4.1.3.5.1 Hybrid clustering density-based method

Wong (1982) used the k-means partitioning method to create aggregated clusters (micro clusters). The method is modified to use the stepwise h-means, h-median and k-median partitioning methods instead because these are much faster. The CLUSTER partitioning variations are also used to aggregate.

The customer points grouped in the same micro cluster in phase one are treated as a single customer point in the second phase. These points are clustered together incrementally in the second phase, calculating new cluster centres after every merge. The method is modified to look at three possible cluster centre calculations to give three variants of this method:

- 1) **Variant 1** – Use the biggest cluster’s centre.

When two clusters are merged, the biggest cluster’s centre is used as the centre of the new merged cluster, while the other cluster’s centre becomes obsolete.

- 2) **Variant 2** – Calculate the means of all points in the cluster.

The average longitude and latitude of all the points in the two clusters are used as the centre of the new merged cluster.

- 3) **Variant 3** – Calculate the weighted means of all points in the cluster.

The average of the two centres of the clusters becomes the centre of the new merged cluster - both cluster centres contribute equally to the new centre.

4.1.3.6 Hybrid clustering

An alternative approach in dealing with big datasets and reducing the time to find good starting solutions for partitioning methods, is to divide the clustering into two phases. This approach has been suggested by numerous authors like Min (1987), Han and Kamber (2006), Drezner and Hamacher (2001) and Everitt et al. (2011) for varying clustering methods and reasons. While some refer to it as the aggregation of data, others call it hybrid clustering methods. In the first phase the data is aggregated into smaller groups using one method. In the second phase the aggregated data is clustered further into the final number of clusters using a second method. All customer points in the same aggregated cluster are treated as one by the second clustering method.

An important decision, when it comes to aggregation, is to determine how much clustering should be done by the first method versus the second. In most cases a quick uncomplicated method that can deal with large datasets will be selected as a first method while the second method focuses on returning the best possible results. If the second method is known to return better results, it is preferable to let this method do as much of the clustering as possible. It makes sense to only let the first method aggregate until the problem has been reduced to a reasonable size for the second method to continue.

Three types of hybrid clustering combinations are tested.

- **Hybrid SAHN - SAHN clustering methods**

The SAHN methods have polynomial time complexity when used with the Lance–Williams parameters given in Table 2.1. In large datasets the Lance–Williams parameters of some of the SAHN methods, like the average linkage, centroid linkage and Ward’s method start running into data overflow issues.

One of the methods to overcome this problem is to first aggregate the data with another method like the single or complete linkage methods until the problem size is reduced. We refer to this as hybrid SAHN - SAHN clustering methods because SAHN methods are used in both phases. Because Ward’s method returned the best SSR results compared to the other SAHN methods, it is the method of choice to use for the second phase of clustering.

The data was tested using DataS3. Various number of clusters to use in the first phase were tested to investigate the impact of this variation on the solution. The data was first aggregated with the single linkage or complete linkage method followed by Ward’s method. In order to use the Lance–Williams parameters, the distance calculation values are stored in a distance matrix. In larger datasets the size of the distance matrix becomes too large to store. In Appendix C.2 alternatives are discussed to deal with too large distance matrix.

ces. We recommend decomposing the problem into subproblems, (Louis and Tang, 1999).

- **Hybrid SAHN - partitioning clustering methods**

In an attempt to eliminate the long CPU times of the k-means method, Min (1987) suggested making use of aggregation. This method first uses Ward's method to aggregate the data followed by the k-means method. The partitioning methods are also tested with various starting solutions, but the stepwise function seemed to be the most effective. Unfortunately, the stepwise function generates more starting solutions the bigger the dataset and can result in non-polynomial time complexity.

Based on the positive results from the hybrid SAHN - SAHN clustering methods, it was decided to test all partitioning methods in combination with both the single and complete linkage methods in phase one. In the second phase, the stepwise function was still used to generate starting solutions from the aggregated data, but because the data was aggregated the number of generated solutions was much less.

Instead of aggregating the data into micro clusters in the first phase, a second variant was tested. Here the points clustered into groups are used to generate aggregated points to use in the stepwise function. The difference with the previous method is that points clustered together in the first phase in this variant are not treated as single points by the second method. Instead, the number of starting solutions become less and the stepwise function does not generate solutions very close to each other (because these points were aggregated into micro clusters treated as a single point). This is referred to as the aggregated stepwise partitioning method.

- **Hybrid partitioning clustering methods**

These hybrid clustering methods are similar to the hybrid clustering density-based method because a partitioning method is used to create c aggregated clusters, where c is given by Eq. (2.20). Once the partitioning method has created the aggregated clusters, a second partitioning method is used to create the predefined number of clusters.

Unlike the hybrid clustering density-based method, distances are not calculated using a density function, but rather Euclidean distances are used.

4.2 Computational results

A summary of the results for all the clustering methods for DataS3 with the five measures and CPU times are illustrated in Figures 4.1 - 4.6. The detailed results for DataS1 - DataS3 are also given in Appendix A, in Tables A.1 - A.36. A few

clustering methods became too time consuming for DataS3 or were excluded for other reasons discussed in the respective sections. This is followed by the results for USA13509 in Tables A.37 - A.39, where only the methods with acceptable CPU times were tested.

The second column in the result tables gives the results of the clustering methods. In the case of the partitioning methods, the stepwise function is used to generate starting solutions. This function is explained in Section 2.2.7. All partitioning methods are therefore given the same number of iterations and the same set of starting solutions. The results of the best iteration are returned and noted in the second column.

The other columns in the results tables give the results when the solutions from the second column are used as starting solutions for the various partitioning methods. In these cases the partitioning methods are run only once with the given starting solution to see if the partitioning method can better the solution from the first method. In the tables containing the CPU times, the total time includes the time to get to the starting solution (second column) as well as the time to run the one iteration of the partitioning method afterwards.

The results for DataS3 are given separately in Table A.40. Various hybrid clustering methods have also been tested, the results are given as follows:

- **Hybrid SAHN - SAHN clustering methods**

The SSR results and CPU times are illustrated in Figures 4.37 to 4.38. The results are given in Table A.41.

- **Hybrid SAHN - partitioning clustering methods**

The results of these hybrid methods are illustrated in Figures 4.40 - 4.44 and in Tables A.42 - A.47.

- **Hybrid partitioning clustering methods**

The results are given in Tables A.49 - A.51.

4.3 Discussion

When comparing different clustering methods, it became clear that the methods that provide better well-balanced clusters show worse MSR results and vice versa. Han and Kamber (2006, p. 412) recommended using the number of points in a cluster, the SSR and AWCD measures to compare cluster solutions. They refer to these three measures as the clustering feature of a cluster and describe these as summary parameters representing the zero, first, and second moments of the cluster statistically.

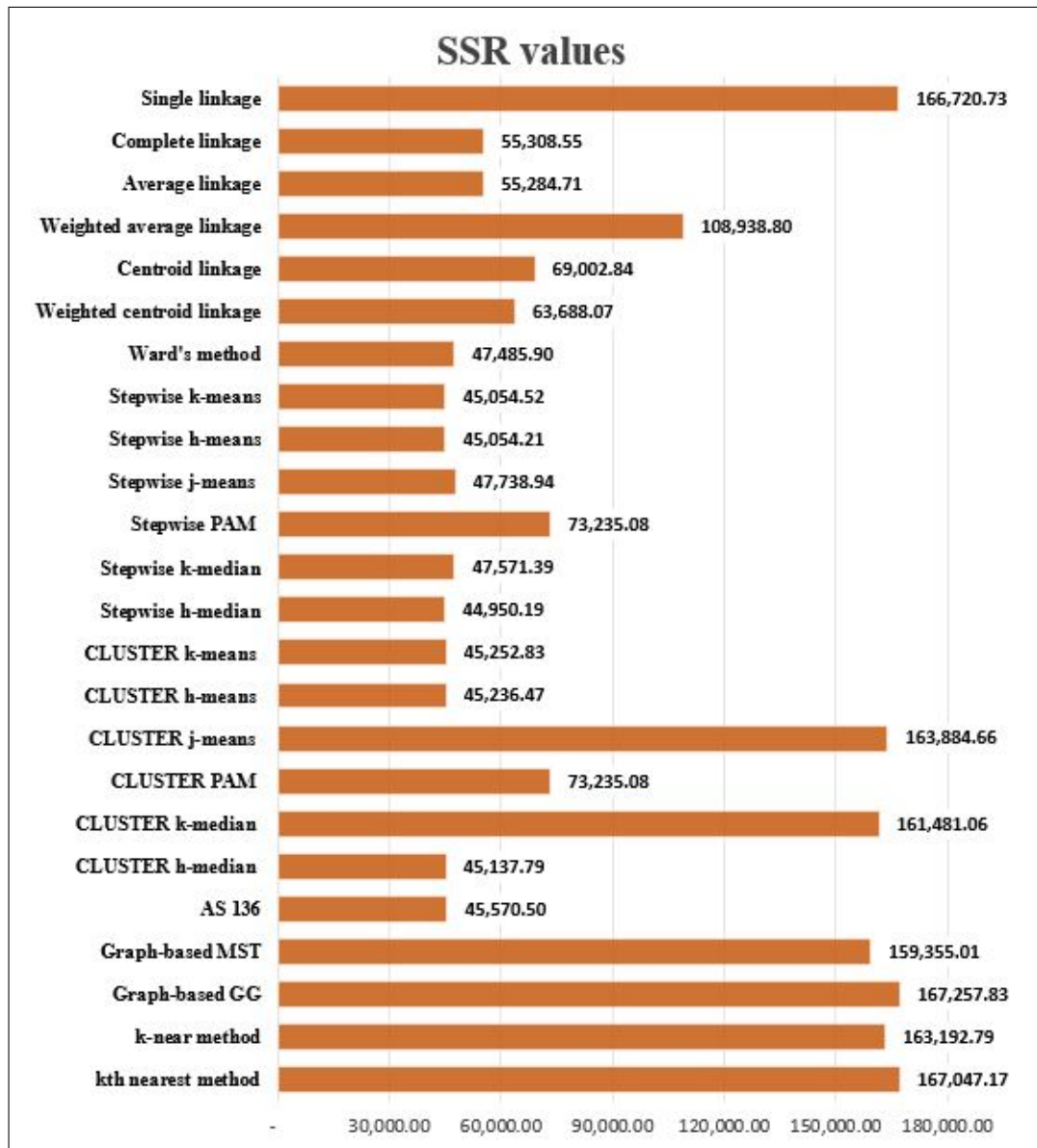


Figure 4.1: A summary of the results of the SSR measure for the various clustering methods using DataS3.

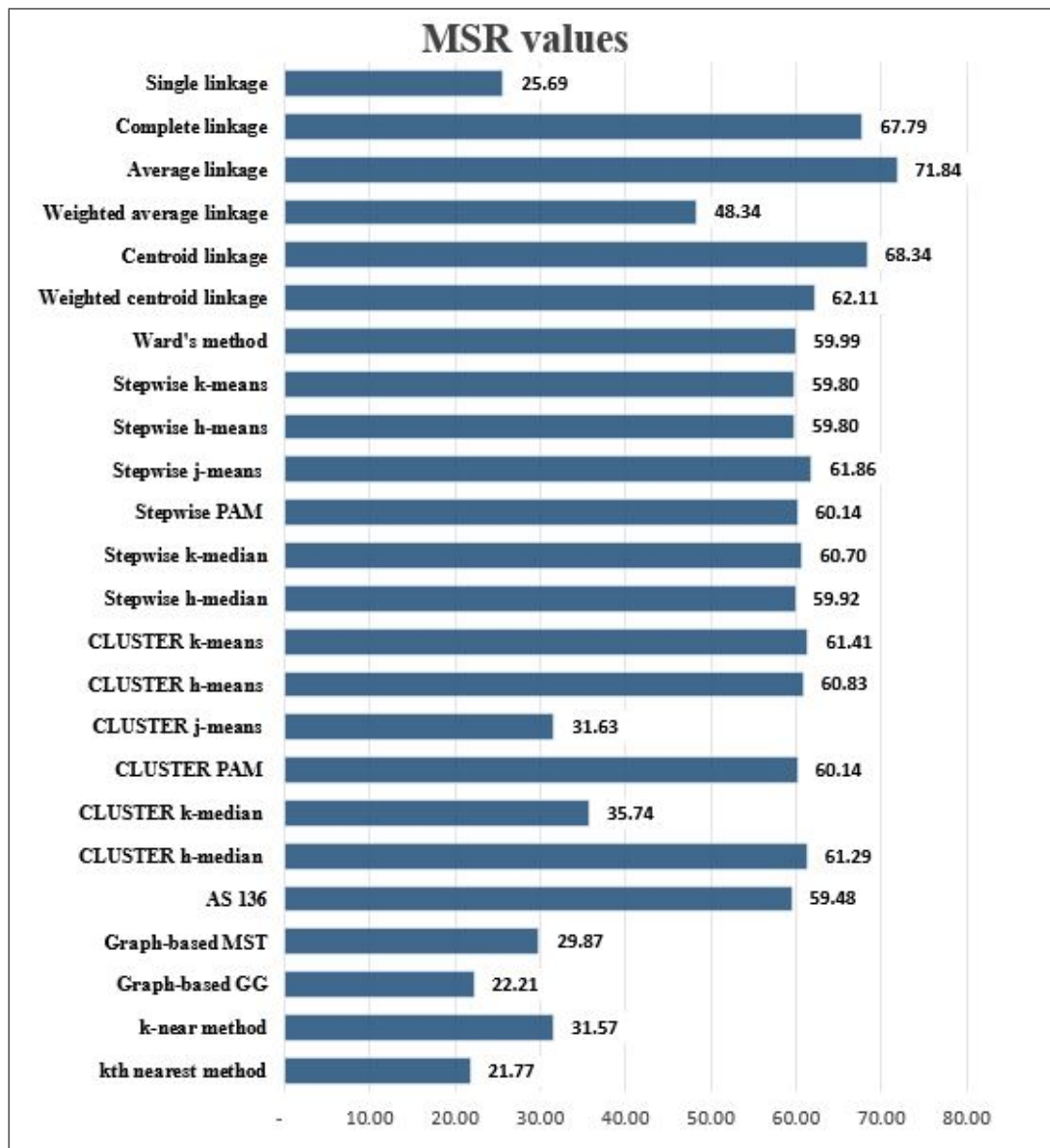


Figure 4.2: A summary of the results of the MSR measure for the various clustering methods using DataS3.

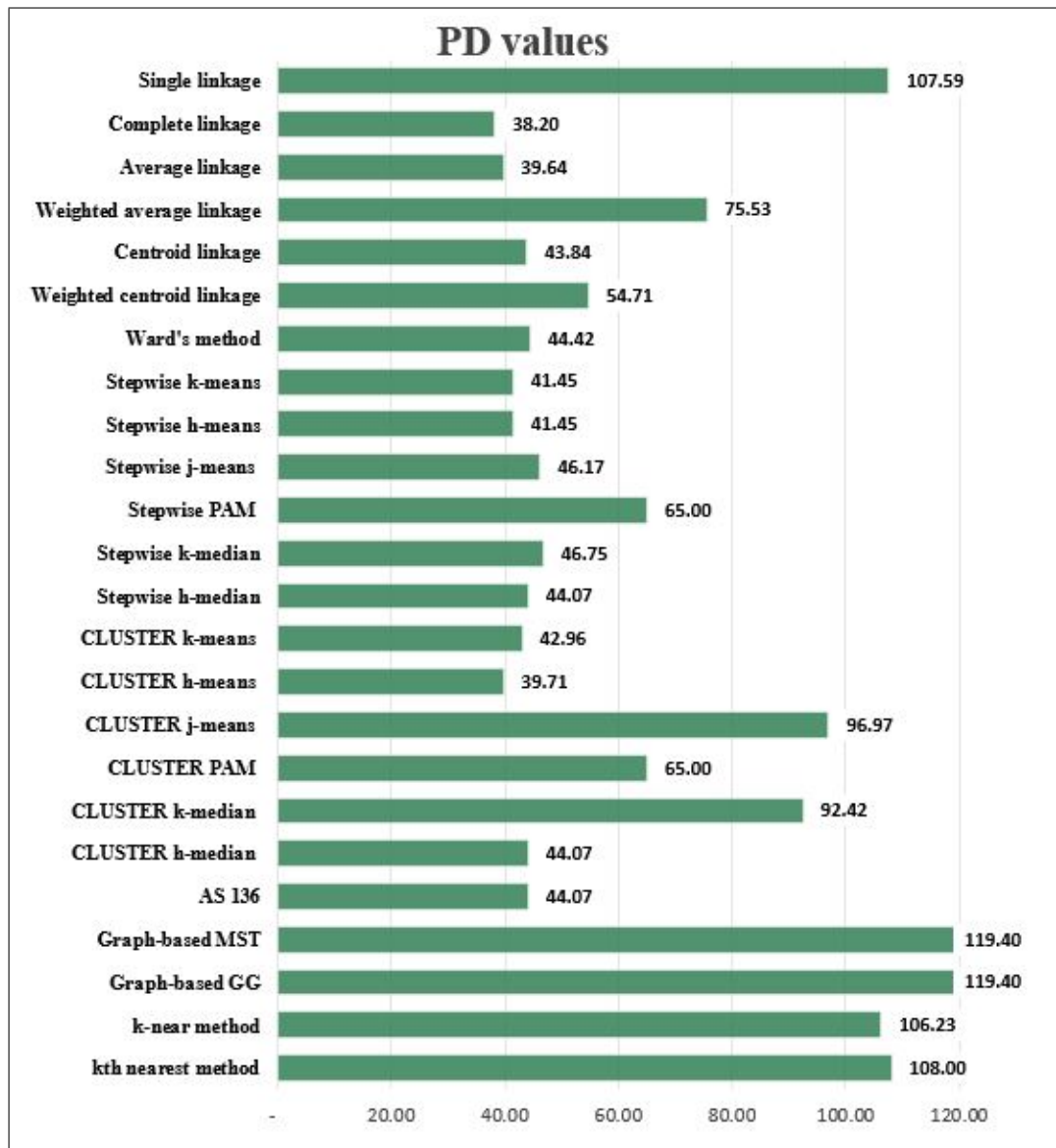


Figure 4.3: A summary of the results of the PD measure for the various clustering methods using DataS3.

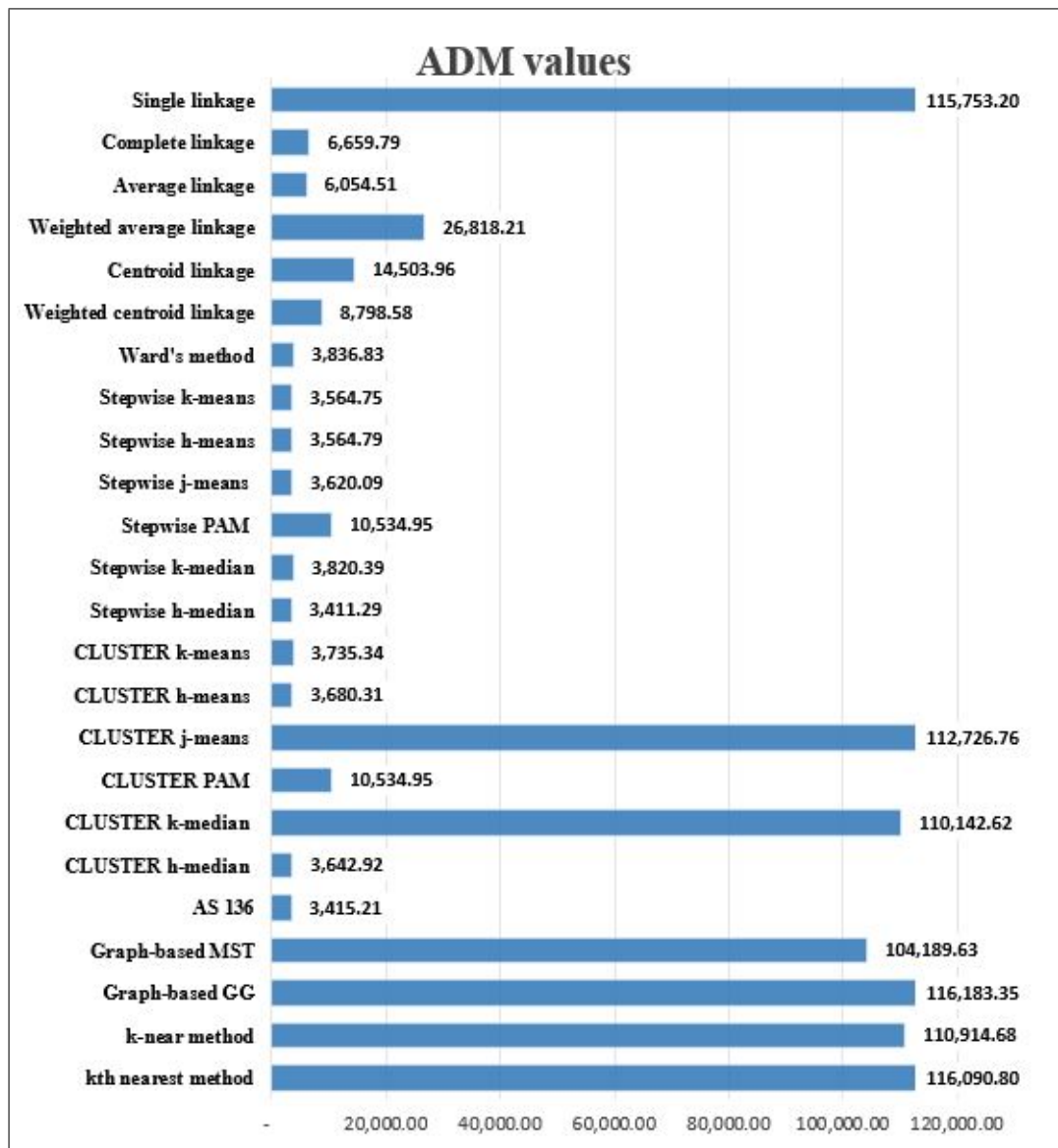


Figure 4.4: A summary of the results of the ADM measure for the various clustering methods using DataS3.

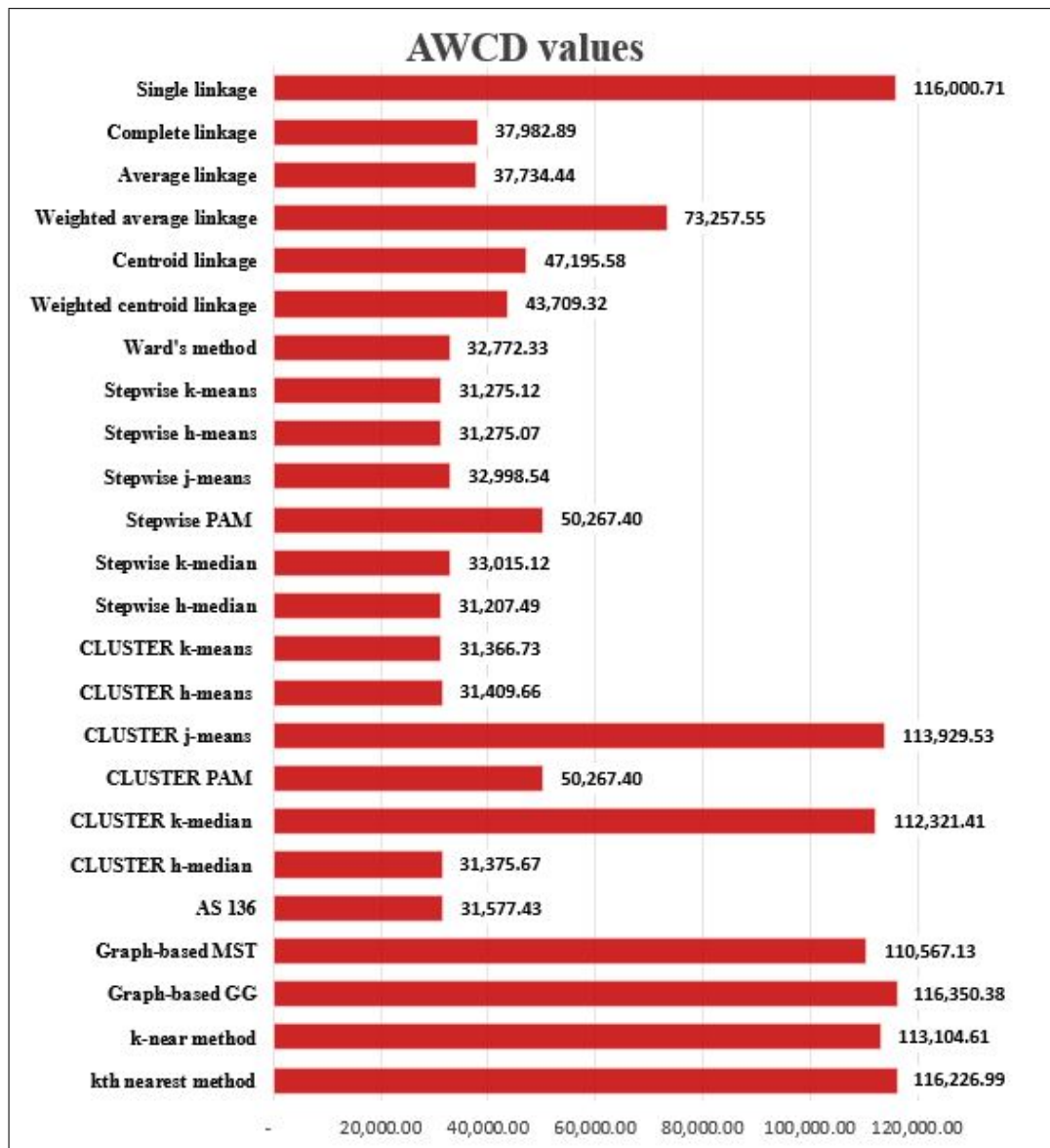


Figure 4.5: A summary of the results of the AWCD measure for the various clustering methods using DataS3.

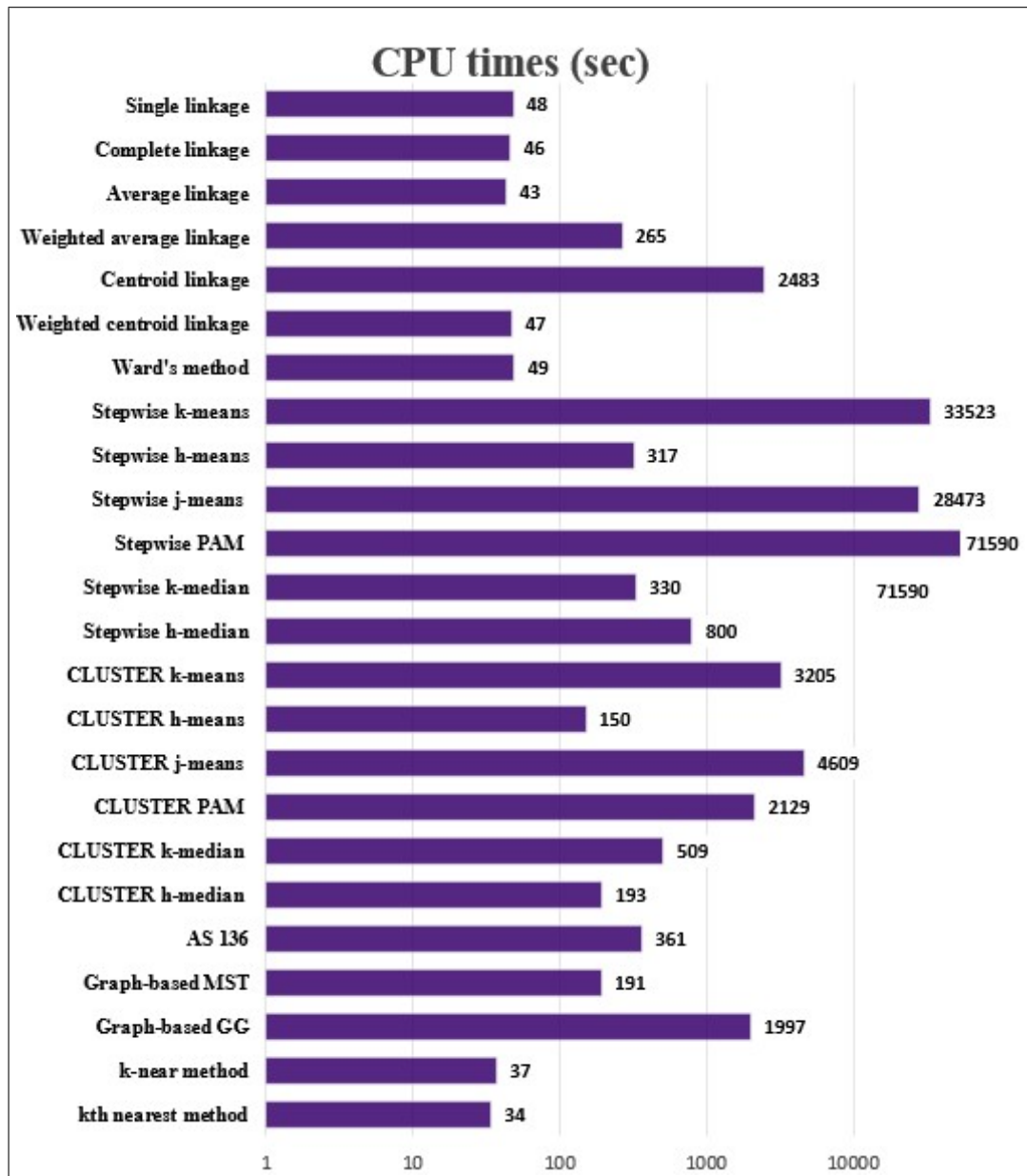


Figure 4.6: The CPU times for the various clustering methods using DataS3, up to 5 000 seconds.

The ADM measure gives an indication of how evenly balanced the clusters are. This is an important concept for creating capacitated clusters, that will be discussed further in Chapter 5. Although Brusco and Stahl (2005) refer to the MSR criterion as the "well-known k-means criterion" it is clear that this is not the same as the objective function given by Han and Kamber (2006) and Geetha et al. (2009). This is because there are no division of the number of points per cluster before summation and also becomes obvious in the results. The conclusion is therefore that the SSR, ADM and AWCD are better measures than the MSR for the type of clusters wanted for the LRP. The PD measure provides the maximum diameter for any cluster in the solution and although informative, cannot be used on its own to measure good cluster solutions.

The results from each clustering method are discussed next. In all the figures representing the different clustering methods, DataS3 were used with ten clusters and the distinction of a cluster is made by both the symbol and colour, as discussed in Section 4.1.1.

4.3.1 SAHN clustering

4.3.1.1 Single linkage

Single linkage is known to be sensitive to outliers. The method tends to create unbalanced partitions with small or single point outlier clusters and mega clusters in the dense areas. Even when obvious outliers are removed from a dataset, the method finds other outlier groups and still groups the biggest portion of points in mega clusters. This can also be seen in Figure 4.7, where the cluster represented by the points with red circles clearly dominates the map. The method is clearly not suitable for the location phase of the LRP.

Lam (2008) observed however that the single linkage method can create good clusters to minimise route distance, because of its string-like clustering. Results of the routing phase of the LRP are discussed further in Section 5.1.4.

When compared to other SAHN clustering methods, the single linkage method tends to provide worse results for all measures except the MSR measure. This is because the MSR ignores uneven cluster sizes by only looking at the average within cluster dissimilarities. Even though the method uses shortest distances to merge, it also has not provided good partition diameters compared to the other linkage methods. The method is however very fast and an excellent test for finding outlying points.

4.3.1.2 Complete linkage

The complete linkage method tends to create more compact clusters than the single linkage method, as can be seen in Figure 4.8. In the figure it can also be seen that outliers do not have such a big impact on the clustering as is the case with single linkage. The clusters are more balanced in terms of the number of customers per cluster, although the central red cluster is still much bigger than some of the other clusters. In the rural area on the right, only three customers were grouped together in the cyan group.

Except for Ward's method, this method gave the best PD and AWCD results compared to the other SAHN methods in all datasets tested. The ADM measure shows that the method does not tend to create well-balanced number of customers per cluster, but the SSR results are much more acceptable than the single linkage method.

4.3.1.3 Average linkage

Average linkage uses the average distance between all inter-cluster pairs and merge clusters based on closest average distance. The results of the average linkage method are illustrated in Figure 4.9. In the figure it can be seen that the method was able to create well rounded clusters that were more balanced than the complete linkage, but still contained a cluster (in cyan colour) with only four customers.

When coded according to the description, the average linkage method was the second most time consuming SAHN method, after Ward's method. This is because of the number of inter-cluster pair distance calculations required to compute the next two clusters to merge. When coded according to the Lance-Williams formulation, the method seemed less sensitive to outliers than the single linkage method and complete linkage method. On the tested datasets, the measures were in most cases comparable but not always better than the complete linkage method.

4.3.1.4 Weighted average linkage

The weighted average linkage is very similar to the average linkage method but each cluster contributes 50% each towards the average distance when two clusters merge. The results did not compare well to the other hierarchical methods. Figure 4.10 illustrates the results from the weighted average linkage method. In this figure it can be seen that the method has not created clusters with well-defined borders and even has some overlapping clusters. The method is also sensitive to outliers and has four big clusters while the others contain only a couple of points at the most.

Finally, it can also be seen in Table A.35 that the method is slower than all other SAHN methods in DataS3.

4.3.1.5 Centroid linkage

The results of the centroid linkage method are illustrated in Figure 4.11. In this figure it can be seen that the method has not returned well-balanced clusters, but the clusters are more evenly sized than the single linkage and complete linkage methods.

Using the Lance–Williams formula (given in Table 2.1) to code the method caused overflow problems for the bigger datasets. When code to recalculate the inter-cluster distances after every merge, the method took more than 40 minutes for DataS3 and it became clear that it is not a polynomial time algorithm. The SSR, ADM and AWCD measures did not perform particularly well, although the clusters were better defined than the weighted average linkage method. Because the method was slow compared to the other SAHN methods, it is not considered suitable for larger datasets.

4.3.1.6 Weighted centroid linkage

The results of the weighted centroid linkage are illustrated in Figure 4.12. In this figure it can be seen that although the method returned better defined clusters than the weighted average linkage, there is still a bit of overlap between the orange and blue clusters in the bottom right corner. There are also two small clusters in the top right and middle left with only a couple of points in them, indicating that the method does not always return well balanced clusters.

The weighted centroid linkage method is less sensitive to outliers than the centroid linkage method. Clusters with a small number of points are given more emphasis than in the centroid method. Since centroids start to form further away from the denser areas, the sizes of clusters are more the same and the solution is therefore better balanced. When looking at the SSR, ADM and AWCD measures, the weighted centroid linkage method always outperforms the weighted average linkage, but the average linkage method performs better in all three instances.

4.3.1.7 Ward’s method

The results for Ward’s method can be seen in Figure 4.13. The figure shows that Ward’s method is the least sensitive to outliers and the best at creating balanced clusters with the same number of points. The figure looks very similar to the average linkage method, but when comparing the results in Tables A.1 - A.36, it is clear that the method outperforms all other SAHN methods tested with respect to the SSR,

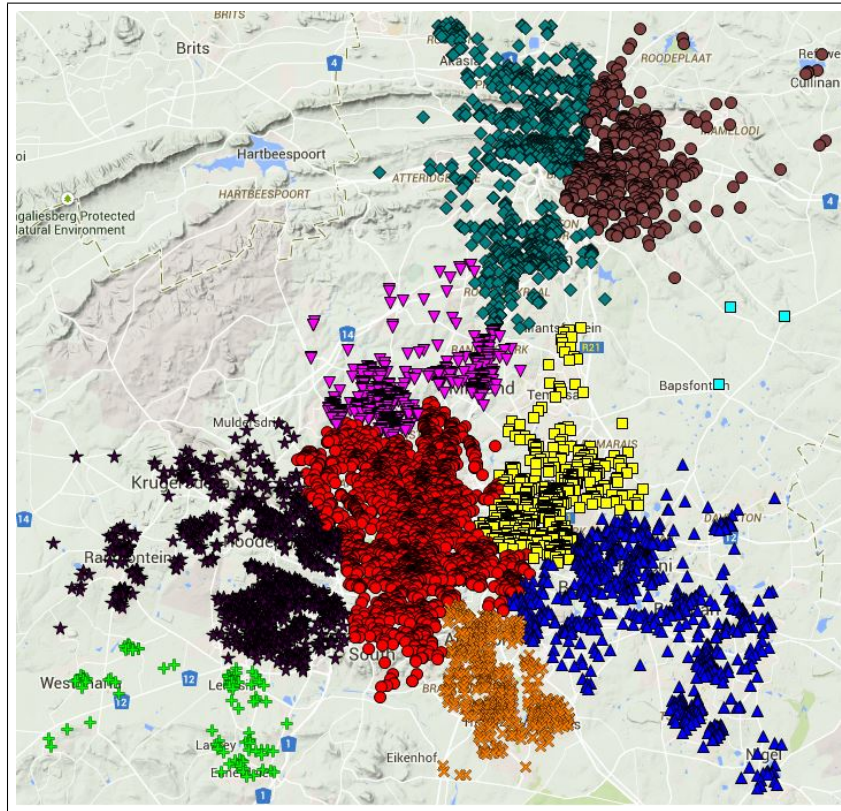


Figure 4.8: An illustration of the complete linkage method using DataS3.

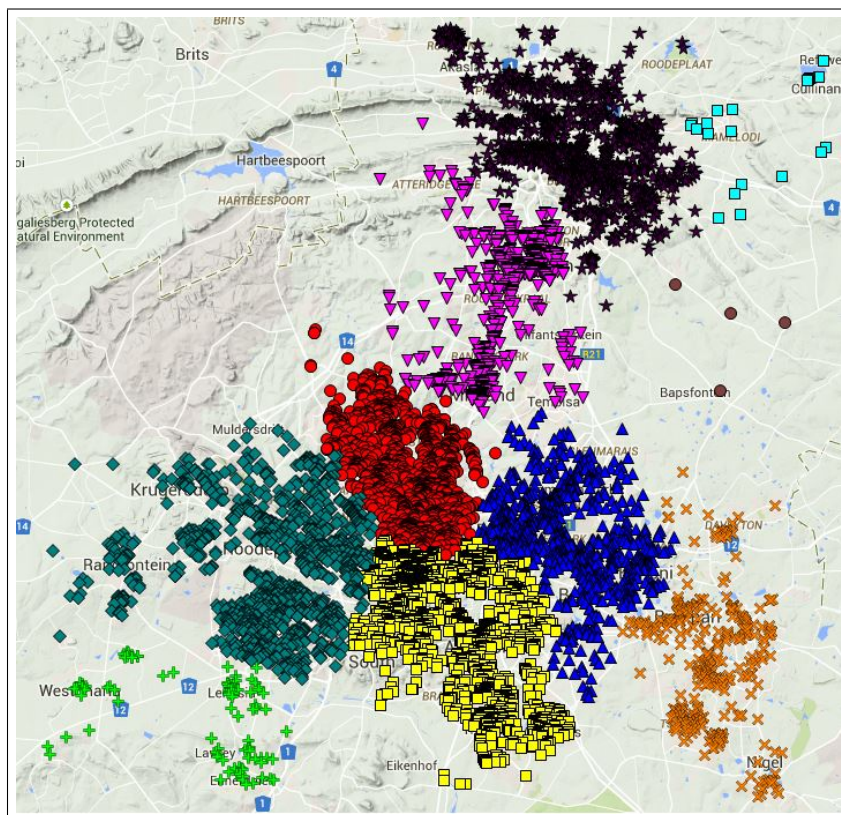


Figure 4.9: An illustration of the average linkage method using DataS3.

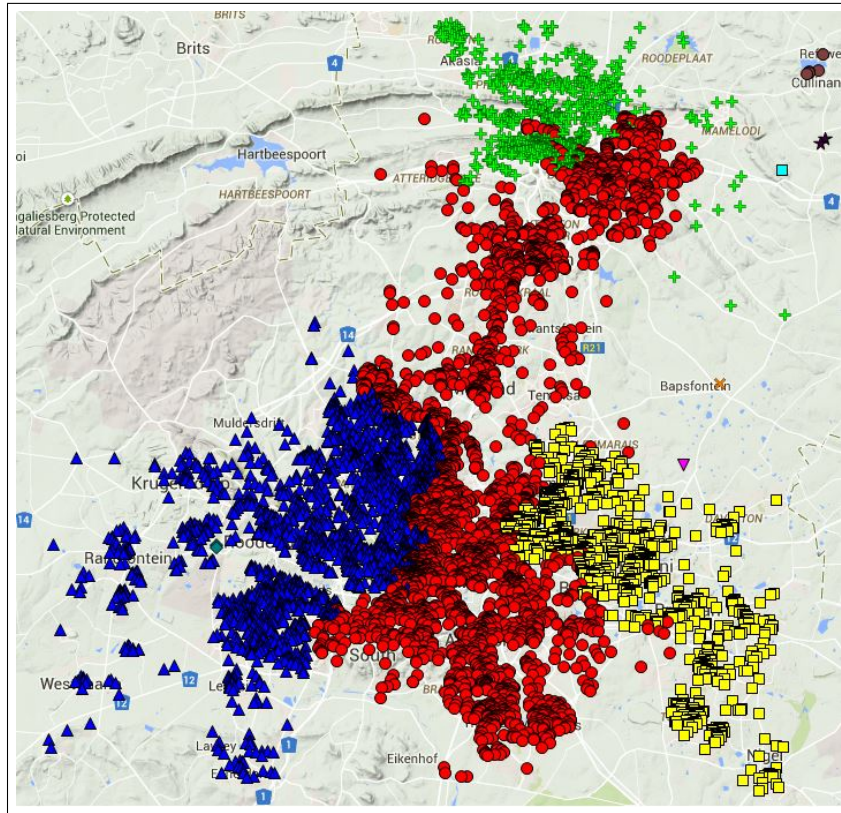


Figure 4.10: An illustration of the weighted average linkage method using DataS3.

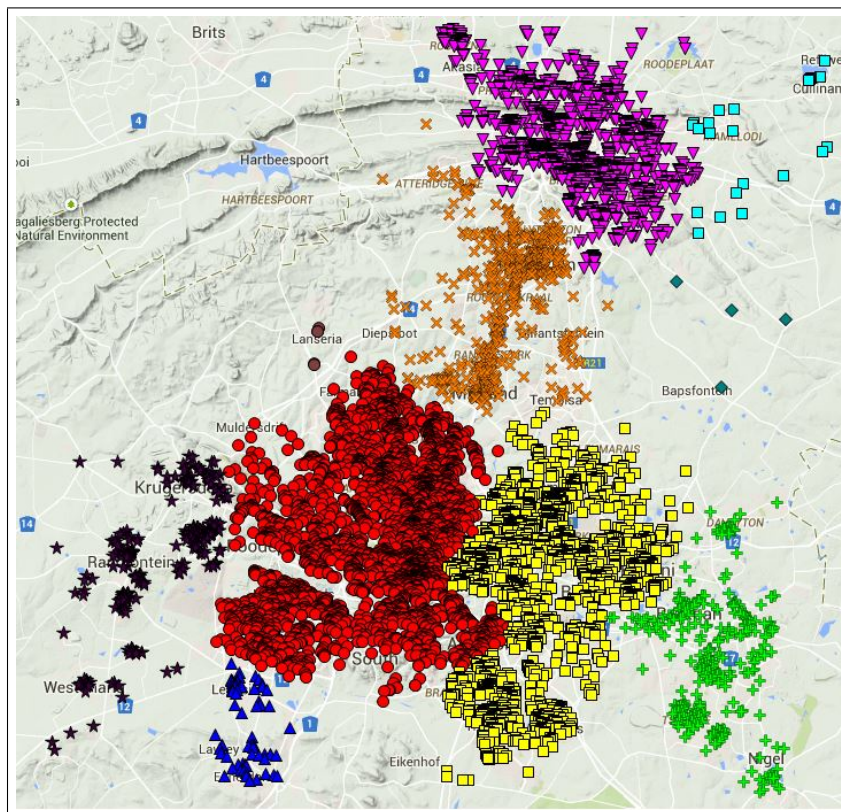


Figure 4.11: An illustration of the centroid linkage method using DataS3.

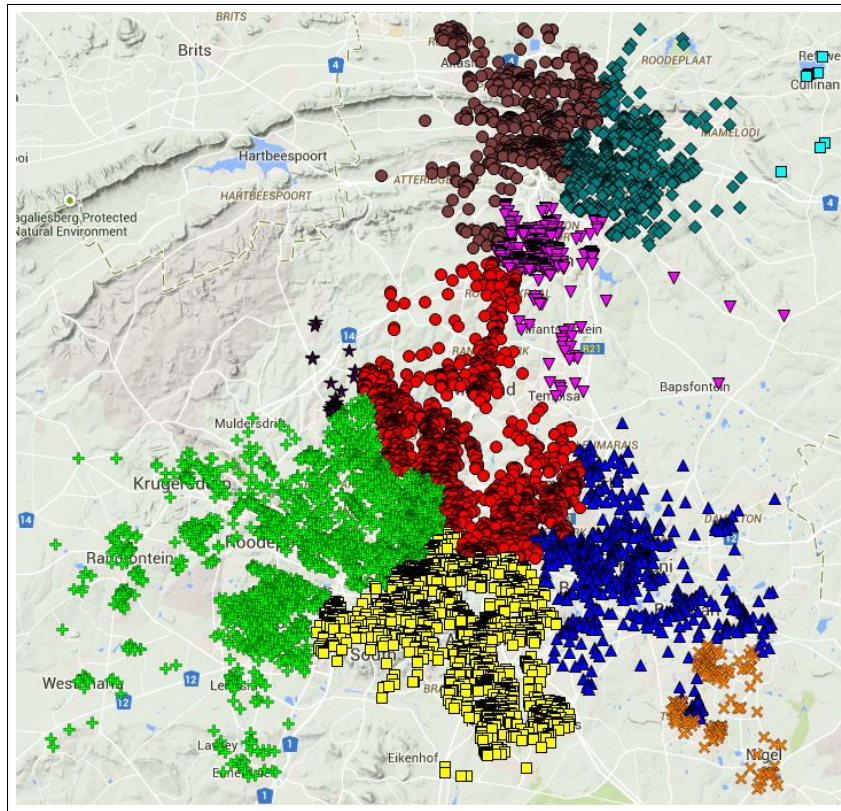


Figure 4.12: An illustration of the weighted centroid linkage method using DataS3.

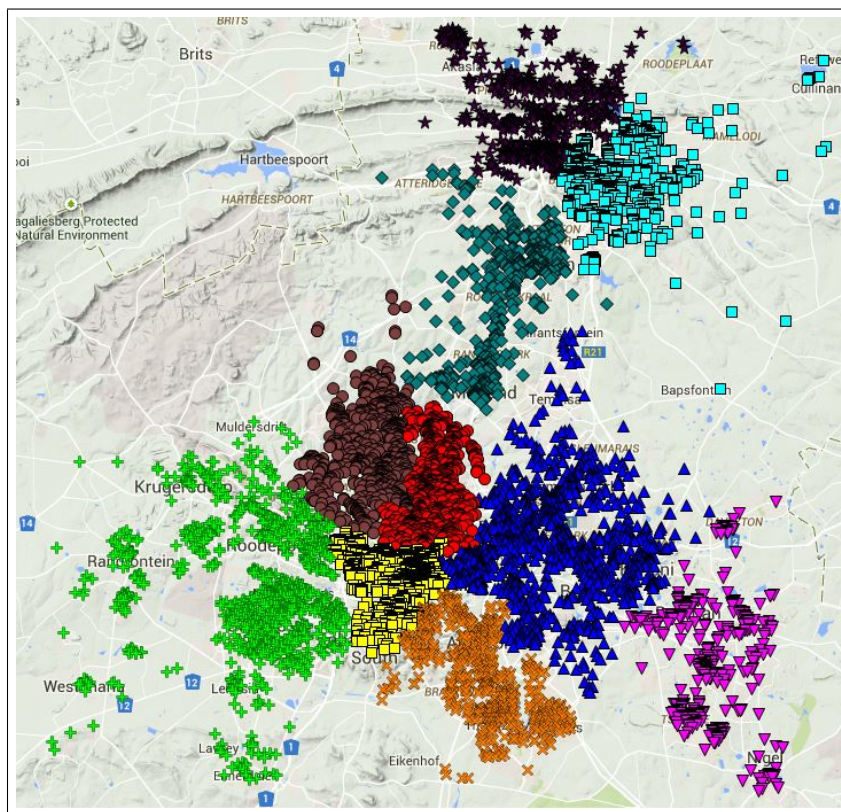


Figure 4.13: An illustration of the Ward's method using DataS3.

minimise the squared distances to the centroids. The complete linkage and average linkage returned similar results and returned in most cases the second best results for the SSR, ADM and AWCD measures.

The idea of calculating and merging the RNN (reciprocal nearest neighbours) as a means of bettering the CPU times for the SAHN methods was discussed in Section 2.1.9. The methods were tested with both merging RNN and without. The impact of merging the neighbours did not produce a significant time saving for the three datasets and was at most two seconds less. The clustering results are also dependent on the order in which RNNs are merged and do not necessarily guarantee an improvement. In the above datasets for example, the results were, for most methods, a bit worse than running the method without this. Because the impact was so insignificant and does not guarantee better results, it was decided not to merge the RNNs.

4.3.2 Iterative partitioning clustering

As describe in Chapter 2, iterative partitioning methods do not cluster sequentially, but need p seed points as input, where p is the number of clusters required. All customer points are assigned to the closest seed point to create an initial solution, called a starting solution. Different starting solutions were generated using the stepwise function and other clustering methods were also used as starting solutions for the iterative partitioning methods as described in Section 4.1.3.2. Once a starting solution has been created, the method iteratively calculates the cluster centres and re-assign points to the new cluster centres, where the calculations used depends on the chosen partitioning method.

4.3.2.1 The k-means method

As expected the k-means method is very slow and took 9 hours and 18 minutes to complete, as shown in Table A.35 and Figure 4.6. Figure 4.15 illustrates the results of the k-means method using a stepwise function to generate starting solutions. The method creates well balanced clusters with no overlap. Unfortunately, the time to complete makes the method unacceptable for big datasets.

4.3.2.2 The h-means method

The results of the h-means method are shown in Figure 4.16. When compared to the k-means solution in Figure 4.15, there are only slight differences between the brown circle and blue triangle clusters at the bottom left corners. When comparing

the results in Figures 4.1 - 4.6, the results differ only slightly for all measures except the CPU time. The h-means completes in a CPU time of 5 minutes, 17 seconds. This is a significant improvement in time and indicates that there is no real benefit for using the k-means method rather than the h-means method.

Hansen and Mladenović (2001) suggest that the k-means method will always return better results than the h-means. As can be seen in the results in Tables A.1 - A.36, this assumption depends on the measure and is not always the case.

The hk-means method

The hk-means method first runs the h-means method until it converges and then the k-means. Hansen and Mladenović (2001) suggested that we should be able to improve the results produced by the h-means method by using them as starting solution for the k-means. To test this method we identified solutions from our previous results that had worse SSR results for the h-means method than the k-means method.

For DataS2, the AS 136 and k-near methods starting solutions were tested. In both cases, the hk-means returned the same results as the h-means method. For DataS3 the k_{th} -nearest and Gabriel graph based methods were used. Here the hk-means method also returned the same solution as the h-means solution. These results suggest that it does not help to run the k-means method after the h-means method because the solution has already converged to a local minimum. The hk-means method was therefore not used in any further datasets.

4.3.2.3 The j-means (k-medoid) method

As expected, the j-means method is time consuming, because the sum of squares using each point as the centre within a cluster, needs to be calculated before the new cluster centre can be chosen. Like the k-means method, the method is not scalable to big datasets, as can be seen from the CPU times of 7 hour and 54 minutes for DataS3 in Table A.36.

Figure 4.17 depicts the results of the j-means method using DataS3. The clusters all look well balanced and the only critique found was that the blue triangle cluster could have included a couple of the red circle cluster points in the middle bottom. Han and Kamber (2006) stated that the j-means method is very effective and less sensitive to outliers than the k-means method. When however comparing the results of the measures of the j-means method to the results of the k-means and h-means solutions, in Figures 4.1 - 4.6, this does not seem to be the case. Even on the tests done later in this chapter, where different starting solutions were tried, the k-means and h-means method consistently returned better results for the SSR, ADM and

AWCD measures than the j-means method.

The j-means method seems to also be much more sensitive to starting solutions as can be seen in the bad clustering solution produced by the CLUSTER j-means method in Figure 4.24. The results from the CLUSTER partitioning methods are discussed in more detail in Section 4.3.2.9.

4.3.2.4 The PAM (h-medoid) method

The partitioning around medoids method (PAM), also called the h-medoid method, is similar to the j-means method described above. In the PAM method, all points are re-assigned between iterations and the best medoid for all clusters are chosen simultaneously.

Even though the PAM method makes more changes per iteration than the j-means method, the method seems to consistently take longer to converge than the j-means method. The method was illustrated in Figure 4.18. The method creates an interesting pie-chart type of clustering. This tendency was depicted in all the illustrations of the solutions created by the PAM method, shown in Figures 4.14, 4.25 and 4.28.

Compared to other partitioning methods, the method returned worse SSR, ADM and AWCD results. Clearly the method is not a good clustering method for large-scale datasets.

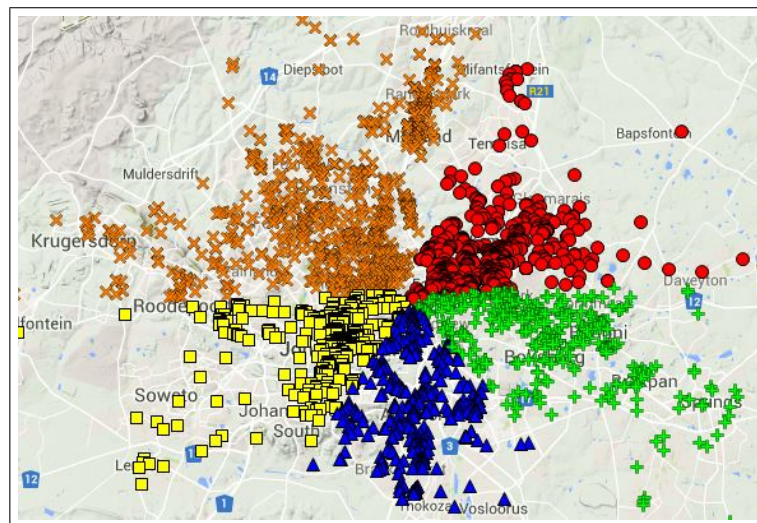


Figure 4.14: The results of the PAM clustering method using DataS2.

4.3.2.5 The k–median method

Figure 4.19 illustrates the k–median method. In this figure it looks like the cluster borders are defined using straight lines. This makes sense if the median values are used to re-assign customers. When comparing the clusters formed in this figure it also has a pie-like tendency, although not as much as the PAM method described in the previous section.

The k–median method has a CPU time of 5 minutes and 30 seconds and is the only partitioning method where only one point is re-assigned at a time, that has a polynomial time complexity. When comparing the results of the different measures to the other partitioning methods, in Figures 4.1 - 4.6, the method seems to get results very similar to the j–means method. Similar to the j–means method, this method also seems to be quite sensitive to starting solutions as can be seen when looking at the results of the CLUSTER k–median method.

4.3.2.6 The h–median method

The h–median method calculates the median after all points are re-assigned. For the SSR, ADM and AWCD measures the k–median did not return results as good as the h–median on average. For the MSR and PD measures, the results were inconclusive. When compared to other partitioning results, the h–median method gave similar results as the k–means and h–means, but is much quicker than the k–means. The results between the h–means and h–median methods were very similar in the three datasets.

On the other hand, the CPU times of the h–median are slower than the h–means method. This is because the method takes longer to converge. This is seen in the results for DataS3 in Table A.36, where the h–median method took 13 minutes and 20 seconds to complete, compared to the 5 minutes and 17 seconds and 5 minutes and 30 seconds of the h–means and k–median, respectively.

It is also interesting to note that the h–median method took longer than the k–median method. Since both methods used the same starting solutions and the same type of calculations, it can only be assumed that the method takes longer to converge to a final solution. On average the CPU time per repetition was 1 second or less for both the k–median and h–median, but when used together with the stepwise function to generate the starting solutions for multiple repetitions, k–median had longer CPU times. This indicates that the h–median is a good refinement method, but should not be repeated with too many starting solutions.

This method is illustrated in Figure 4.20. From this figure it is clear that the method

created well balanced clusters, similar to the k-means and h-means methods. When comparing the measures for the different methods as illustrated in Figures 4.1 - 4.6, the h-median method found the best SSR, ADM and AWCD values of all the variants tested for DataS3.

It is therefore clear that this is the best method to use to create the smallest SSR values. A downfall however is that the method seems to take longer than the k-median and h-means methods. Where these methods were used as starting solutions, the h-median was not able to find the same minimum SSR value. In Section 4.3.2.7 various ways to identify better starting solutions rather than using the stepwise function to generate starting solutions are investigated.

4.3.2.7 Starting solutions for iterative partitioning methods

In Section 2.2.7, it was highlighted that the quality of the results of the partitioning methods depends greatly on the initial solution. In order to identify a "good" starting solution, solutions from all other clustering methods were tested as starting solutions for the partitioning methods. The partitioning methods tested were: k-means, h-means, j-means, PAM, k-median and h-median. The CLUSTER partitioning variants of these methods were also tested. The results are given in Appendix A. The tests were again repeated for all three datasets and the five measures used above were used as the measuring criteria.

When comparing the SSR, ADM and AWCD values, the following conclusions became apparent: the h-means, h-median and CLUSTER h-median methods seemed less sensitive to different starting solutions and returned overall better solutions compared to the other solutions. The AS 136 method as a starting solution returned good results in combination with most partitioning methods. There were times where the stepwise function still returned better solutions, particularly for the k-means, h-means, h-median and CLUSTER h-median methods. The results of the two k-medoids methods (the j-means and PAM) were on average worse than the other methods, even though all methods used exactly the same starting solutions.

The h-median method consistently outperformed the other methods with the same starting solutions and was also able to produce good results for most starting solutions regardless how good they were. For example, Figure 4.21 illustrates the seed points used to create a starting solution that resulted in the best SSR value found for DataS3, in Figure 4.20. This solution was one of the starting solutions generated by the stepwise function. Interestingly, the seed points in Figure 4.21 are very closely located to each other and the starting solution did not return particularly good values. This indicates that the h-median method is less dependent on good starting solutions to produce good results than the other partitioning methods.

When comparing CPU times, the average linkage, centroid linkage, weighted centroid linkage and Ward’s SAHN clustering methods are slow and not good methods to use as starting solutions. This is consistent with the report that the SAHN methods has a time complexity of $O(n^2 \log(n))$ (Everitt et al., 2011). The two graph-based methods: (graph based RNG and graph based GG) are also too time consuming.

When used in conjunction with the stepwise function, some of the partitioning methods also do not fare well. This is because the number of generated starting solutions provides the input for the number of times the method is repeated, ρ . The number of repetitions is therefore $\rho = n/p$, where p is the number of clusters, n the number of points. Han and Kamber (2006) reported the time complexity for the k-means and h-means methods as $O(npt)$ per repetition, where t is the number of iterations in a repetition. When used in conjunction with the stepwise function, the CPU time then becomes $O(n^2t)$. Since the k-means only allows one move per iteration, the value of t is much larger than it would be for the h-means method.

The CLUSTER PAM method took much longer than any of the other partitioning methods to converge for all starting solutions. The time complexity for this method is given by Han and Kamber (2006) as $O(p(n-p)^2)t$. It is therefore interesting that the PAM method took longer on average than the j-means method.

4.3.2.8 Methods to prevent the number of clusters from declining

The two methods to prevent empty clusters from forming discussed in Section 2.2.8 were tested:

1. The restriction on the minimum number of points assigned to a cluster caused overlapping or unequal sized clusters.
2. Creating new clusters using the FDR (furthest distance rule) gave better results. It was a very handy method to create good solutions from bad starting solutions, where two or more clusters are too close together, for example when using the stepwise function to generate starting solutions. Another advantage is that other methods that produce less clusters than the specified number, like the graph-based or nearest neighbour methods, can be used as starting solutions. This is because the FDR will create extra clusters to ensure the solution has the correct number of clusters. All the results given in this thesis used the FDR when new clusters had to be added in the partitioning methods.

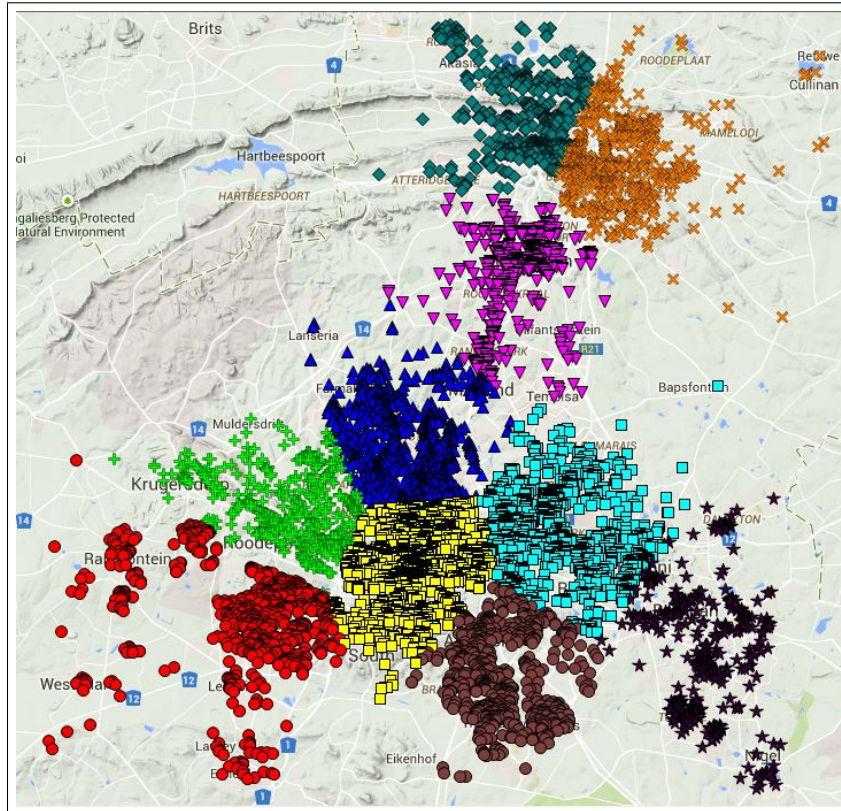


Figure 4.15: An illustration of the k-means partitioning method using DataS3.

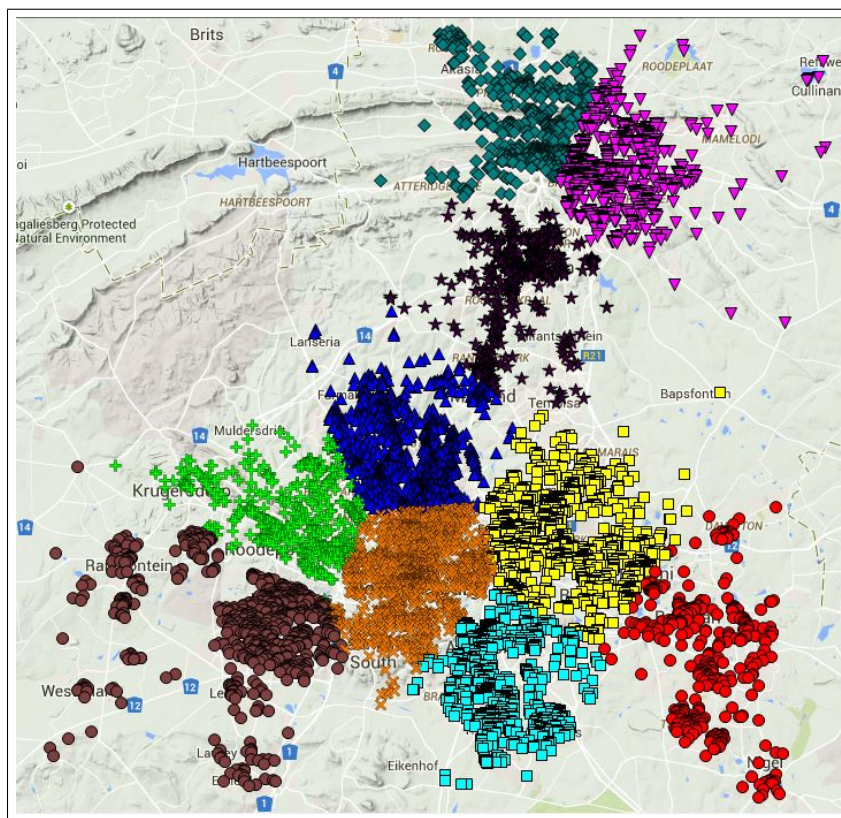


Figure 4.16: An illustration of the h-means partitioning method using DataS3.

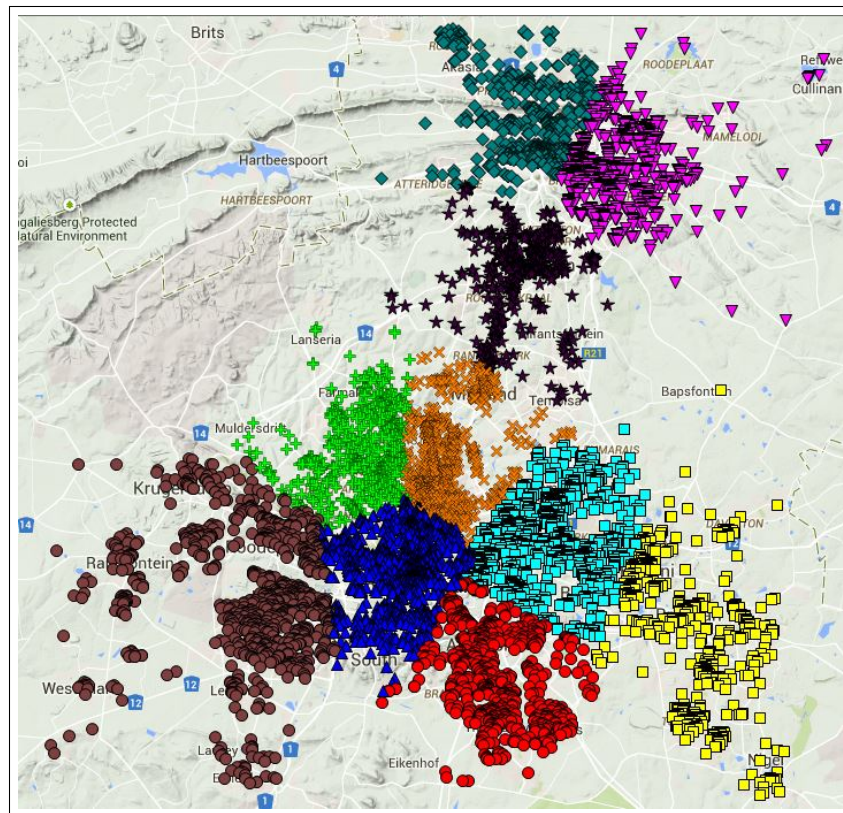


Figure 4.17: An illustration of the j-means partitioning method using DataS3.

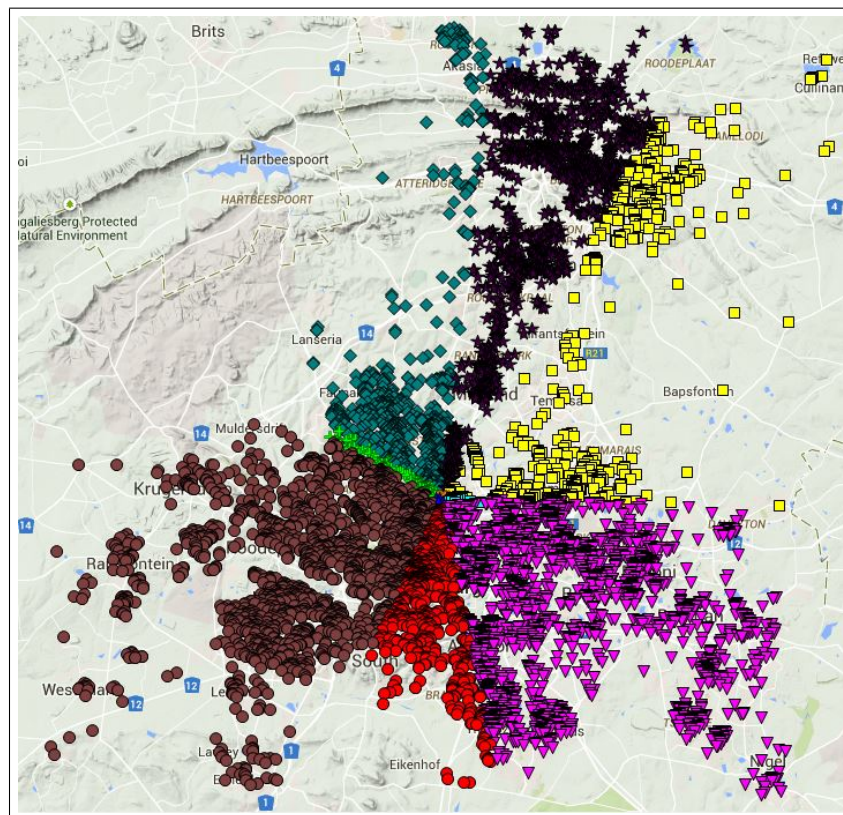


Figure 4.18: An illustration of the PAM (h-medoid) partitioning method using DataS3.

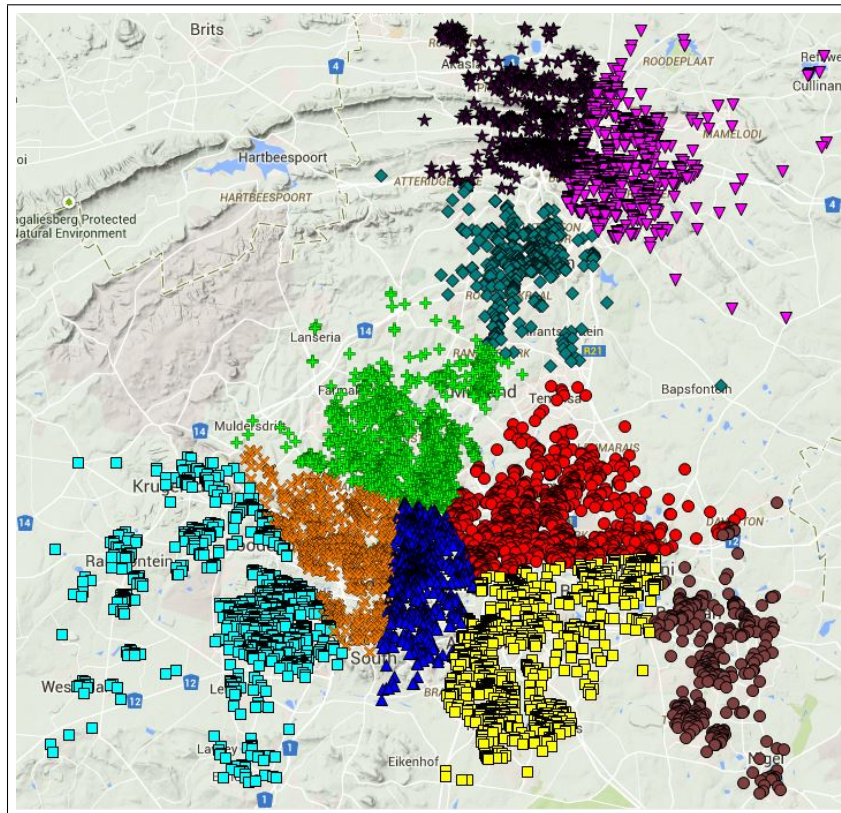


Figure 4.19: An illustration of the k -median partitioning method using DataS3.

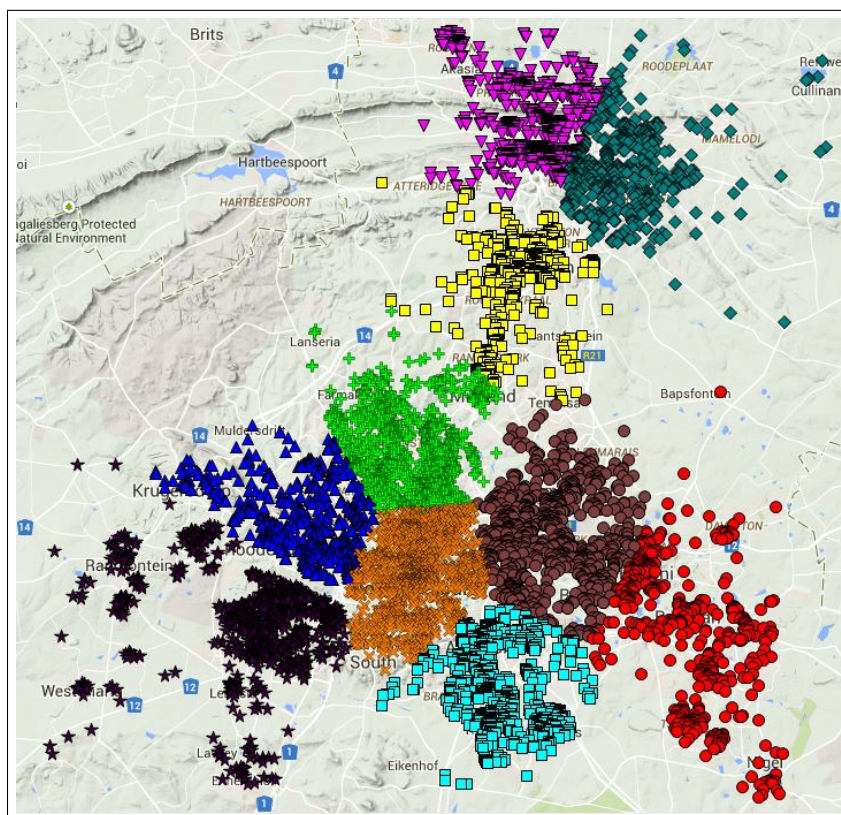


Figure 4.20: An illustration of the h -median partitioning method using DataS3.

4.3.2.9 The CLUSTER partitioning method

From the results, it is clear that using the central point as a starting solution does not give as good results as other starting solutions found by using the stepwise function, although the h-median method for DataS1 returned the same best answer as the stepwise function. The CLUSTER partitioning methods are illustrated in Figures 4.22 to 4.27. The CPU times for the different CLUSTER partitioning methods are given in Appendix A for DataS1 - DataS3. Figure 4.28 also illustrates the interesting pie-chart division typically created by the PAM method.

The results of using CLUSTER partitioning methods with the other methods as starting solutions are shown in Tables A.1 – A.39. Here it can be seen that the results are very similar and sometimes better than the starting solutions, because the CLUSTER variants have been programmed to return the starting solution if it could not better it's SSR value. This is also the reason why the CLUSTER variants was able to return the best SSR solution found by the stepwise h-median method in DataS2 and DataS3 (Tables A.13 and A.25).

The CLUSTER variants also did not outperform their partitioning counterparts. As expected, the computation time of these methods are also longer than the counterparts, because the methods consist of more cycles.

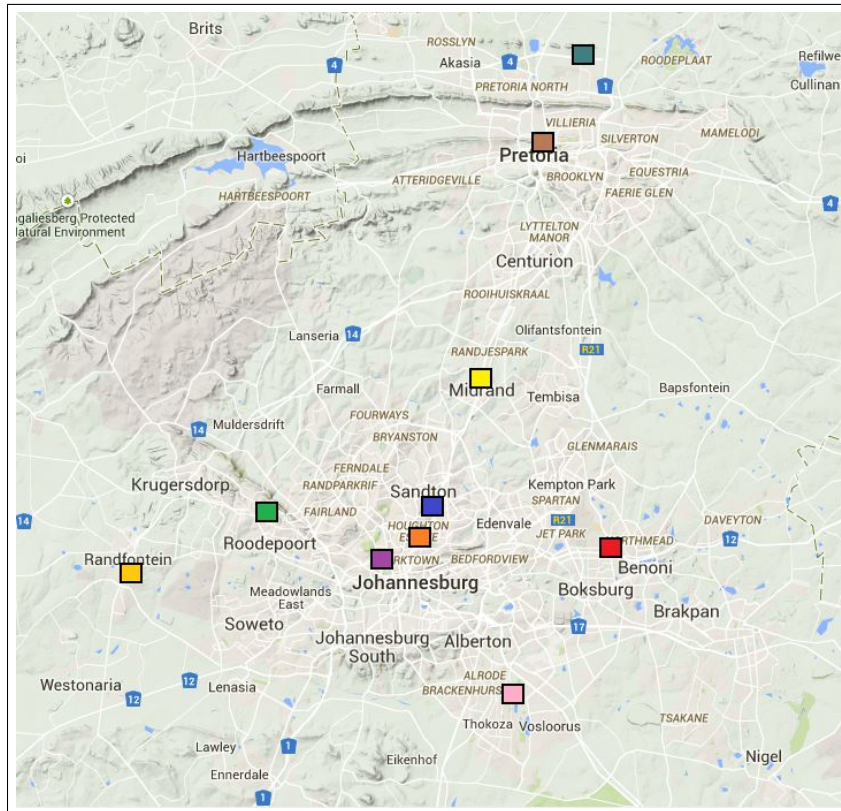


Figure 4.21: An illustration of the best h-median starting solution using DataS3.

4.3 Discussion

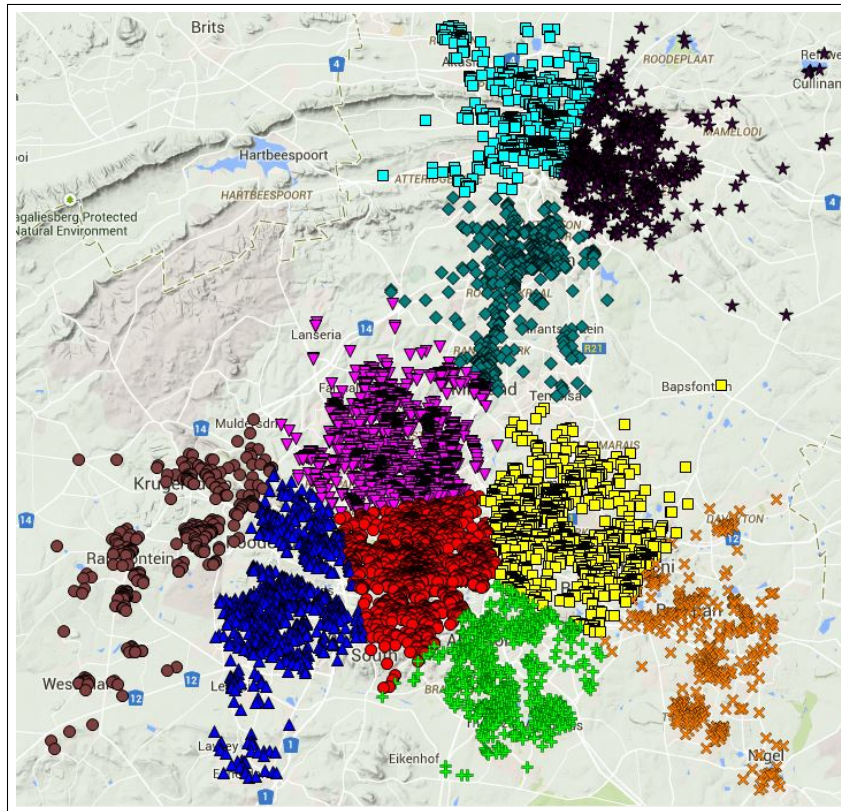


Figure 4.22: An illustration of the CLUSTER k-means partitioning method using DataS3.

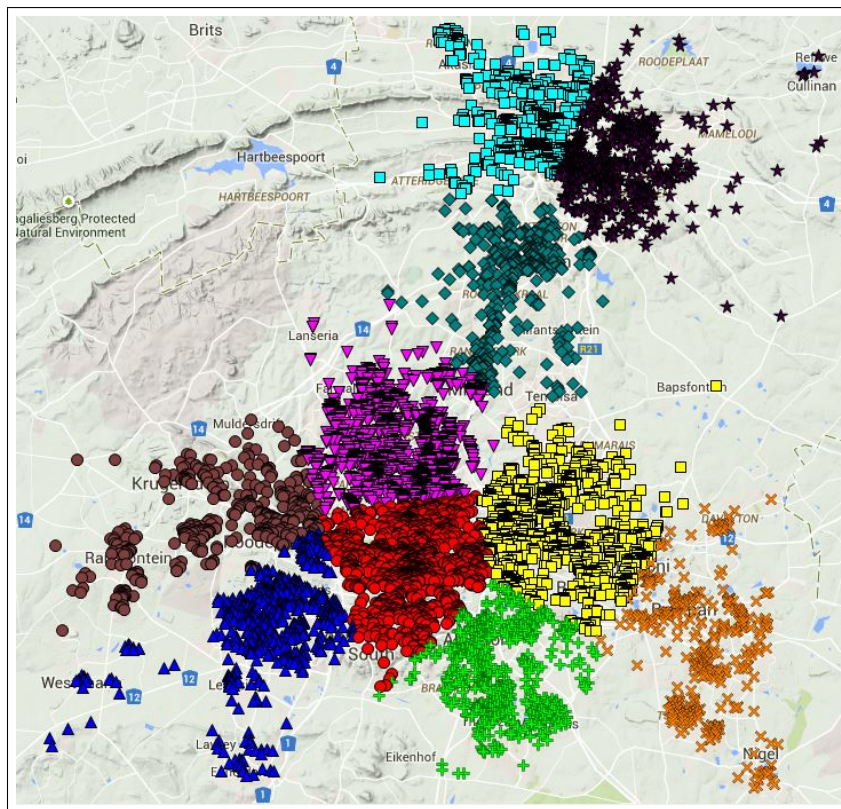


Figure 4.23: An illustration of the CLUSTER h-means partitioning method using DataS3.

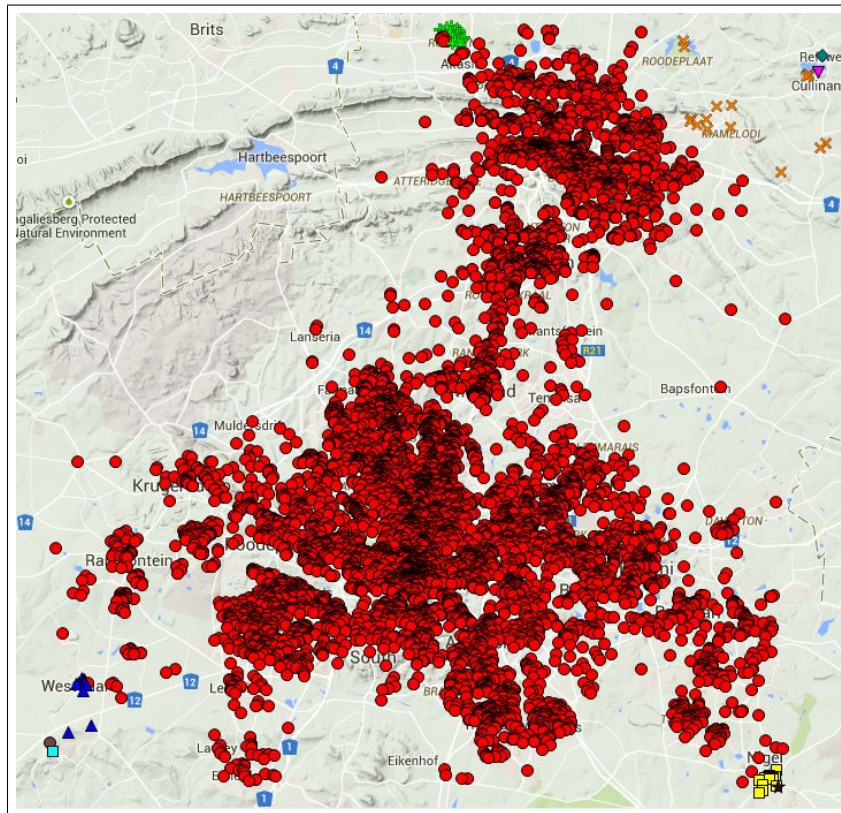


Figure 4.24: An illustration of the CLUSTER j-means partitioning method using DataS3.

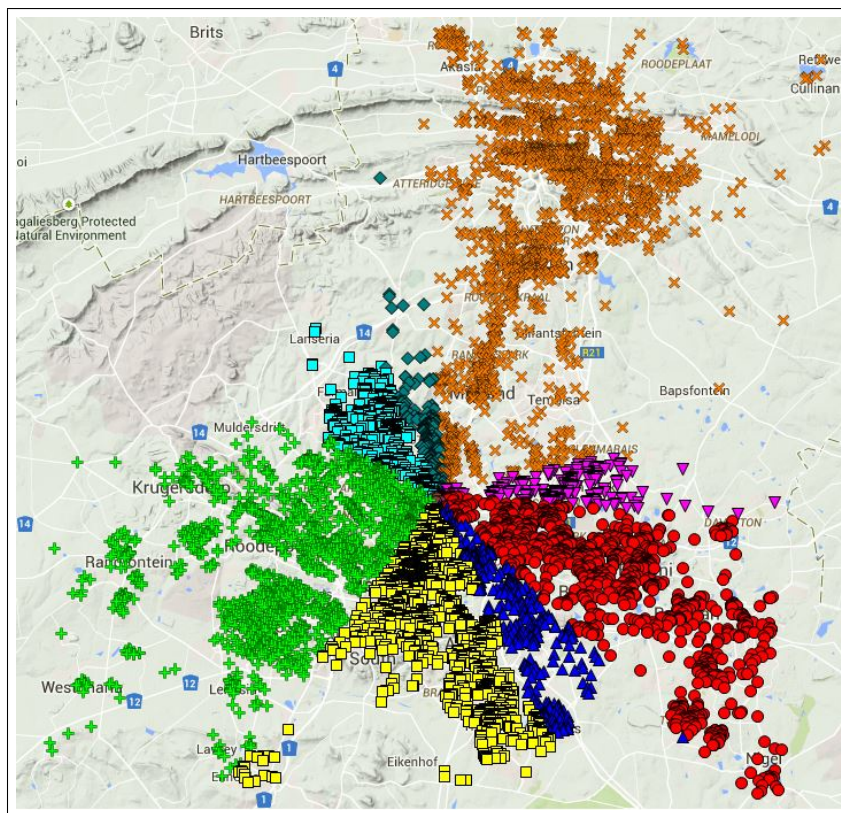


Figure 4.25: An illustration of the CLUSTER PAM partitioning method using DataS3.

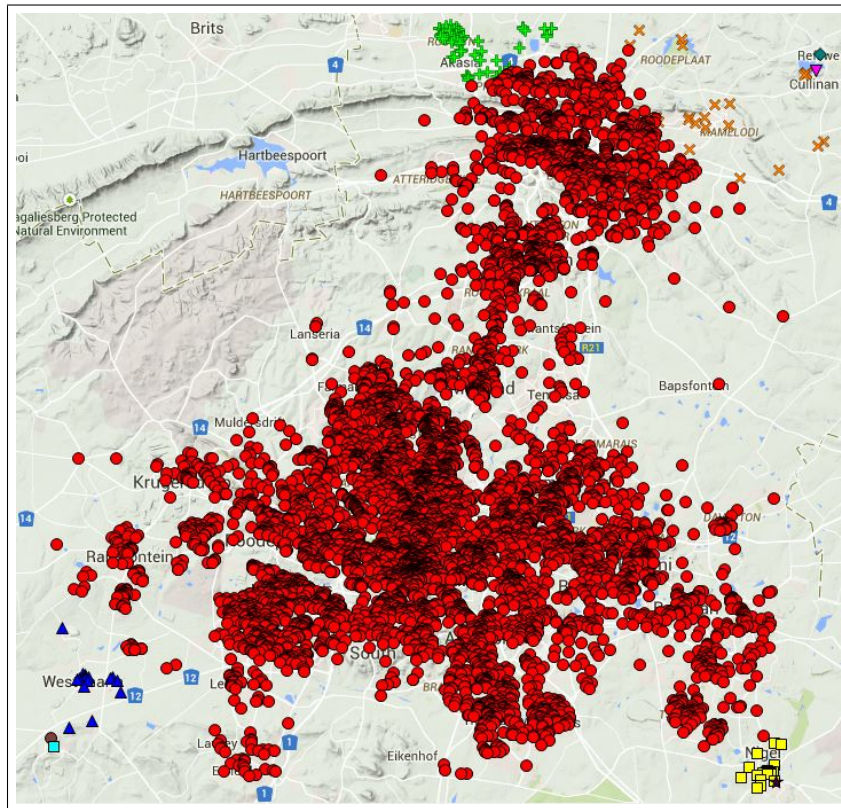


Figure 4.26: An illustration of the CLUSTER k -median partitioning method using DataS3.

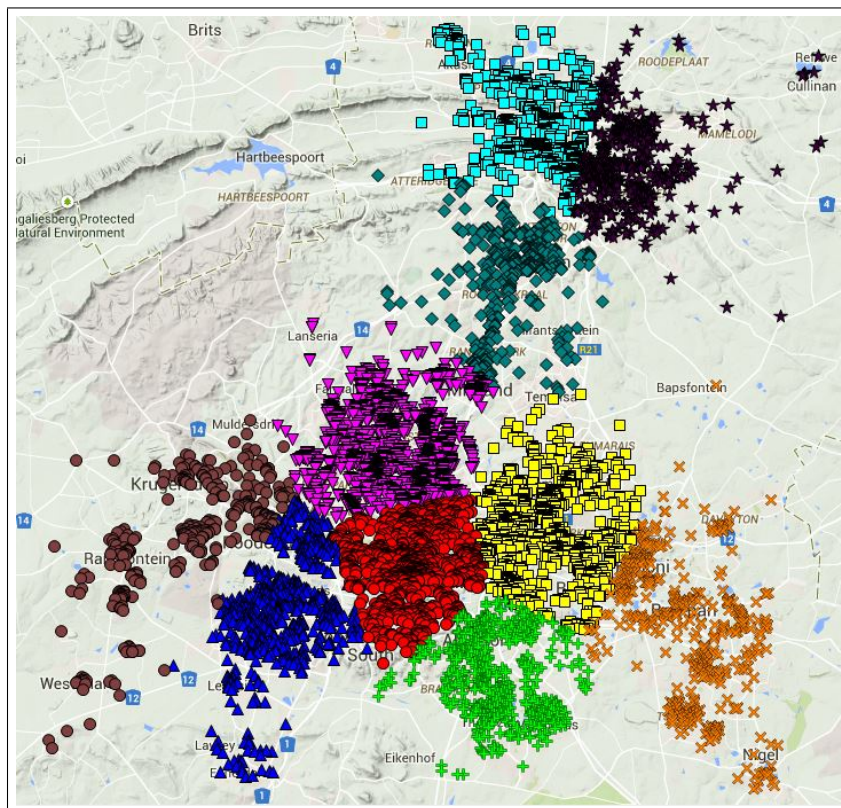


Figure 4.27: An illustration of the CLUSTER h -median partitioning method using DataS3.

4.3 Discussion

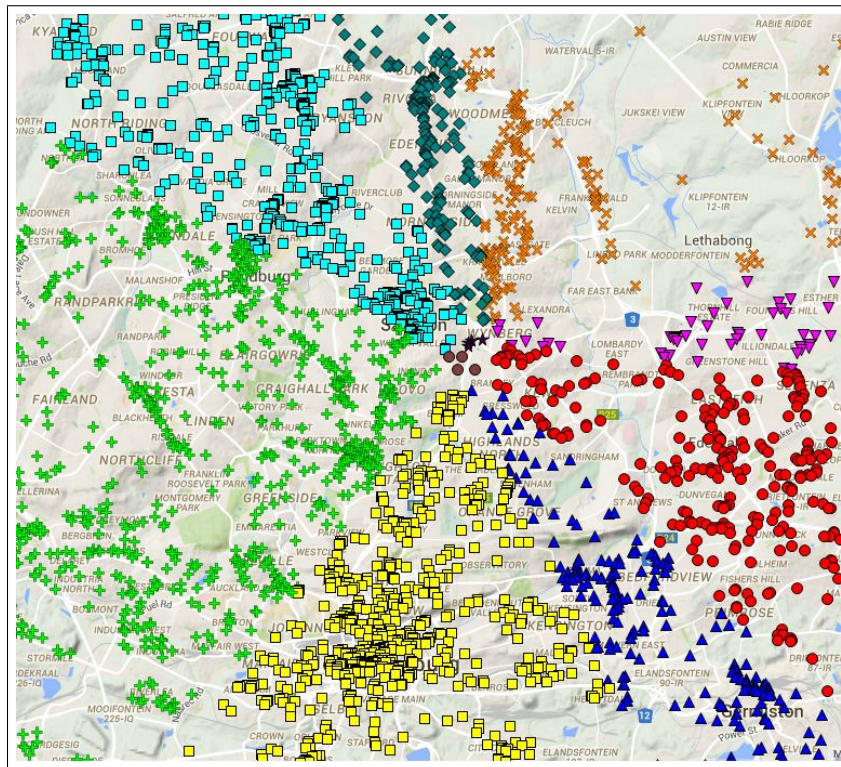


Figure 4.28: A zoomed in view of the solution created by the CLUSTER PAM method using DataS3.

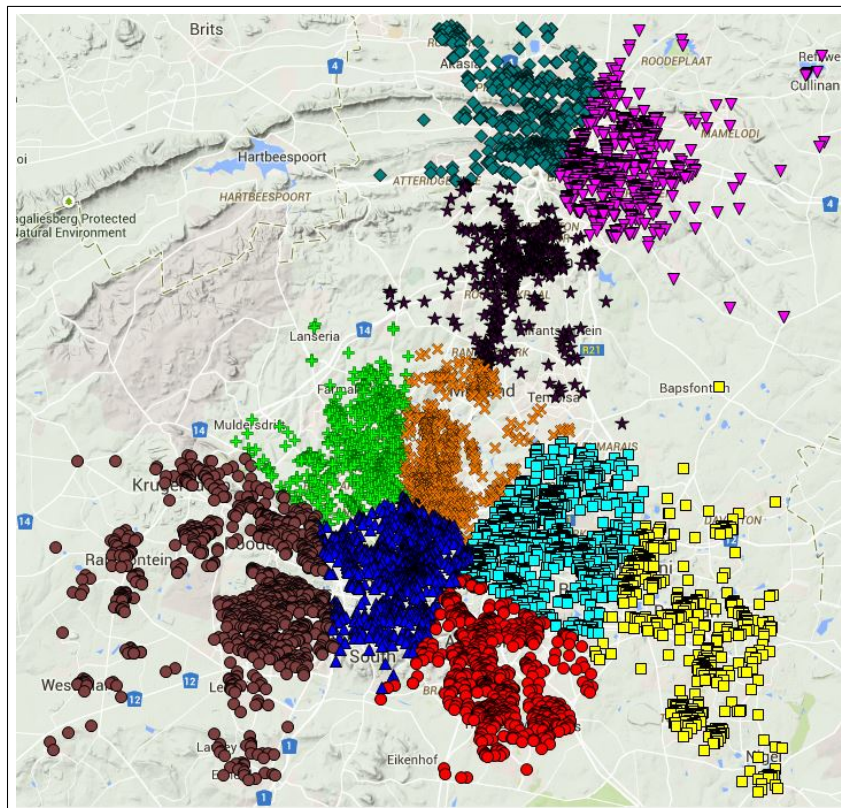


Figure 4.29: An illustration of the AS 136 algorithm using DataS3.

4.3.2.10 Algorithm AS 136 method

The AS 136 algorithm was described by Hartigan and Wong (1979) and has not received a lot of attention in recent years. The results of the AS 136 algorithm are illustrated in Figure 4.29. From the figure it is clear that the AS 136 like the h-means and h-median methods create well balanced clusters.

The CPU time and results of the AS 136 algorithm are given in Appendix A, Tables A.1 – A.39. In general the method's SSR, ADM and AWCD values outperforms Ward's method but are worse than the stepwise h-means and h-median methods. Even when using the AS 136 as starting solution for the h-median, the method could unfortunately not better the stepwise h-median.

The method showed potential to produce good results and the starting solutions were used as input with partitioning methods. Because these solutions were promising, it was decided to again use these solutions as starting solutions for the partitioning methods a second time. The results can be seen in Tables A.1 - A.36 under the AS 136 partitioning methods and AS 136 CLUSTER partitioning methods sections. Unfortunately, these solutions did not outperform the h-means and h-median methods.

Like the other partitioning methods, the method needs a starting solution to generate good solutions and the stepwise function was used to generate starting solutions. The increase in CPU times between the datasets are not as big as the h-median, showing that the method can be used for big datasets. The method has a CPU time of under 15 minutes for the USA13579 instance when combined with the stepwise function to generate starting solutions.

4.3.3 Graph-based clustering

Despite introducing a distance cut-off in Eq. (2.12) and Eq. (2.13), it was not possible to create a RNG graph-based solution within a reasonable amount of time for the DataS3. For DataS1 for example, the MST and GG graph-based methods took 3 and 15 seconds respectively, but because of the significant amount of edges included, it took over 2 hours and 46 minutes to generate a solution for the RNG graph-based method.

Upon further investigation it was determined that for DataS3, the Gabriel graph contained 34 480 edges and most of the CPU time was spent calculating the edges to include. The RNG graph on the other hand included 29 487 424 edges. When using a complete graph, calculation of the edges to eliminate in the RNG took a couple of seconds, but the program slowed down significantly when determining which points should be assigned to which clusters.

The results for the graph-based methods on DataS1 and DataS2, given in Tables

A.1 - A.36 are quite similar for all three methods. The results also did not fare well when compared to other clustering methods as can be seen in Figures 4.1 - 4.6. Because of this, getting results for the RNG graph-based method for DataS3 were not pursued any further.

The results of the MST and GG graph-based methods for the DataS3 are illustrated in Figures 4.30 and 4.31. Figure 4.32 illustrates the results of the RNG graph-based method for DataS1. From these figures it can be seen that the methods tend to create one mega cluster and groups of smaller outlier clusters, very similar to the single linkage method illustrated in Figure 4.7. The difference is that the outlying clusters are allocated at different locations. For the MST the clusters are bigger and overlap the mega cluster.

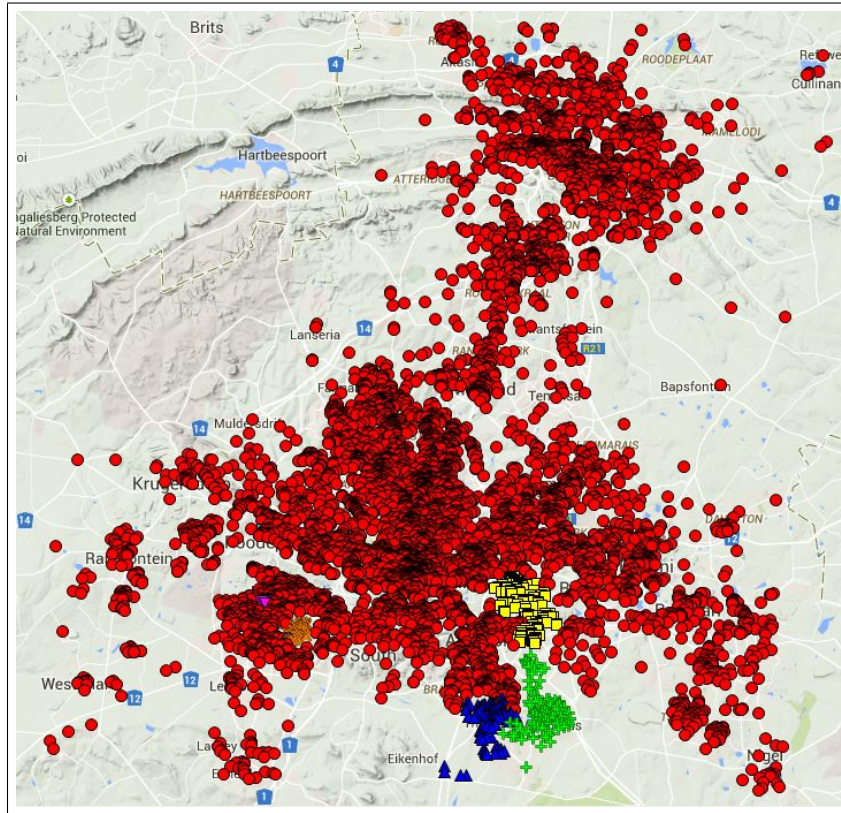


Figure 4.30: An illustration of the MST graph-based method using DataS3.

4.3.4 Nearest neighbour clustering

Section 4.1.3.4 describes how the nearest neighbour methods can be adapted to ensure the correct number of clusters are created.

4.3.4.1 The k-near neighbours method

Figure 4.33 illustrates the results of the adapted k-near neighbours method. Like the graph-based methods, this method also creates one mega cluster with the other points in small pockets of clusters, but interestingly enough, these small clusters are not all outliers on the outside of the mega cluster. A couple of the small clusters are

4.3 Discussion

embedded in the mega cluster. Because of the difference in size, the small clusters are difficult to see, for example the light blue circle cluster at the top of the map.

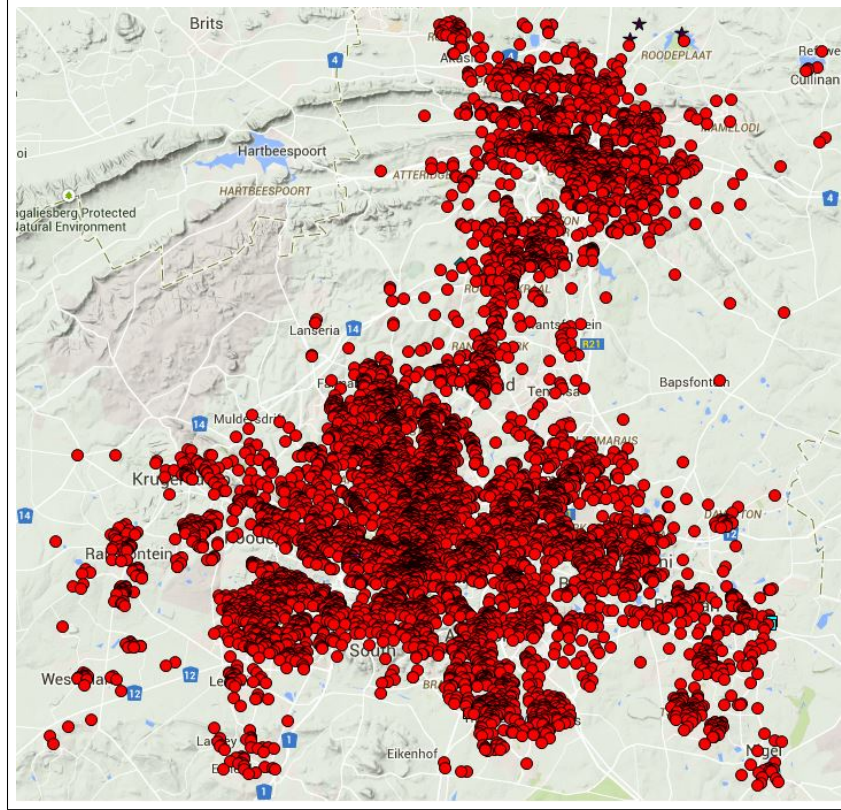


Figure 4.31: An illustration of the GG graph-based method using DataS3.

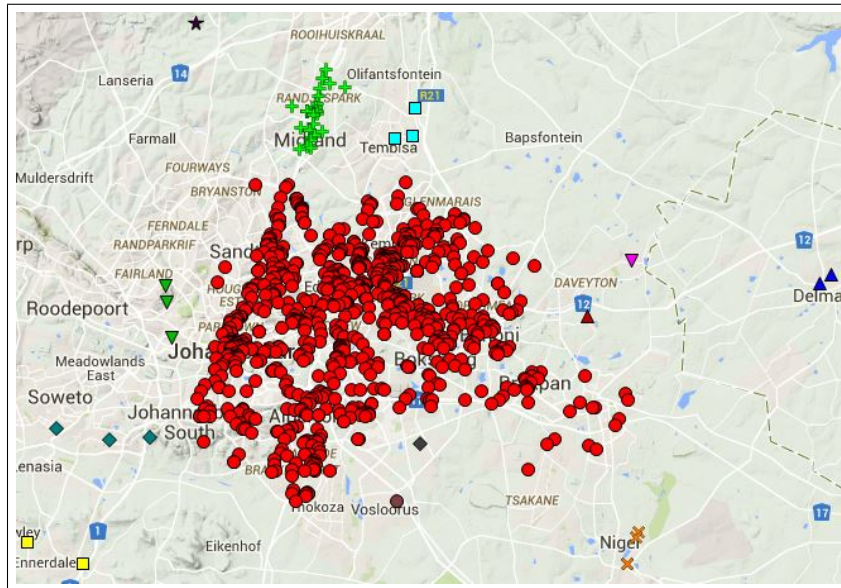


Figure 4.32: An illustration of the RNG graph-based method using DataS1.

The CPU time of the k-near neighbours method is quite fast and completed within 1 second for DataS1. The method however, does not have a tendency to create well spaced, equal-sized clusters and it is therefore not recommended for the LRP.

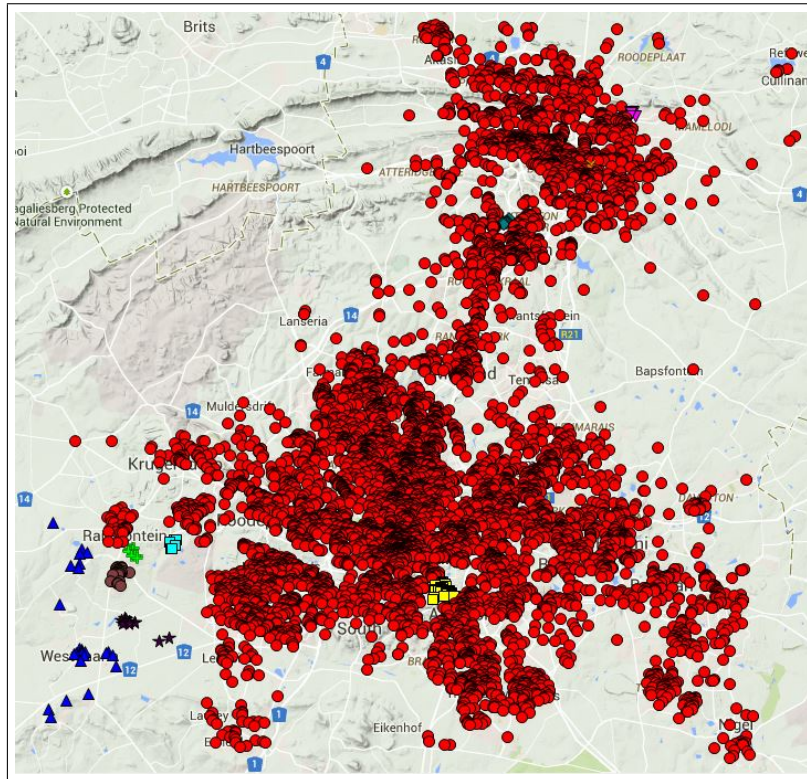


Figure 4.33: An illustration of the k-near neighbours clustering method using DataS3.

4.3.4.2 The MNV nearest neighbour method

As explained in Section 4.1.3.4, the MNV nearest neighbour method becomes quite complex when the algorithm is adapted to find exactly p clusters with an exponential time complexity. The results of the MNV nearest neighbour method are known to be the same as the graph-based RNG. The method was therefore not tested further in this thesis.

A word of caution regarding real case study data is needed here. It is easy to overlook the accurateness of geocoding points. This is especially the case in big datasets, where geocoding tools are used to locate the coordinates. Multiple points can sometimes be located at exactly the same coordinates. For example, when two points are located in the same shopping mall. Furthermore, if the address information is incomplete, the coordinates could be based on the coordinates of a suburb or if the street number is unknown, multiple addresses on the same street will all be allocated to a central point in the street.

The effect of this type of duplication on the calculation of the nearest neighbours, is that these points allocate each other as their RNN. In clustering methods using a certain number of nearest neighbours, these duplicates can create mini-clusters with all duplicates clustered together as one single point and distort the actual picture.

On one hand this can be argued as correct when one wants to create capacitated clusters, it makes perfect sense to cluster all points with the same geocodes to the same cluster. The creation of capacitated clusters is discussed further in Sections 5.2.3 – 5.2.4. On the other hand, it does not create visually attractive clusters[†]. This is especially the truth if the clustering method is dependent on distance and is not capacity constrained.

Two approaches are suggested to overcome this problem:

1. In the case of uncapacitated clustering, ignore any duplicate points when determining the nearest neighbours and allocate all these points to the same cluster
2. In the case of capacitated clustering, aggregate the points ensuring that the total demand per aggregated group stay below the capacity constraint. These aggregated customers can then be treated as single customer points.

4.3.5 Density-based clustering

Two density-based methods were tested, the results are as follows:

4.3.5.1 The k_{th} nearest neighbour clustering method

The results of the k_{th} nearest neighbour density-based method for DataS3 is illustrated in Figure 4.34. This figure clearly shows that the method returns worse results than the single linkage method. All the clusters except for the one mega cluster, contain only one or two points per cluster. The results for the five measures are given in Appendix A and in Figures 4.1 - 4.6 under the density-based heading. These results also show that the k_{th} nearest neighbour density-based method returned worse results for the SSR, ADM and AWCD values than the single linkage. The method is therefore not recommended for the LRP.

4.3.5.2 Hybrid clustering density-based method

The results of the different hybrid clustering density variations for DataS3 can be seen in Table A.40 in Appendix A. Both the results of the aggregation phase and the end results of the three variations are given. The first phase created $c = 70$ aggregated clusters. The second phase used the three variants of the hybrid clustering described in Section 4.1.3.5 to create ten clusters as usual.

From the results it is clear that variant 2, using the means of all points as the cluster centres, gave the worst solutions. No conclusive results could be made regarding variant 1 compared to variant 3, but using the CLUSTER h-median method to

[†]See Section 2 for a discussion on visually attractive clusters.

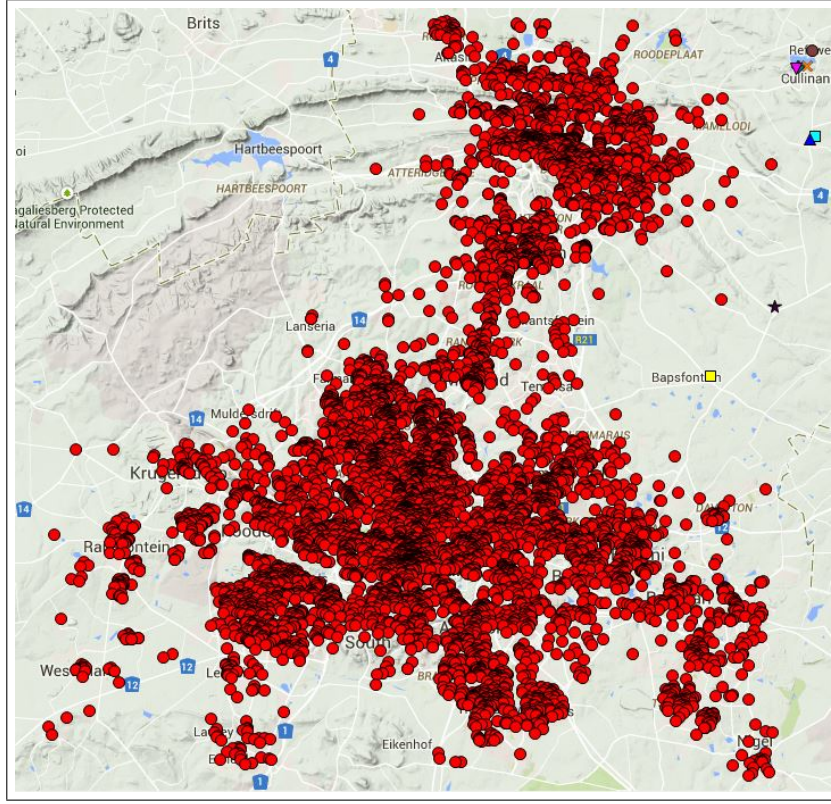


Figure 4.34: An illustration of the density-based k_{th} nearest neighbour clustering method using DataS3.

create aggregated clusters returned the best SSR values in both cases. This is particularly interesting because of the bad SSR values it created in phase 1. The results do not compare well to the other clustering methods.

The CPU times came from the aggregation of the solution (phase 1), the second phase took less than a second in all variants. For this reason only the CPU times of the aggregation phase were significant enough to report.

The solutions for variants 1 and 3 using CLUSTER h-median to aggregate are illustrated in Figures 4.35 and 4.36 respectively. In these figures it can be seen that clusters can merge even if they are not adjacent clusters with other cluster points between them. This is similar to the effect caused by SAHN methods when ignoring capacitated clusters during merging, discussed in Section 5.2.4.

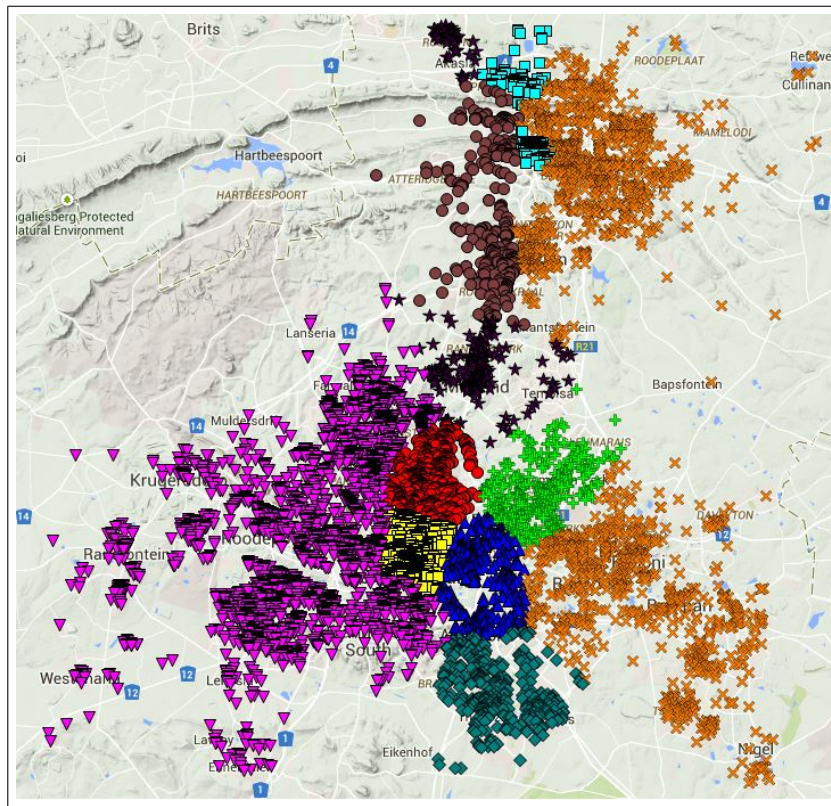


Figure 4.35: An illustration of the CLUSTER h-median density-based variant 1 method using DataS3.

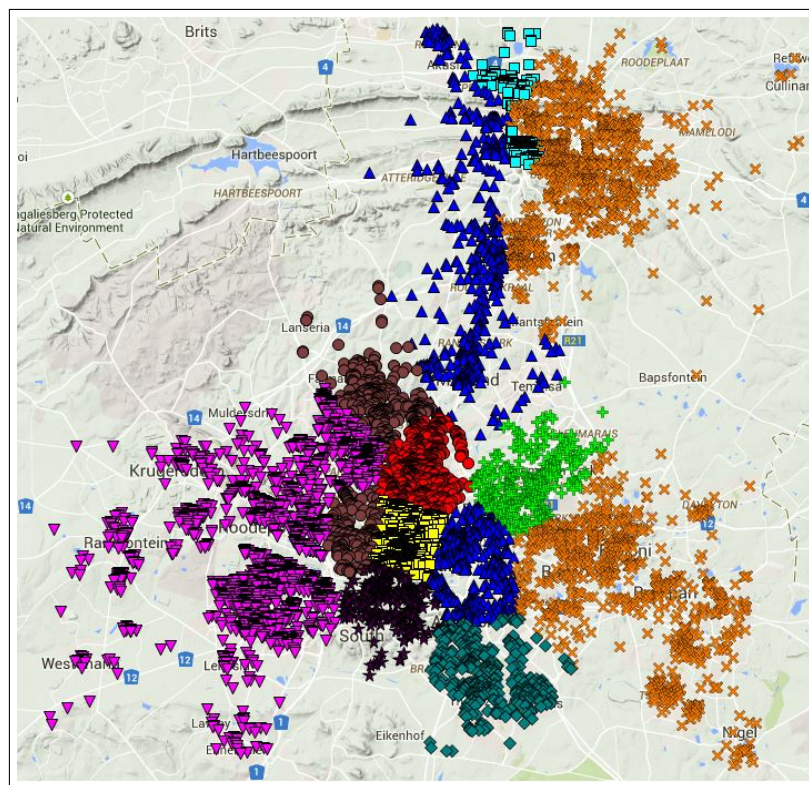


Figure 4.36: An illustration of the CLUSTER h-median density-based variant 3 method using DataS3.

4.3.6 Hybrid clustering methods

The hybrid clustering methods aggregate points into small clusters, which are then treated as single customer points and clustered further using the same or another method. These methods were suggested as a technique to cluster big datasets more easily. The methodology for the hybrid clustering methods were described in Section 4.1.3.6 and the results of these methods are discussed next.

4.3.6.1 Hybrid SAHN - SAHN clustering method

Both the single linkage and complete linkage were used to aggregate the points in the first phase before using Ward's method. Figure 4.37 display the result of the SSR measure for both the single linkage and complete linkage, using different number of clusters in phase 1. From the figure it is clear that the single linkage method has a bigger influence on the outcome and should not be used to aggregate to less than 2 000 clusters. Using the complete linkage method this impact was smaller. Both methods returned results between 46 000 and 49 000 from 2 000 aggregated clusters upward. The SSR results of the aggregated methods are both volatile and cannot be predetermined.

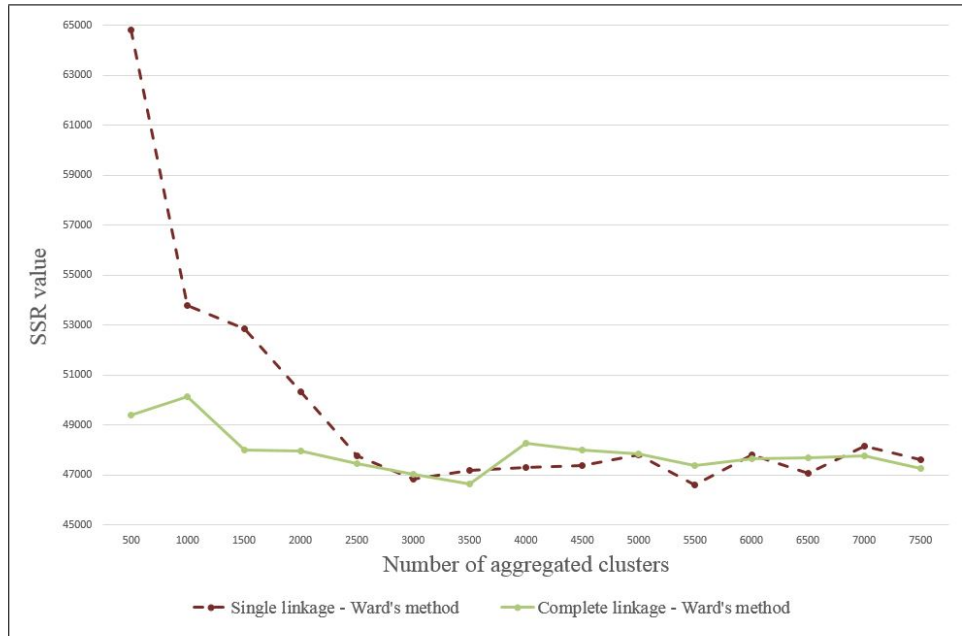


Figure 4.37: The SSR values for the hybrid single and complete linkage - Ward's method.

When comparing the SSR results with 3 000 and 4 000 aggregated clusters, illustrated in Figures 4.37 and 4.38, it is not possible to predict the peak in the SSR results at 3 600 aggregated clusters in Figure 4.37. It is therefore recommended to try different numbers of aggregated clusters in phase one before deciding on a final number of clusters to use.

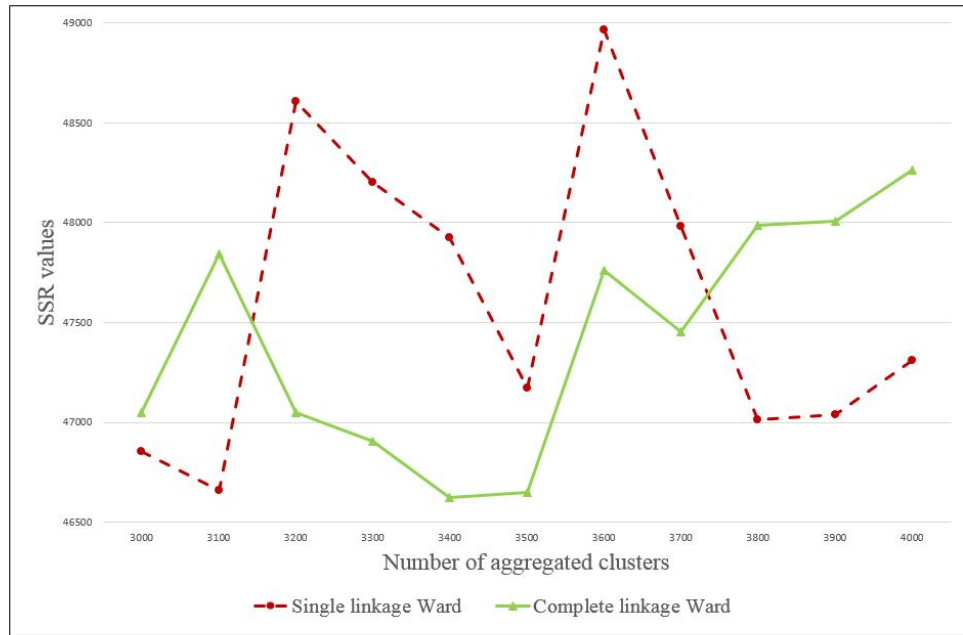


Figure 4.38: The SSR values for the hybrid single and complete linkage - Ward's method with 3 000 – 4 000 aggregated clusters.

The CPU times of the SAHN methods are not very high, making this a viable possibility. The CPU times of the two methods are illustrated in Figure 4.39. The times grow gradually between 39 and 54 seconds. This is similar to the 49 seconds it took to cluster the whole dataset using Ward's method.

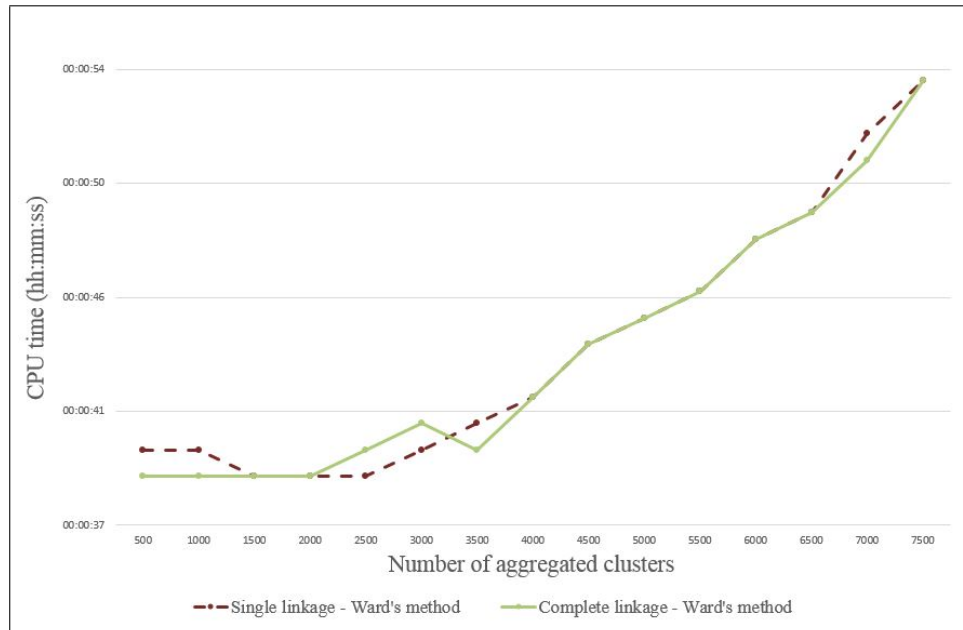


Figure 4.39: The CPU times for the hybrid single and complete linkage - Ward's method.

From Figure 4.38 it is seen that the best solution found was a SSR value of 46 625.44 for 3 400 aggregated clusters when using the complete linkage. This value was found

4.3 Discussion

in less than a minute and is better than the current SSR value of 47 485.90, found by only using Ward's method as discussed in Section 4.3.1.7.

Although it is possible to get good SSR results from the hybrid Ward's method, it is also clear that the results are volatile and finding a good solution cannot be guaranteed.

4.3.6.2 Hybrid SAHN - partitioning methods clustering

The SSR values of the partitioning methods used with the single linkage to aggregate are illustrated in Figure 4.40 and for complete linkage in Figure 4.41. In these figures some results for the PAM, j-means and k-means partitioning methods were left out for the bigger number of aggregated clusters. This is because the methods took too long.

It is also clear in the figures that the PAM method did not return good SSR results compared to the other partitioning methods. The Ward, j-means and k-median methods were also consistently outperformed in both figures by the k-means, h-means and h-median methods. The latter three methods also showed much less volatility and therefore more stable SSR results. The single linkage methods did

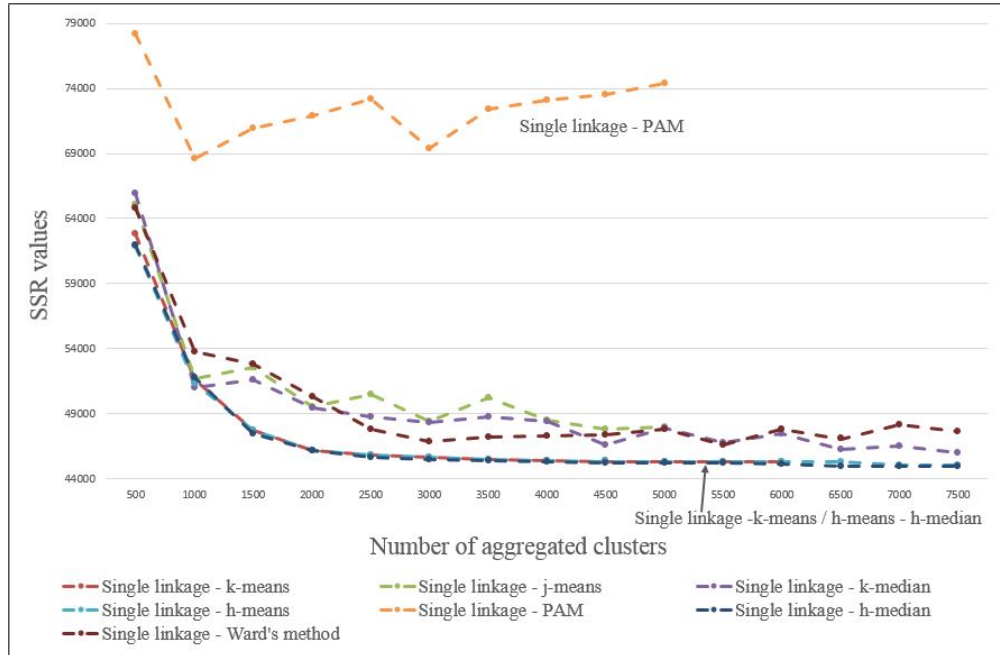


Figure 4.40: The SSR values for the hybrid single linkage - partitioning methods.

not perform well where the number of aggregated clusters were low, resulting in this method doing most of the clustering. The volatility of the SSR results were also less when using the complete linkage method to aggregate rather than the single linkage method. The detailed results of all the measures for 500 - 7 500 aggregated clusters

4.3 Discussion

are given in Tables A.42 - A.44 for the single linkage method and Tables A.45 - A.47 for the complete linkage method.

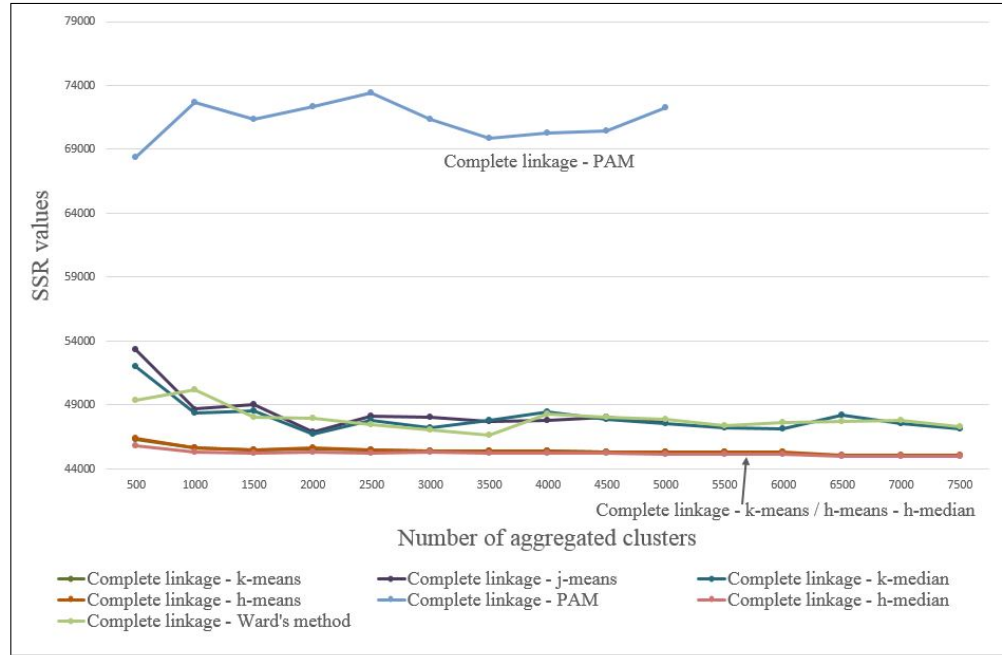


Figure 4.41: The SSR values for the hybrid complete linkage - partitioning methods.

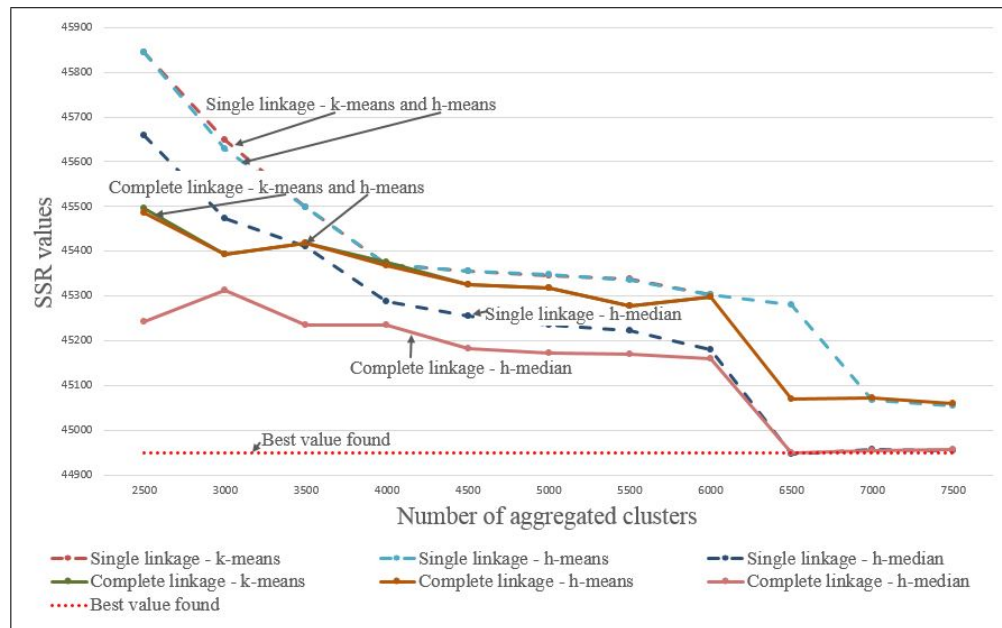


Figure 4.42: The SSR values for the best SAHN - partitioning methods with 2 500 – 7 500 aggregated clusters.

Figure 4.42 illustrates the best found SSR results for the hybrid k-means, h-means and h-median methods from 2 500 aggregated clusters upwards using both single and complete linkage to aggregate. The best found SSR value of 44 950.19 found by the h-median method in Section 4.3.2.6 is also shown. For 6 500 aggregated

4.3 Discussion

clusters, both the single and complete linkage methods with h-median were able to better this value with 44 945.96 and 44 948.28 respectively.

It can also be seen in Figure 4.42, that the hybrid h-means and k-means methods for both SAHN aggregation methods return the same results in most cases. This is particularly interesting because the CPU times differ significantly. A similar observation was found when comparing the h-means and k-means methods in Section 4.3.2.2. The CPU times of all the various hybrid SAHN - partitioning methods are illustrated in Figure 4.43.

When looking at Figure 4.43, one can also see that the CPU times of the particular partitioning methods were the same for both single and complete linkage aggregation. This is because both the single and complete linkage methods have polynomial time complexity and very similar CPU times.

Once the aggregated number of clusters has been reached the partitioning method took similar times to complete for the different starting solutions. The time complexity of the partitioning method does therefore not seem to be dependant on the starting solution and converges in relatively the same amount of time.

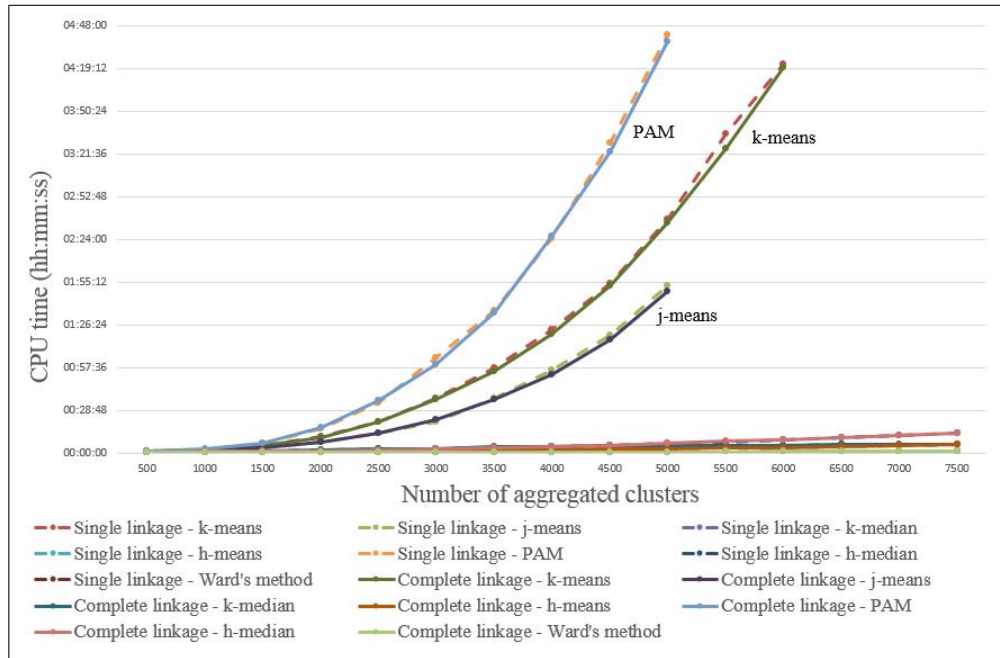


Figure 4.43: The CPU times of the hybrid SAHN - partitioning methods.

In the figure it is also clear that the PAM, j-means and k-means methods are not scalable to large datasets. These methods were excluded in Figure 4.44 to put focus on the quickest methods. The h-median method showed the most growth in CPU time as the aggregated number of clusters becomes more. Ward's method on the other hand has a stable CPU time with only a small amount of growth

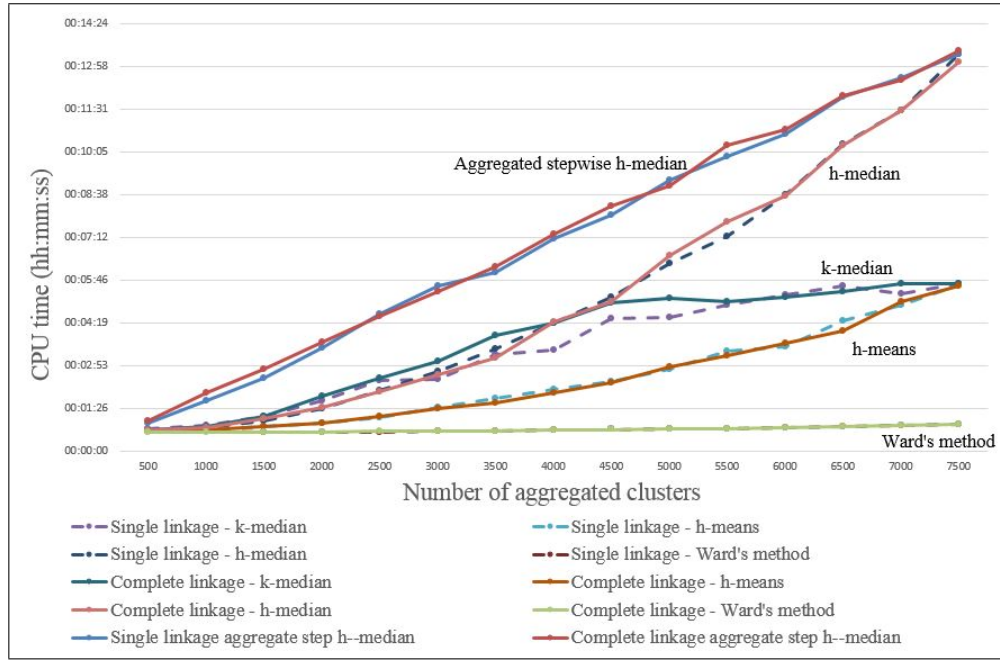


Figure 4.44: The CPU times of the hybrid SAHN - partitioning methods with best SSR values.

compared to the other methods. The k-median method is the only method that does not follow a conventional polynomial time regression line. It could be that this method is influenced more by the quality of the starting solutions than the other methods, making the CPU time more volatile. The method that returned the best SSR value of 44 945.96 is the hybrid single linkage h-median method using 6 500 aggregated clusters. This solution is illustrated in Figure 4.45. The method created well balance clusters that all seem to be of the same size. There are, however, a couple of borderline customers that could be assigned to the same cluster. See the blue triangle and brown circle clusters at the middle bottom of the figure, for example.

However, when comparing the single linkage h-median and complete linkage h-median methods in Figure 4.42, it is clear that the complete linkage h-median method returned better SSR results on average. It is therefore recommended to rather use the complete linkage h-median method where the number of aggregated clusters to use are unknown.

In an effort to better the results of this method even further and to see whether the volatility of the SSR results can be lessened, the results were used as starting solutions for the h-median. The results were between 44 953 and 45 140. Although the results fluctuated less, they were still volatile. When using Ward's method as the second method, it had more volatility than the other partitioning methods. Individual SSR values were better for some cases, but this technique was not able to beat the current best SSR result.

4.3 Discussion

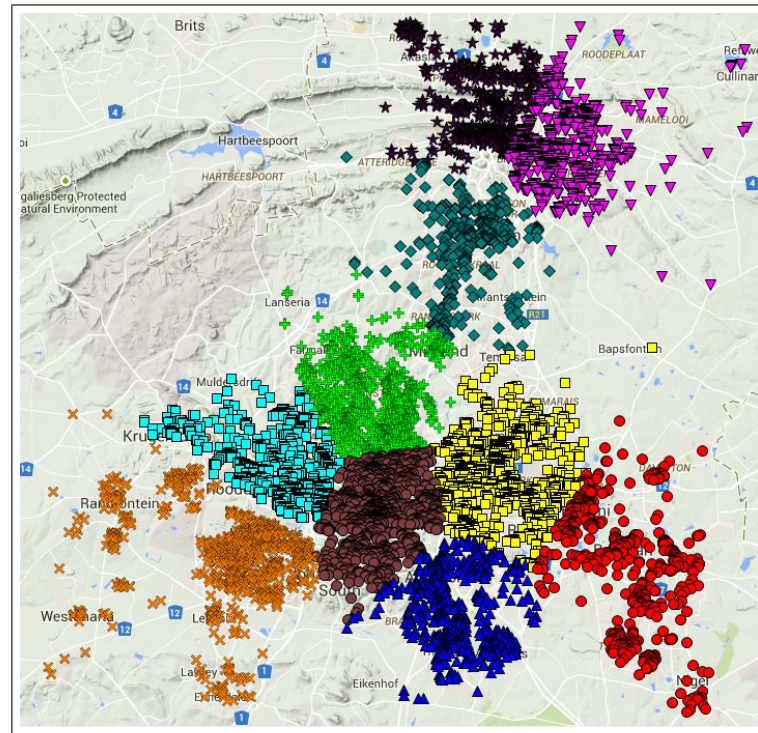


Figure 4.45: The hybrid single linkage - h-median method with 6 500 aggregated clusters.

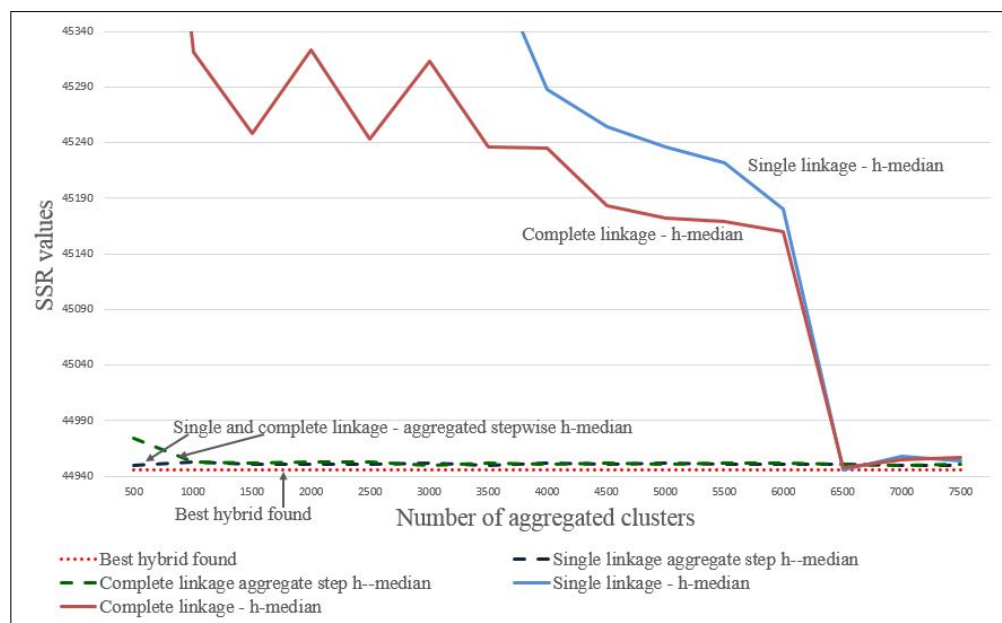


Figure 4.46: The SSR values for the aggregated stepwise- and hybrid SAHN - h-median methods.

4.3.6.2.1 Aggregated stepwise partitioning methods

Another variant of the hybrid SAHN-partitioning method was also tested. Here, instead of aggregating the data, the stepwise function was used to generate aggregated starting solutions. Only the h-median method was tested, because it is clear that this method returns the best SSR results and the goal is to lessen the CPU time needed to generate good starting solutions.

The results can be seen in Figure 4.46. In this figure, the SSR results of the single linkage h-median and complete linkage h-median were compared with the aggregated stepwise functions. Although the results were not able to better the best found SSR value, it was remarkably less volatile than the other hybrid methods. All SSR values were between 44 950 and 44 953 for all numbers of aggregated clusters, the only exception being the complete linkage with 500 clusters. The results were within 0.015% of the best found SSR value of the stepwise h-median, illustrated in Figure 4.45.

The CPU times of the aggregated stepwise h-median methods are shown in Figure 4.44. The CPU times correlate with the h-median methods, with additional fixed CPU time to create the aggregated stepwise clusters. For the complete linkage, the CPU times for the aggregated step was just below 2 minutes and for the single linkage, just below 1 minute. The CPU times of the best found SSR values for the single linkage h-median and complete linkage h-median were 10 minutes 19 seconds and 10 minutes 17 seconds respectively. This resulted in a time saving of 22.63% compared to the stepwise h-median method.

4.3.6.3 Hybrid partitioning clustering methods

A third hybrid method is tested using partitioning methods in both the first and second phases. Similar to the hybrid clustering density-based method, a predefined value of c clusters were created in the first phase. The value of c was calculated using Eq. (2.20). This solution is then clustered further using another partitioning clustering method in phase two. Variations of this hybrid method were tested using the partitioning methods with stepwise function as well as the CLUSTER variants of the partitioning methods, described before.

The results of the hybrid methods for the DataS3 can be seen in Tables A.49 - A.51, for $c = 70$. The best SSR and AWCD values were found using the h-means method in phase one, followed by the h-median method in phase two. This is interesting because these methods did not return the best SSR results in phase one.

The overall CPU times were at best 2 minutes, 20 seconds when using the k-median in the aggregation phase of the hybrid and at worst over 14 hours when using the CLUSTER k-means method. The best SSR value was 45 971.56 and did not compare

well against the overall best SSR value of 44 945.96, with a time of 3 minutes, 59 seconds. Even the partitioning hybrid methods that used the h–median method were not as effective as other types of hybrid methods that used the h–median method. The h–means h–median solution that resulted in the best SSR value for the partitioning hybrid methods, can be seen in Figure 4.47.

The partitioning hybrid methods were also tested for other c values, with $c = 1000$ and $c = 2500$. It became clear that the partitioning methods battled to create solutions for too many cluster centres. The CLUSTER partitioning methods especially, spent large amounts of time calculating cluster centres close to each other. The step-wise partitioning methods had problems converging on its own. Clusters can easily have no customers assigned to them, because the centres of the different clusters are so close to each other. This caused more time being spent on finding new depots using the FDR rather than completing the iterations.

It became clear that partitioning methods function better when the solution only needs to be divided into a small number of clusters compared to the number of customers. This defeats the purpose of using aggregation and it is therefore concluded that partitioning methods function better when not used in the first phase of a hybrid method.

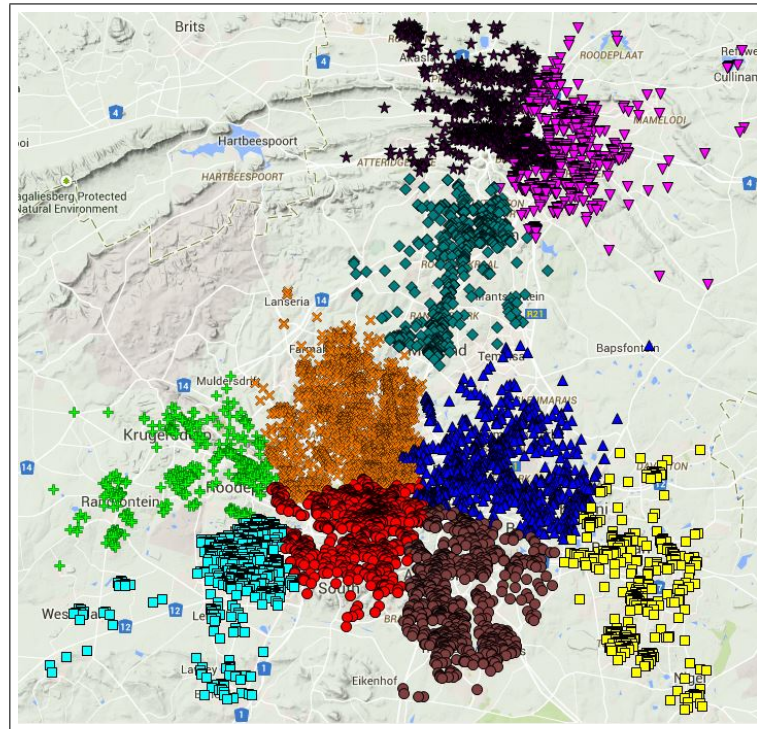


Figure 4.47: The hybrid partitioning method with 70 clusters created with the h–means method in phase one followed by the h–median method.

4.4 Conclusions

With the exception of Ward's method, the SAHN clustering methods did not return results as well as some of the iterative partitioning methods. When using the Lance-Williams parameters together with a distance matrix, all but the centroid method could be solved within polynomial time complexity.

The iterative partitioning methods were tested with a stepwise function to generate starting solutions. Of these methods, the h-median method returned the best SSR values and the j-means and PAM methods returned the worst. The h-means and k-means methods returned good results very similar to each other, but the time complexity of the k-means method makes it ineffective for big datasets.

The h-median method also proved to be less sensitive to starting solutions than the other partitioning methods. Unfortunately, the stepwise method is not an effective method to use in combination with the h-median for bigger datasets because the time complexity grows more than for other methods. Other methods can also be used as starting solutions for the partitioning method, but the stepwise function returned the best SSR results.

The CLUSTER partitioning methods were coded so that they can be given a starting solution or can generate their own starting solutions. The CLUSTER methods were coded to return the starting solution in the cases where it could better the SSR value of the given starting solution. The CLUSTER h-median method did not find results as good as the stepwise h-median. It only returned good starting solutions without being able to better it. A final partitioning method variant called the AS 136 method was also tested, and even though the SSR results were good compared to other methods, it was not as good as the h-median method either.

The graph-based, nearest neighbour and density-based methods did not perform well compared to the SAHN and iterative partitioning methods. In the hybrid methods, the latter two types were combined. The best SSR results were achieved using the hybrid single linkage – h-median method with 6 500 aggregated clusters, but this method is volatile and the number of clusters to use in the aggregated step is unknown beforehand. The complete linkage h-median method returned better SSR results on average and is therefore recommended whenever a hybrid SAHN - partitioning method is to be used.

The aggregated stepwise h-median method clusters the points before it is used in the stepwise function to generate starting solutions. This method provided a way to generate less starting solutions but still found SSR results within 0.015% of the best-known value. This clustering method is therefore recommended as the best method to use when solving the four variants of the LRP.

Chapter 5

Cluster-based approaches to solve variants of the CLRP

In this chapter, cluster-based methods are proposed to solve four variants of the CLRP:

- the Hamiltonian p -median problem (HpMP);
- the planar CLRP (plCLRP);
- the concentrator discrete CLRP (cdCLRP); and
- the standard discrete CLRP (sdCLRP).

For the HpMP, a new problem formulation is introduced and a modified Christofides algorithm (MCA) is proposed to create Hamiltonian cycles. A problem definition for the planar multi-depot single-source LRP with depot capacity constraints could not be found in the literature. The problem is introduced as the plCLRP and a problem formulation is proposed. The two-phased PROBUC method is also proposed. This is a regret order-based method that can be used to create capacitated clusters. Together with the MCA it can be used to solve the plCLRP.

New cluster-based approaches using Hamiltonian cycles are also proposed to solve the cdCLRP and sdCLRP. Both approaches decompose the problem into location and routing phases. In the cdCLRP, the solution approach makes use of lollipop routes to calculate approximate distribution costs. These are used to calculate the depot locations and determine if more depots are needed. Standard homogeneous depot costs and supply capacities are assumed, because of the large number of depots in the problem. In the case of the sdCLRP, the depot costs and supply capacities may vary per depot. An alternative cluster-based approach is proposed

that starts with the routing phase. Once routes have been created, the method uses a perturbation based heuristic to select depots in the location phase.

5.1 Solving the HpMP

In the HpMP, customers are clustered into p mutually exclusive clusters. Each group of customers is then structured into a Hamiltonian cycle with the goal to minimise the distribution costs. The problem was described in Section 3.3.1. A new problem formulation for the HpMP is now introduced. Afterwards, a new cluster-based solution approach is suggested.

5.1.1 Proposed new problem formulation

In the problem formulation for the HpMP, proposed by Glaab and Pott (2000), the subtour number constraints (SNC) count the number of direct cuts in the graph representing the problem. This can be seen in Eq. (3.71). The calculation of all cuts in a HpMP graph can, however, be quite tedious. In order to simplify the SNC we propose replacing this with an adapted variation of the subtour constraint from the VRP formulation.

The proposed problem formulation is as follows:

$$\text{Minimise} \quad \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk} \quad (5.1)$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \quad \forall i \in V, \quad (5.2)$$

$$\sum_{k \in K} \sum_{i \in V} x_{ijk} = 1 \quad \forall j \in V, \quad (5.3)$$

$$\sum_{i \in V} x_{ijk} - \sum_{l \in V} x_{jlk} = 0 \quad \forall j \in V, \forall k \in K, \quad (5.4)$$

$$\sum_{k \in K} \sum_{i \in V} \sum_{j \in V} x_{ijk} \leq n, \quad (5.5)$$

$$\sum_{i \in V} \sum_{j \in V} x_{ijk} \geq 3 \quad \forall k \in K, \quad (5.6)$$

$$x_{iik} = 0 \quad \forall i \in V, \forall k \in K, \quad (5.7)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K, \quad (5.8)$$

where c_{ij} is the distribution cost associated with travelling from customer i to j and the number of cycles is $|K| = p$.

Constraints (5.2) and (5.3) ensure that each node has only one arc leaving and one arc entering the node respectively. Constraints (5.4) ensure that each node is assigned to one and only one cycle and that the same number of incoming and exiting edges exist per customer point. Constraints (5.5) limit the total number of edges to ensure cycles are created.

The SNC in Eq. (3.71) is replaced with Eq. (5.5). This is based on the SNC in the problem formulation of the VRP, Eq. (3.52). Here the number of edges per route is restricted to $|S| - 1$. When the depot is included in this constraint set, a cycle with exactly $|S|$ edges is required to avoid subtours. The same can be said of any cycle in the HpMP consisting of $|S|$ points. No depots need to be identified and all customer points in a cycle are considered the same. Because the cycles are mutually exclusive, the total number of edges in all the cycles of the HpMP should add up to the total number of customer points, n .

As suggested by Gollowitzer et al. (2014), a constraint set is introduced to prevent cycles with only one or two customer points in Eq. (5.6). An extra constraint set is also added to prevent single customer point cycles from forming in Eq. (5.7). These two constraint sets can also be used to prevent less than p cycles from forming when used in conjunction with Eq. (5.5). This eliminates the possibility of subtours. Together with Eq. (5.4), these constraints ensure that exactly p cycles will exist, because each customer point must have both an entering and leaving arc. If p Hamiltonian cycles are constructed, these constraints will automatically be satisfied, because $|K| = p$. Finally, Eq. (5.8) restricts the decision variables to binary values.

5.1.2 A new cluster-based method to solve the HpMP

The proposed cluster-based method to solve the HpMP is illustrated in Figure 5.1. The solution approach can be divided into two phases, similar to the approach given by Branco and Coelho (1990) described in Section 3.3.2. The two phases are the clustering and routing phase.

In the clustering phase, any of the clustering methods investigated in Chapter 4 can be used to create route clusters. Here the method ensures that the number of cycles are exactly $|K| = p$. The clustering phase also checks that every cycle contains at least three customers to satisfy Eq. (5.6). If this is not the case, the two customers closest to this cluster are re-assigned to the cluster to ensure the minimum requirement is met. An exception is made in cases where the re-assignment of these customers will cause another cluster to also become too small. In this case, only customers from clusters with more than three points are considered for re-assignment.

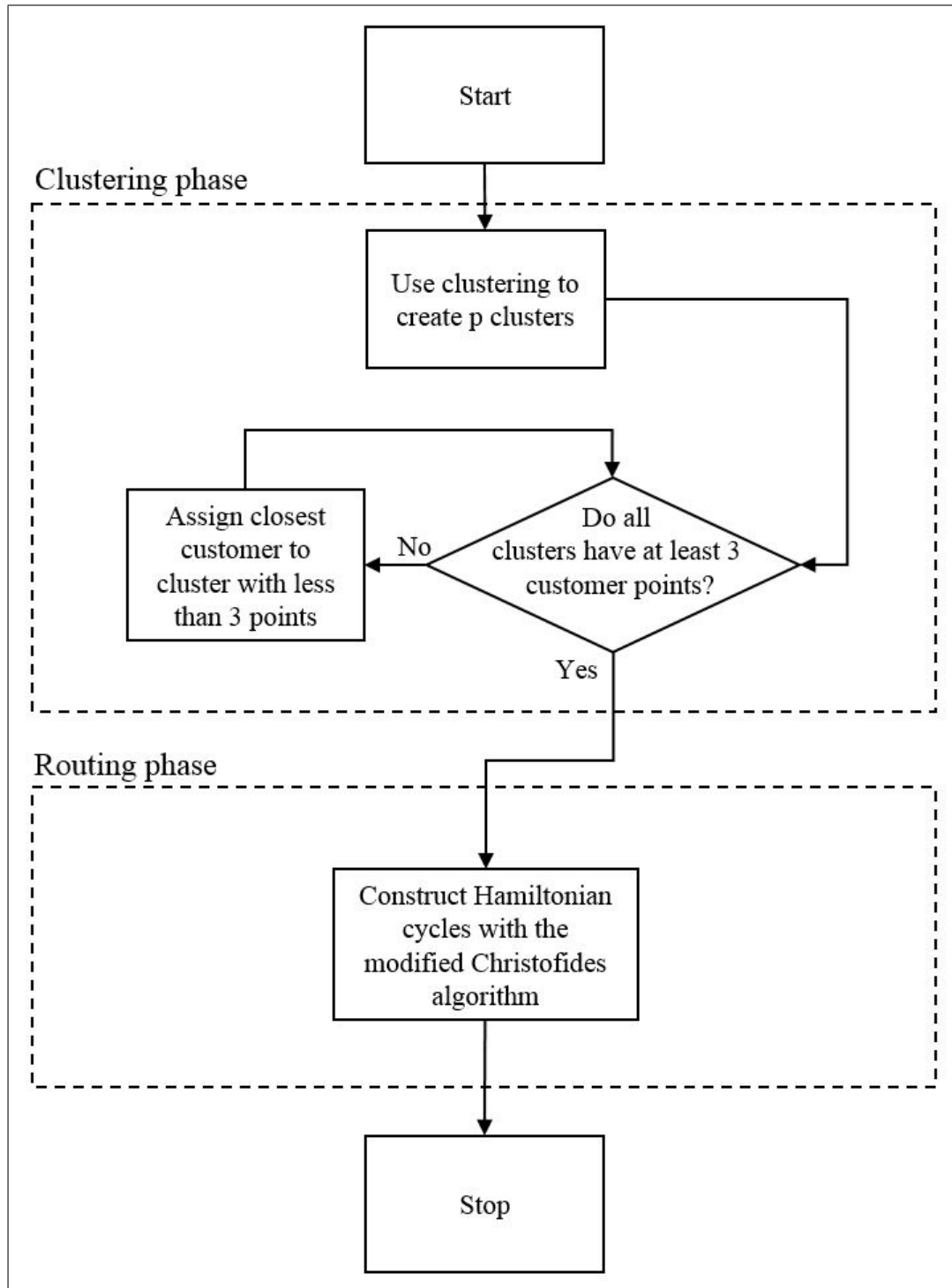


Figure 5.1: The proposed cluster-based method to solve the HpMP.

In the routing phase, Hamiltonian cycles are constructed from the route clusters, using the modified Christofides algorithm (MCA). The routing phase is used to satisfy Eq. (5.4) and Eq. (5.5). The MCA is now described in more detail.

5.1.2.1 Proposed method to create Hamiltonian cycles in the routing phase - the modified Christofides algorithm (MCA)

The MCA consist of three possibilities to construct a cycle from a route cluster in the routing phase of the cluster-based HpMP method. The three possibilities depend on the number of customers in the route cluster:

1. Where the number of customers in the cluster is below a predetermined threshold value, η , the branch-and-bound search method is used to find the shortest Hamiltonian cycles. The method uses an enumerated search to find the shortest cycle within a cluster, while eliminating the branch of a partially constructed cycle as soon as it becomes clear that it cannot produce a shorter cycle than the current best. This method is explained in Section 3.1.2.1. Here the Christofides algorithm (Papadimitriou and Steiglitz, 1998, p 416–418) as described on page 70, can be used to determine an upper bound and the initial best solution.

To construct a cycle the method incrementally adds one customer at a time from the specified route cluster. This create a partially constructed cycle. The remaining group of customer points and the two endpoints of the partially constructed cycle is referred to as the *remaining subset*. Every time a customer is added, the method calculates a lower bound for the *remaining subset*. The lower bound refers to the shortest distance of any route that could possibly come from the *remaining subset*. This lower bound is added to the distance of the already partially constructed cycle and is used to determine if this partial cycle could better the current best cycle. If the branch is not worth pursuing, the method removes the last added customer and add the next one to the partial cycle. The method continues in this way until all combinations of customers in a cycle have been tested or eliminated using the lower bound.

Two lower bound calculations were tested. In the first, the MST for the *remaining subset* is calculated to find the shortest possible route that could result from this subset. In the second lower bound calculation, the shortest edge for every point in the *remaining subset* to another point also in the subset, is identified. The distance of these edges are added together. Unlike the first, it is therefore possible in the second approach to create a disconnected graph from the *remaining subset* that will not result in a connected route and where Eq. (5.4) is not satisfied. Although the first approach therefore gives a tighter lower bound, the calculation time was longer in the numerical tests than the

time to enumerate all possible cycles that can result from this branch. It was therefore decided to use the second method to calculate lower bounds.

The value of η will obviously differ per dataset and programming environment. It will also differ depending on what is seen as an acceptable CPU time by the user. Barreto et al. (2007), for example, suggest solving a maximum of 40 customers exactly. In the tests conducted in this study a value of $\eta = 20$ was used.

2. In the case of more than η customers in a given route cycle, the Christofides algorithm is used to construct the route. In order to improve the solution, two different local searches are introduced. We refer to this as the enumerated Christofides algorithm. The first improvement is to construct all the different possible Eulerian walks from the eligible edges in the graph G , instead of only one Eulerian walk as described in step 3e on page 70. The different constructed Eulerian walks all start with the same starting point to prevent testing the same solution from different starting points. This enumeration gives various TSP answers and the shortest Hamiltonian cycle is picked.

The best solution is subjected to a second local search. It uses the same methodology as the branch-and-bound, but only a subset of the edges are used. The subset is similar to the neighbourhood of the Granular Tabu search method described by Toth and Vigo (2003) on page 74. It includes all edges from the MST (minimum spanning tree), as well as the edges in the best solution. A threshold length is calculated, see Eq. (3.55), and all edges shorter than this threshold length are also included. Finally, we also add the edges of each customer's closest three neighbours, should the neighbours be in the same route cluster. To ensure the method stays within a time limit, the number of tested complete routes are limited.

3. When the number of customers in a cluster becomes more than a predetermined number, ρ with $\rho > \eta$, the time to solve the Hamiltonian cycles using the enumerated Christofides algorithm with local search becomes unacceptable. In these cases the normal Christofides algorithm is used and only one Eulerian walk is constructed. We adapt the method by also testing the TSP route created when the walk is followed backwards in Step 3e of the Christofides algorithm on page 72. The second local search is also excluded because the construction of the neighbourhood becomes too time consuming. In this thesis a value of $\rho = 40$ was used.

5.1.3 Methodology and data preparation

The cluster-based HpMP method was tested the using various clustering methods investigated in Chapter 4, to determine which methods provide better routing cost

solutions. Since there are no capacity constraints, the clustering methods can be used as-is. The only adaption is that the method checks afterwards if all clusters have at least three customers per cycle.

Three different instances were tested. These are as follows:

5.1.3.1 DataS1

In order to compare the routing costs of the different clustering methods and to determine which measure could be used to indicate the lowest routing costs, DataS1 were clustered into 50 clusters. The different clustering methods from Chapter 4 were tested. The number of clusters were set to 50 to test the HpMP as part of a typical CLRP scenario with an estimated average of 5 routes per depot when using 10 depots.

Hybrid methods were excluded because the number of customers per route are normally quite small and do not justify the use of hybrid methods. The MCA discussed above was used to create the Hamiltonian cycles and to calculate the routing costs per clustering method.

5.1.3.2 Small instances

Three of the TSPLIB instances, listed by Reinelt (2015) and tested by Gollowitzer et al. (2014), were tested here. These are the dantzig42, gr96 and u159 instances. For every instance, two p values were used, as given by Gollowitzer et al. (2014). The calculation of the distances differ per instance and can be described as follows:

- dantzig42 – a distance matrix is given;
- gr96 – geo-coordinates are given, straight-line distances are calculated using the haversine formula (see below); and
- u159 – node coordinates are given, straight-line Euclidean distances are calculated using Eq. (2.32).

For the second instance, the distances are calculated in kilometres using the haversine formula. This formula is considered the most acceptable to calculate distances between longitude and latitude points and was recommended by Veness (2015). The author suggests using the spherical law of cosines simplification for the calculations, because it would only differ slightly in results where float variables with 15 significant figures of precision are used. Veness (2015) also provides a comprehensive explanation of the various distance calculation formulas.

The haversine formula with the spherical law of cosines simplification is given as follows (Veness, 2015):

$$d(x_i, x_j) = \arccos(\sin \varphi_i \cdot \sin \varphi_j + \cos \varphi_i \cdot \cos \varphi_j \cdot \cos \Delta\lambda) \times R, \quad (5.9)$$

where R is the earth's mean radius 6 371 km*, $(\varphi_i; \lambda_i)$ are respectively the radian latitude and longitude of point i and $\Delta\lambda = \lambda_j - \lambda_i$.

If the coordinates are given in decimal degrees, as is the case in instance gr96, the coordinates are converted to radians using the following calculation (Veness, 2015):

$$\varphi_i = \hat{x}_i \times \pi/180 \quad \text{and} \quad (5.10)$$

$$\lambda_i = \hat{y}_i \times \pi/180, \quad (5.11)$$

where $x_i = (\hat{x}_i, \hat{y}_i)$, $\hat{x}_i, \hat{y}_i$ are the latitude and longitude of point i in decimal degrees and $\pi = 3.141592653589793$.

5.1.3.3 Large TSPLIB instance

In order to test how well the solution approach works on big datasets, it was tested on the USA13509 TSPLIB instance (Reinelt, 2015) with 13 509 customer points, because there exists no big academic instances for the HpMP. Three instances, USA13509a – USA13509c, were tested with 500, 1 000 and 1 500 cycles respectively. Only the clustering methods that were quick enough and showed potential in Chapter 4, were used.

5.1.4 Computational results

The results for DataS1 using the various clustering methods together with the clustering measures from Chapter 4 are shown in Table 5.1. The route costs are the objective function values given in Eq. (5.1), assuming that $c_{ij} = d(x_i, x_j)$, the Euclidean distance between points i and j . These are also illustrated in Figure 5.2. This is followed by the results of the TSPLIB instances given by Gollowitzer et al. (2014) in Table 5.2 and the results for the TSPLIB instances, USA13509a – USA13509c, in Table 5.3. The minimum values in every column are shown in bold.

5.1.5 Discussion

In the results for DataS1 (Table 5.1 and Figure 5.2), there seems to be no correlation between the routing costs and the five measures established in Chapter 4. The Hamiltonian cycles were created using the proposed MCA method described above. The total routing cost, as calculated in Eq. (5.1), consist of adding up the total

* According to Veness (2015), different radii can be used for the earth's radius, for example 6 378 km (equatorial) and 6 357 km (polar), but the most generally accepted value is the mean radius at 6 371 km.

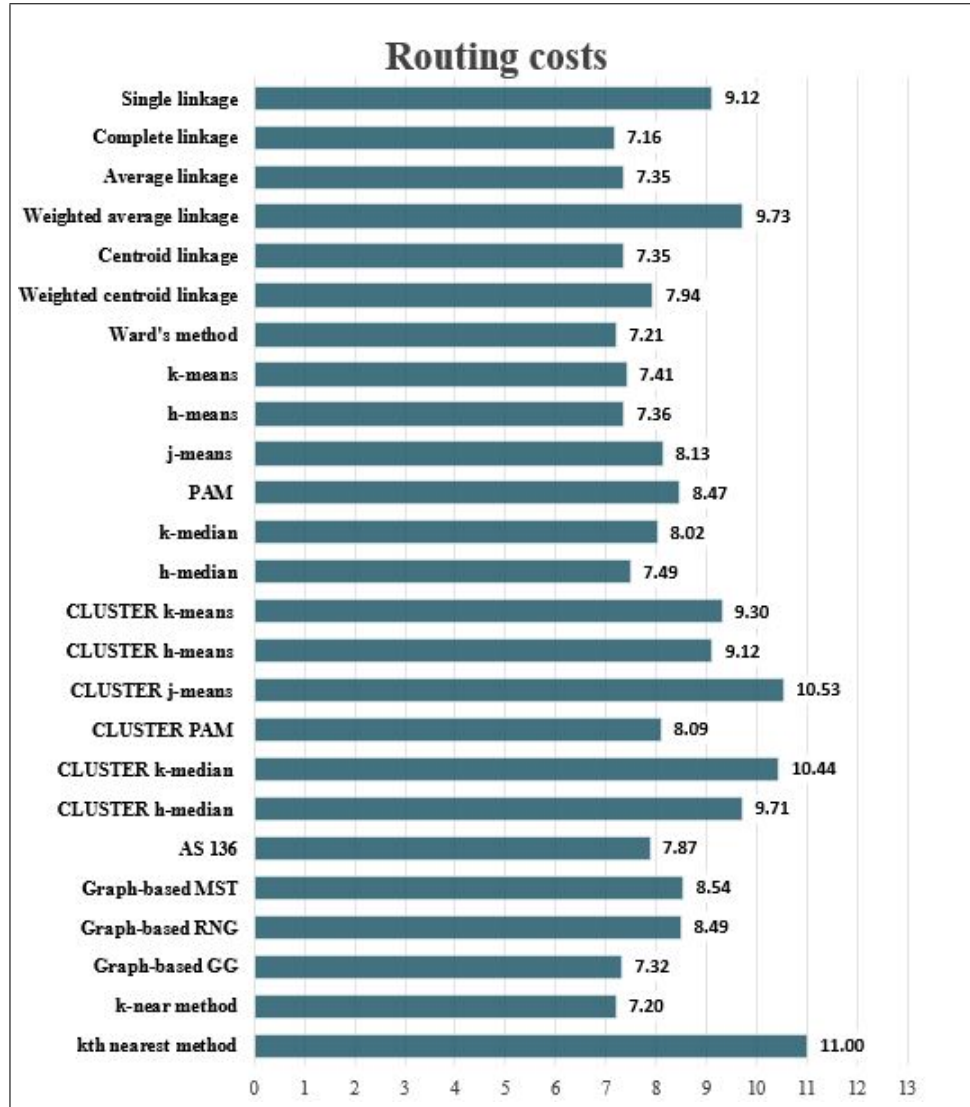


Figure 5.2: The routing costs for various clustering methods with 50 clusters using DataS1.

distances of the edges in the cycles. The complete linkage, k-near and Ward's methods returned the best routing costs results.

Different clustering methods were also tested to create the number of clusters specified by Gollowitz et al. (2014) in Table 5.2. Because the authors only listed the LP (linear programming) percentage gap for the various HpMP models and not the actual costs, it was not possible to compare the results.

From Table 5.2, it can be seen that the results vary and no clustering method consistently resulted in the best routing costs. The k-near and Ward's methods were both able to return the best results in two of the six instances compared to the other clustering methods. The methods also returned results relatively good compared to the best found for the other four cases. The percentage gap for the k-near method were at worst 12.40% and for Ward's methods at worst 13.62%,

5.1 Solving the HpMP

respectively, from the best found solution.

The weighted average linkage and graph-based RNG resulted in the best routing costs in one instance each, but their results did not compare well in the other test cases. Although the k-means, h-means, k-median and h-median methods did not result in the best routing costs, their percentage gap were also low, with 13.38% for the k-means and h-means, and 16.77% for the k-median and h-median methods.

Methods:	Routing costs	SSR	MSR	PD	ADM	AWCD	CPU times (mm:ss)
Hierarchical methods:							
Single linkage	9.12	7 462.70	105.92	34.89	3 697.10	5 003.94	00 : 02
Complete linkage	7.16	1 400.44	82.55	15.03	30.79	910.87	00 : 08
Average linkage	7.35	1 481.78	84.20	15.03	47.05	977.86	00 : 04
Weighted average linkage	9.73	5 377.50	118.18	31.14	1 099.77	3 637.01	00 : 03
Centroid linkage	7.35	1 481.78	84.20	15.03	47.05	977.86	00 : 05
Weighted centroid linkage	7.94	2 155.11	96.75	14.99	100.99	1 398.45	00 : 01
Ward's method	7.21	1 214.89	71.44	15.03	19.63	810.29	00 : 10
Partitioning methods:							
Stepwise k-means	7.41	1 283.54	71.49	17.51	22.93	848.96	00 : 16
Stepwise h-means	7.36	1 279.69	70.01	17.51	21.39	848.71	00 : 05
Stepwise j-means	8.13	1 560.18	76.03	17.86	30.13	1 034.16	00 : 16
Stepwise PAM	8.47	3 335.14	58.95	28.33	220.62	2 235.99	00 : 13
Stepwise k-median	8.02	1 484.54	72.34	20.16	28.91	988.54	00 : 05
Stepwise h-median	7.49	1 258.59	68.32	17.51	22.50	850.05	00 : 10
CLUSTER Partitioning methods:							
CLUSTER k-means	9.30	1 987.23	113.27	15.03	79.96	1 315.49	06 : 08
CLUSTER h-means	9.12	1 832.45	114.20	15.03	65.68	1 207.05	01 : 22
CLUSTER j-means	10.53	7 793.01	129.68	32.66	4 089.39	5 231.90	01 : 11
CLUSTER PAM	8.09	3 052.63	58.12	27.69	180.53	2 056.65	05 : 36
CLUSTER k-median	10.44	4 779.00	128.98	26.61	1 540.12	3 270.06	02 : 04
CLUSTER h-median	9.71	1 887.52	119.52	17.86	66.38	1 253.93	01 : 26
AS 136 Partitioning method:							
AS 136	7.87	1 371.72	73.30	17.51	25.83	910.12	00 : 09
Graph-based methods:							
Graph-based MST	8.54	7 276.04	56.53	51.82	2 943.40	4 983.91	00 : 04
Graph-based RNG	8.49	4 341.91	102.91	24.78	675.28	2 842.08	08 : 34
Graph-based GG	7.32	2 939.80	66.31	20.16	279.02	1 959.79	00 : 04
Nearest neighbours methods:							
k-near method	7.20	2 199.14	55.84	28.80	90.53	1 465.38	00 : 00
Density-based methods:							
k_{th} nearest method	11.00	7 921.71	141.63	33.43	4 067.73	5 293.93	00 : 03

Table 5.1: The routing costs for the HpMP and clustering measures when creating 50 clusters using DataS1.

5.1 Solving the HpMP

	dantzig42		gr96		u159	
Methods	3 clusters	10 clusters	5 clusters	20 clusters	5 clusters	30 clusters
Hierarchical methods:						
Single linkage	1 071.00	1 651.00	63 518.50	75 924.45	50 713.70	63 439.77
Complete linkage	759.00	968.00	66 248.64	65 143.06	52 728.79	48 993.55
Average linkage	809.00	1 047.00	65 466.46	69 388.84	48 088.28	50 839.10
Weighted average linkage	834.00	1 424.00	64 371.53	89 397.34	47 283.18	82 860.06
Centroid linkage	809.00	1 047.00	65 466.46	69 388.84	48 088.28	50 839.10
Weighted centroid linkage	756.00	1 446.00	64 885.29	75 033.66	52 147.73	53 315.49
Ward's method	785.00	901.00	64 199.14	58 070.25	49 877.34	46 309.25
Partitioning methods:						
Stepwise k-means	839.00	844.00	67 523.41	62 810.43	48 948.40	48 191.79
Stepwise h-means	839.00	844.00	67 523.41	59 436.30	48 237.24	48 191.79
Stepwise j-means	872.00	911.00	70 701.55	64 045.28	52 277.53	48 720.64
Stepwise PAM	848.00	979.00	73 466.13	71 325.22	56 052.31	58 392.11
Stepwise k-median	795.00	849.00	70 559.56	60 841.97	51 144.96	48 682.81
Stepwise h-median	820.00	849.00	70 559.56	61 487.30	49 667.31	47 827.28
CLUSTER Partitioning methods:						
CLUSTER k-means	840.00	1 125.00	65 018.33	75 117.47	48 237.24	62 402.71
CLUSTER h-means	840.00	1 011.00	65 798.91	68 772.95	48 990.09	57 836.51
CLUSTER j-means	833.00	1 054.00	64 147.73	68 879.23	57 024.92	73 362.77
CLUSTER PAM	848.00	903.00	64 214.85	67 957.60	58 854.73	62 551.89
CLUSTER k-median	779.00	1 125.00	66 855.21	67 258.17	54 972.83	76 609.51
CLUSTER h-median	795.00	1 034.00	71 179.69	71 240.21	49 667.31	61 487.16
AS 136 Partitioning method:						
AS 136	770.00	932.00	68 156.15	108 763.97	47 835.91	75 427.47
Graph-based methods:						
Graph-based MST	807.00	1 128.00	68 695.65	139 741.50	48 514.89	93 623.66
Graph-based RNG	894.00	1 179.00	60 428.45	75 857.85	49 917.41	55 668.22
Graph-based GG	816.00	943.00	65 199.19	75 413.33	51 271.68	62 991.31
Nearest neighbours methods:						
k-near method	740.00	793.00	64 908.85	62 287.55	53 146.83	51 987.49
Density-based methods:						
k_{th} nearest method	1 103.00	1 864.00	64 071.47	125 048.69	53 774.51	93 905.94

Table 5.2: The routing cost for the HpMP using three TSPLIB instances given by Gollowitzer et al. (2014).

5.1 Solving the HpMP

The CPU times for almost all the methods in Table 5.2, were less than one second and therefore not listed. The exceptions were that the k–near and k_{th} nearest methods took 37 and 34 seconds respectively in all instances. The AS 136 method had volatile CPU times with a maximum of 2 minutes 23 seconds for the u159 instance with 30 clusters, showing that the method do not always converge quickly.

The results for the USA13509a – USA13509c instances, given in Table 5.3, show that the complete linkage and Ward’s method returned the best routing costs. The results from Ward’s and the k–near methods were once again the methods with the lowest percentage gap to the best found solutions with 0.3% and 4.4% respectively. The results from the h–means and h–median were also good compared to other results. These methods had a maximum percentage gap of 5% and 7% respectively.

Instances	USA13509a		USA13509b		USA13509c	
	500 clusters		1 000 clusters		1 500 clusters	
	Cost	CPU times	Cost	CPU times	Cost	CPU times
Single linkage	3 422.26	11 : 32	6 706.06	09 : 49	9 527.23	06 : 16
Complete linkage	2 409.24	02 : 51	2 423.16	03 : 11	2 712.97	03 : 27
Average linkage	2 434.86	02 : 49	2 620.05	03 : 16	3 555.15	03 : 30
Weighted average linkage	3 393.73	20 : 18	6 413.32	20 : 28	10 392.56	20 : 36
Weighted centroid linkage	2 552.05	02 : 42	3 506.20	03 : 07	5 951.03	03 : 26
Ward’s method	2 416.05	03 : 37	2 348.75	03 : 36	2 299.20	03 : 36
Stepwise h–means	2 492.28	11 : 28	2 457.60	07 : 34	2 414.08	04 : 57
Stepwise k–median	2 630.30	06 : 28	2 648.02	08 : 20	2 679.34	05 : 10
Stepwise h–median	2 518.60	09 : 12	2 512.94	07 : 06	2 439.12	05 : 24
AS 136	2 573.14	09 : 10	2 569.33	07 : 33	2 556.60	06 : 20
Graph based MST	6 067.50	19 : 28	12 517.85	18 : 53	18 020.10	18 : 42
k–near method	2 411.00	03 : 25	2 391.35	03 : 01	2 399.25	02 : 54
k_{th} nearest method	4 904.63	10 : 44	11 188.69	10 : 18	17 306.89	08 : 50

Table 5.3: The results for the HpMP using the USA13509 TSPLIB instance.

5.1.6 Conclusions - HpMP

It is clear that the clustering methods used to find good distribution costs are not necessarily the same as the clustering methods recommended for the location phase. The clustering methods that resulted in the best routing costs were the Ward’s and k–near methods. The complete linkage method sometimes found the best routing costs within acceptable CPU times, but had varying results. Other methods that can also be recommended to return good results with reasonable CPU times are the h–means and h–median methods.

5.2 Solving the plCLRP

A new problem called the planar (or continuous) CLRP (plCLRP) is now introduced. Instead of opening p depots, as is the case in the plLRP, the objective function is changed to a fixed charge problem, where the goal is to minimise the total costs involved similar to the discrete CLRP. According to literature addressing the plLRP in survey papers by Nagy and Salhi (2007) and Drexl and Schneider (2015), the capacitated and the fixed charge objective function variations have not been introduced in literature before. Hence our new problem formulation is justified. A homogeneous fixed charge depot opening cost and a depot fleet size are also introduced.

In this section, a new problem formulation for the plCLRP is introduced. This is the first capacitated location–routing problem addressed without a predefined number of depots. Methods that can be used in general to estimate an initial number of depots/vehicles and create capacitated clusters are discussed. This is followed by a description of a cluster-based method introduced specifically to solve the plCLRP. This cluster-based method to solve the plCLRP makes use of the general methods to estimate the number of depots/vehicles and to create capacitated clusters.

5.2.1 Proposed new problem formulation

Based on the problem formulations of the plLRP (Salhi and Nagy, 2009) and the discrete CLRP (Prins et al., 2007), a new problem formulation for the plCLRP is now introduced. Given a weighted directional graph $G(V, A)$ with I the set of customers and J the index set of the depots. Let c_{ij} represent the cost associated with travelling from point i to point j . In the case where point j is a depot, the cost c_{ij} is equivalent to the distance between the depot j and the customer i , $d(\mathbf{X}_j, \mathbf{a}_i)$. Here $\mathbf{a}_i = (a_i^1, a_i^2)$ represents the coordinates of customer i with $\mathbf{a}_i \in \mathbb{R}^2$. The coordinates of depot j are represented by the additional decision variables $\mathbf{X}_j = (X_j^1, X_j^2)$ with $\mathbf{X}_j \in \mathbb{R}^2$.

As in the other continuous distribution network problems discussed in Chapter 3, the coordinates of the depots need to be determined in such a way that the total route costs are minimised. The distances can be calculated using any formula given in Section 2.8. In the case of Euclidean distances, Eq. (2.33) is used.

Let $V = I \cup J$ be the set containing all points. The vehicle and depot capacity constraints are Q and W respectively, and each customer has a demand of w_i . The fixed depot opening cost is f_j for depot j and the vehicle fixed cost is F_k for vehicle k . In this case homogeneous costs are assumed and we refer to the depot opening cost as f per depot and the vehicle fixed cost as F per vehicle.

Let K be the set of vehicles allowed to all depots, with $|K|$ the maximum number

of vehicles allowed. Only one route per vehicle is assumed. The chosen number of vehicles can be less than $|K|$ and is a function of the cost in the objective function.

In real-world case studies, it is common for depots to also be restricted to a fixed set of vehicles, referred to as the fleet of the depot, K_j for depot j , (Fu and Ishkhanov, 2004). The total number of vehicles is therefore at most $|K| = \sum_{j=1}^m |K_j|$ where $K = K_1 \cup K_2 \cup \dots \cup K_m$. We will refer to $|K_j|$ as the maximum number of vehicles per depot or fleet capacity and assume the same value applies for all depots.

The new problem formulation for the planar CLRP is proposed as follows:

$$\text{Minimise} \quad \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} z_{ijk} + \sum_{j \in J} f y_j + \sum_{j \in J} \sum_{i \in I} \sum_{k \in K} F z_{ijk} \quad (5.12)$$

$$\text{subject to} \quad \sum_{k \in K} \sum_{j \in V} z_{ijk} = 1 \quad \forall i \in I, \quad (5.13)$$

$$\sum_{i \in I} \sum_{j \in V} w_i z_{ijk} \leq Q \quad \forall k \in K_j, \forall K_j \subseteq K, \quad (5.14)$$

$$\sum_{j \in V} z_{ijk} - \sum_{j \in V} z_{jik} = 0 \quad \forall i \in V, \forall k \in K_j, \forall K_j \subseteq K, \quad (5.15)$$

$$\sum_{i \in I} \sum_{j \in J} z_{ijk} \leq 1 \quad \forall k \in K_j, \forall K_j \subseteq K, \quad (5.16)$$

$$\sum_{i \in S} \sum_{j \in S} z_{ijk} \leq |S| - 1 \quad \forall S \subseteq I, |S| \geq 2, \forall k \in K, \quad (5.17)$$

$$\sum_{i \in I} \sum_{k \in K_j} w_i z_{ijk} \leq W y_j \quad \forall j \in J, \forall K_j \subseteq K, \quad (5.18)$$

$$\sum_{i \in I} z_{ijk} \leq y_j \quad \forall j \in J, \forall k \in K_j, \forall K_j \subseteq K, \quad (5.19)$$

$$z_{ijk}, y_j \in \{0, 1\} \quad \forall i, j \in V, \forall k \in K_j, \forall K_j \subseteq K, \quad (5.20)$$

$$(X_j^1, X_j^2) \in \mathbb{R}^2 \quad \forall j \in J. \quad (5.21)$$

In the problem formulation the objective function, Eq. (5.12) consists of the total distribution costs of the routes, the total depot opening costs and the vehicle costs respectively. Constraints (5.13) – (5.16) are similar to the plLRP, Constraints (3.75) – (3.78). Constraints (5.13) ensure that each customer is only assigned to one route. Constraints (5.14) stipulate that the total demand assigned to a vehicle should not be more than the vehicle's capacity. Constraints (5.15) enforce continuity of the

routes by ensuring that there is an equal number of entry and exit edges per point on the same route. Constraints (5.16) denote that each route may leave from only one depot. If the vehicle is not used, the value will be zero.

Constraints (5.17) are similar to the subtour elimination constraints (3.52), for the CVRP. Here the subset S contains only customers from the set I , excluding depots from the set J and there should be at least two customers in S . Constraints (5.18) enforce the depot capacity constraints and Constraints (5.19) restrict the endpoint of a route to one open depot. Constraints (5.20) defines the binary decision variables x_{ijk} and y_j and Constraints (5.21) allow the coordinates of the depots to be anywhere in the Euclidean space. The maximum time limit constraints from Eq. (3.82), given by Salhi and Nagy (2009) for the pLRP, have been excluded, but can be added if route times need to be restricted.

There are only two sets of binary variables defined, z_{ijk} and y_j , where $z_{ijk} = 1$ if point i precedes point j on route k and $y_j = 1$ indicates that depot j is to be used. Unlike the discrete CLRP problem formulation given in Section 3.3.5 by Prins et al. (2007), the decision variable set x_{ij} have been excluded. This set was used to indicate that customer i is allocated to depot j on any route. It was left out because $\sum_{k \in K} z_{ijk} = x_{ij}$ and the set has no additional decision contribution to make.

5.2.2 Determining an initial number for depots and vehicles

When creating a solution approach for the fixed charge capacitated location–routing problems, two differences with the HpMP need to be addressed.

1. The number of depots and vehicles to use are unknown.
2. The problem needs to adhere to vehicle and depot capacity constraints.

The first step in the cluster-based method is to determine the preliminary initial number of depots and number of vehicles to use. At this point the distribution costs are unknown and an estimate is made based on the opening depot and vehicle costs. Two basic approaches can be followed.

1. Determine the minimum number of vehicles to use $k_{\min}$, using Eq. (3.101), given by Barreto et al. (2007). From here the number of depots to use, m , can be determined by grouping vehicles into depot clusters, adhering to the depot capacity constraint. Here a depot cluster contains all customers assigned to the same depot, no matter which vehicle they are assigned to.

In this approach the creation of clusters to construct vehicle routes precedes the creation of depot clusters. Depot clusters are created by merging routes

close to each other until the depot capacity is reached. Using this approach, the number of depots, m , to use can be calculated as

$$k_{\min} = \left\lceil \frac{\sum_{i \in I} w_i}{Q} \right\rceil \quad (5.22)$$

$$m = k_{\min} / \left\lfloor \frac{W}{Q} \right\rfloor \quad (5.23)$$

$$= \left\lceil \frac{\sum_{i \in I} w_i}{Q} \right\rceil \times \left\lceil \frac{Q}{W} \right\rceil, \quad (5.24)$$

where the notation $\lceil x \rceil$ represents the smallest integer larger than or equal to x and $\lfloor y \rfloor$ represents the largest integer smaller or equal to y .

2. The second approach is to first determine the minimum number of depots $m_{\min}$, by dividing the depot capacity constraint W by the total demand as given in Eq. (5.25). Once $m_{\min}$ has been determined, the number of vehicles k is calculated as the number of depots to use multiplied by the number of vehicles needed per depot. Here the clustering of customers to depots precedes the creation of vehicles clusters per depot. The number of vehicles is then calculated as

$$m_{\min} = \left\lceil \frac{\sum_{i \in I} w_i}{W} \right\rceil; \quad (5.25)$$

$$k = m_{\min} \times \left\lceil \frac{W}{Q} \right\rceil \quad (5.26)$$

$$= \left\lceil \frac{\sum_{i \in I} w_i}{W} \right\rceil \times \left\lceil \frac{W}{Q} \right\rceil. \quad (5.27)$$

The difference between the two approaches becomes apparent when the depot capacity is not a multiple of the vehicle capacity. For example, let the number of customers be $n = 20$, each with a unit demand of 1, the depot capacity $W = 4$, but the vehicle capacity $Q = 3$, which is not a divisor of the depot capacity. If the objective is to minimise the number of vehicles to use, Eq. (5.22) - (5.24) are used. In this case $k_{\min} = \left\lceil \frac{20}{3} \right\rceil = 7$ vehicles, the number of vehicles per depot is $\left\lfloor \frac{W}{Q} \right\rfloor = \left\lfloor \frac{4}{3} \right\rfloor = 1$. So the total number of depots $m = (7 \text{ vehicles needed} / 1 \text{ vehicle per depot}) = 7$ depots.

Whereas if the objective is to minimise the number of depots, Eq. (5.25) - (5.27) are used. Here $m_{\min} = \left\lceil \frac{20}{4} \right\rceil = 5$ depots, the number of vehicles per depot is a maximum of $k = \left\lceil \frac{W}{Q} \right\rceil = 2$ vehicles per depot and the total number of vehicles is $5 \times 2 = 10$ vehicles. Here it is clear that the vehicles cannot be filled up to capacity. This would be enforced by the depot capacity constraint where each depot is only allowed a maximum of four customers per depot, leaving a maximum of four customers to be distributed between at most two vehicles.

If the fixed depot opening costs are higher than the vehicle costs, as would be the case in most real world studies, the emphasis falls on minimising the number of depots to use. This example also highlights the importance of first determining the depot clusters before creating route clusters. It would prevent vehicles routes forming at full capacity that cannot be allocated to the same depot because of the depot capacity constraint. This is in contrast with clustering methods that create route clusters first as suggested by Barreto et al. (2007).

The solution using the minimum depots and vehicle costs is used to determine the actual distribution costs involved. However, if the distribution cost of a single route is more than the fixed cost to open a depot, other depot - vehicle number combinations should also be pursued.

Unlike the HpMP, which only has distribution costs, the cost function of the planar CLRP, Eq. (5.12), include the depots opening costs. The distribution costs are also dependent on the placement of the depots. For this reason the results from Chapter 4 becomes relevant when selecting a clustering method to create depot clusters. The aggregated stepwise h-median approach described on page 167, are recommended because the method provides good SSR, ADM and AWCD values with short time periods.

5.2.3 Creating capacitated depot and route clusters

Before a cluster-based method to solve the plCLRP can be introduced, we discuss how the capacitated depot and route clusters can be created. In both depot and vehicle allocation, clusters of customer points can be used to represent the different depot and/or vehicle assignments. In the case of the CLRP it is necessary to create capacitated clusters to ensure that the solution will meet the depot and/or vehicle capacity constraints. In order to measure the effectiveness of using the clustering methods discussed in Chapter 4, a technique is needed to create capacitated clusters from the unconstrained clustering solutions.

When using the SAHN methods, stopping rules must be used to ensure capacity constraints are met. If the stopping rule is the predefined number of clusters, one

needs to ensure that there are enough clusters to adhere to the capacity constraints by dividing the total demand by the capacity limit. This rule of thumb does not guarantee that the clustering methods would adhere to the capacity limit however and will only work for methods that create clusters with equal number of customers or demand. Different stopping rules to enforce the capacity constraints were investigated and are discussed next.

5.2.3.1 Stopping rules for SAHN clustering methods

The goal of using stopping rules in the SAHN methods is to find the best place to stop clustering. Here the focus is on stopping rules that can be used to create capacitated clusters. A number of different stopping rules, described in Section 2.1.10, were investigated:

1. Barreto et al. (2007) suggested the stopping rule to ignore the clusters that has reached capacity, while continuing merging the other clusters. This is effectively the same as ignoring full clusters and continuing the clustering with only the clusters that still has capacity. As suspected, this caused clusters to "jump" across from one outlying area over a dense area and merge with another cluster of outliers on the other side of the dense area. Although this created capacitated clusters, it defied the reasoning behind the clustering methods, namely to merge points in close proximity to each other.

An example of such a jump is shown in Figure 5.3, where DataS3 is clustered using the complete linkage method. Twenty clusters instead of the standard ten are used, because this illustrates the effect better. In the figure it can be seen that the light pink star cluster is split by the yellow square and orange cross clusters in the bottom left corner of the figure. To the right from this the black star cluster is split in two by the light purple cross cluster. At the top right corner, the green triangle cluster also overlaps the grey diamond cluster.

2. The second stopping rule considered, was suggested by Min (1987); to stop merging clusters the first time the capacity limit is encountered. This method returned more than double the number of clusters needed, some with only a few or single outlying points. If this rule was to be implemented, outlying clusters would need reclustering in order to merge more points together when forming capacitated clusters. This could lead to more capacitated clusters with outlying clusters and could cause multiple iterations of reclustering. It is also not straightforward to identify which outlying clusters should be reclustered together while minimising new outlying clusters from forming.
3. Everitt et al. (2011) suggest using a distance measure to determine where to

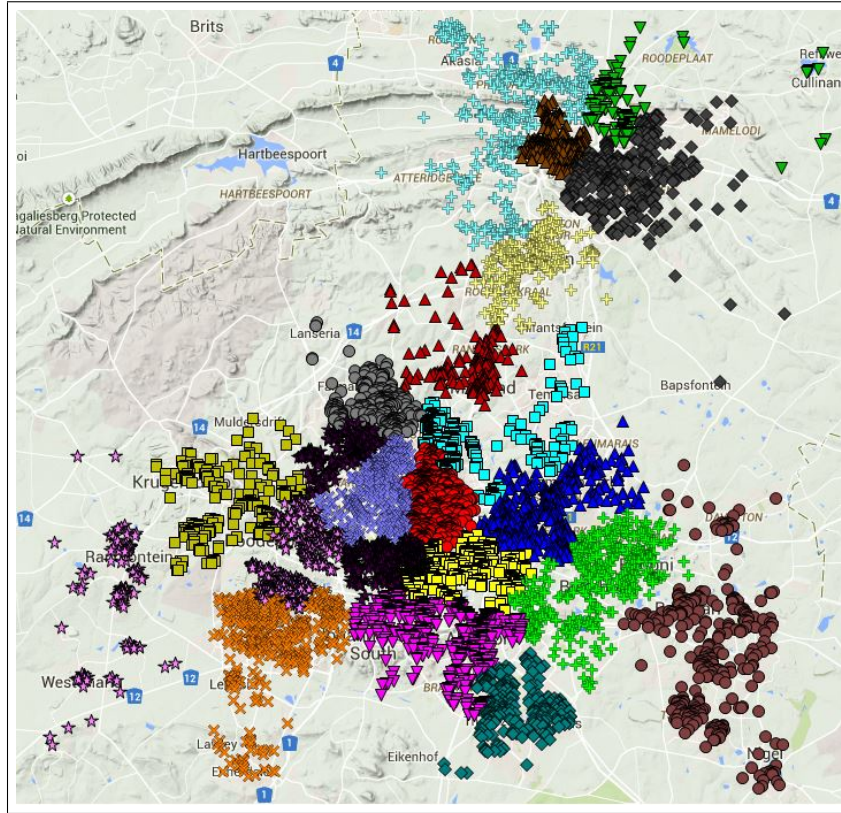


Figure 5.3: The results from the complete linkage clustering method where capacitated clusters are ignored using DataS3.

stop a cluster, but the correct distance can only be determined with trial and error. This means that the same clustering method have to be used repeatedly. If the clusters are not equally dense, as is often the case with geographical data, the best cut-off distance will differ per area, making a fixed distance not ideal either. This stopping rule also does not cater for capacity limitations directly and would therefore not work as-is for the CLRP.

4. Lam (2008) suggested using the pseudo F-statistic as a stopping rule. This is a better stopping rule because the relative difference in distance per area is taken into account instead of actual distances. The method does however become impractical for bigger instances like DataS3 and USA1350 as it results in memory overflow errors. The errors are caused by the multiplication of large numbers to calculate the pseudo F-statistic. The stopping rule also does not cater for capacity limits directly, it only looks at variances in distance.
5. Another possibility investigated is to use the standard deviation of the distances as a cut-off. This approach is similar to the functionality used in the density-based clustering discussed in Section 4.3.5, but uses the single linkage method to decide which clusters should be merged. When implemented using DataS3, the results still had mega clusters and single outlier points. The

stopping rule thus did not help to make the oversized clusters smaller.

6. Lastly, the possibility of modifying an existing solution in order to make the clusters meet the capacity constraints was investigated. This method is discussed in the next section.

5.2.4 The two-phase PROBUC method

The SAHN methods are the only clustering methods that make use of stopping rules. The other clustering methods create clusters based on the number of clusters specified beforehand or other input parameters. Another technique is therefore needed to create capacitated clusters. One such a method is to modify an unconstrained clustering solution to adhere to capacitated constraints after the clustering is completed. This method needs to be able to modify solutions created by any of the clustering methods. A method called the two-phase PROBUC (proportional regret order based unconstrained to constrained) method is now proposed for such purposes.

Figure 5.4 illustrates the work flow diagram of the two-phase PROBUC method. The method consists of the following two main phases:

1. **Re-assignment phase**

In the re-assignment phase, customers are re-assigned to other depots using a regret order based function until a feasible solution is found that satisfies the capacity constraints. This can be to either create depot capacitated clusters, Eq. (5.18), or vehicle capacitated clusters, Eq. (5.14), in the case of the pCLRP.

The method keeps on re-assigning customers while there are still overcapacitated clusters. If all clusters have been made capacitated or no more moves can be made in the re-assignment phase, the method moves to the improvement phase.

2. **Improvement phase**

In the improvement phase customer assignments are improved by allocating them to the same depot as used by most of their nearest neighbours. This is done in order to keep customers close to each other on the same routes. The improvement phase can cause clusters to become overcapacitated again and in this case the re-assignment phase is repeated a second time to ensure the solution is feasible again.

The two phases as well as the different proximity measures to calculate the regret values will now be discussed in more detail. Afterwards a new cluster-based method to solve the pCLRP is introduced. This method uses the PROBUC method to create capacitated clusters in both the location and routing phases.

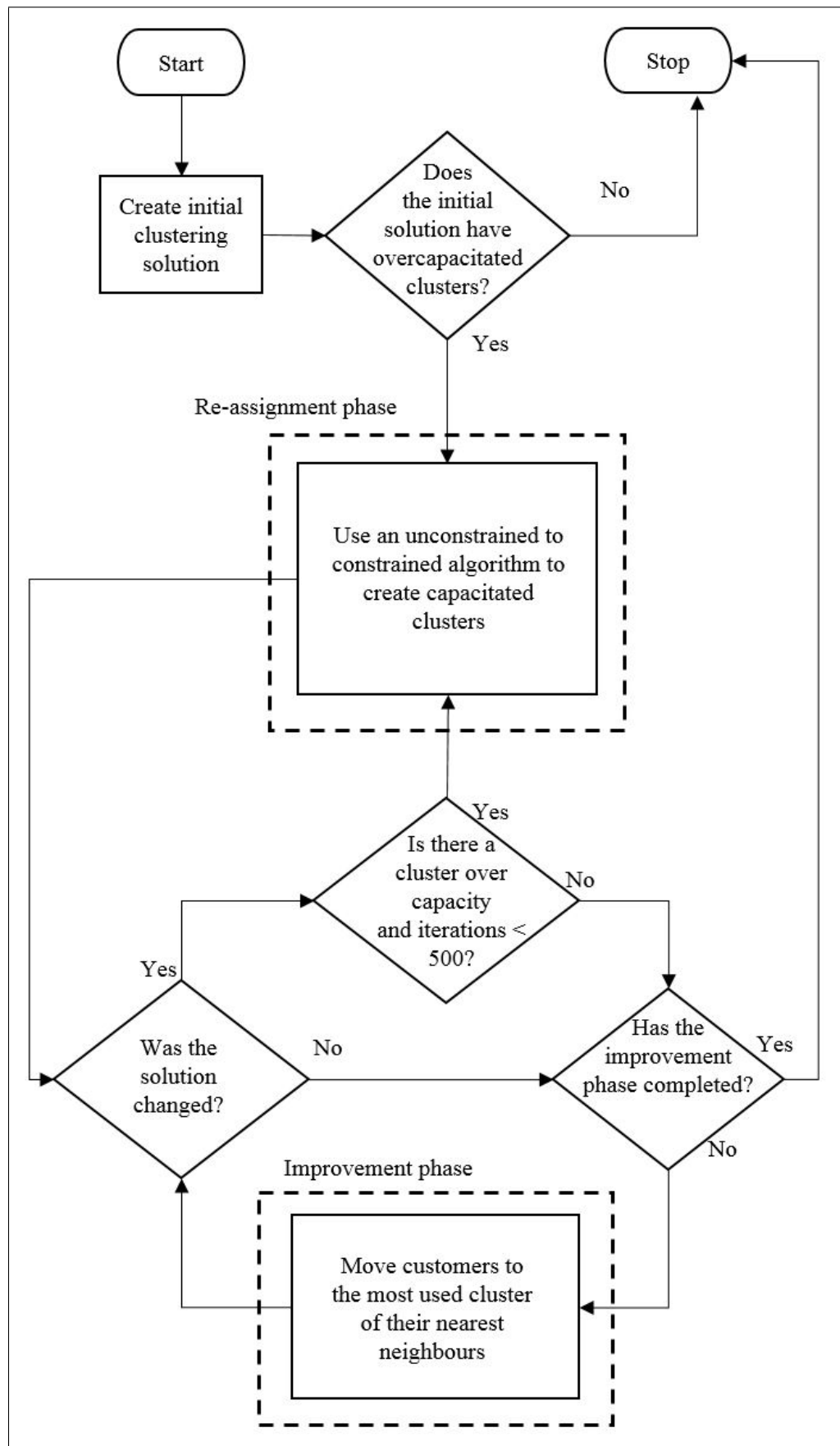


Figure 5.4: The work flow diagram of the suggested two-phase PROBUC method.

5.2.4.1 The re-assignment phase

The proposed unconstrained to constrained algorithm is based on a regret order based function. Here the *regret* is defined as the difference in the objective function values, using Eq.(5.12), between assigning the customer to a particular depot versus the original chosen depot from the clustering method. Customers are ordered in a list from smallest to largest regret values. The customers with the smallest regret values are re-assigned to the chosen depots until capacities are reached. The method focuses on forming mutually exclusive clusters that do not have overlapping boundaries.

Other regret order based functions have been proposed by various authors in the past, including Mulvey and Beck (1984), Negreiros and Palhano (2006) and Barreto et al. (2007). However, these authors do not make use of an iterative regret order based function. The iterative nature ensures that customers stay close to their assigned depots while preventing the creation of overlapping clusters.

When creating capacitated clusters two observations were made:

1. Overcapacitated clusters can easily be made capacitated by only assigning the customers closest to the centres of the clusters to them. The problem with this approach is that points can fall into pockets between neighbouring clusters. If these customers are assigned to a third cluster further away, the clusters will overlap, incurring extra travelling costs. It is therefore important when re-assigning points to ensure that customers stay as close to their current assigned cluster as possible and do not fall into these pockets.
2. Visually attractive clusters with non-overlapping boundaries tend to provide solutions with good SSR, ADM and AWCD values. Although not directly aligned, these tend to result in good objective function values for the CLRP. If the demand per customer varies, the best customer to re-assign might be too big because of capacity restrictions. The visual attractiveness of the solution will therefore decline with big variances in demand.

To keep the characteristic of the clustering solutions that were formed by the various clustering methods as much the same as possible, the re-assignment phase does not allow all moves in the regret ordering function. Here the focus falls only on re-assigning customers from overfull clusters where possible. A regret list containing possible moves from overcapacitated clusters to clusters that have capacity is composed. The regret values are then ordered from smallest to largest and moves are made in the order the customers appear in the list.

The algorithm is illustrated in Figure 5.5. It consists of multiple iterations. With each iteration new cluster centres are calculated and a new list of possible moves are calculated. Before a customer is re-assigned, the method first checks whether the depot has enough capacity to meet the customer's demand.

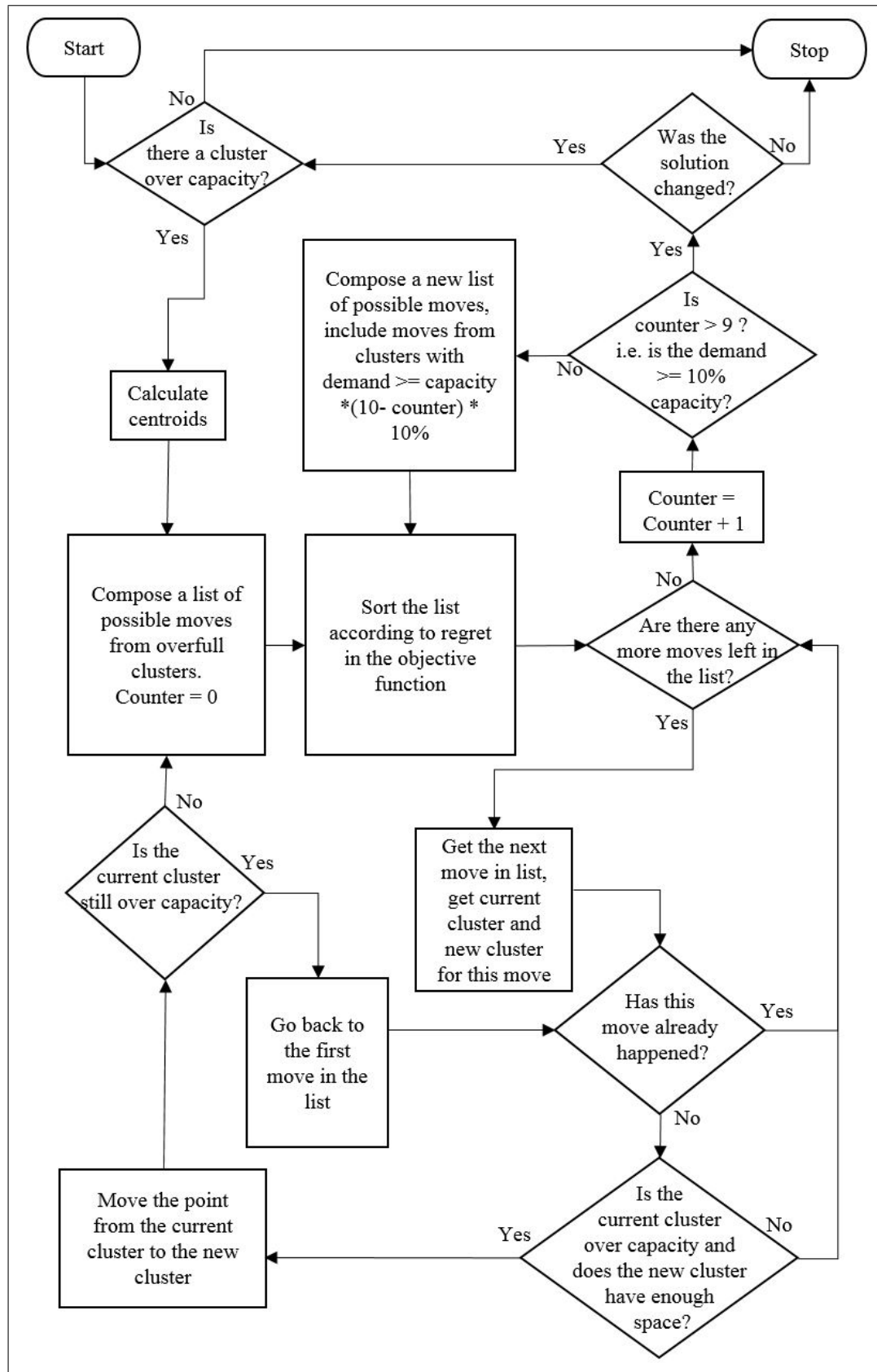


Figure 5.5: The re-assignment phase of the suggested two-phased PROBUC method.

There are however situations, like the single linkage method, where many points need to be re-assigned. In these situations only a limited number of customers can move from the overcapacitated clusters to their neighbours before the neighbouring clusters also reach capacity. In these cases it is normal in the literature to allow customers to be assigned to their third closest cluster and upward. However, this causes a "jumping" effect, similar to the stopping rule to ignore full capacities in the SAHN methods, illustrated in Figure 5.3.

In order to avoid this, customers in overfull clusters are not allowed to move to a third or further away clusters in the proposed algorithm. Here moves are only considered from the current to the next closest cluster. Instead of re-assigning customers to further away clusters, the algorithm starts re-assigning customers in clusters that are within 10% of the capacity limit to their next closest cluster. This is done to make more space for the overcapacitated clusters' customers.

The percentage from the capacity limit, needed before re-assignments from the cluster will be considered, drops in decrements of 10%. The decremental values were chosen to ensure that the focus falls firstly on moving customers from the fullest clusters to their neighbouring clusters. This will limit the number of unnecessary moves from underutilised clusters that would not resolve the overcapacitated issue.

There remains however, a restriction that customers are not allowed to move out of clusters with less than 90% of the capacity limit. This is used to avoid empty clusters from forming. After a move was made, the algorithm resets and starts with the next iteration, again only considering moves from overcapacitated to undercapacitated clusters (counter = 0).

In extreme cases where the problem becomes deadlocked even after moves from undercapacitated clusters with only 10% of the capacity limit have been considered, the algorithm will also consider re-assigning customers to overcapacitated clusters in order to change the cluster centres for the next loop. The number of moves to overcapacitated clusters are restricted to 10% of the capacity limit, after this the algorithm resets to the normal behaviour of only allowing the re-assigning of customers from overcapacitated clusters.

To prevent cycling, a list similar to the tabu list in the tabu search heuristic is introduced. Here customers are not allowed to move back to their previously assigned clusters. The last move for every customer is remembered for the next 10 iterations before the customer is allowed unrestricted movement again. The re-assignment phase stops after all clusters have reached capacity or no moves are possible for three consecutive iterations.

Although the method has the potential to take long to converge in cases where the cluster sizes are out of balance, this does not seem to be the case. In the case of

the single linkage method, for example, the method was able to converge within one minute for DataSet3. To stay within polynomial time, the method can also be restricted to perform a maximum number of iterations, e.g. 500.

Proximity Measures

It was observed that different proximity measures used in the regret function caused different types of customer groupings. Seven variations of the proximity measures for the regret function were tested in order to determine the best one to use in the re-assignment phase.

It is assumed that the demand of the customers are homogeneous so that the impact of the different measures under equivalent circumstances can be tested. The distribution costs, c_{ij} , are assumed to be proportionally equivalent to the travelling distances, $d(x_i, x_j)$. Straight-line Euclidean distances are used. Since only movements from the customer's current assignment and the next closest cluster are considered, the following definitions can be made:

- The *current proximity*, d_{cur} , refers to the distance between the customer and the means of the currently allocated cluster centre.
- The *second best proximity*, d_{2nd} , is the distance between the customer and the means of the closest cluster centre, excluding the current assignment.
- The *total proximity*, d_{tot} , is the sum of the distances from the customer to all cluster centres, including the current assignment.

These proximity variables differ per customer and can be used to calculate a regret value $r(i)$ associated with moving customer i to the second closest depot. It should also be noted that the current assignment of a customer is not necessarily the closest depot centre. This is because in the different clustering methods, customers can be assigned to depots other than the closest depending on the clustering algorithm used. As such, it also means that the regret values can be negative.

Different variants of the proximity measures were tested. We test the effectiveness of all variants using DataS3 before justifying our choice for regret function. Figures 5.6 – 5.12 illustrate the use of the different proximity measures in the two-phase PROBUC method using both the single linkage and complete linkage methods to generate starting solutions. The same dataset used in Chapter 4, DataS3 described in Section 4.1.1, was used and clusters were given a homogeneous capacity constraint of 1 000 customers with unit demand. Circles were drawn to highlight obvious problems with regards to the clusters that were visually detected. The results and CPU times are given in Table 5.4.

The formulations of the different proximity measures are given in Eq. (5.28) – (5.34).

$$r_1(i) = d_{2_{nd}}, \quad (5.28)$$

$$r_2(i) = d_{2_{nd}} - d_{\text{cur}}, \quad (5.29)$$

$$r_3(i) = d_{\text{tot}} - d_{2_{nd}}, \quad (5.30)$$

$$r_4(i) = \frac{d_{2_{nd}}}{d_{\text{cur}}}, \quad (5.31)$$

$$r_5(i) = \frac{d_{2_{nd}}}{d_{\text{tot}}}, \quad (5.32)$$

$$r_6(i) = \frac{d_{2_{nd}} - d_{\text{cur}}}{d_{\text{cur}}}, \quad (5.33)$$

$$r_7(i) = \frac{d_{2_{nd}} - d_{\text{cur}}}{d_{\text{tot}}}. \quad (5.34)$$

The unconstrained solutions for the single linkage and complete linkages are shown in Figures 5.6(a) and 5.10(a) respectively. The distance to the second best cluster being considered for assignment, $d_{2_{nd}}$ is used quite frequently in the literature. This method of re-assignment tends to create concentric circular clusters around the centres and allows points to easily fall between the circular clusters when capacity is reached. These points will then remain assigned to the original unconstrained clusters, because there was not enough capacity to re-assign them as well. This measure does not tend to create tightly clustered assignments, as illustrated by the red circles in Figures 5.6(b) and 5.10(b).

Mulvey and Beck (1984) suggested using the difference between the *current proximity* and the *second best proximity*. In this case, the regret values will be negative where customers were not assigned to the closest cluster centroids. Examples are the solutions from the single linkage and complete linkage SAHN methods for the DataS3, illustrated in Figures 5.7(a) and 5.11(a). Although this method gives tighter clusters, there are still customers that can fall between cluster assignments and are assigned to a third cluster. One can also use the difference between the *total proximity* and the *second proximity*, as illustrated in Figures 5.7(b) and 5.11(b). This method also left points furthest from the cluster centres to remain assigned to the original clusters. The values in Table 5.4 did not compare well against the other proximity measures either.

Instead of using the difference in distance, another type of calculation is to use proportional regret. An example is the ratio between the *second best proximity* and the *current proximity*, as suggested by Barreto et al. (2007). The solutions of using

a proportional ratio as regret calculation are shown in Figures 5.8(a) - 5.8(b) for the single linkage and Figures 5.12(a) - 5.12(b) for the complete linkage methods. Figures 5.9(a) - 5.9(b) and Figures 5.13(a) - 5.13(b) illustrate combining both the difference and proportional ratios in the regret calculations.

Different proximity measures were effective for the single and complete linkage starting methods and the results were inconclusive. It was however clear that the SSR, ADM and AWCD results for $r_1(i)$ and $r_3(i)$ returned on average worse results than the other regret functions in both test cases.

When comparing the visual attractiveness of the single linkage solutions, measures $r_6(i)$ and $r_7(i)$ (Figures 5.9(a) and 5.9(b)) returned solutions with the least cluster overlap that could easily be solved in the improvement phase. For the complete linkage no obvious problems could be found when using measure $r_7(i)$ (Figure 5.13(b)). It was therefore decided to use the regret function Eq. (5.34) associated with the proximity measure $r_7(i)$. Because the regret function is based on a proportional calculation, the method is named a proportional regret order based function. This measure also returned the best SSR, ADM and AWCD values for the complete linkage in Table 5.4.

The PROBUC method can be used for customers with or without varying demand. As recommended by Geetha et al. (2009), there are two other ordering methods that can be used to deal with varying demand:

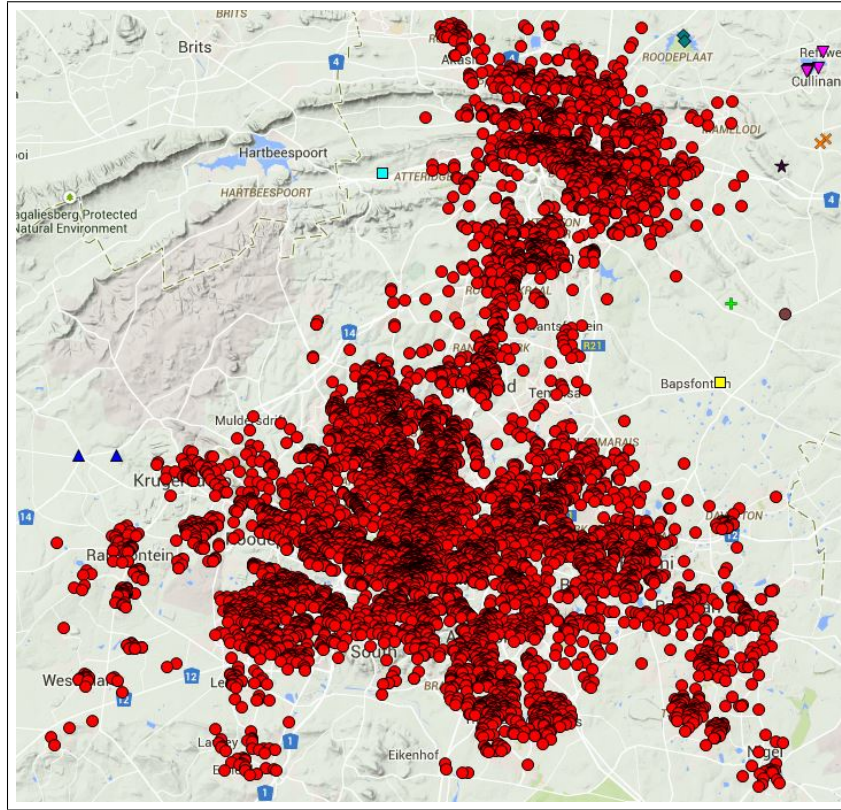
- order by demand, w_i , and
- order by the demand to proximity distance ratio, e.g. $\frac{w_i}{r(i)}$.

The last suggestion was used in cases with varying demand where the proximity measures alone could not create capacitated clusters. Ordering the customers based on a demand to distance ratio allowed customers with the biggest demand to be moved first while there was still enough space in a cluster to move them.

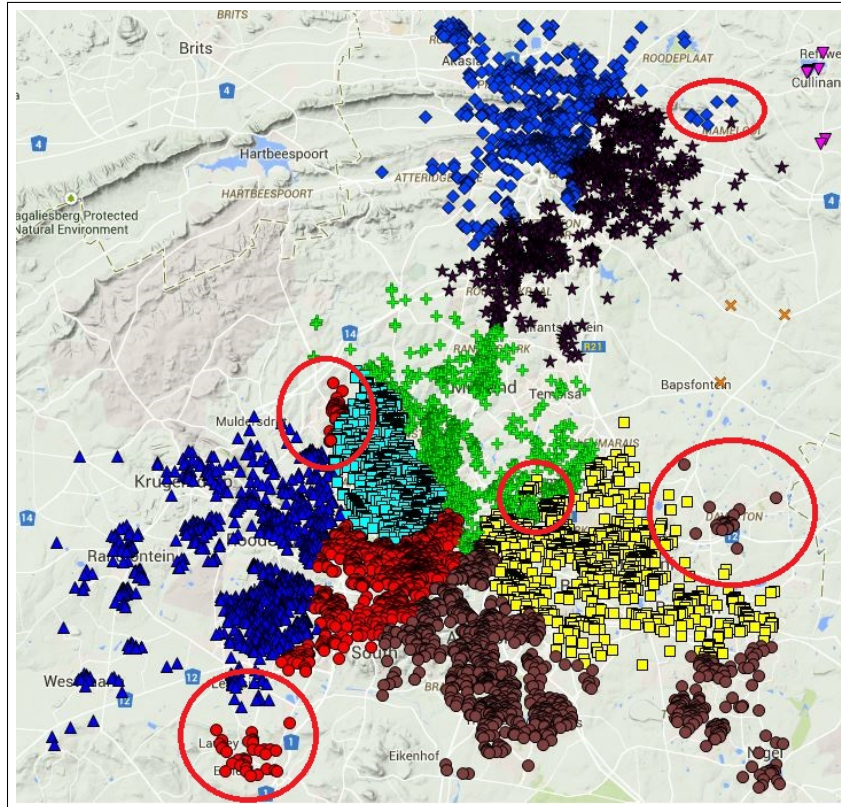
5.2.4.2 Improvement phase

The improvement phase is proposed to prevent clusters from overlapping and customers from being re-assigned to other clusters in isolation. Similar to the k-nearest clustering method, a customer is assigned to the same cluster as its h nearest neighbours. In the case where the nearest neighbours belong to different clusters, the customer is assigned to the cluster used the most by its nearest neighbours. The difference with the k-nearest method is twofold. The first is a predetermined cut-off distance. Secondly a standard deviation of the distances between points are calculated to exclude customers that are too far away from the nearest neighbour set. This is similar to the technique used in the graph-based methods to identify inconsistent edges.

5.2 Solving the pCLRP



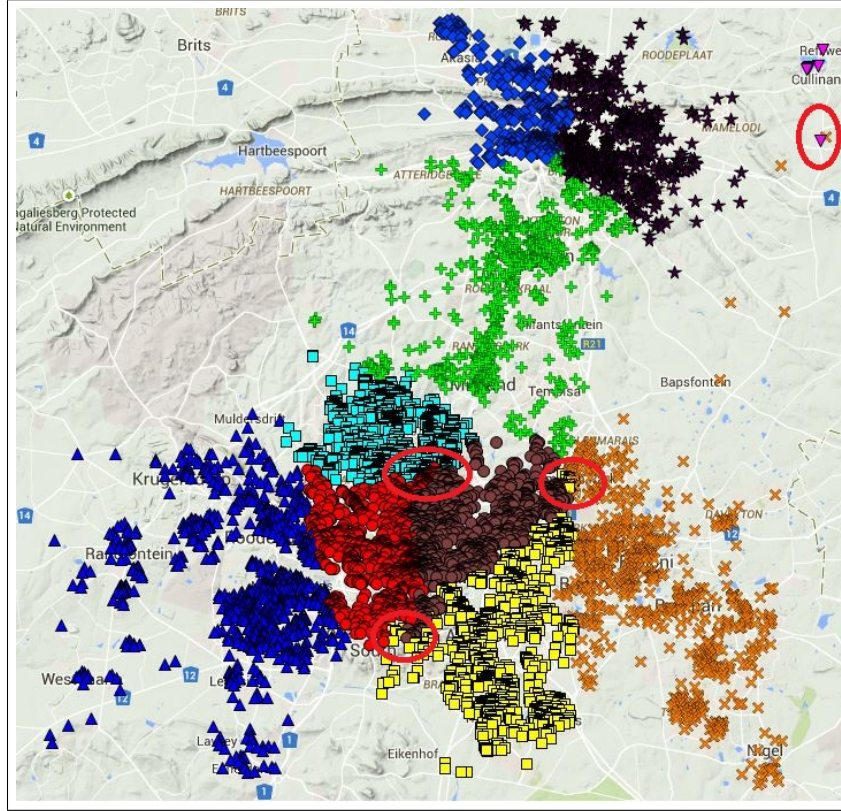
(a) The unconstrained single linkage method using DataS3.



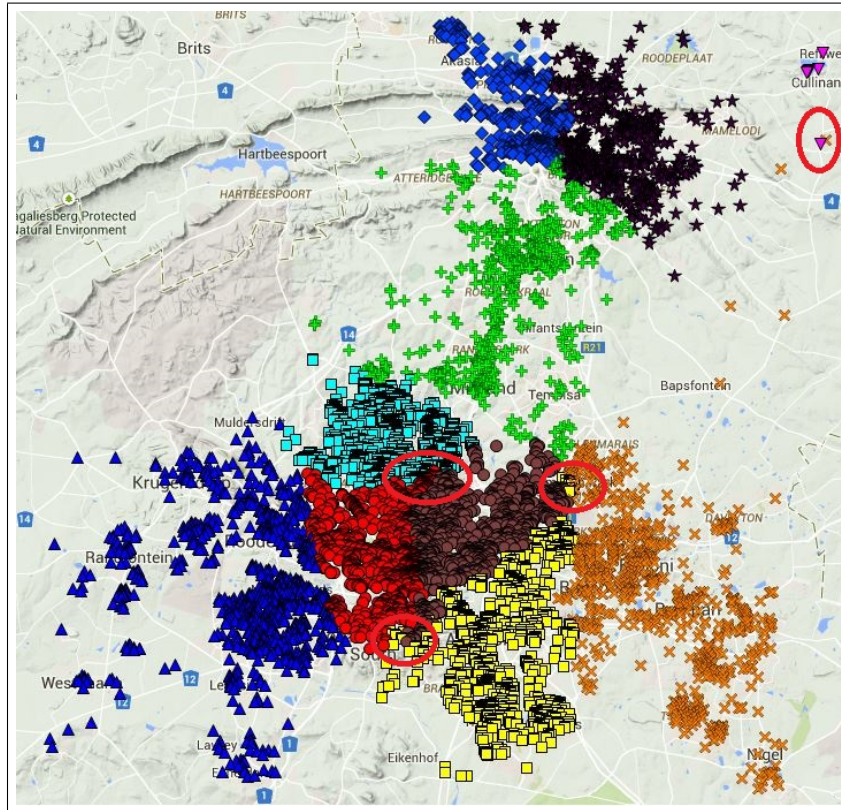
(b) Regret function $r_1(i) = d_{2_{nd}}$; Eq. (5.28).

Figure 5.6: Different regret functions for the single linkage method using DataS3.

5.2 Solving the pCLRP



(a) Regret function $r_2(i) = d_{2_{nd}} - d_{cur}$; Eq. (5.29).



(b) Regret function $r_3(i) = d_{tot} - d_{2_{nd}}$; Eq. (5.30).

Figure 5.7: Different regret functions for the single linkage method using DataS3 (cont. a).

5.2 Solving the pCLRP

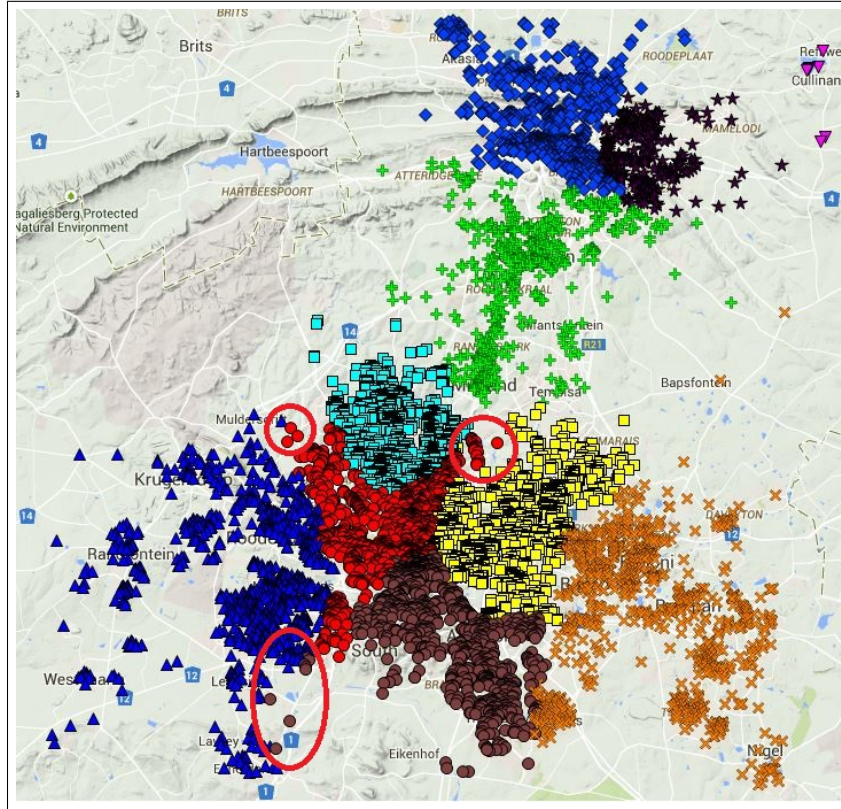
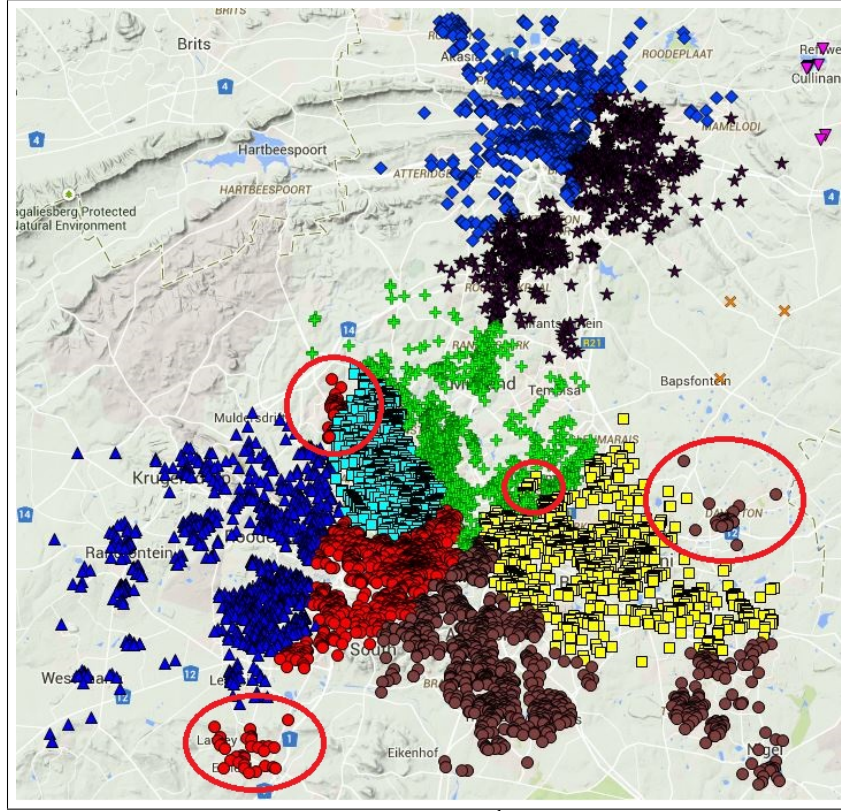
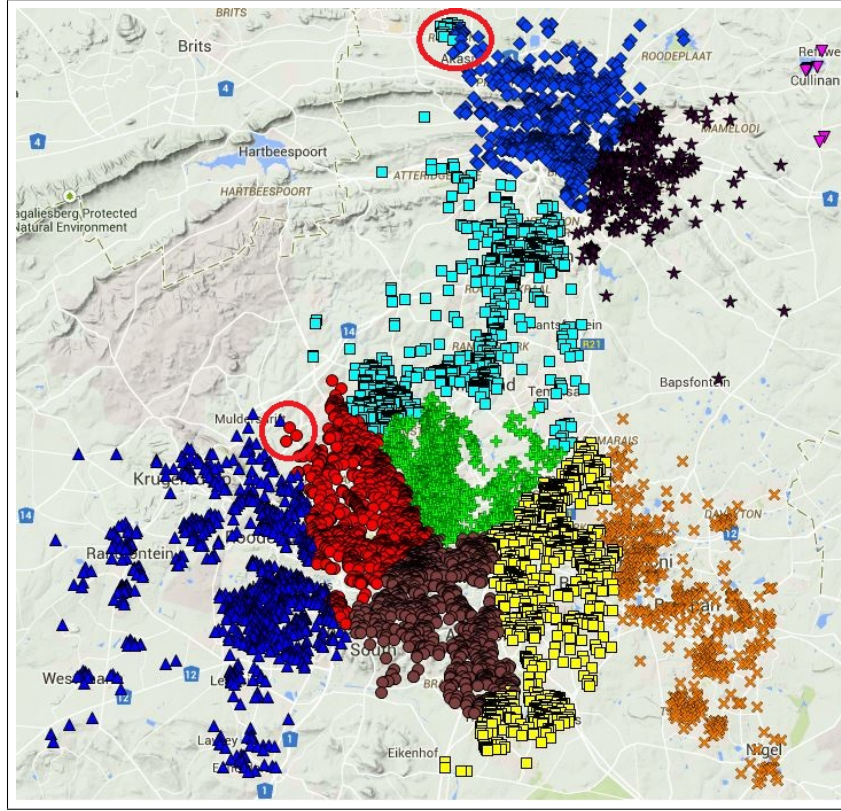
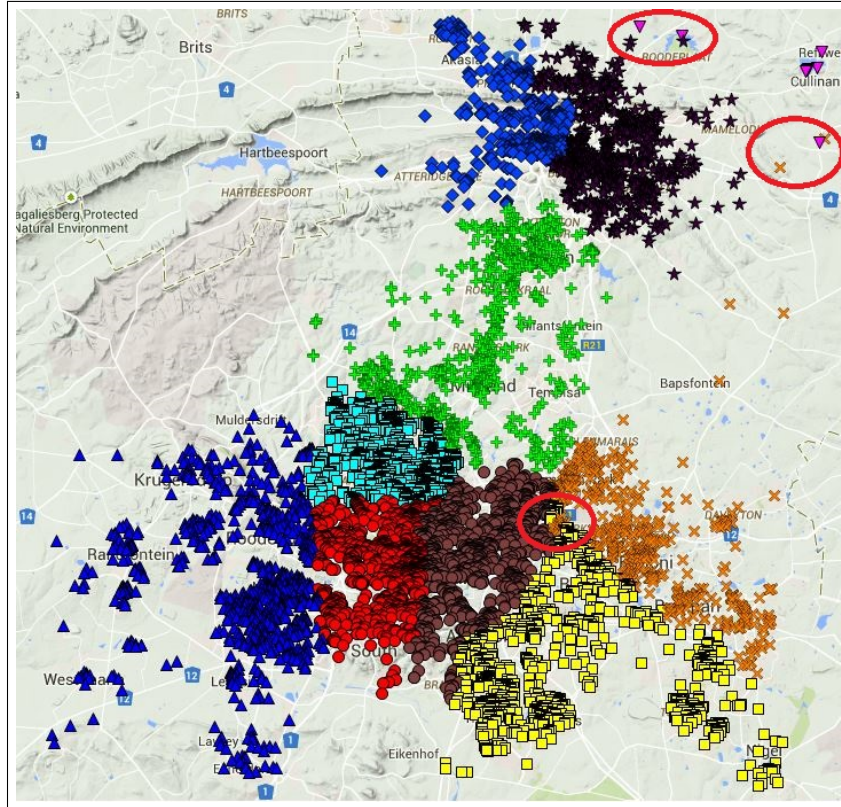


Figure 5.8: Different regret functions for the single linkage method using DataS3 (cont. b).

5.2 Solving the pCLRP



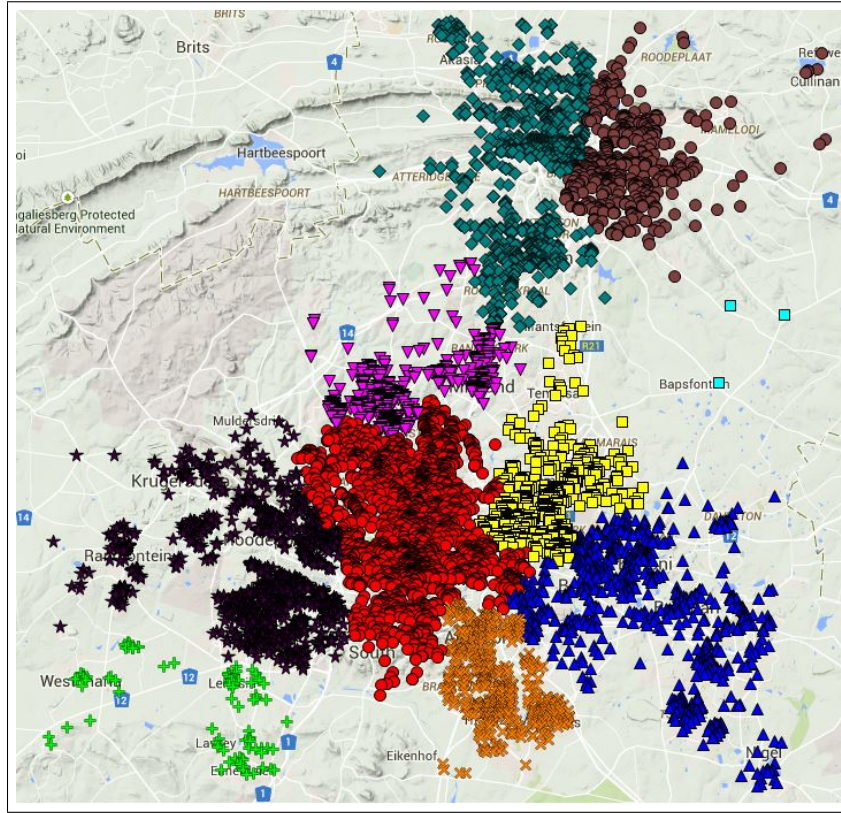
(a) Regret function $r_6(i) = \frac{d_{2nd} - d_{cur}}{d_{cur}}$; Eq. (5.33).



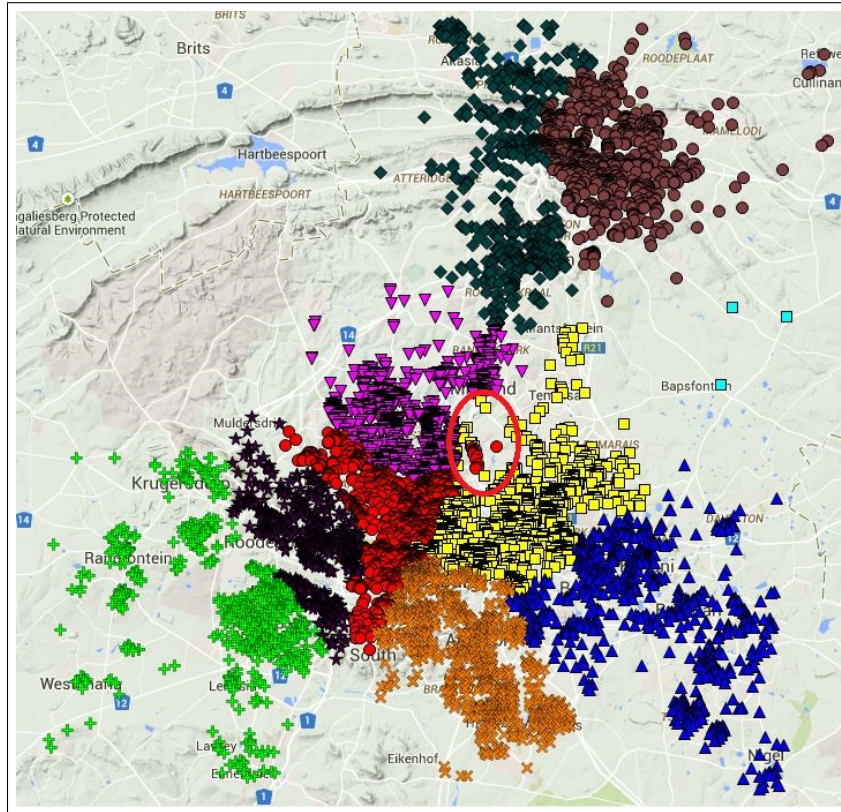
(b) Regret function $r_7(i) = \frac{d_{2nd} - d_{cur}}{d_{tot}}$; Eq. (5.34).

Figure 5.9: Different regret functions for the single linkage method using DataS3 (cont. c).

5.2 Solving the pCLR

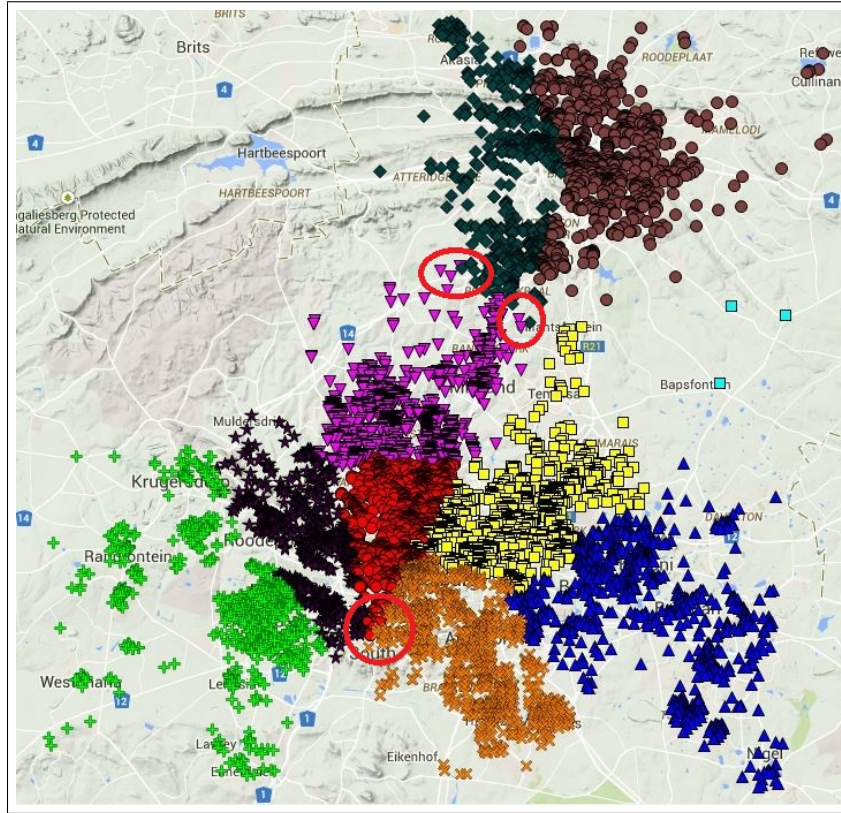


(a) The unconstrained complete linkage method using DataS3.

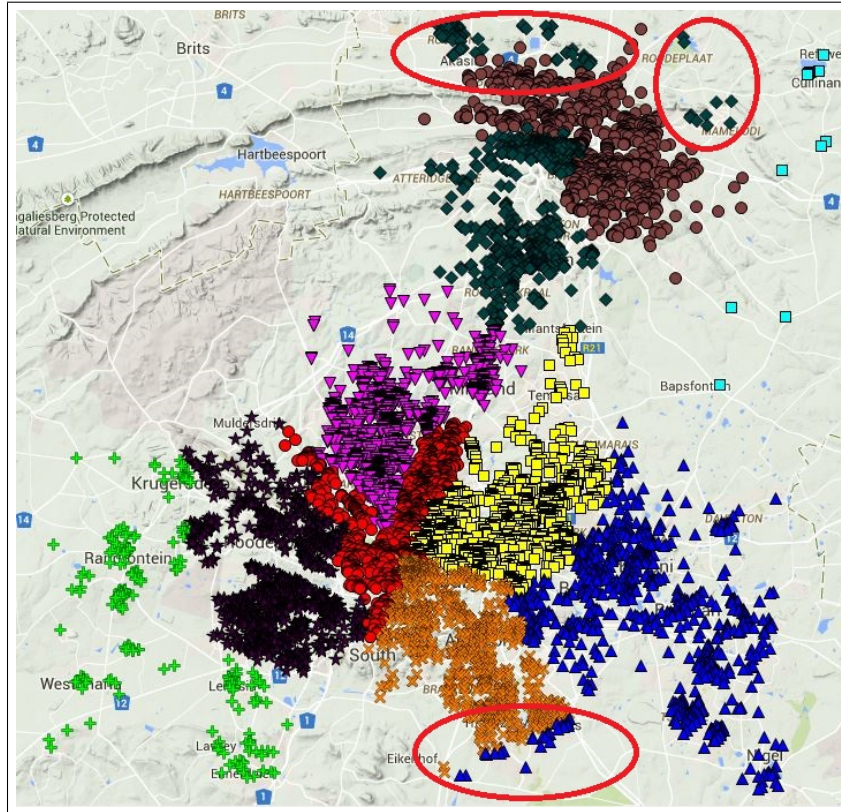


(b) Regret function $r_1(i) = d_{2nd}$; Eq. (5.28).

Figure 5.10: Different regret functions for the complete linkage method using DataS3.



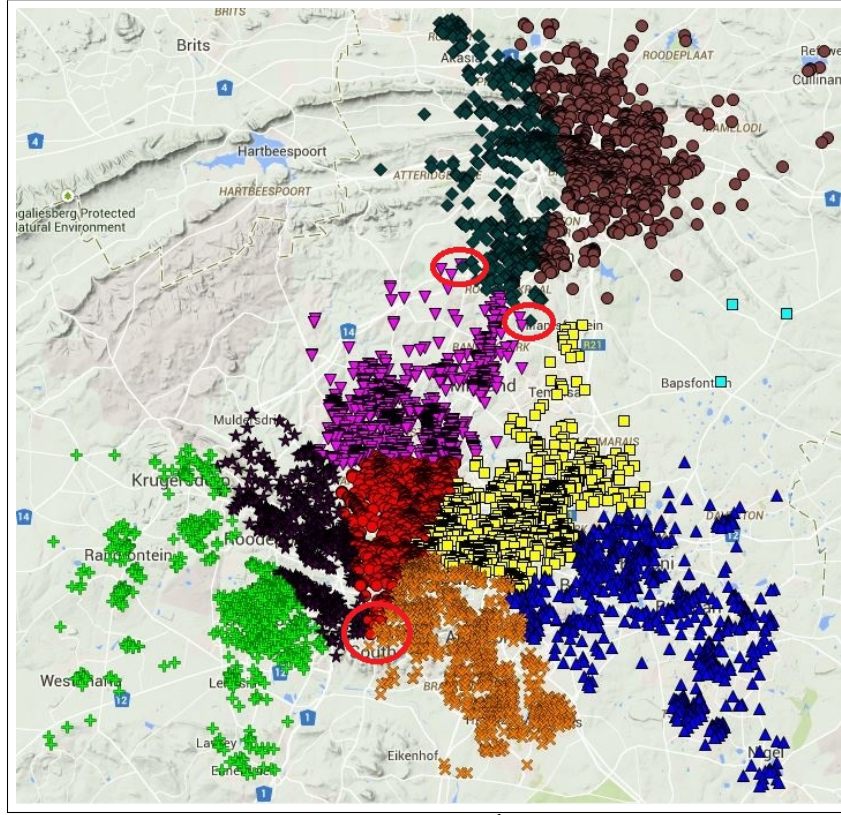
(a) Regret function $r_2(i) = d_{2_{nd}} - d_{cur}$; Eq. (5.29).



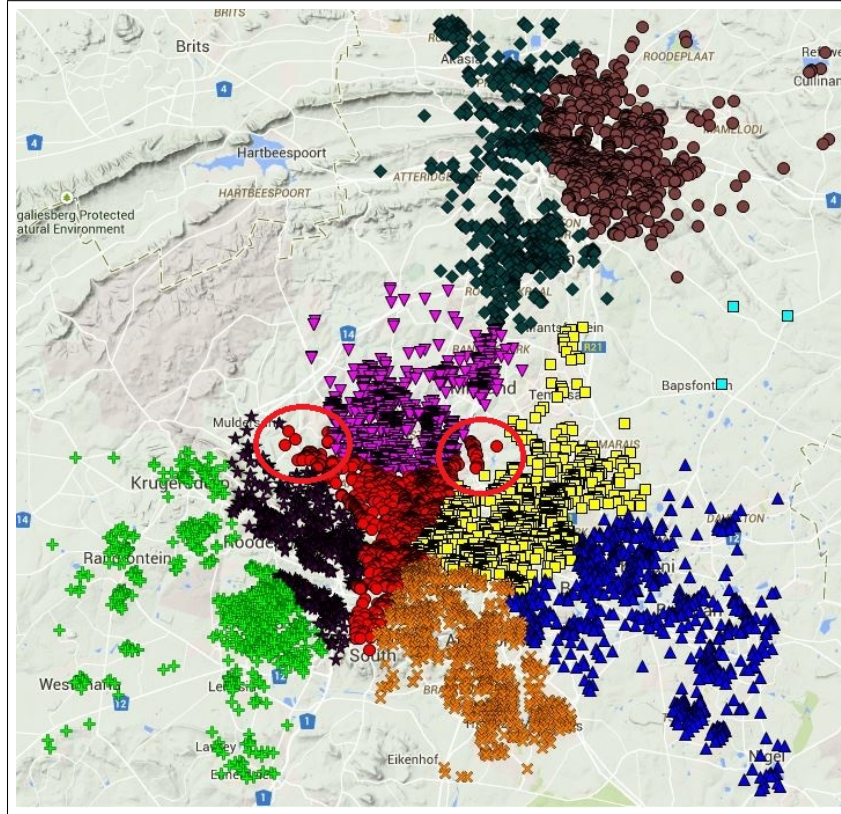
(b) Regret function $r_3(i) = d_{tot} - d_{2_{nd}}$; Eq. (5.30).

Figure 5.11: Different regret functions for the complete linkage method using DataS3 (cont. a).

5.2 Solving the pCLRP



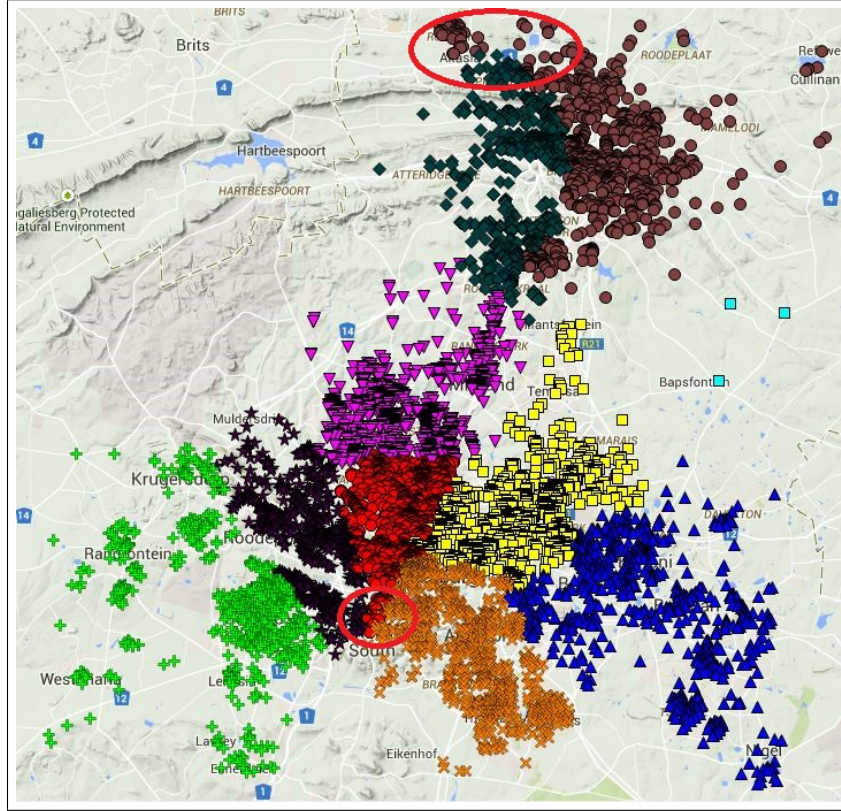
(a) Regret function $r_4(i) = \frac{d_{2nd}}{d_{cur}}$; Eq. (5.31).



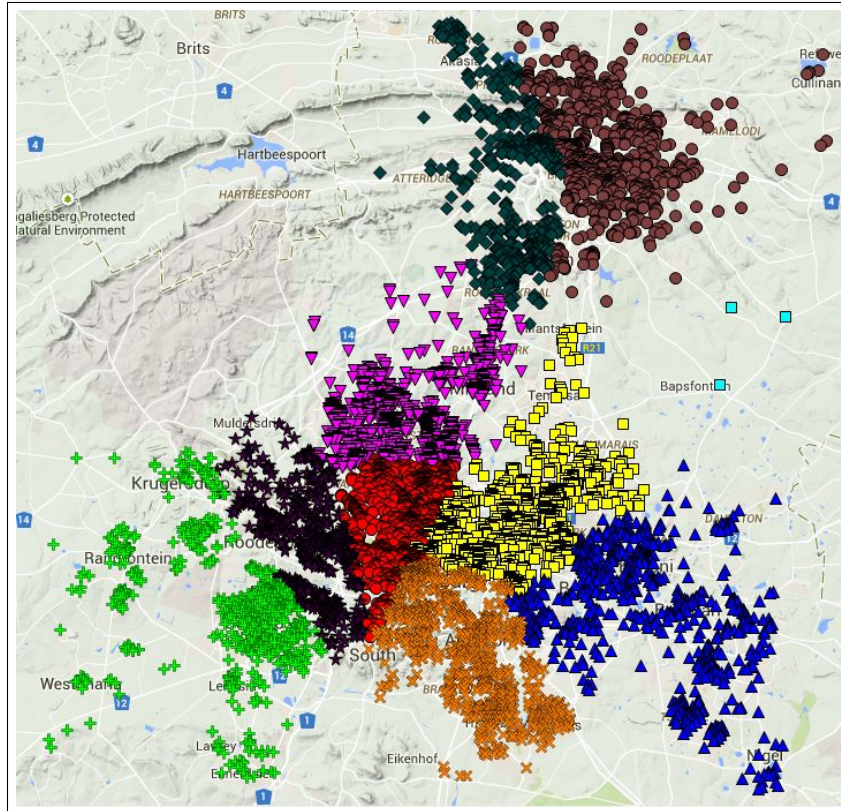
(b) Regret function $r_5(i) = \frac{d_{2nd}}{d_{tot}}$; Eq. (5.32).

Figure 5.12: Different regret functions for the complete linkage method using DataS3 (cont. b).

5.2 Solving the pCLRP



(a) Regret function $r_6(i) = \frac{d_{2nd} - d_{cur}}{d_{cur}}$; Eq. (5.33).



(b) Regret function $r_7(i) = \frac{d_{2nd} - d_{cur}}{d_{tot}}$; Eq. (5.34).

Figure 5.13: Different regret functions for the complete linkage method using DataS3 (cont. c).

Measures:	SSR		MSR		PD		ADM		AWCD		CPU times	
Linkage type	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage
Unconstrained	166 720.73	55 308.55	25.69	67.79	58.42	26.53	115 753.20	6 659.79	116 000.71	37 982.89	00 : 48	00 : 46
$r_1(i) = d_{2_{nd}}$	59 424.95	53 520.17	70.03	68.63	52.12	41.45	5 141.94	4 127.28	40 893.79	36 520.77	00 : 53	00 : 46
$r_2(i) = d_{2_{nd}} - d_{\mathbf{cur}}$	53 112.95	52 979.63	64.35	67.95	70.62	42.12	4 204.31	4 098.85	36 508.13	36 327.47	00 : 56	00 : 25
$r_3(i) = d_{\mathbf{tot}} - d_{2_{nd}}$	60 305.91	55 581.12	73.13	79.81	60.33	48.27	4 809.03	4 415.53	41 352.33	37 851.46	00 : 47	00 : 45
$r_4(i) = \frac{d_{2_{nd}}}{d_{\mathbf{cur}}}$	55 443.70	53 276.40	68.03	68.02	48.12	48.15	4 412.52	4 153.31	37 653.34	36 635.49	00 : 49	00 : 25
$r_5(i) = \frac{d_{2_{nd}}}{d_{\mathbf{tot}}}$	52 678.53	53 473.31	63.98	68.57	55.03	41.45	4 261.89	4 123.54	36 158.79	36 492.75	00 : 51	00 : 47
$r_6(i) = \frac{d_{2_{nd}} - d_{\mathbf{cur}}}{d_{\mathbf{cur}}}$	55 443.70	53 276.40	68.03	68.02	48.12	48.15	4 412.52	4 153.31	37 653.34	36 635.49	00 : 48	00 : 26
$r_7(i) = \frac{d_{2_{nd}} - d_{\mathbf{cur}}}{d_{\mathbf{tot}}}$	55 803.87	52 958.50	72.64	67.94	58.70	41.45	4 419.67	4 085.35	38 048.48	36 234.69	00 : 54	00 : 25

Table 5.4: The different proximity measures in the two-phase PROBUC method using DataS3.

In order to identify the h nearest neighbours of a customer, we propose calculating h as:

$$h = \max \left\{ b, \left\lceil \frac{n}{p} \times \sigma_a \right\rceil \right\} . \quad (5.35)$$

where n is the number of customers in the solution, p is the number of clusters and $\frac{n}{p}$ is the average number of customers per cluster. The values of σ_a and b are sensitivity parameters and can be varied per dataset. The values can be fine tuned by the user, but we recommend the starting values $\sigma_a = 0.01$ and $b = 2$.

To identify which customers to exclude from the nearest neighbours Eq. (2.12) and Eq. (2.13), used in the graph-based methods, are calculated. The cut-off values l_T and σ_T are also defined by the user and we recommend using $l_T = 0.5$ and $\sigma_T = 0.1$ values as a rule of thumb.

The customer is now re-assigned to the most used cluster that its remaining neighbours are assigned to. This creates again an unconstrained solution. The re-assignment phase is repeated to make the solution feasible again. In the first iteration it allows all customers to move from all clusters within 10% of the capacity limit to all other clusters, then it resets and only allow moves from over-utilised clusters.

5.2.5 A new cluster-based method to solve the plCLRP

Based on the two-phase PROBUC, a new cluster-based method is now proposed to solve the plCLRP. The proposed method is illustrated in Figure 5.14. An initial number of depots is calculated. This value will be optimised with respect to the objective function, Eq. (5.12), after the final route costs have been calculated. The cluster-based method consist of the following three phases:

5.2.5.1 The location phase for solving the plCLRP method

The initial number of depots is used to create depot clusters in the location phase. We recommend using the aggregated stepwise h-median method to create depot clusters, because in Chapter 4 it was able to create quality clusters with good SSR, ADM and AWCD values within reasonable times. There are however times where this clustering method can take too long to cluster, in these cases other methods, like Ward's method can also be used to create depot clusters. In order to make the depot clusters capacitated and satisfy Eq. (5.18), the proposed two-phase PROBUC method is used.

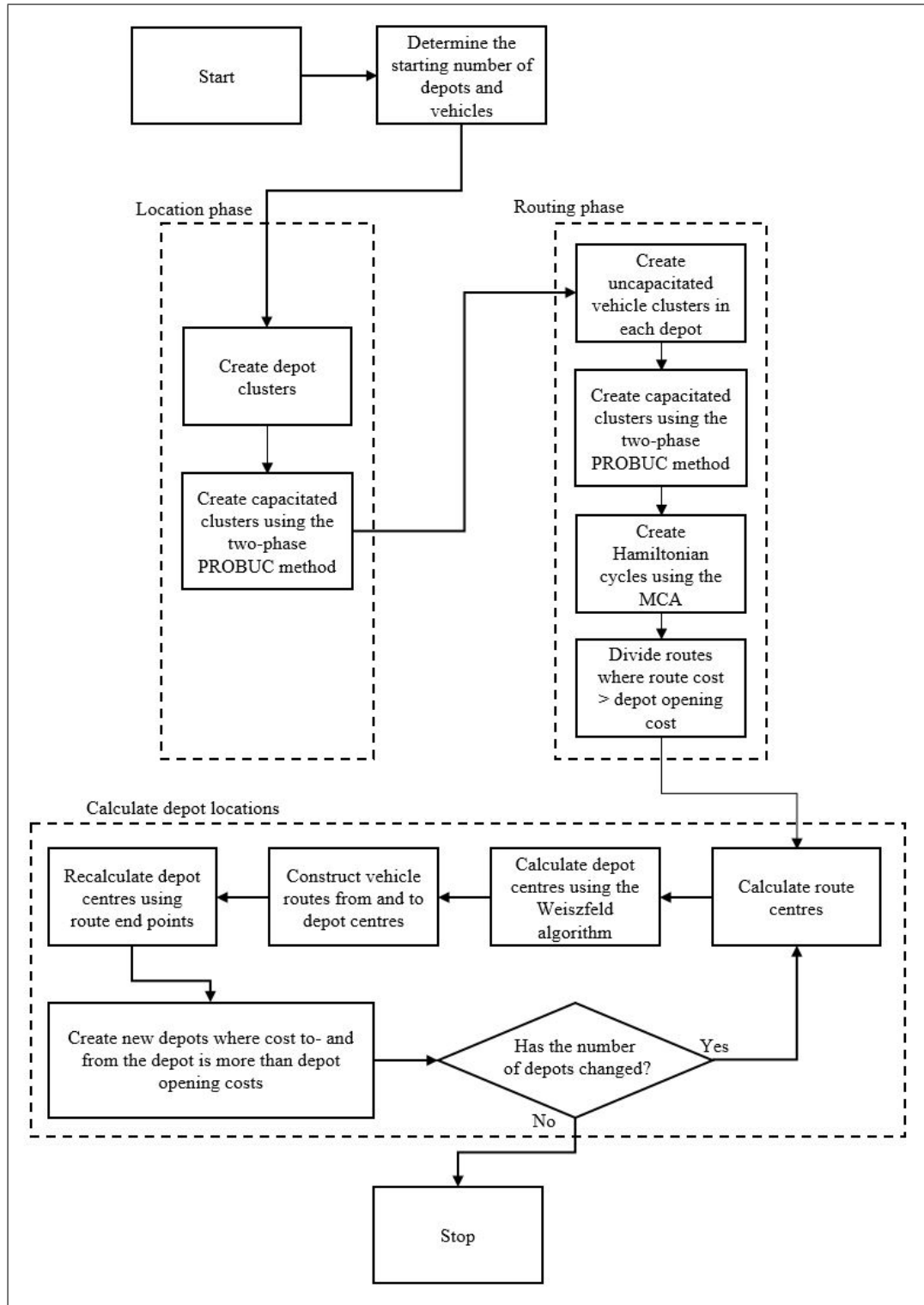


Figure 5.14: The proposed cluster-based method to solve the pCLRP.

5.2.5.2 The routing phase for solving the plCLRP method

In the routing phase, route clusters are created for every depot, based on the initial number of routes calculated per depot. By only allowing customers in the same depot cluster to be clustered in a route cluster, Eq. (5.13) and Eq. (5.19) are satisfied. Based on the results from Section 5.1, we recommend using Ward's method, the k-near method or the complete linkage method to create route clusters. The route clusters are modified to adhere to the vehicle capacity constraint, Eq. (5.14), with the two-phase PROBUC method. Similar to the HpMP, the MCA is used to construct route cycles for every vehicle that meets the constraints in Eq. (5.15) – (5.17).

If any route cycle has a route cost higher than the depot opening costs, the route must be divided. The route sequence is kept as-is, but the route is divided at the point in the cycle where the distribution costs becomes higher than the depot costs. This step can therefore change the number of routes in the solution in order to save costs in the objective function, Eq. (5.12).

5.2.5.3 Calculating depot locations in the plCLRP method

The next step is to determine the actual depot locations, as illustrated in Figure 5.15. This step is only needed in the continuous problem, because a list of potential depot locations are given in the discrete case. Here, the route centre of each route cycle is calculated as the means of all customers assigned to the route (Figure 5.15.a). The depot locations can now be calculated using the Weiszfeld equation iteratively with the route centres of all vehicles assigned to the depot as the customer points in Eq. (3.44). This is illustrated in Figure 5.15.b. The means of the calculated route centres are used as the initial depot coordinates.

To determine the final depot locations, two endpoints for each route cycle are identified. These endpoints are the adjacent pair of customers in the cycle, closest to the current calculated depot location the route is assigned to, shown in Figure 5.15.c. The Weiszfeld algorithm is now repeated for a second time, this time the route endpoints are used as the customer points, as illustrated in Figure 5.15.d. Once completed, the route costs can now be finalised using the sequence of the customers as it is in the route cycle with the two endpoints going to- and from the depot.

If the distance costs to- and from a depot for any given route are more than the costs of opening a new depot, the route is re-assigned to its own depot and the depot locations are recalculated. The number of depots in the solution can therefore change at this point, should it result in a cost saving. To create a route from the tour cycles, the endpoints are connected to the depot and the connection between the two

endpoints in the tour cycle are removed. The distances between sequential points on the route can now be added together to calculate the distribution costs. The final cost of the solution, as given in Eq. (5.12), is determined by adding together the total depot opening costs, total fixed vehicle costs and total route distribution costs.

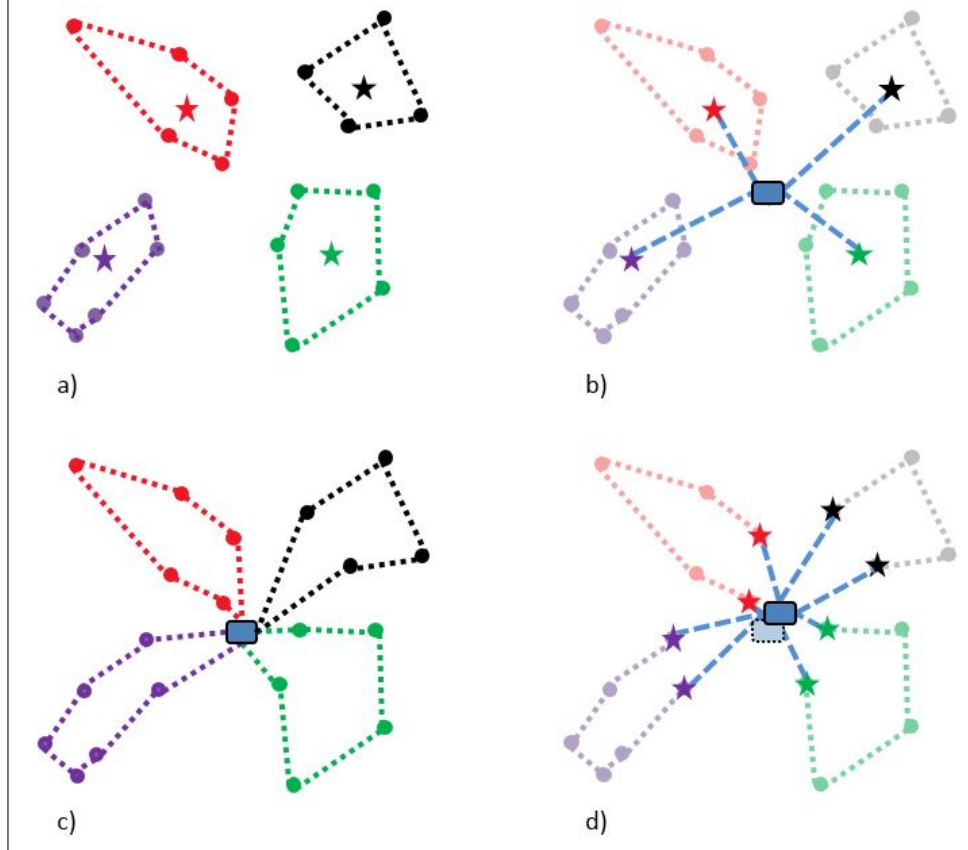


Figure 5.15: The calculation of the depot locations for the pCLRP.

5.2.6 Methodology and data preparation

The pCLRP is a new problem and instances used to test the pLRP are not comparable, because the problem does not have depot capacity constraints. The pLRP also does not make use of a fixed charge objective function to determine the number of depots to use. In order to test the use of the cluster-based approach on large datasets with up to 20 000 data points, two TSPLIB instances were selected. The instances are described as follows:

1. USA TSP data

The USA13509 instance by Reinelt (2015) used in Chapter 4 and for the HpMP is used to test the pCLRP. A unit customer demand, $w_i = 1$, and homogeneous vehicle cost value, $F = 10$, are assumed. The Euclidean distance formula, Eq.

(2.33), is used to calculate the distribution costs between points. The vehicle capacity (Q), depot costs (f) and depot capacity (W) in Eq. (5.12) are varied, resulting in eight instances, illustrated in Table 5.5.

Instance	Vehicle capacity (Q)	Depot opening cost (f)	Depot capacity (W)
Instance1	10	100	500
Instance2	20	100	500
Instance3	10	1000	500
Instance4	20	1000	500
Instance5	10	100	1000
Instance6	20	100	1000
Instance7	10	1000	1000
Instance8	20	1000	1000

Table 5.5: A list of instances tested using the TSP13509 dataset.

2. World TSP data

A WorldTSP instance with 17 237 data points, is also used to test the pCLRP. In this thesis it is referred to as WTSP17237. It is based on a subset of the World TSP provided by Cook (2013). The instance was used to benchmark solution approaches for the cdCLRP, by Alvim and Taillard (2013), but does not contain any depot capacity constraints or vehicle costs.

The formula chosen to calculate distance is the haversine formula with the spherical law of cosines simplification as given in Eq. (5.9) by Veness (2015). The earth's mean radius, R , is set to 6 371 km. The distance equation differs from the method used by Alvim and Taillard (2013), but was chosen because it resulted in quicker calculation times and can determine distances accurately anywhere on earth (Veness, 2015). The distances are converted to meters to make the values comparable to the results from the cdCLRP by Alvim and Taillard (2013).

Similar to Alvim and Taillard (2013) a standard problem instance is chosen, this is referred to as Instance1. Parameters are identified and each of the parameters are varied one by one, using the standard problem. The standard problem used here has a customer unit demand with vehicle capacity $Q = 20$. The depot capacities are kept homogeneous with $W = 1\,000$. The depot and vehicle costs are set to $f = 100\,000$ and $F = 1\,000$ respectively, in Eq. (5.12). A fleet capacity of $K_j = 100$ per depot was used.

The variables identified to vary in the sensitivity tests from the standard instance are listed in Table 5.6, resulting in 9 instances.

The best-known solutions (BKS) given by Alvim and Taillard (2013) are also provided, although the distance formula differs. These benchmark values also

5.2 Solving the plCLRP

Instance	Depot opening cost (f)	Vehicle cost (F)	Depot capacity (W)	Fleet capacity (K_j)	Vehicle capacity (Q)	Customer demand (w)
Instance1 (standard)	100 000	1 000	1 000	100	20	1
Instance2	50 000	1 000	1 000	100	20	1
Instance3	200 000	1 000	1 000	100	20	1
Instance4	100 000	10 000	1 000	100	20	1
Instance5	100 000	1 000	100	100	20	1
Instance6	100 000	1 000	1 000	10	20	1
Instance7	100 000	1 000	1 000	100	10	1
Instance8	100 000	1 000	1 000	100	40	1
Instance9 (variable w)	100 000	1 000	1 000	100	20	1 – 30

Table 5.6: A list of instances tested using the WTSP17237 dataset.

did not include vehicle costs, F , or depot- and fleet capacity constraints, W and K_j , and were used to solve the cdCLRP, not the plCLRP. Here we assume the benchmarks will give comparable results in the cases where the constraints have enough slack to return the same solution as the unconstrained problem provided by Alvim and Taillard (2013).

Other instances were also provided by Alvim and Taillard (2013), but contained more than 20 000 data points. In order to solve these instances, we recommend dividing the problem into subproblems as discussed in Appendix C.2. These instances fell outside the scope of this thesis.

5.2.7 Computational results

The results and CPU times from the USA13509 and WTSP17237 instances are given in Tables 5.7 and 5.8 respectively. Table 5.8 also gives the closest BKS from Alvim and Taillard (2013) for the WTSP17237 instances where they solved the cdCLRP with no depot capacity, fleet constraints or vehicle costs. The symbol, ‘–’, is used to indicate that there is no BKS related to the specific instance.

In the results, it is assumed that the distribution costs are equal to the calculated distances of a route and the total costs reflected in the objective function values calculated using Eq. (5.12). The aggregated stepwise h–median method was used to create depot clusters and Ward’s method was used to create route clusters. It was also found that different starting number of depots returned different end results. It is therefore recommended to try different starting numbers to find the best solution. Only the starting number that resulted in the minimum objective costs are listed in the results. The results are rounded to the nearest integer values.

5.2.8 Discussion

In Table 5.7, denoting the results for the USA13509 instances, it makes logical sense that the number of depots needed decreases in cases where the depot capacity increases. The number of routes are also almost double in the cases where the vehicle capacities are 10 compared to their counterpart cases where the vehicle capacities are 20. The bigger vehicle capacities also influenced the CPU times; for $Q = 10$, part of the CPU times used to construct routes were around 8 minutes versus CPU times close to a 1 minute to construct routes where $Q = 20$. This is because the MCA method, proposed to construct routes, use the branch-and-bound in cases where the number of customers per route are small and the branch-and-bound is very time consuming.

The cases where the depot costs were ten times higher than their counterparts, used less depots, although the minimum number of possible depots was not used (14 depots in the cases where $W = 1000$ and 27 depots where $W = 500$).

Instance7 was the slowest instance to solve, taking almost 23 minutes. The majority of the CPU time was spent creating route clusters. This took just below 12 minutes and was caused by both the chosen number of depots, $m = 16$, and the vehicle capacity, $Q = 10$, being low. These two factors caused the number of route clusters needed within a depot to be higher than in any of the other instances.

The instances that returned the lowest costs are Instance6 and Instance2 respectively. Both instances have a maximum vehicle capacity of 20 and lowest depot cost of 100. This indicates that the vehicle capacity and depot cost have a higher influence on the objective function value than the depot capacity constraint (having values $W = 1\ 000$ and $W = 500$ respectively).

In Table 5.8, the objective function values varied within 1% – 12% from the BKS values provided by Alvim and Taillard (2013). This could be because the problem solved by Alvim and Taillard (2013) did not have any depot capacity constraints. A different distance calculation, the haversine formula, was also used.

When the solutions found by the cluster-based approach are compared to each other, Instance2 and Instance8 returned the smallest cost values respectively. This is similar to the results provided by Alvim and Taillard (2013), but in their case, Instance8 returned the best values. Instance2 returned the best cost value, because it had the lowest depot cost ($f = 50\ 000$) and more depots could be opened causing the distribution costs of the vehicles also to be lower than in any of the other instances. In the case of Instance8, the method was able to open only 182 depots and this made the depot cost lower than any other instance.

Instance5 and Instance6 are new and have no BKS from Alvim and Taillard (2013).

In Instance5, the method used 316 depots, because of the depot capacity constraint, $W = 100$. Instance6 returned the same number of routes as the standard problem, Instance1, but used more depots to ensure the fleet capacity constraints ($K_j = 10$) were met.

The high percentage cap for Instance8 of 12%, compared to the other instances, could be because the vehicle capacity is higher ($Q = 40$). This could indicate that either the MCA is not very effective when used for bigger vehicle capacities or that the clustering methods are less suitable for creating routing solutions.

5.2.9 Conclusions - plCLRP

Due to the fact that the plCLRP is a capacitated problem, a method is needed to get capacitated cluster solutions from clustering methods. Different stopping rules for the SAHN methods are investigated, but the rules can only be used for hierarchical clustering methods. It became clear that another method is needed to create capacitated solutions that can be used by all clustering methods. A new unconstrained to constrained method, the two-phase PROBUC method, is introduced. Different regret functions to use in the re-assignment phase were tested. A proximity measure was chosen that returns visually attractive solutions from both balanced and unbalanced uncapacitated clustering solutions.

When compared to the results given by Alvim and Taillard (2013) for the WTSP17237 instances, the results were within 1% – 12% of the BKS. The method seems to not fare as well for instances where the number of customers per route increased.

Instances	Initial num of depots	Final num of depots	Initial num of routes	Final num of routes	Distri- bution costs	Depot cost	Vehicle costs	Total cost	Total CPU times (mm:ss)
Instance1	35	35	1 377	1 377	5 003	3 500	13 770	22 273	12 : 12
Instance2	34	34	709	709	4 479	3 400	7 090	14 969	04 : 59
Instance3	34	34	1 374	1 374	5 120	34 000	13 740	52 860	12 : 12
Instance4	30	31	756	756	5 824	30 000	7 560	43 384	07 : 04
Instance5	24	27	1 371	1 371	5 518	2 700	13 710	21 928	12 : 04
Instance6	21	21	702	702	5 250	2 100	7 020	14 370	08 : 37
Instance7	16	16	1 355	1 355	8 169	16 000	13 550	37 719	22 : 52
Instance8	15	15	679	679	6 901	15 000	6 790	28 691	03 : 41

Table 5.7: The results for the plCLRP using the USA13509 TSPLIB instances.

Instances	Initial num of depots	Final num of depots	Initial num of routes	Final num of routes	Distri- bution costs	Depot cost	Vehicle costs	Total cost (excl. veh costs)	BKS (excl. veh costs)	% gap	Total cost	Total CPU times (mm:ss)
Instance1	225	261	963	1 006	101 727 205	26 100 000	1 006 000	127 827 205	118 581 395	8%	128 833 205	24 : 35
Instance2	365	478	1 044	1 182	88 372 126	23 900 000	1 182 000	112 272 126	105 527 279	6%	113 454 126	26 : 24
Instance3	120	151	949	961	119 198 468	30 200 000	961 000	149 398 468	139 253 415	7%	150 359 468	22 : 42
Instance4	225	261	963	1 006	101 727 205	26 100 000	10 060 000	127 827 205	118 581 395	8%	137 887 205	24 : 35
Instance5	225	316	963	1 006	98 978 084	31 600 000	1 006 000	130 578 084	—	—	131 584 084	24 : 46
Instance6	225	265	963	1 006	101 488 840	26 500 000	1 006 000	127 988 840	—	—	128 994 840	24 : 47
Instance7	365	394	1 887	1 914	112 017 746	39 400 000	1 914 000	151 417 746	146 590 678	3%	153 331 746	30 : 48
Instance8	120	182	488	575	99 277 361	18 200 000	575 000	117 477 361	104 962 416	12%	118 052 361	08 : 22
Instance9	220	235	675	731	98 012 639	23 500 000	731 000	121 512 639	120 606 754	1%	122 243 639	12 : 20

Table 5.8: The results for the plCLRP using the WTSP17237 instances.

5.3 Solving the cdCLRP

Two variations of the discrete CLRP are addressed: the cdCLRP (concentrator discrete CLRP) and the sdCLRP (standard discrete CLRP). In the cdCLRP, all customers are considered potential depots, while in the sdCLRP a separate list of potential depots are given. In the literature, both problems are considered the discrete CLRP, but it normally refers to the standard discrete CLRP (sdCLRP). The problem formulation for the discrete CLRP was provided by Prins et al. (2007), given in Eq. (3.85) - (3.96). This problem formulation applies to both the cdCLRP and the sdCLRP.

The generality of the depots in the cdCLRP forces standard homogeneous depot costs and supply capacities to be used. A cluster-based solution approach to solve the cdCLRP is proposed. The approach is similar to the approach introduced for the plCLRP, but the location phase is solved differently. Routes created from Hamiltonian cycles are discussed in more detail in the routing phase.

5.3.1 A new cluster-based method to solve the cdCLRP

Tuzun and Burke (1999) observed that the discrete CLRP is often solved by splitting it into two subproblems: the location phase and the routing phase. According to Nagy and Salhi (2007) when using a cluster-based approach the problem is normally split into a clustering phase, location phase and the routing phase.

In this thesis, a new cluster-based approach is presented where the clustering phase is incorporated into the location and routing phases rather than preceding them. This is followed by another phase where the locations of the depots are finalised.

The new cluster-based solution approach is illustrated in Figure 5.16. The method starts by calculating an initial number of depots and vehicles. Depending on the costs, the initial number of depots and vehicles can be calculated using either equation sets Eq. (5.22) – (5.24) or Eq. (5.25) – (5.27), as described in Section 5.2.2. The method consists of three phases as follows:

5.3.1.1 The location phase for solving the cdCLRP method

In the location phase, the method creates depot clusters using a clustering method, as described in the plCLRP. The aggregated stepwise h-median method is recommended to create depot clusters. Should this method take too long, it can be replaced by a SAHN clustering method like Ward's method or a hybrid SAHN - partitioning clustering method, as discussed in Section 4.3.6.2. The two-phased

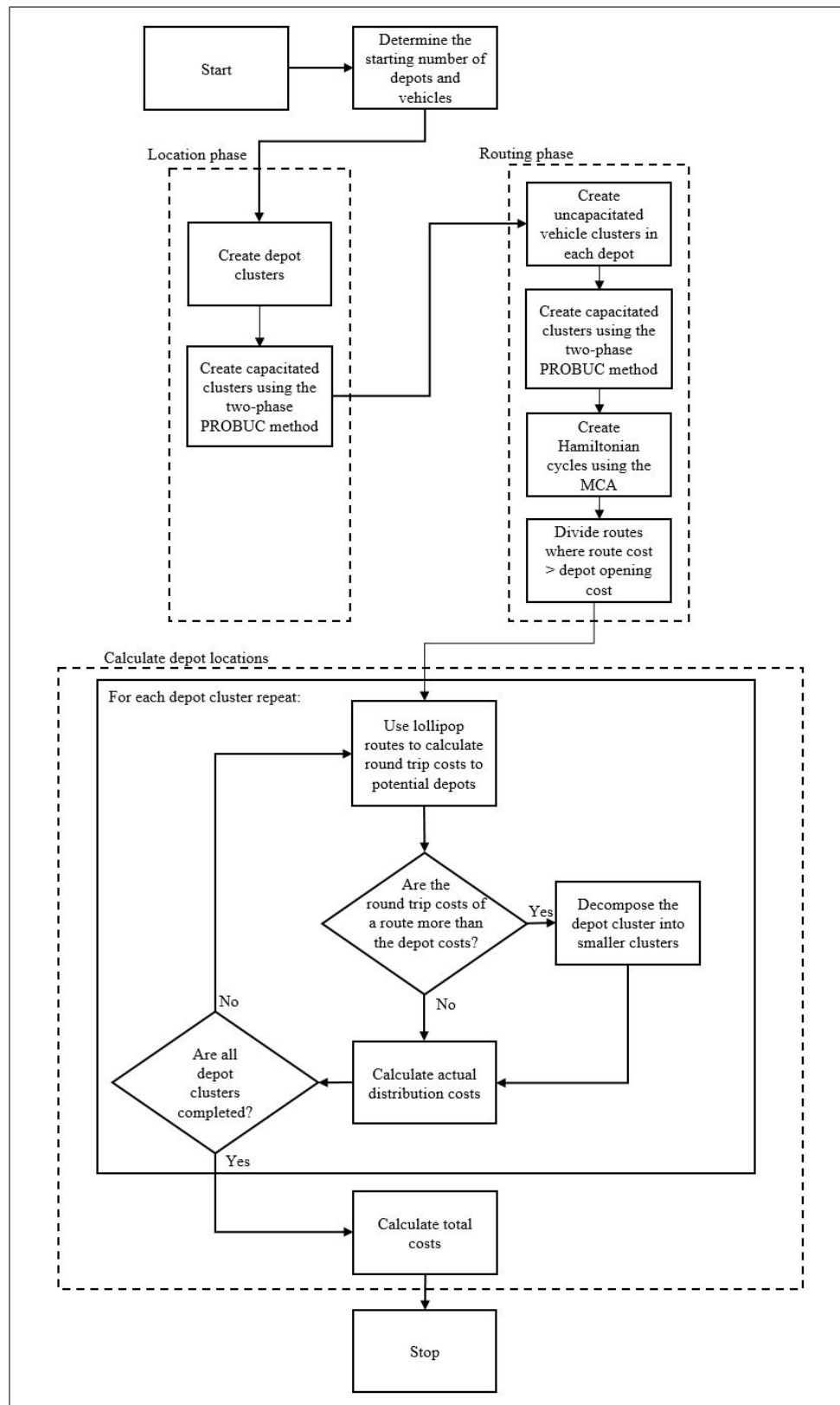


Figure 5.16: The proposed cluster-based method to solve the cdCLRP.

PROBUC method is used to create capacitated depot clusters to ensure Eq. (3.92) is satisfied.

5.3.1.2 The routing phase for solving the cdCLRP method

Each depot's customers are clustered further into route clusters in the routing phase. We recommend using Ward's method to create the route clusters. The two-phased PROBUC method is used to create capacitated route clusters that meet the vehicle capacity constraints from Eq. (3.87). Afterwards, the routes are established by constructing Hamiltonian cycles with the use of the MCA. The MCA is used to ensure the constraints in Eq. (3.86), Eq. (3.88) – (3.91) and Eq. (3.93) are satisfied. A route is split into multiple routes if the cost of the route cycle created by the MCA is more than the cost of opening a depot, in Eq.(3.85).

5.3.1.3 Proposed method to find depot locations using lollipop routes

The method can now search for better depot locations using the route cycles as-is. This is done using lollipop routes to estimate the distribution costs associated with each route. All the customers on a route can then be seen as a single customer point representing the whole route when calculating the depot locations. Similar to the location based heuristic, suggested by Simchi-Levi and Bramel (1997), this approach also divides the route costs into two parts.

1. The costs to travel to- and from the depot (round trip costs), and
2. the route costs involved in visiting each individual customer.

Here, we define the round trip costs as the distance from a particular depot to the nearest customer on the route and back. In our proposed method, a Hamiltonian cycle is created for every route. The approximate distance cost of a route is therefore the distance of the Hamiltonian cycle plus the round trip costs. These are referred to as lollipop routes because of the type of route graphs created by this approach.

For each depot cluster, all the customers assigned to the depot cluster are considered potential depot locations. The method looks at one depot cluster at a time and tries to find the best depot position. The distances between each potential depot and the closest customer in the route cycles are used to determine the distance to the depot for a particular route.

The lollipop routes will be equal to or larger than the total route costs, because the nearest customer is visited twice. By eliminating the last leg of the Hamiltonian cycle and returning directly to the depot, the distance could be shortened or at most

remain the same. Lopes (2011) used the closest two customers to calculate the route distance; The difference between this and the lollipop route approach is illustrated in Figure 5.17.

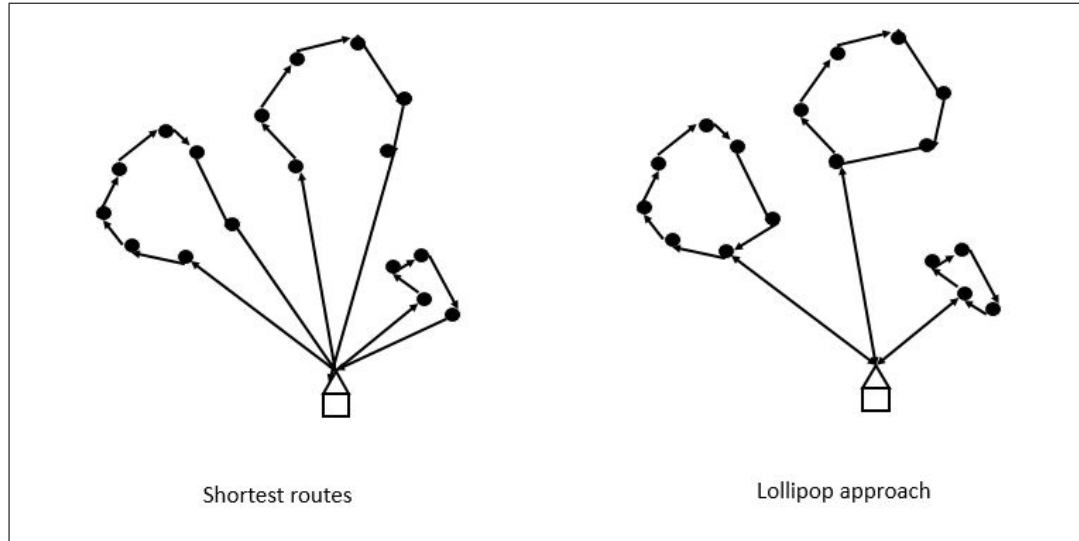


Figure 5.17: The difference between using the closest two customers as endpoints versus lollipop routes to calculate the distance of a route.

The lollipop routes are particularly helpful when trying to find better depot locations for the current solution. If the round trip costs of a specific route are more expensive than the cost of opening a new depot, the method needs to divide the depot cluster into more depot clusters. This is done by clustering the routes into smaller depot clusters. Each route cluster centre is considered a single customer point, similar to the capacitated hierarchical clustering heuristic of Lam and Mittenenthal (2013).

In order to determine the correct number of depot clusters to divide the current depot cluster into, the method iterates from two depots to the maximum possible number of depots, which is one depot per route. In every iteration the method uses iterative partitioning to cluster the routes into new depot clusters. For every iteration the objective function costs, including the total depot opening costs, the fixed vehicle costs and the total distribution costs of the routes, are calculated using Eq.(3.85). The iteration that results in the smallest objective function costs becomes the number of depots to use for this depot cluster. Once the correct number of depots has been finalised for all depot clusters, the method stops and returns the objective function value of the best solution.

5.3.2 Methodology and data preparation

The same two TSPLIB instances tested for the plCLRP, described in Section 5.2.6, were tested. The difference here is that depots cannot be placed anywhere on a

plane, but are restricted to customer positions. The parameters that are varied in the instances are listed in Tables 5.5 and 5.6.

5.3.3 Computational results

The results and CPU times for both the USA13509 and WTSP17237 instances to solve the cdCLRP are given in Tables 5.9 and 5.10 respectively. Similar to the cluster-based method for the plCLRP, the aggregated stepwise h-median method was used to create depot clusters and Ward's method was used to create route clusters. The results are rounded to the nearest integer values.

5.3.4 Discussion

In Table 5.9, similar to the plCLRP case studies, Instance6 and Instance2 were the instances with the highest vehicle capacity of 20 and lowest depot cost of 100 and returned the lowest objection function costs.

In Table 5.10, Instance2 - the instance with the lowest depot costs ($f = 50\,000$), were again the solution with the lowest objective function costs. The cost values differed from the BKS values between 2% – 13%. The results were similar to the plCLRP results in Table 5.8 and used the same number of depots and routes.

5.3.5 Conclusions - cdCLRP

As expected the results from the cdCLRP in Tables 5.9 and 5.10, were similar, but not as good as those found for the plCLRP in Tables 5.7 and 5.8. This is because a solution for the discrete concentrator problem can at most be as good as it's continuous counterpart. For both instances the number of depots and routes used in the plCLRP and cdCLRP solutions were the same. The CPU times are also comparable with the results of the plCLRP, because only the calculation method for depot positions has changed.

Instances	Initial num of depots	Final num of depots	Initial num of routes	Final num of routes	Distri- bution costs	Depot cost	Vehicle costs	Total cost	Total CPU times (mm:ss)
Instance1	35	35	1 377	1 377	5 036	3 500	13 770	22 306	12 : 16
Instance2	34	34	709	709	4 525	3 400	7 090	15 015	05 : 00
Instance3	34	34	1 374	1 374	5 158	34 000	13 740	52 898	12 : 16
Instance4	30	31	756	756	5 841	31 000	7 560	44 401	07 : 06
Instance5	24	27	1 371	1 371	5 565	2 700	13 710	21 975	12 : 15
Instance6	21	21	702	702	5 301	2 100	7 020	14 421	08 : 37
Instance7	16	16	1 355	1 355	8 240	16 000	13 550	37 790	23 : 05
Instance8	15	15	679	679	6 995	15 000	6 790	28 785	03 : 42

Table 5.9: The results for the cdCLRP using the USA13509 TSPLIB instances.

Instances	Initial num of depots	Final num of depots	Initial num of routes	Final num of routes	Distri- bution costs	Depot cost	Vehicle costs	Total cost (excl. veh costs)	BKS (excl. veh costs)	% gap	Total cost	Total CPU times (mm:ss)
Instance1	225	261	963	1 006	102 903 843	26 100 000	1 006 000	129 003 843	118 581 395	9%	130 009 843	24 : 35
Instance2	365	478	1 044	1 182	89 476 462	23 900 000	1 182 000	113 376 462	105 527 279	7%	114 558 462	26 : 25
Instance3	120	151	949	961	120 087 886	30 200 000	961 000	150 287 886	139 253 415	8%	151 248 886	22 : 52
Instance4	225	261	963	1 006	102 903 843	26 100 000	10 060 000	129 003 843	118 581 395	9%	139 063 843	24 : 36
Instance5	225	316	963	1 006	100 092 138	31 600 000	1 006 000	131 692 138	—	—	132 698 138	24 : 48
Instance6	225	265	963	1 006	102 503 843	26 500 000	1 006 000	129 003 843	—	—	130 009 843	24 : 48
Instance7	365	394	1 887	1 914	113 576 748	39 400 000	1 914 000	152 976 748	146 590 678	4%	154 890 748	30 : 49
Instance8	120	182	488	575	100 105 972	18 200 000	575 000	118 305 972	104 962 416	13%	118 880 972	08 : 23
Instance9	220	235	675	731	98 944 945	23 500 000	731 000	122 444 945	120 606 754	2%	123 175 945	12 : 21

Table 5.10: The results for the cdCLRP using the WTSP17237 instances.

5.4 Solving the sdCLRP

The sdCLRP is the standard discrete CLRP. It differs from the cdCLRP because a list of potential depots is given as a separate list of points, not necessarily restricted to customer locations. This list is usually much smaller than the total number of customers. The depot costs and supply capacities can be either homogeneous or varied. The problem formulation for the discrete CLRP as given by Prins et al. (2007) can be used for the sdCLRP. It is Eq. (3.85) - (3.96) described in Section 3.3.5.

The cluster-based approach used to solve the cdCLRP cannot be used to solve the sdCLRP. We first discuss why the approach needs to be adapted, before presenting a new cluster-based method for solving the sdCLRP.

5.4.1 The effect of using the cdCLRP location phase in the sdCLRP

In the cdCLRP, the depot costs and supply capacities are in general homogeneous and each customer is a potential depot. A solution can therefore be found by selecting a customer assigned to each created depot cluster in the location phase. This differs from the sdCLRP where the depot costs and supply capacities can vary and the potential depots are not necessarily close to the depot clusters found by the clustering method. The approach, proposed for the cdCLRP, to assign the depot clusters to potential depots after the clusters have been created, can therefore not guarantee good results for the sdCLRP. The success of finding a good solution is highly dependent on the location of the potential depots with regards to the depot clusters and the costs and capacity constraints of the chosen depot.

In addition, no two depot clusters in the cdCLRP will ever find the same cluster centre, because it is a single-source problem and only customers assigned to the depot cluster can become the centre. In the sdCLRP, the number of potential depots are generally significantly smaller than the number of customers. Moreover, the potential depots are not necessarily evenly distributed and it is possible that two clusters can find the same potential depot the closest. When multiple depot clusters are assigned to the same potential depot, the depot clusters are effectively merged.

In the case of the sdCLRP, cluster assignments cannot be used as-is when a potential depot could not be found near a depot cluster. This is because the cluster will not be centred around the chosen depot. The cluster-based cdCLRP method also does not take variances in depot costs and supply capacities into consideration when picking depots. It is possible that the closest potential depot might not be able to meet the demand of the complete depot cluster. This would result in another depot being chosen to meet the demand, instead of choosing a depot with more capacity a bit further away that would have been more cost effective.

To illustrate the ineffectiveness of using the location phase of the cluster-based cdCLRP to solve the sdCLRP, we refer to Figure 5.18. A hundred potential depots were identified for DataS3 in Figure 5.18(a). The depots were distributed evenly across the solution plane at suitable locations. The solution returned by the single linkage method for DataS3 using ten clusters as described in Section 4.3.1, is used. Figure 5.18(b) illustrates the results when the potential depot locations nearest to the depot cluster centres are chosen. Nine of the ten clusters found potential depots far from the actual depot cluster centres. The three clusters in the top right corner found the same nearest potential depot and will merge if this solution is to be made discrete as-is.

From this illustration it is clear that the cluster-based method for the cdCLRP is not effective to solve the sdCLRP. We now present a new cluster-based method to solve the sdCLRP improved with logic to choose between depots, taking varying supply capacities and depot costs into account.

5.4.2 A new cluster-based method to solve the sdCLRP

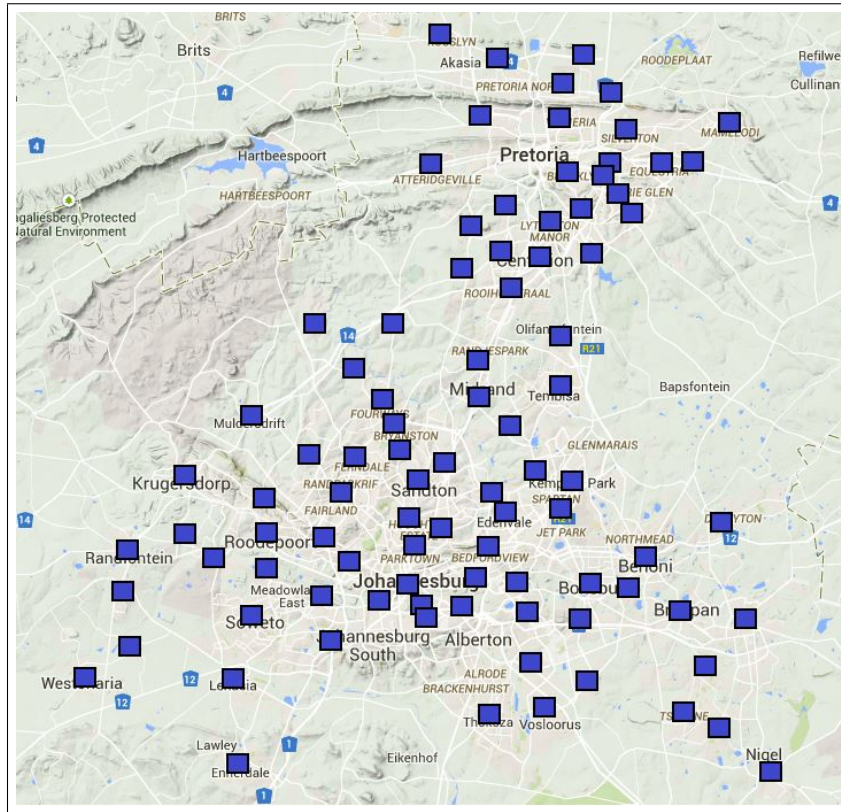
The cluster-based method for the sdCLRP is illustrated in Figure 5.19. It starts with the routing phase and is followed by a location phase. In order to choose depots in the location phase, the routing costs involved need to be known in advance. The sequence of the location and routing phase is therefore swapped so that the location phase is preceded by the routing phase.

In order to create route clusters at the beginning, all customers are assigned to a single depot cluster and use the routing phase method proposed in the cdCLRP. The route clusters are fixed and cannot be modified in the location phase. The proposed location phase makes use of a perturbation heuristic with a depot cost rate to rank depots. The two phases are now discussed in more detail.

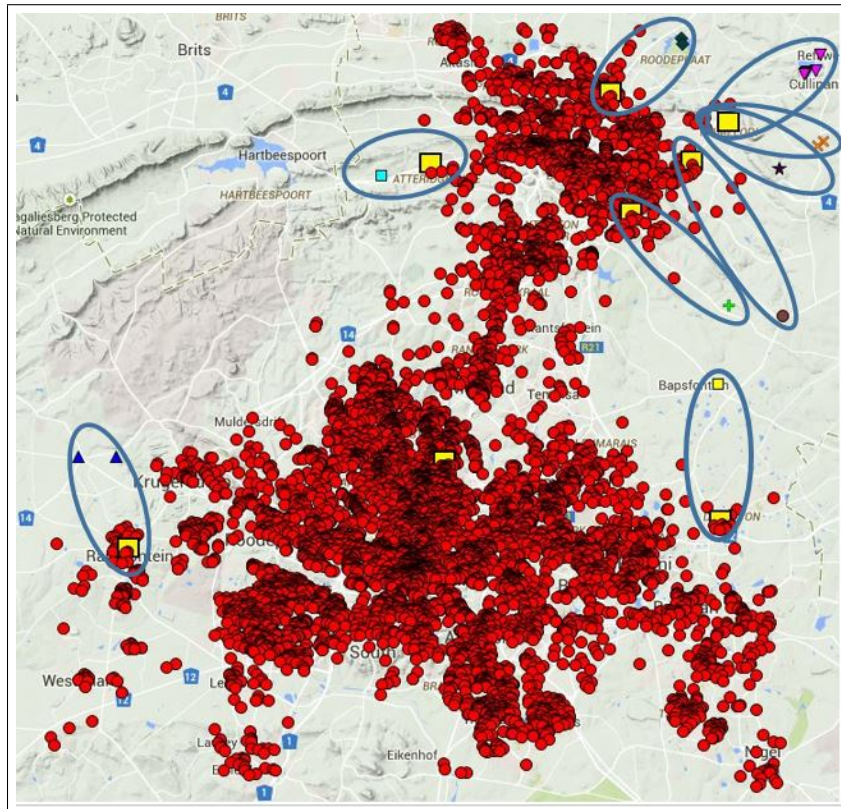
5.4.2.1 The routing phase for solving the sdCLRP method

The routing phase is similar to the routing phase of the cluster-based cdCLRP method. Route clusters are created using a clustering method like Ward's method. The two-phased PROBUC method is used to create capacitated route clusters that meet the vehicle capacity constraints from Eq. (3.87). This is followed by the MCA that creates route cycles and ensures that the constraints in Eq. (3.86), Eq. (3.88) – (3.91) and Eq. (3.93) are satisfied. The number of routes are also updated should the route cost of a vehicle be more than the cost to open a depot in Eq.(3.85).

5.4 Solving the sdCLRP



(a) The 100 potential depots for the discrete CLRP using DataS3.



(b) An example using a single linkage clustering solution together with the location phase of the cluster-based cdCLRP to find depot centres.

Figure 5.18: An example of using the location phase method suggested for the cdCLRP to solve the sdCLRP.

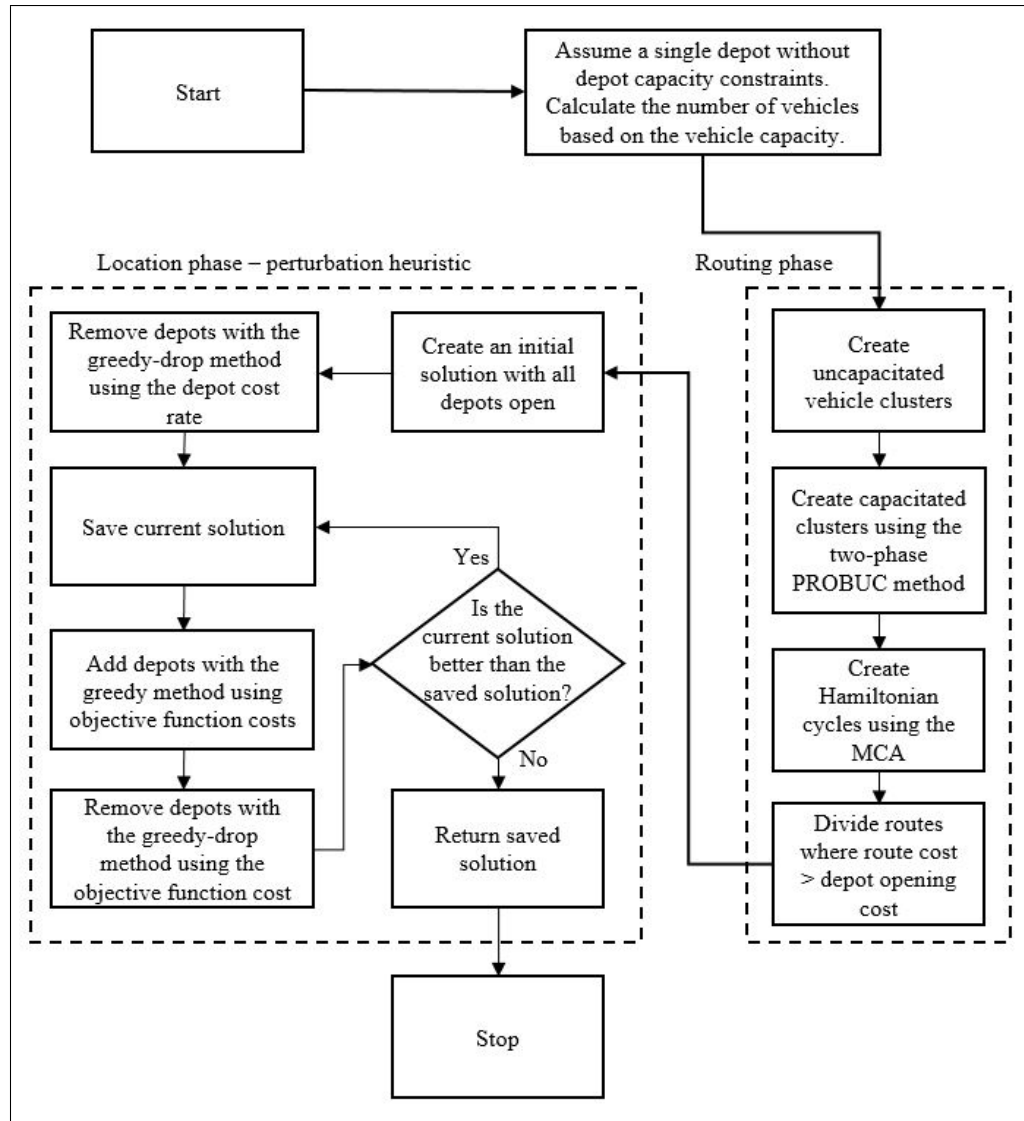


Figure 5.19: The proposed cluster-based method to solve the sdCLRP.

5.4.2.2 The location phase for solving the sdCLRP method

To determine the best set of depots to open, a heuristic is needed that considers how much supply each depot has, the depot opening costs, as well as the locations of the depots in relation to other depots. We now introduce the *depot cost rate*, described as follows:

- 1) For a particular set of chosen depots, assign all customers to their closest depot.
- 2) For each chosen depot: sum together the demand of all customers assigned to the depot, this is referred to as the demand of the depot.
- 3) If the demand is more than the depot capacity can supply, the customers furthest away from the depot are unassigned until the demand is equal or less than the depot supply. These customer are now not assigned to any depot. If the demand is less or equal, the solution can remain as-is. This step ensures that the depot capacity constraints, Eq. (3.92), are met. The remaining demand of the depot that adheres to the capacity constraint is called the depot contribution.
- 4) Calculate the *depot cost rate* as the fixed opening depot cost divided by the depot contribution.

The *depot cost rate* is therefore the cost per demand of the customers that will benefit if the depot is part of the solution. In cases where the depots can meet the total customer demand, the *depot cost rate* will default to the cheapest depot to open.

A perturbation heuristic used in conjunction with the *depot cost rate* to determine the potential depots to open is now proposed. The perturbation heuristic was introduced by Salhi (1997) and is described in Section 3.1.2.7. It is a quick yet effective heuristic, that is scalable to use in larger instances.

The perturbation heuristic iteratively alternates between the greedy-drop and greedy phases. In the first greedy-drop phase, all depots are opened and the *depot cost rate* is calculated for every depot in the current solution. Depots are iteratively closed based on the highest *depot cost rate* in the current solution. The greedy-drop stops if it cannot find a depot to remove without resulting in customers becoming unassigned.

In order to use the route costs calculated in the routing phase, complete routes are always assigned to the same depot cluster when adding or removing a depot in the perturbation heuristic. Once the greedy-drop phase has stopped, the costs of the objective function value, using Eq.(3.85) by Prins et al. (2007) is calculated and saved as the current solution.

The greedy-drop phase using the *depot cost rate* is followed by the greedy phase. In the greedy phase, q depots are added to the solution iteratively. The value of q is calculated using Eq. (3.27). Every time the depot with the smallest impact on the objective function is selected.

Once q depots have been chosen, the method returns to the greedy-drop phase. This time depots are removed based on the best saving in the objective function value, given by Eq. (3.85), and not on the *depot cost rate*. The objective function value is calculated for each candidate depot that can be removed from the solution. The depot that will result in the best objective function value is then removed.

The method continues removing depots from the solution in this manner until there are no more cost improvements to be made or the solution cannot meet the depot capacity constraint should another depot be removed. The method continues cycling between the greedy-drop and greedy phases of the perturbation heuristic until no more improvements can be made. After every greedy-drop phase, the objective function value of the current solution is compared with the previous saved solution. If the current solution is worse than the saved solution the perturbation heuristic stops and the saved solution is returned. If the current solution is better than the saved solution, the current solution replaces the saved solution and another iteration of the greedy and greedy-drop phases is repeated.

Unlike the cluster-based method for the cdCLRP, the depot locations are chosen in the perturbation heuristic and do not have to be calculated afterwards. Once the perturbation heuristic has completed, the routing phase can be repeated again, using the chosen depot assignments. However, this additional step did not result in any significant improvements in the cost function and only added additional CPU time. It was therefore not included in the proposed method.

5.4.3 Methodology and data preparation

Vast amounts of instances are listed in the literature for the sdCLRP. Our focus is on big instances with up to 20 000 data points. The biggest instance we could find for the sdCLRP with a potential depot list is given by Harks et al. (2013) who created three randomly generated large scale instances for the sdCLRP. These can be downloaded from CLRLib (2014).

These instances have 1 000; 5 000 and 10 000 customers with 100; 500 and 1 000 potential depots, respectively, and are referred to as the M, L and XL instances based on their size. However, the problem solved by Harks et al. (2013) allows for multi-source customer assignments and no vehicle costs are incurred. Another difference is that all depot supply capacities are the same as the full customer demand and

did not cause any restriction on the number of depots to use. We introduce depot capacity values: 750, 2 000 and 2 500 for the three size instances, respectively, and a fixed vehicle cost of 10 per vehicle.

Three ranges of depot opening costs $f = ([0; 100]; [100; 200]; [200; 500])$ and vehicle capacity variants of $Q = (9; 100; 1\ 000)$ are given by Harks et al. (2013). The authors refer to these variations using two index numbers with values 1 – 3, where the first index number refers to the depot cost and the second to the vehicle capacity. For example, instance $L_{1,3}$ refers to the instance with 5 000 customers, 500 depots, depot opening costs $f = ([0; 100]$ and vehicle capacity $Q = 1\ 000$. Some examples of the instances are listed in Table 5.11.

	Num of customers	Num of depots	Depot capacity (W)	Depot opening cost (f)	Vehicle capacity (Q)
$M_{1,1}$	1 000	100	750	$[0; 100]$	9
$M_{1,2}$	1 000	100	750	$[0; 100]$	100
$M_{1,3}$	1 000	100	750	$[0; 100]$	1 000
$M_{2,1}$	1 000	100	750	$[100; 200]$	9
$M_{2,2}$	1 000	100	750	$[100; 200]$	100
$M_{2,3}$	1 000	100	750	$[100; 200]$	1 000
$M_{3,1}$	1 000	100	750	$[200; 500]$	9
$M_{3,2}$	1 000	100	750	$[200; 500]$	100
$M_{3,3}$	1 000	100	750	$[200; 500]$	1 000
$L_{1,3}$	5 000	500	2 000	$[0; 100]$	1 000
$XL_{1,3}$	10 000	1 000	2 500	$[0; 100]$	1 000

Table 5.11: A list of instances tested from Harks et al. (2013) for the sdCLRP.

In these instances, customer demands are varied with $w_i \in [0; 10]$. For instances with vehicle capacity $Q = 9$ (e.g. the instances ending with a ‘1’) the customer demands had to be adapted for the single source problem. All customer demands higher than 9 are changed to 9. Distances are calculated using the Euclidean distance formula given by Eq. (2.33). All combinations of all the problem variants are tested. This resulted in 27 instances in total, see Table 5.12.

5.4.4 Computational results

The results and CPU times of the 27 problem instances from Harks et al. (2013) are given in Table 5.12. The total costs are the values from the objective function given in Eq.(3.85) by Prins et al. (2007), where it is assumed that the distribution cost between points i and j , c_{ij} , is equal to the Euclidean distance between points i and j . The total cost excluding vehicle costs does not contain the fixed vehicle cost term in the objective function value. These values are compared to the BKS values

5.4 Solving the sdCLRP

listed by CLRLib (2014), because the associated problem instances did not include fixed vehicle costs.

	Num of depots	Num of routes	Distri- bution costs	Depot cost	Vehicle costs	Total cost (excl. veh costs)	BKS (excl. veh costs)	% gap	Total cost	Total CPU times (mm:ss)
M _{1,1}	74	764	9 757.80	2 648.01	7 640	12 405.80	13 478.90	−8%	20 045.80	00 : 02
M _{1,2}	25	72	3 516.48	287.11	720	3 803.59	3 499.05	9%	4 523.59	00 : 17
M _{1,3}	7	7	2 901.92	44.31	70	2 946.23	2 478.90	19%	3 016.23	00 : 04
M _{2,1}	42	764	11 986.85	5 206.09	7 640	17 192.93	17 977.00	−4%	24 832.93	00 : 02
M _{2,2}	10	72	4 171.07	1 062.18	720	5 233.25	4 468.74	17%	5 953.25	00 : 17
M _{2,3}	2	7	3 106.76	203.78	70	3 310.54	2 620.84	26%	3 380.54	00 : 02
M _{3,1}	27	764	15 359.42	6 635.96	7 640	21 995.38	22 926.80	−4%	29 635.38	00 : 02
M _{3,2}	6	72	4 742.52	1 280.75	720	6 023.26	5 345.92	13%	6 743.26	00 : 18
M _{3,3}	2	7	3 106.76	411.34	70	3 518.10	2 779.25	27%	3 588.10	00 : 02
L _{1,1}	261	3 703	22 808.60	7 355.04	37 030	30 163.64	32 325.90	−7%	67 193.64	00 : 24
L _{1,2}	101	353	7 698.31	1 415.33	3 530	9 113.64	8 106.10	12%	12 643.64	01 : 43
L _{1,3}	28	34	6 366.71	104.89	340	6 471.61	5 463.71	18%	6 811.61	00 : 26
L _{2,1}	126	3 703	31 192.42	14 922.77	37 030	46 115.19	50 229.70	−8%	83 145.19	00 : 31
L _{2,2}	29	353	10 285.09	3 230.61	3 530	13 515.70	12 059.50	12%	17 045.70	01 : 42
L _{2,3}	8	34	6 764.75	811.58	340	7 576.33	6 624.78	14%	7 916.33	00 : 25
L _{3,1}	81	3 703	39 481.20	19 172.63	37 030	58 653.84	63 905.10	−8%	95 683.84	00 : 24
L _{3,2}	15	353	12 200.22	3 483.34	3 530	15 683.56	14 372.40	9%	19 213.56	01 : 42
L _{3,3}	4	34	7 228.37	814.69	340	8 043.05	6 966.54	15%	8 383.05	00 : 26
XL _{1,1}	467	7 529	34 523.66	11 312.16	75 290	45 835.82	48 677.10	−6%	121 125.82	01 : 23
XL _{1,2}	190	721	11 465.73	1 793.12	7 210	13 258.85	11 872.00	12%	20 468.85	04 : 18
XL _{1,3}	66	75	9 080.97	198.14	750	9 279.11	7 754.66	20%	10 029.11	01 : 33
XL _{2,1}	207	7 529	49 599.71	23 612.31	75 290	73 212.02	77 580.60	−6%	148 502.02	01 : 24
XL _{2,2}	46	721	15 998.04	4 810.58	7 210	20 808.62	18 159.10	15%	28 018.62	04 : 21
XL _{2,3}	10	75	10 093.30	1 005.42	750	11 098.72	9 296.10	19%	11 848.72	04 : 52
XL _{3,1}	126	7 529	61 994.21	28 723.58	75 290	90 717.79	101 454.00	−11%	166 007.79	01 : 25
XL _{3,2}	27	721	18 942.00	5 568.75	7 210	24 510.75	21 341.50	15%	31 720.75	04 : 19
XL _{3,3}	6	75	10 576.83	1 262.54	750	11 839.38	10 389.90	14%	12 589.38	01 : 36

Table 5.12: The results for the sdCLRP using the CLRLib instances generated by Harks et al. (2013).

5.4.5 Discussion

When comparing the solutions found for the single-source problem in Table 5.12 with the BKS values found by Harks et al. (2013) for the multi-source version, the costs excluding the vehicle costs were used. This is because the multi-source problem did not have vehicle costs. The total costs excluding vehicle costs, were at most 27% more than the BKS values found for the M size (1 000 customers, 100 depots) multi-source problem. The gap decreased to a maximum of 20% for the XL size (10 000 customers, 1 000 depots) instances.

It is logical that the BKS for the multi-source solutions had better results, because more combinations to create routes are possible and there are no depot capacity constraints. It also makes sense that the costs found for the instances with vehicles

capacity, $Q = 9$, would be less than the costs of the multi-source solutions because the demand of the customers with $w_i > 9$ were made $w_i = 9$ in order to produce feasible single-source solutions. This resulted in less visits.

The CPU times for the cluster-based method took less than 5 minutes to complete for all instances. Most of the CPU time was spent solving the routing sequence of customers. In the cases where the number of customers were low enough to use the branch and bound algorithm in the MCA (the case studies with $Q = 100$), the CPU times took longer than where the number of customers resulted in the use of the enumerated Christofides algorithm in the MCA (case studies with $Q = 1\,000$).

5.4.6 Conclusions - sdCLRP

When comparing the results of the sdCLRP cluster-based method with those given by Harks et al. (2013) in the multi-source problem, the cluster-based method looks promising for bigger instances. Not only did the percentage gap lessen as the instances became bigger, but also the CPU times for the biggest instances were less than 5 minutes. It is clear that the cluster-based method is a good method to use to quickly create initial solutions for big datasets. These solutions can then be improved further with the help of a local search or other heuristic method.

5.5 Conclusions

In this chapter, cluster-based methods to solve the uncapacitated HpMP and capacitated plCLRP, cdCLRP and sdCLRP are proposed. For the HpMP a new variant of the problem formulation is suggested that excludes unnecessary depot variables.

A new problem called the plCLRP is introduced with a new problem formulation adapted from the pLLRP to include depot capacities and a fleet size constraint per depot. The objective function is changed to a fixed charge problem.

The two-phase PROBUC method is introduced to create capacitated clusters from unconstrained clusters. This method can be used to create capacitated clusters in both the location and routing phases. Seven different proximity measures are compared for the regret function of the PROBUC method when using the single linkage and complete linkage methods to create clusters. The results differed per clustering method. The proportional proximity measure given in Eq. (5.34) was chosen, because it returned the best SSR, ADM and AWCD values for the complete linkage method and gave visually attractive solutions for both clustering methods. The two-phase PROBUC method is able to create good solutions even from completely unbalanced starting solutions like the single linkage method.

A method to determine the initial number of depots and vehicles was discussed next. Since the depot opening costs are normally much higher than the vehicle fixed costs, it is recommended to minimise the number of depots first, before determining the number of vehicles to use.

The proposed cluster-based approach for the plCLRP start by determining an initial number of depots and vehicles based on the depot and vehicle costs. Next, it determines the capacitated depot clusters followed by capacitated routing clusters. Hamiltonian cycles are created as vehicle routes. Finally the Weiszfeld algorithm is used to determine the depot centres, as illustrated in Figure 5.15.

In the cdCLRP, all customers are considered potential depots. The depot clusters are created similar to the plCLRP. The routes in the cdCLRP are converted to lollipop routes to aid in determining the depot locations. Should any route have a round trip cost larger than the cost of opening a depot, the depot cluster is decomposed into multiple depot clusters.

In the sdCLRP, a separate list of potential depots is given, normally much smaller than the number of customers. A perturbation heuristic that incorporates the depot costs, depot supply and depot location into a *depot cost rate* is used to select the depots to use from the list.

From the results of the cluster-based approaches used in the four CLRP variants, it is clear that clustering methods do have a positive impact on solving the medium and large-scale plCLRP and cdCLRP. This impact can be seen in both the CPU times and the cost-effective solutions that results. The cluster-based methods, however, seem to be less effective than other methods in creating routes. They are also not effective to use in the sdCLRP where a list of potential depots with varying depot costs and supply capacities are given.

The use of Ward's method, the k-near method and the complete linkage method as clustering methods are suggested for the routing phase, because they resulted in the best routing costs for the HpMP. The aggregated stepwise h-median method is recommend for the location phase, but can be replaced by Ward's method or the complete linkage h-median hybrid method where this clustering method takes too long.

Although the cluster-based methods proposed in this chapter might not return the best solutions, the methods have proven to be effective in finding initial solutions quickly, which can then be improved further using other heuristics.

Chapter 6

A cluster-based approach to solve the PLRP

In this chapter, the focus falls on the periodic location-routing problem (PLRP) and how the proposed cluster-based approaches of the cdCLRP and sdCLRP can be adapted for the PLRP. These approaches need to be adapted to allow for multiple clusters in the same area across the given time horizons.

A modification to the current problem formulation as provided by Prodhon (2011) is also proposed. A single-source constraint is introduced to prevent customers from being serviced by different depots on different days. A constraint set is also added to restrict the maximum number of vehicles across all depots on any given day. The results of the cluster-based approach are compared to best-known solutions for existing PLRP instances. The large-scale instances generated by Harks et al. (2013) for the sdCLRP are modified for the PLRP. The results are listed as a new set of best known solutions for future studies.

6.1 Proposed adapted problem formulation

The PLRP problem is an extension of the discrete CLRP problem. It includes a time horizon dimension and allows multiple visits to the same customer over the given time horizon. The problem was introduced by Prodhon and Prins (2008) and the problem formulation was suggested by Prodhon (2011). It is described in more detail in Section 3.4.

In the current model customers are not restricted to single depot assignments and can be serviced from different depots on different days. A new problem formulation for the single-source PLRP, adapted from the formulation given by Prodhon (2011), is now proposed in Eq. (6.1) - (6.17).

$$\text{Minimise} \quad \sum_{j \in V} \sum_{i \in V} \sum_{k \in K} \sum_{l \in H} c_{ij} z_{ijkl} + \sum_{j \in J} f_j y_j + \sum_{j \in J} F T_j \quad (6.1)$$

$$\text{subject to} \quad \sum_{i \in I} \sum_{k \in K} z_{ijkl} \leq T_j \quad \forall j \in J, \forall l \in H, \quad (6.2)$$

$$\sum_{i \in I} \sum_{j \in V} w_{ilr} z_{ijkl} \leq Q \quad \forall k \in K, \forall l \in H, \forall r \in Comb_i, \quad (6.3)$$

$$\sum_{i \in V} z_{ijkl} - \sum_{i \in V} z_{jikl} = 0 \quad \forall k \in K, \forall j \in V, \forall l \in H, \quad (6.4)$$

$$\sum_{i \in I} \sum_{j \in J} z_{ijkl} \leq 1 \quad \forall k \in K, \forall l \in H, \quad (6.5)$$

$$\sum_{i \in S} \sum_{j \in S} z_{ijk} \leq |S| - 1 \quad \forall S \subseteq I, |S| \geq 2, \forall k \in K, \forall l \in H, \quad (6.6)$$

$$\sum_{u \in I} z_{iukl} + \sum_{v \in V \cap \{j\}} z_{v jkl} \leq 1 + x_{ij} \quad \forall i \in I, \forall j \in J, \forall k \in K, \forall l \in H, \quad (6.7)$$

$$\sum_{i \in I}^n w_{ilr} x_{ij} \leq W_j y_j \quad \forall j \in J, \forall l \in H, \forall r \in Comb_i, \quad (6.8)$$

$$\sum_{r \in Comb_i} b_{ir} = 1 \quad \forall i \in I, \quad (6.9)$$

$$\sum_{j \in V} \sum_{k \in K} z_{ijkl} - \sum_{r \in Comb_i} b_{ir} a_{rl} = 0 \quad \forall i \in I, \forall l \in H, \quad (6.10)$$

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I, \quad (6.11)$$

$$\sum_{j \in J} T_j \leq |K|, \quad (6.12)$$

$$z_{ijkl} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K, \forall l \in H, \quad (6.13)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \quad (6.14)$$

$$y_j \in \{0, 1\} \quad \forall j \in J, \quad (6.15)$$

$$b_{ir} \in \{0, 1\} \quad \forall i \in I, \forall r \in Comb_i, \quad (6.16)$$

$$T_j \in N \quad \forall j \in J. \quad (6.17)$$

A single-source variant of the PLRP model is proposed by adding Eq. (6.11). This prevents more than one depot assignment variable x_{ij} for a given customer i from being equal to one. It therefore forces each customer to be assigned to one and only one depot. It is proposed the Eq. (6.12) be added to force the sum of the maximum number of vehicles on any given day to be less or equal to the maximum number of vehicles available, K . Except for these two added constraints, the description of the problem formulation remains the same as the model in the literature review, given by Prodhon (2011). It is repeated here for convenience.

Given a time horizon H , consisting of l periods (normally days), each customer, i , has a set of possible combinations of visit days allowed, called $Comb_i$. The demand for a customer i can now be defined as w_{ilr} , where l specifies the day of the visit and r specifies the combination of visit days from $Comb_i$ that is used.

The maximum number of vehicles used on any given day within the j_{th} depot is called T_j . This is also referred to as the fleet of the depot, (Fu and Ishkhanov, 2004). The fixed cost associated with having a vehicle in the fleet is F . A set of K vehicles are available each day, with the maximum number of vehicles available as $|K|$. The routes have a capacity restriction of Q per day. Subsets I and J represent the set of customers and depots respectively with $V = I \cup J$. The depots each have a supply capacity of W_j that cannot be exceeded on any given day in H .

The cost to travel from a customer or depot point i to another point j is c_{ij} and z_{ijkl} is a binary variable to indicate travel between points i and j on day l using vehicle k . The fixed depot cost f_j is only incurred if the j_{th} depot is open, indicated by the binary variable y_j . The fixed vehicle cost F is multiplied by T_j , the maximum number of vehicles used on any given day by depot j . This forces the model to balance the number of vehicles used per day. x_{ij} is a binary variable to indicate whether customer i is assigned to depot j .

Eq. (6.1) is the objective function value of the PLRP. Similar to the sdCLRP, the objective function value is composed of three terms: the distribution costs, the depot opening costs and the fixed vehicle costs. Constraints (6.2) ensure that T_j is the maximum number of vehicles used on any given day by the j_{th} depot. Constraints (6.3) restrict the customer demand allowed on any given route to the vehicle capacity Q . Constraints (6.4) guarantee continuity of the routes and that each route returns to the depot of origin. Constraints (6.5) ensure that each vehicle has only one starting point and thus that every vehicle has only one route per day.

Constraints (6.6) are used to avoid cycles or subtours, while constraints (6.7) ensure that a customer is assigned to the same depot as the depot of the route he belongs to. The capacity constraints (6.8) limit the total demand of customers assigned to an open depot to the supply capacity. Constraints (6.9) ensure that only one

combination of visit days per customer is selected, represented by the binary decision variable b_{ij} . Constraints (6.10) restrict the routes to one and only one visit per customer per day. These visits are only allowed on the selected set of visit days r , referred to as a_{rl} .

The newly introduced constraints (6.11) enforce a single-source constraint and constraints (6.12) ensure the maximum number of vehicles per depot on any given day remains below the maximum allowed. Constraints (6.13) - (6.16) restrict the decision variables to boolean values, while Constraints (6.17) restrict T_j to positive integer values.

6.2 A new cluster-based method to solve the PLRP

Based on the literature review done in the PVRP and PLRP sections (see Sections 3.2.4 and 3.4.1.6), we propose a solution approach to first simplify the PLRP to a discrete CLRP that spans across all visit days. This is similar to an approach proposed by Prodhon and Prins (2008) to solve the PLRP using an iterative metaheuristic method. The daily depot supply capacities are multiplied by the time horizon to determine the total supply capacity per depot.

In the proposed approach either of the cluster-based methods described for the cdCLRP and sdCLRP can be used to solve the discrete CLRP spanning across all visit days, depending on whether the particular PLRP is a concentrator or standard discrete problem. Ward's method, the k-near method and the complete linkage method are recommended to create route clusters. In the case of the concentrator problem, the aggregated stepwise h-median method, Ward's method or the complete linkage h-median hybrid method are recommended for the location phase. The cluster-based approach to solve the PLRP is illustrated in Figure 6.1.

To enforce the single-source constraint set across the days that has been introduced, customers are assigned to depots in the location phase, before determining their visit days and routes. In the routing phase customers are assigned to mega routes spanning across all visit days. The demand of these routes must be met by the total supply capacity per vehicle across the time horizon. Depot locations are calculated as described in the cluster-based approaches of the cdCRLP and sdCRLP.

In order to solve the problem per day, the mega routes are divided into daily routes. The assignment of customers to visit days is similar to the order based heuristic suggested by Christofides and Beasley (1984), described in Section 3.2.4.3. Daily customers are assigned to the routes first, followed by assigning the customers with the biggest demand and highest visit frequencies to a day. Lastly the smaller demand and single day customers are assigned to the days with the most capacity left. The focus in this phase is to create a balanced demand across all days to minimise the fleet costs. Less emphasis is thus given to minimising route distance. The reason

for this is that customers have already been clustered when the mega route clusters were created.

Once the visit days have been allocated, it is possible that a number of partially full vehicles on the same day can be combined. The method tries to merge the routes per day for every depot. Finally, the method uses a neighbourhood search to find and merge routes of different days. The merging of routes is only possible if customers with multiple visits can still be visited on the new day combination. To prevent the single-source constraints from being violated, the search does not span across different depots.

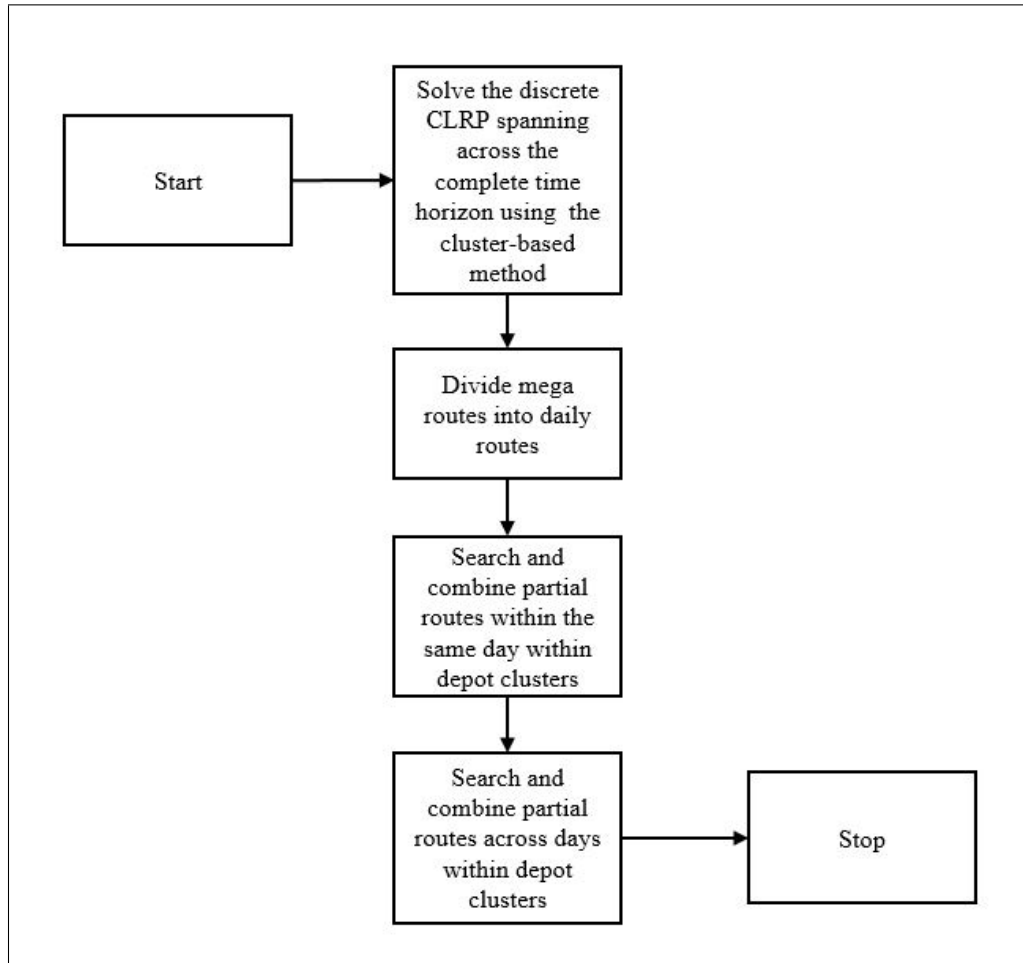


Figure 6.1: The proposed solution to solve the PLRP.

6.3 Methodology and data preparation

Two sets of instances were used to test the results:

1. **Instances from Prodhon**

Pirkwieser and Raidl (2010) gave a set of results based on the instances provided by Prodhon (2009a). According to these results, they were able to better the best-known solution (BKS) for 29 out of the 30 datasets. Only

the six instances with the biggest number of customers and potential depots (200 customers and 10 potential depots) were tested here to get an indication how the cluster-based heuristic performs on small case studies with less than a thousand customers.

As done by Prodhon and Prins (2008), the Euclidean distances were multiplied by 100 and rounded up to calculate the distance costs, c_{ij} in Eq. (6.1). In this equation, the fixed vehicle cost, F , from the CLRP was defined as a fleet cost, the cost incurred for using a vehicle at least once during the time period.

2. Instances by Harks et al. (2013) for the sdCLRP

In order to test a large case study, the same instances used by Harks et al. (2013) to test the sdCLRP were used. These instances are discussed in Section 5.4.3 and some examples are given in Table 6.1. In order to expand across a time horizon, the instances were adapted for the PLRP as follows:

- Similar to the instances by Prodhon (2009a), the time horizon is set to seven day cyclic horizon, with five working days. Saturdays and Sundays are seen as idle days.
- Visit day frequencies of 1, 2 and 3 days were sequentially introduced to all the customers in the order in which they appeared in the instance by CLRLib (2014), i.e. customer 1 has a visit frequency of 1 day, customer 2 – 2 days, customer 3 – 3 days, customer 4 – 1 day etc.
- The same visit day combinations as allowed by Prodhon (2009a) were used:
 - 1 day visit - Monday, Tuesday, Wednesday, Thursday, Friday
 - 2 day visits - Monday – Thursday, Monday – Friday, Tuesday – Friday
 - 3 day visits - Monday – Wednesday – Friday
- The same distribution of demand as used by Prodhon (2009a), is used for multiple days. Here the total demand to be delivered in one week is taken as the demand listed by CLRLib (2014). The demand to be delivered on a particular day is proportional to the number of days (including Saturday and Sunday) after the previous visit. As an example, if a customer has a demand of 7 and is to be visited 3 times in a week, the demand is distributed as: Monday – 3, Wednesday – 2, Friday – 2. More about the demand calculation can also be found in Prodhon and Prins (2008), Prodhon (2009b) and Prodhon (2011).
- In order to meet the single-source constraint for customers with a demand higher than nine, the instances specified to have a vehicle capacity limit of $Q = 9$ is changed to $Q = 10$. These are the listed instances ending with an index number ‘1’. Examples of the medium instances (customers = 1 000, potential depots = 100) are listed in Table 6.1.

- The distance cost, c_{ij} , is equal to the Euclidean distances between points x_i and x_j and was not multiplied with 100, as in the case of the instances from Prodhon.
- The fixed depot costs, f , specified in Table 6.1, are multiplied by 100 in the objective function cost calculations given in Eq. (6.1).
- The fixed fleet cost, F , the cost for using a vehicle belonging to a particular depot at least once, is set to 100. The final objective function cost value are rounded to two decimal values.

Instances	Num of customers	Num of depots	Depot capacity (W)	Depot opening cost (f) (use $f \times 100$)	Vehicle capacity (Q)
$M_{1,1}$	1 000	100	750	[0; 100]	10
$M_{1,2}$	1 000	100	750	[0; 100]	100
$M_{1,3}$	1 000	100	750	[0; 100]	1 000
$M_{2,1}$	1 000	100	750	[100; 200]	10
$M_{2,2}$	1 000	100	750	[100; 200]	100
$M_{2,3}$	1 000	100	750	[100; 200]	1 000
$M_{3,1}$	1 000	100	750	[200; 500]	10
$M_{3,2}$	1 000	100	750	[200; 500]	100
$M_{3,3}$	1 000	100	750	[200; 500]	1 000

Table 6.1: A list of instances tested from Harks et al. (2013) adapted for the PLRP.

6.4 Computational results

The results for the instances by Prodhon (2009a) and CLRLib (2014) are given in Tables 6.2 and Tables 6.3. The BKS is based on the values listed by Prodhon (2009a), Duhamel et al. (2010) and Pirkwieser and Raidl (2010).

The fleet size column gives the maximum number of vehicles used by the depots across the time horizon. It is used to determine the vehicle cost associated with the last term in Eq. (6.1). The cluster-based approach for solving the sdCLRP was used, because the instances were not concentrator problems and the depot capacities and costs varied.

6.5 Discussion

In the instances from Prodhon (2009a), listed in Table 6.2, the cluster-based method was able to better the current BKS values listed by Pirkwieser and Raidl (2010) in one instance. In the other instances the results varied with gaps between 15% – 21%. There is therefore a chance that the cluster-based method might return good results, but it should be used in conjunction with other methods.

6.5 Discussion

Instances	Num of depots	Num of routes	Fleet size	Distri- bution costs	Depot costs	Fleet costs	Total costs	BKS costs	% gap	CPU times (mm:ss)
PLRP_200_10_1a	6	99	23	288 929	94 050	115 000	497 979	431 373	15%	00 : 05
PLRP_200_10_1b	2	65	14	259 036	29 240	70 000	358 276	362 143	−1%	00 : 04
PLRP_200_10_2a	4	43	22	290 227	58 500	110 000	458 727	380 262	21%	00 : 06
PLRP_200_10_2b	2	65	14	259 271	33 240	70 000	362 511	310 110	17%	00 : 08
PLRP_200_10_3a	4	112	29	427 200	64 300	130 000	636 500	530 195	20%	00 : 08
PLRP_200_10_3b	3	70	14	274 061	54 550	70 000	398 611	342 944	16%	00 : 05

Table 6.2: The results for the PLRP for instances given by Prodhon (2009a).

Instances	Num of depots	Num of routes	Fleet size	Distri- bution costs	Depot costs	Fleet costs	Total cost	CPU times (mm:ss)
M _{1,1}	21	613	136	17 498.89	38 422.61	13 600.00	69 521.50	00 : 06
M _{1,2}	6	67	15	9 765.14	19 315.49	1 500.00	30 580.62	00 : 07
M _{1,3}	1	9	2	2 906.97	635.24	200.00	3 742.21	00 : 04
M _{2,1}	10	573	120	23 500.03	107 339.	12 000.00	142 839.03	00 : 04
M _{2,2}	2	65	14	11 670.66	25 243.5	1 400.00	38 314.16	00 : 10
M _{2,3}	1	9	2	8 137.88	10 635.2	200.00	18 973.08	00 : 03
M _{3,1}	7	572	135	26 560.08	151 637.8	13 500.00	191 697.88	00 : 05
M _{3,2}	2	72	16	11 752.07	45 771.8	1 600.00	59 123.87	00 : 07
M _{3,3}	1	9	2	8 137.88	21 905.7	200.00	30 243.58	00 : 02
L _{1,1}	81	2 892	639	43 027.83	131 727.96	63 900.00	238 655.79	02 : 35
L _{1,2}	21	391	90	22 102.08	37 529.68	9 000.00	68 631.76	01 : 31
L _{1,3}	4	41	10	18 491.11	10 289.54	1000.00	29 780.65	00 : 38
L _{2,1}	28	2 770	583	64 018.76	304 149.2	58 300.00	426 467.96	02 : 45
L _{2,2}	11	376	85	25 604.17	112 711.7	8 500.00	146 815.87	01 : 27
L _{2,3}	1	41	10	19 445.4	11 706.6	1000.00	32 152.00	00 : 39
L _{3,1}	18	2 739	566	78 245.21	394 147.2	56 600.00	528 992.41	02 : 37
L _{3,2}	5	376	85	29 976.29	111 595.7	8 500.00	150 071.99	01 : 29
L _{3,3}	1	41	10	19 445.4	25 119.9	1000.00	45 565.30	00 : 39
XL _{1,1}	148	5 741	1 261	64 046.49	160 122.06	126 100.00	350 268.55	18 : 03
XL _{1,2}	36	686	150	32 063.47	42 370.62	15 000.00	89 434.08	03 : 31
XL _{1,3}	5	75	17	27 018.01	9 927.36	1 700.00	38 645.37	02 : 10
XL _{2,1}	43	5 493	1 152	100 985.06	452 571.8	115 200.00	668 756.86	18 : 07
XL _{2,2}	14	721	160	37 626.35	147 029.7	16 000.00	200 656.05	03 : 29
XL _{2,3}	3	84	20	28 155.25	32 380.8	2 000.00	62 536.05	02 : 10
XL _{3,1}	26	5 427	1 129	126 898.61	541 216.9	112 900.00	781 015.51	18 : 15
XL _{3,2}	10	686	150	41 126.79	208 354.9	15 000.00	264 481.69	03 : 24
XL _{3,3}	2	66	14	28 566.36	42 420.9	1 400.00	72 387.26	02 : 22

Table 6.3: The results for the PLRP using the CLRLib instances generated by Harks et al. (2013).

On average the cluster-based method returned better percentage gap values for vehicle capacities, $Q = 130$, per day than for the smaller vehicle capacities, $Q = 80$, per day. This could indicate that using a clustering method in the routing phase can assist with improving the route costs in instances with bigger vehicle capacities. All instances had CPU times of 8 seconds or less.

Similar to the instances from Prodhon (2009a), the results for the instances from Harks et al. (2013) in Table 6.3, indicate that the cluster-based approach seems to be more effective for larger vehicle capacities. Here it can be seen that the method was able to produce solutions within 3 minutes, 31 seconds or less for the XL instances with $Q = 100$ (instances $XL_{1,2}$, $XL_{2,2}$ and $XL_{3,2}$) and within 2 minutes 22 seconds for the XL instances with $Q = 1\,000$ (instances $XL_{1,3}$, $XL_{2,3}$ and $XL_{3,3}$).

However, for the instances with smaller vehicle capacity, $Q = 10$, (instances $XL_{1,1}$, $XL_{2,1}$ and $XL_{3,1}$) the CPU times were over 18 minutes. Even though these instances also created more depots than the other instances, more than 16 minutes of the CPU times were spent creating the route sequences using the MCA. This could be a direct consequence of using the branch-and-bound method in the MCA, because the number of customers per route are small enough.

To test the impact of excluding the use of the branch-and-bound method in the MCA, the cluster-based approach for the PLRP was repeated for the XL size instances with $Q = 10$ (instances $XL_{1,1}$, $XL_{2,1}$ and $XL_{3,1}$). The results can be seen in Table 6.4. In Table 6.5, the results were compared to the results from Table 6.3, where the branch-and-bound method was still part of the MCA. The improvements in CPU times were significant with a saving on average of 89.4%. The difference in objective function costs was at most 0.16% worst - in the case of instance $XL_{3,1}$, and even showed an improvement of 0.43% for instance $XL_{1,1}$.

Instances	Num of depots	Num of routes	Fleet size	Distrib- ution costs	Depot costs	Fleet costs	Total cost	CPU times (mm:ss)
$XL_{1,1}$	147	5 732	1 259	64 746.2	158 118.	125 900.00	348 764.20	01 : 52
$XL_{2,1}$	43	5 486	1 144	102 387.03	452 032.	114 400.00	668 819.03	01 : 56
$XL_{3,1}$	26	5 426	1 130	128 020.	541 216.9	113 000.00	782 236.90	01 : 58

Table 6.4: The results for the PLRP using XL size instances generated by Harks et al. (2013), excluding the branch-and-bound method in the MCA.

Instances	Total cost (incl B&B)	Total cost (excl B&B)	% cost saving	CPU time (incl B&B)	CPU time (excl B&B)	% CPU saving
$XL_{1,1}$	350 268.55	348 764.20	−0.43%	18 : 03	01 : 52	−89.66%
$XL_{2,1}$	668 756.86	668 819.03	0.01%	18 : 07	01 : 56	−89.33%
$XL_{3,1}$	781 015.51	782 236.90	0.16%	18 : 15	01 : 58	−89.22%

Table 6.5: A comparison of the results for the PLRP, between including and excluding the branch-and-bound method in the MCA.

The branch-and-bound method does therefore not improve the distribution costs in the objective function value with more than 1.1%. Based on these results we therefore recommend excluding the branch-and-bound method from the MCA when creating routes for large instances. This will result in a non-exponential time algorithm. In these cases, the saving in costs does not validate the extra CPU time spent to find the improvement. This is especially relevant for instances where the distribution costs are not significantly large compared to the other terms in the objective function value.

The results for the instances from Harks et al. (2013) in Table 6.3 are the first and therefore best known solutions for these instances of the PLRP. In the case of instance $XL_{1,1}$, the best known solution is listed in Table 6.4.

6.6 Conclusions

A new problem formulation for the PLRP, including single-source constraints to ensure that customers are only serviced by one depot, was introduced. Constraints were also added to ensure that the maximum number of vehicles used on any given day remains below that total maximum number of vehicles available to all depots.

A solution approach was proposed to solve the PLRP. This is similar to a suggestion by Prodhon and Prins (2008), to first solve the PLRP as a discrete CLRP that spans across all visit days before dividing the routes into days. The PLRP can then be solved using the cluster-based approaches for the cdCLRP and sdCLRP proposed in Chapter 5. The method then tries to improve the current solution by first trying to combine routes from the same depot on the same day and next to combine routes from the same depot on different days, without breaking the visit day combination constraints while still adhering to the daily capacity limits per depot and vehicle.

The methods compared well against and returned better BKS values for one instance when compared to the values listed by Pirkwieser and Raidl (2010). The results varied when compared to the BKS values and it is recommended not to use the cluster-based method in isolation when solving the PLRP.

The instances generated by Harks et al. (2013) for the sdCLRP were modified for the PLRP. This is the first solutions for these instances of the PLRP and can therefore be used as best known solutions in future studies. The impact of the branch-and-bound method in the MCA was also compared. It was concluded that the improvement in costs are not significant enough to validate the extra CPU time spent to find the improvement. It is therefore recommended to excluded the branch-and-bound method in the MCA when solving large instances to result in a non-exponential time algorithm.

Chapter 7

Conclusions

7.1 Contribution

In this thesis the effects of using clustering methods to solve the location–routing problem was studied. When looking at larger datasets, it became clear that clustering methods can contribute towards effective methods to solve variants of the CLRP, especially where other heuristics become too time consuming to use. Care must be taken when choosing the clustering method to use. Some methods were able to find good solutions fast, while others turned out not to be effective when scaled to bigger instances.

Here follows a summary of the contributions made in the different chapters.

7.1.1 Clustering methods

From the start, certain clustering methods stood out as more effective than others, even in an unconstrained environment. Figures 4.1 - 4.6 give an overview of the results found for DataS3, to illustrate this. The h–median method was able to find the best SSR results compared to any of the other methods tested for all datasets. Although iterative partitioning methods like the h–median method requires a starting solution, the performance of the h–median method proved less dependent on the quality of the starting solution than other partitioning methods. However, when scaled to larger datasets, the stepwise function caused the method to become slower than the other iterative partitioning methods, even though the same number of starting solutions was generated. It was obvious that a different approach was needed to find starting solutions.

Although most of the SAHN clustering methods did not find solutions with results as good as iterative partitioning methods, Ward’s method did return good results

as indicated by Min (1987) and Lam (2008). The method did not outperform the h-means and h-median methods. The complete and average linkage methods also did surprisingly well. The SAHN methods have a polynomial time complexity when using the Lance-Williams parameters, but a distance matrix of size n^2 is needed. For large datasets, there is a threshold beyond which the storage needed for a complete distance matrix becomes too big to use in programming languages. Alternatives to deal with large distance matrices are discussed in Appendix C.2. It is suggested to decompose the problem into smaller subproblems as proposed by Louis and Tang (1999) (see page 314).

Hybrid methods were also tested. This is where two clustering methods are combined by first aggregating the data into small clusters that are treated as single customer points, before clustering the small clusters into the correct number of clusters. The hybrid SAHN-partitioning methods performed well but were not able to improve the SSR values of the h-median method.

The aggregation of customers in the stepwise function, used to generate solutions for the iterative partitioning methods, was also tested. This resulted in less starting solutions being generated and better CPU times. The conclusion was that although the aggregated stepwise h-median method did not find solutions better than the stepwise h-median, it still found SSR results within 0.015% of the best-known SSR value for DataS3 within 5 minutes, 23 seconds, resulting in a 22.63% CPU time saving when compared to the stepwise h-median method. The aggregated stepwise h-median method is therefore the recommended clustering method to use in large scale datasets. In the cases where a big number of clusters is required, the partitioning methods become too slow and Ward's method is recommended. This was discussed in Section 4.3.6.3.

7.1.2 The HpMP

To solve the HpMP an algorithm was needed to create Hamiltonian cycles from the points clustered together. A modified problem formulation for the HpMP was introduced and the MCA method was proposed to construct route cycles fast. This method uses the branch-and-bound for small sets of customers. The Christofides algorithm together with or without an enumerated search to construct routes depending on the number of customer points in the route, is used otherwise. Clusters created by the clustering methods had to be modified afterwards to ensure each cycle had at least three points. Based on the polynomial CPU times and route costs, Ward's method, the k-near method and the complete linkage methods were the best performers to create route clusters.

7.1.3 The plCLRP

A new problem called the plCLRP was introduced. This problem is similar to the plLRP, because it is a continuous LRP problem. Unlike the plLRP, the plCLRP is a fixed charge capacitated location-routing problem with depot capacity constraints. Homogeneous depot and vehicles opening costs as well as a depot fleet size are introduced. A new problem formulation for the plCLRP was also introduced.

To create capacitated clusters for the CLRP variations, the two-phase PROBUC method was proposed. This is an unconstrained to constrained method that creates capacitated clusters from the clustering methods used in Chapter 4. Customers are moved based on a regret function. Different regret calculations were tested and a proportional regret formulation was chosen. After the clusters have been modified to adhere to the depot or vehicle capacity constraints, clustering methods like the nearest neighbour and graph-based methods are used to identify the nearest neighbours to assist in assigning borderline customers to the most appropriate cluster in the improvement phase. A final unconstrained to constrained loop ensures that all clusters still meet capacity after the improvement phase.

The two-phase PROBUC method proved to be an effective method to create good solutions from even completely unbalanced starting solutions like those produced by the single linkage method. The CPU time is dependent on the number of moves needed to make the solution constrained, but even in the case of the single linkage method for DataS3, the re-assignment phase was able to produce capacitated clusters in less than a minute.

When solving the variants of the CLRP, the number of depots and vehicles to use is unknown beforehand and driven by the cost function. A rule of thumb is proposed to determine the starting number of depots and vehicles that adheres to the capacity constraints. This rule does not necessarily guarantee the best solution, but it does eliminate searching through multiple depot-vehicle combinations that can already be excluded beforehand.

Based on the above description, a cluster-based method is introduced to solve the plCLRP. The method starts by creating depot clusters in the location phase. It then creates route clusters in the routing phase, followed by the calculation of the depot locations in the final phase.

7.1.4 The cdCLRP

Two different discrete CLRP were identified based on the placement of the potential depots. The cdCLRP is the concentrator discrete case in which all customers serve as potential depots.

The cluster-based method introduced for the plCLRP was adapted in the final phase where the depot locations are determined. Here the concept of lollipop routes was introduced. These routes consist of Hamiltonian cycles and a path from the depot to the closest customer and back, giving the impression of a lollipop. Approximate distribution costs are calculated to simplify solving the placement of the depots. Afterwards the actual distribution costs are calculated to determine the final distribution costs for the solution.

7.1.5 The sdCLRP

The sdCLRP is a discrete CLRP where a separate list of potential depots are provided. Different depots can have different capacities and costs associated with them. It became clear that the clustering methods used in the location phase of the cluster-based method for the cdCLRP had to be replaced when solving the sdCLRP, because potential depots are not necessarily located close to the chosen depot centres. Using a clustering method in the location phase can also not take varying depot cost and depot capacity constraints into consideration when determining the best depot centre. An alternative heuristic called the perturbation heuristic in conjunction with a *depot cost rate* was proposed to solve the location phase of the sdCLRP.

7.1.6 The PLRP

A single-source PLRP problem formulation was introduced where a single-source constraint across the time horizon is suggested. A constraint to restrict the daily number of vehicles to remain within the maximum number of vehicles available is suggested as well. Different approaches were listed for solving the periodic component of the PVRP and PLRP problems in Sections 3.2.4 and 3.4.1.6.

A heuristic by Prodhon and Prins (2008) was chosen that combines the problem into one CLRP before dividing the problem into visit days. The method was able to better the current BKS values listed by Pirkwieser and Raidl (2010) for one instance. The impact of using the branch-and-bound method in the MCA on the objective function costs and CPU times were investigated. Based on the results from the test instances, we recommend excluding the branch-and-bound method in the MCA when solving larger instances.

More research can be done on how to approach the periodic component of the PLRP. Mostly random assignments with local searches or basic intuitive heuristic and metaheuristic methods have been suggested in the past.

7.1.7 Test instances

For all four variants of the CLRP and the PLRP, large-scale instances were introduced. In the HpMP a new instance was introduced based on a TSPLIB instance. In the case of the cdCLRP and plCLRP, an existing instance from Alvim and Taillard (2013) was adapted to cater for depot capacity constraints and vehicle costs. Three randomly generated large-scale instances for the sdCLRP, introduced by Harks et al. (2013) for the multi-source problem were tested and additional depot capacity constraints and vehicle costs were introduced. For the PLRP, the three instance sets from Harks et al. (2013) were adapted to incorporate a five day time horizon with varying visit frequencies.

7.2 Conclusive remarks

Although clustering methods do have a positive impact when used to solve the CLRP and PLRP, they should be seen as tools to add to the approximate algorithms toolbox. They can be used to provide initial solutions to give guidelines in decision-making and complement discrete heuristic methods. In cases where the problems are very constrained, for example where the depot capacity is very close to the demand, clustering methods are less effective and other heuristic methods will be more useful to solve the problem.

When dealing with big datasets the question that often comes to mind is: "How good is good enough?" meaning, when is a solution deemed good enough. One also needs to consider how close to getting to the optimal solution is a person willing to offer up in return for a gain in CPU time? A classic example of this compromise is shown in Table 6.5. Here we discussed excluding the branch-and-bound method from the MCA to construct route cycles. Excluding this method would cause an improvements in CPU times of 89.4% on average, while the difference in objective function costs is at most 0.16% worst.

Another example is where Alvim and Taillard (2013) limits the number of generated starting solutions to 150, in the POPMUSIC method used to solve the cdCLRP. The method resulted in a $O(n^3/2)$ time complexity by limiting the number of iterations allowed. Obviously, as the problem size grows, the 150 starting solutions represent less and less of the problem size and with this the chances of finding the best starting solution becomes smaller. Nevertheless, good solutions were still found in polynomial time with an acceptable compromise while at the same time instances with more than 10^6 customers could be solved.

In the case of DataS3, a number of clustering methods, especially hybrid methods

were able to find unconstrained solutions with SSR values below 46 000 (see Tables A.25, A.26, A.41, A.42, A.45, A.48, A.49). However, only two methods were able to find values below 45 000. Here again, we can ask how important it is to find a solution below 45 000? If we look at bigger instances, it probably would not be acceptable to conduct all promising clustering methods in the hope of finding an improvement that is less than 0.02%.

With the growth in data also comes a new shift in methodology. A change in the approach to solving large instances has become imminent, because most classic heuristic and metaheuristic methods do not have polynomial time complexity and these methods become ineffective to solve large instances. A new set of tools consisting of approximate algorithms is needed and education on the time complexity of algorithms cannot be underestimated.

The use of the wrong sorting algorithm in the clustering methods described in this study, for example, could have a huge impact on the overall CPU time of the methods. The impact of using large quantities of storage memory, accessing huge arrays and making complicated calculations on the CPU time should also not be underestimated. It is therefore clear that the need to analyse the time and storage complexity needed by algorithms to solve large-scale instances are more often becoming the focus of attention than finding the optimal solution for academia.

7.2.1 Observations regarding clustering methods

During the course of this study, many interesting observations were made with regards to the clustering methods used. One such example is the interesting pie chart distributions made by the PAM method, as illustrated in Figure 4.14. This is very similar to the sweep algorithm suggested by Simchi-Levi and Bramel (1997) to solve the VRP discussed in Section 3.2.2.5 on page 73.

The huge impact of having duplicate points, with the same longitude and latitude values, on the clustering methods was also unexpected. When displaying these duplicate points on a map, they look like a single point. In, for example, the nearest neighbour methods, these duplicates will look like a single point cluster surrounded by another cluster. Care must therefore be taken when clustering data, because it is not uncommon in real-life case studies to find multiple points geocoded to the same coordinates by geocoding tools.

In most solutions, the k-means method has become the default method to use when creating clusters. This study highlighted some interesting facts regarding the k-means method, including:

- The method is slow compared to the h-means method. This being said, care

must be taken because reference to the k-means method in the literature more often actually refers to the faster h-means method.

- The method needs a good starting solution in order to return good results and cannot be used in isolation.
- The method is outperformed by the h-median method, which is less dependent on good starting solutions to produce good results.
- The myth that the k-means method can be used to better solutions from the h-means method was shown to not be true in all cases. This was discussed on page 140.

7.2.2 Visualisation of data points

Finding visually attractive solutions, as described in Section 4.1.2 on page 117, is becoming more important to users. Techniques to display GIS points on a map has become easier than in the past. Not only is it now possible to judge solutions based on objective function costs, but also visual confirmation makes the results much more acceptable to clients. An example of how to display points on a map using GoogleMaps (2005) is given in Appendix B.

In Section 5.2.4.1 for example, visually attractiveness was used to compare solutions to determine the best proximity measure to use in the two-phase PROBUC method. The solutions of the different proximity measures are illustrated in Figures 5.6 to 5.12. Even though the results in Table 5.4 show only small variances and were inconclusive, by looking at the maps, it became easier to pick the best proximity measure based on visual attractiveness.

7.3 Areas for future research

In this study, the hybrid SAHN - partitioning clustering methods proved very successful in returning good results, but more research can be done around the volatility of the results and determining the exact number of clusters to use when dividing the solution approach into two phases.

More solution approaches can also be identified to test the impact of variations in depot cost and capacity constraints in the location phase when solving the sdCLRP. Different approaches to solve unconstrained to constrained clustering methods can also be investigated further, including focussing on more in-depth approaches for customers with varying demand. The impact of using clustering methods in the

VRP can be expanded further. More research can also be done on finding more effective algorithms to plan across time horizons for both the PVRP and PLRP.

In general, the world of distribution networks is opening up a vast number of problem variations that can be solved in the academic literature. Examples of practical restrictions are growing so fast that academics have a difficult time keeping up with proposing problem formulations and methods for all the possible variants. Examples of the VRP variants are listed by Poot et al. (1999) and Kant et al. (2008).

Sadly, a lot of businesses end up using limited basic heuristics because more complicated methods are often not understood, are too complicated to code or do not fit the exact problem they need to solve. This research is an attempt to get pass the "cluster equals k-means" methodology.

A couple of websites are also seeing the light of day where academics publish algorithms to interactively solve route schedules on the fly in an attempt to get companies more involved in finding better VRP solutions. However, one thing remains certain, the number of opportunities for academics to become involved in solving real-life case studies are vast.

References

- Alvim, A. C. and Taillard, E. D. (2013). Popmusic for the world location-routing problem. *EURO Journal on Transportation and Logistics*, 2(3):231–254.
- Ballou, R. H. (2004). *Business Logistics / Supply Chain Management: Planning, Organizing, and Controlling the Supply Chain*. Pearson Prentice Hall, Upper Saddle River, New Jersey, 5th edition. ISBN: 0-13-066184-8.
- Barreto, S., Ferreira, C., Paixão, J., and Santos, B. S. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179(3):968–977.
- Bektas, T. (2006). The multiple travelling salesman problem: an overview of formulations and solution procedures. *OMEGA The International Journal of Management Sciences*, 34:209–219.
- Bischoff, M. and Dächert, K. (2009). Allocation search methods for a generalized class of location-allocation problems. *European Journal of Operational Research*, 192(3):793–807.
- Blum, C. and Roli, A. (2003). Metaheuristics in Combinatorial Optimization : Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308.
- Branco, I. and Coelho, J. D. (1990). The hamiltonian p-median problem. *European Journal of Operational Research*, 47(1):86–95.
- Brimberg, J., Hansen, P., Mladenović, N., and Taillard, E. D. (2000). Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. *Operational Research*, 48(3):444–460.
- Brusco, M. and Stahl, S. (2005). *Branch-and-Bound Applications in Combinatorial Data Analysis*. Springer Science + Business Media, Inc., Florida, 1st edition. ISBN: 0-387-25037-9.
- Burke, E. K. and Kendall, G., editors (2005). *Search methodologies*, chapter 8. Very Large-Scale Neighborhood Search, Altner, D. S., Ahuja, R. K., Ergun, O., and Orlin, J. B., pages 211–238. Springer, Berlin. ISBN: 0-387-23460-8.

REFERENCES

- Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, 63(1):2–15.
- Ceselli, A., Libertore, F., and Righini, G. (2009). A computational evaluation of a general branch-and-price framework for the capacitated network location problems. *Annals of Operations Research*, 167(1):209–251.
- Chao, I., Golden, B. L., Wasil, E., et al. (1995). An improved heuristic for the period vehicle routing problem. *Networks*, 26(1):25–44.
- Chaves, A. A. and Lorena, L. A. N. (2010). Clustering search algorithm for the capacitated centered clustering problem. *Computers & Operations Research*, 37(3):552–558.
- Chaves, A. A. and Lorena, L. A. N. (2011). Hybrid evolutionary algorithm for the Capacitated Centered Clustering Problem. *Expert Systems with Applications*, 38(5):5013–5018.
- Chen, C.-H. and Ting, C.-J. (2008). Combining Lagrangian heuristic and Ant Colony System to solve the Single Source Capacitated Facility Location Problem. *Transportation Research Part E: Logistics and Transportation review*, 44(6):1099–1122.
- Christofides, N. and Beasley, J. E. (1984). The period routing problem. *Networks*, 14(2):237–256.
- Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.
- CLRLib (2014). The Capacitated Location Routing Instance Library (CLRLib). <http://www.coga.tu-berlin.de/clrlib>. Cited 24 September 2014. Last updated 13 August 2014.
- Cook, W. (2013). World TSP. INTERNET. <http://www.math.uwaterloo.ca/tsp/world/index.html>. Cited 16 February 2015. Last updated 11 September 2013.
- Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.
- Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Chapter 6 vehicle routing. In Barnhart, C. and Laporte, G., editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367 – 428. Elsevier. ISSN: 0927-0507.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, Massachusetts, 3 edition. ISBN: 0-262-03384-4.
- CPLEX. IBM ILOG. INTERNET. <http://www.ilog.com/products/cplex/>. Cited 19 September 2014.

REFERENCES

- Drexl, M. and Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308.
- Drexl, M., Schneider, M., and Professorship, D. S. E. A. (2014). A survey of the standard location-routing problem. Technical report, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).
- Drezner, Z. and Hamacher, H., editors (2001). *Facility Location: Applications and Theory*. Springer-Verlag, Heidelberg, Germany, 1st edition. ISBN: 3-540-21345-7.
- Drummond, L., Ochi, L. S., and Vianna, D. S. (2001). An asynchronous parallel metaheuristic for the period vehicle routing problem. *Future generation computer systems*, 17(4):379–386.
- Duhamel, C., Lacomme, P., and Prodhon, C. (2010). A GRASPxELS approach for the capacitated location-routing problem. *Computers & Operations Research*, 37:1912–1923.
- Erlenkotter, D. (1978). A Dual-Based Procedure for Uncapacitated Facility Location. *Operations Research*, 26(6):992–1009.
- Everitt, B., Landau, S. L., Leese, M., and Stahl, D. (2011). *Cluster analysis*. John Wiley & Sons, Ltd, West Sussex, United Kingdom, 5th edition. ISBN: 978-0-470-74991-3.
- Fisher, M. L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124.
- Foster, B. A. and Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly*, (1976):367–384.
- Francis, P., Smilowitz, K., and Tzur, M. (2006). The period vehicle routing problem with service choice. *Transportation Science*, 40(4):439–454.
- Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In *The vehicle routing problem: latest advances and new challenges*, pages 73–102. Springer.
- Fu, L. and Ishkhanov, G. (2004). Fleet size and mix optimization for paratransit services. *Transportation Research Record: Journal of the Transportation Research Board*, (1884):39–46.
- Geetha, S., Poonthalir, G., and Vanathi, P. T. (2009). Improved K-Means Algorithm for Capacitated Clustering Problem. *INFOCOMP Journal of Computer Science*, 8(4):52–59.

REFERENCES

- Glaab, H. and Pott, A. (2000). The hamiltonian p -median problem. *The electronic journal of combinatorics*, 7(R42):2.
- Gollowitzer, S., Gouveia, L., Laporte, G., Pereira, D. L., and Wojciechowski, A. (2014). A comparison of several models for the hamiltonian p -median problem. *Networks*, 63(4):350–363.
- Gonzalez, T. F., editor (2007). *Handbook of approximation algorithms and meta-heuristics*. Chapman & Hall/CRC computer & information science, Danvers, MA 01923, USA, 1st edition. ISBN-13: 978-1-58488-550-4, ISBN-10: 1-58488-550-5.
- GoogleAPI (2014). Free Google Maps JavaScript API v3. INTERNET. <https://developers.google.com/maps/documentation/javascript/>. Cited 23 September 2014.
- GoogleMaps (2005). INTERNET. <http://www.google.com/maps/>. Cited 23 September 2014.
- Gould, R. (2012). *Graph theory*. Dover books on mathematics, Dover publications, USA, dover edition edition. ISBN-13: 978-0-486-49806-5, ISBN-10: 0-486-49806-9.
- Grimm, E. C. (1987). CONISS: a FORTRAN 77 program for stratigraphically constrained cluster analysis by the method of incremental sum of squares. *Computer and Geosciences*, 13(1):13–35.
- Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*. Morgan Kauffman Publishers, Elsevier Inc., 2nd edition. ISBN: 978-1-55860-901-3.
- Hansen, P. and Jaumard, B. (1997). Cluster Analysis and Mathematical Programming. *Mathematical Programming*, 79(1-3):191–215.
- Hansen, P. and Mladenovic, N. (1997). Variable Neighbourhood Search for the p -median. *Location Science*, 5(4):207–226.
- Hansen, P. and Mladenović, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34(2):405–413.
- Harks, T., König, F. G., and Matuschke, J. (2013). Approximation algorithms for capacitated location routing. *Transportation Science*, 47(1):3–22.
- Hartigan, J. (1975). *Cluster algorithms*. John Wiley & Sons, Ltd, New York, USA. ISBN: 0-471-35645-X.
- Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

REFERENCES

- Hosny, M. I. (2010). *Investigating heuristic and meta-heuristic algorithms for solving pickup and delivery problems*. Phd thesis, Cardiff University School of Computer Science and Informatics, United Kingdom.
- Hribar, M. and Daskin, M. S. (1997). A dynamic programming heuristic for the P-median problem. *European Journal of Operational Research*, 101(3):499–508.
- Jain, A. K. and Dubes, R. C. (1998). *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey. ISBN: 0-13-022278-X.
- Joubert, J. W. (2006). *An Integrated and Intelligent Metaheuristic for constrained Vehicle Routing*. Phd thesis, University of Pretoria, South Africa.
- Kant, G., Jacks, M., and Aantjes, C. (2008). Coca-cola enterprises optimizes vehicle routes for efficient product delivery. *Interfaces*, 38(1):40–50.
- Klose, A. (1990). An LP-based heuristic for two-stage capacitated facility location problems. *Journal of the Operational Research Society*, 50(2):157–166.
- Klose, A. and Görtz, S. (2007). A branch-and-price algorithm for the capacitated facility location problem. *European Journal of Operational Research*, 179(3):1109–1125.
- Lam, M. (2008). *Three essays on clustering in location routing problems*. Phd thesis, The University of Alabama, USA.
- Lam, M. and Mittenthal, J. (2013). Capacitated hierarchical clustering heuristic for multi depot location-routing problems. *International Journal of Logistics Research and Applications*, 16(5):433–444.
- Lam, M., Mittenthal, J., and Gray, B. (2009). The impact of stopping rules on hierarchical capacitated clustering in location routing problems. *Academy of Information and Management Sciences*, 12(1):13–29.
- Lance, G. N. and Williams, W. T. (1967). A general theory of classificatory sorting strategies: 1. Hierarchical systems. *Computer Journal*, 9(3):373–380.
- Langevin, A. and Riopel, D., editors (2005). *Logistics Systems: Design and Optimization*. GERAD 25th Anniversary Series. Springer Science + Business Media, Inc., New York. ISBN: 0-387-24971-0.
- Laporte, G. (1992a). The Travelling Salesman Problem: An overview of exact and approximation algorithms. *European Journal Of Operational Research*, 59(2):231–247.
- Laporte, G. (1992b). The Vehicle Routing Problem: An overview of exact and approximation algorithms. *European Journal Of Operational Research*, 59(3):345–358.

REFERENCES

- Laporte, G., Gendreaub, M., Potvinb, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International transactions in Operational Research*, 7(4-5):285–300.
- Li, Q., Kecman, V., and Salman, R. (2010). A chunking method for euclidean distance matrix calculation on large dataset using multi-gpu. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 208–213. Ieee.
- Lopes, R. B. (2011). *Location-Routing Problems of Semi-Obnoxious Facilities: Approaches and Decision Support*. Phd thesis, University of Aveiro, Portugal.
- Lorena, L. A. N. and Senne, E. L. F. (2004). A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876.
- Louis, S. J. and Tang, R. (1999). Interactive Genetic Algorithms for the Travelling Salesman Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, pages 385–392. Orlando, Florida, USA, 13-17 July 1999, San Francisco: Morgan Kaufmann.
- Luis, M., Salhi, S., and G., N. (2009). Region-rejection based heuristics for the capacitated multi-source Weber problem. *Computers & Operations Research*, 36(6):2007–2017.
- Luis, M., Salhi, S., and G., N. (2011). A guided reactive GRASP for the capacitated multi-source Weber problem. *Computers & Operations Research*, 38(7):1014–1024.
- Manzour-al-Ajdad, S. M. H., Torabi, S. A., and Salhi, S. (2012). A hierarchical algorithm for the planar single-facility location routing problem. *Computers & Operations Research*, 39(2):461–470.
- Maranzana, F. (1964). On the location of supply points to minimize transport costs. *OR*, pages 261–270.
- Min, H. (1987). *The Vehicle Routing Problem with Product/Spatial Consolidation and Backhauling*. Phd thesis, Ohio State University, USA.
- Min, H. (1996). Consolidation terminal location–allocation and consolidated routing problems. *Journal of Business Logistics*, 17(2):235–263.
- Minoux, M. (1987). A class of combinatorial problems with polynomially solvable large scale set covering/partitioning relaxations. *Revue française d’automatique, d’informatique et de recherche opérationnelle. Recherche opérationnelle*, 21(2):105–136.

REFERENCES

- Mladenović, N., Brimberg, J., Hansen, P., and Moreno-Pérez, J. (2007). The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939.
- Müllner, D. (2013). fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. *Journal of Statistical Software*, 53(9):1–18.
- Mulvey, J. and Beck, M. (1984). Solving capacitated clustering problems. *European Journal Of Operational Research*, 18(3):339–348.
- Mulvey, J. and Crowder, H. (1979). Cluster analysis: an application of Lagrangian relaxation. *Management Science*, 25(4):392–340.
- Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359.
- Nagy, G. and Salhi, S. (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672.
- Neebe, A. and Rao, M. R. (1983). An Algorithm for the Fixed-Charge Assigning Users to Sources. *The Journal of the Operational Research Society*, 34(11):1107–1113.
- Negreiros, M. and Palhano, A. (2006). The capacitated centred clustering problem. *Computers & Operations Research*, 33(6):1639–1663.
- Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications Inc., New York, 2nd edition. ISBN: 0-486-40258-4.
- Pirkwieser, S. and Raidl, G. R. (2010). Variable neighborhood search coupled with ILP-based very large-neighborhood searches for the (periodic) location-routing problem. In *Proceedings of HM 2010 – Seventh International Workshop on Hybrid Metaheuristics*, volume 6373 of *Lecture Notes in Computer Science*, pages 174–189. Springer Verlag, Berlin, Germany.
- Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.
- Poot, A., Kant, G., and Wagelmans, A. (1999). A savings based method for real-life vehicle routing problems. Technical report, Erasmus University Rotterdam, Erasmus School of Economics (ESE), Econometric Institute.
- Prins, C. (2004). INTERNET. http://prodhonc.free.fr/Instances/instances_us.htm. Cited 19 July 2013.

REFERENCES

- Prins, C., Prodhon, C., Soriano, P., and Calvo, R. W. (2007). Solving the Capacitated Location-Routing Problem by a Cooperative Lagrangean Relaxation-Granular Tabu Search Heuristic. *Transportation Science*, 41(4):470–483.
- Prins, C., Prodhon, C., and Wolfler-Calvo, R. (2006). Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. *4OR*, 4:221–238.
- Prodhon, C. (2009a). INTERNET. http://prodhonc.free.fr/Instances/instancesPLRP_us.htm. Cited 19 July 2013.
- Prodhon, C. (2009b). An ELS x path relinking hybrid for the periodic location-routing problem. *Lecture Notes in Computer Science*, 5818:15–29. In M.J. Blesa et al. (Eds.), Hybrid Metaheuristics, HM 2009.
- Prodhon, C. (2011). A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal Of Operational Research*, 210(2):204–212.
- Prodhon, C. and Prins, C. (2008). A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. *Lecture Notes in Computer Science*, 5296:43–57. In: M.J. Blesa et al. (Eds.) HM 2008.
- Rand, G. K. (1976). Methodological Choices in Depot Location Studies. *Operational Research Quarterly*, 27(1 Part ii):241–249.
- Ravi, R. and Sinha, A. (2006). Approximation algorithms for problems combining facility location and network design. *Operations Research*, 54(1):73–81.
- Reinelt, G. (2015). Universität Heidelberg, Germany. INTERNET. <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/>. Cited 9 March 2015, Last updated 2015.
- Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, 9(1):1–17.
- Salhi, S. (1997). A perturbation heuristic for a class of location problems. *Journal of the Operational Research Society*, 48:1233–1240.
- Salhi, S. and Atkinson, R. A. (1995). Subdrop: A modified drop heuristic for location problems. *Location Science*, 3(4):267–273.
- Salhi, S. and Nagy, G. (2009). Local improvements in planar facility location using vehicle routing. *Annals of Operational Research*, 167(1):287–296.
- Salhi, S. and Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal Of Operational Research*, 39(2):150–156.

REFERENCES

- Schwardt, M. and Dethloff, J. (2005). Solving a continuous location-routing problem by use of a self-organizing map. *International Journal of Physical Distribution & Logistics Management*, 35(6):390–408.
- Schwardt, M. and Fischer, K. (2009). Combined location-routing problems—a neural network approach. *Annals of Operations Research*, 167(1):253–269.
- Simchi-Levi, D. and Bramel, J. (1997). *The Logic of Logistics : Theory, Algorithms, and Applications for Logistics Management*. Springer-Verlag, New York, 2nd edition. ISBN: 0-387-94921-6.
- Soroker, D. (1988). Fast parallel algorithms for finding hamiltonian paths and cycles in a tournament. *Journal of Algorithms*, 9(2):276–286.
- Tan, C. and Beasley, J. E. (1984). A heuristic algorithm for the period vehicle routing problem. *Omega*, 12(5):497–504.
- Toth, P. and Vigo, D. (2003). The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4):333–346.
- Tuzun, D. and Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European Journal Of Operational Research*, 116(1):87–99.
- Veness, C. (2015). Movable type scripts: Calculate distance, bearing and more between latitude/longitude points. INTERNET. <http://www.movable-type.co.uk/scripts/latlong.html> Cited 11 March 2015.
- Wang, Y. (2013). Research of Multi-Depot Vehicle Routing Problem with Cellular Ant Algorithm. *Journal of Computers*, 8(7):1722–1727.
- Ward, J. H. (1963). Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.
- Winston, W. L. (2004). *Operations Research: Applications and Algorithms*. Brooks / Cole - Thomson Learning. ISBN: 0-534-42362-0.
- Wong, M. A. (1982). A Hybrid Clustering Method for Identifying High-Density Clusters. *Journal of the American Statistical Association*, 77(380):841–847.
- Wong, M. A. and Lane, T. (1981). A kth Nearest Neighbour Clustering Procedure. Working paper; Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge Massachusetts 02139.
- Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing Gestalt clusters. *IEEE Transactions on Computers*, C 20:68–86.
- Zainuddin, Z. M. and Salhi, S. (2007). A perturbation-based heuristic for the capacitated multisource Weber problem. *European Journal of Operational Research*, 179(3):1194–1207.

- Zohrehbandian, M. (2007). A new formulation of the hamiltonian p-median problem. *Applied Mathematical Sciences*, 1(8):355–361.
- Zohrehbandian, M. and Hamidnia, N. S. (2010). Ant colony optimization techniques for the hamiltonian p-median problem. *Mathematical Sciences*, 4(4):383–390.

Appendix A

Clustering method results

In this Appendix the results obtained from the numerical tests performed in Chapter 4 are listed. Section A.1 gives the results of the various clustering methods and the results when using these as starting solutions for the partitioning and CLUSTER partitioning methods.

The results are divided into the four instances, called DataS1, DataS2, DataS3 and USA13509 with 1 000 points, 3 000 points, 7 681 and 13 509 points respectively. Each section gives the results of the five measures described in Section 4.1.2, as well as the CPU times. The CPU times include the time to run the starting solution. The times are recorded in the format "mm:ss" or "hh:mm:ss".

The objective is to minimise the different measures in order to find quality solutions. The minimum values for each measure is therefore highlighted in bold. In Tables A.1 – A.39, the second column gives the result of the specified clustering method. The other columns give the results where the solutions from the second column are used as starting solutions for the various partitioning methods.

In Section A.2, the results of the different variants of the hybrid density methods, discussed in Section 4.3.5.2 are given. Section A.3 gives the results of the various hybrid clustering methods tested in Section 4.3.6. These were tested using DataS3. The methods include the hybrid SAHN - SAHN clustering methods, hybrid SAHN - partitioning clustering methods and the hybrid partitioning methods. The results from the aggregated stepwise partitioning methods for DataS3, discussed in Section 4.3.6.2.1, are also provided in Table A.48.

A.1 Clustering method results

A.1.1 Clustering method results - DataS1

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	10 796.11	4 808.33	5 518.98	10 743.64	6 301.48	10 714.22	5 422.69	4 805.25	5 110.26	10 403.12	6 301.48	9 954.46	5 422.69
Complete linkage	6 533.47	4 805.25	4 806.01	6 507.44	6 301.48	6 433.62	4 794.52	4 805.25	5 464.28	6 533.47	6 301.48	6 533.47	4 794.52
Average linkage	6 389.59	5 739.97	5 741.00	6 360.53	6 660.52	6 369.57	5 725.28	4 805.25	5 464.28	6 360.53	6 301.48	6 369.57	4 794.52
Weighted average linkage	10 767.05	5 936.70	5 936.70	10 730.21	6 771.27	10 445.87	5 903.67	4 804.43	4 804.30	10 288.08	6 301.48	10 098.17	4 794.52
Centroid linkage	8 759.83	5 741.00	6 062.16	8 713.85	6 301.48	8 557.70	5 724.88	4 805.25	5 464.28	8 600.56	6 301.48	7 832.83	4 794.52
Weighted centroid linkage	7 744.16	4 808.33	4 805.25	7 720.07	6 771.27	7 724.99	4 794.52	4 805.25	5 110.27	7 700.43	6 301.48	7 702.45	5 422.69
Ward's method	5 200.16	4 807.38	4 807.38	5 186.81	6 410.00	5 118.16	4 891.95	4 805.25	4 804.30	5 200.16	5 200.16	5 200.16	4 794.52
Partitioning methods:													
Stepwise k-means	4 804.43	4 804.43	4 803.54	4 799.30	4 804.43	6 410.00	4 794.52	4 804.43	4 804.43	4 804.43	4 804.30	4 804.43	4 794.52
Stepwise h-means	4 804.30	4 804.30	4 803.45	4 801.89	4 804.30	6 410.00	4 794.52	4 804.30	4 804.30	4 804.30	4 804.30	4 804.30	4 794.52
Stepwise j-means	5 025.12	4 804.30	5 012.37	5 017.33	4 804.30	6 771.27	4 794.52	4 804.30	5 025.12	5 025.12	4 804.30	5 025.12	4 794.52
Stepwise PAM	5 772.77	5 123.84	5 772.77	5 755.15	5 123.84	6 771.27	4 889.97	4 804.30	5 772.77	5 772.77	4 804.30	5 772.77	4 794.52
Stepwise k-median	4 961.17	4 807.38	4 965.66	4 947.62	4 807.38	6 771.27	4 891.95	4 961.17	4 961.17	4 961.17	4 961.17	4 961.17	4 961.17
Stepwise h-median	4 794.52	4 804.30	4 794.04	4 794.52	4 804.30	6 771.27	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52
CLUSTER partitioning methods:													
CLUSTER k-means	5 470.28	5 470.28	5 470.28	5 472.03	6 481.34	5 450.04	5 422.69	5 470.28	5 464.28	5 470.28	5 470.28	5 470.28	4 794.52
CLUSTER h-means	5 112.56	5 112.56	5 112.56	5 120.55	6 481.34	5 105.98	5 147.67	5 112.56	5 112.56	5 112.56	5 112.56	5 112.56	5 112.56
CLUSTER j-means	10 681.04	10 654.80	5 741.00	10 654.80	6 771.27	10 654.80	5 724.88	10 654.80	5 464.28	10 654.80	6 301.48	10 654.80	4 794.52
CLUSTER PAM	6 255.77	4 807.38	5 110.27	6 255.77	6 301.48	6 171.33	4 891.95	4 804.30	4 804.30	6 255.77	6 255.77	6 138.32	4 794.52
CLUSTER k-median	10 351.54	5 112.56	5 112.56	5 120.55	6 481.34	5 105.98	5 147.67	5 112.56	5 112.56	5 112.56	5 112.56	5 112.56	5 112.56
CLUSTER h-median	4 794.52	4 804.30	4 804.30	4 794.04	6 771.27	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52

Table A.1: The results of the SSR measure for the various clustering methods using DataS1.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	4 808.29	4 804.30	4 804.30	4 804.47	6 481.34	4 802.62	4 794.52	4 804.30	4 804.30	4 808.29	4 808.29	4 808.29	4 794.52
k-means AS 136	4 804.30	4 804.30	4 804.30	4 803.45	6 410.00	4 801.89	4 794.52	4 804.30	4 804.30	4 804.30	4 804.30	4 804.30	4 794.52
h-means AS 136	4 804.30	4 804.30	4 804.30	4 803.45	6 410.00	4 801.89	4 794.52	4 804.30	4 804.30	4 804.30	4 804.30	4 804.30	4 794.52
j-means AS 136	4 804.47	4 804.30	4 804.30	4 800.65	6 481.34	4 799.59	4 794.52	4 804.30	4 804.30	4 804.47	4 804.47	4 804.47	4 794.52
PAM AS 136	6 481.34	5 123.84	5 112.33	6 481.34	6 301.48	6 388.84	4 891.95	5 111.62	5 112.33	6 452.71	6 301.48	6 396.93	4 794.52
k-median AS 136	4 802.62	4 804.30	4 804.30	4 800.03	6 481.34	4 799.59	4 794.52	4 802.62	4 802.62	4 802.62	4 802.62	4 802.62	4 794.52
h-median AS 136	4 794.52	4 804.30	4 804.30	4 794.04	6 771.27	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	4 804.30	4 804.30	4 804.30	4 803.45	6 410.00	4 801.89	4 794.52	4 804.30	4 804.30	4 804.30	4 804.30	4 804.30	4 794.52
CLUSTER h-means AS136	4 804.30	4 804.30	4 804.30	4 803.45	6 410.00	4 801.89	4 794.52	4 804.30	4 804.30	4 804.30	4 804.30	4 804.30	4 794.52
CLUSTER k-mediod AS 136	4 808.29	4 804.30	4 804.30	4 804.47	6 481.34	4 802.62	4 794.52	4 804.30	4 804.30	4 808.29	4 808.29	4 808.29	4 794.52
CLUSTER h-mediod AS 136	4 808.29	4 804.30	4 804.30	4 804.47	6 481.34	4 802.62	4 794.52	4 804.30	4 804.30	4 808.29	4 808.29	4 808.29	4 794.52
CLUSTER k-median AS 136	4 808.29	4 804.30	4 804.30	4 804.47	6 481.34	4 802.62	4 794.52	4 804.30	4 804.30	4 808.29	4 808.29	4 808.29	4 794.52
CLUSTER h-median AS 136	4 794.52	4 804.30	4 804.30	4 794.04	6 771.27	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52	4 794.52
Graph-based methods:													
Graph based MST	10 867.51	5 110.26	5 110.27	10 848.57	7 003.23	10 848.57	5 328.72	5 110.27	5 110.27	10 794.86	6 301.48	10 794.86	4 794.52
Graph based RNG	10 762.79	5 110.26	5 470.28	10 710.37	6 301.48	10 680.98	5 422.69	5 474.49	5 464.28	10 681.17	6 301.48	10 681.17	5 422.69
Graph based GG	10 719.28	6 058.65	6 058.65	10 677.33	6 365.73	10 584.18	6 032.21	5 741.00	5 464.28	10 623.31	6 301.48	10 391.01	4 794.52
Nearest neighbours methods:													
k-near method	9 974.74	5 110.27	5 110.27	9 954.74	7 003.23	9 944.72	5 147.67	4 805.25	4 804.30	9 881.35	6 301.48	9 720.79	4 794.52
Density-based methods:													
k_{th} nearest method	9 935.68	5 739.97	5 738.61	9 893.38	6 690.09	9 707.46	5 725.28	5 738.61	5 464.28	9 851.25	6 301.48	9 417.05	4 794.52

Table A.2: The results of the SSR measure for the various clustering methods using DataS1 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	10.84	24.83	27.07	14.69	25.89	14.35	25.89	24.71	24.75	14.84	25.89	19.35	25.89
Complete linkage	25.02	24.71	24.56	25.24	25.89	24.96	24.53	24.71	26.45	25.02	25.89	25.02	24.53
Average linkage	20.56	23.27	23.28	20.71	24.84	20.65	21.88	24.71	26.45	20.71	25.89	20.65	24.53
Weighted average linkage	14.42	23.28	23.28	15.52	26.05	21.49	21.39	24.55	24.70	15.01	25.89	15.71	24.53
Centroid linkage	17.23	23.28	23.08	20.24	25.89	20.02	21.87	24.71	26.45	19.27	25.89	21.41	24.53
Weighted centroid linkage	23.68	24.83	24.71	23.69	26.05	24.22	24.53	24.71	24.75	21.69	25.89	21.18	25.89
Ward's method	25.18	24.82	24.82	25.16	28.05	24.93	24.66	24.71	24.70	25.18	25.18	25.18	24.53
Partitioning methods:													
Stepwise k-means	24.55	24.55	24.54	24.50	24.55	28.05	24.53	24.55	24.55	24.55	24.70	24.55	24.53
Stepwise h-means	24.70	24.70	24.70	24.65	24.70	28.05	24.53	24.70	24.70	24.70	24.70	24.70	24.53
Stepwise j-means	22.54	24.70	23.01	22.62	24.70	26.05	24.53	24.70	22.54	22.54	24.70	22.54	24.53
Stepwise PAM	27.66	24.86	27.66	27.56	24.86	26.05	24.64	24.70	27.66	27.66	24.70	27.66	24.53
Stepwise k-median	25.37	24.82	25.39	25.34	24.82	26.05	24.66	25.37	25.37	25.37	25.37	25.37	25.37
Stepwise h-median	24.53	24.70	24.51	24.53	24.70	26.05	24.53	24.53	24.53	24.53	24.53	24.53	24.53
CLUSTER partitioning methods:													
CLUSTER k-means	26.38	26.38	26.38	26.49	25.58	25.49	25.89	26.38	26.45	26.38	26.38	26.38	24.53
CLUSTER h-means	24.27	24.27	24.27	24.69	25.58	24.30	24.94	24.27	24.27	24.27	24.27	24.27	24.27
CLUSTER j-means	18.30	18.44	23.28	18.44	26.05	18.44	21.87	18.44	26.45	18.44	25.89	18.44	24.53
CLUSTER PAM	27.72	24.82	24.75	27.72	25.89	27.02	24.66	24.70	24.70	27.72	27.72	26.93	24.53
CLUSTER k-median	16.12	24.27	24.27	24.69	25.58	24.30	24.94	24.27	24.27	24.27	24.27	24.27	24.27
CLUSTER h-median	24.53	24.70	24.70	24.51	26.05	24.53	24.53	24.53	24.53	24.53	24.53	24.53	24.53

Table A.3: The results of the MSR measure for the various clustering methods using DataS1.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	24.16	24.70	24.70	24.27	25.58	24.32	24.53	24.70	24.70	24.16	24.16	24.16	24.53
k-means AS 136	24.70	24.70	24.70	24.70	28.05	24.65	24.53	24.70	24.70	24.70	24.70	24.70	24.53
h-means AS 136	24.70	24.70	24.70	24.70	28.05	24.65	24.53	24.70	24.70	24.70	24.70	24.70	24.53
j-means AS 136	24.27	24.70	24.70	24.37	25.58	24.43	24.53	24.70	24.70	24.27	24.27	24.27	24.53
PAM AS 136	25.58	24.86	24.32	25.58	25.89	25.48	24.66	24.30	24.32	30.40	25.89	24.58	24.53
k-median AS 136	24.32	24.70	24.70	24.36	25.58	24.43	24.53	24.32	24.32	24.32	24.32	24.32	24.53
h-median AS 136	24.53	24.70	24.70	24.51	26.05	24.53	24.53	24.53	24.53	24.53	24.53	24.53	24.53
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	24.70	24.70	24.70	24.70	28.05	24.65	24.53	24.70	24.70	24.70	24.70	24.70	24.53
CLUSTER h-means AS136	24.70	24.70	24.70	24.70	28.05	24.65	24.53	24.70	24.70	24.70	24.70	24.70	24.53
CLUSTER k-mediod AS 136	24.16	24.70	24.70	24.27	25.58	24.32	24.53	24.70	24.70	24.16	24.16	24.16	24.53
CLUSTER h-mediod AS 136	24.16	24.70	24.70	24.27	25.58	24.32	24.53	24.70	24.70	24.16	24.16	24.16	24.53
CLUSTER k-median AS 136	24.16	24.70	24.70	24.27	25.58	24.32	24.53	24.70	24.70	24.16	24.16	24.16	24.53
CLUSTER h-median AS 136	24.53	24.70	24.70	24.51	26.05	24.53	24.53	24.53	24.53	24.53	24.53	24.53	24.53
Graph-based methods:													
Graph based MST	10.91	24.75	24.75	10.91	28.66	10.91	26.47	24.75	24.75	10.87	25.89	10.87	24.53
Graph based RNG	12.72	24.75	26.38	16.57	25.89	16.23	25.89	26.54	26.45	17.76	25.89	17.76	25.89
Graph based GG	10.95	22.86	22.86	11.36	26.21	12.12	21.65	23.28	26.45	21.09	25.89	20.64	24.53
Nearest neighbours methods:													
k-near method	12.36	24.75	24.75	12.51	28.66	12.51	24.94	24.71	24.70	13.41	25.89	14.61	24.53
Density-based methods:													
k_{th} nearest method	12.04	23.27	23.47	17.40	27.98	16.51	21.88	23.47	26.45	17.93	25.89	17.35	24.53

Table A.4: The results of the MSR measure for the various clustering methods using DataS1 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	31.96	18.14	18.39	31.65	26.04	31.55	17.27	18.14	18.65	23.56	26.04	23.32	17.27
Complete linkage	17.35	18.14	18.31	17.35	26.04	17.32	18.42	18.14	18.17	17.35	26.04	17.35	18.42
Average linkage	19.27	18.56	18.53	19.19	24.28	19.22	18.73	18.14	18.17	19.19	26.04	19.22	18.42
Weighted average linkage	31.79	23.62	23.62	31.76	28.26	31.56	25.60	18.31	18.17	23.50	26.04	23.40	18.42
Centroid linkage	18.74	18.53	22.40	18.68	26.04	18.62	18.76	18.14	18.17	18.90	26.04	19.11	18.42
Weighted centroid linkage	19.76	18.14	18.14	19.67	28.26	19.70	18.42	18.14	18.65	19.63	26.04	19.64	17.27
Ward's method	24.37	18.17	18.17	24.33	28.26	24.33	23.95	18.14	18.17	24.37	24.37	24.37	18.42
Partitioning methods:													
Stepwise k-means	18.31	18.31	18.34	18.31	18.31	28.26	18.42	18.31	18.31	18.31	18.17	18.31	18.42
Stepwise h-means	18.17	18.17	18.20	18.17	18.17	28.26	18.42	18.17	18.17	18.17	18.17	18.17	18.42
Stepwise j-means	20.16	18.17	20.14	20.14	18.17	28.26	18.42	18.17	20.16	20.16	18.17	20.16	18.42
Stepwise PAM	25.09	27.22	25.09	25.09	27.22	28.26	23.95	18.17	25.09	25.09	18.17	25.09	18.42
Stepwise k-median	21.32	18.17	21.28	21.24	18.17	28.26	23.95	21.32	21.32	21.32	21.32	21.32	21.32
Stepwise h-median	18.42	18.17	18.51	18.42	18.17	28.26	18.42	18.42	18.42	18.42	18.42	18.42	18.42
CLUSTER partitioning methods:													
CLUSTER k-means	18.17	18.17	18.17	18.23	30.92	17.43	17.27	18.17	18.17	18.17	18.17	18.17	18.42
CLUSTER h-means	18.54	18.54	18.54	18.54	30.92	18.53	18.75	18.54	18.54	18.54	18.54	18.54	18.54
CLUSTER j-means	31.62	31.65	18.53	31.65	28.26	31.65	18.76	31.65	18.17	31.65	26.04	31.65	18.42
CLUSTER PAM	23.19	18.17	18.65	23.19	26.04	23.19	23.95	18.17	18.17	23.19	23.19	23.19	18.42
CLUSTER k-median	31.77	18.54	18.54	18.54	30.92	18.53	18.75	18.54	18.54	18.54	18.54	18.54	18.54
CLUSTER h-median	18.42	18.17	18.17	18.51	28.26	18.42	18.42	18.42	18.42	18.42	18.42	18.42	18.42

Table A.5: The results of the PD measure for the various clustering methods using DataS1.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	18.22	18.17	18.17	18.22	30.92	18.22	18.42	18.17	18.17	18.22	18.22	18.22	18.42
k-means AS 136	18.17	18.17	18.17	18.20	28.26	18.17	18.42	18.17	18.17	18.17	18.17	18.17	18.42
h-means AS 136	18.17	18.17	18.17	18.20	28.26	18.17	18.42	18.17	18.17	18.17	18.17	18.17	18.42
j-means AS 136	18.22	18.17	18.17	18.22	30.92	18.22	18.42	18.17	18.17	18.22	18.22	18.22	18.42
PAM AS 136	30.92	27.22	18.65	30.92	26.04	30.61	23.95	18.65	18.65	28.90	26.04	26.46	18.42
k-median AS 136	18.22	18.17	18.17	18.31	30.92	18.22	18.42	18.22	18.22	18.22	18.22	18.22	18.42
h-median AS 136	18.42	18.17	18.17	18.51	28.26	18.42	18.42	18.42	18.42	18.42	18.42	18.42	18.42
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	18.17	18.17	18.17	18.20	28.26	18.17	18.42	18.17	18.17	18.17	18.17	18.17	18.42
CLUSTER h-means AS136	18.17	18.17	18.17	18.20	28.26	18.17	18.42	18.17	18.17	18.17	18.17	18.17	18.42
CLUSTER k-mediod AS 136	18.22	18.17	18.17	18.22	30.92	18.22	18.42	18.17	18.17	18.22	18.22	18.22	18.42
CLUSTER h-mediod AS 136	18.22	18.17	18.17	18.22	30.92	18.22	18.42	18.17	18.17	18.22	18.22	18.22	18.42
CLUSTER k-median AS 136	18.22	18.17	18.17	18.22	30.92	18.22	18.42	18.17	18.17	18.22	18.22	18.22	18.42
CLUSTER h-median AS 136	18.42	18.17	18.17	18.51	28.26	18.42	18.42	18.42	18.42	18.42	18.42	18.42	18.42
Graph-based methods:													
Graph based MST	32.70	18.65	18.65	32.70	23.22	32.70	20.35	18.65	18.65	31.95	26.04	31.95	18.42
Graph based RNG	31.90	18.65	18.17	31.58	26.04	31.48	17.27	18.54	18.17	31.99	26.04	31.99	17.27
Graph based GG	31.88	22.42	22.42	31.86	27.29	31.82	23.18	18.53	18.17	31.54	26.04	31.74	18.42
Nearest neighbours methods:													
k-near method	31.47	18.65	18.65	31.49	23.22	31.49	18.75	18.14	18.17	31.55	26.04	31.71	18.42
Density-based methods:													
k _{th} nearest method	31.45	18.56	18.53	31.51	24.78	26.74	18.73	18.53	18.17	31.16	26.04	26.79	18.42

Table A.6: The results of the PD measure for the various clustering methods using DataS1 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	7 458.31	778.17	1 013.84	7 393.16	1 164.30	7 360.72	1 111.40	781.54	820.21	6 978.72	1 164.30	6 420.85	1 111.40
Complete linkage	1 902.24	781.54	781.17	1 870.43	1 164.30	1 818.48	768.48	781.54	1 104.88	1 902.24	1 164.30	1 902.24	768.48
Average linkage	1 904.37	1 196.69	1 198.22	1 881.95	1 593.76	1 889.44	1 310.34	781.54	1 104.88	1 881.95	1 164.30	1 889.44	768.48
Weighted average linkage	7 401.30	1 154.33	1 154.33	7 359.72	1 465.48	7 002.95	1 499.15	782.53	779.97	6 821.13	1 164.30	6 592.23	768.48
Centroid linkage	4 794.97	1 198.22	1 201.43	4 717.00	1 164.30	4 529.88	1 309.16	781.54	1 104.88	4 695.23	1 164.30	3 406.69	768.48
Weighted centroid linkage	2 563.21	778.17	781.54	2 545.51	1 465.48	2 550.30	768.48	781.54	819.23	2 539.41	1 164.30	2 534.98	1 111.40
Ward's method	769.37	776.59	776.59	764.59	1 451.52	744.39	704.90	781.54	779.97	769.37	769.37	769.37	768.48
Partitioning methods:													
Stepwise k-means	782.53	782.53	781.02	786.08	782.53	1 451.52	768.48	782.53	782.53	782.53	779.97	782.53	768.48
Stepwise h-means	779.97	779.97	778.44	781.71	779.97	1 451.52	768.48	779.97	779.97	779.97	779.97	779.97	768.48
Stepwise j-means	905.22	779.97	885.37	898.17	779.97	1 465.48	768.48	779.97	905.22	905.22	779.97	905.22	768.48
Stepwise PAM	905.91	789.66	905.91	900.06	789.66	1 465.48	703.25	779.97	905.91	905.91	779.97	905.91	768.48
Stepwise k-median	714.69	776.59	714.00	716.54	776.59	1 465.48	704.90	714.69	714.69	714.69	714.69	714.69	714.69
Stepwise h-median	768.48	779.97	765.39	768.48	779.97	1 465.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48
CLUSTER partitioning methods:													
CLUSTER k-means	1 115.76	1 115.76	1 115.76	1 110.91	1 483.03	1 181.73	1 111.40	1 115.76	1 104.88	1 115.76	1 115.76	1 115.76	768.48
CLUSTER h-means	844.17	844.17	844.17	837.46	1 483.03	835.26	813.66	844.17	844.17	844.17	844.17	844.17	844.17
CLUSTER j-means	7 304.72	7 275.11	1 198.22	7 275.11	1 465.48	7 275.11	1 309.16	7 275.11	1 104.88	7 275.11	1 164.30	7 275.11	768.48
CLUSTER PAM	1 178.71	776.59	819.23	1 178.71	1 164.30	1 133.93	704.90	779.97	779.97	1 178.71	1 178.71	1 173.75	768.48
CLUSTER k-median	6 910.63	844.17	844.17	837.46	1 483.03	835.26	813.66	844.17	844.17	844.17	844.17	844.17	844.17
CLUSTER h-median	768.48	779.97	779.97	765.39	1 465.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48

Table A.7: The results of the ADM measure for the various clustering methods using DataS1.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	781.89	779.97	779.97	777.30	1 483.03	775.05	768.48	779.97	779.97	781.89	781.89	781.89	768.48
k-means AS 136	779.97	779.97	779.97	778.44	1 451.52	781.71	768.48	779.97	779.97	779.97	779.97	779.97	768.48
h-means AS 136	779.97	779.97	779.97	778.44	1 451.52	781.71	768.48	779.97	779.97	779.97	779.97	779.97	768.48
j-means AS 136	777.30	779.97	779.97	772.82	1 483.03	770.75	768.48	779.97	779.97	777.30	777.30	777.30	768.48
PAM AS 136	1 483.03	789.66	837.38	1 483.03	1 164.30	1 430.22	704.90	838.20	837.38	1 455.37	1 164.30	1 407.04	768.48
k-median AS 136	775.05	779.97	779.97	769.16	1 483.03	770.75	768.48	775.05	775.05	775.05	775.05	775.05	768.48
h-median AS 136	768.48	779.97	779.97	765.39	1 465.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	779.97	779.97	779.97	778.44	1 451.52	781.71	768.48	779.97	779.97	779.97	779.97	779.97	768.48
CLUSTER h-means AS136	779.97	779.97	779.97	778.44	1 451.52	781.71	768.48	779.97	779.97	779.97	779.97	779.97	768.48
CLUSTER k-mediod AS 136	781.89	779.97	779.97	777.30	1 483.03	775.05	768.48	779.97	779.97	781.89	781.89	781.89	768.48
CLUSTER h-mediod AS 136	781.89	779.97	779.97	777.30	1 483.03	775.05	768.48	779.97	779.97	781.89	781.89	781.89	768.48
CLUSTER k-median AS 136	781.89	779.97	779.97	777.30	1 483.03	775.05	768.48	779.97	779.97	781.89	781.89	781.89	768.48
CLUSTER h-median AS 136	768.48	779.97	779.97	765.39	1 465.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48	768.48
Graph-based methods:													
Graph based MST	7 519.05	820.21	819.23	7 491.52	1 778.36	7 491.52	791.36	819.23	819.23	7 439.28	1 164.30	7 439.28	768.48
Graph based RNG	7 416.11	820.21	1 115.76	7 351.20	1 164.30	7 318.89	1 111.40	1 099.45	1 104.88	7 321.97	1 164.30	7 321.97	1 111.40
Graph based GG	7 372.35	1 210.83	1 210.83	7 323.69	1 184.35	7 208.35	1 297.41	1 198.22	1 104.88	7 243.47	1 164.30	6 966.16	768.48
Nearest neighbours methods:													
k-near method	6 385.54	819.23	819.23	6 350.73	1 778.36	6 337.81	813.66	781.54	779.97	6 295.14	1 164.30	6 127.27	768.48
Density-based methods:													
k_{th} nearest method	6 370.49	1 196.69	1 140.74	6 314.82	1 501.17	6 112.50	1 310.34	1 140.74	1 104.88	6 253.32	1 164.30	5 802.66	768.48

Table A.8: The results of the ADM measure for the various clustering methods using DataS1 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	7 488.27	3 311.06	3 720.95	7 443.84	4 243.73	7 420.92	3 705.62	3 311.53	3 462.11	7 179.97	4 243.73	6 852.31	3 705.62
Complete linkage	4 370.92	3 311.53	3 312.60	4 356.43	4 243.73	4 311.71	3 309.16	3 311.53	3 705.42	4 370.92	4 243.73	4 370.92	3 309.16
Average linkage	4 386.69	3 890.86	3 891.74	4 369.02	4 469.75	4 374.01	3 917.73	3 311.53	3 705.42	4 369.02	4 243.73	4 374.01	3 309.16
Weighted average linkage	7 471.67	3 994.18	3 994.18	7 446.43	4 729.30	7 254.23	4 056.11	3 312.27	3 310.58	7 088.90	4 243.73	6 950.60	3 309.16
Centroid linkage	5 925.39	3 891.74	4 065.54	5 893.95	4 243.73	5 786.26	3 917.32	3 311.53	3 705.42	5 827.04	4 243.73	5 310.99	3 309.16
Weighted centroid linkage	5 368.62	3 311.06	3 311.53	5 354.37	4 729.30	5 355.40	3 309.16	3 311.53	3 461.69	5 352.29	4 243.73	5 353.17	3 705.62
Ward's method	3 547.15	3 310.11	3 310.11	3 537.54	4 441.50	3 488.89	3 378.61	3 311.53	3 310.58	3 547.15	3 547.15	3 547.15	3 309.16
Partitioning methods:													
Stepwise k-means	3 312.27	3 312.27	3 311.39	3 311.93	3 312.27	4 441.50	3 309.16	3 312.27	3 312.27	3 312.27	3 310.58	3 312.27	3 309.16
Stepwise h-means	3 310.58	3 310.58	3 309.71	3 310.87	3 310.58	4 441.50	3 309.16	3 310.58	3 310.58	3 310.58	3 310.58	3 310.58	3 309.16
Stepwise j-means	3 509.09	3 310.58	3 510.19	3 504.95	3 310.58	4 729.30	3 309.16	3 310.58	3 509.09	3 509.09	3 310.58	3 509.09	3 309.16
Stepwise PAM	3 890.66	3 509.55	3 890.66	3 881.56	3 509.55	4 729.30	3 378.34	3 310.58	3 890.66	3 890.66	3 310.58	3 890.66	3 309.16
Stepwise k-median	3 435.28	3 310.11	3 437.14	3 425.66	3 310.11	4 729.30	3 378.61	3 435.28	3 435.28	3 435.28	3 435.28	3 435.28	3 435.28
Stepwise h-median	3 309.16	3 310.58	3 307.88	3 309.16	3 310.58	4 729.30	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16
CLUSTER partitioning methods:													
CLUSTER k-means	3 704.95	3 704.95	3 704.95	3 707.08	4 506.78	3 721.97	3 705.62	3 704.95	3 705.42	3 704.95	3 704.95	3 704.95	3 309.16
CLUSTER h-means	3 470.78	3 470.78	3 470.78	3 472.62	4 506.78	3 466.67	3 495.91	3 470.78	3 470.78	3 470.78	3 470.78	3 470.78	3 470.78
CLUSTER j-means	7 389.03	7 368.37	3 891.74	7 368.37	4 729.30	7 368.37	3 917.32	7 368.37	3 705.42	7 368.37	4 243.73	7 368.37	3 309.16
CLUSTER PAM	4 224.19	3 310.11	3 461.69	4 224.19	4 243.73	4 167.69	3 378.61	3 310.58	3 310.58	4 224.19	4 224.19	4 135.28	3 309.16
CLUSTER k-median	7 134.29	3 470.78	3 470.78	3 472.62	4 506.78	3 466.67	3 495.91	3 470.78	3 470.78	3 470.78	3 470.78	3 470.78	3 470.78
CLUSTER h-median	3 309.16	3 310.58	3 310.58	3 307.88	4 729.30	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16

Table A.9: The results of the AWCD measure for the various clustering methods using DataS1.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	3 328.16	3 310.58	3 310.58	3 323.37	4 506.78	3 320.92	3 309.16	3 310.58	3 310.58	3 328.16	3 328.16	3 328.16	3 309.16
k-means AS 136	3 310.58	3 310.58	3 310.58	3 309.71	4 441.50	3 310.87	3 309.16	3 310.58	3 310.58	3 310.58	3 310.58	3 310.58	3 309.16
h-means AS 136	3 310.58	3 310.58	3 310.58	3 309.71	4 441.50	3 310.87	3 309.16	3 310.58	3 310.58	3 310.58	3 310.58	3 310.58	3 309.16
j-means AS 136	3 323.37	3 310.58	3 310.58	3 318.38	4 506.78	3 316.53	3 309.16	3 310.58	3 310.58	3 323.37	3 323.37	3 323.37	3 309.16
PAM AS 136	4 506.78	3 509.55	3 470.68	4 506.78	4 243.73	4 417.73	3 378.61	3 469.68	3 470.68	4 481.41	4 243.73	4 419.11	3 309.16
k-median AS 136	3 320.92	3 310.58	3 310.58	3 317.21	4 506.78	3 316.53	3 309.16	3 320.92	3 320.92	3 320.92	3 320.92	3 320.92	3 309.16
h-median AS 136	3 309.16	3 310.58	3 310.58	3 307.88	4 729.30	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	3 310.58	3 310.58	3 310.58	3 309.71	4 441.50	3 310.87	3 309.16	3 310.58	3 310.58	3 310.58	3 310.58	3 310.58	3 309.16
CLUSTER h-means AS136	3 310.58	3 310.58	3 310.58	3 309.71	4 441.50	3 310.87	3 309.16	3 310.58	3 310.58	3 310.58	3 310.58	3 310.58	3 309.16
CLUSTER k-mediod AS 136	3 328.16	3 310.58	3 310.58	3 323.37	4 506.78	3 320.92	3 309.16	3 310.58	3 310.58	3 328.16	3 328.16	3 328.16	3 309.16
CLUSTER h-mediod AS 136	3 328.16	3 310.58	3 310.58	3 323.37	4 506.78	3 320.92	3 309.16	3 310.58	3 310.58	3 328.16	3 328.16	3 328.16	3 309.16
CLUSTER k-median AS 136	3 328.16	3 310.58	3 310.58	3 323.37	4 506.78	3 320.92	3 309.16	3 310.58	3 310.58	3 328.16	3 328.16	3 328.16	3 309.16
CLUSTER h-median AS 136	3 309.16	3 310.58	3 310.58	3 307.88	4 729.30	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16	3 309.16
Graph-based methods:													
Graph based MST	7 549.25	3 462.11	3 461.69	7 536.74	4 692.87	7 536.74	3 577.51	3 461.69	3 461.69	7 491.73	4 243.73	7 491.73	3 309.16
Graph based RNG	7 464.27	3 462.11	3 704.95	7 419.94	4 243.73	7 397.07	3 705.62	3 701.59	3 705.42	7 397.94	4 243.73	7 397.94	3 705.62
Graph based GG	7 439.82	4 066.01	4 066.01	7 406.32	4 417.93	7 332.52	4 077.84	3 891.74	3 705.42	7 357.77	4 243.73	7 174.45	3 309.16
Nearest neighbours methods:													
k-near method	6 840.53	3 461.69	3 461.69	6 827.39	4 692.87	6 820.82	3 495.91	3 311.53	3 310.58	6 767.90	4 243.73	6 643.27	3 309.16
Density-based methods:													
k_{th} nearest method	6 803.69	3 890.86	3 882.33	6 768.39	4 507.73	6 623.37	3 917.73	3 882.33	3 705.42	6 735.01	4 243.73	6 412.02	3 309.16

Table A.10: The results of the AWCD measure for the various clustering methods using DataS1 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	00 : 00 : 01	00 : 00 : 02	00 : 00 : 01	00 : 00 : 02	00 : 00 : 05	00 : 00 : 01	00 : 00 : 01	00 : 00 : 07	00 : 00 : 02	00 : 00 : 09	00 : 00 : 24	00 : 00 : 03	00 : 00 : 03
Complete linkage	00 : 00 : 01	00 : 00 : 01	00 : 00 : 01	00 : 00 : 02	00 : 00 : 04	00 : 00 : 02	00 : 00 : 01	00 : 00 : 08	00 : 00 : 02	00 : 00 : 11	00 : 00 : 26	00 : 00 : 02	00 : 00 : 02
Average linkage	00 : 00 : 00	00 : 00 : 00	00 : 00 : 00	00 : 00 : 01	00 : 00 : 03	00 : 00 : 00	00 : 00 : 00	00 : 00 : 08	00 : 00 : 01	00 : 00 : 08	00 : 00 : 25	00 : 00 : 02	00 : 00 : 01
Weighted average linkage	00 : 00 : 01	00 : 00 : 02	00 : 00 : 01	00 : 00 : 01	00 : 00 : 05	00 : 00 : 01	00 : 00 : 01	00 : 00 : 08	00 : 00 : 02	00 : 00 : 13	00 : 00 : 25	00 : 00 : 03	00 : 00 : 02
Centroid linkage	00 : 00 : 09	00 : 00 : 10	00 : 00 : 09	00 : 00 : 11	00 : 00 : 13	00 : 00 : 09	00 : 00 : 09	00 : 00 : 17	00 : 00 : 10	00 : 00 : 20	00 : 00 : 33	00 : 00 : 12	00 : 00 : 10
Weighted centroid linkage	00 : 00 : 01	00 : 00 : 02	00 : 00 : 01	00 : 00 : 03	00 : 00 : 04	00 : 00 : 01	00 : 00 : 01	00 : 00 : 08	00 : 00 : 02	00 : 00 : 09	00 : 00 : 25	00 : 00 : 03	00 : 00 : 02
Ward's method	00 : 00 : 01	00 : 00 : 02	00 : 00 : 01	00 : 00 : 02	00 : 00 : 03	00 : 00 : 01	00 : 00 : 01	00 : 00 : 07	00 : 00 : 02	00 : 00 : 07	00 : 00 : 25	00 : 00 : 02	00 : 00 : 03
Partitioning methods:													
Stepwise k-means	00 : 02 : 01	00 : 02 : 01	00 : 02 : 01	00 : 02 : 01	00 : 02 : 01	00 : 02 : 03	00 : 02 : 01	00 : 02 : 02	00 : 02 : 05	00 : 02 : 02	00 : 02 : 01	00 : 02 : 24	00 : 02 : 02
Stepwise h-means	00 : 00 : 04	00 : 00 : 04	00 : 00 : 05	00 : 00 : 04	00 : 00 : 04	00 : 00 : 06	00 : 00 : 04	00 : 00 : 06	00 : 00 : 07	00 : 00 : 05	00 : 00 : 05	00 : 00 : 27	00 : 00 : 06
Stepwise j-means	00 : 03 : 41	00 : 03 : 41	00 : 03 : 43	00 : 03 : 41	00 : 03 : 41	00 : 03 : 43	00 : 03 : 41	00 : 03 : 42	00 : 03 : 46	00 : 03 : 42	00 : 03 : 41	00 : 04 : 02	00 : 03 : 42
Stepwise PAM	00 : 06 : 54	00 : 06 : 54	00 : 06 : 55	00 : 06 : 55	00 : 06 : 54	00 : 06 : 55	00 : 06 : 54	00 : 06 : 56	00 : 06 : 59	00 : 06 : 55	00 : 06 : 55	00 : 07 : 15	00 : 06 : 55
Stepwise k-median	00 : 00 : 09	00 : 00 : 09	00 : 00 : 10	00 : 00 : 09	00 : 00 : 09	00 : 00 : 11	00 : 00 : 09	00 : 00 : 10	00 : 00 : 13	00 : 00 : 10	00 : 00 : 10	00 : 00 : 30	00 : 00 : 10
Stepwise h-median	00 : 00 : 12	00 : 00 : 12	00 : 00 : 13	00 : 00 : 12	00 : 00 : 12	00 : 00 : 14	00 : 00 : 12	00 : 00 : 13	00 : 00 : 16	00 : 00 : 13	00 : 00 : 13	00 : 00 : 34	00 : 00 : 13
CLUSTER partitioning methods:													
CLUSTER k-means	00 : 00 : 05	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 07	00 : 00 : 05	00 : 00 : 05	00 : 00 : 07	00 : 00 : 05	00 : 00 : 11	00 : 00 : 29	00 : 00 : 07	00 : 00 : 06
CLUSTER h-means	00 : 00 : 02	00 : 00 : 02	00 : 00 : 02	00 : 00 : 04	00 : 00 : 04	00 : 00 : 02	00 : 00 : 02	00 : 00 : 03	00 : 00 : 03	00 : 00 : 07	00 : 00 : 23	00 : 00 : 04	00 : 00 : 03
CLUSTER j-means	00 : 00 : 11	00 : 00 : 11	00 : 00 : 11	00 : 00 : 11	00 : 00 : 14	00 : 00 : 11	00 : 00 : 11	00 : 00 : 12	00 : 00 : 12	00 : 00 : 15	00 : 00 : 34	00 : 00 : 13	00 : 00 : 13
CLUSTER PAM	00 : 00 : 48	00 : 00 : 49	00 : 00 : 48	00 : 00 : 49	00 : 00 : 51	00 : 00 : 48	00 : 00 : 49	00 : 00 : 50	00 : 00 : 49	00 : 00 : 53	00 : 01 : 09	00 : 00 : 49	00 : 00 : 49
CLUSTER k-median	00 : 00 : 04	00 : 00 : 05	00 : 00 : 04	00 : 00 : 06	00 : 00 : 06	00 : 00 : 04	00 : 00 : 04	00 : 00 : 06	00 : 00 : 05	00 : 00 : 09	00 : 00 : 26	00 : 00 : 05	00 : 00 : 05
CLUSTER h-median	00 : 00 : 03	00 : 00 : 03	00 : 00 : 03	00 : 00 : 04	00 : 00 : 05	00 : 00 : 03	00 : 00 : 03	00 : 00 : 04	00 : 00 : 04	00 : 00 : 07	00 : 00 : 26	00 : 00 : 04	00 : 00 : 04

Table A.11: The CPU times for the various clustering methods using DataS1.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	00 : 00 : 05	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 07	00 : 00 : 05	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 10	00 : 00 : 28	00 : 00 : 06	00 : 00 : 06
k-means AS 136	00 : 00 : 05	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 07	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 05	00 : 00 : 09	00 : 00 : 29	00 : 00 : 06	00 : 00 : 06
h-means AS 136	00 : 00 : 05	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 07	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 05	00 : 00 : 09	00 : 00 : 29	00 : 00 : 06	00 : 00 : 06
j-means AS 136	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 08	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 07	00 : 00 : 11	00 : 00 : 28	00 : 00 : 07	00 : 00 : 07
PAM AS 136	00 : 00 : 07	00 : 00 : 08	00 : 00 : 07	00 : 00 : 07	00 : 00 : 09	00 : 00 : 07	00 : 00 : 07	00 : 00 : 08	00 : 00 : 07	00 : 00 : 13	00 : 00 : 29	00 : 00 : 08	00 : 00 : 09
k-median AS 136	00 : 00 : 05	00 : 00 : 05	00 : 00 : 05	00 : 00 : 06	00 : 00 : 08	00 : 00 : 05	00 : 00 : 05	00 : 00 : 07	00 : 00 : 06	00 : 00 : 11	00 : 00 : 27	00 : 00 : 07	00 : 00 : 06
h-median AS 136	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 08	00 : 00 : 08	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 06	00 : 00 : 11	00 : 00 : 31	00 : 00 : 08	00 : 00 : 07
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 08	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 06	00 : 00 : 10	00 : 00 : 29	00 : 00 : 07	00 : 00 : 07
CLUSTER h-means AS136	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 08	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 06	00 : 00 : 10	00 : 00 : 29	00 : 00 : 07	00 : 00 : 07
CLUSTER k-mediod AS 136	00 : 00 : 10	00 : 00 : 10	00 : 00 : 10	00 : 00 : 11	00 : 00 : 12	00 : 00 : 11	00 : 00 : 10	00 : 00 : 11	00 : 00 : 11	00 : 00 : 15	00 : 00 : 32	00 : 00 : 11	00 : 00 : 11
CLUSTER h-mediod AS 136	00 : 00 : 28	00 : 00 : 28	00 : 00 : 29	00 : 00 : 29	00 : 00 : 31	00 : 00 : 28	00 : 00 : 29	00 : 00 : 30	00 : 00 : 29	00 : 00 : 32	00 : 00 : 51	00 : 00 : 30	00 : 00 : 29
CLUSTER k-median AS 136	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 08	00 : 00 : 06	00 : 00 : 06	00 : 00 : 08	00 : 00 : 07	00 : 00 : 11	00 : 00 : 29	00 : 00 : 07	00 : 00 : 07
CLUSTER h-median AS 136	00 : 00 : 06	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 07	00 : 00 : 06	00 : 00 : 06	00 : 00 : 07	00 : 00 : 07	00 : 00 : 11	00 : 00 : 30	00 : 00 : 07	00 : 00 : 07
Graph-based methods:													
Graph based MST	00 : 00 : 03	00 : 00 : 04	00 : 00 : 03	00 : 00 : 04	00 : 00 : 06	00 : 00 : 03	00 : 00 : 03	00 : 00 : 06	00 : 00 : 04	00 : 00 : 07	00 : 00 : 23	00 : 00 : 05	00 : 00 : 04
Graph based RNG	02 : 46 : 35	02 : 46 : 36	02 : 46 : 35	02 : 46 : 35	02 : 46 : 38	02 : 46 : 35	02 : 46 : 35	02 : 46 : 37	02 : 46 : 36	02 : 46 : 39	02 : 46 : 54	02 : 46 : 37	02 : 46 : 36
Graph based GG	00 : 00 : 15	00 : 00 : 16	00 : 00 : 16	00 : 00 : 16	00 : 00 : 18	00 : 00 : 15	00 : 00 : 15	00 : 00 : 17	00 : 00 : 16	00 : 00 : 19	00 : 00 : 36	00 : 00 : 17	00 : 00 : 17
Nearest neighbours methods:													
k-near method	00 : 00 : 00	00 : 00 : 02	00 : 00 : 01	00 : 00 : 01	00 : 00 : 03	00 : 00 : 00	00 : 00 : 00	00 : 00 : 03	00 : 00 : 01	00 : 00 : 04	00 : 00 : 20	00 : 00 : 02	00 : 00 : 01
Density-based methods:													
k_{th} nearest method	00 : 00 : 01	00 : 00 : 02	00 : 00 : 01	00 : 00 : 02	00 : 00 : 05	00 : 00 : 01	00 : 00 : 01	00 : 00 : 03	00 : 00 : 02	00 : 00 : 05	00 : 00 : 22	00 : 00 : 02	00 : 00 : 02

Table A.12: The CPU times for the various clustering methods using DataS1 (continue).

A.1.2 Clustering method results - DataS2

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	37 581.37	23 820.75	37 511.77	37 470.74	23 816.34	22 814.18	23 935.10	23 819.30	37 296.32	37 118.13	22 014.57	21 953.99	21 548.69
Complete linkage	30 382.62	22 014.57	30 344.05	30 308.67	21 958.95	21 953.99	21 548.69	23 819.30	30 344.05	29 898.44	22 014.57	21 953.99	21 548.26
Average linkage	36 310.11	26 451.67	36 159.52	36 233.18	21 595.40	22 814.18	21 548.69	23 819.30	36 124.72	36 049.03	22 014.57	21 953.99	21 548.69
Weighted average linkage	37 751.18	23 807.70	37 684.15	37 684.15	19 658.34	23 595.87	19 285.65	23 819.30	37 296.32	37 221.20	21 957.55	21 953.99	21 548.69
Centroid linkage	36 289.74	26 451.67	36 259.23	35 724.71	22 114.92	22 814.18	21 548.69	23 819.30	36 230.09	35 455.13	22 014.57	21 953.99	21 548.69
Weighted centroid linkage	36 306.17	23 820.75	36 249.50	36 242.18	23 819.30	21 953.99	23 905.48	23 819.30	36 232.04	36 157.79	22 014.57	21 953.99	21 548.69
Ward's method	20 389.38	19 460.40	20 378.98	20 376.61	19 460.40	24 098.66	19 337.21	20 389.38	20 389.38	20 389.38	20 389.38	20 389.38	20 389.38
Partitioning methods:													
Stepwise k-means	19 368.26	19 368.26	19 365.78	19 354.50	19 368.26	22 407.37	19 283.70	19 368.26	19 368.26	19 368.26	19 368.26	19 368.26	19 368.26
Stepwise h-means	19 369.28	19 369.28	19 366.80	19 346.59	19 369.28	22 407.37	19 283.70	19 369.28	19 369.28	19 369.28	19 369.28	19 369.28	19 369.28
Stepwise j-means	19 485.99	19 373.57	19 479.62	19 470.02	19 373.57	23 404.33	19 285.65	19 485.99	19 485.99	19 485.99	19 485.99	19 485.99	19 485.99
Stepwise PAM	20 665.35	19 455.91	20 658.03	20 660.51	19 460.40	21 953.99	19 332.51	20 665.35	20 665.35	20 665.35	20 665.35	20 665.35	20 665.35
Stepwise k-median	19 481.08	19 373.57	19 475.80	19 471.02	19 373.57	23 404.33	19 285.65	19 481.08	19 481.08	19 481.08	19 481.08	19 481.08	19 481.08
Stepwise h-median	19 090.17	19 567.01	19 090.27	19 090.17	19 537.92	23 595.87	19 107.81	19 090.17	19 090.17	19 090.17	19 090.17	19 090.17	19 090.17
CLUSTER partitioning methods:													
CLUSTER k-means	23 820.75	23 820.75	23 820.75	23 806.43	22 474.84	23 821.13	23 776.09	23 820.75	22 014.57	23 806.43	21 953.99	23 820.75	21 548.69
CLUSTER h-means	22 014.57	22 014.57	22 014.57	22 013.52	22 230.25	22 012.87	21 548.69	22 014.57	22 014.57	22 013.52	21 953.99	22 012.87	21 548.69
CLUSTER j-means	37 383.78	23 820.75	23 816.34	37 299.23	21 953.99	37 341.10	23 812.62	37 383.78	22 014.57	37 383.78	21 953.99	37 383.78	21 548.69
CLUSTER PAM	21 953.99	19 461.89	19 462.91	21 952.28	21 953.99	21 892.15	19 885.95	21 909.81	21 909.81	21 866.51	21 953.99	21 883.42	21 548.69
CLUSTER k-median	37 533.25	22 014.57	22 014.57	22 013.52	22 230.25	22 012.87	21 548.69	22 014.57	22 014.57	22 013.52	21 953.99	22 012.87	21 548.69
CLUSTER h-median	21 548.69	21 957.55	21 957.55	21 550.97	22 814.18	21 548.69	21 548.69	21 548.69	21 548.69	21 548.69	21 548.69	21 548.69	21 548.69

Table A.13: The results of the SSR measure for the various clustering methods using DataS2.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	19 251.10	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
k-means AS 136	19 521.53	19 521.53	19 521.53	19 529.76	21 953.99	19 518.91	19 111.09	19 521.53	19 521.53	19 521.53	19 521.53	19 521.53	19 521.53
h-means AS 136	19 537.92	19 537.92	19 537.92	19 546.82	21 953.99	19 505.59	19 111.09	19 537.92	19 537.92	19 537.92	19 537.92	19 537.92	19 537.92
j-means AS 136	19 246.33	19 521.53	19 537.92	19 243.89	21 953.99	19 237.66	19 111.09	19 246.33	19 246.33	19 246.33	19 246.33	19 246.33	19 246.33
PAM AS 136	21 953.99	19 461.89	19 462.91	21 952.28	21 953.99	21 892.15	19 885.95	21 909.81	21 909.81	21 866.51	21 953.99	21 883.42	21 548.69
k-median AS 136	19 239.51	19 521.53	19 537.92	19 235.51	21 953.99	19 234.50	19 111.09	19 239.51	19 239.51	19 239.51	19 239.51	19 239.51	19 239.51
h-median AS 136	19 111.09	19 560.06	19 537.92	19 109.65	21 953.99	19 111.09	19 110.51	19 111.09	19 111.09	19 111.09	19 111.09	19 111.09	19 111.09
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	19 251.10	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
CLUSTER h-means AS136	19 537.92	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
CLUSTER k-mediod AS 136	19 251.10	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
CLUSTER h-mediod AS 136	19 251.10	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
CLUSTER k-median AS 136	19 251.10	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
CLUSTER h-median AS 136	19 251.10	19 521.53	19 537.92	19 246.33	21 953.99	19 239.51	19 111.09	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10	19 251.10
Graph-based methods:													
Graph based MST	37 564.49	20 107.46	19 537.92	37 542.80	22 814.18	37 409.13	19 090.77	21 957.55	21 957.55	37 302.62	21 953.99	37 285.50	21 534.70
Graph based RNG	37 695.99	37 639.48	22 014.57	37 639.48	22 814.18	37 639.48	21 548.69	37 639.48	22 014.57	37 639.48	21 953.99	37 639.48	21 548.69
Graph based GG	37 348.67	26 440.16	26 440.16	37 269.52	21 953.99	37 269.52	21 630.07	37 348.67	22 014.57	37 348.67	21 953.99	37 348.67	21 548.69
Nearest neighbours methods:													
k-near method	37 417.25	19 651.67	20 113.04	37 387.94	22 814.18	37 388.02	20 122.61	22 014.57	21 938.74	37 163.21	21 953.99	36 372.40	21 548.69
Density-based methods:													
k_{th} nearest method	37 619.03	23 864.57	23 864.57	37 531.53	22 814.18	37 366.53	23 935.10	23 819.75	22 014.57	37 359.89	21 953.99	37 403.34	21 548.69

Table A.14: The results of the SSR measure for the various clustering methods using DataS2 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	21.52	42.36	32.22	31.20	50.00	31.30	44.46	42.36	39.12	37.87	48.81	29.39	46.62
Complete linkage	47.64	48.81	47.61	47.55	48.76	29.39	46.62	42.36	47.61	44.69	48.81	29.39	46.62
Average linkage	43.23	45.14	42.74	42.97	47.75	31.30	46.62	42.36	47.76	36.95	48.81	29.39	46.62
Weighted average linkage	15.87	47.63	21.69	21.69	35.09	35.75	33.04	42.36	39.12	38.77	48.75	29.39	46.62
Centroid linkage	43.66	45.14	43.62	41.98	47.89	31.30	46.62	42.36	49.16	46.70	48.81	29.39	46.62
Weighted centroid linkage	41.07	42.36	41.00	40.91	42.36	29.39	41.85	42.36	31.14	30.83	48.81	29.39	46.62
Ward's method	34.04	32.97	34.02	34.03	32.97	28.53	32.38	34.04	34.04	34.04	34.04	34.04	34.04
Partitioning methods:													
Stepwise k-means	33.82	33.82	33.80	33.74	33.82	30.12	33.10	33.82	33.82	33.82	33.82	33.82	33.82
Stepwise h-means	33.81	33.81	33.79	33.69	33.81	30.12	33.10	33.81	33.81	33.81	33.81	33.81	33.81
Stepwise j-means	34.08	33.85	34.07	34.09	33.85	29.86	33.04	34.08	34.08	34.08	34.08	34.08	34.08
Stepwise PAM	33.12	32.96	33.11	33.13	32.97	29.39	32.33	33.12	33.12	33.12	33.12	33.12	33.12
Stepwise k-median	34.09	33.85	34.09	34.10	33.85	29.86	33.04	34.09	34.09	34.09	34.09	34.09	34.09
Stepwise h-median	31.52	33.65	31.53	31.52	33.53	35.75	31.87	31.52	31.52	31.52	31.52	31.52	31.52
CLUSTER Partitioning methods:													
ClusterKMeans	42.36	42.36	42.36	47.14	34.90	42.36	42.26	42.36	48.81	47.14	29.39	42.36	46.62
Cluster h-means	48.81	48.81	48.81	48.81	31.20	48.81	46.62	48.81	48.81	48.81	29.39	48.81	46.62
ClusterJMeans	41.77	42.36	50.00	37.50	29.39	38.89	44.87	41.77	48.81	41.77	29.39	41.77	46.62
ClusterPAM	29.39	32.97	32.98	29.28	29.39	29.38	32.05	47.41	47.41	51.16	29.39	46.38	46.62
ClusterKMedian	33.65	48.81	48.81	48.81	31.20	48.81	46.62	48.81	48.81	48.81	29.39	48.81	46.62
ClusterHMedian	46.62	48.75	48.75	46.63	31.30	46.62	46.62	46.62	46.62	46.62	46.62	46.62	46.62

Table A.15: The results of the MSR measure for the various clustering methods using DataS2.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 Partitioning methods:													
AS136 k-means	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
K-Means AS 136	33.98	33.98	33.98	33.95	29.39	33.97	31.88	33.98	33.98	33.98	33.98	33.98	33.98
H-Means AS 136	33.53	33.53	33.53	33.51	29.39	33.47	31.88	33.53	33.53	33.53	33.53	33.53	33.53
K-Mediod AS 136	32.78	33.98	33.53	32.78	29.39	32.77	31.88	32.78	32.78	32.78	32.78	32.78	32.78
H-Mediod AS 136	29.39	32.97	32.98	29.28	29.39	29.38	32.05	47.41	47.41	51.16	29.39	46.38	46.62
K-Median AS 136	32.78	33.98	33.53	32.77	29.39	32.78	31.88	32.78	32.78	32.78	32.78	32.78	32.78
H-Median AS 136	31.88	33.63	33.53	31.87	29.39	31.88	31.89	31.88	31.88	31.88	31.88	31.88	31.88
AS 136 CLUSTER partitioning methods:													
Cluster K-Means AS 136	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
Cluster H-Means AS 136	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
Cluster K-Mediod AS 136	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
Cluster H-Mediod AS 136	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
Cluster K-Median AS 136	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
Cluster H-Median AS 136	32.79	33.98	33.53	32.78	29.39	32.78	31.88	32.79	32.79	32.79	32.79	32.79	32.79
Graph-based methods:													
Graph based MST	13.76	33.31	33.53	16.28	31.30	21.26	31.51	48.75	48.75	32.56	29.39	33.50	46.28
Graph based RNG	21.55	21.53	48.81	21.53	31.30	21.53	46.62	21.53	48.81	21.53	29.39	21.53	46.62
Graph based GG	38.28	50.34	50.34	39.11	29.39	39.11	28.72	38.28	48.81	38.28	29.39	38.28	46.62
Nearest neighbours methods:													
k-near method	13.01	35.07	33.32	14.49	31.30	14.50	32.99	48.81	54.00	31.89	29.39	25.86	46.62
Density-based methods:													
k_{th} nearest method	12.56	39.75	39.75	29.99	31.30	40.81	44.46	42.36	48.81	44.40	29.39	41.89	46.62

Table A.16: The results of the MSR measure for the various clustering methods using DataS2 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	57.54	47.08	57.53	57.52	58.93	83.61	60.45	47.06	51.25	51.18	59.95	83.26	59.59
Complete linkage	45.61	59.95	45.61	45.61	59.63	83.26	59.59	47.06	45.61	46.43	59.95	83.26	59.58
Average linkage	50.92	45.21	50.85	50.88	58.53	83.61	59.59	47.06	50.83	50.73	59.95	83.26	59.59
Weighted average linkage	91.65	47.04	91.63	91.63	78.36	83.48	75.51	47.06	51.25	51.22	59.65	83.26	59.59
Centroid linkage	50.89	45.21	50.88	50.64	60.71	83.61	59.59	47.06	50.85	50.50	59.95	83.26	59.59
Weighted centroid linkage	57.02	47.08	57.00	57.00	47.06	83.26	47.35	47.06	57.56	57.57	59.95	83.26	59.59
Ward's method	82.13	76.54	82.12	82.10	76.54	83.26	76.41	82.13	82.13	82.13	82.13	82.13	82.13
Partitioning methods:													
Stepwise k-means	76.58	76.58	76.58	76.45	76.58	78.70	75.67	76.58	76.58	76.58	76.58	76.58	76.58
Stepwise h-means	76.58	76.58	76.58	76.39	76.58	78.70	75.67	76.58	76.58	76.58	76.58	76.58	76.58
Stepwise j-means	74.57	76.62	74.60	74.66	76.62	83.26	75.51	74.57	74.57	74.57	74.57	74.57	74.57
Stepwise PAM	83.04	76.54	83.04	83.02	76.54	83.26	76.41	83.04	83.04	83.04	83.04	83.04	83.04
Stepwise k-median	74.60	76.62	74.63	74.66	76.62	83.26	75.51	74.60	74.60	74.60	74.60	74.60	74.60
Stepwise h-median	75.12	76.70	75.15	75.12	76.25	83.48	75.73	75.12	75.12	75.12	75.12	75.12	75.12
CLUSTER partitioning methods:													
CLUSTER k-means	47.08	47.08	47.08	47.08	83.04	47.08	47.28	47.08	59.95	47.08	83.26	47.08	59.59
CLUSTER h-means	59.95	59.95	59.95	59.95	78.70	59.95	59.59	59.95	59.95	59.95	83.26	59.95	59.59
CLUSTER j-means	57.46	47.08	58.93	57.45	83.26	57.46	59.23	57.46	59.95	57.46	83.26	57.46	59.59
CLUSTER PAM	83.26	76.38	76.38	83.26	83.26	83.25	76.25	59.19	59.19	58.41	83.26	58.41	59.59
CLUSTER k-median	57.51	59.95	59.95	59.95	78.70	59.95	59.59	59.95	59.95	59.95	83.26	59.95	59.59
CLUSTER h-median	59.59	59.65	59.65	59.55	83.61	59.59	59.59	59.59	59.59	59.59	59.59	59.59	59.59

Table A.17: The results of the PD measure for the various clustering methods using DataS2.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
k-means AS 136	76.58	76.58	76.58	76.58	83.26	76.57	75.76	76.58	76.58	76.58	76.58	76.58	76.58
h-means AS 136	76.25	76.25	76.25	76.25	83.26	76.27	75.76	76.25	76.25	76.25	76.25	76.25	76.25
j-means AS 136	75.88	76.58	76.25	75.91	83.26	75.91	75.76	75.88	75.88	75.88	75.88	75.88	75.88
PAM AS 136	83.26	76.38	76.38	83.26	83.26	83.25	76.25	59.19	59.19	58.41	83.26	58.41	59.59
k-median AS 136	75.91	76.58	76.25	75.93	83.26	75.93	75.76	75.91	75.91	75.91	75.91	75.91	75.91
h-median AS 136	75.76	76.70	76.25	75.76	83.26	75.76	75.76	75.76	75.76	75.76	75.76	75.76	75.76
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
CLUSTER h-means AS136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
CLUSTER k-mediod AS 136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
CLUSTER h-mediod AS 136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
CLUSTER k-median AS 136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
CLUSTER h-median AS 136	75.85	76.58	76.25	75.88	83.26	75.91	75.76	75.85	75.85	75.85	75.85	75.85	75.85
Graph-based methods:													
Graph based MST	91.79	73.18	76.25	91.80	83.61	91.87	75.12	59.65	59.65	57.52	83.26	57.51	59.35
Graph based RNG	57.54	57.46	59.95	57.46	83.61	57.46	59.59	57.46	59.95	57.46	83.26	57.46	59.59
Graph based GG	57.55	45.19	45.19	57.57	83.26	57.57	75.74	57.55	59.95	57.55	83.26	57.55	59.59
Nearest neighbours methods:													
k-near method	91.63	78.36	73.18	91.62	83.61	91.62	73.18	59.95	59.63	57.71	83.26	58.06	59.59
Density-based methods:													
k_{th} nearest method	57.54	58.93	58.93	57.53	83.61	57.47	60.45	47.04	59.95	51.29	83.26	51.32	59.59

Table A.18: The results of the PD measure for the various clustering methods using DataS2 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	26 503.61	5 854.75	26 402.67	26 355.21	5 843.44	4 076.68	6 596.02	5 854.00	26 049.24	25 802.87	4 287.39	4 375.07	4 443.19
Complete linkage	13 047.27	4 287.39	12 992.78	12 958.72	4 252.33	4 375.07	4 443.19	5 854.00	12 992.78	12 865.30	4 287.39	4 375.07	4 440.89
Average linkage	24 620.97	7 645.18	24 400.27	24 508.59	4 457.75	4 076.68	4 443.19	5 854.00	24 318.12	24 219.27	4 287.39	4 375.07	4 443.19
Weighted average linkage	26 662.68	5 823.20	26 573.34	26 573.34	3 221.18	4 534.98	3 243.41	5 854.00	26 049.24	25 940.12	4 255.81	4 375.07	4 443.19
Centroid linkage	24 544.09	7 645.18	24 490.98	23 723.53	3 812.27	4 076.68	4 443.19	5 854.00	24 415.69	23 261.48	4 287.39	4 375.07	4 443.19
Weighted centroid linkage	24 788.18	5 854.75	24 707.98	24 704.87	5 854.00	4 375.07	6 569.39	5 854.00	24 654.29	24 542.70	4 287.39	4 375.07	4 443.19
Ward's method	3 068.13	3 147.21	3 064.16	3 064.00	3 147.21	5 711.69	3 316.51	3 068.13	3 068.13	3 068.13	3 068.13	3 068.13	3 068.13
Partitioning methods:													
Stepwise k-means	3 241.43	3 241.43	3 238.16	3 253.19	3 241.43	4 440.71	3 250.04	3 241.43	3 241.43	3 241.43	3 241.43	3 241.43	3 241.43
Stepwise h-means	3 236.12	3 236.12	3 232.85	3 250.19	3 236.12	4 440.71	3 250.04	3 236.12	3 236.12	3 236.12	3 236.12	3 236.12	3 236.12
Stepwise j-means	3 326.76	3 251.71	3 320.73	3 314.32	3 251.71	5 164.39	3 243.41	3 326.76	3 326.76	3 326.76	3 326.76	3 326.76	3 326.76
Stepwise PAM	3 245.48	3 142.35	3 244.91	3 241.74	3 147.21	4 375.07	3 300.87	3 245.48	3 245.48	3 245.48	3 245.48	3 245.48	3 245.48
Stepwise k-median	3 323.55	3 251.71	3 318.47	3 315.29	3 251.71	5 164.39	3 243.41	3 323.55	3 323.55	3 323.55	3 323.55	3 323.55	3 323.55
Stepwise h-median	3 324.61	3 296.47	3 320.95	3 324.61	3 329.92	4 534.98	3 301.83	3 324.61	3 324.61	3 324.61	3 324.61	3 324.61	3 324.61
CLUSTER partitioning methods:													
CLUSTER k-means	5 854.75	5 854.75	5 854.75	5 837.01	3 867.87	5 858.25	6 003.58	5 854.75	4 287.39	5 837.01	4 375.07	5 854.75	4 443.19
CLUSTER h-means	4 287.39	4 287.39	4 287.39	4 286.50	4 408.93	4 285.53	4 443.19	4 287.39	4 287.39	4 286.50	4 375.07	4 285.53	4 443.19
CLUSTER j-means	26 084.99	5 854.75	5 843.44	26 036.49	4 375.07	26 055.41	6 034.39	26 084.99	4 287.39	26 084.99	4 375.07	26 084.99	4 443.19
CLUSTER PAM	4 375.07	3 163.12	3 161.15	4 375.07	4 375.07	4 360.61	3 106.04	3 931.77	3 931.77	4 337.21	4 375.07	4 357.13	4 443.19
CLUSTER k-median	26 387.18	4 287.39	4 287.39	4 286.50	4 408.93	4 285.53	4 443.19	4 287.39	4 287.39	4 286.50	4 375.07	4 285.53	4 443.19
CLUSTER h-median	4 443.19	4 255.81	4 255.81	4 431.74	4 076.68	4 443.19	4 443.19	4 443.19	4 443.19	4 443.19	4 443.19	4 443.19	4 443.19

Table A.19: The results of the ADM measure for the various clustering methods using DataS2.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
k-means AS 136	3 273.89	3 273.89	3 273.89	3 261.45	4 375.07	3 273.34	3 308.34	3 273.89	3 273.89	3 273.89	3 273.89	3 273.89	3 273.89
h-means AS 136	3 329.92	3 329.92	3 329.92	3 317.74	4 375.07	3 335.29	3 308.34	3 329.92	3 329.92	3 329.92	3 329.92	3 329.92	3 329.92
j-means AS 136	3 320.54	3 273.89	3 329.92	3 315.70	4 375.07	3 319.63	3 308.34	3 320.54	3 320.54	3 320.54	3 320.54	3 320.54	3 320.54
PAM AS 136	4 375.07	3 163.12	3 161.15	4 375.07	4 375.07	4 360.61	3 106.04	3 931.77	3 931.77	4 337.21	4 375.07	4 357.13	4 443.19
k-median AS 136	3 323.01	3 273.89	3 329.92	3 317.59	4 375.07	3 321.61	3 308.34	3 323.01	3 323.01	3 323.01	3 323.01	3 323.01	3 323.01
h-median AS 136	3 308.34	3 294.87	3 329.92	3 306.54	4 375.07	3 308.34	3 304.86	3 308.34	3 308.34	3 308.34	3 308.34	3 308.34	3 308.34
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
CLUSTER h-means AS136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
CLUSTER k-mediod AS 136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
CLUSTER h-mediod AS 136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
CLUSTER k-median AS 136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
CLUSTER h-median AS 136	3 326.13	3 273.89	3 329.92	3 320.54	4 375.07	3 323.01	3 308.34	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13	3 326.13
Graph-based methods:													
Graph based MST	26 170.60	3 725.54	3 329.92	26 130.29	4 076.68	25 887.95	3 328.12	4 255.81	4 255.81	25 804.11	4 375.07	25 722.64	4 423.75
Graph based RNG	26 617.84	26 559.64	4 287.39	26 559.64	4 076.68	26 559.64	4 443.19	26 559.64	4 287.39	26 559.64	4 375.07	26 559.64	4 443.19
Graph based GG	26 106.60	7 634.23	7 634.23	26 016.50	4 375.07	26 016.50	4 447.69	26 106.60	4 287.39	26 106.60	4 375.07	26 106.60	4 443.19
Nearest neighbours methods:													
k-near method	26 276.71	3 211.94	3 724.72	26 206.90	4 076.68	26 206.82	3 924.84	4 287.39	4 228.85	25 947.58	4 375.07	24 881.49	4 443.19
Density-based methods:													
k_{th} nearest method	26 559.35	5 876.82	5 876.82	26 402.70	4 076.68	26 101.93	6 596.02	5 852.21	4 287.39	26 156.29	4 375.07	26 250.97	4 443.19

Table A.20: The results of the ADM measure for the various clustering methods using DataS2 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	26 556.83	16 907.03	26 490.01	26 457.06	16 883.69	15 895.85	17 101.54	16 905.80	26 289.82	26 139.29	15 048.80	15 549.53	14 955.53
Complete linkage	20 945.04	15 048.80	20 919.67	20 889.90	15 011.55	15 549.53	14 955.53	16 905.80	20 919.67	20 593.98	15 048.80	15 549.53	14 954.82
Average linkage	25 455.79	18 498.72	25 339.03	25 396.23	15 345.35	15 895.85	14 955.53	16 905.80	25 303.51	25 256.40	15 048.80	15 549.53	14 955.53
Weighted average linkage	26 719.37	16 876.54	26 656.81	26 656.81	13 544.26	16 117.05	13 467.04	16 905.80	26 289.82	26 224.87	15 012.26	15 549.53	14 955.53
Centroid linkage	25 441.03	18 498.72	25 417.65	24 993.40	15 042.34	15 895.85	14 955.53	16 905.80	25 392.55	24 777.49	15 048.80	15 549.53	14 955.53
Weighted centroid linkage	25 534.53	16 907.03	25 493.86	25 487.85	16 905.80	15 549.53	17 070.27	16 905.80	25 507.87	25 458.19	15 048.80	15 549.53	14 955.53
Ward's method	13 889.19	13 492.79	13 881.52	13 880.38	13 492.79	16 988.94	13 507.81	13 889.19	13 889.19	13 889.19	13 889.19	13 889.19	13 889.19
Partitioning methods:													
Stepwise k-means	13 492.39	13 492.39	13 490.20	13 492.60	13 492.39	15 463.75	13 468.46	13 492.39	13 492.39	13 492.39	13 492.39	13 492.39	13 492.39
Stepwise h-means	13 492.96	13 492.96	13 490.77	13 490.22	13 492.96	15 463.75	13 468.46	13 492.96	13 492.96	13 492.96	13 492.96	13 492.96	13 492.96
Stepwise j-means	13 694.52	13 494.75	13 690.30	13 683.82	13 494.75	16 467.50	13 467.04	13 694.52	13 694.52	13 694.52	13 694.52	13 694.52	13 694.52
Stepwise PAM	14 518.40	13 489.37	14 516.02	14 513.54	13 492.79	15 549.53	13 501.89	14 518.40	14 518.40	14 518.40	14 518.40	14 518.40	14 518.40
Stepwise k-median	13 691.20	13 494.75	13 687.45	13 684.23	13 494.75	16 467.50	13 467.04	13 691.20	13 691.20	13 691.20	13 691.20	13 691.20	13 691.20
Stepwise h-median	13 635.05	13 612.91	13 635.68	13 635.05	13 614.38	16 117.05	13 588.75	13 635.05	13 635.05	13 635.05	13 635.05	13 635.05	13 635.05
CLUSTER partitioning methods:													
CLUSTER k-means	16 907.03	16 907.03	16 907.03	16 885.86	15 676.34	16 907.18	16 910.40	16 907.03	15 048.80	16 885.86	15 549.53	16 907.03	14 955.53
CLUSTER h-means	15 048.80	15 048.80	15 048.80	15 048.19	15 494.44	15 047.24	14 955.53	15 048.80	15 048.80	15 048.19	15 549.53	15 047.24	14 955.53
CLUSTER j-means	26 382.21	16 907.03	16 883.69	26 325.56	15 549.53	26 351.98	16 943.29	26 382.21	15 048.80	26 382.21	15 549.53	26 382.21	14 955.53
CLUSTER PAM	15 549.53	13 500.56	13 500.99	15 548.90	15 549.53	15 509.24	13 712.69	14 996.89	14 996.89	15 429.37	15 549.53	15 449.42	14 955.53
CLUSTER k-median	26 507.06	15 048.80	15 048.80	15 048.19	15 494.44	15 047.24	14 955.53	15 048.80	15 048.80	15 048.19	15 549.53	15 047.24	14 955.53
CLUSTER h-median	14 955.53	15 012.26	15 012.26	14 954.71	15 895.85	14 955.53	14 955.53	14 955.53	14 955.53	14 955.53	14 955.53	14 955.53	14 955.53

Table A.21: The results of the AWCD measure for the various clustering methods using DataS2.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
k-means AS 136	13 609.47	13 609.47	13 609.47	13 610.76	15 549.53	13 608.50	13 585.57	13 609.47	13 609.47	13 609.47	13 609.47	13 609.47	13 609.47
h-means AS 136	13 614.38	13 614.38	13 614.38	13 616.81	15 549.53	13 602.71	13 585.57	13 614.38	13 614.38	13 614.38	13 614.38	13 614.38	13 614.38
j-means AS 136	13 547.74	13 609.47	13 614.38	13 546.67	15 549.53	13 543.82	13 585.57	13 547.74	13 547.74	13 547.74	13 547.74	13 547.74	13 547.74
PAM AS 136	15 549.53	13 500.56	13 500.99	15 548.90	15 549.53	15 509.24	13 712.69	14 996.89	14 996.89	15 429.37	15 549.53	15 449.42	14 955.53
k-median AS 136	13 544.82	13 609.47	13 614.38	13 541.99	15 549.53	13 542.06	13 585.57	13 544.82	13 544.82	13 544.82	13 544.82	13 544.82	13 544.82
h-median AS 136	13 585.57	13 610.57	13 614.38	13 586.76	15 549.53	13 585.57	13 585.67	13 585.57	13 585.57	13 585.57	13 585.57	13 585.57	13 585.57
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
CLUSTER h-means AS136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
CLUSTER k-mediod AS 136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
CLUSTER h-mediod AS 136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
CLUSTER k-median AS 136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
CLUSTER h-median AS 136	13 550.69	13 609.47	13 614.38	13 547.74	15 549.53	13 544.82	13 585.57	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69	13 550.69
Graph-based methods:													
Graph based MST	26 606.75	13 856.54	13 614.38	26 590.55	15 895.85	26 488.72	13 634.96	15 012.26	15 012.26	26 353.55	15 549.53	26 340.37	14 965.73
Graph based RNG	26 653.45	26 604.07	15 048.80	26 604.07	15 895.85	26 604.07	14 955.53	26 604.07	15 048.80	26 604.07	15 549.53	26 604.07	14 955.53
Graph based GG	26 337.34	18 467.23	18 467.23	26 268.18	15 549.53	26 268.18	15 063.83	26 337.34	15 048.80	26 337.34	15 549.53	26 337.34	14 955.53
Nearest neighbours methods:													
k-near method	26 504.27	13 540.04	13 860.15	26 473.41	15 895.85	26 473.39	13 895.04	15 048.80	14 973.77	26 252.31	15 549.53	25 615.71	14 955.53
Density-based methods:													
k_{th} nearest method	26 594.81	16 943.41	16 943.41	26 504.68	15 895.85	26 369.07	17 101.54	16 906.51	15 048.80	26 343.05	15 549.53	26 382.89	14 955.53

Table A.22: The results of the AWCD measure for the various clustering methods using DataS2 (continue).

Methods:	Initial	Partitioning methods, using initial solution as starting solution:											
	solution:	k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	00 : 00 : 07	00 : 00 : 16	00 : 00 : 13	00 : 00 : 07	00 : 00 : 07	00 : 01 : 00	00 : 00 : 07	00 : 00 : 55	00 : 01 : 10	00 : 00 : 20	00 : 00 : 14	00 : 04 : 48	00 : 00 : 14
Complete linkage	00 : 00 : 07	00 : 00 : 13	00 : 00 : 17	00 : 00 : 08	00 : 00 : 07	00 : 00 : 33	00 : 00 : 08	00 : 00 : 53	00 : 01 : 13	00 : 00 : 20	00 : 00 : 13	00 : 04 : 49	00 : 00 : 15
Average linkage	00 : 00 : 07	00 : 00 : 12	00 : 00 : 28	00 : 00 : 07	00 : 00 : 07	00 : 00 : 47	00 : 00 : 08	00 : 00 : 52	00 : 01 : 23	00 : 00 : 20	00 : 00 : 13	00 : 04 : 47	00 : 00 : 14
Weighted average linkage	00 : 00 : 28	00 : 00 : 39	00 : 00 : 33	00 : 00 : 28	00 : 00 : 29	00 : 00 : 51	00 : 00 : 28	00 : 01 : 12	00 : 01 : 37	00 : 00 : 41	00 : 00 : 34	00 : 05 : 16	00 : 00 : 36
Centroid linkage	00 : 02 : 38	00 : 02 : 43	00 : 02 : 43	00 : 02 : 39	00 : 02 : 38	00 : 03 : 18	00 : 02 : 38	00 : 03 : 23	00 : 03 : 41	00 : 02 : 52	00 : 02 : 45	00 : 07 : 18	00 : 02 : 46
Weighted centroid linkage	00 : 00 : 08	00 : 00 : 15	00 : 00 : 16	00 : 00 : 09	00 : 00 : 08	00 : 00 : 41	00 : 00 : 09	00 : 00 : 53	00 : 01 : 06	00 : 00 : 21	00 : 00 : 15	00 : 04 : 49	00 : 00 : 15
Ward's method	00 : 00 : 07	00 : 00 : 10	00 : 00 : 12	00 : 00 : 07	00 : 00 : 07	00 : 00 : 23	00 : 00 : 07	00 : 00 : 53	00 : 01 : 28	00 : 00 : 15	00 : 00 : 13	00 : 04 : 34	00 : 00 : 14
Partitioning methods:													
Stepwise k-means	00 : 41 : 56	00 : 41 : 56	00 : 42 : 03	00 : 41 : 56	00 : 41 : 56	00 : 42 : 11	00 : 41 : 56	00 : 42 : 09	00 : 42 : 40	00 : 42 : 05	00 : 42 : 01	00 : 46 : 29	00 : 42 : 04
Stepwise h-means	00 : 00 : 42	00 : 00 : 42	00 : 00 : 50	00 : 00 : 42	00 : 00 : 42	00 : 00 : 57	00 : 00 : 42	00 : 00 : 55	00 : 01 : 26	00 : 00 : 51	00 : 00 : 48	00 : 05 : 16	00 : 00 : 49
Stepwise j-means	01 : 36 : 35	01 : 36 : 37	01 : 36 : 43	01 : 36 : 35	01 : 36 : 35	01 : 36 : 53	01 : 36 : 35	01 : 36 : 47	01 : 37 : 41	01 : 36 : 44	01 : 36 : 40	01 : 40 : 33	01 : 36 : 42
Stepwise PAM	02 : 51 : 38	02 : 51 : 41	02 : 51 : 43	02 : 51 : 38	02 : 51 : 38	02 : 51 : 51	02 : 51 : 38	02 : 51 : 49	02 : 52 : 31	02 : 51 : 46	02 : 51 : 44	02 : 56 : 34	02 : 51 : 45
Stepwise k-median	00 : 01 : 07	00 : 01 : 08	00 : 01 : 12	00 : 01 : 07	00 : 01 : 07	00 : 01 : 25	00 : 01 : 07	00 : 01 : 19	00 : 02 : 11	00 : 01 : 16	00 : 01 : 13	00 : 05 : 05	00 : 01 : 14
Stepwise h-median	00 : 02 : 19	00 : 02 : 21	00 : 02 : 29	00 : 02 : 19	00 : 02 : 19	00 : 02 : 39	00 : 02 : 19	00 : 02 : 31	00 : 03 : 05	00 : 02 : 27	00 : 02 : 25	00 : 06 : 42	00 : 02 : 26
CLUSTER partitioning methods:													
CLUSTER k-means	00 : 00 : 37	00 : 00 : 37	00 : 00 : 37	00 : 00 : 45	00 : 01 : 02	00 : 00 : 37	00 : 00 : 38	00 : 00 : 50	00 : 00 : 43	00 : 01 : 20	00 : 05 : 20	00 : 00 : 47	00 : 00 : 44
CLUSTER h-means	00 : 00 : 18	00 : 00 : 18	00 : 00 : 18	00 : 00 : 24	00 : 00 : 41	00 : 00 : 18	00 : 00 : 18	00 : 00 : 32	00 : 00 : 24	00 : 01 : 01	00 : 05 : 19	00 : 00 : 29	00 : 00 : 25
CLUSTER j-means	00 : 02 : 24	00 : 02 : 33	00 : 02 : 24	00 : 02 : 35	00 : 03 : 05	00 : 02 : 24	00 : 02 : 25	00 : 02 : 39	00 : 02 : 31	00 : 02 : 59	00 : 07 : 07	00 : 02 : 39	00 : 02 : 32
CLUSTER PAM	00 : 10 : 14	00 : 10 : 21	00 : 10 : 14	00 : 10 : 19	00 : 10 : 27	00 : 10 : 15	00 : 10 : 15	00 : 10 : 25	00 : 10 : 20	00 : 11 : 09	00 : 15 : 02	00 : 10 : 24	00 : 10 : 22
CLUSTER k-median	00 : 00 : 33	00 : 00 : 33	00 : 00 : 34	00 : 00 : 38	00 : 00 : 56	00 : 00 : 33	00 : 00 : 33	00 : 00 : 47	00 : 00 : 39	00 : 01 : 15	00 : 05 : 25	00 : 00 : 43	00 : 00 : 40
CLUSTER h-median	00 : 00 : 21	00 : 00 : 23	00 : 00 : 21	00 : 00 : 36	00 : 00 : 54	00 : 00 : 21	00 : 00 : 21	00 : 00 : 34	00 : 00 : 27	00 : 01 : 13	00 : 05 : 14	00 : 00 : 31	00 : 00 : 28

Table A.23: The CPU times for the various clustering methods using DataS2.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	00 : 00 : 49	00 : 00 : 50	00 : 00 : 49	00 : 00 : 57	00 : 01 : 05	00 : 00 : 49	00 : 00 : 49	00 : 01 : 01	00 : 00 : 55	00 : 01 : 42	00 : 05 : 15	00 : 00 : 59	00 : 00 : 57
k-means AS 136	00 : 00 : 50	00 : 00 : 50	00 : 00 : 50	00 : 01 : 13	00 : 01 : 06	00 : 00 : 50	00 : 00 : 50	00 : 01 : 01	00 : 00 : 56	00 : 02 : 03	00 : 05 : 15	00 : 00 : 59	00 : 00 : 57
h-means AS 136	00 : 00 : 49	00 : 00 : 49	00 : 00 : 50	00 : 01 : 12	00 : 01 : 04	00 : 00 : 49	00 : 00 : 49	00 : 01 : 01	00 : 00 : 55	00 : 01 : 57	00 : 05 : 19	00 : 00 : 59	00 : 00 : 56
j-means AS 136	00 : 00 : 57	00 : 00 : 57	00 : 00 : 57	00 : 01 : 04	00 : 01 : 12	00 : 00 : 57	00 : 00 : 58	00 : 01 : 09	00 : 01 : 03	00 : 01 : 50	00 : 05 : 19	00 : 01 : 06	00 : 01 : 05
PAM AS 136	00 : 01 : 05	00 : 01 : 13	00 : 01 : 06	00 : 01 : 11	00 : 01 : 18	00 : 01 : 05	00 : 01 : 06	00 : 01 : 17	00 : 01 : 11	00 : 02 : 00	00 : 05 : 56	00 : 01 : 15	00 : 01 : 12
k-median AS 136	00 : 00 : 49	00 : 00 : 50	00 : 00 : 49	00 : 00 : 57	00 : 01 : 05	00 : 00 : 50	00 : 00 : 49	00 : 01 : 01	00 : 00 : 55	00 : 01 : 41	00 : 05 : 11	00 : 00 : 58	00 : 00 : 56
h-median AS 136	00 : 00 : 49	00 : 00 : 50	00 : 00 : 49	00 : 00 : 59	00 : 01 : 04	00 : 00 : 50	00 : 00 : 50	00 : 01 : 01	00 : 00 : 54	00 : 01 : 41	00 : 05 : 24	00 : 00 : 58	00 : 00 : 57
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	00 : 01 : 01	00 : 01 : 02	00 : 01 : 01	00 : 01 : 09	00 : 01 : 17	00 : 01 : 01	00 : 01 : 02	00 : 01 : 13	00 : 01 : 07	00 : 01 : 54	00 : 05 : 24	00 : 01 : 11	00 : 01 : 08
CLUSTER h-means AS136	00 : 00 : 55	00 : 00 : 56	00 : 00 : 55	00 : 01 : 03	00 : 01 : 10	00 : 00 : 56	00 : 00 : 55	00 : 01 : 07	00 : 01 : 01	00 : 01 : 47	00 : 05 : 18	00 : 01 : 04	00 : 01 : 03
CLUSTER k-mediod AS 136	00 : 01 : 42	00 : 01 : 42	00 : 01 : 42	00 : 01 : 50	00 : 01 : 58	00 : 01 : 42	00 : 01 : 42	00 : 01 : 54	00 : 01 : 48	00 : 02 : 35	00 : 06 : 05	00 : 01 : 52	00 : 01 : 49
CLUSTER h-mediod AS 136	00 : 05 : 15	00 : 05 : 16	00 : 05 : 15	00 : 05 : 23	00 : 05 : 31	00 : 05 : 15	00 : 05 : 16	00 : 05 : 27	00 : 05 : 21	00 : 06 : 08	00 : 09 : 38	00 : 05 : 25	00 : 05 : 23
CLUSTER k-median AS 136	00 : 00 : 59	00 : 01 : 00	00 : 00 : 59	00 : 01 : 07	00 : 01 : 15	00 : 00 : 59	00 : 01 : 00	00 : 01 : 11	00 : 01 : 05	00 : 01 : 52	00 : 05 : 21	00 : 01 : 09	00 : 01 : 06
CLUSTER h-median AS 136	00 : 00 : 57	00 : 00 : 58	00 : 00 : 57	00 : 01 : 04	00 : 01 : 13	00 : 00 : 57	00 : 00 : 58	00 : 01 : 09	00 : 01 : 03	00 : 01 : 50	00 : 05 : 20	00 : 01 : 07	00 : 01 : 05
Graph-based methods:													
Graph based MST	00 : 00 : 26	00 : 00 : 35	00 : 00 : 26	00 : 00 : 32	00 : 00 : 52	00 : 00 : 27	00 : 00 : 27	00 : 00 : 45	00 : 00 : 34	00 : 01 : 11	00 : 05 : 09	00 : 00 : 40	00 : 00 : 34
Graph based RNG	03 : 23 : 47	03 : 23 : 47	03 : 23 : 47	03 : 23 : 50	03 : 24 : 33	03 : 23 : 48	03 : 23 : 48	03 : 24 : 00	03 : 23 : 54	03 : 24 : 21	03 : 28 : 32	03 : 24 : 00	03 : 23 : 54
Graph based GG	00 : 09 : 46	00 : 09 : 52	00 : 09 : 46	00 : 09 : 51	00 : 10 : 20	00 : 09 : 46	00 : 09 : 46	00 : 10 : 00	00 : 09 : 53	00 : 10 : 21	00 : 14 : 48	00 : 10 : 00	00 : 09 : 54
Nearest neighbours methods:													
k-near method	00 : 00 : 06	00 : 00 : 16	00 : 00 : 06	00 : 00 : 11	00 : 00 : 37	00 : 00 : 07	00 : 00 : 07	00 : 00 : 24	00 : 00 : 13	00 : 00 : 51	00 : 04 : 52	00 : 00 : 21	00 : 00 : 15
Density-based methods:													
k_{th} nearest method	00 : 00 : 05	00 : 00 : 13	00 : 00 : 05	00 : 00 : 12	00 : 00 : 59	00 : 00 : 05	00 : 00 : 06	00 : 00 : 23	00 : 00 : 12	00 : 01 : 00	00 : 04 : 53	00 : 00 : 19	00 : 00 : 13

Table A.24: The CPU times for the various clustering methods using DataS2 (continue).

A.1.3 Clustering method results - DataS3

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	166 720.73	51 543.33	52 163.58	166 357.82	78 615.82	166 614.78	55 852.23	45 236.47	45 256.73	164 582.39	74 168.81	163 992.75	45 137.79
Complete linkage	55 308.55	45 743.81	45 398.83	55 298.17	79 804.55	55 229.38	45 072.59	45 252.83	45 256.73	55 189.35	55 308.55	55 261.89	45 072.59
Average linkage	55 284.71	46 809.00	46 773.69	55 274.33	82 086.85	55 246.57	46 685.65	45 252.83	45 062.54	55 142.00	55 284.71	55 280.44	45 137.79
Weighted average linkage	108 938.80	49 989.95	51 571.75	108 723.26	89 536.81	108 732.77	50 928.70	45 236.47	45 256.73	108 126.16	78 509.07	107 883.60	45 138.00
Centroid linkage	69 002.84	50 777.60	50 761.15	79 878.35	87 770.73	79 060.70	50 876.11	79 929.82	50 472.78	79 929.82	79 929.82	79 929.82	51 637.38
Weighted centroid linkage	63 688.07	46 821.47	46 737.11	63 615.41	81 110.15	63 633.15	46 587.79	45 236.47	45 256.73	63 653.28	63 688.07	63 688.07	45 137.79
Ward's method	47 485.90	45 276.33	45 276.33	47 468.23	75 060.01	47 432.44	45 199.47	45 236.47	45 062.54	47 485.90	47 485.90	47 485.90	45 125.49
Partitioning methods:													
Stepwise k-means	45 054.52	45 054.52	45 054.52	45 056.55	80 622.77	45 044.40	44 959.75	45 054.52	45 054.52	45 054.52	45 054.52	45 054.52	44 975.05
Stepwise h-means	45 054.21	45 054.21	45 054.21	45 056.24	80 622.77	45 044.09	44 959.75	45 054.21	45 054.21	45 054.21	45 054.21	45 054.21	44 975.05
Stepwise j-means	47 738.94	45 261.80	45 264.97	47 738.94	78 878.74	47 711.52	45 156.54	45 252.83	45 062.54	47 738.94	47 738.94	47 738.94	45 125.49
Stepwise PAM	73 235.08	46 149.26	45 472.18	73 235.08	80 132.68	73 127.44	45 441.02	45 236.47	45 256.73	72 905.43	73 235.08	73 050.02	45 137.79
Stepwise k-median	47 571.39	45 317.84	45 322.48	47 555.81	80 622.77	47 525.04	44 957.48	45 252.83	45 062.54	47 571.39	47 571.39	47 571.39	45 124.79
Stepwise h-median	44 950.19	45 054.52	45 055.77	44 951.43	80 622.77	44 950.98	44 956.88	44 950.19	44 950.19	44 950.19	44 950.19	44 950.19	44 950.19
CLUSTER partitioning methods:													
CLUSTER k-means	45 252.83	45 252.83	45 252.83	45 254.79	80 662.79	45 242.60	45 131.56	45 252.83	45 062.54	45 252.83	45 252.83	45 252.83	45 137.79
CLUSTER h-means	45 236.47	45 236.47	45 236.47	45 237.77	78 964.66	45 236.64	44 961.21	45 236.47	45 236.47	45 236.47	45 236.47	45 236.47	44 974.79
CLUSTER j-means	163 884.66	50 080.00	49 950.38	163 791.97	84 768.66	163 741.53	49 840.25	45 236.47	45 236.47	162 602.63	78 509.07	161 523.03	45 137.79
CLUSTER PAM	73 235.08	46 149.26	45 472.18	73 235.08	80 132.68	73 127.44	45 441.02	45 236.47	45 256.73	72 905.43	73 235.08	73 050.02	45 137.79
CLUSTER k-median	161 481.06	45 236.47	45 236.47	45 237.77	78 964.66	45 236.64	44 961.21	45 236.47	45 236.47	45 236.47	45 236.47	45 236.47	44 974.79
CLUSTER h-median	45 137.79	45 257.48	45 256.73	45 155.01	84 499.25	45 137.79	45 138.28	45 137.79	45 137.79	45 137.79	45 137.79	45 137.79	45 137.79

Table A.25: The results of the SSR measure for the various clustering methods using DataS3.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	45 570.50	45 263.01	45 263.04	45 562.06	80 826.03	45 543.87	45 158.81	45 252.83	45 062.54	45 570.50	45 570.50	45 570.50	45 125.27
k-means AS 136	45 263.01	45 263.01	45 263.01	45 262.67	80 826.03	45 242.90	45 159.29	45 259.76	45 256.73	45 263.01	45 263.01	45 263.01	45 125.49
h-means AS 136	45 263.04	45 263.04	45 263.04	45 262.69	80 826.03	45 238.98	45 159.29	45 259.76	45 256.73	45 263.04	45 263.04	45 263.04	45 125.49
j-means AS 136	45 562.06	45 263.01	45 263.04	45 559.30	80 826.03	45 543.14	45 158.81	45 252.83	45 062.54	45 562.06	45 562.06	45 562.06	45 125.27
PAM AS 136	80 826.03	45 056.04	46 015.92	80 826.03	81 059.34	80 562.65	45 440.07	45 252.83	45 056.95	80 339.29	75 881.15	80 097.08	45 137.79
k-median AS 136	45 543.87	45 263.01	45 263.04	45 538.86	80 826.03	45 523.28	45 158.81	45 252.83	45 062.54	45 543.87	45 543.87	45 543.87	45 125.27
h-median AS 136	45 158.81	45 276.33	45 276.33	45 166.53	77 071.91	45 158.81	45 159.15	45 158.81	45 158.81	45 158.81	45 158.81	45 158.81	45 125.49
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	45 252.83	45 252.83	45 252.83	45 254.79	80 662.79	45 242.60	45 131.56	45 252.83	45 062.54	45 252.83	45 252.83	45 252.83	45 137.79
CLUSTER h-means AS136	45 062.54	45 062.54	45 062.54	45 064.57	80 622.77	45 052.39	44 959.75	45 062.54	45 062.54	45 062.54	45 062.54	45 062.54	44 975.05
CLUSTER k-mediod AS 136	45 570.50	45 263.01	45 263.04	45 562.06	80 826.03	45 543.87	45 158.81	45 252.83	45 062.54	45 570.50	45 570.50	45 570.50	45 125.27
CLUSTER h-mediod AS 136	45 570.50	45 263.01	45 263.04	45 562.06	80 826.03	45 543.87	45 158.81	45 252.83	45 062.54	45 570.50	45 570.50	45 570.50	45 125.27
CLUSTER k-median AS 136	45 570.50	45 263.01	45 263.04	45 562.06	80 826.03	45 543.87	45 158.81	45 252.83	45 062.54	45 570.50	45 570.50	45 570.50	45 125.27
CLUSTER h-median AS 136	45 125.27	45 257.48	45 256.73	45 129.20	84 499.25	45 125.27	45 125.49	45 125.27	45 125.27	45 125.27	45 125.27	45 125.27	45 125.27
Graph-based methods:													
Graph based MST	159 355.01	45 488.16	45 581.68	159 309.18	91 640.76	159 297.81	48 058.83	45 236.47	45 256.73	158 164.90	74 977.45	156 367.93	45 138.00
Graph based GG	167 257.83	49 829.16	50 032.57	166 990.34	79 157.98	167 095.01	49 105.09	45 236.47	45 236.47	165 057.80	75 060.01	163 439.91	45 138.00
Nearest neighbours methods:													
k-near method	163 192.79	48 073.36	48 072.11	163 097.90	79 056.60	163 097.90	47 844.22	45 236.47	45 236.47	161 658.51	75 881.15	160 671.46	45 138.00
Density-based methods:													
k_{th} nearest method	167 047.17	51 748.87	56 121.41	166 945.54	81 747.27	166 945.10	74 972.87	45 252.83	45 256.73	165 381.54	78 509.07	163 744.06	45 138.00

Table A.26: The results of the SSR measure for the various clustering methods using DataS3 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	25.69	61.55	64.55	28.49	70.53	28.37	63.47	60.83	61.43	46.20	59.12	30.56	61.29
Complete linkage	67.79	62.34	60.81	71.13	71.98	67.76	59.88	61.41	61.43	67.08	67.79	63.80	59.88
Average linkage	71.84	59.63	59.46	71.95	56.78	72.61	59.14	61.41	59.92	73.94	71.84	69.02	61.29
Weighted average linkage	48.34	63.81	61.72	51.64	75.19	48.28	58.05	60.83	61.43	49.00	55.87	52.43	61.28
Centroid linkage	68.34	72.59	72.60	68.37	61.65	67.52	78.40	67.32	81.10	67.32	67.32	67.32	69.78
Weighted centroid linkage	62.11	59.61	59.33	62.15	55.66	62.08	58.61	60.83	61.43	60.94	62.11	62.11	61.29
Ward's method	59.99	58.98	58.98	59.98	56.28	59.96	58.81	60.83	59.92	59.99	59.99	59.99	61.28
Partitioning methods:													
Stepwise k-means	59.80	59.80	59.80	59.85	57.05	59.80	59.91	59.80	59.80	59.80	59.80	59.80	59.92
Stepwise h-means	59.80	59.80	59.80	59.85	57.05	59.80	59.91	59.80	59.80	59.80	59.80	59.80	59.92
Stepwise j-means	61.86	58.78	58.79	61.86	55.83	61.83	58.90	61.41	59.92	61.86	61.86	61.86	61.28
Stepwise PAM	60.14	59.25	60.79	60.14	53.14	60.49	59.47	60.83	61.43	67.61	60.14	60.90	61.29
Stepwise k-median	60.70	60.59	60.60	60.71	57.05	60.73	59.91	61.41	59.92	60.70	60.70	60.70	61.28
Stepwise h-median	59.92	59.80	59.80	59.92	57.05	59.92	59.91	59.92	59.92	59.92	59.92	59.92	59.92
CLUSTER partitioning methods:													
CLUSTER k-means	61.41	61.41	61.41	61.46	59.40	61.43	61.28	61.41	59.92	61.41	61.41	61.41	61.29
CLUSTER h-means	60.83	60.83	60.83	60.87	56.11	60.85	59.94	60.83	60.83	60.83	60.83	60.83	59.92
CLUSTER j-means	31.63	64.38	63.15	31.58	54.34	31.62	63.39	60.83	60.83	33.47	55.87	35.82	61.29
CLUSTER PAM	60.14	59.25	60.79	60.14	53.14	60.49	59.47	60.83	61.43	67.61	60.14	60.90	61.29
CLUSTER k-median	35.74	60.83	60.83	60.87	56.11	60.85	59.94	60.83	60.83	60.83	60.83	60.83	59.92
CLUSTER h-median	61.29	61.45	61.43	61.31	66.72	61.29	61.30	61.29	61.29	61.29	61.29	61.29	61.29

Table A.27: The results of the MSR measure for the various clustering methods using DataS3.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	59.48	58.78	58.78	59.44	69.13	59.42	58.94	61.41	59.92	59.48	59.48	59.48	61.28
k-means AS 136	58.78	58.78	58.78	58.79	69.13	58.73	58.94	61.46	61.43	58.78	58.78	58.78	61.28
h-means AS 136	58.78	58.78	58.78	58.79	69.13	58.72	58.94	61.46	61.43	58.78	58.78	58.78	61.28
j-means AS 136	59.44	58.78	58.78	59.44	69.13	59.42	58.94	61.41	59.92	59.44	59.44	59.44	61.28
PAM AS 136	69.13	59.80	60.49	69.13	51.78	69.32	59.47	61.41	59.79	72.09	52.85	70.31	61.29
k-median AS 136	59.42	58.78	58.78	59.41	69.13	59.39	58.94	61.41	59.92	59.42	59.42	59.42	61.28
h-median AS 136	58.94	58.98	58.98	58.95	58.05	58.94	58.94	58.94	58.94	58.94	58.94	58.94	61.28
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	61.41	61.41	61.41	61.46	59.40	61.43	61.28	61.41	59.92	61.41	61.41	61.41	61.29
CLUSTER h-means AS136	59.92	59.92	59.92	59.96	57.05	59.90	59.91	59.92	59.92	59.92	59.92	59.92	59.92
CLUSTER k-mediod AS 136	59.48	58.78	58.78	59.44	69.13	59.42	58.94	61.41	59.92	59.48	59.48	59.48	61.28
CLUSTER h-mediod AS 136	59.48	58.78	58.78	59.44	69.13	59.42	58.94	61.41	59.92	59.48	59.48	59.48	61.28
CLUSTER k-median AS 136	59.48	58.78	58.78	59.44	69.13	59.42	58.94	61.41	59.92	59.48	59.48	59.48	61.28
CLUSTER h-median AS 136	61.28	61.45	61.43	61.28	66.72	61.28	61.28	61.28	61.28	61.28	61.28	61.28	61.28
Graph-based methods:													
Graph based MST	29.87	60.62	60.49	33.06	55.66	34.73	58.25	60.83	61.43	41.06	53.92	47.92	61.28
Graph based GG	22.21	63.68	64.09	23.10	64.64	22.65	61.80	60.83	60.83	47.51	56.28	38.70	61.28
Nearest neighbours methods:													
k-near method	31.57	62.25	62.24	34.01	76.68	34.01	61.18	60.83	60.83	41.80	52.85	45.37	61.28
Density-based methods:													
k_{th} nearest method	21.77	61.80	63.44	26.87	61.64	29.20	56.97	61.41	61.43	35.17	55.87	28.81	61.28

Table A.28: The results of the MSR measure for the various clustering methods using DataS3 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	107.59	42.96	42.96	107.59	76.53	107.59	47.18	39.71	42.96	97.40	65.00	101.32	44.07
Complete linkage	38.20	41.45	41.45	38.20	66.35	38.20	44.07	42.96	42.96	39.58	38.20	39.58	44.07
Average linkage	39.64	41.45	41.45	38.74	65.13	38.74	44.07	42.96	37.06	38.08	39.64	39.64	44.07
Weighted average linkage	75.53	42.96	41.45	75.53	65.00	75.53	49.61	39.71	42.96	75.53	65.13	75.53	44.07
Centroid linkage	43.84	46.68	46.68	40.79	64.33	40.77	47.28	40.77	38.96	40.77	40.77	40.77	47.29
Weighted centroid linkage	54.71	41.45	41.45	54.71	65.71	54.71	44.07	39.71	42.96	48.64	54.71	54.71	44.07
Ward's method	44.42	41.45	41.45	44.42	65.00	44.42	44.07	39.71	37.06	44.42	44.42	44.42	44.07
Partitioning methods:													
Stepwise k-means	41.45	41.45	41.45	41.45	65.00	41.45	44.07	41.45	41.45	41.45	41.45	41.45	44.07
Stepwise h-means	41.45	41.45	41.45	41.45	65.00	41.45	44.07	41.45	41.45	41.45	41.45	41.45	44.07
Stepwise j-means	46.17	41.45	41.45	46.17	65.00	46.17	44.07	42.96	37.06	46.17	46.17	46.17	44.07
Stepwise PAM	65.00	52.05	52.05	65.00	65.13	64.67	52.05	39.71	42.96	56.01	65.00	63.02	44.07
Stepwise k-median	46.75	41.45	41.45	46.75	65.00	46.75	44.07	42.96	37.06	46.75	46.75	46.75	44.07
Stepwise h-median	44.07	41.45	41.45	44.07	65.00	44.07	44.07	44.07	44.07	44.07	44.07	44.07	44.07
CLUSTER partitioning methods:													
CLUSTER k-means	42.96	42.96	42.96	42.96	65.20	42.96	44.07	42.96	37.06	42.96	42.96	42.96	44.07
CLUSTER h-means	39.71	39.71	39.71	39.71	65.20	39.71	44.07	39.71	39.71	39.71	39.71	39.71	44.07
CLUSTER j-means	96.97	44.07	40.45	95.50	65.13	95.50	44.33	39.71	39.71	95.50	65.13	92.42	44.07
CLUSTER PAM	65.00	52.05	52.05	65.00	65.13	64.67	52.05	39.71	42.96	56.01	65.00	63.02	44.07
CLUSTER k-median	92.42	39.71	39.71	39.71	65.20	39.71	44.07	39.71	39.71	39.71	39.71	39.71	44.07
CLUSTER h-median	44.07	42.96	42.96	44.07	65.00	44.07	44.07	44.07	44.07	44.07	44.07	44.07	44.07

Table A.29: The results of the PD measure for the various clustering methods using DataS3.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
	AS 136 partitioning methods:												
AS 136	44.07	41.45	41.45	44.07	65.21	44.07	44.07	42.96	37.06	44.07	44.07	44.07	44.07
k-means AS 136	41.45	41.45	41.45	41.45	65.21	41.45	44.07	42.96	42.96	41.45	41.45	41.45	44.07
h-means AS 136	41.45	41.45	41.45	41.45	65.21	41.45	44.07	42.96	42.96	41.45	41.45	41.45	44.07
j-means AS 136	44.07	41.45	41.45	44.07	65.21	44.07	44.07	42.96	37.06	44.07	44.07	44.07	44.07
PAM AS 136	65.21	41.45	52.05	65.21	65.13	65.21	52.05	42.96	41.45	56.22	65.00	62.19	44.07
k-median AS 136	44.07	41.45	41.45	44.07	65.21	44.07	44.07	42.96	37.06	44.07	44.07	44.07	44.07
h-median AS 136	44.07	41.45	41.45	44.07	65.00	44.07	44.07	44.07	44.07	44.07	44.07	44.07	44.07
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	42.96	42.96	42.96	42.96	65.20	42.96	44.07	42.96	37.06	42.96	42.96	42.96	44.07
CLUSTER h-means AS136	37.06	37.06	37.06	37.06	65.00	37.32	44.07	37.06	37.06	37.06	37.06	37.06	44.07
CLUSTER k-mediod AS 136	44.07	41.45	41.45	44.07	65.21	44.07	44.07	42.96	37.06	44.07	44.07	44.07	44.07
CLUSTER h-mediod AS 136	44.07	41.45	41.45	44.07	65.21	44.07	44.07	42.96	37.06	44.07	44.07	44.07	44.07
CLUSTER k-median AS 136	44.07	41.45	41.45	44.07	65.21	44.07	44.07	42.96	37.06	44.07	44.07	44.07	44.07
CLUSTER h-median AS 136	44.07	42.96	42.96	44.07	65.00	44.07	44.07	44.07	44.07	44.07	44.07	44.07	44.07
Graph-based methods:													
Graph based MST	119.40	52.05	52.05	119.40	67.98	119.40	54.81	39.71	42.96	100.86	65.00	95.50	44.07
Graph based GG	119.40	42.96	44.07	119.40	65.00	119.40	47.96	39.71	39.71	101.01	65.00	100.04	44.07
Nearest neighbours methods:													
k-near method	106.23	45.62	45.62	106.23	65.21	106.23	46.73	39.71	39.71	96.91	65.00	94.89	44.07
Density-based methods:													
k _{th} nearest method	108.00	42.96	44.70	108.00	65.00	108.00	60.68	42.96	42.96	100.13	65.13	100.04	44.07

Table A.30: The results of the PD measure for the various clustering methods using DataS3 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	115 753.20	5 124.55	5 463.93	115 405.13	12 711.19	115 649.07	6 600.42	3 680.31	3 732.95	113 262.56	10 601.56	112 944.02	3 642.92
Complete linkage	6 659.79	3 587.64	3 523.45	6 653.55	10 295.68	6 621.20	3 391.77	3 735.34	3 732.95	6 648.54	6 659.79	6 657.51	3 391.77
Average linkage	6 054.51	4 088.40	4 103.34	6 048.09	13 239.87	6 041.49	4 067.66	3 735.34	3 581.31	6 020.23	6 054.51	6 054.00	3 642.92
Weighted average linkage	26 818.21	4 854.83	5 103.53	26 727.99	14 153.69	26 736.60	5 080.27	3 680.31	3 732.95	26 506.18	12 569.78	26 386.89	3 642.81
Centroid linkage	14 503.96	5 183.38	5 177.24	26 966.13	16 341.86	26 384.99	4 990.09	27 007.73	5 515.68	27 007.73	27 007.73	27 007.73	5 316.09
Weighted centroid linkage	8 798.58	4 121.06	4 116.61	8 769.67	13 673.12	8 786.38	4 061.55	3 680.31	3 732.95	8 753.06	8 798.58	8 798.58	3 642.92
Ward's method	3 836.83	3 488.23	3 488.23	3 833.59	11 357.84	3 824.13	3 431.47	3 680.31	3 581.31	3 836.83	3 836.83	3 836.83	3 636.19
Partitioning methods:													
Stepwise k-means	3 564.75	3 564.75	3 564.75	3 558.74	12 732.73	3 560.45	3 421.67	3 564.75	3 564.75	3 564.75	3 564.75	3 564.75	3 435.91
Stepwise h-means	3 564.79	3 564.79	3 564.79	3 558.79	12 732.73	3 560.50	3 421.67	3 564.79	3 564.79	3 564.79	3 564.79	3 564.79	3 435.91
Stepwise j-means	3 620.09	3 513.19	3 511.16	3 620.09	13 476.54	3 615.70	3 410.58	3 735.34	3 581.31	3 620.09	3 620.09	3 620.09	3 636.19
Stepwise PAM	10 534.95	3 731.92	3 773.47	10 534.95	13 189.81	10 507.97	3 537.36	3 680.31	3 732.95	10 403.29	10 534.95	10 473.91	3 642.92
Stepwise k-median	3 820.39	3 532.64	3 530.33	3 816.73	12 732.73	3 809.66	3 420.87	3 735.34	3 581.31	3 820.39	3 820.39	3 820.39	3 636.18
Stepwise h-median	3 411.29	3 564.75	3 566.72	3 409.27	12 732.73	3 411.72	3 419.38	3 411.29	3 411.29	3 411.29	3 411.29	3 411.29	3 411.29
CLUSTER partitioning methods:													
CLUSTER k-means	3 735.34	3 735.34	3 735.34	3 729.32	13 607.73	3 727.45	3 637.69	3 735.34	3 581.31	3 735.34	3 735.34	3 735.34	3 642.92
CLUSTER h-means	3 680.31	3 680.31	3 680.31	3 675.01	12 277.05	3 679.06	3 418.94	3 680.31	3 680.31	3 680.31	3 680.31	3 680.31	3 436.11
CLUSTER j-means	112 726.76	4 584.99	4 568.81	112 625.52	14 708.49	112 576.15	4 560.57	3 680.31	3 680.31	111 356.78	12 569.78	110 181.98	3 642.92
CLUSTER PAM	10 534.95	3 731.92	3 773.47	10 534.95	13 189.81	10 507.97	3 537.36	3 680.31	3 732.95	10 403.29	10 534.95	10 473.91	3 642.92
CLUSTER k-median	110 142.62	3 680.31	3 680.31	3 675.01	12 277.05	3 679.06	3 418.94	3 680.31	3 680.31	3 680.31	3 680.31	3 680.31	3 436.11
CLUSTER h-median	3 642.92	3 731.32	3 732.95	3 645.66	14 919.98	3 642.92	3 649.04	3 642.92	3 642.92	3 642.92	3 642.92	3 642.92	3 642.92

Table A.31: The results of the ADM measure for the various clustering methods using DataS3.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	3 415.21	3 512.59	3 513.00	3 416.14	10 924.94	3 416.60	3 405.62	3 735.34	3 581.31	3 415.21	3 415.21	3 415.21	3 636.30
k-means AS 136	3 512.59	3 512.59	3 512.59	3 510.86	10 924.94	3 510.60	3 405.74	3 729.79	3 732.95	3 512.59	3 512.59	3 512.59	3 636.19
h-means AS 136	3 513.00	3 513.00	3 513.00	3 511.26	10 924.94	3 510.60	3 405.74	3 729.79	3 732.95	3 513.00	3 513.00	3 513.00	3 636.19
j-means AS 136	3 416.14	3 512.59	3 513.00	3 414.80	10 924.94	3 416.91	3 405.62	3 735.34	3 581.31	3 416.14	3 416.14	3 416.14	3 636.30
PAM AS 136	10 924.94	3 566.68	3 607.91	10 924.94	13 936.60	10 876.60	3 537.26	3 735.34	3 569.25	10 871.67	11 700.46	10 778.61	3 642.92
k-median AS 136	3 416.60	3 512.59	3 513.00	3 417.22	10 924.94	3 417.63	3 405.62	3 735.34	3 581.31	3 416.60	3 416.60	3 416.60	3 636.30
h-median AS 136	3 405.62	3 488.23	3 488.23	3 405.76	12 089.27	3 405.62	3 405.69	3 405.62	3 405.62	3 405.62	3 405.62	3 405.62	3 636.19
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	3 735.34	3 735.34	3 735.34	3 729.32	13 607.73	3 727.45	3 637.69	3 735.34	3 581.31	3 735.34	3 735.34	3 735.34	3 642.92
CLUSTER h-means AS136	3 581.31	3 581.31	3 581.31	3 575.30	12 732.73	3 577.78	3 421.67	3 581.31	3 581.31	3 581.31	3 581.31	3 581.31	3 435.91
CLUSTER k-mediod AS 136	3 415.21	3 512.59	3 513.00	3 416.14	10 924.94	3 416.60	3 405.62	3 735.34	3 581.31	3 415.21	3 415.21	3 415.21	3 636.30
CLUSTER h-mediod AS 136	3 415.21	3 512.59	3 513.00	3 416.14	10 924.94	3 416.60	3 405.62	3 735.34	3 581.31	3 415.21	3 415.21	3 415.21	3 636.30
CLUSTER k-median AS 136	3 415.21	3 512.59	3 513.00	3 416.14	10 924.94	3 416.60	3 405.62	3 735.34	3 581.31	3 415.21	3 415.21	3 415.21	3 636.30
CLUSTER h-median AS 136	3 636.30	3 731.32	3 732.95	3 636.32	14 919.98	3 636.30	3 636.19	3 636.30	3 636.30	3 636.30	3 636.30	3 636.30	3 636.30
Graph-based methods:													
Graph based MST	104 189.63	3 952.48	4 063.85	104 120.00	17 806.03	104 074.00	4 659.37	3 680.31	3 732.95	102 811.54	11 608.78	100 204.43	3 642.81
Graph based GG	116 183.35	4 857.15	4 621.92	115 935.69	12 439.05	116 034.22	4 471.50	3 680.31	3 680.31	113 844.67	11 357.84	112 110.37	3 642.81
Nearest neighbours methods:													
k-near method	110 914.68	4 051.94	4 052.81	110 808.60	9 058.76	110 808.60	4 038.44	3 680.31	3 680.31	108 698.65	11 700.46	107 833.79	3 642.81
Density-based methods:													
k_{th} nearest method	116 090.80	5 394.57	6 413.57	115 985.45	13 860.59	115 985.45	18 272.61	3 735.34	3 732.95	114 183.98	12 569.78	112 624.16	3 642.81

Table A.32: The results of the ADM measure for the various clustering methods using DataS3 (continue).

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	116 000.71	35 476.30	35 654.52	115 774.71	54 209.13	115 932.81	38 499.47	31 409.66	31 366.03	114 449.96	51 028.77	114 070.36	31 375.67
Complete linkage	37 982.89	31 486.63	31 363.24	37 973.29	54 625.18	37 929.78	31 316.83	31 366.73	31 366.03	37 888.83	37 982.89	37 951.92	31 316.83
Average linkage	37 734.44	32 416.80	32 411.38	37 724.65	55 867.93	37 703.99	32 408.40	31 366.73	31 288.21	37 633.68	37 734.44	37 742.42	31 375.67
Weighted average linkage	73 257.55	34 199.84	35 476.15	73 128.02	59 926.12	73 139.87	35 352.64	31 409.66	31 366.03	72 737.48	53 474.72	72 628.35	31 373.00
Centroid linkage	47 195.58	34 921.82	34 923.58	56 004.62	60 966.49	55 437.73	35 684.73	56 054.96	34 917.97	56 054.96	56 054.96	56 054.96	35 914.63
Weighted centroid linkage	43 709.32	32 461.62	32 432.16	43 672.29	55 975.97	43 670.55	32 376.32	31 409.66	31 366.03	43 696.07	43 709.32	43 709.32	31 375.67
Ward's method	32 772.33	31 457.56	31 457.56	32 757.65	51 565.14	32 726.46	31 428.33	31 409.66	31 288.21	32 772.33	32 772.33	32 772.33	31 342.61
Partitioning methods:													
Stepwise k-means	31 275.12	31 275.12	31 275.12	31 277.36	55 024.64	31 268.93	31 215.06	31 275.12	31 275.12	31 275.12	31 275.12	31 275.12	31 248.65
Stepwise h-means	31 275.07	31 275.07	31 275.07	31 277.31	55 024.64	31 268.90	31 215.06	31 275.07	31 275.07	31 275.07	31 275.07	31 275.07	31 248.65
Stepwise j-means	32 998.54	31 458.68	31 461.53	32 998.54	54 027.62	32 978.65	31 377.55	31 366.73	31 288.21	32 998.54	32 998.54	32 998.54	31 342.61
Stepwise PAM	50 267.40	32 035.68	31 701.69	50 267.40	54 541.97	50 197.77	31 754.37	31 409.66	31 366.03	49 971.41	50 267.40	50 131.84	31 375.67
Stepwise k-median	33 015.12	31 352.87	31 351.11	33 001.73	55 024.64	32 976.53	31 216.03	31 366.73	31 288.21	33 015.12	33 015.12	33 015.12	31 341.92
Stepwise h-median	31 207.49	31 275.12	31 275.51	31 207.93	55 024.64	31 207.90	31 212.87	31 207.49	31 207.49	31 207.49	31 207.49	31 207.49	31 207.49
CLUSTER partitioning methods:													
CLUSTER k-means	31 366.73	31 366.73	31 366.73	31 368.93	55 714.28	31 360.80	31 345.92	31 366.73	31 288.21	31 366.73	31 366.73	31 366.73	31 375.67
CLUSTER h-means	31 409.66	31 409.66	31 409.66	31 411.09	54 309.92	31 409.74	31 212.97	31 409.66	31 409.66	31 409.66	31 409.66	31 409.66	31 248.69
CLUSTER j-means	113 929.53	34 374.23	34 373.07	113 858.65	57 691.51	113 824.99	34 355.18	31 409.66	31 409.66	113 025.22	53 474.72	112 341.84	31 375.67
CLUSTER PAM	50 267.40	32 035.68	31 701.69	50 267.40	54 541.97	50 197.77	31 754.37	31 409.66	31 366.03	49 971.41	50 267.40	50 131.84	31 375.67
CLUSTER k-median	112 321.41	31 409.66	31 409.66	31 411.09	54 309.92	31 409.74	31 212.97	31 409.66	31 409.66	31 409.66	31 409.66	31 409.66	31 248.69
CLUSTER h-median	31 375.67	31 366.46	31 366.03	31 392.17	57 651.57	31 375.67	31 374.54	31 375.67	31 375.67	31 375.67	31 375.67	31 375.67	31 375.67

Table A.33: The results of the AWCD measure for the various clustering methods using DataS3.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	31 577.43	31 459.80	31 459.59	31 570.14	54 490.31	31 559.48	31 381.76	31 366.73	31 288.21	31 577.43	31 577.43	31 577.43	31 341.76
k-means AS 136	31 275.12	31 459.80	31 459.80	31 460.13	54 490.31	31 449.29	31 381.59	31 366.24	31 366.03	31 459.80	31 459.80	31 459.80	31 342.61
h-means AS 136	31 275.07	31 459.59	31 459.59	31 459.92	54 490.31	31 447.14	31 381.59	31 366.24	31 366.03	31 459.59	31 459.59	31 459.59	31 342.61
j-means AS 136	32 998.54	31 459.80	31 459.59	31 566.97	54 490.31	31 558.02	31 381.76	31 366.73	31 288.21	31 570.14	31 570.14	31 570.14	31 341.76
PAM AS 136	50 267.40	31 275.58	31 874.97	54 490.31	55 234.84	54 318.89	31 752.63	31 366.73	31 276.85	54 181.64	52 151.91	54 015.27	31 375.67
k-median AS 136	33 015.12	31 459.80	31 459.59	31 555.43	54 490.31	31 547.46	31 381.76	31 366.73	31 288.21	31 559.48	31 559.48	31 559.48	31 341.76
h-median AS 136	31 207.49	31 457.56	31 457.56	31 386.66	52 718.32	31 381.76	31 381.74	31 381.76	31 381.76	31 381.76	31 381.76	31 381.76	31 342.61
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	31 366.73	31 366.73	31 366.73	31 368.93	55 714.28	31 360.80	31 345.92	31 366.73	31 288.21	31 366.73	31 366.73	31 366.73	31 375.67
CLUSTER h-means AS136	31 288.21	31 288.21	31 288.21	31 290.45	55 024.64	31 282.19	31 215.06	31 288.21	31 288.21	31 288.21	31 288.21	31 288.21	31 248.65
CLUSTER k-mediod AS 136	31 577.43	31 459.80	31 459.59	31 570.14	54 490.31	31 559.48	31 381.76	31 366.73	31 288.21	31 577.43	31 577.43	31 577.43	31 341.76
CLUSTER h-mediod AS 136	31 577.43	31 459.80	31 459.59	31 570.14	54 490.31	31 559.48	31 381.76	31 366.73	31 288.21	31 577.43	31 577.43	31 577.43	31 341.76
CLUSTER k-median AS 136	31 577.43	31 459.80	31 459.59	31 570.14	54 490.31	31 559.48	31 381.76	31 366.73	31 288.21	31 577.43	31 577.43	31 577.43	31 341.76
CLUSTER h-median AS 136	31 341.76	31 366.46	31 366.03	31 343.40	57 651.57	31 341.76	31 342.61	31 341.76	31 341.76	31 341.76	31 341.76	31 341.76	31 341.76
Graph-based methods:													
Graph based MST	110 567.13	31 561.83	31 817.58	110 528.90	61 590.61	110 511.25	33 590.40	31 409.66	31 366.03	109 719.95	51 411.98	108 226.82	31 373.00
Graph based GG	116 350.38	34 079.79	34 378.60	116 181.59	53 842.43	116 247.55	34 073.16	31 409.66	31 409.66	114 765.73	51 565.14	113 698.94	31 373.00
Nearest neighbours methods:													
k-near method	113 104.61	33 045.40	33 046.02	113 041.06	52 845.11	113 041.06	33 097.33	31 409.66	31 409.66	111 977.38	52 151.91	111 372.31	31 373.00
Density-based methods:													
k _{th} nearest method	116 226.99	35 421.03	38 485.27	116 160.60	56 021.03	116 159.23	51 153.60	31 366.73	31 366.03	114 937.87	53 474.72	113 837.74	31 373.00

Table A.34: The results of the AWCD measure for the various clustering methods using DataS3 (continue).

Methods:	Initial	Partitioning methods, using initial solution as starting solution:											
	solution:	k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
Hierarchical methods:													
Single linkage	00 : 00 : 48	00 : 02 : 43	00 : 00 : 48	00 : 02 : 48	00 : 06 : 46	00 : 00 : 48	00 : 00 : 50	00 : 25 : 33	00 : 01 : 41	00 : 27 : 56	01 : 17 : 04	00 : 03 : 56	00 : 01 : 56
Complete linkage	00 : 00 : 46	00 : 01 : 23	00 : 00 : 47	00 : 01 : 20	00 : 05 : 19	00 : 00 : 46	00 : 00 : 48	00 : 24 : 37	00 : 01 : 31	00 : 37 : 37	01 : 14 : 27	00 : 02 : 16	00 : 01 : 45
Average linkage	00 : 00 : 43	00 : 01 : 35	00 : 00 : 44	00 : 01 : 18	00 : 05 : 16	00 : 00 : 44	00 : 00 : 44	00 : 25 : 27	00 : 01 : 29	00 : 36 : 09	01 : 18 : 16	00 : 02 : 13	00 : 01 : 43
Weighted average linkage	00 : 04 : 25	00 : 06 : 06	00 : 04 : 26	00 : 05 : 51	00 : 09 : 33	00 : 04 : 26	00 : 04 : 28	00 : 29 : 05	00 : 05 : 12	00 : 29 : 11	01 : 19 : 13	00 : 06 : 19	00 : 05 : 27
Centroid linkage	00 : 41 : 23	00 : 42 : 15	00 : 41 : 24	00 : 41 : 58	00 : 45 : 56	00 : 41 : 24	00 : 41 : 24	01 : 06 : 07	00 : 42 : 09	01 : 16 : 49	01 : 58 : 56	00 : 42 : 53	00 : 42 : 23
Weighted centroid linkage	00 : 00 : 47	00 : 01 : 51	00 : 00 : 48	00 : 03 : 38	00 : 04 : 46	00 : 00 : 48	00 : 00 : 48	00 : 24 : 59	00 : 01 : 32	00 : 28 : 59	01 : 11 : 41	00 : 02 : 13	00 : 01 : 46
Ward's method	00 : 00 : 49	00 : 01 : 05	00 : 00 : 50	00 : 01 : 41	00 : 04 : 14	00 : 00 : 50	00 : 00 : 49	00 : 24 : 12	00 : 01 : 34	00 : 38 : 55	01 : 15 : 19	00 : 02 : 13	00 : 01 : 47
Partitioning methods:													
Stepwise k-means	09 : 18 : 43	09 : 18 : 43	09 : 18 : 43	09 : 20 : 55	09 : 23 : 07	09 : 18 : 43	09 : 18 : 44	09 : 41 : 51	09 : 19 : 27	09 : 56 : 07	10 : 31 : 36	09 : 20 : 15	09 : 19 : 44
Stepwise h-means	00 : 05 : 17	00 : 05 : 18	00 : 05 : 17	00 : 07 : 30	00 : 09 : 42	00 : 05 : 18	00 : 05 : 18	00 : 28 : 32	00 : 06 : 03	00 : 42 : 54	01 : 18 : 17	00 : 06 : 49	00 : 06 : 17
Stepwise j-means	07 : 54 : 33	07 : 54 : 51	07 : 54 : 33	07 : 54 : 47	07 : 56 : 15	07 : 54 : 33	07 : 54 : 34	08 : 17 : 57	07 : 55 : 17	08 : 12 : 06	08 : 24 : 57	07 : 55 : 57	07 : 55 : 31
Stepwise PAM	19 : 53 : 10	19 : 53 : 10	19 : 53 : 10	19 : 53 : 10	19 : 54 : 10	19 : 53 : 10	19 : 53 : 10	20 : 15 : 10	19 : 53 : 10	20 : 08 : 10	20 : 24 : 10	19 : 54 : 10	19 : 54 : 08
Stepwise k-median	00 : 05 : 30	00 : 05 : 47	00 : 05 : 31	00 : 06 : 03	00 : 10 : 11	00 : 05 : 31	00 : 05 : 31	00 : 29 : 28	00 : 06 : 16	00 : 42 : 21	01 : 20 : 27	00 : 06 : 57	00 : 06 : 29
Stepwise h-median	00 : 13 : 20	00 : 13 : 25	00 : 13 : 21	00 : 15 : 00	00 : 18 : 01	00 : 13 : 21	00 : 13 : 21	00 : 36 : 44	00 : 14 : 05	00 : 49 : 50	01 : 25 : 30	00 : 14 : 43	00 : 14 : 20
CLUSTER partitioning methods:													
CLUSTER k-means	00 : 53 : 25	00 : 53 : 26	00 : 53 : 25	00 : 55 : 38	00 : 57 : 50	00 : 53 : 26	00 : 53 : 26	01 : 17 : 12	00 : 54 : 10	01 : 28 : 47	02 : 06 : 04	00 : 54 : 47	00 : 54 : 24
CLUSTER h-means	00 : 02 : 30	00 : 02 : 30	00 : 02 : 30	00 : 04 : 29	00 : 06 : 12	00 : 02 : 30	00 : 02 : 31	00 : 25 : 09	00 : 03 : 15	00 : 38 : 22	01 : 15 : 21	00 : 03 : 54	00 : 03 : 30
CLUSTER j-means	01 : 16 : 49	01 : 19 : 02	01 : 16 : 50	01 : 17 : 23	01 : 20 : 24	01 : 16 : 50	01 : 16 : 51	01 : 41 : 50	01 : 17 : 42	01 : 39 : 39	02 : 30 : 12	01 : 19 : 58	01 : 17 : 58
CLUSTER PAM	00 : 35 : 29	00 : 36 : 27	00 : 35 : 30	00 : 35 : 50	00 : 36 : 45	00 : 35 : 30	00 : 35 : 31	00 : 58 : 33	00 : 36 : 13	00 : 51 : 06	01 : 07 : 18	00 : 36 : 50	00 : 36 : 28
CLUSTER k-median	00 : 08 : 29	00 : 08 : 30	00 : 08 : 30	00 : 10 : 25	00 : 12 : 05	00 : 08 : 30	00 : 08 : 30	00 : 31 : 00	00 : 09 : 13	00 : 44 : 09	01 : 20 : 52	00 : 09 : 53	00 : 09 : 30
CLUSTER h-median	00 : 03 : 13	00 : 03 : 21	00 : 03 : 14	00 : 06 : 32	00 : 07 : 05	00 : 03 : 13	00 : 03 : 14	00 : 27 : 10	00 : 03 : 58	00 : 40 : 32	01 : 15 : 46	00 : 04 : 34	00 : 04 : 13

Table A.35: The CPU times for the various clustering methods using DataS3.

Methods:	Initial solution:	Partitioning methods, using initial solution as starting solution:											
		k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
AS 136 partitioning methods:													
AS 136	00 : 06 : 01	00 : 06 : 08	00 : 06 : 02	00 : 06 : 52	00 : 10 : 01	00 : 06 : 01	00 : 06 : 02	00 : 29 : 58	00 : 06 : 46	00 : 46 : 31	01 : 18 : 10	00 : 07 : 25	00 : 06 : 59
k-means AS 136	00 : 06 : 08	00 : 06 : 16	00 : 06 : 09	00 : 07 : 49	00 : 14 : 01	00 : 06 : 09	00 : 06 : 10	00 : 53 : 36	00 : 07 : 37	01 : 25 : 38	02 : 30 : 41	00 : 08 : 59	00 : 08 : 05
h-means AS 136	00 : 06 : 02	00 : 06 : 09	00 : 06 : 03	00 : 07 : 43	00 : 13 : 55	00 : 06 : 03	00 : 06 : 04	00 : 53 : 29	00 : 07 : 31	01 : 25 : 33	02 : 30 : 20	00 : 08 : 52	00 : 07 : 58
j-means AS 136	00 : 06 : 52	00 : 07 : 06	00 : 06 : 53	00 : 08 : 49	00 : 14 : 44	00 : 06 : 53	00 : 06 : 54	00 : 54 : 33	00 : 08 : 23	01 : 27 : 26	02 : 30 : 30	00 : 09 : 39	00 : 08 : 49
PAM AS 136	00 : 10 : 01	00 : 11 : 27	00 : 10 : 03	00 : 11 : 42	00 : 17 : 37	00 : 10 : 02	00 : 10 : 03	00 : 58 : 23	00 : 11 : 32	01 : 27 : 45	02 : 38 : 44	00 : 13 : 00	00 : 11 : 58
k-median AS 136	00 : 06 : 01	00 : 06 : 15	00 : 06 : 03	00 : 07 : 42	00 : 13 : 51	00 : 06 : 02	00 : 06 : 03	00 : 53 : 41	00 : 07 : 32	01 : 26 : 45	02 : 29 : 48	00 : 08 : 47	00 : 07 : 58
h-median AS 136	00 : 06 : 02	00 : 06 : 14	00 : 06 : 04	00 : 08 : 50	00 : 14 : 27	00 : 06 : 03	00 : 06 : 04	00 : 55 : 01	00 : 07 : 33	01 : 25 : 29	02 : 30 : 21	00 : 08 : 48	00 : 07 : 59
AS 136 CLUSTER partitioning methods:													
CLUSTER k-means AS 136	00 : 29 : 58	00 : 30 : 06	00 : 29 : 59	00 : 33 : 02	00 : 38 : 23	00 : 29 : 59	00 : 30 : 00	01 : 17 : 43	00 : 31 : 28	01 : 45 : 51	02 : 54 : 48	00 : 32 : 45	00 : 31 : 55
CLUSTER h-means AS136	00 : 06 : 46	00 : 06 : 53	00 : 06 : 47	00 : 09 : 49	00 : 15 : 11	00 : 06 : 47	00 : 06 : 48	00 : 53 : 59	00 : 08 : 15	01 : 24 : 55	02 : 32 : 49	00 : 09 : 36	00 : 08 : 44
CLUSTER k-mediod AS 136	00 : 46 : 31	00 : 46 : 45	00 : 46 : 33	00 : 48 : 12	00 : 54 : 23	00 : 46 : 32	00 : 46 : 33	01 : 34 : 22	00 : 48 : 01	02 : 07 : 20	03 : 10 : 34	00 : 49 : 19	00 : 48 : 28
CLUSTER h-mediod AS 136	01 : 18 : 10	01 : 18 : 24	01 : 18 : 11	01 : 19 : 51	01 : 26 : 02	01 : 18 : 10	01 : 18 : 12	02 : 06 : 01	01 : 19 : 41	02 : 39 : 04	03 : 42 : 11	01 : 20 : 59	01 : 20 : 07
CLUSTER k-median AS 136	00 : 07 : 25	00 : 07 : 39	00 : 07 : 27	00 : 09 : 06	00 : 15 : 17	00 : 07 : 26	00 : 07 : 27	00 : 55 : 13	00 : 08 : 55	01 : 28 : 14	02 : 31 : 29	00 : 10 : 14	00 : 09 : 22
CLUSTER h-median AS 136	00 : 06 : 59	00 : 07 : 11	00 : 07 : 01	00 : 09 : 29	00 : 14 : 52	00 : 07 : 00	00 : 07 : 01	00 : 55 : 03	00 : 08 : 29	01 : 23 : 05	02 : 31 : 19	00 : 09 : 45	00 : 08 : 56
Graph-based methods:													
Graph based MST	00 : 03 : 11	00 : 05 : 48	00 : 03 : 12	00 : 03 : 46	00 : 07 : 44	00 : 03 : 12	00 : 03 : 13	00 : 28 : 20	00 : 04 : 03	00 : 34 : 16	01 : 18 : 20	00 : 06 : 12	00 : 04 : 17
Graph based GG	00 : 33 : 17	00 : 35 : 27	00 : 33 : 18	00 : 33 : 51	00 : 35 : 10	00 : 33 : 18	00 : 33 : 19	00 : 58 : 08	00 : 34 : 09	00 : 45 : 52	01 : 05 : 05	00 : 36 : 25	00 : 34 : 24
Nearest neighbours methods:													
k-near method	00 : 00 : 37	00 : 02 : 59	00 : 00 : 38	00 : 01 : 12	00 : 04 : 36	00 : 00 : 38	00 : 00 : 40	00 : 26 : 08	00 : 01 : 29	00 : 29 : 36	01 : 16 : 02	00 : 03 : 46	00 : 01 : 44
Density-based methods:													
k_{th} nearest method	00 : 00 : 34	00 : 03 : 07	00 : 00 : 35	00 : 01 : 08	00 : 05 : 58	00 : 00 : 35	00 : 00 : 37	00 : 25 : 59	00 : 01 : 28	00 : 25 : 47	01 : 17 : 59	00 : 03 : 44	00 : 01 : 43

Table A.36: The CPU times for the various clustering methods using DataS3 (continue).

A.1.4 Clustering method results - USA13509

Methods	Starting solution	k-means	k-median	h-means	h-median	Cluster k-median	Cluster h-means	Cluster h-median
SSR values:								
Single linkage	11 533 590.97	2 144 592.60	11 516 968.05	2 097 499.86	2 482 627.44	10 823 003.29	1 785 848.99	1 955 526.99
Complete linkage	2 078 444.23	1 796 929.77	2 073 534.52	1 755 553.52	1 827 051.81	2 075 862.27	1 689 306.33	1 747 943.77
Average linkage	1 922 666.29	1 730 670.52	1 919 333.66	1 728 617.74	1 720 938.97	1 920 658.65	1 722 921.51	1 715 635.61
Weighted average	8 472 005.24	1 769 331.39	8 276 042.53	1 751 119.37	1 870 251.59	7 059 169.67	1 724 157.44	1 727 909.93
Weighted centroid linkage	2 555 983.41	1 778 634.42	2 545 059.22	1 762 605.37	1 767 504.01	2 428 507.33	1 707 568.30	1 718 315.60
Ward's method	1 767 758.62	1 645 386.71	1 764 600.64	1 647 646.49	1 630 249.50	1 767 758.62	1 660 660.93	1 649 533.74
Stepwise k-median	1 870 820.64	1 692 427.94	1 862 461.48	1 661 423.06	1 699 369.58	1 870 820.64	1 650 471.02	1 695 917.34
Stepwise h-means	1 626 646.38	1 626 646.38	1 626 088.96	1 626 646.38	1 618 437.79	1 626 646.38	1 626 646.38	1 619 253.08
Stepwise h-median	1 611 881.56	1 615 047.80	1 611 881.56	1 617 158.50	1 611 938.12	1 611 881.56	1 611 881.56	1 611 881.56
CLUSTER k-median	11 615 361.26	1 877 266.36	1 877 068.78	1 877 266.36	1 871 691.24	1 877 266.36	1 877 266.36	1 871 445.90
CLUSTER h-means	1 877 266.36	1 877 266.36	1 877 068.78	1 877 266.36	1 871 691.24	1 877 266.36	1 877 266.36	1 871 445.90
CLUSTER h-median	2 140 322.71	1 914 523.51	2 140 322.71	1 978 995.90	2 140 260.49	2 140 322.71	1 866 193.96	2 140 312.30
AS 136	1 729 701.35	1 645 874.28	1 722 848.08	1 645 810.53	1 641 835.93	1 729 701.35	1 656 623.44	1 650 977.90
k-near method	12 978 827.48	1 826 775.21	12 860 241.08	1 887 630.75	1 959 685.75	11 740 375.74	1 855 663.64	1 892 426.82
k_{th} nearest method	15 304 969.50	2 072 392.98	15 243 008.60	2 025 568.66	2 209 025.83	13 559 874.31	1 771 093.37	1 889 372.62
MSR values:								
Single linkage	2 356.41	6 795.11	2 386.06	6 633.68	6 534.63	2 669.11	6 639.90	6 515.62
Complete linkage	7 414.01	6 941.04	7 401.98	6 889.88	6 871.61	7 295.44	6 576.96	6 639.75
Average linkage	6 973.08	6 876.73	6 961.21	6 868.75	6 784.63	6 867.32	6 658.16	6 771.95
Weighted average	3 544.54	6 840.12	3 401.11	6 787.24	6 719.91	3 229.41	6 513.41	6 693.60
Weighted centroid linkage	7 099.30	6 710.39	7 101.81	6 687.16	6 649.23	6 800.36	6 647.65	6 640.53
Ward's method	6 835.08	6 611.39	6 826.42	6 619.82	6 517.19	6 835.08	6 623.81	6 554.59
Stepwise k-median	6 777.20	6 453.43	6 761.87	6 446.07	6 373.47	6 777.20	6 546.13	6 411.93
Stepwise h-means	6 534.51	6 534.51	6 530.12	6 534.51	6 457.62	6 534.51	6 534.51	6 409.86
Stepwise h-median	6 406.54	6 442.19	6 406.54	6 456.99	6 408.54	6 406.54	6 406.54	6 406.54
CLUSTER k-median	1 679.19	6 404.75	6 400.80	6 404.75	6 341.98	6 404.75	6 404.75	6 327.77
CLUSTER h-means	6 404.75	6 404.75	6 400.80	6 404.75	6 341.98	6 404.75	6 404.75	6 327.77
CLUSTER h-median	5 840.99	6 357.75	5 840.99	6 340.07	5 852.05	5 840.99	6 357.28	5 842.00
AS 136	6 716.36	6 530.80	6 672.62	6 532.94	6 380.03	6 716.36	6 577.65	6 401.59
k-near method	2 329.01	6 358.05	2 357.12	6 394.94	6 110.04	1 951.87	6 351.12	6 212.57
k_{th} nearest method	1 137.07	6 717.12	1 217.28	6 664.66	6 689.33	1 620.58	6 485.71	6 568.03

Table A.37: The results of the SSR and MSR measures for various clustering methods using USA13509.

Methods	Starting solution	k-means	k-median	h-means	h-median	Cluster k-median	Cluster h-means	Cluster h-median
PD values:								
Single linkage	4 779.81	858.37	4 710.05	858.37	1 329.56	3 846.81	774.20	849.27
Complete linkage	695.85	672.34	695.85	762.34	696.30	754.24	815.73	866.39
Average linkage	759.96	718.56	759.96	718.56	755.46	759.96	768.64	742.15
Weighted average	3 273.38	742.15	3 203.73	718.56	972.65	2 467.28	870.43	768.64
Weigthed Centroid linkage	1 139.34	709.11	1 121.31	784.06	818.03	871.12	784.06	818.03
Ward's method	1 158.88	895.97	1 107.42	895.97	901.04	1 158.88	895.97	901.04
Stepwise k-median	817.79	1 178.50	1 563.20	1 101.66	1 462.62	1 563.20	793.91	1 462.62
Stepwise h-means	475.41	793.91	793.91	793.91	820.65	793.91	793.91	1 089.92
Stepwise h-median	475.71	793.91	820.65	793.91	820.65	820.65	820.65	820.65
CLUSTER k-median	2 837.45	599.69	599.69	599.69	618.51	599.69	599.69	615.15
CLUSTER h-means	599.69	599.69	599.69	599.69	618.51	599.69	599.69	615.15
CLUSTER h-median	790.43	597.46	790.43	644.26	790.43	790.43	599.24	790.43
AS 136	657.76	512.10	598.52	512.10	729.56	657.76	515.44	715.05
k-near method	5 527.96	991.44	5 527.96	937.91	1 386.33	4 983.59	991.44	1 038.95
k_{th} nearest method	5 812.16	858.37	5 804.90	850.61	1 034.62	5 456.35	796.66	853.52
ADM values:								
Single linkage	6 827 524.90	73 660.23	6 812 604.89	69 398.37	137 189.41	6 135 587.34	35 264.35	52 906.00
Complete linkage	56 687.90	37 539.25	56 580.10	34 541.63	40 144.71	56 644.46	29 028.65	31 838.59
Average linkage	47 259.40	32 895.69	47 165.58	32 792.67	32 938.20	47 176.77	31 665.33	32 039.29
Weighted average	1 815 076.17	37 003.05	1 796 684.07	34 261.76	46 895.44	1 453 196.22	29 686.19	32 023.38
Weighted centroid linkage	91 336.45	34 996.13	90 532.28	34 210.37	34 812.71	79 453.23	30 704.48	31 346.74
Ward's method	28 593.13	25 349.12	28 529.89	25 303.25	25 110.81	28 593.13	27 238.75	26 907.03
Stepwise k-median	29 924.51	24 958.09	29 673.61	25 056.09	24 954.49	29 924.51	25 853.50	25 049.61
Stepwise h-means	23 871.62	23 871.62	23 876.73	23 871.62	23 670.05	23 871.62	23 871.62	23 490.22
Stepwise h-median	24 157.41	24 303.01	24 157.41	24 435.98	24 124.17	24 157.41	24 157.41	24 157.41
CLUSTER k-median	7 297 982.64	39 328.56	39 354.75	39 328.56	40 282.70	39 328.56	39 328.56	40 107.14
CLUSTER h-means	39 328.56	39 328.56	39 354.75	39 328.56	40 282.70	39 328.56	39 328.56	40 107.14
CLUSTER h-median	60 871.75	40 465.25	60 871.75	43 926.79	60 858.60	60 871.75	38 819.48	60 859.92
AS 136	26 565.86	24 102.36	26 535.10	23 972.18	24 164.34	26 565.86	25 329.69	25 632.35
k-near method	8 455 315.40	36 342.77	8 352 313.56	38 750.44	45 316.51	7 208 458.46	38 312.89	41 251.14
k_{th} nearest method	10 753 775.86	67 324.78	10 697 280.64	62 037.15	84 572.67	8 995 956.32	33 726.28	46 457.54

Table A.38: The results of the PD and ADM measures for various clustering methods using USA13509.

Methods	Starting solution	k-means	k-median	h-means	h-median	Cluster k-median	Cluster h-means	Cluster h-median
AWCD values:								
Single linkage	7 970 327.67	1 469 297.55	7 957 612.07	1 441 252.89	1 712 173.23	7 431 703.76	1 225 050.88	1 345 237.08
Complete linkage	1 410 856.30	1 230 933.67	1 407 531.00	1 205 447.92	1 256 461.30	1 409 402.66	1 162 090.01	1 203 128.82
Average linkage	1 317 103.75	1 188 178.54	1 314 807.49	1 187 021.89	1 190 747.08	1 315 824.38	1 182 262.67	1 184 652.20
Weighted average	5 768 463.30	1 214 176.91	5 641 880.40	1 201 485.86	1 286 872.64	4 804 747.27	1 186 785.64	1 195 294.58
Weighted centroid linkage	1 728 097.07	1 219 637.03	1 721 025.30	1 210 849.60	1 223 267.50	1 644 515.78	1 175 293.41	1 188 060.38
Ward's method	1 214 929.95	1 133 434.24	1 212 938.92	1 133 123.60	1 132 409.98	1 214 929.95	1 142 652.59	1 143 711.07
Stepwise k-median	1 283 997.22	1 161 449.28	1 279 688.60	1 147 728.96	1 179 136.66	1 283 997.22	1 138 545.79	1 176 314.46
Stepwise h-means	1 125 328.35	1 125 328.35	1 125 333.99	1 125 328.35	1 123 873.44	1 125 328.35	1 125 328.35	1 125 376.07
Stepwise h-median	1 122 482.68	1 120 647.98	1 122 482.68	1 120 851.33	1 121 781.71	1 122 482.68	1 122 482.68	1 122 482.68
CLUSTER k-median	8 085 186.01	1 284 887.71	1 284 917.94	1 284 887.71	1 290 180.59	1 284 887.71	1 284 887.71	1 289 582.76
CLUSTER h-means	1 284 887.71	1 284 887.71	1 284 917.94	1 284 887.71	1 290 180.59	1 284 887.71	1 284 887.71	1 289 582.76
CLUSTER h-median	1 477 821.51	1 310 770.57	1 477 821.51	1 353 781.05	1 477 756.67	1 477 821.51	1 283 281.80	1 477 800.43
AS 136	1 191 749.25	1 134 313.04	1 188 065.34	1 133 830.88	1 141 169.31	1 191 749.25	1 138 357.36	1 146 713.35
k-near method	9 134 241.81	1 252 644.72	9 047 273.45	1 288 109.66	1 352 292.64	8 180 439.16	1 274 644.88	1 303 178.19
k_{th} nearest method	10 792 924.07	1 422 963.44	10 749 945.37	1 388 227.67	1 513 375.10	9 532 971.92	1 217 784.75	1 298 700.15
CPU times:								
Single linkage	00 : 03 : 08	00 : 46 : 10	00 : 03 : 10	00 : 03 : 15	00 : 03 : 19	00 : 50 : 14	00 : 27 : 19	00 : 27 : 56
Complete linkage	00 : 02 : 21	00 : 08 : 24	00 : 02 : 24	00 : 02 : 27	00 : 02 : 25	00 : 26 : 14	00 : 25 : 27	00 : 28 : 47
Average linkage	00 : 02 : 11	00 : 06 : 18	00 : 02 : 13	00 : 02 : 15	00 : 02 : 19	00 : 22 : 10	00 : 25 : 31	00 : 29 : 02
Weighted average	00 : 19 : 53	00 : 41 : 40	00 : 20 : 06	00 : 20 : 03	00 : 20 : 02	00 : 49 : 40	00 : 42 : 36	00 : 48 : 48
Weighted centroid linkage	00 : 02 : 21	00 : 11 : 04	00 : 02 : 24	00 : 02 : 30	00 : 02 : 25	00 : 28 : 25	00 : 26 : 22	00 : 28 : 44
Ward's method	00 : 02 : 30	00 : 05 : 19	00 : 02 : 32	00 : 02 : 33	00 : 02 : 34	00 : 25 : 26	00 : 26 : 12	00 : 30 : 09
Stepwise k-median	00 : 31 : 16	00 : 38 : 57	00 : 31 : 24	00 : 31 : 21	00 : 31 : 26	00 : 54 : 34	00 : 56 : 47	01 : 01 : 21
Stepwise h-means	00 : 23 : 14	00 : 23 : 15	00 : 23 : 17	00 : 23 : 15	00 : 23 : 17	00 : 48 : 46	00 : 49 : 02	00 : 50 : 57
Stepwise h-median	00 : 26 : 27	00 : 27 : 13	00 : 26 : 28	00 : 26 : 32	00 : 26 : 29	00 : 53 : 24	01 : 47 : 29	00 : 53 : 46
CLUSTER k-median	03 : 03 : 02	03 : 03 : 03	03 : 03 : 04	03 : 03 : 03	03 : 03 : 05	03 : 29 : 04	03 : 26 : 29	03 : 29 : 11
CLUSTER h-means	01 : 01 : 14	01 : 01 : 15	01 : 01 : 16	01 : 01 : 15	01 : 01 : 18	01 : 27 : 58	01 : 25 : 20	01 : 27 : 59
CLUSTER h-median	01 : 10 : 22	01 : 26 : 46	01 : 10 : 23	01 : 10 : 31	01 : 10 : 24	01 : 34 : 24	01 : 34 : 11	01 : 34 : 02
AS 136	00 : 14 : 54	00 : 19 : 04	00 : 14 : 59	00 : 15 : 00	00 : 14 : 58	00 : 35 : 27	00 : 36 : 58	00 : 42 : 12
k-near method	00 : 02 : 07	00 : 28 : 16	00 : 02 : 13	00 : 02 : 12	00 : 02 : 15	00 : 51 : 09	00 : 26 : 23	00 : 28 : 20
k_{th} nearest method	00 : 02 : 12	00 : 50 : 32	00 : 02 : 15	00 : 02 : 24	00 : 02 : 28	00 : 56 : 53	00 : 26 : 10	00 : 28 : 45

Table A.39: The results of the AWCD measure and CPU times for various clustering methods using USA13509.

A.2 Variations of the hybrid density method

	k–median	h–means	h–median	CLUSTER k–median	CLUSTER h–means	CLUSTER h–median
SSR values						
Input: aggregated 70 clusters	19 043.82	15 691.68	15 829.50	156 616.79	22 815.29	23 377.69
Hybrid density variant 1	116 004.07	115 045.62	119 838.13	101 523.21	158 287.52	90 998.17
Hybrid density variant 2	144 308.58	148 912.48	147 039.02	166 921.65	167 204.83	166 618.85
Hybrid density variant 3	115 720.27	105 349.38	112 613.54	102 178.71	158 897.84	94 765.85
MSR values						
Input: aggregated 70 clusters	171.72	155.64	151.68	32.97	136.61	130.53
Hybrid density variant 1	126.10	137.76	122.27	82.69	77.34	76.81
Hybrid density variant 2	48.22	44.79	40.65	23.52	21.79	24.13
Hybrid density variant 3	123.01	120.43	128.52	113.49	77.80	81.16
PD values						
Input: aggregated 70 clusters	25.29	28.79	31.28	86.94	21.67	27.16
Hybrid density variant 1	50.35	47.65	47.90	47.58	53.19	50.89
Hybrid density variant 2	57.80	61.86	62.15	61.65	61.69	61.60
Hybrid density variant 3	47.40	47.38	46.62	49.90	72.38	50.89
ADM values						
Input: aggregated 70 clusters	241.22	172.48	175.39	1 052.53	649.88	683.51
Hybrid density variant 1	16 656.86	16 749.15	14 226.73	17 978.33	105 257.97	14 219.52
Hybrid density variant 2	83 630.09	88 929.72	86 710.08	115 739.75	116 104.04	115 362.65
Hybrid density variant 3	10 021.76	8 132.19	8 705.88	8 327.56	105 263.56	11 604.37
AWCD values						
Input: aggregated 70 clusters	12 985.98	10 772.83	10 945.75	10 894.55	15 594.50	16 099.17
Hybrid density variant 1	73 197.69	73 655.74	76 158.86	65 656.48	109 875.27	58 542.09
Hybrid density variant 2	97 492.21	101 881.18	100 636.78	115 989.36	116 240.24	115 718.23
Hybrid density variant 3	73 201.72	67 724.00	70 881.11	65 644.45	110 334.71	60 500.25
CPU times						
Hybrid density variant 1	00 : 02 : 19	00 : 03 : 57	00 : 04 : 07	00 : 32 : 10	00 : 14 : 41	00 : 15 : 12
Hybrid density variant 2	00 : 02 : 19	00 : 03 : 57	00 : 04 : 07	00 : 32 : 10	00 : 14 : 41	00 : 15 : 12
Hybrid density variant 3	00 : 02 : 19	00 : 03 : 57	00 : 04 : 07	00 : 32 : 10	00 : 14 : 41	00 : 15 : 12

Table A.40: The results for the three variants of the hybrid density method using DataS3.

A.3 Hybrid clustering methods

A.3.1 Hybrid SAHN - SAHN clustering methods - DataS3

	SSR		MSR		PD		ADM		AWCD		CPU times	
Number of clusters	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage
500	64 816.62	49 387.39	70.36	63.56	33.30	28.88	12 950.43	4 538.40	44 724.01	33 962.57	00 : 39	00 : 39
1 000	53 790.54	50 149.69	68.11	63.41	29.67	27.36	4 931.79	5 037.24	36 814.80	34 422.52	00 : 39	00 : 39
1 500	52 843.25	48 006.99	67.53	61.51	26.70	32.95	5 423.73	4 354.39	36 260.38	33 301.23	00 : 39	00 : 39
2 000	50 331.53	47 952.20	66.19	59.08	34.52	28.03	4 053.72	4 542.22	34 390.74	33 412.50	00 : 39	00 : 39
2 500	47 778.26	47 453.72	61.36	63.73	30.77	27.65	3 783.22	3 796.14	32 784.73	32 610.97	00 : 40	00 : 40
3 000	46 855.36	47 050.58	62.87	62.63	31.56	34.09	3 810.42	3 951.33	32 283.78	32 383.28	00 : 40	00 : 41
3 500	47 171.40	46 650.05	59.19	62.18	34.02	34.05	4 245.00	3 553.39	32 760.73	32 068.63	00 : 41	00 : 40
4 000	47 311.99	48 265.89	63.37	62.67	31.80	37.21	3 597.39	4 320.30	32 493.43	33 149.74	00 : 42	00 : 42
4 500	47 366.21	48 011.23	64.26	61.10	28.85	27.56	3 914.81	4 147.87	32 593.28	33 058.95	00 : 44	00 : 44
5 000	47 826.58	47 833.55	61.68	63.39	31.71	29.68	4 089.34	4 028.18	33 071.90	32 706.48	00 : 45	00 : 45
5 500	46 601.40	47 388.70	59.99	60.05	33.62	27.53	3 923.55	3 853.40	32 319.20	32 879.55	00 : 46	00 : 46
6 000	47 803.39	47 636.59	61.91	58.27	27.65	27.91	4 131.98	4 503.27	32 908.43	33 069.94	00 : 48	00 : 48
6 500	47 083.31	47 683.00	59.89	61.87	27.89	27.80	3 801.99	3 790.46	32 655.49	32 694.42	00 : 49	00 : 49
7 000	48 168.04	47 789.17	61.93	59.89	27.96	31.33	4 045.05	3 975.98	33 145.28	33 040.70	00 : 52	00 : 51
7 500	47 604.86	47 266.38	58.33	58.11	29.07	31.11	4 333.50	4 021.01	33 188.23	32 791.65	00 : 54	00 : 54

Table A.41: The results for the hybrid single linkage Ward's and complete linkage Ward's methods using DataS3.

A.3.2 Hybrid SAHN - partitioning clustering methods - DataS3

A.3.2.1 Hybrid SAHN - single linkage methods - DataS3

Number of clusters	k-means	j-means	k-median	h-means	PAM	h-median
SSR values						
500	62 807.33	65 111.44	65 972.48	61 862.74	78 213.23	61 997.34
1000	51 598.47	51 714.54	51 043.74	51 371.31	68 642.35	51 746.04
1500	47 741.75	52 598.58	51 645.22	47 741.75	70 950.91	47 429.51
2000	46 211.51	49 580.37	49 473.26	46 194.18	71 878.91	46 164.13
2500	45 844.53	50 472.71	48 723.62	45 844.86	73 234.00	45 659.93
3000	45 650.00	48 445.08	48 325.54	45 629.60	69 372.78	45 473.41
3500	45 498.88	50 209.48	48 729.65	45 497.67	72 433.68	45 410.96
4000	45 369.91	48 483.21	48 431.56	45 369.91	73 106.66	45 288.06
4500	45 356.29	47 830.38	46 616.10	45 356.35	73 500.62	45 254.74
5000	45 345.18	47 997.15	47 903.02	45 348.92	74 394.95	45 236.10
5500	45 337.92	-	46 780.12	45 335.13	-	45 222.02
6000	45 302.04	-	47 478.11	45 302.76	-	45 180.13
6500	-	-	46 301.68	45 280.64	-	44 945.97
7000	-	-	46 550.52	45 067.78	-	44 957.67
7500	-	-	46 002.37	45 055.30	-	44 953.73
MSR values						
500	64.63	67.78	67.39	65.95	78.25	65.79
1000	65.72	65.31	64.09	64.41	57.46	64.91
1500	63.39	64.58	62.19	63.39	62.35	63.33
2000	61.86	62.85	62.77	61.85	54.82	62.19
2500	61.52	64.79	62.81	61.55	56.88	61.56
3000	61.45	64.16	64.50	61.58	55.84	61.62
3500	62.07	67.22	62.15	62.06	60.51	61.46
4000	61.72	61.64	61.31	61.72	55.09	61.30
4500	61.68	64.24	61.42	61.68	74.89	61.24
5000	61.62	62.29	62.11	61.62	65.66	61.28
5500	61.62	-	59.38	61.60	-	61.31
6000	61.48	-	60.30	61.48	-	61.50
6500	-	-	61.13	61.39	-	59.90
7000	-	-	62.68	59.89	-	59.90
7500	-	-	59.70	59.91	-	59.92

Table A.42: The results of the SSR and MSR measures for the hybrid single linkage - partitioning clustering methods using DataS3.

Number of clusters	k-means	j-means	k-median	h-means	PAM	h-median
PD values						
500	29.37	28.17	29.48	30.01	34.85	29.41
1000	25.83	29.36	29.68	25.72	37.38	29.52
1500	26.67	26.72	29.16	26.67	40.03	27.84
2000	28.38	28.38	28.18	28.38	36.48	29.43
2500	28.27	29.70	29.47	28.88	37.73	29.24
3000	29.06	28.66	28.66	30.21	42.59	31.03
3500	27.35	31.18	31.22	27.35	48.01	28.48
4000	27.30	28.66	28.66	27.30	37.52	28.33
4500	27.30	30.68	34.93	27.30	47.48	28.22
5000	27.29	30.97	30.97	27.29	38.77	28.10
5500	27.30	-	31.02	27.30	-	27.96
6000	27.28	-	30.18	27.28	-	27.80
6500	-	-	31.23	27.28	-	27.34
7000	-	-	31.27	27.29	-	27.32
7500	-	-	29.92	27.28	-	27.34
ADM values						
500	13 888.60	14 212.42	15 131.62	12 660.56	19 226.64	12 685.52
1000	5 112.88	5 355.46	5 564.43	5 869.10	10 313.58	5 934.91
1500	4 152.50	5 926.36	5 675.84	4 152.50	8 977.77	4 079.26
2000	3 887.71	4 624.01	4 611.33	3 879.32	10 086.57	3 779.26
2500	3 777.17	4 392.12	4 190.78	3 804.25	10 613.41	3 715.32
3000	3 733.57	3 668.10	3 635.58	3 750.02	9 513.79	3 681.36
3500	3 704.84	4 049.32	5 049.12	3 706.21	10 350.42	3 739.04
4000	3 724.00	4 348.19	4 346.76	3 724.00	9 940.25	3 719.04
4500	3 732.07	3 897.00	3 864.17	3 731.35	9 873.90	3 715.07
5000	3 736.21	4 317.65	4 302.28	3 736.74	10 356.52	3 712.77
5500	3 734.97	-	3 884.85	3 735.97	-	3 701.28
6000	3 751.45	-	4 177.55	3 752.41	-	3 664.08
6500	-	-	3 816.79	3 753.45	-	3 419.46
7000	-	-	3 594.44	3 596.73	-	3 432.64
7500	-	-	3 493.97	3 575.99	-	3 417.45

Table A.43: The results of the PD and ADM measures for the hybrid single linkage - partitioning clustering methods using DataS3.

Number of clusters	k-means	j-means	k-median	h-means	PAM	h-median
AWCD values						
500	43 652.90	45 390.65	46 317.22	42 811.03	54 781.43	42 863.59
1000	35 358.01	35 608.60	35 204.46	35 275.54	47 397.22	35 465.92
1500	32 925.18	36 088.87	35 582.54	32 925.18	48 386.90	32 783.71
2000	31 997.60	34 362.84	34 272.87	31 983.18	49 407.72	31 927.25
2500	31 701.75	34 608.75	33 384.80	31 736.77	50 144.36	31 642.72
3000	31 583.20	33 355.17	33 291.03	31 571.04	48 133.04	31 503.36
3500	31 447.90	34 564.25	33 875.23	31 447.96	49 586.17	31 463.99
4000	31 400.65	33 438.53	33 393.57	31 400.65	50 280.10	31 409.56
4500	31 396.14	33 033.03	32 813.25	31 396.21	50 159.98	31 400.63
5000	31 398.93	33 429.32	33 367.07	31 400.07	50 681.45	31 402.69
5500	31 390.80	-	32 588.71	31 389.81	-	31 391.66
6000	31 387.23	-	32 964.28	31 387.45	-	31 360.94
6500	-	-	32 108.01	31 376.37	-	31 189.56
7000	-	-	32 269.57	31 291.45	-	31 211.12
7500	-	-	32 114.50	31 281.94	-	31 204.02
CPU times						
500	00 : 00 : 48	00 : 00 : 46	00 : 00 : 44	00 : 00 : 41	00 : 00 : 51	00 : 00 : 42
1000	00 : 01 : 48	00 : 01 : 20	00 : 00 : 53	00 : 00 : 44	00 : 02 : 11	00 : 00 : 48
1500	00 : 04 : 45	00 : 03 : 09	00 : 01 : 11	00 : 00 : 50	00 : 06 : 05	00 : 01 : 01
2000	00 : 10 : 45	00 : 07 : 03	00 : 01 : 43	00 : 00 : 57	00 : 16 : 06	00 : 01 : 27
2500	00 : 20 : 34	00 : 13 : 05	00 : 02 : 23	00 : 01 : 08	00 : 33 : 51	00 : 02 : 02
3000	00 : 36 : 56	00 : 20 : 53	00 : 02 : 25	00 : 01 : 28	01 : 04 : 15	00 : 02 : 40
3500	00 : 57 : 17	00 : 37 : 00	00 : 03 : 15	00 : 01 : 47	01 : 35 : 45	00 : 03 : 27
4000	01 : 22 : 45	00 : 56 : 02	00 : 03 : 24	00 : 02 : 05	02 : 24 : 10	00 : 04 : 19
4500	01 : 54 : 05	01 : 18 : 58	00 : 04 : 27	00 : 02 : 21	03 : 29 : 12	00 : 05 : 10
5000	02 : 37 : 23	01 : 52 : 58	00 : 04 : 30	00 : 02 : 46	04 : 41 : 22	00 : 06 : 19
5500	03 : 34 : 56	-	00 : 04 : 54	00 : 03 : 21	-	00 : 07 : 13
6000	04 : 22 : 19	-	00 : 05 : 15	00 : 03 : 31	-	00 : 08 : 38
6500	-	-	00 : 05 : 34	00 : 04 : 24	-	00 : 10 : 19
7000	-	-	00 : 05 : 18	00 : 04 : 54	-	00 : 11 : 29
7500	-	-	00 : 05 : 38	00 : 05 : 37	-	00 : 13 : 22

Table A.44: The results of the AWCD measure and CPU times for the hybrid single linkage - partitioning clustering methods using DataS3.

A.3.2.2 Hybrid SAHN - complete linkage methods - DataS3

Number of clusters	k-means	j-means	k-median	h-means	PAM	h-median
SSR values						
500	46 260.78	53 283.74	51 997.09	46 337.02	68 396.56	45 804.44
1000	45 595.25	48 650.19	48 352.25	45 667.61	72 666.49	45 321.19
1500	45 462.40	49 046.83	48 523.77	45 462.40	71 336.73	45 248.33
2000	45 589.47	46 900.38	46 726.60	45 605.80	72 375.41	45 323.00
2500	45 496.27	48 150.83	47 803.13	45 486.31	73 442.30	45 243.12
3000	45 393.99	48 034.78	47 235.09	45 394.14	71 373.24	45 313.36
3500	45 419.18	47 734.06	47 772.50	45 419.18	69 839.60	45 235.88
4000	45 374.51	47 755.96	48 458.43	45 369.01	70 284.91	45 235.26
4500	45 325.94	48 029.01	47 851.04	45 325.94	70 448.62	45 183.08
5000	45 317.95	47 622.29	47 549.69	45 317.29	72 234.38	45 172.35
5500	45 277.55	-	47 188.83	45 278.09	-	45 169.12
6000	45 297.81	-	47 146.63	45 297.81	-	45 159.82
6500	-	-	48 167.31	45 069.48	-	44 948.28
7000	-	-	47 550.51	45 072.20	-	44 955.01
7500	-	-	47 090.80	45 059.18	-	44 956.97
MSR values						
500	61.33	69.13	67.25	61.44	58.01	61.53
1000	61.12	64.19	63.54	61.27	48.97	60.77
1500	60.81	64.42	65.20	60.81	52.05	60.64
2000	61.22	61.06	60.95	61.52	59.12	61.20
2500	61.17	63.99	63.58	61.92	50.36	61.15
3000	61.81	64.06	63.38	61.80	54.79	61.33
3500	61.00	62.36	63.10	61.00	59.57	61.26
4000	61.78	62.21	63.90	61.75	65.41	61.28
4500	61.59	63.06	62.73	61.59	68.44	61.51
5000	61.57	61.83	61.70	61.57	67.33	61.54
5500	61.36	-	63.85	61.36	-	61.48
6000	61.43	-	59.27	61.43	-	61.44
6500	-	-	63.51	59.77	-	59.91
7000	-	-	61.97	59.90	-	59.91
7500	-	-	62.21	59.91	-	59.93

Table A.45: The results of the SSR and MSR measures for the hybrid complete linkage - partitioning clustering methods using DataS3.

Number of clusters	k-means	j-means	k-median	h-means	PAM	h-median
PD values						
500	27.62	30.74	29.05	27.74	38.94	29.31
1000	28.59	29.50	28.41	28.59	38.24	28.19
1500	28.34	32.18	28.07	28.34	37.74	28.49
2000	28.70	26.84	26.84	30.20	46.56	30.97
2500	29.35	31.07	31.12	27.71	38.74	31.02
3000	27.30	28.59	25.60	27.25	37.06	28.70
3500	29.39	30.89	29.89	29.39	61.08	28.64
4000	27.25	31.30	29.18	27.25	37.71	28.49
4500	27.28	31.59	30.06	27.28	37.04	28.25
5000	27.28	34.68	34.79	27.28	36.03	28.25
5500	27.29	-	28.79	27.29	-	28.07
6000	27.29	-	30.65	27.29	-	28.03
6500	-	-	31.35	27.28	-	27.34
7000	-	-	32.15	27.28	-	27.32
7500	-	-	35.39	27.28	-	27.34
ADM values						
500	4 010.94	4 655.44	4 390.85	3 984.33	9 754.10	3 782.00
1000	3 773.98	4 047.63	4 053.54	3 763.39	10 878.60	3 643.36
1500	3 735.11	3 689.37	3 507.08	3 735.11	10 430.38	3 621.72
2000	3 749.54	3 790.27	3 762.94	3 753.79	10 294.01	3 695.43
2500	3 738.12	3 841.46	3 786.93	3 720.78	11 826.63	3 686.39
3000	3 708.55	3 933.66	3 906.66	3 720.45	10 473.13	3 713.59
3500	3 736.28	4 259.91	3 699.68	3 736.28	9 284.25	3 708.75
4000	3 718.61	3 782.67	4 002.95	3 721.36	9 518.27	3 710.11
4500	3 740.73	3 999.14	4 024.02	3 740.73	7 503.49	3 647.71
5000	3 739.96	3 950.99	3 938.32	3 740.12	8 888.34	3 634.50
5500	3 765.64	-	3 699.54	3 765.78	-	3 650.90
6000	3 761.52	-	3 811.22	3 761.52	-	3 652.79
6500	-	-	4 028.65	3 579.48	-	3 423.49
7000	-	-	4 550.39	3 597.38	-	3 430.11
7500	-	-	3 894.95	3 576.86	-	3 423.72

Table A.46: The results of the PD and ADM measures for the hybrid complete linkage - partitioning clustering methods using DataS3.

Number of clusters	k-means	j-means	k-median	h-means	PAM	h-median
AWCD values						
500	32 026.99	36 278.35	35 323.96	32 049.56	46 983.76	31 721.17
1000	31 613.81	33 730.89	33 538.28	31 636.94	49 912.03	31 453.57
1500	31 540.35	33 878.61	33 115.81	31 540.35	49 084.72	31 413.97
2000	31 582.52	32 542.18	32 424.85	31 580.69	49 521.16	31 441.02
.2500	31 531.49	33 295.83	33 093.97	31 479.18	50 427.90	31 430.50
3000	31 406.42	33 345.29	32 633.24	31 421.33	49 059.05	31 430.03
3500	31 495.33	33 111.31	33 000.54	31 495.33	48 089.70	31 410.86
4000	31 397.51	33 288.37	33 404.72	31 395.55	48 025.12	31 433.72
4500	31 390.44	33 435.81	33 331.06	31 390.44	48 266.68	31 360.79
5000	31 387.68	33 121.49	33 064.61	31 387.27	49 561.76	31 366.93
5500	31 381.08	-	32 728.11	31 381.20	-	31 355.59
6000	31 387.22	-	32 533.99	31 387.22	-	31 352.68
6500	-	-	33 306.84	31 281.23	-	31 193.38
7000	-	-	33 256.93	31 294.24	-	31 209.02
7500	-	-	33 047.51	31 283.76	-	31 209.11
CPU times						
500	00 : 00 : 48	00 : 00 : 45	00 : 00 : 42	00 : 00 : 40	00 : 00 : 51	00 : 00 : 42
1000	00 : 01 : 52	00 : 01 : 19	00 : 00 : 50	00 : 00 : 43	00 : 02 : 21	00 : 00 : 48
1500	00 : 04 : 59	00 : 03 : 09	00 : 01 : 10	00 : 00 : 49	00 : 06 : 32	00 : 01 : 05
2000	00 : 10 : 35	00 : 07 : 07	00 : 01 : 50	00 : 00 : 56	00 : 17 : 25	00 : 01 : 28
2500	00 : 21 : 00	00 : 12 : 56	00 : 02 : 27	00 : 01 : 10	00 : 35 : 36	00 : 02 : 00
3000	00 : 35 : 47	00 : 22 : 22	00 : 03 : 02	00 : 01 : 25	00 : 59 : 31	00 : 02 : 34
3500	00 : 54 : 39	00 : 35 : 56	00 : 03 : 54	00 : 01 : 38	01 : 34 : 20	00 : 03 : 09
4000	01 : 20 : 15	00 : 52 : 26	00 : 04 : 18	00 : 01 : 57	02 : 25 : 40	00 : 04 : 21
4500	01 : 52 : 45	01 : 15 : 54	00 : 04 : 59	00 : 02 : 19	03 : 23 : 08	00 : 05 : 03
5000	02 : 35 : 14	01 : 48 : 57	00 : 05 : 08	00 : 02 : 51	04 : 37 : 11	00 : 06 : 35
5500	03 : 25 : 02	-	00 : 05 : 02	00 : 03 : 13	-	00 : 07 : 43
6000	04 : 19 : 51	-	00 : 05 : 11	00 : 03 : 38	-	00 : 08 : 35
6500	-	-	00 : 05 : 23	00 : 04 : 03	-	00 : 10 : 17
7000	-	-	00 : 05 : 39	00 : 05 : 01	-	00 : 11 : 28
7500	-	-	00 : 05 : 38	00 : 05 : 33	-	00 : 13 : 06

Table A.47: The results of the AWCD measure and CPU times for the hybrid complete linkage - partitioning clustering methods using DataS3.

A.3.3 Stepwise aggregation partitioning clustering methods - DataS3

	SSR		MSR		PD		ADM		AWCD		CPU times	
Number of clusters	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage	Single linkage	Complete linkage
500	44 950.19	44 973.91	59.92	59.92	44.07	44.07	3 411.29	3 435.56	31 207.49	31 247.88	00 : 56	01 : 01
1 000	44 952.93	44 952.69	59.95	59.95	44.07	44.07	3 410.02	3 408.95	31 204.16	31 203.82	01 : 43	01 : 57
1 500	44 950.84	44 952.20	59.93	59.95	44.07	44.07	3 411.29	3 408.83	31 206.01	31 203.99	02 : 27	02 : 46
2 000	44 951.32	44 952.76	59.91	59.95	44.07	44.07	3 411.73	3 409.00	31 205.76	31 203.63	03 : 28	03 : 39
2 500	44 950.98	44 952.86	59.92	59.95	44.07	44.07	3 411.72	3 409.97	31 207.90	31 204.35	04 : 37	04 : 33
3 000	44 951.72	44 950.19	59.94	59.92	44.07	44.07	3 409.72	3 411.29	31 205.15	31 207.49	05 : 34	05 : 23
3 500	44 950.19	44 951.54	59.92	59.94	44.07	44.07	3 411.29	3 408.70	31 207.49	31 204.62	06 : 02	06 : 12
4 000	44 951.54	44 950.98	59.94	59.92	44.07	44.07	3 408.70	3 411.72	31 204.62	31 207.90	07 : 10	07 : 19
4 500	44 951.32	44 951.54	59.91	59.94	44.07	44.07	3 411.73	3 408.70	31 205.76	31 204.62	07 : 57	08 : 15
5 000	44 951.54	44 950.98	59.94	59.92	44.07	44.07	3 408.70	3 411.72	31 204.62	31 207.90	09 : 07	08 : 56
5 500	44 951.32	44 952.33	59.91	59.95	44.07	44.07	3 411.73	3 409.12	31 205.76	31 205.03	09 : 54	10 : 18
6 000	44 951.32	44 951.72	59.91	59.94	44.07	44.07	3 411.73	3 409.72	31 205.76	31 205.15	10 : 41	10 : 50
6 500	44 951.32	44 951.32	59.91	59.91	44.07	44.07	3 411.73	3 411.73	31 205.76	31 205.76	11 : 55	11 : 57
7 000	44 950.19	44 950.19	59.92	59.92	44.07	44.07	3 411.29	3 411.29	31 207.49	31 207.49	12 : 34	12 : 30
7 500	44 950.19	44 951.32	59.92	59.91	44.07	44.07	3 411.29	3 411.73	31 207.49	31 205.76	13 : 23	13 : 28

Table A.48: The results for the stepwise aggregated partitioning methods using DataS3.

A.3.4 Hybrid partitioning clustering - DataS3

Aggregated method:	k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
SSR values:												
k-means	49 135.91	48 274.70	51 386.62	60 953.59	47 549.89	49 099.43	53 047.06	52 289.83	54 110.02	55 014.68	51 262.80	52 528.52
h-means	46 818.83	46 220.61	50 975.01	56 175.63	49 554.65	45 971.56	52 092.11	52 608.61	55 431.53	58 114.34	54 826.73	53 328.09
j-means	48 941.59	48 941.59	49 048.57	59 805.74	50 773.67	48 052.95	51 096.85	52 901.80	54 437.47	57 251.27	52 404.12	51 723.24
PAM	64 589.96	65 254.96	93 936.99	80 020.27	81 187.96	65 823.28	63 511.63	63 511.63	63 618.55	63 511.63	63 511.63	63 511.63
k-median	46 104.62	46 104.62	50 459.93	56 727.54	47 083.99	47 997.96	50 545.19	50 545.19	56 308.22	53 854.57	55 648.74	50 370.69
h-median	46 212.71	47 338.73	50 046.56	57 555.17	47 150.10	48 240.22	50 641.84	50 590.47	52 106.50	57 856.80	51 153.30	47 844.87
Cluster k-means	62 889.00	62 889.00	66 043.02	52 328.04	63 646.57	63 646.57	74 838.94	72 506.48	75 997.45	54 234.68	74 350.65	74 897.40
Cluster h-means	64 488.79	62 243.28	67 582.43	49 929.58	55 146.03	61 890.69	71 895.88	70 945.17	75 083.50	52 454.27	74 098.09	70 223.99
Cluster j-means	157 016.16	157 015.04	159 537.51	156 976.30	157 411.49	157 399.90	157 419.26	157 030.19	157 912.59	157 607.29	157 446.06	157 030.19
Cluster PAM	69 886.45	69 884.08	75 133.45	70 682.14	76 378.83	71 645.08	63 724.31	63 724.31	63 724.31	63 724.31	63 724.31	63 724.31
Cluster k-median	141 396.01	141 396.01	150 623.12	141 226.95	141 484.93	141 937.07	141 943.20	141 374.82	141 562.65	141 234.35	141 973.21	141 383.73
Cluster h-median	61 665.02	62 468.21	74 084.76	59 788.03	61 995.77	60 542.01	73 803.32	74 357.46	79 206.64	54 215.95	69 414.23	72 175.82
MSR values:												
k-means	65.63	63.14	64.75	61.54	59.78	65.79	67.20	67.92	68.00	53.67	68.48	69.52
h-means	63.03	62.88	64.30	60.38	61.96	61.58	67.35	70.25	69.72	59.22	66.77	68.53
j-means	64.23	64.23	61.25	58.99	67.28	63.96	63.85	64.56	68.20	64.46	64.47	64.83
PAM	56.09	60.21	55.14	44.91	51.47	52.09	75.24	75.24	75.67	75.24	75.24	75.24
k-median	61.27	61.27	63.63	54.22	61.84	62.01	63.20	63.20	65.15	58.78	66.90	63.06
h-median	59.16	60.04	69.36	62.31	61.55	63.99	60.69	60.58	62.75	63.40	61.21	63.58
Cluster k-means	60.43	60.43	61.36	65.71	64.16	64.16	64.89	59.62	55.24	66.81	56.65	60.01
Cluster h-means	66.09	70.54	63.85	61.40	66.82	65.99	55.00	61.52	55.39	66.60	50.36	58.62
Cluster j-means	46.98	47.04	44.64	47.81	45.47	45.15	47.90	44.71	48.88	49.67	44.47	44.71
Cluster PAM	54.52	54.81	53.91	54.03	54.48	52.79	74.86	74.86	74.86	74.86	74.86	74.86
Cluster k-median	46.26	46.26	45.11	46.53	50.57	42.54	44.34	44.48	41.17	48.31	43.47	43.87
Cluster h-median	65.70	66.01	71.62	66.83	66.73	67.55	61.28	61.42	58.39	65.78	62.76	66.33

Table A.49: The results of the SSR and MSR measures for the hybrid partitioning methods using DataS3.

	k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
PD values:												
k-means	28.73	28.73	32.13	38.51	31.58	29.88	29.05	23.36	29.30	33.49	24.54	28.73
h-means	27.93	28.88	32.81	33.51	30.70	31.73	28.88	29.24	34.17	37.18	28.88	27.93
j-means	31.38	31.38	30.03	40.54	31.46	28.89	27.66	31.92	29.17	36.22	31.71	31.92
PAM	51.72	51.72	51.72	57.48	58.88	51.72	49.79	49.79	49.79	49.79	49.79	49.79
k-median	32.08	32.08	29.11	33.31	27.87	27.87	27.13	27.13	32.08	33.31	32.95	27.13
h-median	30.13	28.85	34.98	37.68	31.45	32.73	29.10	29.10	33.68	33.05	32.73	29.22
Cluster k-means	25.23	25.23	27.69	31.25	25.23	25.23	27.43	27.43	28.26	31.25	31.00	27.43
Cluster h-means	24.19	24.63	34.15	30.03	26.66	25.51	27.84	27.84	28.35	30.22	31.01	27.84
Cluster j-means	45.74	45.74	48.34	45.74	45.81	45.81	45.81	45.74	45.74	45.74	45.81	45.74
Cluster PAM	39.77	39.77	44.53	39.77	46.60	39.77	39.75	39.75	39.75	39.75	39.75	39.75
Cluster k-median	43.20	43.20	47.32	43.20	43.20	43.30	43.30	43.20	43.20	43.20	43.30	43.20
Cluster h-median	25.91	25.91	28.09	38.41	25.91	25.87	28.44	28.44	28.99	32.53	25.88	28.24
ADM values:												
k-means	3 789.69	3 654.71	4 220.47	6 796.01	3 876.05	3 715.39	6 374.98	4 886.33	5 056.64	6 224.40	4 703.62	4 860.96
h-means	3 661.86	3 750.28	4 529.42	5 825.98	4 265.92	3 757.11	5 631.61	5 170.12	6 871.14	6 229.98	6 136.84	5 343.31
j-means	4 441.93	4 441.93	4 379.74	6 810.91	4 137.28	3 801.19	4 915.91	5 211.10	5 277.85	5 507.03	5 284.01	5 022.08
PAM	8 137.79	8 310.51	18 142.75	14 807.24	16 162.09	8 663.08	7 208.20	7 208.20	7 219.12	7 208.20	7 208.20	7 208.20
k-median	3 876.23	3 876.23	3 924.90	6 946.52	3 511.25	3 648.29	5 399.77	5 399.77	7 856.44	5 404.38	6 477.19	5 085.64
h-median	3 953.68	4 108.15	4 348.52	5 900.71	3 722.01	3 704.80	5 342.24	5 344.34	4 964.97	5 727.82	5 071.75	4 052.34
Cluster k-means	11 200.73	11 200.73	14 088.62	5 376.00	11 068.45	11 068.45	21 203.35	20 925.87	24 029.50	5 775.92	19 451.29	21 214.78
Cluster h-means	10 274.45	8 915.02	11 264.09	4 750.84	5 978.91	9 181.76	18 261.74	17 500.09	22 746.06	4 817.17	18 922.55	17 511.15
Cluster j-means	105 397.43	105 397.40	107 890.89	105 397.13	105 959.95	105 960.00	105 960.02	105 397.71	105 408.35	105 406.94	105 960.34	105 397.71
Cluster PAM	9 668.99	9 668.56	11 280.44	9 847.21	12 239.30	10 995.19	6 489.94	6 489.94	6 489.94	6 489.94	6 489.94	6 489.94
Cluster k-median	89 882.33	89 882.33	99 018.48	89 870.80	89 882.43	90 594.69	90 594.70	89 882.52	89 886.05	89 870.82	90 594.79	89 882.56
Cluster h-median	8 345.17	9 122.36	19 018.70	7 921.76	8 816.46	9 130.83	18 179.84	18 189.48	25 952.42	4 935.88	14 954.33	14 781.29

Table A.50: The results of the PD and ADM measures for the hybrid partitioning methods using DataS3.

AWCD values:												
	k-means	h-means	j-means	PAM	k-median	h-median	CLUSTER k-means	CLUSTER h-means	CLUSTER j-means	CLUSTER PAM	CLUSTER k-median	CLUSTER h-median
k-means	33 441.55	33 078.87	35 209.00	41 620.19	32 841.59	33 688.67	36 280.50	35 748.30	36 910.01	38 127.37	35 486.28	36 109.25
h-means	32 179.96	31 992.94	35 109.83	38 990.36	34 179.03	31 909.91	35 522.86	36 083.44	38 144.10	40 085.78	37 897.27	36 739.44
j-means	33 917.05	33 917.05	33 972.90	41 008.86	34 874.95	33 121.15	35 025.50	36 516.84	37 130.80	39 465.67	36 163.85	35 646.29
PAM	44 870.87	45 213.63	62 150.56	54 914.24	55 700.96	45 875.51	43 674.76	43 674.76	43 761.52	43 674.76	43 674.76	43 674.76
k-median	32 029.00	32 029.00	34 555.73	39 400.59	32 583.84	33 192.60	34 697.82	34 697.82	38 691.05	37 575.11	38 002.34	34 512.22
h-median	32 056.40	32 708.89	34 298.31	39 867.04	32 394.67	33 105.95	34 829.79	34 854.74	35 796.54	39 715.50	35 125.51	32 918.52
Cluster k-means	43 248.98	43 248.98	45 512.39	35 906.73	43 877.03	43 877.03	51 557.52	50 095.28	52 733.88	37 125.77	51 395.19	51 608.02
Cluster h-means	44 327.22	42 771.05	45 960.15	34 542.57	37 604.51	42 382.55	49 952.11	48 915.73	52 249.57	35 917.69	51 487.04	48 514.91
Cluster j-means	109 254.08	109 254.41	110 902.57	109 228.42	109 600.99	109 594.18	109 603.56	109 266.72	109 813.11	109 602.15	109 625.54	109 266.72
Cluster PAM	48 035.27	48 031.58	51 925.07	48 608.42	52 932.90	49 409.96	43 934.10	43 934.10	43 934.10	43 934.10	43 934.10	43 934.10
Cluster k-median	99 292.51	99 292.51	105 111.65	99 194.15	99 338.15	99 762.66	99 765.86	99 281.56	99 415.48	99 198.39	99 785.58	99 287.40
Cluster h-median	42 199.94	43 010.51	50 789.97	40 826.43	42 319.73	41 550.92	50 759.99	51 095.56	54 678.93	37 237.03	47 836.11	49 306.91
CPU times:												
k-means	06 : 03 : 09	06 : 03 : 09	06 : 03 : 10	06 : 03 : 10	06 : 03 : 10	06 : 03 : 10	06 : 03 : 12	06 : 03 : 12	06 : 03 : 12	06 : 03 : 12	06 : 03 : 12	06 : 03 : 12
h-means	00 : 03 : 58	00 : 03 : 59	00 : 03 : 59	00 : 03 : 59	00 : 03 : 59	00 : 03 : 59	00 : 04 : 01	00 : 04 : 00	00 : 04 : 01	00 : 04 : 01	00 : 04 : 01	00 : 04 : 00
j-means	01 : 00 : 12	01 : 00 : 12	01 : 00 : 12	01 : 00 : 12	01 : 00 : 11	01 : 00 : 11	01 : 00 : 14	01 : 00 : 14	01 : 00 : 14	01 : 00 : 14	01 : 00 : 14	01 : 00 : 14
PAM	06 : 47 : 08	06 : 47 : 07	06 : 47 : 07	06 : 47 : 07	06 : 47 : 08	06 : 47 : 08	06 : 47 : 10	06 : 47 : 10	06 : 47 : 10	06 : 47 : 10	06 : 47 : 10	06 : 47 : 10
k-median	00 : 02 : 21	00 : 02 : 21	00 : 02 : 21	00 : 02 : 20	00 : 02 : 21	00 : 02 : 21	00 : 02 : 23	00 : 02 : 22	00 : 02 : 24	00 : 02 : 23	00 : 02 : 23	00 : 02 : 23
h-median	00 : 04 : 09	00 : 04 : 09	00 : 04 : 09	00 : 04 : 08	00 : 04 : 09	00 : 04 : 09	00 : 04 : 10	00 : 04 : 10	00 : 04 : 10	00 : 04 : 11	00 : 04 : 10	00 : 04 : 11
Cluster k-means	14 : 56 : 52	14 : 56 : 52	14 : 56 : 52	14 : 56 : 52	14 : 56 : 52	14 : 56 : 51	14 : 56 : 56	14 : 56 : 55	14 : 56 : 56	14 : 56 : 54	14 : 56 : 56	14 : 56 : 56
Cluster h-means	00 : 37 : 47	00 : 37 : 46	00 : 37 : 46	00 : 37 : 47	00 : 37 : 46	00 : 37 : 47	00 : 37 : 50	00 : 37 : 50	00 : 37 : 51	00 : 37 : 49	00 : 37 : 50	00 : 37 : 51
Cluster j-means	00 : 59 : 50	00 : 59 : 51	00 : 59 : 51	00 : 59 : 51	00 : 59 : 51	00 : 59 : 50	00 : 59 : 59	00 : 59 : 59	01 : 00 : 00	01 : 00 : 00	00 : 59 : 59	00 : 59 : 59
Cluster PAM	08 : 17 : 08	08 : 17 : 08	08 : 17 : 08	08 : 17 : 09	08 : 17 : 09	08 : 17 : 08	08 : 17 : 11	08 : 17 : 10	08 : 17 : 11	08 : 17 : 11	08 : 17 : 10	08 : 17 : 11
Cluster k-median	01 : 42 : 21	01 : 42 : 22	01 : 42 : 21	01 : 42 : 21	01 : 42 : 21	01 : 42 : 22	01 : 42 : 30	01 : 42 : 29	01 : 42 : 30	01 : 42 : 30	01 : 42 : 30	01 : 42 : 30
Cluster h-median	00 : 40 : 45	00 : 40 : 45	00 : 40 : 46	00 : 40 : 45	00 : 40 : 45	00 : 40 : 46	00 : 40 : 49	00 : 40 : 49	00 : 40 : 50	00 : 40 : 48	00 : 40 : 49	00 : 40 : 48

Table A.51: The results of the AWCD measure and CPU times for the hybrid partitioning methods using DataS3.

Appendix B

Displaying points in Google maps

The code below uses the free GoogleAPI (2014) tool to display markers in GoogleMaps (2005). Markers can be images downloaded from another map or created and saved as .png files on the hard drive of a computer.

Copy the code and save it as a .html file in a separate folder on the computer hard drive. Create a folder called **images** in the same directory as the html file and save the markers in this folder. (For example 1.png for image 1 etc.) To execute the code, open the html file in a web browser. The html file can be edited to show different sets of points using any standard text editor. To change the points to plot, edit the customers array in the html file. To change the type of view and centre point, edit the myOptions variable in the initialize function.

```
<!DOCTYPE html>
<html>
<head>
<meta name="Customer base" content="initial-scale=1.0, user-scalable=yes" />
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>Google Maps client plots in clusters</title>
<link href="http://code.google.com/apis/maps/documentation/javascript/examples/default.css"
rel="stylesheet" type="text/css" />
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">
function initialize() {
  var myOptions = {
    zoom: 9,
    center: new google.maps.LatLng(-26.22392, 28.09368),
    mapTypeId: google.maps.MapTypeId.TERRAIN
  } /** Change the centre coordinates of the map and zoom here **/
  var map = new google.maps.Map(document.getElementById("map-canvas"),
                                myOptions);

  setMarkers(map, customers);
  showOverlays;
}
/**
```

```

* This code comes from a Google Maps JavaScript API v3 Example - Modified by J. Buhrmann 2014
* Place your data in the customers array that can easily be created in a spreadsheet
* Columns: Customer name, Latitude, Longitude, Cluster name, Image to display
*/
var customers = [
['c001', -26.2053, 28.044, 'CLUSTER_1', 1],
['c002', -26.1743, 28.1263, 'CLUSTER_2', 2],
['c003', -26.1384, 28.2148, 'CLUSTER_3', 3],
['c004', -26.205, 28.2657, 'CLUSTER_4', 4],
['c005', -26.3966, 27.8176, 'CLUSTER_5', 5]
];

function setMarkers(map, locations) {
    // Add markers to the map
    // Marker sizes are expressed as a Size of X,Y
    // where the origin of the image (0,0) is located
    // in the top left of the image.

    // Origins, anchor positions and coordinates of the marker
    // increase in the X direction to the right and in
    // the Y direction down.

    for (var i = 0; i < locations.length; i++) {
        var cust = locations[i];
        var myLatLng = new google.maps.LatLng(cust[1], cust[2]);
        var image = new google.maps.MarkerImage('images/'+ cust[4]+' .png');
        var marker = new google.maps.Marker({
            position: myLatLng,
            map: map,
            icon: image,
            title: cust[0]
        });
        attachAMessage(marker, '<font face="Arial">' + cust[0] + '<br>' + cust[1] + ", "
+ cust[2] + '<br>' + " Cluster: " + cust[3] + '</font>');
    }
}

function showOverlays() {
    if (markersArray) {
        for (i in markersArray) {
            markersArray[i].setMap(map);
        }
    }
}

function attachAMessage(marker, message) {
    var infowindow = new google.maps.InfoWindow({
        content: message
    });
    google.maps.event.addListener(marker, 'click', function() {
        infowindow.open(marker.get('map'), marker);
    });
}

google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>
<body onload="initialize()" scrollbar="yes" position:relative>
    <div id="map-canvas"></div>
</body>
</html>

```


Appendix C

Program optimisation techniques

The following techniques have been implemented in Chapters 4 – 6 to make the code of the tested clustering methods more effective and faster. Although some techniques were only useful for certain methods, like the SAHN methods, others are general improvements useful when dealing with geographical data or to save CPU times and/or storage space.

C.1 The use of arrays versus recalculations

When the same calculations are repeated in a program, it makes sense to rather store the values in arrays. An example is to store distances between all points in a distance matrix (Li et al., 2010), where the distance matrix is a $n \times n$ sized array. Using the Lance–Williams parameters for the SAHN methods is an example of how effective a distance matrix can reduce CPU times, as discussed in Section 4.3.1.8. When calculating the distances, any of the distance calculation methods discussed in Section 2.8, including Euclidean distances, can be used.

However, with further investigation, it became clear that storing and accessing multi-dimensional array elements in Visual Basic are more time consuming than recalculating values that use simple calculations, like plus or minus. More complicated calculations like square and square root, on the other hand, also took their toll on CPU times. This brought up the question whether it is faster to calculate values once and store in an array or recalculate values on the fly as needed.

A midway was chosen where multi-dimensional arrays were only used to avoid repeatedly costly calculations. Where the same array elements were accessed multiple times, for example in a for loop, the element was also assigned to a variable,

because accessing single variables were a lot less time consuming than accessing multi-dimensional array elements.

C.2 Methods to store a big distance matrix

While using a distance matrix can save CPU time, distance matrices can only be used if they are small enough to be stored in the program's memory. For a big dataset, the size can be too big and can cause memory overflow issues if it is stored in an n^2 sized array.

A number of alternatives were therefore investigated to overcome this problem:

- 1) If symmetrical distances are used, with $d_{ij} = d_{ji}$, only half of the declared array is actually needed. Furthermore, the distances between all points and themselves are assumed to be zero and do not have to be stored either. This means that when calculating a distance matrix, only the upper (or lower) triangular distance matrix (Li et al., 2010) needs to be calculated. This results in only $n(n - 1)/2 = (n^2 - n)/2$ variables of the n^2 sized array actually being necessary.

When this approach is used, a different mechanism is needed to declare the variables in the program, because an array declaration always assumes the same number of columns for all rows. Despite the technicality of how to declare the distance matrix, the program will still start running into storage issues with n values two times bigger. This approach is therefore only a temporary fix and will eventually also have memory problems with big enough datasets.

- 2) Another alternative for the SAHN methods, is to only use a distance matrix for clusters with more than one point. The distances for single point clusters can easily be calculated and do not need to be stored. An extra one dimensional array will be needed to keep track where the cluster's distances are stored. Using this approach could potentially half the size of the original distance matrix, but as is the case with the previous alternative will also not work if n is too big.
- 3) A third approach to overcome storage space is to divide the distance matrix into smaller blocks. Li et al. (2010) describes this as the "chunking" method. Even though there seem to be a limit in storage size of one array, it is possible to declare multiple arrays of the maximum size. This being said, there is however a point after which any program will declare that it has reached its storage space limit.
- 4) The final alternative is the one recommended. Instead of dividing the distance matrix, one could also divide the area where the customer points lie. The area can be divided into mutually exclusive clusters that are treated as smaller subproblems to be solved separately. Each subproblem's solution will then contribute to

the holistic solution and care needs to be taken that the final solution also adhere to all constraints.

The problem with this approach is that clusters close to the borders of the subproblems will not be able to merge and this could result in suboptimal solutions being found compared to solving the holistic problem. Care must therefore be taken to divide the problem space into obvious logical divisions. The advantage of this decomposition method is that if the division is done correctly, the complexity of the problem will be reduced and the solution will compare well to non-decomposed solutions.

Louis and Tang (1999) suggest letting the user divide the problem into subproblems with the help of visualisation to solve the TSP. They refer to this as interactive visual decomposition, which they used to enhance the performance of a GA heuristic. The problem is first plotted on a map and a user decompose the problem into logical segments where after each subproblem is solved separately. A disadvantage of this method is the dependency on the workspeed of the user to divide the problem into subproblems as well as setting up the subproblems for the program to solve. The more subproblems are required, the more manual work is involved.

C.3 Sorting methods

Clustering methods rely heavily on the comparison of distances and therefore also sorting methods. Different sorting methods exist, straightforward sorting methods with easy algorithms, like quick sort and bucket sort, have a time complexity of $O(n^2)$, while more sophisticated sorting methods, like heapsort and merge sort can take at worst $O(n \log(n))$ computation time (Cormen et al., 2009).

Because of the huge volume of distances to sort, it was decided to use the merge sort method in this thesis. It starts by going through the list and comparing two items adjacent to each other. For example items 1 and 2 and next items 3 and 4. The two adjacent items are then sorted in the correct order. The method then loops through the list again, this time comparing the ordered pairs of items with the next pair. In the above example, the pairs 1, 2 and 3, 4 are now compared. If item 1 is smaller than 3, item 2 will next be compared to items 3 and 4. If item 1 is larger than 3, item 1 will next be compared to item 4 etc., until the order of the four items have been established. With each comparison, the compared items are again placed in the correct order. The size of the number of elements to compare keeps growing until the complete list is sorted.

The sequence of the items in the distance list are recorded in another list, called a sorted list. After sorting, the clustering method can proceed with clustering by just

referring the next item in the sorted list without extra calculations. A comparison of the different sorting methods and an explanation of the merge sort method is given by Cormen et al. (2009, p 35, 150).

C.4 Keeping track of a cluster's nearest neighbours

The merge sort method works well for clustering methods with static distance lists that only needed sorting once. For dynamic distance lists, like the SAHN methods using the Lance–Williams parameters to update distances, the calculated distances in the list change over time. Here the focus change from sorting the complete list to finding the next smallest item in the list.

One way to achieve this effectively is by keeping track of the shortest distance found for every point or cluster while the distances are being calculated. This is similar to the suggestion by Murtagh (1983) to keep track of the nearest neighbours, described in Section 2.1.9. As soon as two clusters are merged, the algorithm only recalculates the newly merged cluster's distances with the other clusters. An extra check is required to see if any of the other clusters have one of the two clusters that merged listed as their closest cluster. If this is the case the cluster's nearest neighbour needs to be calculated again. When deciding which distance is the next smallest distance overall, the program only has to loop through the list of nearest neighbours per point or cluster and return the smallest value.

C.5 Using coordinates instead of distance calculations for comparisons

Instead of using costly calculations to determine and compare distances to find the nearest neighbours, this technique tries to eliminate unlikely candidates beforehand. This is done by comparing the absolute difference in coordinates to those of the current nearest neighbour. If the absolute differences of one of the coordinates are smaller than the current best, the distances are calculated and compared as usual to determine if this is a closest neighbour.

In the case of Euclidean decimal coordinates, the absolute differences are calculated as $\|a_i^1 - b_i^1\|$ for the longitude values and $\|a_i^2 - b_i^2\|$ for the latitude values between the customer $\mathbf{a}_i = (a_i^1, a_i^2)$ and the point being considered $\mathbf{b}_i = (b_i^1, b_i^2)$.

C.6 Identifying adjacent clusters

In order to find the nearest neighbour for a cluster, Grimm (1987) suggested only considering the inter-cluster distances of adjacent clusters. Unfortunately, determining adjacent clusters is not an easy task when dealing with discrete geographical data, because multiple clusters could be close to different points in the cluster and clusters may even overlap. To determine adjacent clusters, the following methods were investigated:

- The most straightforward method to determine adjacent clusters is to find, for every point in a given cluster, the closest point outside this cluster. The adjacent clusters are then the clusters to which these points belong.

This method can be quite time consuming and cannot guarantee an inclusive list of all adjacent clusters. If, for example, there is an outlier point in a given cluster, the point can be adjacent to multiple clusters. Using the method above, only one of these clusters will be listed. The other clusters might not necessarily be the closest cluster to any of the other points in the cluster either. In this case these clusters will not be in adjacent cluster list.

- Another method to find adjacent clusters is to divide the geographical data into smaller blocks, similar to a suggestion made by Murtagh (1983). If a block is centred around the cluster in question, then all clusters in this block can be considered adjacent. This method cannot guarantee that all adjacent clusters will be listed either, because it depends on the size of the block. If the cluster is too big for the block, adjacent clusters can fall outside the border. If the block size is too big, some non-adjacent clusters can also be included. A lot of time and effort can also be spent on determining the correct size of the blocks with no guarantee that all adjacent clusters are listed accurately.
- A third method, that uses the distance to the closest cluster, was therefore used. This method calculates the distances from a given cluster's centroid to the centroids of all other clusters. The distance to the closest centroid is then identified and if any given point from another cluster is closer than this distance to a point in the given cluster, the cluster to which this point belongs is listed as an adjacent cluster. This method works well for non-overlapping clusters.

C.7 Arrays versus text string variables

During the coding of the graph-based clustering method another optimisation technique was used. To make the methods faster, it was important to keep track of the

points connected to each point, that represented a constructed graph. An array per point listing all the connected points would take up too much memory, because a big enough storage space needs to be reserved even though the size could vary per point and is unknown beforehand. The speed to access such big arrays also took too long.

It was then decided to associate a text string with each point. The text string contained all point numbers that were connected to the point, separated with semicolon characters. When needed the text string was parsed inside a loop to find all the connected customers. This method proved to be much faster than looping through all points to find the connected ones, but became quite sluggish when dealing with strings with many connecting points.

C.8 Linked lists

Linked lists, described in Cormen et al. (2009, p 236), are an effective method to store ordered lists as an alternative to using multi-dimensional arrays. These lists can keep track of the sorting order of a list of points. It can also be used to keep track of all points assigned to a cluster or the sequence of a number of points, for example when constructing routes. Here, each element in the list points to the next element in the list, effectively creating a path through the data. Linked lists are therefore ideal to prevent looping through large numbers of data to identify all the points in a cluster. In this thesis it was also used in the MCA method to construct routes, by storing the sequence of customers to be visited.

A one dimensional array is used to store the next point associated to each point in the list. If the points are divided into mutually exclusive sets, as is the case when creating route clusters for the single-source CLRP, the sequences of the different sets (routes) can be stored in a single one dimensional array. When using sorted lists it is important to be able to identify the start and endpoints of a list to ensure the list can be read correctly. To determine the first point in a linked list, a separate array can be used. If the list can be given the same index number as the first element in the list, then finding the first element is trivial. The last point in a cluster has a value of zero, which means there are no more points in the cluster. If the number of elements in a list must be known, it needs to be stored in a separate array.

C.9 Summary

In this Chapter, programming techniques that were useful when dealing with geographical data and to save CPU times and/or storage space, are discussed. The

highlights are:

- The use of interactive visual decomposition to divide of a problem into sub-problems when dealing with too large distance matrices, (Louis and Tang, 1999).
- A method is proposed to identify adjacent clusters for discrete datasets.
- The use of the absolute difference of coordinates as a quick estimate when comparing distances between geographical points, instead of calculating the actual distance.
- The use of linked lists to construct route sequences, that can all be stored in a single one dimensional array if the problem is single-source.