

https://dspace.org/

Software modules to further assist the rapid production of optimised electrical reticulation schemes

Rajakanthan, Thurairajah

2014-03-25
http://hdl.handle.net10539/14323

Downloaded from a DSpace repository. https://dspace.org/

SOFTWARE MODULES TO FURTHER ASSIST THE RAPID PRODUCTION OF OPTIMISED ELECTRICAL RETICULATION SCHEMES

Thurairajah Rajakanthan

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1999

DECLARATION

This project is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg.

I declare that this dissertation is my own unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

J. Kjaltth

(Signature of candidate)

9th day of April 1999

ABSTRACT

To assist with the mass rural electrification drive being undertaken in South Africa, special CAD based software is being developed by the Software for Electrical Distribution (SED) Group at the University of the Witwatersrand. Various optimisation modules have been previously developed by the SED group that relieve the designer of many routine tasks, thus facilitating more creative decisions. This dissertation describes a research project in which two further aspects were investigated.

1. In order to use many advanced optimisation techniques, a specific form of "intelligent" map is required that will enable the software to identify relevant features on the map of the township. A software module that can create an "intelligent" map with minimal operator intervention from conventional CAD sources is presented. Only the conceptual overview and the high level design of this software module is discussed.
2. Determining optimum distribution transformer locations and their service areas (transformer zones) is still a time consuming activity. Delineation of optimum transformer zones minimises conductor lengths and improves transformer utilisation. The software described in this dissertation, delineates suitable transformer zones rapidly thus permitting various alternative strategies to be speedily investigated. The program was implemented and has been tested on maps of actual townships. Although the proposed algorithm performed well it was found that the final layouts required minor manual adjustment. Reasons for this are discussed.

Recommendations regarding the further development of certain aspects of the software are provided.

To My Family and Friends For Their Love And Support

ACKNOWLEDGEMENTS

AS Meyer, for his supervision, encouragement and patience.

Professor B Dwolatzky, for his assistance in various aspects of the software design and with the advice in compilation of the paper model dissertation.

Professor Walker, for his guidance on managing the project using SEAL QMS.

This research was partially funded by the following parties:
Tertiary Education Support Program of Eskom (TESP),
Technology and Human Resources Industrial Program of the Department of Trade and Industry (THRIP).

Members of the SEAL Laboratory, past and present, for their advice and help.

TABLE OF CONTENTS

	Page Number
Declaration	ii
Abstract	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
Foreword	ix
Project Introduction	1
MSc Paper I	5
MSc Paper II	11
MSc Paper III	15
Conclusions and Recommendations	21
References	24
Bibliography	31

APPENDICES

Document Number

Appendix A: "Intelligent" Map Creation

Problem Statement and Specifications: "Intelligent" Maps A1

Literature Survey: "Intelligent" Maps A2

Conceptual Design: "Intelligent" Maps A3

Appendix B: Transformer Zoning Program

Problem Statement and Specifications : Transformer Zoning B1

Literature Survey: Transformer Zoning B2

Conceptual Design: Transformer Zoning B3

High Level Software Design B4

Low Level Software Design B5

Testing and Results B6

Appendix C: DXF to MAP file converter

User Reference Manual C1

High Level Software Design C2

Low Level Software Design C3

Appendix D: "Intelligent" Map creation program

Abstract of the project undertaken by final year honours students D1

Appendix E: Management Products

Master Document List (MDL) ..... TB 001

Configuration Management Plan ..... TB 006

FOREWORD

The SEAL (Software Engineering Applications Laboratory) Quality Management System (QMS) is an ISO 9000 conforming system for the production of quality assured software engineering projects. This project is one of a series of projects which have used, or are using, this quality system. The QMS system involves the maintenance of working documents that reflect the work under progress which continually evolve over the project lifetime.

The format for this Masters Dissertation differs from that of the standard dissertation. The format used is the *paper model* which consists of a short body and multiple appendices. The body comprises the Introduction, technical papers, Conclusions, Recommendations, References and Bibliography sections. The technical papers along with the supporting documents in the body provides an overview of the work done and highlights important knowledge that has been gained. The appendices are the supporting documentation which provides the reader with comprehensive details of the research work performed, thus allowing the examination of the various aspects in greater detail. Each document in the appendices are self contained, in that they each have a table of contents, introduction and conclusions being the manifestation of continuos work from the start of the project. They should not be considered as chapters of a conventional thesis were the chapters flow smoothly from one to another. However, there is logical flow in the documentation so that the material presented is coherent to the reader.

Thus this dissertation does not contain all the documentation that has been produced during the project lifetime. Rather, only the most appropriate documents are presented here in the respective appendices. The management documents lists all the documents that were created during the lifetime of this project and are included in Appendix E.

The rest of this forward provides the reader with some guidance concerning the various documentation that is presented. The documents are discussed in the order that they appear in the table of contents. This dissertation, as the title suggests, involves the research into two aspects that will further aid the SED (Software for Electrical Distribution) group's goals. The goal of the SED group is to develop and integrate various software modules (as developed in this research) into a complete package called ASED (Automated Software for Electrical Distribution). It is hoped that ASED will significantly aid the design of optimised electrical distribution schemes and do so in much shorter time.

As mentioned above, the technical papers encompass the whole project, but in doing so is unable to concentrate on any particular aspect in as much detail as the appendices. It should be noted that the papers in the published format lists the names of the supervisors as co-authors. They were being given

credit for their guidance in the research work that was done. The nature and the contents of the papers are summarised below:

Paper 1: This paper was published in the *IEEE Computer Applications in Power*, Jan 1998. It initially describes the concept and purpose of an "intelligent" map and then proceeds to describe how a specific form of an "intelligent" map, as required by ASED, can be created. Only the important underlying algorithms are presented.

Paper 2: Published for the *South African Universities Power Engineering Conference* (SAUPEC), Jan 1998. This paper provides an overview of the thesis, briefly describing the concepts of "intelligent maps" and transformer zoning.

Paper 3: This paper provides a complete overview of the primary algorithms implemented in the transformer zoning program. Although some primary algorithms are discussed, the description of the software architecture is somewhat neglected for the reason that the paper was submitted to a power engineering transactions and not to a journal that is concerned with software development. It is important to stress, however, that equal effort has been put into both the design of the application as well as its implementation.

A summary of the conclusions and recommendations is presented. (Detailed conclusions and recommendations can be found in the documents A3 and B6 of the appendices.) This is followed by the *References* and *Bibliography* section, which contains the reference list and bibliography for *all* the documentation presented in the introduction, conclusions, recommendations and the appendices. This naturally excludes the references for the technical papers which are self contained.

The appendices are divided into three self contained sections, one for each aspect researched. Each of these documents will be found to have a few additional pages which are used for management purposes. SEAL QMS requires certain conventions, such as official project titles. This project is titled by the acronym "ASED". The title is over representative of the work done, but it is merely being used to recognise the fact that the work done forms part of the bigger project being undertaken, namely ASED.

Appendix A contains all the relevant documentation describing the design of the "intelligent" map creation program. Appendix A comprises three sub-sections and they are briefly discussed:

1. *Problem Statement and Specifications.* The motivation for this aspect of the research is presented by discussing the problems associated with the manual procedures followed by the derivation of a list of suitable functional specifications.
2. *Literature Survey.* All the research conducted regarding existing means of solving the problem and their shortcomings are presented.
3. *Conceptual Design.* Based on the literature survey and careful consideration a suitable solution is formulated taking alternatives into consideration. The conclusions and recommendations for this part of the research is also included in this document.

The solution talks about employing a file conversion program (DXF-MAP file converter) and since it was developed and documented separately by the author, it is listed in Appendix C. Although, this is not part of the research project, it has been included in an appendix for the sake of completeness. In addition, the implementation for the software module that would create "intelligent" maps was done by final year honours student in electrical engineering and the abstract is provided in Appendix D.

Appendix B contains all the relevant documentation describing the transformer zoning program. This is complete as all aspects from conception to the final delivery of a fully operational software product is discussed. These documents relate to the various phases of the software engineering life cycle.

Each appendix is further divided into sub-sections and they are briefly discussed:

1. Problem Statement and Specifications. The motivation for this aspect of the research is presented by discussing the problems associated with the manual procedures followed by the derivation of a list of suitable functional specifications.
2. Literature Survey. All the research conducted regarding existing means of solving the problem and their shortcomings are presented.
3. Conceptual Design. Based on the literature survey and careful consideration a suitable solution is formulated taking alternatives into consideration.
4. High Level Software Design. An overview is provided of the software design schematic and architecture that would be employed to solve the problem. Design methodology plays a key role in the evolution of a good software system and is thus described.
5. Low Level Design. This document follows from the High Level Design and it focuses on the low level implementation. It can be considered as delving into the "nuts and bolts" of the system.
6. Testing and Results. All the testing, results and associated discussions are presented in this document. Based on the results conclusions are drawn and recommendations are proposed.

Appendix C contains all the relevant documentation describing the DXF to MAP file converter program. This is complete as all aspects from conception to the final delivery of a fully operational software product is discussed.

1. User Reference Manual: This document was drawn up not only to assist users in using the this program but also used as a medium for defining the functional specifications.
2. High Level Software Design. An overview is provided of the software design schematic and architecture that would be employed to solve the problem. Design methodology plays a key role in the evolution of a good software system and is thus described.

3. Low Level Design. This document follows from the High Level Design and it focuses on the low level implementation. It can be considered as delving into the "nuts and bolts" of the system.

It can be noticed that documentation format is different to that of the former appendices. These documents were complied using an older software documentation system which was designed for procedural programming. The documentation styles adopted for appendices A and B are more suited towards the use of object oriented programming. Regardless of the different styles in software documentation, all relevant information has been adequately captured.

Appendix D is a very short section which contains the abstract from the final year thesis. The reason for listing it as a separate appendix is that it was not implemented as part of the this research. Thus, this is the only *section* that was not performed by the author.

Appendix E contains two of the management products for this project. The MDL document lists all the documentation that was created throughout the project lifetime. The configuration management plan describes the type of files created and the file naming conventions utlised. Management information such as data residence and backup procedures are also described.

PROJECT INTRODUCTION

Background

It is a well known fact that the majority of the population in South Africa do not have electricity. In the past, only the affluent community were provided with electricity and these areas tended to be in well established urban and industrial areas. The provision of electricity to those people, who are generally located in informal settlements scattered around the country, will uplift the living standards due to the obvious benefits it provides of which some are listed below [Ranganathan: 1992]:-

- Rural electrification improves the quality of life by facilitating, lighting, cooking and heating.
- Education is enhanced by the ability to increase pupils' studying time.
- Shop keepers are able to extend their trading times and new electricity dependent industries can develop.
- Health facilities are improved, since medicines can be stored in refrigerators.
- Electricity is also a catalyst to the development of the economy provided that other factors such as fertile soils, water supply, good communications infrastructure, market and credit facilities are present.

In the past, electrical infrastructures were typically built over a long period of time not only in South Africa but also in other developed countries. The situation being faced now is that the majority of the population must be reticulated in a relatively short period of time. Thus the number of rural electrification projects that have to be implemented has increased by an order of magnitude. Eskom, the national distributor, commenced a massive low cost electrification drive in the early 1990's.

Contrary to initial predictions the schemes being built were found to be more costly. It was originally assumed that the cost per connection would be about R2,000 which would decrease as time progressed while the energy consumption would reach 350kWh per month [Dwolatzky: 1998]. In practise it was found that the cost per connection was at least R4,000 to R5,000 and the energy consumption averaged about 80kWh per month [Stephen: 1998]. Furthermore, recently connected consumers were averaging about 30kWh per month. Continuation of the current reticulation practises will thus result in a gap of R480 million between the target and actual network costs [Stephen: 1998].

In undertaking such a large scale venture, some important problems have been identified and are now described:-

1. The majority of the consumers being connected earn minimum wages and therefore their energy consumption is low [Gaunt: 1998]. The availability of cheaper alternative energy sources (wood, dung, candles, paraffin) discourages the use of electricity. Rural electrification schemes are difficult

to justify financially since the revenue from unit sales are insufficient to outlay the capital costs unlike with schemes in urban areas. Thus the time taken to recover the electrical infrastructure costs is excessive. Therefore, distribution schemes must be optimised so that a given network optimally satisfies the demand of the consumers being serviced. Networks that are designed to provide optimum energy are inevitably cheaper as they utilise smaller cables and significantly less if less phases are utilised [Stephen: 1998]. (It should be noted that cables contribute to a significant portion of a distribution network cost.) Furthermore, optimally designed distribution networks have lower running costs due to the reduction in no loads losses of the transformer during off-peak periods.

2. As mentioned before, the electrification drive requires connections to be made in a short period of time. Designing electrification schemes manually is a tedious and time consuming process as many of the calculation that have to be performed are either done by hand, transferred into a spreadsheet or into other non-integrated tools. The need for repetitive calculations makes it difficult for a designer to change and optimise the design as revisions takes time which increases the design costs. Thus, the more time that is spent on achieving the most cost effective design the higher the design costs.
3. Skilled manpower is required to design effective low cost electrical distribution schemes. Not many techniques or skills have been developed in the field of low cost electrification since the need was virtually non-existent in most developed countries.

However, the political situation in South Africa brings about uncertainties that must be recognised [Gaunt: 1998]. Two extreme scenarios exist and they are:-

1. People located in rural areas may tend to migrate to the cities where there are jobs opportunities.
2. On the other hand, they may continue to stay in the rural areas resulting in their economic development.

There will obviously be many other scenarios that are located between these two extremes, but at this point it is not possible to predict the trend. A further uncertainty is the ever increasing threat of AIDS. The only certainty is that the mass electrification program will proceed well into the foreseeable future. Therefore, networks must be carefully planned to meet the customer demand with provision for adequate expansion with minimal costs [Gaunt: 1998]

Aims an objective of this research group

In the past, various software tools have been developed to assist distribution planning by the Software for Electrical Distribution (SED) research group at the Department of Electrical Engineering, University of the Witwatersrand [Meyer: 1991, 1996, Dwolatzky: 1998]. To further aid the current national electrification drive, a set of integrated software tools, called ASED, is being developed. ASED incorporates various optimisation modules, developed by

previous MSc students, to automate and optimise as much as possible various aspects of the design process. This includes the, determination of non-domestic loads [Patricious: 1996, 1997], optimum cable routes [West: 1996a, 1996b, 1997], optimum junction positions [Apostolellis: 1996a, 1996b], optimum transformer-junction cable connectivity [Tumazos: 1996, 1997], and calculation of voltage drops and selection of cable sizes accounting for the statistical consumer load distribution [Nicolson: 1993].

Thus the end goal of ASED is firstly to enable a design of an electrification scheme to be automatically generated, with sufficient accuracy, for initial project assessment. The cost benefit is that such a design could be created within a matter of hours as opposed to matter of days or weeks as with conventional manual design. Secondly, at a later stage a senior engineer would use this design to make appropriate changes after a site visit and all dependent factors are re-computed thus updating the design. Lastly, at the end of the design process, a complete database is available from which the full bill of materials can be automatically generated.

Aims and objectives of this research

Two additional aspects were identified as processes that needed to be streamlined in order to further alleviate design time and they are now described below.

Creation of "Intelligent" maps

For manual computer based distribution design a township map in the form of a CAD file, must be available at the start of the project. However, any tool aimed at automated facilities design requires an "intelligent" map which implies a non-graphic database linked to the CAD file. The database provides appropriate information about the features represented on the map. Thus an intelligent map enables the optimisation software to perceive features on a map in the same the way as a human. Creating an intelligent map manually would be a time consuming task defeating the purpose of the time saved by the optimisation routines. ASED requires a specific form of "intelligent" map as defined in various specification documents relating to the MSc projects already completed. The first section of this dissertation describes a module for rapidly creating an intelligent map from various types of topographical sources. All the algorithms for this module were designed and described of which some were implemented as part of an undergraduate thesis project.

Transformer zoning

As is customary with distribution design, the designer then starts by delineating service areas or transformer zones which involves the designation of every consumer to be supplied by a specific transformer. This is done by drawing a boundary around a group of consumers to be supplied by a single

transformer using visual inspection. Thus the entire area of the township is broken up into transformer zones. The designer now counts the stands within each zone and if any have more than the transformer's capacity then they are assigned to adjacent zones as deemed appropriate and vice versa. Clearly for large townships, this is a tedious and time consuming task. Once the zones have been finalised the transformers are located at a suitable position near the load centre of gravity of each zone (again determined by visual inspection). This methodology may be adequate for small areas but the degree of complexity increase substantially when considering areas that require five or more transformers. Therefore locations identified manually not only would have taken a great deal of time but may also be far from optimum.

In addition, optimum delineation of transformer zones minimises conductor lengths and improves transformer utilisation, thus further reducing capital and running costs. As the size of the township increases, the time taken to delineate zones increases exponentially. These reasons motivated the investigation of techniques that can be used to efficiently determine transformer zones. In order for this module to function, it too will require an "intelligent" map.

The latter aspect is covered in detail discussing both design and implementation issues. Documents entailing the high and low level design are in the appendices. Extensive testing was done and the results are also discussed at length in the appendices.

Important Terminology

The terminology used in America and European communities varies significantly from those that are used locally. For the reason that that most of the papers submitted to journals were based in the United states of America, different terminology was used. Wherever appropriate, all care has been taken to ensure that the terminology can be understood. To avoid confusion some of the more commonly used terminology is defined in this section.

The term *land parcel*, *lot*, *stand* and *erf* mean the same and refer to the end consumers premises to which customers are being electrically fed. The terms lot and land parcel are used extensively in the first and third paper which was targeted at an American audience. The second paper and all the appendices use the term stand, which is more oriented towards South African audience.

SMART MAPS STREAMLINE DISTRIBUTION DESIGN

T Rajakanthan

Department of Electrical Engineering
University of the Witwatersrand, Pvt Bag 3, PO Wits 2050, Johannesburg, SA

As part of a research program to aid the national electrification drive underway in South Africa, a set of integrated software tools, called ASED, is being developed to automate the electrification design process. In order for these tools to be able to perform any optimisation, a specific form of "intelligent" map is required that will enable the software to determine significant features on the township map. For the different optimising routines special features such as identifiable cost regions as well as the usual map features of lot and road boundaries are required. Manually defining these for the computer would be a tedious and time consuming task, defeating the time and money saved by ASED. This paper presents a software module that can create an "intelligent" map with minimal operator intervention from conventional CAD sources for use by ASED. The operation of the modules discussed here have been verified and are being implemented in C++ using object oriented techniques as part of ASED.

In the third world, the need for low cost electrification to uplift the living standards of the community is a major concern. In South Africa, the majority of the population are at present without electricity and in many instances live in informal settlements. Major strides are underway to supply electricity to these people in a short period of time. Designing electrification schemes manually is time consuming and little or no optimisation is done due to time constraints. Furthermore, skilled manpower is required to design low cost and effective electrical distribution schemes. Thus, the more time that is spent on achieving the most cost effective design the higher the design costs.

To aid electrification projects a set of integrated software tools, called ASED, is being developed by a research group in the Department of Electrical Engineering, University of the Witwatersrand. It incorporates various optimisation modules, developed by MSc students, to automate and optimise as much as possible the design of electrification schemes. This includes the determination of optimum transformer positions, cable routes and cable sizes taking into account the statistical consumer load distribution. Thus the end goal of ASED is firstly to enable a design of an electrification scheme to be automatically generated, with sufficient accuracy, for initial project assessment. The cost benefit is that such a design could be created within a matter of hours as opposed to matter of days or weeks as with manual design. Secondly, at a later stage a senior engineer would use this design to control his/her design team so that any changes necessary can be easily incorporated into the design. While computer software will never be able to completely automate the design process, such a tool will prove valuable if it is convenient to use and ensures that the degree of user intervention is low.

For manual computer based distribution design a township map in the form of a CAD file, must be available at the start of the project. However, any tool aimed at automated facilities design requires an "intelligent" map which implies a non-graphic database linked to the CAD file. The database provides appropriate information about the features represented on the map. Thus an intelligent map enables the optimisation software to perceive features on a map in the same way as a human. Creating an intelligent map manually would be a time consuming task defeating the purpose of the time saved by the optimisation routines. This article describes a module for rapidly creating an intelligent map from CAD, GIS, hardcopy or aerial photographic sources.

A specific form of an intelligent map is required by the ASED optimisation routines. The relevant features, after defining, are filed in a text file (*.map) and are as follows:

1. The co-ordinates of the vertices that make up road or lot boundaries must all be listed. Text labels for all lots must be defined since their location is assumed to coincide with the termination point for the consumer service cable.
2. In order for the cable router to function cost regions must be defined. For this purpose the townplan must be divided into non-overlapping cost regions and assigned a cost multiplier that defines the degree of difficulty in erecting a line over that region. Therefore, an additional list, similar to the one for lots and roads, is required for the cost region. In this case the label will be a number representing the cost multiplier.

The process of intelligent map creation is shown in the following flowchart

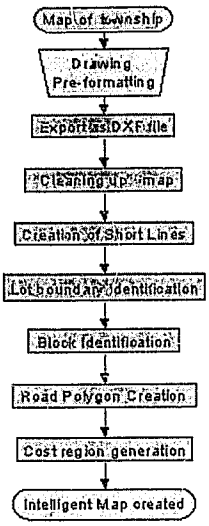


Figure 1 - Flowchart of Intelligent Map creation

The following sections will describe each of the processes in detail.

INITIAL MAP PROCEDURES

Map of Township. If the maps were drawn specifically for ASED then the relevant features would appear in the required format. In the general case where the maps have been produced by a third party, pre-formatting may be required as explained in the following section. In ASED any form of CAD package can be used provided that they can export drawing files in AutoCAD™ DXF format.

Drawing Pre-formatting. The method used here is to segregate those groups of features necessary only for creating an intelligent map. Each individual group of features is located on a unique layer in the drawing file. While good drawing practise often stipulates that features with common properties are grouped together on the same layer, many of the maps supplied have many different features, most of which are not required to create an intelligent map, located on the same layer. This makes it difficult for computer software to extract the relevant information as it has no way of distinguishing between the various features. In such cases, the different features of interest represented on the map have to be grouped and manually moved on the screen to other layers using CAD commands. The relevant features and the required procedures are described below:

- Using standard CAD commands the layer containing the lot boundaries has to be identified and labelled "LOT". On this layer of the town plan, lines that are incorrectly delineating lots must be deleted or moved to a different layer.
- The layer containing the lot text labels must be named "LOTNAME". If none of these labels exist the lots must be identified by placing text labels, located approximately at the centre of each lot. The location of these labels will be used later in identifying their respective lot boundaries.
- The designer is also expected to define the township boundary as a polygon on a layer which is named "BOUNDARY". This boundary will be used in creating the roads and cost regions.

On completion of these steps, the map of the township is exported from its native format to an AutoCAD™ DXF format. DXF is an internationally accepted standard for exchanging drawing information. Since DXF files are in text (ASCII) format, the following required lists are extracted:

- ListX- lines from the layer "LOT".
- ListY- lot labelling text and the co-ordinates of the node from the layer "LOTNAME".
- ListZ- vertices of the township boundary from the layer "BOUNDARY".

"Cleaning up" Maps. Digitised maps as drawn vary in quality which may imply one or both of the following defects (as shown by circles in figure 2):

- Lines drawn may not terminate exactly on other lines as intended but fall short or overshoot them. Either of these conditions will not allow computer software to correctly determine the connectivity between the various lines.
- Several vertices may be located within close range of each other near the same intended location. Therefore, there may be more than one line representing the same feature.

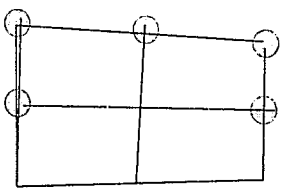


Figure 2

The following cleaning up operations are performed.

- Any two endpoints apart by less than a distance r (which has a default value of 1m but can be altered by the user) are assumed to be identical.
- Any line that falls short or overshoots a line by a distance less than r is adjusted to terminate on that line.
- All duplicate lines are eliminated.

Algorithms are developed for each of these steps to clean up digitised maps and these routines are executed (in the order that they appear). For this operation ListX is required.

Multiple-Endpoint eliminating algorithm. This routine calculates the distance between each of the co-ordinates of the line endpoints and every other line endpoint. If any of the distances are less than r, then those endpoints are merged into one common point.

Intercept Correcting Algorithm. In the digitised map, even those lines that initially were correctly terminating on other

lines, may now not terminate correctly as a result of the multi-endpoint eliminating algorithm. A new list (ListA) consisting of a copy of all the lines that do not have one of their endpoints coinciding with another is now created. (Clearly, ListA is a subset of ListX.)

Each line in ListA is checked with every other line in ListX to determine any intercepts. Intercepts are assumed if the point of intersection, after the linear equations are solved, is between the endpoints of the line. The intercept point will replace the endpoint in ListX if it is less than a distance r from the endpoint. Every time a line is corrected, it is deleted from ListA.

The remaining lines in ListA, are therefore the lines that fall short and are now extended by a distance of r. The process, as described above, is once more used to determine any intercepts that may now exist and the overshoots are corrected. All the corrected lines are again deleted from ListA. If there are lines still remaining in ListA then these special cases are brought to the attention of the user for manual correction. At the end of this operation, ListX will contain only correct lines.

Duplicate Line Eliminating Algorithm. Although ListX only contains correct lines, it may contain two or more identical lines delineating the same boundary. The duplicate line eliminating algorithm will compare each line with every other line and if any matches are found then only one line is retained.

Creation of Short lines. Execution of the short line algorithm will break up all the long lines into short ones so that each of those define a section of the boundary for not more than two adjacent lots.

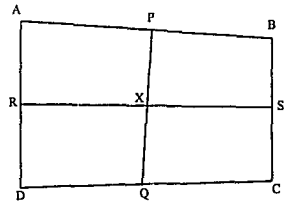


Figure 3

As an example, in drawing the block of lots (figure 3) *ABCD* was first drawn as a polygon and *PQ* and *RS* as long lines. These long lines and the polygon need to be segmented as necessary for the lot boundary recognition algorithm. Therefore intercept points *P*, *Q*, *R*, *S* and *X* must be inserted and all the long lines segmented accordingly.

The procedure for this is as follows. Each line in ListX is checked with every other line for intercepts. If an intercept is found that coincides with the endpoints on a line then that intercept is ignored. When a given line is found to have n intercepts between the endpoints then the long line is segmented into $n+1$ short lines and are added to a new list (ListB). If for a given line $n=0$ then that line is added to ListB, without segmentation. Thus ListB will contain complete list of short boundary lines. The flow chart for this operation is given below.

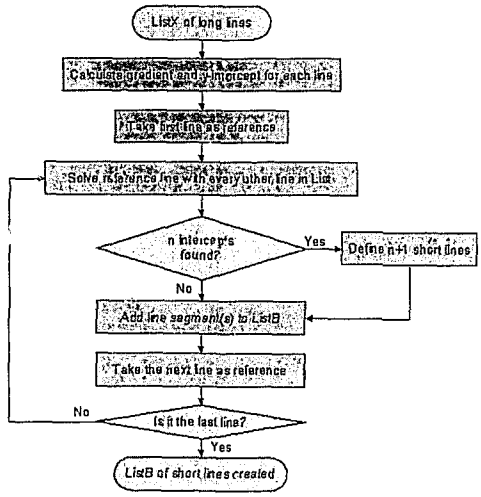


Figure 4 - Short Line Creation Algorithm

FEATURE RECOGNITION FOR THE OPTIMISATION ROUTINES

Lot Boundary Identification. This routine will identify lot boundaries given the consumer cable termination point which is the location of the lot labelling text node. Thus, this routine utilises ListB (list of short lines) and ListY (list of lot labels).

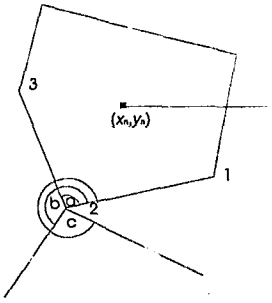


Figure 5

From each lot labelling text node $x_n y_n$ (figure 5) the equation of a horizontal line ($y=x_n$) is defined. Calculations are performed to determine intercepts of line $y=x_n$ with lines in ListB. The intercept closest to $x_n y_n$ and having a x-co-ordinate greater than x_n is found. This intercept is on a line that forms part of the boundary. The endpoint of this line with the lower y value is now selected. In the example in figure 5, vertex 1 has now been identified. In this instance the boundary lines will be identified in the clockwise direction.

Having identified a vertex (vertex 1), the program will look for other lines coincident at that endpoint. If only one such line is found then that line is another boundary line. The other endpoint of the new line is used in the same fashion to determine subsequent lines. If more than one such line is found at an endpoint (as in the case in figure 5, vertex 2) then the line subtending the smallest angle with the original line will be chosen as this line is part of the boundary. These steps are repeated until the program once more finds the first vertex. The boundary lines for that lot are then identified.

An output list (ListC) will be produced with a list containing the following information for each lot:

- the lot label,
- the co-ordinates of its text node and
- a list of vertices that define the lot boundary.

The information in ListC will be written to the MAP file. The flow chart for this algorithm is given below.

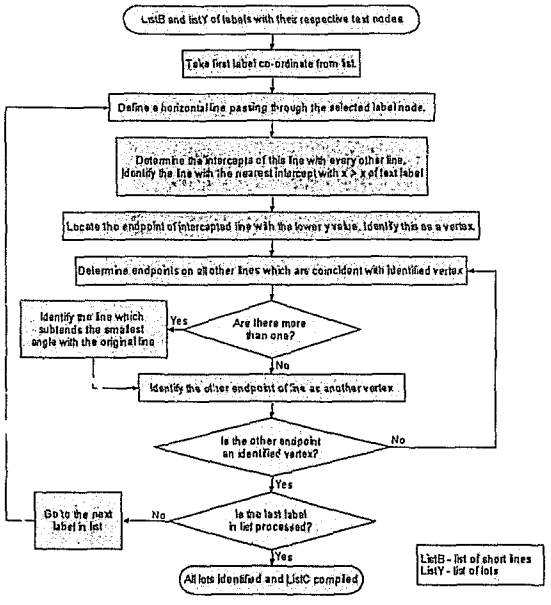


Figure 6 - Lot Identification Algorithm

Street Block Identification. Before roads can be identified the street blocks must be defined. A block is defined as a boundary encompassing a group of contingent lots. While roads are often drawn on maps as separate polylines a short distance from the blocks, in this work all the regions between blocks are considered as the road reserve. There will be no separate space for the kerb. This procedure is to simplify the complexity of the problem and is an acceptable assumption particularly if the areas being electrified are informal settlements. Thus routines are required which will first create block polygons.

The algorithm below is used to generate block polygons. For this process the co-ordinates of the lot boundaries defined in ListC are utilised.

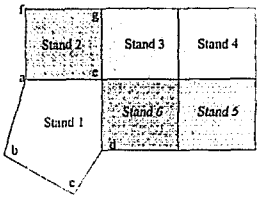


Figure 7

In the first stage it compares each boundary line in each lot with every other lot. If another match is not found for a given boundary line then that line is added to a temporary list (ListD). If a line has no match it clearly implies that the it is not sharing a boundary with another lots and therefore is part of the block boundary. For example, in figure 7 *cd*, *bc* and *ab* will form part of the block boundary (ListD). However *ae* and *ed* will clearly have matches, in which case both lines are ignored. ListD will now contain a list of block boundary lines.

For the next stage, a line endpoint from ListD is taken as a reference and the other endpoint is used to trace the lines which form the block boundary. Every time a line is identified, it is removed from ListD. Therefore all block polygons would have been created when there are no more lines are remaining in ListD. The groups of vertices for each of the identified block polygons are added to a new list (ListE).

Road Polygon Algorithm. The areas between all the identified blocks and the township boundary is to be defined as road polygons. Consider the following simple township consisting of a boundary and two blocks as shown in figure 8a.

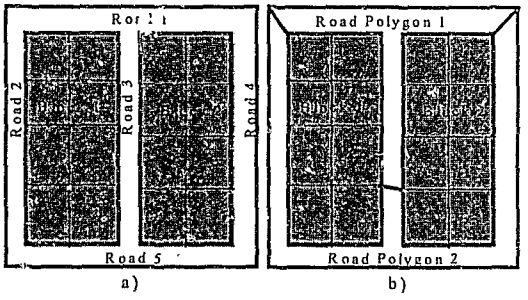


Figure 8

It can be clearly seen that valid road polygons cannot be created merely by tracing vertices on either the township or block boundaries.

Thus interconnecting lines are required, as shown in figure 8b, in order for road polygons to be defined. The criteria for successful road polygon creation is that each block must have at least two interconnecting lines. The two shortest lines from one vertex on each block to a vertex on the township boundary or a vertex on another block are taken as

interconnecting lines. If, in a particular map, an interconnecting line should intersect one or more blocks then the nearest block is selected. A further simplification rule is that an interconnecting line can neither intersect nor share an endpoint with another. These interconnecting lines are compiled in a new list (ListF).

Once the interconnecting lines have been created the road polygons can be traced and these are compiled in a new list (ListH). The road creation algorithm uses the first available interconnecting line in ListF. It takes one endpoint as the reference and uses the other to determine its coincident vertex on either a block (from ListE) or the township boundary (from ListZ). When the coincident vertex is found the routine can trace a polygon in two possible directions (left or right). For tracing the first road polygon, the left direction is always used. When tracing subsequent polygons, the left direction is chosen only if it has not been used for creating a polygon in a previous trace. The vertices along that boundary are now traced checking every new vertex to see whether it coincides with an interconnecting line endpoint (from ListF). When an interconnecting line is encountered, the tracing routine will take this route. A road polygon is created when the reference vertex is found once more. When an interconnecting line has formed part of two road polygons, it is deleted from ListF. Therefore, all the road polygons would have been created when there are no more interconnecting lines remaining in ListF. ListH is now appended to the Map file. The algorithm developed for road polygon creation is capable of handling complicated maps. The flowchart for this operation is given below.

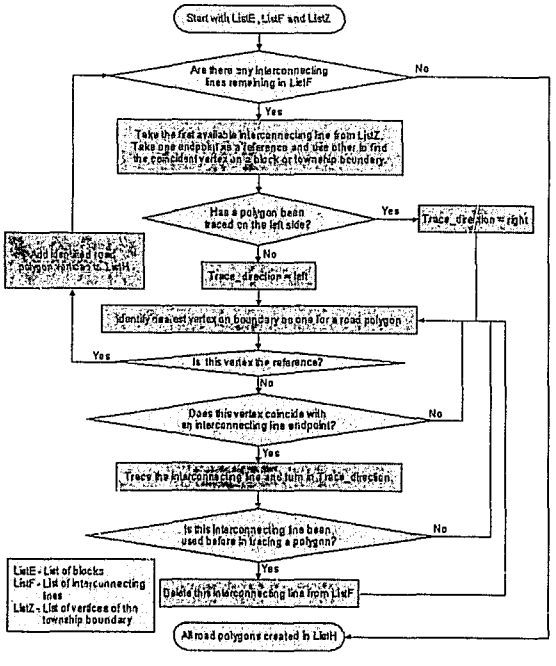


Figure 9 - Road Polygon Creation Algorithm

Cost Region Generation Algorithm. It is a special requirement for the operation of the cable router, that the entire area within the township boundary consist of cost regions. When a draftsman designs the cable layout, the routes are determined by associating certain regions with a cost multiplier measuring the degree of difficulty in erecting a cable through that region. Since computer software is not able to perceive this information it has to be explicitly supplied in the form of convex polygons assigned with a suitable cost multiplier. This information is appended to the MAP file.

The creation of the cost regions have been simplified since all the road and block polygons have been created. It is assumed that all cost regions will be copies of existing polygons defining lots and roads. However, the router requires that the cost region polygons are convex. An algorithm was developed that would convert all concave cost region polygons to convex polygons. The user will be prompted to enter default cost factors for stands and roads.

Thus an intelligent map has been created consisting of lists of lots, roads and cost regions. The designer is provided with the opportunity to change the cost multiplier if the area encompassed by any of the cost regions warrant it.

OVERALL OPERATION OF ASED (AUTOMATED SOFTWARE FOR ELECTRICAL DISTRIBUTION)

The overall operation of ASED is shown in the following flowchart.

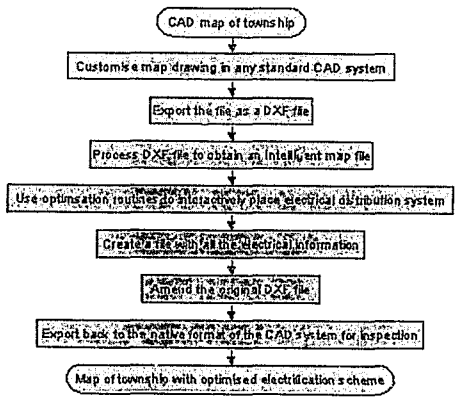


Figure 10 - Flowchart of overall operation of ASED

The optimisation routines have been under development for some years and they route cables, position equipment and calculate the performance of the electrical distribution system. With the intelligent map creation routine, discussed in this paper, the optimisation routines will be implemented in an integrated system for use within a drafting office environment and not merely in a "test bed" as they were originally designed.

At this stage in the development of CAD system, a DXF file provides the most suitable means of coupling the

optimisation routines with a drawing system. With the advances in new CAD software packages, it will become possible for any CAD package to interface directly with the optimisation routines within the Windows environment. This will allow institutions to work with their CAD systems and have the functionality of the optimisation routines available in an integrated seamless environment.

Apart from the use of conventional CAD files, the township map details may only be available in an alternative form. If a geographically referenced drawing file (GIS) is available, then many of the initial routines described can be omitted. Most modern GIS's have built in functionality for cleaning up maps so that duplicate lines are eliminated and overshoots are corrected. Further GIS tools replace long lines with short lines. Generally, the widely used GIS packages are capable of exporting in DXF format.

Further in South Africa, aerial photography is often used as means of rapidly acquiring information on informal settlement layouts. These photographs could be attached (juxtaposed into a mosaic) as a reference file to a CAD or GIS drawing and the relevant features, namely the lot and text labelling, traced.

If hardcopy maps are used they can either be treated in exactly the same way as aerial photographs or converted to vector format using raster to vector conversion routines. On completion the map can be read back into the CAD environment, with the electrification scheme, and printed together with the reference drawing to obtain the final drawing plans.

Used with the ASED optimisation routines the intelligent maps greatly accelerate the design process as compared to the use of conventional GIS which are widely used in conjunction with non-graphical distribution design calculating software packages.

FURTHER READING

Dwolatzky B, Meyer AS, "The Development of an Integrated Set of Software Tools for use in the Design of an Electrical Reticulation Network", South African Universities Power Engineering Conference '95, pp 183-185.

Apostolellis J, Dwolatzky B, Meyer AS, "The evolution of CAD-Based Optimisation Tools for Distribution Network Design", IEEE Africon, Volume 1, September '96, Pg 496-499.

West NA, Dwolatzky B, Meyer AS, "Terrain-Based Routing of Distribution Cables", IEEE Computer Applications in Power, vol 10, no 1, January 1997, pp 42-46.

Pg 638, Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 1: Conceptual Deign", IEEE Transactions on power Delivery, Vol. 8, No. 2, April 1993, pp 637-643.

Sumic Z, Pistorese T, Males-Sumic H, Vankata SS, "Automated Underground Residential Design Part 2: Prototype Implementation and Results", IEEE Transactions on power Delivery, Vol. 8, No. 2, April 1993, pp 644-650.

R Kasturi et al, "Map Data Processing in Geographic Information Systems", IEEE Computer, December 1989, pp 10-21.

BIOGRAPHIES

Thurairajah Rajakanthan received his B.Sc. (Eng) in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa, in 1995. He is currently working towards an M.Sc. at the same University. His work is in developing ASED.

STEPS TOWARDS AUTOMATING THE DESIGN OF ELECTRIFICATION SCHEMES

T Rajakanthan

Department of Electrical Engineering
University of the Witwatersrand, Pvt Bag 3, PO Wits 2050, Johannesburg, SA

As part of the work being done by the Software for Electrical Distribution (SED) Group, at the University of the Witwatersrand to automate the design of rural distribution schemes, this paper focuses on certain aspects of the project. Any software tool aimed at automating facilities design requires an "intelligent" map that will allow it to perceive pertinent features in the same way as a human is able to comprehend information from maps. A software module was developed to create such an intelligent map. Using the "intelligent" map, suitable transformer zones can be determined by the computer. The complexity inherent in the latter problem is discussed and a solution is proposed. Various optimisation modules have been previously developed with the intention of eventually integrating them into a complete package that will relieve the designer of much routine tasks, thus being freed to make more creative decisions.

Keywords: automation of rural distribution design, "intelligent" map, transformer zone

1. INTRODUCTION

As is well known, major strides are underway in South Africa to supply electricity to the majority of the people. Designing electrification schemes manually is time consuming and little or no optimisation can be done due to time constraints. Furthermore, skilled manpower is required to design low cost effective electrical distribution schemes and the time available for optimising layouts is often insufficient.

To aid the electrification projects, the SED Group has for a number of years been involved in the development of several software modules that optimise various aspects of the rural distribution design process. The end goal is to integrate these modules into a single package called ASED (Automation Software for Electrical Distribution) which will enable a design of an electrification scheme to be automatically generated, with adequate accuracy. The cost benefit is that such a design could be created within a matter of hours as opposed to days or weeks.

CART (Computer Aided Reticulation of Townships) was developed initially as a tool to increase the speed of reticulation design [1]. The current version of CART (CART3) has powerful features that make this possible. Even though this tool is valuable, it requires considerable designer interaction. Developing an integrated package that automates the entire design process would save even more time and would facilitate a thorough search for an optimised solution. This package will incorporate the optimisation modules previously developed which include the following:

- Determination of non-domestic loads [2].
- Determination of cable routes [3].
- Calculation of voltage drops and selection of cable sizes accounting for the statistical consumer load distribution [4].
- Determination of optimum junction positions [5].
- Determination of optimum transformer junction cable connectivity [6].

Previously, these modules were incorporated within a "testbed" environment to demonstrate its functionality and interoperability [7]. Maps required by the testbed had to be manually processed so that a database can be created enabling the testbed to "visualise" the map. In order to develop a completely independent package, that can be readily used within a drafting office environment, a module had to be developed for rapidly creating such a database from real township maps. This paper discusses such a module which creates this database or more popularly referred to as an "intelligent" map. The concept was developed [8] and implemented as part of a fourth year research project.

Furthermore, a vital but time consuming part of the design process is the determination of transformer zones. This paper discusses a software module developed for the determination of these zones, which is feasible using the "intelligent" map.

2. INTELLIGENT MAPS

For manual computer based distribution design a township map in the form of a CAD file, must be available at the start of the project. However, any tool aimed at automated facilities design requires an "intelligent" map which implies a non-graphic database linked to the CAD file. The database provides appropriate information about the entities represented on the map. Thus an intelligent map enables the optimisation software to perceive features on a map in the same the way as a human.

A specific form of an intelligent map is required by ASED. The relevant features, after defining, are filed in a text file (*.map) and are as follows:

1. The co-ordinates of the vertices that make up road or stand boundaries must all be listed. Text labels for all stands must be defined since their location is assumed to coincide with the termination point for the consumer service cable.
2. Cost regions must be defined in order for the cable router to function [3]. The townplan must contain non-overlapping cost regions and have a cost multiplier assigned that specifies the degree of difficulty in erecting a line over that region. Therefore, an additional list, similar to the one for stands and roads, is required for representing cost regions. In this case the label will be a number representing the cost multiplier.

The process of intelligent map creation is shown in the following flowchart

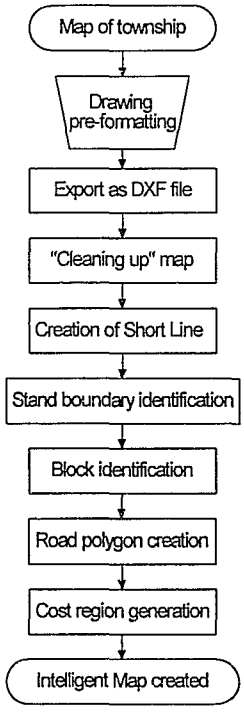


Figure 1 - Flowchart of Intelligent Map creation

The following sections will describe each of the steps shown in Figure 1.

Drawing Pre-formatting. In ASED any form of CAD package or Geographic Information System (GIS) can be used provided that they can export drawing files in AutoCAD™ DXF format. Aerial photographs and harcopy maps are usually digitised into either CAD or GIS files and then dealt with in the same way. The method used here is to segregate only those groups of features necessary for creating an intelligent map. A layer will contain all the stand boundaries while another layer will contain the text labels. A boundary is drawn around the township defining the workspace or optimisation area.

Exporting a DXF file. On completion of these steps, the map of the township is exported from its native format to an AutoCAD™ DXF format. DXF is an internationally accepted standard for exchanging drawing information. The intelligent map creation module now extracts the relevant information from the DXF file.

"Cleaning up" Maps. Digitised maps as drawn vary in quality which may imply one or both of the following defects:

- Lines drawn may not terminate exactly on other lines as intended but fall short or overshoot them. Either of these conditions will not allow computer software to correctly determine the connectivity between the various lines.
- Several vertices may be located within close range of each other near the same intended location. Therefore, there may be more than one line representing the same feature.

The following cleaning up operations are performed.

- Any two endpoints apart by less than a distance r (which has a default value of 1m but can be altered by the user) are assumed to be identical.
- Any line that falls short or overshoots a line by a distance less than r is adjusted to terminate on that line.
- All duplicate lines are eliminated.

Creation of Short lines. Execution of the short line algorithm will break up all the long lines into short ones so that each of those define a section of the boundary for not more than two adjacent stands.

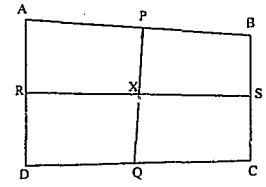


Figure 2

As an example, in drawing the block of stands (Figure 2) *ABCD* was first drawn as a polygon and *PQ* and *RS* as long lines. These long lines and the polygon need to be segmented as necessary for the stand boundary

recognition algorithm. Therefore intercept points P, Q, R, S and X must be inserted and all the long lines segmented accordingly.

Stand Boundary Identification. This routine will identify stand boundaries given the consumer cable termination point which is the location of the stand labelling text node. This routine initially identifies the nearest boundary line and traces the remaining lines by identifying coincident endpoints. All the boundary lines for that stand are thus identified and this information is then written to the MAP file.

Street Block Identification. Before roads can be identified the street blocks must be defined. A block is defined as a boundary encompassing a group of contingent stands.

Road Polygon Creation. While roads are often drawn on maps as separate polylines a short distance from the blocks, in this work all the regions between blocks are considered to be the road reserve. There will be no separate space for the kerb. This procedure is to simplify the complexity of the problem and is an acceptable assumption particularly if the areas being electrified are informal settlements.

The areas between all the identified blocks and the township boundary is now defined as road polygons. Consider the following simple township consisting of a boundary and two blocks as shown in Figure 3a.

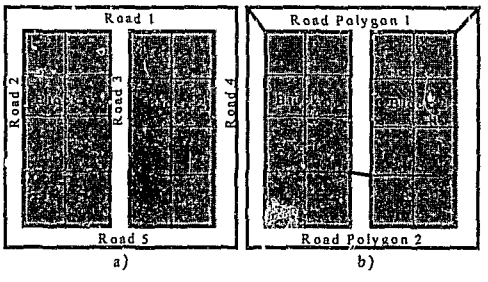


Figure 3

It can be clearly seen that valid road polygons cannot be created merely by tracing vertices on either the township or block boundaries. Interconnecting lines are required, as shown in Figure 3b, in order for road polygons to be defined. These interconnecting lines are used when tracing out road polygons which are then added to the map file.

Cost Region Generation Algorithm. It is a special requirement for the operation of the cable router, that the entire area within the township boundary consist of cost regions. When a draftsman designs the cable layout, the routes are determined by associating certain regions with a cost multiplier measuring the degree of difficulty in erecting a cable through that region. Since computer software is not able to perceive this information it has to be explicitly supplied in the form of convex polygons assigned with a suitable cost multiplier [3]. This information is appended to the MAP file.

Thus an intelligent map has been created consisting of lists of stands, roads and cost regions. The designer is provided with the opportunity to change the cost multiplier if the area encompassed by any of the cost regions warrant it.

3. DETERMINING TRANSFORMER ZONES

At present, transformer zones are determined manually. The designer starts by delineating regions that are to be supplied by a single transformer using visual inspection. The entire area of the township is thus broken up into transformer zones. The designer now counts the stands within each zone and if any have more than the transformer's capacity then they are assigned to adjacent zones as deemed appropriate and vice versa. This is a tedious and time consuming task. Once the zones have been finalised the transformers are located at a suitable position near the load centre of gravity of each zone (again determined by visual inspection).

This methodology is adequate for small areas but the degree of complexity increases substantially when considering areas that require five or more transformers. Therefore locations identified manually not only would have taken a great deal of time but may also be far from optimum.

It is for these reasons that techniques are being investigated for efficiently determining transformer zones. In order to do this an "intelligent" map is required. Existing algorithms, which address this problem, assume that the consumer density is constant and select the best transformer location from a few potential sites. One algorithm (Grimsdale et al [9]) is capable of locating the specified number of transformers on given map of a township. The zones determined using this method were all approximately uniform in area. In a township, with varying load densities, this results in a vast difference in consumers allocated per transformer.

Since this method only produces a partially acceptable solution, further processing is done to resolve this issue whereby zones that have more consumers than the maximum are transferred to the nearest adjacent transformer. Then that transformer will in turn transfer its consumers to other transformers if required and so

on. Optimisation will stop when the total straight line distances from the transformer to every consumer within every allocation is approximately the same. This would result in some zones covering larger areas but encompassing fewer consumers and vice versa. Logically this is the desired result since longer cable runs imply higher voltage drops and thus fewer consumers can be connected to those lines. On the other hand in densely populated areas more consumers can be connected to shorter lines. The transference algorithm is not discussed as its complex and beyond the scope of this paper.

This technique in its present state has a limitation in that it does not allow the locations of the transformers to be influenced by the MV line route. Furthermore, it is accepted that the zones determined will not be completely accurate and a designer may still have to modify the zones as required to facilitate special features on the map.

4. CONCLUSIONS

In the quest for complete automation of distribution design, it is clear that a further step has now been achieved to relieve the user of more routine tasks, thus only requiring him or her to make creative decisions. With the means for rapidly creating "intelligent" maps, ASED optimisation routines is intended to greatly accelerate the design process as compared to the use of conventional GIS which are widely used in conjunction with non-graphical distribution design calculating software packages.

Furthermore, the availability of an "intelligent" map made it feasible to develop software for determining transformer zones. Even though this routine may not calculate completely correct zones, it will provide a good starting point for the designer from which the best zones can be identified quickly.

It is accepted that while computer software will never be able to completely automate the design process, such a tool will prove valuable if it is convenient to use and ensures that the degree of user intervention required is low.

5. REFERENCES

[1] Meyer AS, Dwolatzky B, Design Tools for Mass Electrification, *International World Energy System Conference, Canada, June 1996.*

[2] Patricios MP, Dwolatzky B, Meyer AS, Utilisation of a rule based expert system to aid in Power Reticulation Design, *South African Universities Power Engineering Conference '96*, pp 227-230.

[3] West NA, Dwolatzky B, Meyer AS, Terrain-Based Routing of Distribution Cables, *IEEE Computer Applications in Power, vol 10, no 1, January 1997*, pp 42-46.

[4] Nicolson JM, Object oriented design of cable selection software for low voltage networks, *M.Sc. (Thesis) Engineering - University of the Witwatersrand, 1993.*

[5] Apostolellis J, Dwolatzky B, Meyer AS, The evolution of CAD-Based Optimisation Tools for Distribution Network Design, *IEEE Africon, Volume 1, September '96*, Pg 496-499.

[6] Tumazos SCJ, An expert/algorithm hybrid software system for automatically configuring feeder cables in low voltage distribution networks, *M.Sc. (Thesis) Engineering, University of the Witwatersrand, 1997.*

[7] Dwolatzky B, Meyer AS, A Software based Design Methodology for the Design of Low-Cost Electrical Reticulation Networks, *International Conference on Electricity Distribution (CIRED), 1997.*

[8] Rajakanthan T, Meyer AS, Dwolatzky B, Smart Maps Streamline Distribution Design, *IEEE Computer Applications in Power, January 1998.*

[9] Grimsdale RL, Sinclare PH, The Design of Housing Estate Distribution System Using a Digital Computer, *Proceedings of the IEE 1960*, 107A, pp 295-305.

COMPUTER GENERATED TRANSFORMER ZONES AS PART OF TOWNSHIP ELECTRIFICATION DESIGN SOFTWARE

T Rajakanthan

Department of Electrical Engineering
University of the Witwatersrand, Pvt Bag 3, PO Wits 2050, Johannesburg, SA

Abstract: To assist with the large scale rural electrification program in South Africa, special CAD based software is being developed for the design process. A new tool is presented to further reduce design time. Manually delineating suitable distribution transformer locations and their service areas (transformer zones) is still a time consuming activity. Delineation of optimum transformer zones, minimises cable lengths and improves transformer utilisation. The software routine described in this paper, delineates such transformer zones rapidly producing near optimum solutions. This module can be used to speedily investigate various alternative design strategies. Testing has been done on real maps and the final layouts were found to require only minor adjustment. Software cannot ever produce completely acceptable solutions due to the complexity in township layouts. A sample computer design is presented and discussed.

Keywords: rural electrification design, transformer zones, optimisation software.

I. INTRODUCTION

The electrification drive in South Africa presently connects approximately a 1000 consumers a day. Most of these newly connected consumers earn minimum wages and therefore their energy consumption is low. Thus, the time taken to recover the electrical infrastructure costs is excessive. Skilled labour and time is necessary to optimise designs so that capital and running costs can be reduced.

The SED (Software for Electrical Distribution) Group at the University of the Witwatersrand has for a number of years been involved in the development of software modules that optimise various aspects of rural distribution design [1-3]. The ultimate goal of this group is to develop an integrated software package, called ASED, that can rapidly produce optimum designs with minimum user intervention. This paper discusses a new module that will further reduce design time. Manually, determining optimum transformer locations and service areas is a time consuming activity. Optimised transformer locations and service areas will minimise cable lengths and maximise transformer utilisation. Furthermore, as the size of the township increases, the time taken to manually delineate cost effective zones increases exponentially.

Computer routines for determining the optimum location of distribution transformers exist but most of them require the specification of potential sites from which the most suitable are found [4-10]. In urban areas, the potential sites are well established through master developmental plans. However, in rural areas there are many feasible locations especially since pole-top transformers are used extensively. Some of these techniques require additional data such as possible feeder networks [4,5]. This level of information is not readily available and it will be time consuming and expensive to generate. In some cases, a constant consumer density throughout the township is a prerequisite [11]. Most townships in South Africa do not satisfy this prerequisite. A comparative evaluation done by Willis et al [12] of some techniques lead them to the conclusion that some siting models had an upper limit of 25 transformers. There are many townships in South Africa where significantly greater number of transformers are required. A further criticism on some of the previous work was that optimisation was based on models linearised from non-linear cost functions [10].

Sumic et al's work takes into account various parameters in the definition of zones including the street layout, but operates within one GIS environment [13,14]. The level of information required for this algorithm is enormous and the pre-processing required would be labour intensive. In the majority of the cases, GIS maps are not available in South Africa. Artificial intelligence (AI) [15] and expert system (ES) techniques [16] for load allocation and expansion planning were found to be unsuitable for domestic design requirements.

Grimsdale et al's [17] algorithm was found to operate with minimum data and did not require the specification of potential transformer sites. Initial transformer positions are calculated using an approximation procedure which ignores the road layout. This is accomplished by first breaking up the town into small squares as shown in the first image of fig 1. Then the specified number of square blocks (representing the transformers that would be used) are placed within the shape of the township starting from the bottom right (middle image

of fig 1). The sizes of the blocks are increased sequentially until all the blocks can be just contained within the township as shown in last image fig. 1. The centroid of each block is assumed to be the initial position of the transformer.

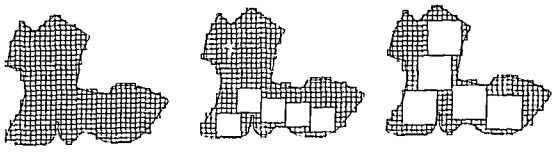


Fig. 1. - Square extraction process [17]

Every consumer is then allocated to the nearest transformer, using the straight line (shortest) distance principle. Better positions are located by moving the transformer from the approximate initial position to the calculated centre of gravity of load for each zone. Every consumer is then reallocated to the nearest transformer and new centres of gravity are calculated for each zone. This process of consumer allocation and movement of the transformer to the centroid of that allocation is repeated until there is no significant difference in transformer positions for two successive iterations.

When this algorithm was tested, it was found that it required a fairly uniform consumer load density which is not typical of rural areas. Results produced by this technique are not suitable for domestic requirements for the following reason. To minimise capital and maintenance costs when designing low cost schemes, minimum number of transformers are used, often loaded to their maximum rated capacity. In urban areas, overloading is not an issue as large transformer are always used.

Nevertheless, Grimsdale's algorithm was considered a good starting point from which further techniques can be developed to advance the solution to an acceptable stage. This paper focuses on the development of those techniques.

The solution could have been advanced using either an AI/ES or an algorithmic technique. AI/ES techniques were found to be unsuitable. "*An AI or ES technique can provide a satisfactory solution but it does not guarantee an optimal solution. If an algorithm exists that optimises the solution, use it*" [18]. The algorithmic technique developed is not based on any linearised function.

II. INITIAL TRANSFORMER ZONING PROCEDURE

Prior to the delineation of transformer zones the following parameters have to be established.

1. **The ADMD (after diversity maximum demand) for the township.** The ADMD represents the statistical simultaneous demand of all the consumers in a township.
2. **Transformer and its rating.** It is common practice to use a single sized transformer and other sizes are used only in cases where it is economically justifiable.
3. ***Maximum zone limit.*** This is the absolute maximum number of consumers allowed per transformer.

For delineating transformer zones, the location of every consumer and their respective load is required. This information is extracted from a database [19] created for use by ASED. This database assumes that the location of the load point coincides with the location of the text labels indicating the land parcel numbering on the original map.

The algorithm proposed by Grimsdale et al [17] is firstly executed. An example is given in fig. 2. of a township with just over 1,100 consumers (represented by each dot) and 8 transformers. Dashed lines are used to represent the roads in the township. The township is approximately 2.7km in width and 2.3km in length. Superfluous map information has been deliberately omitted as they only clutter the map. The number of consumers allocated within each zone is displayed. A small '+' symbol is used to represent the idealised transformer locations.

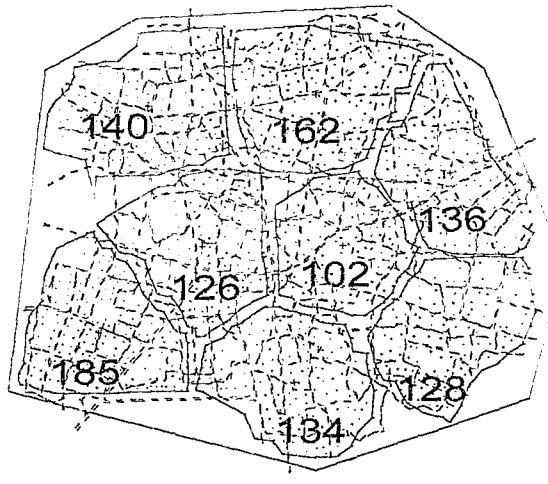


Fig. 2. - Map of Township after execution of Grimsdale's algorithm

It can be seen that the zones delineated using Grimsdale's algorithm are all approximately uniform in area. In a township with varying load densities, this results in a vast difference in consumers allocated per transformer as can be seen form fig. 2. (avg=138, standard deviation σ=25).

III. THE SCANNING ALGORITHM

The scanning algorithm was developed with the intention of re-allocating consumers from overloaded zones to others so that all transformers become evenly loaded. An assumption here is that the transformers being used will be of

uniform rating. (Standardised transformer sizes are preferred since costs do not vary significantly with size and there is no problems with availability.) The basis of this algorithm is that it uses a heuristic approach to minimise the following objective function without violating the *zone limit*. The zone limit represents the maximum number of consumers that can be contained within each transformer zone.

$$TCL = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} abs\left|T_i\, c_j\right| \quad \text{for } M \leq \textit{zone limit}$$

Where T_i - (x,y) position of transformer i,
c_j - (x,y) position of customer j,
N - number of transformers,
M - number of consumers for a particular zone,
$|T_i c_j|$ - represents the scalar distance between T_i and c_j.

TCL is defined as the sum of all the straight line distances from each transformer to every consumer assigned to it. This function is formulated on the logic that when the TCL is minimised for a given number of transformers, the distance to all the consumers have been minimised. Actual road routes are ignored, since rural areas typically have well interconnected pathways.

A "sweep" is an iteration of the scanning algorithm. Theoretically, a sweep can take place in any direction. However, for simplicity, the sweeping was restricted to the positive and negative directions along the x and y axis (total of four directions). The algorithm is described below for the case when it is "sweeping" to the right.

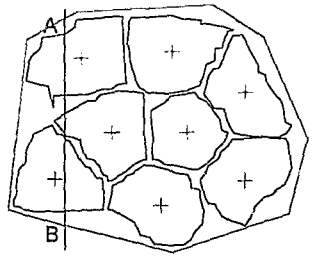


Fig 3 - Line AB "sweeping" to the right

The imaginary line *AB* shown in fig. 3. moves to right and transformers are processed in order as it makes contact with that line. When an overloaded transformer is encountered the following steps are executed:-

1. All transformers located within 1.3 times the distance of the nearest transformer, on the right hand side of this overloaded transformer, are identified (regardless of whether they are overloaded or not). A radius multiplier of 1.3 was intuitively selected and proved adequate in selecting suitable transformers.
2. The consumer that measures the shortest distance to one of the selected transformer is re-assigned.
3. The previous step is repeated until the transformer being processed has shed all the consumers who were overloading it.

The next overloaded transformer is identified and the above procedure is repeated. By transferring consumers from one transformer zone to another, the consumer allocations will eventually even out. New centres of gravity for all zones are calculated at the end of the sweep. The flowchart for the sweeping algorithm is given in fig. 4.

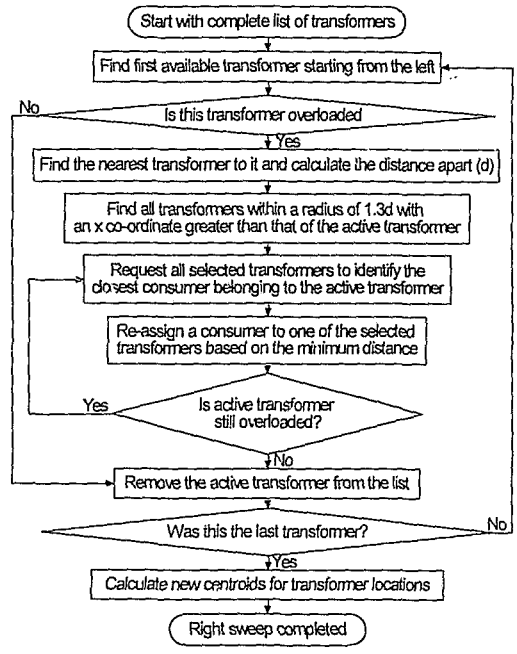


Fig. 4. - Algorithm for a single "sweep"

Load re-allocation can sometimes be accomplished with a single sweep in one or more directions. However, in cases where the zone limit approaches the average, several sweeps may be necessary to converge at a solution. Since multiple sweeps may be necessary, a means of selecting the best direction for every sweeps was required. Such a means would lead the program along the most "promising" path and avoiding the combinatory explosion which will otherwise result. This is accomplished by performing simulation sweeps in all four directions which allows the total number of overloaded consumers to be calculated at the end of every sweep. The direction that resulted in the least number of overloaded consumers is selected for the actual sweep. This is then regarded as the starting point for the subsequent sweep. The process is repeated with the simulations sweeps followed by the actual until a solution is reached, whereby there are no zones with overloaded consumers. A simplification rule is that there cannot be two successive sweeps in the same direction.

Often, there may be more than one minimum in the number of overloaded consumers after the simulation sweeps. When this occurs, the TCL is evaluated to determine the best direction for the actual sweep. The TCL after the execution of Grimsdale's algorithm is taken as the reference. The direction with the least increase in TCL is considered the best direction for the actual sweep. The flowchart for the scanning algorithm as a whole is given in fig. 5.

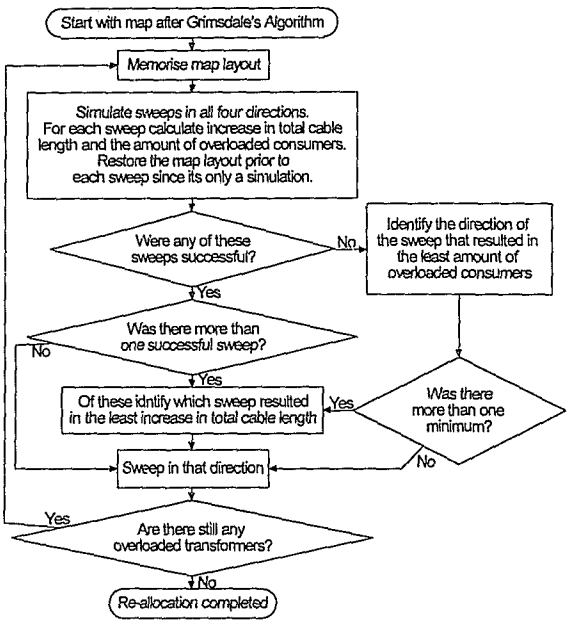


Fig. 5. - Scanning Algorithm

IV. ZONE REFINEMENT PROCESS

An inherent flaw in the sweeping algorithm when multiple sweeps are executed is that it results in the severe distortion of the transformer zones. This results in consumers being assigned to transformers that are not the nearest. An algorithm was evolved to improve the consumer assignment and its basis of operation is described briefly. Assume that there are two consumers at positions x and y who are allocated to transformers at A and B, respectively. If the following mathematical condition is true for the following positive scalar distances:

$$(|Ay| + |Bx|) < (|Ax| + |By|)$$

then the allocations are swapped so that the consumer at x is supplied by the transformer at B while the consumer at y is supplied by the transformer at A. This routine does not alter the number of consumers allocated per zone, since consumer assignments are being swapped. Once again, the percentage increase or rather decrease in TCL with respect to the reference will be used to measure the degree of improvement. The refinement algorithm was found to produce notable improvement to the TCL and a significant improvement to the transformer zones that were delineated.

V. COSTING FORMULA

It would be useful for a designer to investigate the effect on the total cost by increasing the number of transformers used. Adding transformers does not necessarily increase the cost but may result in a decrease since shorter cable runs will then be required. A relationship was empirically found to exist between the TCL and the actual feasible distances which was dependent on the zone limit. The empirically determined costing formula proposed incorporates this relationship. The **Total Project Cost** can be estimated using the following expression.

$$TPC = \frac{TCL \times LVCCF}{1.12e-6(ZL)^3 - 2.9e-4(ZL)^2 + 3.84e-2(ZL) + 1.08} + XFRS(TCF + MVCCF)$$

Where $LVCCF$ Low voltage cable cost factor
TCF Transformer Cost Factor
$MVCCF$ Medium voltage cable cost factor
$XFRS$ Number of transformers
ZL Zone Limit. (It has to be the average zone limit)

The TCF constitutes the actual cost of the transformer, its associated fittings and labour. An additional fixed fee is added to the cost of the transformer to represent the cost of the MV cabling (MVCCF). Once the cost of the transformers and the MV cabling have been accounted for, the remaining cost of the scheme must clearly be for the low voltage cabling and associated street furniture. The low voltage cable cost factor (LVCCF) was empirically determined so that the total cost would be similar to an actual designed electrification layout. The LVCCF is an average figure that accounts for the pole costs and for the different cables that are typically used for feeders and service connections.

A systematic investigation can then be done starting from the minimum number of transformers required to the maximum, with the specification of the average zone limit for each run of the program. Although a higher zone limit can be specified, the factor that accounts for the relationship between the TCL and the feasible routes in the formula is valid only for the average.

The user is then presented with a curve of the results and the one for the sample township is given in fig. 6. In this example, the effect of 6 to 60 transformers starting from a zone limit of 186 to 19 was explored. The user can then closely examine a few of the minima and choose the one deemed best conforming to some criteria practised by that particular drafting office.

An example of a good design as calculated by the software is given in fig. 7. which was 8 transformers (avg=138, σ=2.5). The following empirically determined factors were used:-

LVCCF	=	29
TCF	=	18,000
MVCCF	=	1000

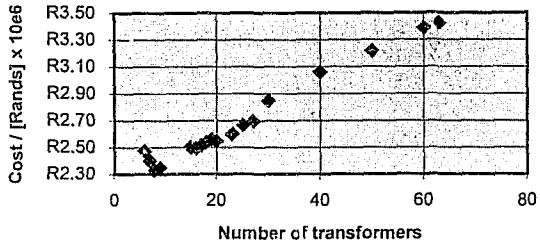


Fig. 6. - Graph of cost vs number of transformers

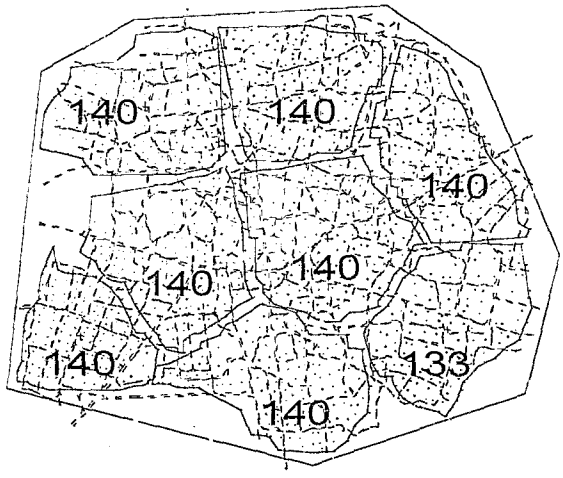


Fig. 7. - The best transformer zone layout

VI. EVALUATION OF PROGRAM

A program was required to assist with transformer zone delineation. Grimsdale's algorithm is used as a first pass at transformer zone delineation. Since the delineation is not acceptable to South African requirements, the scanning algorithm is used in conjunction with the refinement algorithm to advance the solution.

Comparing the differences in TCL between Grimsdale's and the scanning algorithms, a notable difference can be observed between the two depending on the size of township. Generally, as the size of the township increase, the difference in TCL decreases. For the sample township, there was a 1.8% increase in TCL. Despite the increase in TCL, the fact that must be established is that the zones delineated and the transformer locations identified using the proposed technique has minimised the TCL without exceeding the specified zone limit. This implies that the transformer utilisation is maximised in conforming to the designer's criterion. In satisfying this requirement, the program makes an indisputable solution advancement to the one proposed by Grimsdale. It has been found that the TCL may be further reduced if the map is inverted and/or rotated. The map is rotated in 90° steps and then inverted for each position resulting in a possible eight combinations.

Using the proposed transformer zones as a guide, a designer is expected to make some modifications to arrive at an acceptable layout. These modification can be done only after a detailed site visit.

The best transformer location proposed may not be viable as it may be within a consumer's premises. At present, the user will be required relocate the transformers to viable locations. Another limitation is that only a single sized transformer can be position for a single execution of the program.

Several test cases were evaluated, in a similar manner to the sample presented, to determine the number of transformers that will provide the cheapest design. The curve is not always as smooth as the one shown in fig. 6. and its behaviour may be erratic. In all cases the results were found to be acceptable.

The transformer zoning program is computationally intensive, increasing exponentially with greater number of consumers. For a single execution of the program, the times vary from a few minutes for a 1000 consumers to an hour for 6000 consumers. There is no limit to either the size of the township or the number of transformers that can be placed.

VII. CONCLUSIONS

A module is presented for rapidly delineating a first draft of transformer zones given the required number of consumers per zone. This tool can be used to suggest the number of transformers that would provide the cheapest total project cost. The most notable feature is that it produces results with minimum information. The transformer zones produced require minor modifications to create acceptable solutions to satisfy design requirements in South Africa.

The objective function proposed is based on spatial minimisation which ignores all obstacles and land parcel boundaries. Consequently, the transformer locations and zoned determined are idealised. The typical existence of well interconnected pathways ensures that the feasible conductor

routes evolved from the suggested transformer delineation are sufficiently near optimum.

Tests done so far indicates that distribution design time can be significantly reduced by the use of this tool especially for large regions that have well over a 1000 consumers. It has been difficult, however, to show that cheaper designs can be produced. This module relieves the user of a laborious task thus freeing him/her to make creative decisions. It is accepted that while computer software will never be able to completely delineate acceptable distribution transformer zones, such a tool will prove valuable if it is fast and ensures that the degree of user intervention required is low.

VIII. REFERENCES

[1] Meyer AS, Dwolatzky B, "The effective computer generation of designs for township electrical distribution", Proceedings of the IEEE Power Engineering Society, Transmission and Distribution society, Dallas, Texas, September, 1991.

[2] West NA, Dwolatzky B, Meyer AS, "Terrain-Based Routing of Distribution Cables", IEEE Computer Applications in Power, vol 10, no 1, January 1997, pp 42.

[3] Apostolellis J, Dwolatzky B, Meyer AS, "The evolution of CAD-Based Optimisation Tools for Distribution Network Design", IEEE Africon, Volume 1, September '96, Pg 496.

[4] Crawford DM, Holt SB "A Mathematical optimisation technique for locating and sizing distribution substations, and deriving their optimal service areas", IEEE Transactions on Power Apparatus and systems, Vol. PAS-94, No. 2 March/April 1975.

[5] Wall DL, Thompson GL, Northcote-Green JED, "An optimisation Model for planning Radial Distribution Networks", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, No. 3, May/June 1979.

[6] Thompson GL, Wall DL, "A Branch and Bound Model Choosing Optimal Substation Locations", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, No. 5, May 1981.

[7] Sun DI, Farris DR, Cote PJ, Shoults RR, Chen MS, "Optimal Distribution Substation and Primary Feeder Planning via the Fixed Charge Network Formulation", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 3, March 1982.

[8] Fawzi TH, El-Sobki SM, "A New Planning Model for Distribution Systems", IEEE Transactions on Power Apparatus and systems, Vol. PAS-102, No. 9, Semptember 1982.

[9] El-Kady MA, "Computer-Aided Planning of Distribution Substation and Primary Feeders", IEEE Transactions on Power Apparatus and systems, Vol. PAS-103, No. 6 June 1984.

[10] Ponnavaiko M, Prakasa Rao KS, Venkata SS, "Distribution System Planning through a Quadratic Mixed Integer Programming Approach", IEEE Transactions on Power Delivery , Vol. PWRD-2, No. 4, October 1987.

[11] Ben-Dov E, Harley RG, Seymore WJ, "Design of an optimal reticulation system for a Residential Area", IEEE Transactions on Power Systems, Vol. PWRS-2, No.1, February 1987.

[12] Lee Willis H, Northcote-Green JED, "Comparison of Several Computerised Distribution Planning Methods", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 1, January 1985.

[13] Sumic Z, Venkata SS, Pistorese T, "Automated underground residential distribution design, Part 1: Conceptual Deign", IEEE Transactions on power Delivery, Vol. 8, No. 2, April 1993, Pg 638.

[14] Sumic Z, Pistorese T, Males-Sumic H, Venkata SS, "Automated underground residential distribution design, Part 2: Prototype Implementation and Results", IEEE Transactions on power Delivery, Vol. 8, No. 2, April 1993, Pg 644.

[15] Wong KP, Cheung HN, "Artificial Intelligence Approach to Load Allocation in Distribution Substations", IEE Proceedings, Vol. 134, Pt. C, No. 5, September 1987.

[16] Jiann-Liang C, Yuan-Yiu H, "An Expert System for Load Allocation in Distribution Expansion Planning", IEEE Transactions on power Delivery, Vol. 4, No. 3, July 1989, Pg 1910.

[17] Grimsdale RL, Sinclare PH, "The Design of Housing Estate Distribution System Using a Digital Computer", Proceedings of the IEE 1960, 107A, pp 295-305.

[18] Fox MS, "AI and Expert System Myths, Legends and Facts," IEE Expert, February 1990, Pg 8-20.

[19] Rajakanthan T, Meyer AS, Dwolatzky B, "Smart Maps Streamline Distribution Design", IEEE Computer Applications in Power, Volume 11, Number 1, Pg 48, January 1998.

IX. BIOGRAPHY

Thurairajah Rajakanthan received his B.Sc. (Eng) in electrical engineering from the University of the Witwatersrand, Johannesburg, South Africa, in 1995. He has just submitted his M.Sc. at the same University. His work is in developing optimisation software that will eventually lead to the complete automation of the design of rural electrification schemes.

Conclusions and Recommendations

This research investigated two aspects that will contribute to the advancement of the software tool, ASED.

The first aspect was automating the process of creating an intelligent map as required by ASED optimisation routines. The conceptual design for such a software routine has been presented but no algorithms were implemented as part of this research. However, the algorithms have been described in sufficient detail, so that it can be used to effectively accomplish the high and low level design, leading to the implementation of the software. An object oriented design methodology is best suited for the software design and implementation.

The algorithms proposed is designed to create an intelligent map using minimum information. Stand boundaries and their respective labels are present in most commonly available maps sources. The software routine was designed to process this information and create the rest of the information to complete the "intelligent" map. The rest of the information automatically defined includes the workspace boundary, road polygons and cost regions with their respective labels.

Though much of the processing is automated, a limited amount of user intervention is inevitably required to ensure that the "intelligent" map is created with sufficient accuracy. This refers to all ambiguities in feature representation that is detected during the processing of the map and brought to the attention of the operator.

The second aspect researched was the transformer zoning algorithm. There were two objectives that were sought in the development of this algorithm. The primary objective was to be able to produce a first draft in transformer zone delineation. The degree to which these algorithms accomplish the objective has been investigated through a series of extensive tests. The performance of the scanning algorithm with different criteria for determining the sweeping directions produced different results. These criteria and the results have been discussed (Doc B6).

In order to solve an inherent flaw in the scanning algorithm, the refinement algorithm was developed. Alternatives with regards to its usage and its effectiveness has been discussed.

One of the important criteria was to be able to generate acceptable transformer zone delineation with minimum information. In conforming to this requirement, this program produces a draft delineation of the transformer zones using only the geographical locations of the consumers. A designer is expected to make any minor but obvious

modifications to the delineation to arrive at an acceptable solution. Many considerations have to be weighed when modifying the zones. Practical constraints which are identified from the site visit could further influence the transformer zones. Based on these reasons, the author is of opinion that a software tool that can produce completely acceptable transformer zones cannot be created.

Comparisons in TCL were done between Grimsdale's and the scanning algorithms. Generally, as the size of the township increase, the difference in TCL decreases. Despite the increase in TCL in comparison with Grimsdale's algorithm, the fact that must be established is that the zones delineated and the transformer locations identified using the proposed technique has minimised the TCL without exceeding the specified zone limit. This implies that the transformer utilisation is maximised in conforming to South African design requirements. In satisfying this requirement, it has been shown that this program makes an indisputable solution advancement to the one developed by Grimsdale.

Although the original objective was to delineate optimum transformer zones, what this program produces are near optimum solutions. In order to calculate optimum delineation, far greater levels of information is necessary, which is typically not available and time consuming to generate. Since townships have many interconnecting pathways, it is logical to assume that the transformer locations determined by the minimised straight line distances are very likely to produce the cheapest cost. The objective function evolved was based on this premise and has been shown to be effective.

It has been shown that Inverting and/or rotating the map further reduces the TCL for a given zone limit and number of transformers. The program has to be executed completely for the various combinations of inverted and rotated states of the township in order to determine the cheapest configuration. Unfortunately, the best rotated and/or inverted state cannot be determined from the earlier stages of the calculations. For a large township, this implies a substantial amount of computation time and may be impractical at this stage.

The secondary objective was the determination of the optimum number of transformers that would provide the cheapest cost. Using the proposed costing formula, it has been difficult to prove that the scheme proposed is the best. The only indication with regards to the validity of program is that the proposed schemes are reasonable. It is important to note that the ideal number of transformers to be used in a particular township is solely dependent on the costing formula used and not purely on the TCL calculations from the program. The reason for this is that the relationship between the TCL and practical conductor routes for varying zone limits is not linear. The cost equation proposed was empirically determined to

model this behaviour adequately for a specific technology and design strategy (Doc B3).

In an attempt to prove the costing formula proposed, several complete distribution designs were done on a selected township. While the results were encouraging, it was discovered at the end of the survey that there was a gross variation in consumer density on this particular township which was not evident at the start. Since this lead to certain inconsistencies as discussed in Appendix B6, it is recommended that the consumer density be appropriately modelled in the costing formula.

In order to confirm the costing formula a labour intensive study is required whereby a sufficiently large township is obtained and complete electrical schemes are done for varying number of transformers. Using this data, a costing profile can be generated which can then be compared to the one proposed. If necessary, the proposed costing formula can then be modified with ease so that it can more accurately serve in accomplishing the secondary objective. It is recognised that such an exercise, while necessary, would indeed require a great deal of time until ASED is available.

One of the problems with the transformer zoning tool was that as number of transformers involved increases, the computation time increases exponentially. Another problem is that this tool is limited to catering for a single sized transformer for a single execution of the program.

Recommendations have been proposed on improving the effectiveness of this tool. One technique is meant to improve the time taken to make modifications to the delineation, using a more interactive working environment. Once the zones have been created, the user can move a vertex of a boundary and the program will automatically re-delineate the remaining untouched zones. Using this interactive process, a designer can quickly modify the transformer zones into feasible allocations. Every time zones are reformed, updated consumer statistics will be displayed.

It is also recommended that the costing feature described be incorporated into the transformer zoning tool so that time is not wasted in manually transferring information to a spreadsheet.

Using text files as a means of communication between the CAD and the optimisation routines is not an elegant solution. OLE automation is recommended as it creates a seamless environment for the user.

Ultimately, feasible routes have to be considered when trying to accomplish optimum transformer zoning. When that level of information is available some of the existing or more likely new transformer zoning techniques can be considered.

References

Anderson R - Personal communications, GIS Systems Manger, Department of Surveyor General, 1997.

Adams RN, Laughton MA, "Optimal Planning of Power Networks Using Mixed-Integer Programming," Proceedings of the IEE, Vol 121, No. 2, February 1974, Pg 139-47.

Apsotolellis J, "The production of software that aids in t. design of low voltage distribution networks by optimising the locations of junctions" Thesis (M.Sc.)--University of the Witwatersrand,1996a.

Apostolellis J, Tumazos S, West N, Dwolatzky, B. and Meyer, A.S. [1996], The Evolution of CAD-Based Optimisation Tools for Distribution Network Design, *Proceedings of the 4th IEEE Africon Conference*, Stellenbosch South Africa, vol. 1, Sept 1996b, pp 496-499.

Ben-Dov E, Harley RG, Seymore WJ, "Design of an optimal reticulation system for a Residential Area", IEEE Transactions on Power Systems, Vol. PWRS-2, No.1, February 1987, Pg 210-7.

Boardman JT, Meckiff CC, "A Branch and Bound Formulation to and Electricity Distribution Planning Problem," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 8, August 1985, Pg 2112-8.

Carson MJ, Cornfield G, "Design of low-voltage distribution networks", Proceedings of the IEE, Vol. 120, No. 5, May 1973. Pg 585-92

Chen JL, Hsu YY, "An Expert System for Load Allocation in Distribution Expansion Planning", IEEE Transactions on power Delivery, Vol. 4, No. 3, July 1989, Pg 1910.

Crawford D M, Holt S B, "A Mathematical optimisation technique for locating and sizing distribution substations, and deriving their optimal service areas" IEEE Transactions on Power Apparatus and systems, Vol. PAS-94, No. 2 March/April 1975

DeMers M N, "Fundamentals of Geographic Information Systems", John Wiley & Sons, Inc, 1997©. Pg-63.

Dwolatzky B, Meyer AS, "A software based distribution design methodology supporting rural electrification in South Africa", The 42nd IEEE Rural Electric Power Conference, St Louis, Missouri, USA, 27-29 April 1998, pp. B1.1-B1.4

El-Kady M A, "Computer-Aided Planning of Distribution Substation and Primary Feeders", IEEE Transactions on Power Apparatus and systems, Vol. PAS-103, No. 6 June 1984

Fawzi T H, Ali K F, El-Sobki S M, "A New Planning Model for Distribution Systems", IEEE Transactions on Power Apparatus and systems, Vol. PAS-101, No. 5, May 1982. Pg 1129-33.

Fawzi T H, Ali K F, El-Sobki S M, "A New Planning Model for Distribution Systems", IEEE Transactions on Power Apparatus and systems, Vol. PAS-102, No. 9, September 1983. Pg 3011-7

Fox MS, "AI and Expert System Myths, Legends and Facts," IEE Expert, February 1990, Pg 8-20

Gaunt T, "Phased Capital Expenditure - Planning for Uncertainty", 'New' Technologies for Electrification Seminar, CSIR Conference Centre, Pretoria, South Africa, March 1998.

Gee C - Personal communications, ARC/INFO User - Trainee Engineer, Metro Electricity, 1997.

Gillies AC, "The Integration of Expert Systems into Mainstream Software," Chapman &Hall Computing, 1991, Pg 1.

Glamocanin V, Filipovic V, "Open Loop Distribution System Design", IEEE Transactions on Power Delivery, Vol. 8, No. 4, Pg 1900-6, October 1993.

Gönen T, Foote B L, "Distribution-system Planning using mixed integer programming", IEE Proceedings, Vol. 128, Pt. C, No. 2, March 1981.

Gönen T, Ramirez-Rosada IJ, "Review of Distribution System Planning Models: a Model for Optimal Multi-Stage Planning", IEE Proceedings, Vol. 133, Pt. C, No. 7, November 1986, Pg 397-408.

Goswami SK, "Distribution System Planning Using Branch Exchange Technique," IEEE Transactions on Power Systems, Vol.12, No. 2, May 1997, Pg 718-23.

Grimsdale R L, Sinclare P H, "The design of housing-estate distribution systems using a digital computer", Proceedings of the IEE, 1960, 107A, pp 295-305.

Hindi K S, Brameller M I, Gas E, "Design of low-voltage distribution networks: A mathematical programming method", Proceedings of the IEE, Vol. 124, No. 1, January 1977, pg 54-8.

Hsu YY, Chen JJ, " Distribution planning using a knowledge based expert system", IEEE Transactions on Power Delivery, Vol. 5, No. 3, July 1990 pp 1514-9.

Hsu YY, Jwo-Hwu Y, "Planning of Distribution Substations, Feeders and Sectionalising Switches Using Heuristic Algorithms," International Journal of Electrical Power and Energy Systems,Vol.18, No. 5, June 1996, Pg 315-22.

Jager M - Personal communications, Involved with SmallWorld, Department of Information and Technology, ESKOM, 1996.

Kaplan M, Braunstein A, "Contribution to the determination of the optimum site for substations," IEEE Transactions on Power Apparatus and Systems, Vol 100, May 1981, Pg 2263-70.

Kasturi R, Fernandez R, Amlani ML, Feng W, "Map data processing in Geographic Information Systems," IEEE Computer, pp10-21, December 1989.

Knight UGW, "The logical Design of Electrical Networks Using Linear Programming methods", Proceedings of the IEE, 1960, 107A, pp 306-314.

Lawless A - Personal communications, Managing Director - Allyson Lawless (Pty) Ltd. Bently Systems, 1996

Levitt S.P., Dwolatzky, B. and Meyer A.S, Aerial Photograph Interpretation for Electrical Reticulation, *Proceedings of the 6th South African Universities Power Engineering Conference (SAUPEC)*, Johannesburg South Africa, Jan 1996a, pp 219-222.

Levitt, S.P., Dwolatzky, B. and Meyer, A.S, "Aerial Photograph Interpretation for Electrical Reticulation," *Proceedings of the 4th IEEE Africon Conference*, Stellenbosch South Africa, vol. 1, Sept 1996b, pp 500-505.

Levitt, S.P. and Aghdasi, F, "Texture Measures for Building Recognition in Aerial Photographs," *Proceedings of COMSIG '97 (South African Symposium on Communications and Signal Processing)*, Grahamstown South Africa, Sept 1997, pp 75-80.

Levitt S P, "An Aerial Image Interpretation System for Electrical Reticulation", Low Level Software Design - SL 126, MSc Thesis Currently being submitted, 1998.

Lin WM, Tsay MT, Wu SW, "Load Assignment for Determining Substation Service Areas with the Aid of Digital Mapping," Proceedings of the 10th IEEE Region Conference on Computer Communication, Control and Power Engineering, Publ by IEEE June 1993, Pg 430-4.

Lin WM, Tsay MT, Wu SW, "Application of Geographic Information System for Substation and Feeder Planning," International Journal of Electrical Power and Energy Systems,Vol.18, No. 3, June 1996, Pg 175-83.

Meyer AS, Dwolatzky B, "The effective computer generation of designs for township electrical distribution", Proceedings of the IEEE Power Engineering Society, Transmission and Distribution society, Dallas, Texas, September, 1991.

Meyer AS, Dwolatzky B, "Design Tools for Mass Electrification", International World Energy System Conference, Canada, June 1996.

Negotia CV, "Expert Systems and Fuzzy Systems," Hunter College, City University of New York, Benjamin/Cummings Publishing Company, Inc, 1985, Pg 23.

Neill D - Personal communications, Bentley Product Manager (MicroStation™) - Allyson Lawless (Pty) Ltd. Bently Systems, 1997

Newman S, Principles of Interactive Computer Graphics - 2nd edition, McGraw Hill, 1979, Pg 232.

Nicolson JM, "Object oriented design of cable selection software for low voltage networks," Thesis (M.Sc.) - University of the Witwatersrand, 1993.

O'Rourke J, "Computational Geometry in C", 1995, Cambridge University Press, USA, 1993, Pg 80-96.

Patricios MP, Dwolatzky, B. and Meyer, A.S. [1996], Utilisation of a Rule Based Expert to aid in Power Reticulation Design, *Proceedings of the 6th South African Universities Power Engineering Conference (SAUPEC)*, Johannesburg South Africa, Jan, pp 227-230.

Patricios MP, "Utilisation of a knowledge-based expert system for power reticulation design," Thesis (M.Sc)--University of the Witwatersrand, 1997.

Pelser M - Personal communications, Marketing Representative for Computer Foundation, a division of Denel (Pty) Ltd. AutoDesk World, 1997

Ponnavaiko M, Prakasa Rao K S, Venkata S S, "Distribution System Planning through a Quadratic Mixed Integer Programming Approach", IEEE Transactions on Power Delivery , Vol. PWRD-2, No. 4, October 1987.

Quick P - ESKOM employee, Provided information on the GIS usage in 1995.

Ranganathan V, "Rural Electrification in Africa", Biddles Ltd, Guildford and King's Lynn, United Kingdom, 1992, Pg 43-6.

Ranjan R, Das D, Kothari DP, "Distribution System Planning Using Knowledge Based Expert System," Proceedings of the IEEE 1996 International Conference on Power Electronics, Drives & Energy Systems for Industrial Growth, PEDES 1996 (part 1 of 2).

Rao N D, Zhang Y, "An Intelligent Front End for Secondary Power Distribution System Design", IEEE Transactions on Power Delivery, Vol. 7, No. 2, April 1993.

Shao J, Rao N D, Zhang Y, "An Expert System for Secondary Distribution Design", IEEE Transactions on Power Delivery, Vol. 6, No. 4, October 1991.

Shapiro JC, "The application of neural network software and its integration into a design tool for electrical distribution systems" Thesis (M.Sc.)--University of the Witwatersrand 1996.

Stephen R, "Cost Pressure On Electrification", 'New' Technologies for Electrification Seminar, CSIR Conference Centre, Pretoria, South Africa, March 1998.

Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 1: Conceptual Deign", IEEE Transactions on power Delivery, Vol. 8, No. 2, April 1993a, Pg 638.

Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 2: Prototype impiementation and results", IEEE Transactions on power Delivery, Vol. 8, No. 2, April 1993b, Pg 644.

Sun DI, Farris DR, Cote PJ, Shoults RR, Chen MS, "Optimal Distribution Substation and Primary Feeder Planning via the Fixed Charge Network Formulation", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 3, March 1982.

Thompson GL, Wall DL, "A Branch and Bound Model Choosing Optimal Substation Locations", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, No. 5, May 1981.

Tumazos, SJC., Dwolatzky, B. and Meyer, A.S, "A Technique for Automating Stands to Junction Allocation in Electrification Design using Heuristics," *Proceedings of the 6th South African Universities Power Engineering Conference (SAUPEC)*, Johannesburg South Africa, Jan 1996, pp 223-225.

Tumazos SCJ, "An expert/algorithm hybrid software system for automatically configuring feeder cables in low voltage distribution networks. Thesis (M.Sc.)--University of the Witwatersrand, 1997.

Wall D L, Thomson G L, Northcote-Green J E D, "An optimisation Model for planning Radial Distribution Networks", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, No. 3, May/June 1979

Weitzorrek R - Personal communications, GIS Manager, Department of Information and Technology, Sandton Town Council, 1997

West NA, "The development of an automatic electricity distribution network that utilises terrain divided into cost regions," Thesis (Ph.D.)--University of the Witwatersrand, 1996a.

West NA, "The development of an automatic electricity distribution network router that utilises terrain divided into cost regions", PhD Thesis, University of the Witwatersrand, Johannesburg, 1996b

West, N., Dwolatzky, B. and Meyer, A.S. [1997], Terrain-Based Routing of Distribution Cables, *IEEE Computer Applications in Power*, vol. 10, no. 1, January 1997, pp 42-46.

Willis HL, Northcote-Green JED, "Comparison of Several Computerised Distribution Planning Methods", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 1, January 1985a.

Willis HL, Powell RW, Vismor TD, "A Method of Automatically Assessing Load Transfer Costs in Substation Optimisation Studies", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 10, October 1985b, Pg 2771-8.

Willis HL, Northcote-Green JED, Tram HN, "Comparison of Several Computerised Distribution Planning Methods", IEEE Transactions on Power Delivery, Vol. PWRD-2, No. 4, October 1987, Pg 1228-35.

Wong KP, Cheung HN, "Artificial Intelligence Approach to Load Allocation in Distribution Substations", IEE Proceedings, Vol. 134, Pt. C, No. 5, September 1987.

Wong YK, Shi KL, Chan TF, "Effective Algorithms for Designing Power Distribution Networks," Microprocessors and Microsystems, Vol 20, No. 4, June 1996, Pg 251-8.

Yeh E C, Sumic Z, Venkata S S, "APR: A Geographic Information System Based Primary Router for Underground Residential Distribution Design", IEEE Transactions on power Delivery, Vol. 10, No. 1, February 1995.

Bibliography

Booch, G. [1994], *Object-Oriented Analysis and Design with Applications*, 2nd ed. California: The Benjamin/Cummings Publishing Company.

Cantu, M. and Tendon, S. [1994], *Borland C++ 4.0 Object-Oriented Programming*, 1st ed. New York: Random House.

Gillies AC. [1991], *The integration of expert systems into mainstream software*, 1st ed. Great Britain: T.J.Press (Padstow) Ltd.

Holzner S, "Heavy Metal OLE 2.0 Programming", IDG Books Worldwide, Inc, United States, 1994, Pg 367-440.

Understanding GIS: The ARC/INFO Method; Self study workbook version 7 for UNIX and Open VMS, - Environmental System Research Institute, INC, 1995

Parsons D, "Object Oriented Programming with C++", Published by Manish Jain for BPB Publications, Pressworks, New Delhi, India.

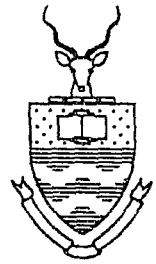
Preparata FP, Shamos MI, "Computational Geometry: An introduction", 1985 by Springer-Verlag New York Inc.

Rajakanthan T, Meyer AS, Dwolatzky B, "Smart Maps Streamline Distribution Design", IEEE Computer Applications in Power, January 1998.

Rajakanthan T, Meyer AS, Dwolatzky B, "Steps Towards Automating The Design of Electrification Schemes", IEEE Computer Applications in Power, January 1998.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. [1991], *Object-Oriented Modelling and Design*, 1st ed. New Jersey: Prentice-Hall.

Stroustrup, B. [1991], *The C++ Programming Language*, 2nd ed. New York: Addison-Wesley Publishing Company.



ASED

Problem Statement And Functional Specifications: "Intelligent" Maps

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Audience 1

1.4 Applicable Documents 1

2 "Intelligent" map 2

2.1 Definition of an "intelligent" map and its purpose 2

2.2 "Intelligent" map contents 2

2.3 Manually procedure for creating "intelligent" maps 4

3 Specifications of a MAP file 8

4 Conclusions 10

Change History

Configuration Control

Project:	SADDIN
Title:	Problem Statement and Functional Specifications: "Intelligent" Maps
Doc. Reference:	1996_34\TP\TB313
Created by:	T Rajakanthan
Creation Date:	31 August 1997

Document History

Version	Date	Status	Who	Saved as:
0.01	97\08\31	Draft	TR	\1995_10\TP\TB313.001
1.00	99\02\04	Approved	TR	\1995_10\TP\TB313.100

Revision History

Version	Date	Changes
0.01	97\01\10	New document created from TB002.100 (Document Creation Template)
0.01	98\04\03	Further revisions were made throughout the document.
0.02	98\09\24	Changes as recommended by Supervisor were made.
1.00	99\02\04	Document was approved

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	99/02/04	Approved	TB 533

Change Forecast

Will be updated as necessary.

1 Scope

1.1 Introduction

ASED (automated software for electrical distribution) is being developed for use within the drafting environment to optimise as much as possible the design of township electrification schemes. An important requirement with regard to the usage of ASED is that an "intelligent" map of the township being reticulated is available. This document outlines v ' at an "intelligent" map is and why it is required. The manual procedure for creating an "intelligent" map is also outlined with the of intention conveying the difficulties that are inherent in the process.

1.2 Purpose

The purpose of this document is to essentially present the specifications and their reasons for their conception. The motivation for undertaking this project is also presented.

1.3 Audience

The project manager and developer, members of the SEAL management board, the external examiner and all other interested parties.

1.4 Applicable Documents

1.4.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.4.2 References

All references made in this document are in the *References and Bibliography* section.

2 "Intelligent" map

2.1 Definition of an "intelligent" map and its purpose

The assumption for traditional CAD based distribution design is the availability of a CAD drawing file, showing the township street and stand layout, at the start of the project. In order for any computer software aimed at automatically designing facilities to function, various features represented on the map must be identifiable. While it might be obvious to the human eye where stand boundaries and roads are located on a town plan, only undifferentiated lines will be detectable by computer software. This problem can be overcome through the use of "intelligent" maps. This implies a non-graphic database linked to a CAD file which provides appropriate information about the relevant features on the map. Thus an intelligent map enables the optimisation software to perceive features on a map in the same the way as a human. "Intelligent" maps are not readily available. The common sources of cadastral information in South Africa are :-

- Digitised town layouts(CAD drawing files).
- Geographical Information System (GIS) maps,
- Aerial photographs, and
- Hardcopy maps.

The reason for developing an "intelligent" map as described here was to have a universal means of transporting map information from various other sources to ASED, thus ensuring its platform independence. It is undisputed that there are many suitable proprietary systems being used by various institutions. If ASED was designed to operate within a particular proprietary system, it would force those organisations that want to utilise ASED to acquire that particular system. More likely, it would exclude a significant portion of the potential users. Designing a system to be independent of any proprietary system avoids this problem. Furthermore, an independent system is not affected by version changes of the individual proprietary systems that could result in the inability to continue using the optimisation tools.

Thus the software unit being designed for creating "intelligent" maps must be able to do so from all sources of maps listed above.

2.2 *"Intelligent" map contents*

Various optimisation routines have been developed by previous students working in the research. For any of these optimisation routines to perform their tasks, a specific form of an "intelligent" map is required. The "intelligent" map of the township must contain the following information:

1. Co-ordinates of the vertices of polygons defining stand boundaries.
2. Co-ordinates of the consumer supply point, which is assumed to be where the stand labelling text node is located.
3. Co-ordinates of the vertices of polygons defining road sections.
4. Co-ordinates of the road labelling text node.
5. Co-ordinates of the vertices that make up cost regions. Cost regions, cost factors and their purpose is described in section 2.3.2.
6. Co-ordinates of the cost factor text node.
7. Co-ordinates of a polygon defining the boundary region in which all the consumers of the township exist. The boundary, referred to as the township boundary, is defined so that it can be used in the road polygon creation process. The transformer zoning program can uses it when placing the transformers, as explained in the document Transformer Zoning: Conceptual Design (Doc no. B3).
8. Co-ordinates of a rectangular boundary defining the region in which all the optimisation will take place. This boundary, referred to as the workspace boundary, is a spatial constraint that literally restricts the optimisation routines operational workspace. All feature located outside these boundaries are ignored.

The distinction between the concepts of the two types of boundaries mentioned above are better described graphically as shown in Figure 1. The township boundary must be geometrically contained within the workspace boundary.

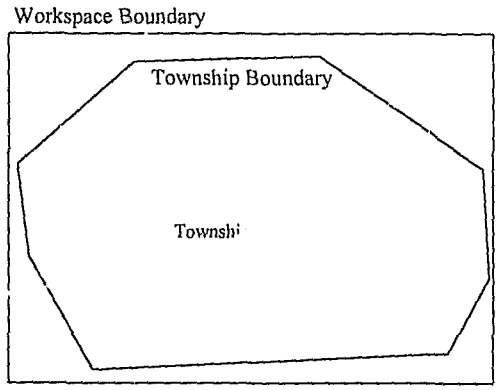


Figure 1 - Differentiation between township and workspace boundary

2.3 Manually procedure for creating "intelligent" maps

The manual methodology for creating "intelligent" maps, as required by the optimisation routines of ASED, is described in this section. The purpose of this exercise is to demonstrate the tedious and time consuming nature of the process. The required format of the "intelligent" map file is also stated.

2.3.1 Creating a listing of stands and roads

CAD maps had to be drawn specifically for the purpose of creating an intelligent map. Original maps of townships would be taken and polygons would be drawn over every stand boundary on a designated unused layer[1]. The layer on which stand labelling text is located is identified. If stand labels are not present, then they are placed approximately at the centroid of the stands on a designated unused layer.

The draftsman is further slowed down when defining polygons for adjacent stands. To show that two stands are adjacent, the polygons drawn must have at least one line of the polygon sharing the same location. In other words, two sets of vertices must be coincidental so that the optimisation routines can recognise that particular line as a boundary between two particular stands. Placing vertices of such polygons with precision that ensures that their location coincides exactly is accomplished through the use of snap to point[2] feature. The required accuracy in vertex placement is demonstrated by a block of six stands in Figure 2, where the boundaries being shared are shown by the thicker lines and vertices being shared are marked with the letter *A*.

[1] Typically, all CAD packages provide the facility for drawing different features on different layers thus enabling users to filter information so that views can be customised by the user as required.

[2] Snap to point is a feature supported by most CAD packages that allows a new vertex being placed to automatically coincide exactly with a previously placed vertex which is nearest to the point of placement

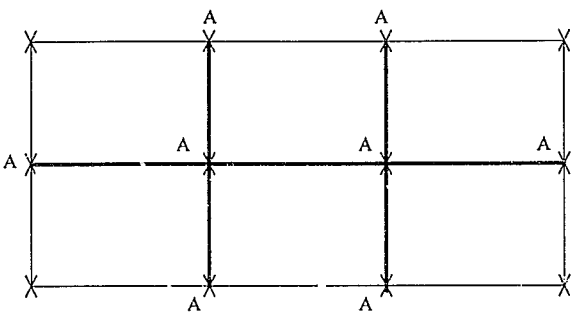


Figure 2 - Drawing of Polygons when represent Blocks

A similar methodology is used for representing roads and road names on yet another two designated unused levels. For purposes of identifying all the relevant layers, they are labelled meaningfully. The labels are STAND, STANDNAME, ROAD and ROADNAME.

The township and workspace boundary are drawn (conceptual difference graphically demonstrated in Figure 1) on separate layers named BOUNDARY and WORKSPACE respectively.

2.3.2 Creating cost regions and cost factors

DNR (Distribution Network Router) was developed by Nicky West as part of a PhD research project in order to automate and optimise point-to-point cable routing. A full description of the algorithm, its software implementation and testing is given in West's PhD thesis [1996]. West et al [1997] gives a formal definition of the Routing of Distribution Networks Problem (RDNP) as follows: *"Given a map that is divided into cost regions and a set of nodes that are to be connected, find the cheapest network that connects them. Nodes correspond to transformers, consumers or intermediate junction point. Each pair of nodes that is to be connected must be specified. The cost in each region must be homogeneous (i.e. non-varying) and isotropic (i.e. not dependant on direction). The regions must be convex polygons (a convex polygon has the property that any line connecting any two points inside the polygon must itself lie entirely inside the polygon). Each region of the map must have some cost associated with it. This value indicates the cost per unit distance of constructing a distribution network over the region. In the case of an obstacle, the cost is considered infinite. The map is a collection of non-overlapping polygonal cost regions. Each region is defined by its vertices and its cost. The distribution network comprises a set of paths and a total cost."*

An analogy to the logic of DNR would be that when a draftsman designs the cable layout, the routes are determined by associating certain regions with a cost multiplier estimating the degree of difficulty in erecting a cable through that region. Since computer software is not able to perceive this information it has to be explicitly supplied in a specific form as described below.

Since ASED will be using this router module, the workspace is required to be broken up into non-overlapping convex polygons with a suitable cost factor. Cost regions have to be defined on an unused layer by drawing polygons typically over homogenous regions such as groups of stands, roads and other terrain in the workspace, paying attention to the fact that those polygons may be convex. Cost factors are defined on yet another unused layer by locating them approximately at the centroid of each cost region polygon. Higher cost factors are assigned to regions which increase difficulty in erecting cables. Sufficient information has to be available to the designer in order to assign suitable cost factors to the regions. Some of these include, in order of importance:

- Environmentally sensitive regions that are difficult to route cables through. A large region would be drawn encompassing that area and assigned a sufficiently high factor to "discourage" the router from traversing that region.
- Altitude information would also be crucial especially if the route has to go through mountainous regions. (dealt with as previous.)
- Sewerage and water reticulation. This information is crucial when erecting poles. Strips of cost regions would be created with sufficiently high factors.
- Rivers and streams. (dealt with as previous.)
- Railway lines, if present, is an important aspect as poles are often erected along the lines. However, this would only apply to the MV side. Strips of cost regions would be created with low factors.
- Telecommunications lines. The possibility of using the same poles for cabling have to be examined with each particular case. (dealt with as previous.)

The layers on which cost regions and cost factors appear must be labelled COSTREGION and COSTFACTOR, respectively.

In defining cost regions it is important that no vertex is situated over a line, which implies that a vertex must coincide with one or more vertices [West:1996]. The constraint is demonstrated by the simple pictorial in Figure 3.

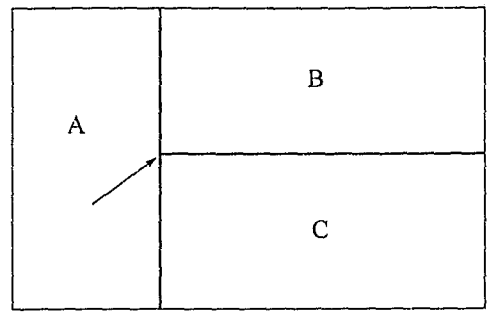


Figure 3 - Three rectangular cost regions

There are three non-overlapping rectangles representing cost regions. Cost regions *B* and *C* are sharing a common vertex at the location pointed to by the arrow. These vertices are located on the right hand side of cost region *A*, which is an unacceptable situation. Cost region *A* must contain an extra vertex so that it coincides with the vertices of cost regions *B* and *C* (at the location being pointed to by the arrow).

2.3.3 File conversion

Once the existing map has been re-drawn as specified above it would have to be exported in AutoCAD DXF format. DXF is the accepted standard for exchanging drawing information. Since DXF files are in standard text format, the required information can be easily extracted.

A file conversion program was developed by the author (DXF to MAP file converter - described in Appendix C) to convert from DXF to MAP file format. This conversion routine was initially used for creating sample maps so that the optimisation routines developed by other MSc students can be tested. The conversion routine is able to identify the relevant polygons since they are located on appropriately labelled layers and compile the information in the format as specified in the following section.

3 Specifications of a MAP file

The relevant features as required by ASED must appear in a text file with an *.map* extension and the format is specified in this section. The format of this file was established as part of the previous research project while developing the optimisation routines [Apsotolellis: 1996]. A sample of the required format for a stand is given below (signalled through the use of a different font):-

```
STAND
120     110         Stand identifier co-ordinate
House 1             Stand identifier (text)
100     80          vertices of the boundaries
100     140
150     140
150     80
100     80
SEQEND
```

and for roads:

```
ROAD
220     210         Road identifier co-ordinate
Road 1              Road identifier (text)
200     180         vertices of the boundaries
200     240
250     240
250     180
200     180
SEQEND
```

An additional list, similar to the one for stands and roads, is required for the cost region. In this case the label will be a number representing the cost multiplier.

```
COSTREGION
120     110         Cost region identifier co-ordinate
COST=3.7            Cost factor (in this case the cost factor is 3.7)
100     80          vertices of the boundaries
100     140
150     140
150     80
100     80
SEQEND
```

A similar data structure is used in defining the workspace and township boundary naturally excluding some information.

```
BOUNDARY
100     80          vertices of the boundaries
100     140
150     140
150     80
100     80
SEQEND
```

and

```
WORKSPACE
100     80          vertices of the boundaries
100     140
150     140
150     80
100     80
SEQEND
```

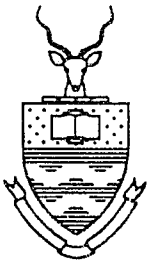
As the optimisation routine reads every listed vertex, it is able to reconstruct the boundary for each polygon, thus defining the relevant features.

4 Conclusions

It has been demonstrated that manually redrawing the township by creating numerous polygons so that the map becomes suitable for conversion is a time consuming and tedious process. Drawing polygons, even for relatively small settlements accommodating under 200 consumers, can easily take a day (estimate based on trial).

It is even more obvious that the time spent on creating an intelligent map manually would defeat the time saved by the optimisation routines. Thus the purpose of this research is to explore techniques that can be used to streamline the process of creating an intelligent map so that ASED can take a step forward in becoming a feasible product.

The research must also consider the methods of acquiring the necessary information from other map sources (GIS, aerial photographs and hard copies of maps) quickly and efficiently.



ASED

Literature Survey: "Intelligent" Map

Management Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Applicability 1

1.4 Definitions 1

1.5 Audience 2

1.6 Applicable Documents 2

1.7 Assumptions 2

1.8 Requirements Traceability 2

2 CAD maps (digitised maps) 3

3 GIS maps 6

3.1 Geographic information system Concepts 6

3.2 GIS usage 6

3.3 GIS in South Africa 10

4 Aerial Photographs 14

5 Hard copy maps 15

6 Conclusions 16

Change History

Configuration Control

Project:	SADDIN
Title:	Literature Survey: "Intelligent" map
Doc. Reference:	E:\1996_34\TP\TB145.001.DOC
Created by:	T Rajakanthan
Creation Date:	24 June 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	96\11\12	Draft	T R	TB 145.001
1.00	99\02\04	Approved	T R	TB 145.100

Revision History

Version	Date	Changes
0.01	96\11\12	Document was created from template TB002.100
0.01	98\06\24	Major changes were made in preparation for submission
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	99\02\04	Approved	TB 533

Change Forecast

Updates will be made as new material is covered.

1 Scope

1.1 Introduction

This part of the project entails investigation into the application of computer based tools to create "intelligent" maps from the numerous map sources. There are four types of sources of cadastral information available in South Africa.

- Digitised town layouts(CAD drawing files)
- Geographical Information System (GIS) maps,
- Aerial photographs and
- Hard copies of maps

Each of these types of maps are investigated to determine its mechanisms for storing data. Understanding the data storage mechanism will allow a method of data extraction to be devised. The map sources are also evaluated within a South African context.

1.2 Purpose

This document presents the findings of the literature survey and serves as a reference to all documents and personnel encountered relating to the various map source.

1.3 Applicability

1.4 Definitions

AM/FM: Automated Mapping/Facilities Management

GIS: Geographic Information System

HV: High voltage

LV: Low voltage

MV: Medium voltage

SCADA: Supervisory Control and Distribution Automation

Informal settlement/Rural area/township - these terms used interchangeably essentially refer to low cost housing settlements.

Block: Refers to a boundary around a group of contingent stands

1.5 Audience

The project supervisor, namely Mr A Meyer

The project developer, namely T Rajakanthan

1.6 Applicable Documents

1.6.1 Specifications

1.6.2 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.6.3 Bibliography

1.6.4 References

All the references made in this document are in the *References and Bibliography* section.

1.7 Assumptions

1.8 Requirements Traceability

2 CAD maps (digitised maps)

Computer aided drafting (CAD) programs have been available for few decades. CAD drawings, in the more popular DXF, DWG or MicroStation™ DGN, are referred to as digitised maps for obvious reasons. CAD systems store map features in vector format to ensure high levels of accuracy necessary for precision drawing. Most CAD packages feature numerous tools to aid precision drawings to be accomplished in a short space of time.

The spatial data storage mechanism within a CAD environment is now described. Every feature, such as a polygon or polyline, is stored as a feature entity. A feature entity consists of information pertaining to the location of all its elements (that are part of the feature) and header information defining its properties such as the layer, colour, thickness and type (closed polygon or polyline). Polygons and polylines are represented in a similar manner in terms of sequentially listing the vertices, except for a difference in type specification. The type specification distinguishes the feature as just a polyline or as a closed polygon.

Single lines are also represented as an entity but the information is contained in a slightly different format. A polyline may be represented either as a series of line entities or as a single polyline entity. Visually, they would appear the same but the storage mechanisms are different. Text is also stored as an entity which contains information on the textual properties (such as font and size and the location of the text node). The text node represents the location at which text is situated within the vector based environment.

Stands are typically represented using polygons and lines. Roads are drawn as polylines a short distance away from the block as shown by the dashed lines in a part of the township in Figure 1.

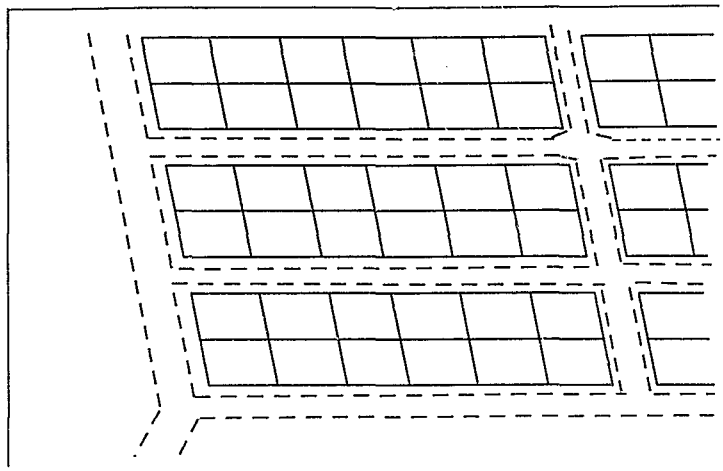


Figure 1 - Part of a township

CAD maps are readily available but, as drawn, vary in quality which may imply one or both of the following defects [14,15]. (The defects have been deliberately exaggerated for visualisation purpose and shown by circles in Figure 2.)

- Lines drawn may not terminate exactly on other lines as intended but fall short or overshoot them. Either of these conditions will not allow computer software to correctly determine the connectivity between the various lines.
- Several vertices may be located within close range of each other near the same intended location. Therefore, there may be more than one line representing the same feature.

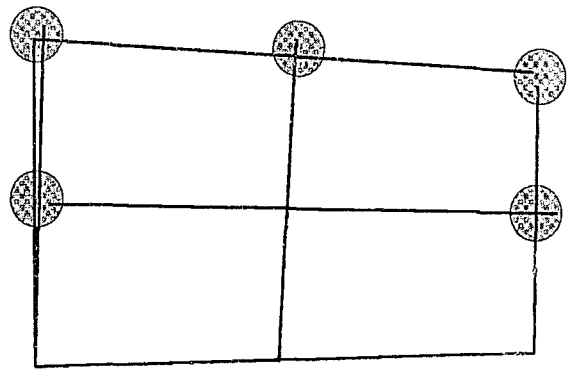


Figure 2 - Defects in the drawing of a block of 4 stands

Manual correction is necessary and there are tools available to aid the process. Some of the most important are now described.

- Tentative point location feature allows the placement or relocation of a line endpoint exactly on a previously placed vertex in the vicinity of the placement. This feature can be used to relocate line endpoints to coincide with others.
- The *extend element to intersection* feature extends or trims a selected line so that it accurately terminates onto another selected target line.
- The *extend 2 elements to intersection* feature extends or trims two selected lines so that they accurately intersect.

This manual cleaning up process is still time consuming but is nevertheless necessary to ensure that map features are correctly represented. Correctly represented features ensure that are no spatial ambiguities will occur when external systems utilise this data.

CAD systems are used widely due to the fact that they have been available for a long time. Thus, CAD maps are the easiest and most readily available cadastral source. AutoCAD™ DXF format was established as the standard for data exchange so that information can be effectively exchanged between a variety of proprietary CAD systems. Thus, most of the popular CAD packages are capable of exporting maps in DXF format which is in ASCII text. Extracting the information from CAD files involves understanding the DXF format for feature representation. Manuals describing DXF file formats in great detail are available from the vendors of AutoCAD™, a proprietary CAD package.

3 GIS maps

3.1 Geographic information system Concepts

The use of geographic information systems (GIS) grew dramatically in the 1980s predominantly in America. It is now commonplace for business government and academia to use GIS for many diverse applications. Consequently, many definitions of GIS have been developed and a definition by Environmental System Research Institute (ESRI:1995:Chapter 1 - Pg 5) is given: "*An organised collection of computer hardware, software, geographic data and personnel designed to efficiently capture, store, update, manipulate, analyse, and display all forms of geographically referenced information.*"

In simple terms, a Geographic Information System (GIS) is a set of software and hardware tools that perform spatial analysis and produce geographic displays, such as maps, using a database that models the real world. Today these systems make vast amounts of data available to many users at many locations and for many purposes. The data includes:

→ Locational data such as streets, rivers or contours
→ Facility data such as buildings, water or gas mains, or electric or telephone lines
→ Attributes related to either the location or the facility data

3.2 GIS usage

Many computer programs, such as spreadsheets, statistics packages or drafting packages can handle simple geographic or spatial data. The speciality of GIS is that it allows spatial operations to be done on the data.

Assuming that sufficient information has been gathered, a user may request certain data conforming to some specified criteria. This is accomplished through the use of queries which compiles information satisfying user defined criteria from all available databases. It is typical for GIS to have relational databases. Every element has a unique reference or identifier and that is used to store different types of information in different databases. Use of the unique identifier makes it possible to compile a variety of information for a single feature from several different databases, even from a variety of different database propriety systems located across the country. The feature attributes are cross referenced using the relation database management systems [Kasturi et al: 1989].

There are many more powerful features offered by GIS but will not be discussed as they are not relevant to this research. It is those features that are used mostly by utilities interested in maintaining automated mapping/ facilities management (AM/FM) systems.

To summarise GIS is able to deal with

- spatial information describing the location and shape of geographic features and their spatial relation to the other features.
- descriptive (thematic) information about the feature (non-spatially oriented information).

3.2.1 Representation of features

As stated earlier, in order to extract features from GIS, the data storage mechanism must be understood. Vector representation, is one of the most frequently used GIS models[1]. In this model a geographic feature is represented by one of the following methodologies [ESRI:1995: Chapter 2 - Pg 5] depending on its nature:

1. Nodes: *xy* co-ordinates representing the discrete location of a point feature. It defines a map object whose boundary or shape is too small to be represented as a line or area. Examples of nodes would be pole-top transformers and poles.
2. Lines: pairs of *xy* ordered co-ordinates that, when connected, represent the linear shape of a map object too narrow to display as area (commonly referred to as polylines[2] in CAD terminology) such as roads and streams.
3. Polygons: a closed figure whose boundary encloses a homogenous area, such as a province, farmland and stands.

This information is represented in the Cartesian co-ordinate system as shown in Figure 3.

[1] The universal transverse Mercator system (UTM) is another spatial representation model which is not evaluated since maps of interest are small and can be reproduced into the Cartesian co-ordinate system. Conceptually, the data storage mechanisms are identical [DeMers: 1997].

[2] Some GIS, including ARC/INFO, refer to lines as *arcs*.

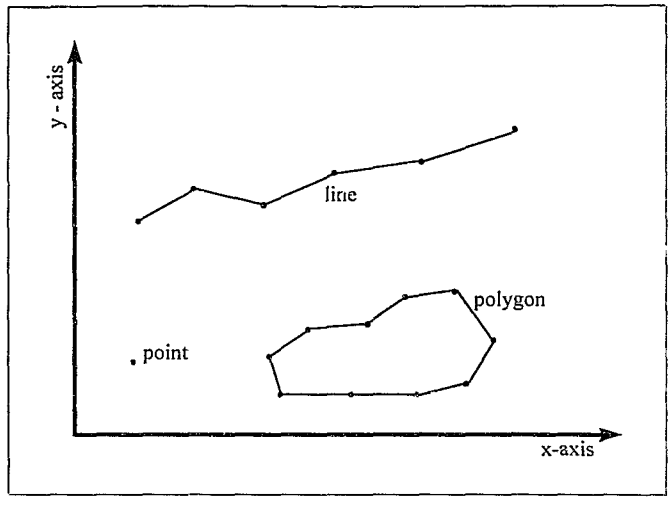


Figure 3 - Representing features on a Cartesian plane

A GIS does not store a map in any conventional sense nor does it store a particular image or view of geographic area. Instead, a GIS stores the data from which desired views can be generated to suit particular purposes.

Each feature is automatically assigned a unique sequence number or identifier. The co-ordinates for each feature are recorded and kept with the unique sequence number so that it can be referenced with the relevant databases. A point feature, similar to the one shown in the figure above, may be stored as shown in the in table below [ESRI:1995: Chapter 2 - Pg 7]:

A point feature may be stored as shown in the following table:

Feature Unique No.	X- co-ordinate	Y- co-ordinate
1	2.34	4.34
2	5.64	2.45

A line (or polyline) feature may be stored as shown in the following table:

Feature Unique No.	X- co-ordinate	Y- co-ordinate
1	2.34	4.34
	3.54	6.76
	8.23	7.89
2	5.64	2.45
	4.34	3.45
	7.45	6.42

A polygon will be stored as shown in the following table:

Feature Unique No.	X- co-ordinate	Y- co-ordinate
1	2.34	4.34
	3.54	6.76
	8.23	7.89
	2.34	4.34
2	5.64	2.45
	4.34	3.45
	7.45	6.42
	5.64	2.45

Note that for representing polygons in ARC/INFO, the last co-ordinate is repeated to show that the feature is a closed region. Some GIS do not use this system of repeating the last polygon but a code referenced to that feature is used to state its type.

Attention must be drawn to the fact that the storage mechanism of GIS is conceptually identical to "intelligent" maps for at least as far as stands and stand labels are concerned. However the roads are not represented as polygons but rather as polylines. Every segment of the polyline describes the type of road paving, width, number of lanes and other related information. Whenever polylines cross each other, a node is defined to indicate the existence of an intersection. These nodes are used whenever point to point routes and distances have to be determined. Most modern GIS's also have powerful features for terrain modelling.

Most GIS's are capable of exporting data in a variety of formats including DXF and ASCII text when data has to be ported form one proprietary GIS system to another. Often, spatial data is exchanged in DXF format while ASCII text files are used for exchanging thematic information.

3.2.2 Feature identification

Often, the cadastral sources for GIS systems being created are standard CAD maps [Weitzorrek:1997, Anderson:1997]. Most GIS products have built in functionality for cleaning up maps (types of map defects outlined in section 2) so that duplicate lines are eliminated and overshoots are corrected [Weitzorrek:1997, Anderson:1997, Neill:1997, Pelser:1997].

Prior to identification of the features on the map further, refinement of the drawing is n cessary. This problem is usually caused when the drawing of the township was originally done. As an example, when drawing the block of stands (as shown in Figure 4) *ABCD*, it is typically drawn as a polygon or rectangular block and *PQ* and *RS* as long lines. Intercept points *P*, *Q*, *R*, *S* and *X* must be inserted to segment the long lines *AB*, *DC*, *AD*, *BC* and *RS* respectively. This effectively creates a new set of short lines such

as *AP*, *PB*, *BS*, *SC* and so on. Creating short lines to replace all the long lines is necessary for the feature identification process [Weitzorrek, Anderson, Neill, Pelser]. Therefore, each short line must not delineate a section of the boundary for not more than two adjacent stands. If there are ambiguities in the identification process, the user will be requested to clarify the specific error.

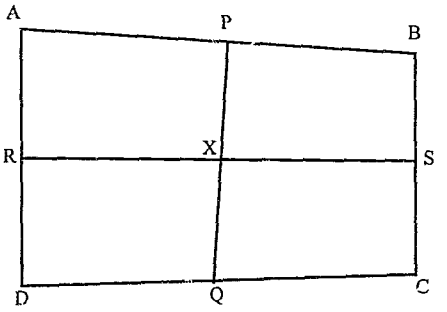


Figure 4 - A block of four stands

Most GIS packages also provide features for rapidly recognising stand boundaries using the location of the stand label [Weitzorrek, Anderson, Neill, Pelser]. This process, sometimes referred to as featurising maps, is where the stand labels and their respective boundaries are identified for compilation in a database. Manual featurisation process involves selecting the stand label and explicitly identifying the stand boundaries [Weitzorrek, Anderson]. Some GIS have an automated identification process which uses the relative position of the stand labelling text node to determine its boundaries [Neill, Pelser]. If they are ambiguities, then the software will prompt the user to make the correction. In order to effectively accomplish this task only the stand boundaries must be located on the layer in which the featurisation process is occurring.

Unfortunately, the road reserve cannot be automatically determined, to the best of the author's knowledge. This has to be accomplished manually by placing lines and specifying connectivity nodes at road intersections.

3.3 GIS in South Africa

In South Africa there is no overall GIS strategy or vision and an unfortunate consequence of this is that there is no uniformity with respect to either the information available or the GIS's currently in use [Anderson]. For example, if one considers Gauteng, one would find no fewer than five different GIS's in use by the various town councils [Quick:1995]. Furthermore, the councils are generally at different stages of information capture.

as *AP*, *PB*, *BS*, *SC* and so on. Creating short lines to replace all the long lines is necessary for the feature identification process [Weitzorrek, Anderson, Neill, Pelser]. Therefore, each short line must not delineate a section of the boundary for not more than two adjacent stands. If there are ambiguities in the identification process, the user will be requested to clarify the specific error.

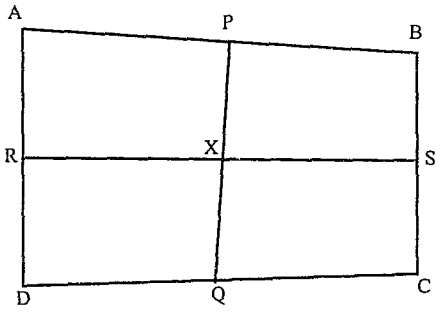


Figure 4 - A block of four stands

Most GIS packages also provide features for rapidly recognising stand boundaries using the location of the stand label [Weitzorrek, Anderson, Neill, Pelser]. This process, sometimes referred to as featurising maps, is where the stand labels and their respective boundaries are identified for compilation in a database. Manual featurisation process involves selecting the stand label and explicitly identifying the stand boundaries [Weitzorrek, Anderson]. Some GIS have an automated identification process which uses the relative position of the stand labelling text node to determine its boundaries [Neill, Pelser]. If they are ambiguities, then the software will prompt the user to make the correction. In order to effectively accomplish this task only the stand boundaries must be located on the layer in which the featurisation process is occurring.

Unfortunately, the road reserve cannot be automatically determined, to the best of the author's knowledge. This has to be accomplished manually by placing lines and specifying connectivity nodes at road intersections.

3.3 GIS in South Africa

In South Africa there is no overall GIS strategy or vision and an unfortunate consequence of this is that there is no uniformity with respect to either the information available or the GIS's currently in use [Anderson]. For example, if one considers Gauteng, one would find no fewer than five different GIS's in use by the various town councils [Quick:1995]. Furthermore, the councils are generally at different stages of information capture.

For the purpose of this project some specific propriety systems were evaluated, to ascertain all possible means of data extraction:

1. ReGIS
2. ARC/INFO
3. AutoDesk
4. SmallWorld
5. Framme
6. MicroStation GeoGraphics

ReGIS is used predominantly by the town councils while SmallWorld is now being launched by ESKOM. ARC/INFO is used by a few of the town councils. AutoDesk, FRAMME and GeoGraphics are well established overseas and are now being launched in South Africa. Of these and other propriety GIS's encountered, two classifications in their spatial data storage mechanisms were identified. These differences are illustrated by using two GIS packages that were selected for this purpose.

3.3.1 ReGIS

Many of the local town councils have opted for ReGIS as this is a relatively low-cost product [Weitzorrek]. ReGIS stores the spatial featurised information in a standard text file and an extract from that file for a stand is shown below.

```
F STANDS   STANDS) [3]     NONE16  1500230001600000
p2         93756.2656      2884865.4920    0.0000
p2         93760.2666      2884878.7550    0.0000
p2         93782.5599      2884871.7368    0.0000
p2         93782.5946      2884871.2350    0.0000
p2         93773.7758      2884855.7490    0.0000
p2         93756.2656      2884865.4920    0.0000
E
```

On the first line, the word starting on the third column (in brackets) describes the type of feature. The code stated at the end of the first line is a unique label or identifier used for referencing this feature. (The unique identifier is used to access information about this particular feature from other databases.) Vertices of the stand are listed with all vertices in a 3D co-ordinate system. Clearly, the last column represents the *z* co-ordinate which is not used in this research. The letter *E* at the end of the list of vertices is to explicitly indicate the termination of sequence. It can be seen that all the spatial information represented in the map is contained in a separate database and the unique identifier is used to maintain a relationship with thematic information contained in other relational databases. This GIS system is an example of complete external database storage mechanism for both spatial and thematic information.

For the purpose of this project some specific propriety systems were evaluated, to ascertain all possible means of data extraction:

1. ReGIS
2. ARC/INFO
3. AutoDesk
4. SmallWorld
5. Framme
6. MicroStation GeoGraphics

ReGIS is used predominantly by the town councils while SmallWorld is now being launched by ESKOM. ARC/INFO is used by a few of the town councils. AutoDesk, FRAMME and GeoGraphics are well established overseas and are now being launched in South Africa. Of these and other propriety GIS's encountered, two classifications in their spatial data storage mechanisms were identified. These differences are illustrated by using two GIS packages that were selected for this purpose.

3.3.1 ReGIS

Many of the local town councils have opted for ReGIS as this is a relatively low-cost product [Weitzorrek]. ReGIS stores the spatial featurised information in a standard text file and an extract from that file for a stand is shown below.

```
F STANDS   STANDS) [3]     NONE16  1500230001600000
p2         93756.2656      2884865.4920    0.0000
p2         93760.2666      2884878.7550    0.0000
p2         93782.5599      2884871.7368    0.0000
p2         93782.5946      2884871.2350    0.0000
p2         93773.7758      2884855.7490    0.0000
p2         93756.2656      2884865.4920    0.0000
E
```

On the first line, the word starting on the third column (in brackets) describes the type of feature. The code stated at the end of the first line is a unique label or identifier used for referencing this feature. (The unique identifier is used to access information about this particular feature from other databases.) Vertices of the stand are listed with all vertices in a 3D co-ordinate system. Clearly, the last column represents the *z* co-ordinate which is not used in this research. The letter *E* at the end of the list of vertices is to explicitly indicate the termination of sequence. It can be seen that all the spatial information represented in the map is contained in a separate database and the unique identifier is used to maintain a relationship with thematic information contained in other relational databases. This GIS system is an example of complete external database storage mechanism for both spatial and thematic information.

It was noted that some town councils are gathering block and kerb information [Weitzorrek, Anderson]. A block is defined as a polygon drawn around a group of contingent stands. Since, the regions of concern are informal settlements, unlike affluent areas, the kerb information is not necessary and assumed not to exist [Anderson, Gee:1997].

No road reserve information is being maintained by most councils [Anderson, Gee]. Block information is maintained and the space in between blocks is assumed to be roads, or rather public property. All the town councils do not necessarily conform to these practises unless they are working in conjunction with the Surveyor General. The surveyor general is considering including a centre-line to represent the road reserve in the future.

Data extraction can be accomplished by one of the two possible methods listed below.

1. Use the GIS environment to export the relevant information as a DXF map.
2. Write a program that will convert the external graphical database to a MAP file seeing that the formats are quite similar.

3.3.2 MicroStation GeoGraphics

GeoGraphics provides many processing features of which one is for breaking long lines into short lines so that they form part of a stand boundary (as explained with Figure 4). Stand boundaries are automatically identified using the stand labelling text node. In fact, the stand labelling text node and the stands do not have to be on separate layer in order to achieve accurate recognition. All the information associated with the stand is stored in a database of the user's choice [Neill]. However, MicroStation GeoGraphics is graphics driven and the stand boundaries are stored in their proprietary graphical database (DGN). Thus, the co-ordinates of the stand boundaries are not stored in an external database. Stand labels have a unique reference number which allows this association between the graphical feature and the external database to be made. If an area calculation is required then the stand identifier is used to identify the nearest surrounding lines as the boundary [Lawless]. The boundary identification process is repeated whenever the stand boundary information is required. This is an example of a partial external database storage mechanism. The spatial data is stored in an internal proprietary database which is part of the graphical user interface and is inaccessible by external programs.

Given the internal spatial data storage mechanisms, spatial information can be extracted using one of following two methods:

1. Use the GIS environment to export the relevant information as a DXF map.

2. Program a routine using a suitable GIS language for exporting spatial data in a specific file format, such as the MAP file.

4 Aerial Photographs

A large body of information has already been captured for well developed urban areas. However, the scenario is dramatically different when considering the more rural and undeveloped areas of South Africa. In such areas, like parts of Kwazulu/Natal, almost no cadastral information of any sort is available.

One manner of gathering cadastral information, involves making use of aerial surveys. Skilled photo-interpreters are able to identify houses, transport systems, crop types, drainage patterns and even structure height from high resolution aerial photographs. The aerial photographs are scanned and juxtaposed into a mosaic and attached as a reference file to either a CAD or GIS environment. The final image will appear in the background within the CAD or GIS environment and whenever a land feature is identified it is manually delineated by tracing (outlining) the feature. Graphical software, specifically designed for outlining features, are often utilised to aid the digitisation process. According to the Surveyor general, this method is not being used locally as yet but will be in the near future [Anderson]. Presently, field surveyors are being employed to gather township layout information.

Photo-interpretation is a laborious and not very challenging task for human workers. A PhD project being done by Levitt [1996a, 1996b, 1997] involves the investigation into and the development of a system which will have the potential for semi-automating the identification, delineation and mapping of certain land features which are visible from aerial photographs. The system is not intended to completely automate the recognition process, rather a user will be required to work in conjunction with the software, in order to achieve adequate interpretation of images.

Applying the digitisation methodologies discussed above will clearly result in the map being captured in either a CAD or GIS environment.

5 Hard copy maps

Some of the old town plans are only available as hard-copies. The need for an old map occurs particularly when extensions to an existing scheme may have to be made. In order to design the electrification scheme the maps must exist in digitised format. It must also be noted that the maps may be potentially outdated.

Digitisation is accomplished by scanning in a hard copy map in bitmap format, placing it on the background within a digitising environment (CAD or GIS) and then manually tracing over the drawing (treated similarly to aerial photographs). Additional tools are provided with most modern CAD and GIS packages to speed up the digitising process such as tools that automatically convert most of the map into vector format. There may be ambiguities and a user may be required to intervene to ensure sufficient accuracy. Finally, a land survey would be required to update the features on the map. The Department of the Surveyor General is using this process to convert all hard copies of old maps to GIS [Anderson].

Some of the hard copies of maps may contain electrical information. Such cases might occur when only a part of a township was electrified and extensions to the network now have to be made. A program was developed by Shapiro [1996] for identifying electrical elements from a scanned in map. This program will produce a list of co-ordinates corresponding to the location of identified electrical elements. Information provided by this program can be used to prompt the user for the specifications of the identified electrical elements. However, the program cannot identify cable routing or any of the stand or road boundaries. Therefore, this feature can only simplify a small task within the information capturing procedure.

6 Conclusions

This literature survey investigated the nature of various sources of maps and their data storage mechanisms. Of the four sources examined, it has been shown that both aerial photographs and hard copies of maps are digitised either into CAD or GIS environments for practical purposes. Thus only two sources effectively exist.

If CAD maps are the sources then they can be easily exported in DXF format. DXF is an ASCII text file format which implies that data can be easily extracted. However, eliminating map defects in CAD maps may be necessary but is time consuming thus requiring some form of automation.

GIS maps do not contain map defects as they are eliminated in the featurisation process. Some GIS's store the data in external databases in a similar format to the required MAP format and therefore extracting stand information from such files is trivial. It does require the development of various different routines for data extraction from the different GIS's that employ this complete external database storage mechanism. On the other hand, this data extraction methodology would prove a problem when faced with GIS's that use internal database storage mechanisms. Data can be forcefully created in a specific format, in this case MAP format, using the macro facilities within every particular GIS environment. Neither of these solutions are ideal since they require the development of various routines for every proprietary system. The necessary routines are not complex but creating one for every proprietary system will be time consuming process.

It would be desirable to utilise a common data extraction mechanism that is applicable to all GIS systems. Knowing that most modern GIS's are capable of exporting spatial data in a user defined format[3] in a DXF file, it would be easier to use this as a means of making spatial data accessible. Importantly, this technique is independent of the spatial data storage mechanism employed by the various GIS packages.

GIS map sources may be provided with centre lines which can be used to speed up the road polygon creation process. Due to the time required to create the road reserve, authorities involved in creating maps may not include this information. It is reasonable to speculate that if the road reserve information is included in the future, it will most probably just represent the main roads within the informal settlement and not the many pathways that typically exist. Thus, it is better to assume that the road

[3] The user defined format includes the ability to export certain layers only.

reserve information would not be available. Block information is also available but only in some instances and the same assumption as for the road reserve must apply.

To conclude, it is observed that of the four source of maps in existence, final representation manifests either in CAD or GIS format. Having narrowed the sources to two, it can be further noted that both sources are capable of exporting maps is DXF format. It would thus be logical to use DXF format as the single source for creating an intelligent map, as it implies that the solution to be defined needs to deal with one specific well defined input.

It is also noted that the only acceptable information that can be extracted from the DXF files are the stand boundaries and stand labels. The remaining information is stored using different techniques or not stored at all. Based on the fact that most institutions regard all regions that are not occupied as public property, it is reasonable to assume that the rest of the information can be extrapolated somehow to produce the complete "intelligent" map. Clearly, the extrapolation must involve generating the road polygons and cost regions.



ASED

Conceptual Design: "Intelligent" Maps

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iv

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Audience 1

1.4 Definitions 2

1.5 Applicable Documents 2

2 Outline of the proposed solution 3

3 Initial Map Procedures 5

3.1 Drawing Pre-formatting 5

3.2 Data Exportation and Extraction 6

3.3 "Cleaning up" Maps 6

3.4 Creation of Short lines 8

4 Feature recognition process 10

4.1 Stand Boundary Identification 10

4.2 Street Block Identification 12

4.3 Road polygon creation 14

4.4 Cost region creation 19

4.5 Workspace boundary breation 23

4.6 Intelligent map creation 23

5 Conclusion 25

6 Recommendations 27

6.1 The need for GIS 27

6.2 Specific problems is utilising a GIS environment 27

6.3 OLE Automation to solve the problem 28

Change History

Configuration Control

Project:	SADDIN
Title:	"Intelligent" Maps: Conceptual Design
Doc. Reference:	\1995_34\TP\TB315
Created by:	T Rajakanthan
Creation Date:	20 June 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	98\06\20	Draft	TR	\1995_34\TP\TB315.001
1.00	99\02\04	Draft	TR	\1995_34\TP\TB315.100

Revision History

Version	Date	Changes
0.01	98\06\20	New document created from TB002.100 (Document Creation Template)
0.01	98\06\20	Information from TB 302 and the paper TB 304 was cut and paste here as new document, while removing non-relevant information.
0.01	98\06\24	The background and specifications were removed from this document to create a new one (TB 313)
0.02	98\07\16	Editorial changes were made throughout this document.
0.03	98\08\22	Editorial changes were made throughout this document.
0.04	98\09\24	Changes as recommend by the supervisor were made.
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	MSR Project Minute Reference
1.00	99/02/04	Approved	TB 533

Change Forecast

Will be updated as necessary.

1 Scope

1.1 Introduction

This document provides a detailed description of the conceptual framework and the various mechanisms that are required to create an "*intelligent*" map. An explanation of some of the important algorithms that are involved are also provided. Alternative options that were explored are also briefly described. Where appropriate references are made to the Literature Survey (Doc no. A2).

This project was however not implemented as part of this research. However, part of this project was implemented, as a final year honours thesis at the Department of Electrical Engineering at the University of the Witwatersrand. An abstract of their report is presented in Appendix D.

1.2 Purpose

The purpose of this document is to present the conceptual framework of the process which is required to create an "intelligent" map. Understanding this document is prudent, since it lays the foundation for the high level design and low level design of the implementation phase.

1.3 Audience

The project manager and developer, members of the SEAL management board, the external examiner and all other interested parties.

1.4 Definitions

The distinction between the concept of the two types of boundaries which was discussed in Doc A1 is shown graphically in Figure 1.

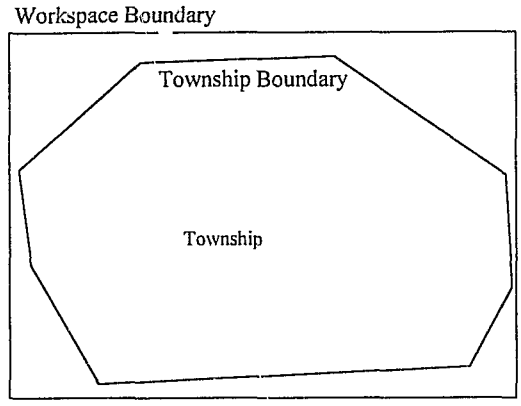


Figure 1 - Township and workspace boundary

1.5 Applicable Documents

1.5.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.5.2 References

Where explicitly stated all the references made in this document are listed in the *References and Bibliography* section.

Literature Survey (Doc No A2)

2 Outline of the proposed solution

The problem of creating an intelligent map was analysed and a solution was evolved. The steps that form part of the solution are shown in the flowchart in Figure 2. Details of the processes are described in the subsequent sections. Algorithmic descriptions are explained using textual and graphical means. The textual aspects reflects some of the important tactical concerns in the algorithm, while the flowcharts convey the sequence of events.

Maps for ASED can be produced on any CAD or GIS proprietary system provided that it can exported in AutoCAD™ DXF format. (It has been shown in the Literature Survey that aerial photographs and hard copies of maps are digitised into either CAD or GIS.) DXF format is the only map source that can be directly used for creating "intelligent" maps.

DXF files generated from CAD maps do vary in quality and require the pre-processing as explained in section 3, prior to the feature identification process described in section 4. On the other hand, DXF maps produced by GIS packages do not require pre-processing and the feature recognition process can be immediately applied. The reason for this is that GIS maps are refined prior to the featurisation process described in the Literature Survey (Doc No. A2). If for some reason a GIS map is of suspect quality then the pre-processing can be undertaken.

It should be noted that in order to create an "intelligent" map for use with ASED only the stand boundaries, stand labels and the township boundary are required. The map is then exported into DXF format. The software module described below, creates a complete map as defined in the manual creation process described in Doc No. A1.

Since this software module assigns user specified default cost factors for specific cost regions, the user is provided with an opportunity to make modifications to cost factors. There may be regions on the map that warrant the need for a different cost factor to those specified in the defaults. The user is then required to quickly modify those cost factors as appropriate. For this purpose, the map has to be loaded back into the CAD package and then re-exported in DXF format with the modifications. It should be noted that this is an optional procedure.

If the cost factor modification routine described above is used then the DXF to MAP file converter program (described in Appendix C) is used to finally create the "intelligent" map.

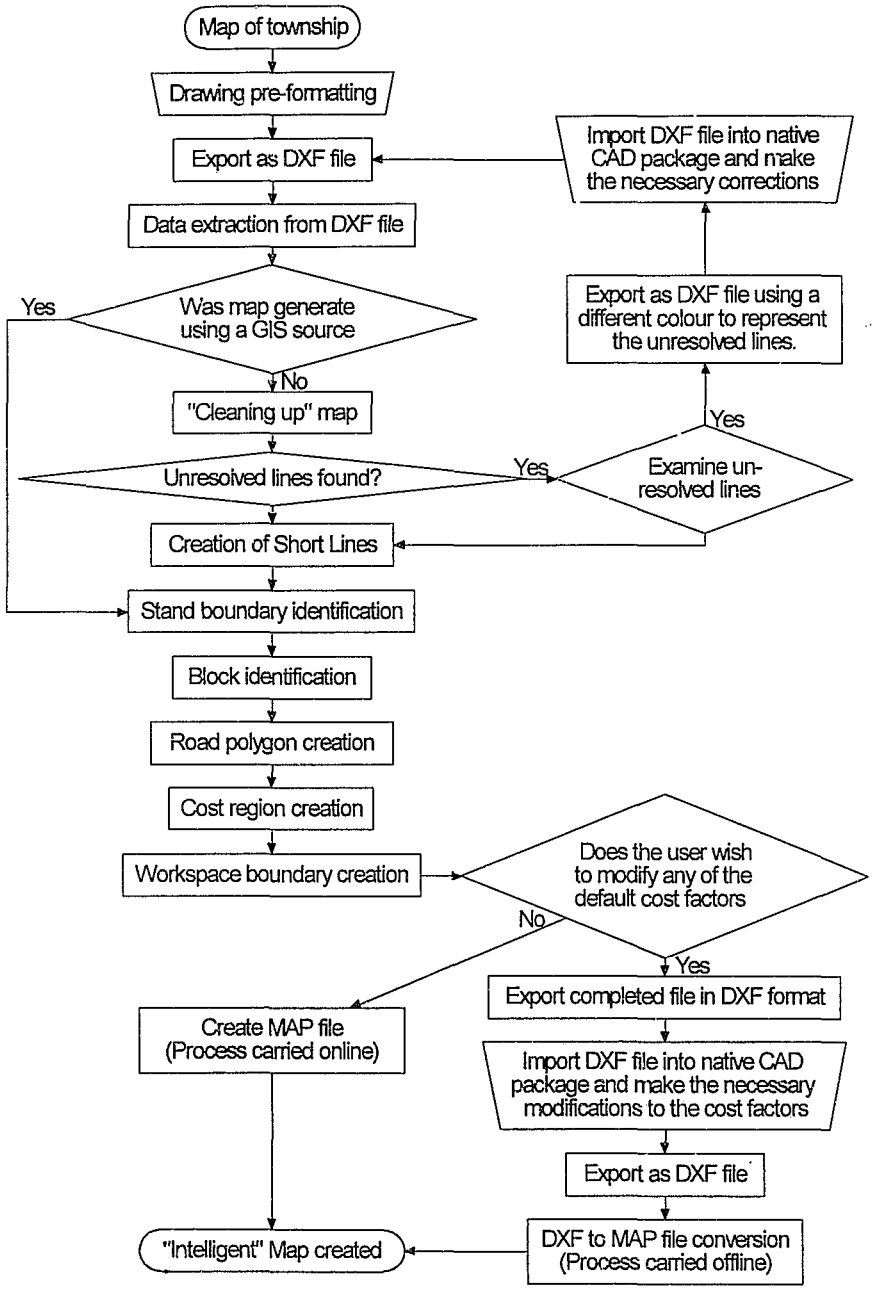


Figure 2 - Flowchart for creating an "intelligent" map

3 Initial Map Procedures

If the maps were produced specifically for ASED then the relevant features would appear in separate layers. In the general case where the maps have been produced on a CAD system by a third party, pre-formatting may be required as explained in the following sections.

3.1 Drawing Pre-formatting

The method used here is to manually segregate those groups of features necessary only for creating an intelligent map. Each individual group of features is located on a unique layer in the drawing file. While good drawing practise often stipulates that features with common properties are grouped together on the same layer, many of the CAD based maps supplied have many different features, most of which are not required to create an intelligent map, located on the same layer [Jager:1996]. This makes it difficult for computer software to extract the relevant information as it has no way of distinguishing between the various features. In such cases, the different features of interest represented on the map have to be grouped and manually moved on the screen to other layers using CAD commands. Some CAD packages provide colour separation facilities which could aid this process [Neill:1997]. For this research, it will be accepted that this is a restriction and labour intensive manual intervention would be necessary to relocate the features to separate layers.

The relevant features and the required procedures are described below:

- Using standard CAD commands the layer containing the stand boundaries has to be identified and labelled "STAND". On this layer of the town plan, lines that are incorrectly delineating stands must be deleted or moved to a different layer. However, lines that are not within the vicinity of the stands may remain as they will not be included in the stand boundary recognition process as described in section 4.1.
- The layer containing the stand text labels must be named "STANDNAME". If no labels exist, the stands must be identified by placing text labels, located approximately at the centroid of each stand. The location of these labels will be used later in identifying their respective stand boundaries (procedure described in section 4.1).
- The designer is also expected to define the township boundary as a closed polygon on a layer named "BOUNDARY". This boundary is required when creating the roads and cost regions as described in section 4.3 and 4.4 respectively. This boundary is also used by the transformer zoning program (described in Appendix B).

3.2 Data Exportation and Extraction

On completion of these steps, the map of the township is exported from its native format to an AutoCAD™ DXF format. Since DXF files are in text (ASCII) format, the following required lists can be easily extracted and compiled in the following lists:

- ListX- consists of all the lines and polygons located on the layer named "STAND".
- ListY- consists of all the stand labelling text and the co-ordinates of the text label node from the layer named "STANDNAME".
- ListZ- consists of all the vertices of the township boundary from the layer named "BOUNDARY".

3.3 "Cleaning up" Maps.

Digitised maps as drawn vary in quality which implies that line connectivity may not be accurate (map defects described in the Literature Survey). The following cleaning up operations are performed:

- Any two endpoints apart by less than a distance r (which has a default value of 1m but can be altered by the user) are assumed to be identical.
- Any line that falls short or overshoots a line by a distance less than r is adjusted to terminate on that line.
- All duplicate lines are eliminated.

Algorithms are developed for each of these steps to clean up digitised maps and these routines are executed (in the order that they appear). For these operations ListX (stand boundaries) is required.

3.3.1 Multiple-Endpoint eliminating algorithm

This routine calculates the distance between each of the co-ordinates of the line endpoints and every other line endpoint. If any of the distances are less than *r*, then those endpoints are merged into one common point. It should be noted that endpoints of the same line are never compared as it will result in the elimination of lines shorter than *r*.

3.3.2 Intercept Correcting Algorithm

In the original map, even those lines that initially were correctly terminating on other lines, may now not terminate correctly as a result of the multi-endpoint eliminating algorithm. This may be caused by the occurrence of a common situation as shown by the example in Figure 3. Suppose that Figure 3(i) shows the original case, where the vertical line is correctly terminating at the horizontal line at A. On the same drawing, the lines do

not intercept at point B. When the vertices at B are merged, as shown in Figure 3(ii), the vertical line no longer terminates correctly at point A.

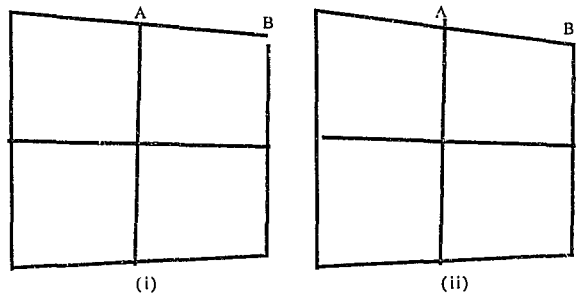


Figure 3 - Error induced by multi-endpoint eliminating algorithm

A new list (ListA) consisting of a copy of all the lines that do not have at least one of their endpoints coinciding with another is now created[1]. (Clearly, ListA is a subset of ListX.) The problem is approached in two steps.

1. The first step is to correct all the lines that overshoot the intended line. Each line in ListA is checked with every other line in ListX to determine any intercepts. Intercepts are assumed if the point of intersection, after the linear equations are solved, is between the endpoints of the line. The intercept point will replace the endpoint in ListX if it is less than a distance *r* from the endpoint. Every time a line is corrected, it is deleted from ListA.

2. Clearly, the remaining lines in ListA are those lines that fall short of their intended lines. All these lines are artificially extended by a distance of *r*. The process of extending these lines will ensure that an intercept will occur with the intended line, wherever one is expected. The process described in the previous step is once more used to determine any intercepts that may now exist and all the overshoots are corrected. All the corrected lines are again deleted from ListA.

3.3.3 User intervention

Naturally, there will be other lines that despite their extension may not have intercepted their intended lines or are non relate lines. These unresolved lines are those still remaining in ListA which need to be brought to the attention of the user for manual evaluation, as this process

[1] To ensure creation of closed polygons every relevant line must share both endpoints with other lines.

cannot be effectively accomplished through automation. However, the user is provided with the option of correcting or ignoring the unresolved lines. If the correction option is selected, a new DXF file is created (a DXF file creation engine was developed as part of the transformer zoning program) replacing all the original lines that were on the stand boundaries with the corrected ones. The DXF file can be loaded back into the user's CAD package for evaluation. All the unresolved lines will appear in a different colour so that the user can easily identify them.

Once those errors have been corrected, the file can be re-exported and processed in the same manner to arrive at this stage. The only difference in the second run of this program is that the option for correction when offered will be declined by the user. Unresolved lines may still exist in the second run but having analysed those lines the user can proceed confidently in the knowledge that those lines are not relevant. At the end of this operation, ListX will contain only correct lines.

It is important to note that for this manual correction process, a third party software is being utilised. A complicated windows based interface could have been developed as part of this software but it would be waste of time re-inventing a CAD type front end. Unfortunately, the utilisation of the third party software requires the process to be terminated so that the information can be ported to the third party software environment. Once the corrections have been made, the information has to be ported back into the program to resume the process. For the proposed version of this software, using a third party software is a more versatile option. (Refer section 6.3 on the recommendations for the most appropriate means of dealing with this type of problem in the next version.)

3.3.4 *Duplicate Line Eliminating Algorithm*

Although ListX only contains correct lines, it may contain two or more identical lines delineating the same boundary. The duplicate line eliminating algorithm will compare each line with every other line and if any matches are found then only one line is retained.

3.4 Creation of Short lines

Execution of the short line algorithm will break up all the long lines into short ones so that each of those define a section of the boundary for not more than two adjacent stands. (Procedure explained within the context of GIS in the Literature Survey.) Short lines are important for determining connectivity in the stand boundary identification process described in section 4.1

The procedure for creating short lines is as follows. Each line in ListX is checked with every other line for intercepts. Intercepts that are found to

coincide with the endpoints on a line are ignored. When a given line is found to have *n* intercepts between the endpoints then the long line is segmented into *n*+1 short lines and are added to a new list (ListB). If for a given line *n*=0 then that line is added to ListB, without segmentation. Thus ListB will contain a complete list of short stand boundary lines. The flowchart for this operation is given in Figure 4.

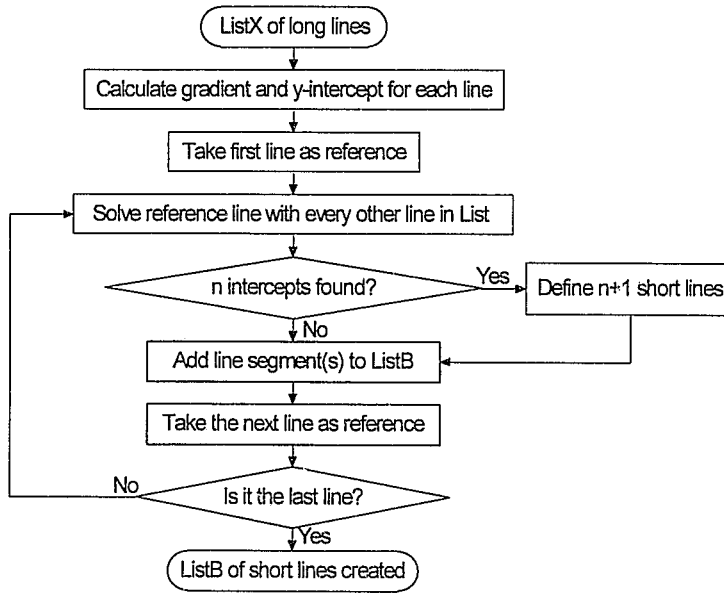


Figure 4 - Short line creation algorithm

4 Feature recognition process

This section describes all the processes that are directly associated with feature identification.

4.1 Stand Boundary Identification

This routine will identify stand boundaries given the consumer cable supply point which is the location of the stand labelling text node. This routine utilises ListB (list of short lines) and ListY (list of stand labels). The stand boundary recognition process is described with respect to a sample stand shown in Figure 5.

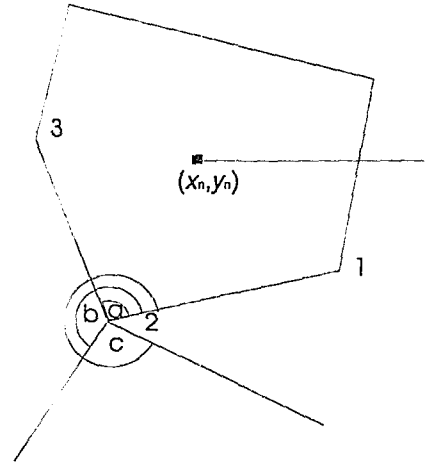


Figure 5 - A single stand

If a stand labelling text node is located at x_n,y_n, then an equation of a horizontal line ($y=y_n$) passing through that point is defined. Calculations are performed to determine intercepts of the line $y=y_n$ with all the lines in ListB. The intercept closest to x_n,y_n and having an x co-ordinate greater than x_n is found. That line is the nearest line, West of that point (x_n,y_n) and must therefore form part of the boundary. The endpoint of this line with the lower y value is now selected. In the example in Figure 5, vertex 1 has now been identified. The lower y value was chosen as in this instance the boundary lines are being identified in the clockwise direction.

Having identified a vertex (vertex 1), the program will look for other lines coincident at that endpoint. If only one such line is found then that line is assumed to be another boundary line. The other endpoint of the new line is used in the same fashion to determine subsequent lines. If more than one such line is found at an endpoint (as in the case with Figure 5, vertex 2) then the line subtending the smallest angle with the original line in anti-

clockwise direction (angle *a*) will be chosen as part of the boundary. These steps are repeated until the program once more finds the first vertex. The boundary lines for that stand would then have been identified.

It should be realised that the nature of this algorithm is such that other lines, not representing stand boundaries would not be detected as they would not have a stand labelling text in their vicinity. Furthermore, this algorithm will also effectively identify polygon boundaries that are concave.

An output list (ListC) will be produced with every element in the list containing the following information for each stand:

- the stand label,
- the co-ordinates of its text node and
- a list of vertices defining the stand boundary.

The information in ListC will be stored in a temporary data file. The flow chart for this algorithm is given in Figure 6.

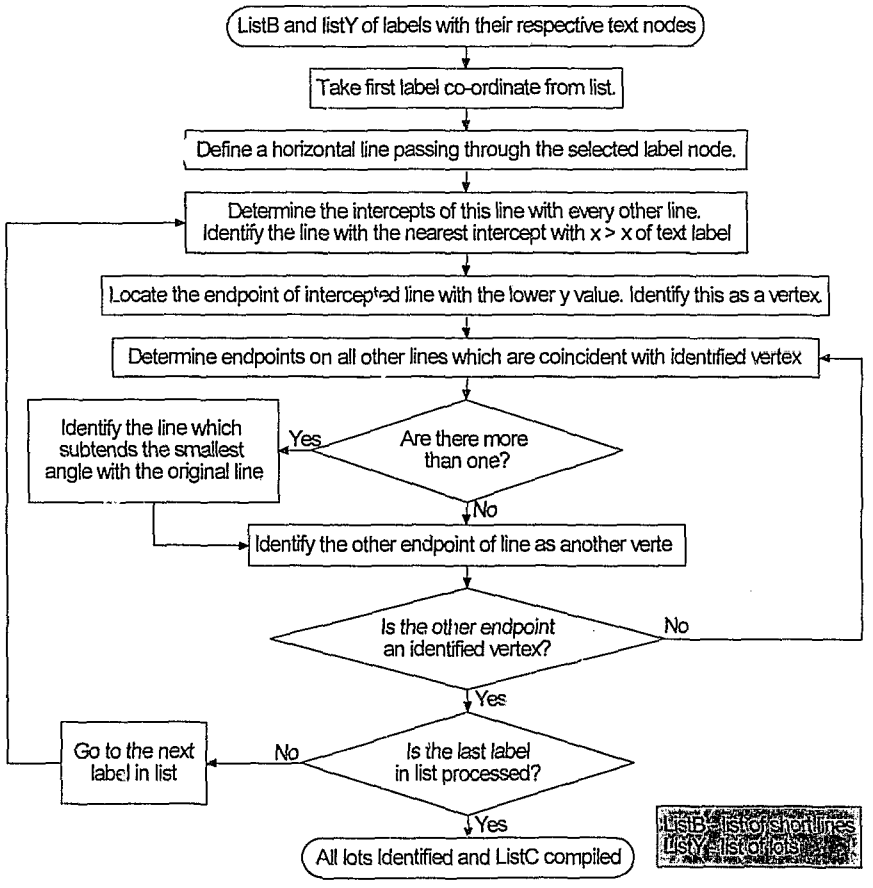


Figure 6 - Stand Identification Algorithm

4.2 Street Block Identification

While roads are often drawn on maps as separate polylines a short distance from the blocks, in this work all the regions between blocks are considered as the road reserve [Anderson:1997, Gee:1997]. The term block in this context refers to a polygon boundary encompassing a group of contingent stands. There will be no separate space for the kerb. This procedure is to simplify the complexity of the problem and is an acceptable assumption particularly if the areas being electrified are informal settlements. Stated another way, it is assumed that all regions or land that are not being utilised are suitable for cable erection unless otherwise specified. Techniques for specific region exclusion are described in section 4.4.

An algorithm has been devised to create road polygons and is described in the following section (4.3). In simple terms this algorithm breaks up all available region into polygons. The road polygon generating algorithm must be provided with spatial constraints so that road polygons are not created over stands. Rather than avoiding the individual stands the problem would be further simplified if the road generating algorithm was to avoid blocks. Thus a routine was developed to first create block polygons. The algorithm used to generate block polygons is now described. For this process 4the co-ordinates of the stand boundaries defined in ListC is required.

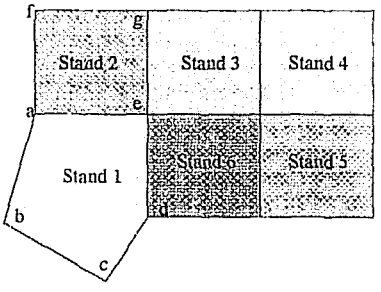


Figure 7 - Block of Stands

In the first stage, the algorithm compares each line in ListC with every other line. If a match is not found for a given boundary line then that line is added to a temporary list (ListD). If a line has no match it clearly implies that the it is not sharing a boundary with another stand and therefore is part of the block boundary. For example, in Figure 7, *cd*, *bc* and *ab* will form part of the block boundary (ListD). However *ae* and *ed* will clearly have matches and consequently both lines are ignored. ListD will now contain a list of block boundary lines. The flowchart for this process is given in Figure 8.

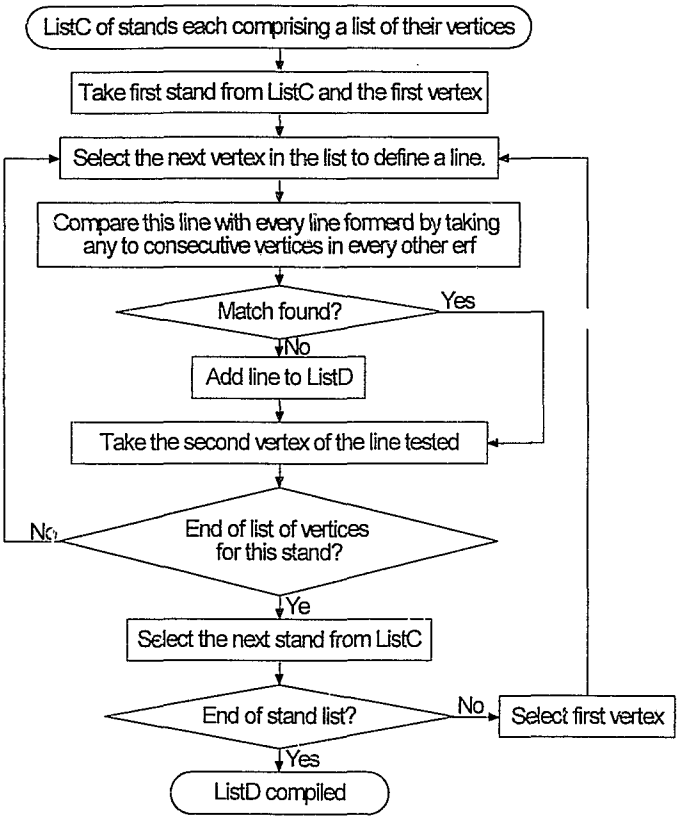


Figure 8 - Flowchart for compiling a list of block boundary lines

For the second stage, one of endpoints of the first line in ListD is taken as a reference and the remaining list is searched for a line with a coincident endpoint. Whenever such a line is found the other endpoint of this line used in the same manner to trace subsequent lines until the other endpoint of the reference line is found once more. When this occurs a block polygon has been defined and those vertices are appended to a new list (ListE).

Every time a line that forms part of the block boundary is identified, it is immediately removed from ListD. The process repeats again starting with the first available line in ListD. All block polygons would have been identified when there are no more lines remaining in ListD. All the block polygons are compiled in ListE. The flowchart for this process is given in Figure 9.

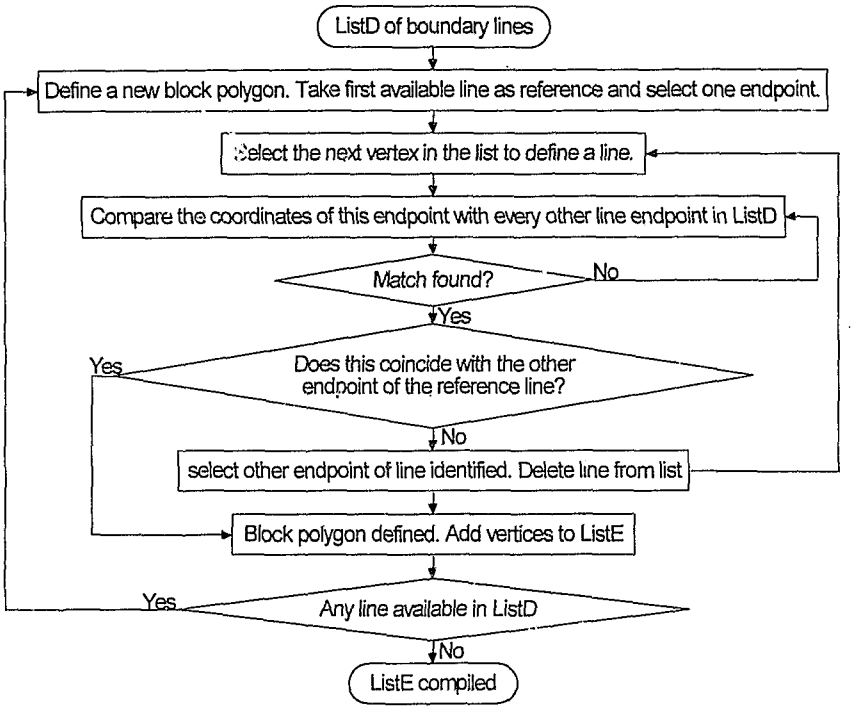


Figure 9 - Flowchart for creating polygons from a list of lines

4.3 Road polygon creation

This section describes how the area between all the identified blocks and the township boundary is defined as roads (reasons presented in the previous section 4.2) and compiled into a similar as list as for stands. Since there are no formal road names for the many pathways within the townships, unique alpha numeric codes will be assigned for each road polygon for reference purposes only. Roads that are leading to and exiting the township will be terminated by the township boundary.

In creating road polygons there is a simple geometrical constraint which needs to be overcome and is illustrated through the following simplified example.

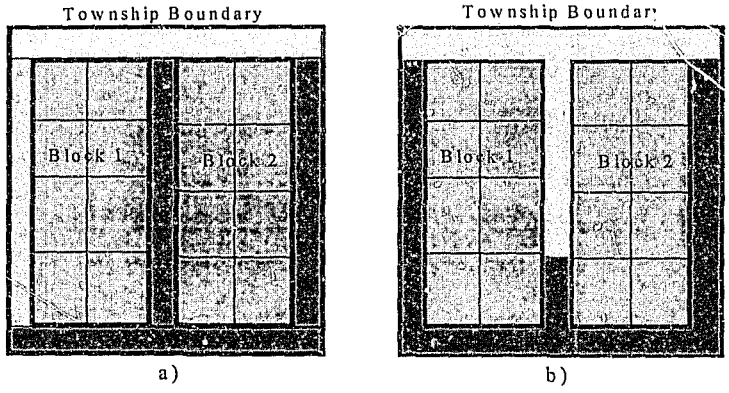


Figure 10 - Road polygon creation

Consider a small township consisting of the township boundary (shown by the outer rectangle) and two blocks, each containing eight stands as shown in Figure 10*a*. It can be clearly seen that valid road polygons cannot be created by merely tracing the sides of either the township or block boundaries.

Interconnecting lines are required, as shown in Figure 10*b*, in order for road polygons to be defined. The interconnecting lines required for the example in Figure 10*b* are the absolute minimum necessary to ensure that valid road polygons can be created. From analysing complex cases, it was established that for successful road polygon creation each block must have at least two interconnecting lines. This rule may result in greater interconnecting lines than necessary but it ensures the existence of sufficient interconnecting lines necessary for complete valid road polygon creation.

The algorithm for creating interconnecting lines is described from a single block perspective and must be repeated for each block. This process requires ListE (list of block polygons) and ListZ (township boundary polygon). However, this algorithm is based on the reasonable assumption that at least two blocks exist within the township.

Firstly, the shortest distances from a vertex on the block to a vertex on the township boundary is identified and defined as an interconnecting line. Secondly, another vertex on the same block is identified which is the furthest away from the previous vertex. From the vertex now selected, the nearest vertex is identified regardless of whether it is on another block or the township boundary. The second interconnecting line is defined between these two vertices. The reason for selecting the vertex that was the furthest away was to firstly minimise the creation of interconnecting lines on one side of the block thus reducing the effectiveness (as shown by interconnecting lines *a* and *b* in Figure 11). Secondly, this tactical

decision is more likely to force the interconnecting line on to the other side of the block, which is the desired effect (as shown by interconnecting lines *c* and *d*). It should be noted that Figure 11 does not show all the interconnecting lines and is merely being used to demonstrate certain cases.

The likelihood of interconnecting lines intersecting one or more blocks is high as visible from the example with lines *d* and *i*. All the lines that are extended to the township boundary are checked to determine interception with other blocks. When intercepts occur, the nearest block is identified and the nearest vertex on that block. In the case of line *d* as shown in Figure 11, Block 6 and its North-western vertex appears to be the nearest. The interconnecting line would then connect to the nearest vertex on the identified line as shown by line e, thus replacing line *d*. The same applies to line *i* which is replaced with line *j*.

A further simplification rule is that an interconnecting line can neither intersect nor share an endpoint with another as this can be ambiguous to the road polygon generating algorithm. The reason for this will become obvious from the description of this algorithm. Whenever two interconnecting lines end up sharing a vertex, one is transferred to the nearest *available* vertex either on another block or on the township boundary. For example, when an interconnecting line is being corrected and the result is line *f*, (as shown in Figure 11) then it is replaced with a line such as *h*. All the interconnecting lines are compiled in a new list (ListF).

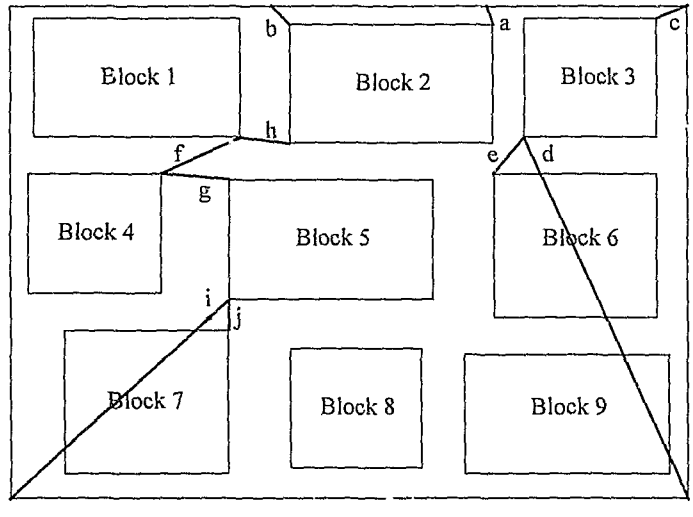


Figure 11 - Interconnecting lines between blocks

The flowchart for creating interconnecting lines is shown in Figure 12.

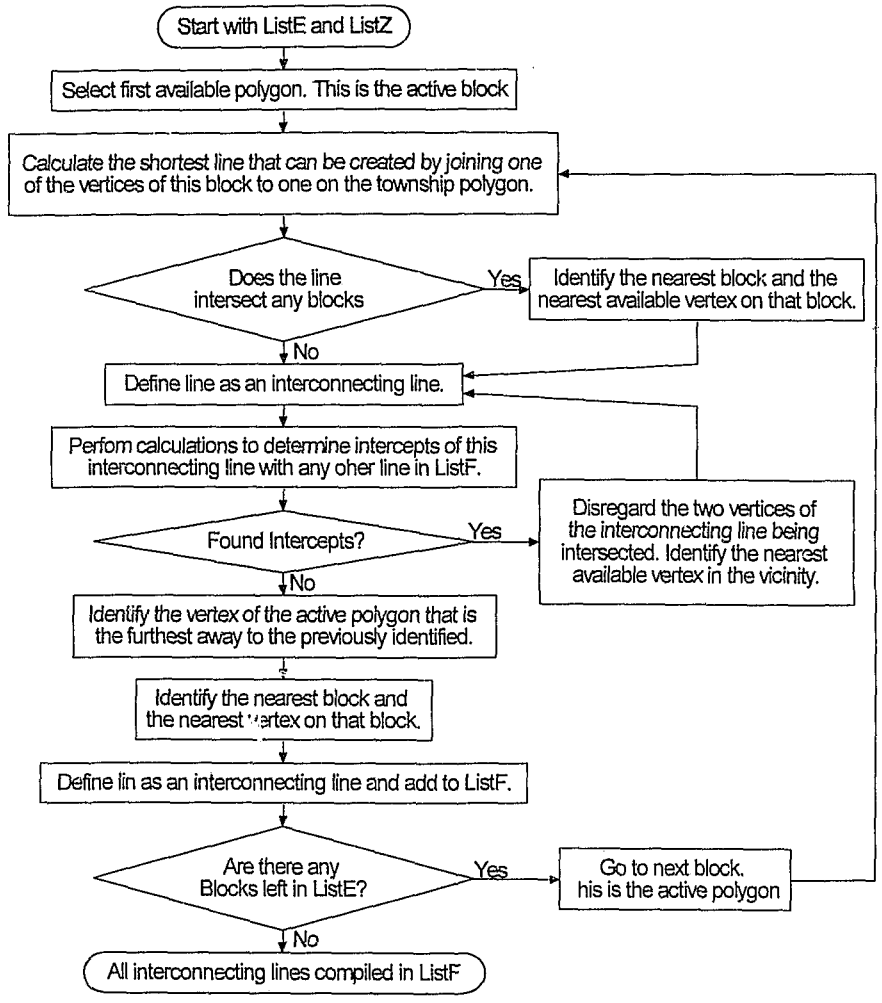


Figure 12 - Flowchart for creating interconnecting lines

Once the interconnecting lines have been created the road polygons can be traced and these are compiled in a new list (ListH). The road polygon creation algorithm uses the first available interconnecting line in ListF. It takes one endpoint as the reference and uses the other to determine its coincident vertex on either a block (from ListE) or the township boundary (from ListZ). When the coincident vertex is found the routine can trace a polygon in two possible directions (left or right). For tracing the first road polygon, the left direction is always used. When tracing subsequent polygons, the left direction is chosen only if it has not been used for creating a polygon in a previous trace. The vertices along that boundary are now traced checking every new vertex to see whether it coincides with an interconnecting line endpoint (from ListF). When an interconnecting

line is encountered, the tracing routine will take this route as it always takes precedence. Most importantly, the direction that the tracing routine must take when it reaches the end of an interconnecting line is the same direction as selected at the start of the trace. Failure to do so will result in incorrect polygon formation. A road polygon is created when the reference vertex is found once more.

When an interconnecting line has formed part of two road polygons, it is deleted from ListF. Therefore, all the road polygons would have been created when there are no more interconnecting lines remaining in ListF. ListH is now appended to the temporary data file. The flowchart for this operation is given in Figure 13.

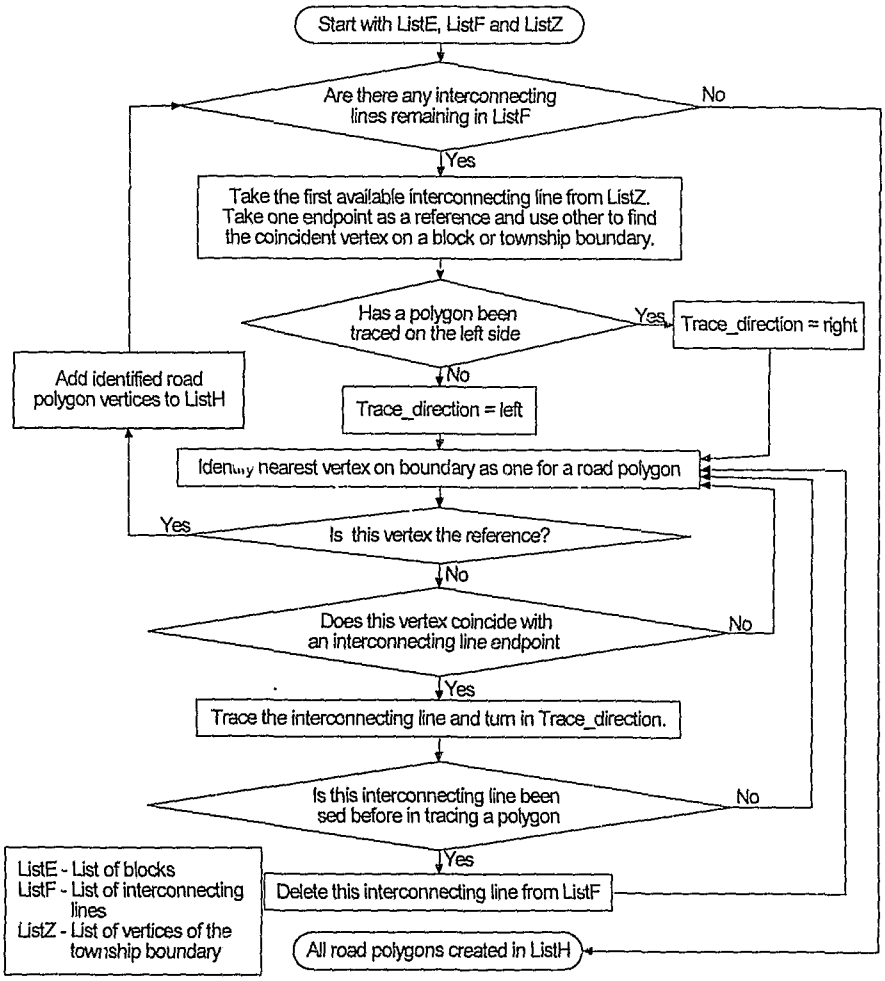


Figure 13 - Road Polygon Creation Algorithm

4.4 Cost region creation

The latter criteria is imposed since, it is not known whether that area is suitable for routing or not. Furthermore, in most cases that area (between the workspace and township boundary) would usually be off limits to the routing program.

The creation of the cost regions have been simplified since all the road and block polygons have been created. It is logical to assume that all cost regions will be copies of existing polygons defining the blocks and roads. Furthermore, the area between the township and workspace boundary must also be broken up into cost regions since the requirement is that entire workspace must consist of non-overlapping cost regions. This process is accomplished using the technique outline for road polygon creation. (i.e. creating interconnecting lines and definig polygons.) All the cost region polygons generated in this manner are stored in a new list (ListI)

It is important to realise that these cost regions may not necessarily be convex as required by the router. No precautions were taken in the creation of all these polygons to ensure that they were convex. Thus an algorithm was developed to break up every cost region polygon into convex polygons.

Assume that a polygon created could look like either one of these polygons shown in Figure 14. There are two cases shown to illustrate a tactical decision employed in this algorithm to demonstrate its flexibility when faced with different cases. It is important to state that while it is known that all the vertices are listed sequentially, the order in which the vertices are listed is not known. The vertices may be listed in a clockwise or anti-clockwise direction. Thus, certain precautions have to be taken in the convex polygon creation process.

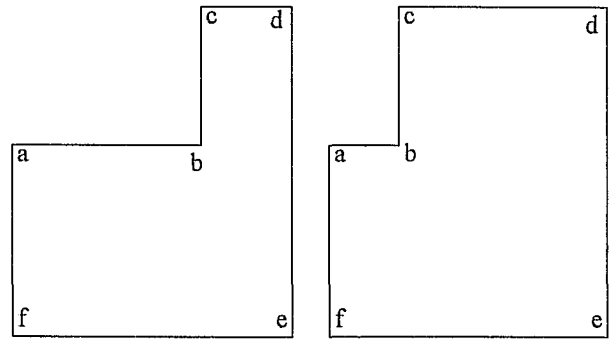


Figure 14 - Concave polygons

The first convex polygon is created by starting with the first vertex in the list, which in this case is *a*. The algorithm takes the first three consecutive vertices of this block which are *abc*. An imaginary line is drawn from the two extreme vertices (between *a* and *c*). Two points, each one hundredth the distance into the line from either vertex, are calculated and checked to see if they are within the block or not. (The reason for this will become apparent shortly.) This is accomplished using the scan conversion algorithm [Newman:1979] which can determine whether a given point is within a polygon or not. Since both of these points are out of the polygon vertex *b* is flagged and vertex *c* is temporarily ignored in the creation of this convex polygon. Flagging vertices such as *b* id done so that once the convex polygon being created is completed, another polygon can be created starting with one of the flagged vertices.

Vertices *abd* are now checked and operated upon. In the same manner as before, an imaginary line is drawn between *a* and *d* and the two points located on this line are checked to see whether they are within the polygon or not. In both cases, one of the points (near vertex *a*) will be located outside the polygon which implies that vertex d is also not suitable. The fact that the other point is located inside the block is irrelevant. The midpoint of the imaginary line was not used in determining whether it was within the polygon or not since it can be influenced by the shape of the polygon. Had the midpoint been used as a test, in case 1, the line would be located outside the polygon while the opposite situation would occur with case 2. The reason for choosing a point one-hundredth of the distance from the end is to ensure that it is sufficiently close to the vertex so that no errors occur as shown by the imaginary line *wz* in Figure 15.

Vertex *e* is now checked using the imaginary line test and is found to be acceptable. The next point (point *f*) is checked in a similar manner and again found to be suitable. When any valid point that is the fourth or more, an additional test is done to determine whether the angle subtended by the lines formed by the starting vertex, newly identified vertex and the previously identified vertex (in the example in Figure 14, $\angle afe$) is less than or equal to 180°. If this test is affirmative then the next vertex is processed and the same angle test is performed if that vertex is found to be valid. This process is repeated until the starting vertex is encountered. The convex polygon is closed. Other convex polygons are created by starting with one of the flagged vertices, (in the example in Figure 14 it would be vertex *b*).

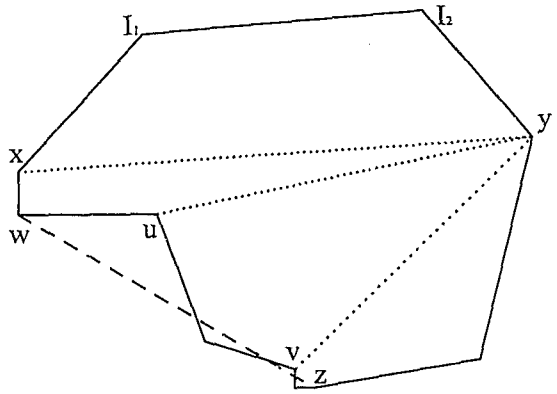


Figure 15 - Convex Polygon

If an convex polygon was created without ending back at the starting polygon (as in the case with polygon xl_1l_2y in Figure 15) then the last identified vertex is used as the starting point for subsequent polygon creation. The second polygon therefore starts at *y* and ends at *v* (including vertex *z*). The third polygon starts at *v* and ends at *u*. The last polygon starts at *u* and ends at *y*. In creating convex polygons as a first pass at a concave polygon, flagged vertices are not preferred. Flagged vertices are used in subsequent passes to create the remaining convex polygons.

It is important to note that all vertices that were identified in the interim are removed from the list to avoid confusion in subsequent polygon creation. An example of where an ambiguity mighty arise can be demonstrated while creating the last polygon *uwxy* in Figure 15. The polygon starts with *u*, proceeding to calculated *w* and *x* as acceptable vertices. However the next two successful points it will find are l_1 and l_2 which have been used in defining a previous polygons. Having deleted these interim points the next successful point to be found will be *y*, thus completing the correct polygon. The flowchart for the operation breaking up a single concave polygon to many convex polygons is complex and is given in Figure 16.

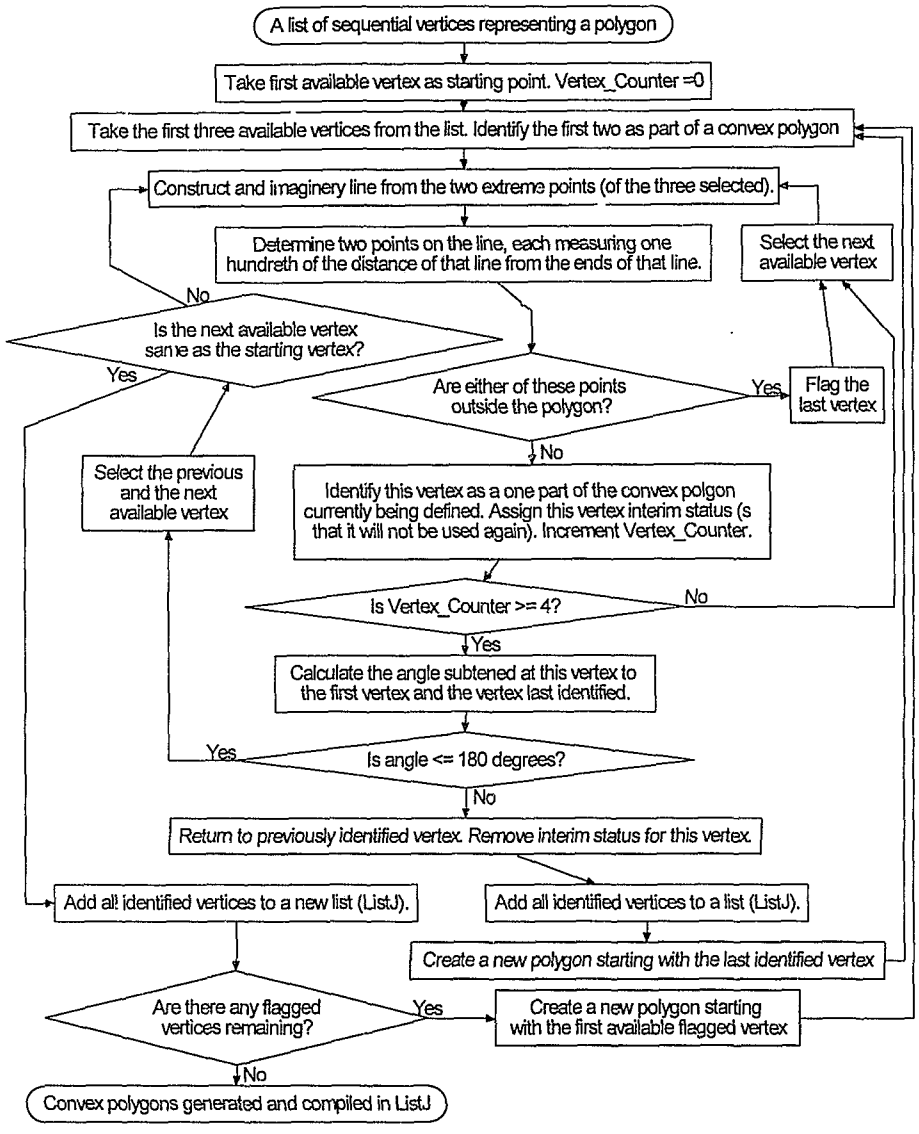


Figure 16 - Flowchart for creating convex polygons

This intense algorithm has to be repeated for every single polygon. This process is anticipated to be time consuming but is necessary for the correct operation of the router program.

Having established the cost regions it is important to ensure that no vertex is situated on a line (constraint explained in Doc No. A1). The problem can be graphically demonstrated as shown in Figure 17 where an additional vertex must be inserted at the location shown by the arrow so that it

coincides with the vertices of polygons B and C. The algorithm proposed to accomplish this is simple but unfortunately processor intensive in nature. It checks every line of every cost region for vertices that are located on it. When such cases are encountered, the lines are broken up to include an additional collinear vertex.

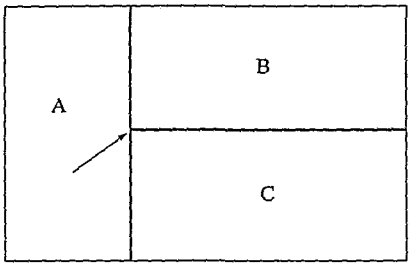


Figure 17 - Three rectangular cost regions

The final aspect is to assign cost factors to the cost regions created. The user is prompted for standard values for the three types of cost regions. All stands will be typically assigned a higher cost factor while roads are assigned lower cost factor. This will "encourage" the router to traverse low cost regions (roads) rather than the higher cost regions (stands). Sufficiently high cost factors will be assigned to the cost regions between the township and workspace boundary to ensure that these regions are not traversed, at all if deemed appropriate by the designer. It should be understood that the automated cost factor assignment process is merely to ensure that all the regions have a cost factor assigned to them.

4.5 Workspace boundary breation

Workspace boundary is created by simply evaluating the vertices of the township boundary to find the maximum and minimum co-ordinates. The user is provided with the option of specifying an additional clearance between the township and workspace boundary, which can be exploited if deemed necessary by the designer.

4.6 Intelligent map creation

Obviously, there will be stands or roads that are encompassed by cost regions that should have different cost factors for one or combinations of the reasons outlined in the specifications (Doc No. A1). Manual intervention may be inevitably required to modify certain cost factors as there is simply insufficient information available for the computer to generate accurate cost factors. The user is provided with the option of

choosing to make modifications or directly creating the MAP file. If the user is satisfied that the default cost factors are adequate then this program will proceed to create the MAP file from the data accumulated in the temporary data file.

However, if the user wishes to modify a few cost factors the program will produce a new DXF file is (a DXF file creation engine was developed as part of the transformer zoning program) replacing all the original features exclusively with the features required by the intelligent map. The DXF file must then be loaded back into the CAD environment where the designer is provided with the opportunity make the necessary changes.

Once the modification process is completed, the file is exported back into DXF format as before. It should be realised that the map is now in the exact state that it would have been had it been manually drawn specifically for ASED. Thus the program originally designed to create intelligent maps directly from DXF files (described in Appendix C) can be used to create the "intelligent" map.

5 Conclusion

The manual process of creating an intelligent map, involves the re-drawing of all relevant features as polygons within the CAD environment, while exercising certain precautions. This is a laborious and time consuming work which could offset the time saved by the optimisation routines that are to be incorporated within ASED. The redrawn map is then exported in DXF format and a conversion program (described in Appendix C) is utilised to create the *map* file.

Thus, the conceptual design for a software routine that can automatically re-format map features so that it can be used in defining an intelligent map has been presented. Precautions were taken in the development of the algorithms to ensure that they were not only efficient but also free of any obvious software bugs. All the conceptual algorithms have been described in sufficient detail, so that it can be used to effectively to accomplish the high and low level design. It is recommended that an object oriented design methodology be adopted in the implementation of this software routine.

In order for this software to operate a CAD file which has been exported in DXF format is required containing minimum information. The information required is the township boundary, stands and their respective stand labels. The software routine processes this information and creates the rest of the information to complete the map. The rest of the information defined includes the workspace boundary, road polygons and cost regions with their respective labels. Since all the cost factors assigned are default values as specified by the user, there may be certain regions that do not fit the default profiles. Thus, the user is provided with an opportunity to modify cost factors for certain regions if an area encompassed warrants it. If the user declines this opportunity to make changes, the MAP file will be created immediately. Otherwise, a new DXF file is created with all the newly defined features. This file must then be read back into any standard CAD package and modifications made to the cost factors. Once the modifications have been made, the map is exported in DXF format and the separate conversion program (described in Appendix C) is utilised to create the map file.

During the map cleaning up process (described in section 3.3) it is possible that some lines might not have been resolved. In order to ensure that these ambiguous conditions do not affect the remainder of the process, the user is alerted to this condition and offered the option of manual intervention. Should the user accept this offer, a DXF file of the map is created in its current processed stage, with the unresolved features emphasised using a different colour. The designer can quickly evaluate and modify the drawing in a standard CAD system and export it

back into DXF format. This program will then re-load the DXF file and then continue the processing.

The algorithms proposed is designed to create an intelligent map using minimum information. Stand boundaries and their respective labels are present in most commonly available maps sources. The availability of other information is not always guaranteed and that is the reason why this algorithm is not dependent on that information. This property allows greater flexibility with regards to the usage of different types of maps, whether it be CAD, GIS, aerial photograph or hard copy sources.

Though much of the processing is automated, a limited amount of user intervention is inevitably required to ensure that the "intelligent" map is created with sufficient accuracy. This accuracy is necessary to ensure the correct functioning of the optimisation routines.

As stated earlier, part of this design was implemented as an electrical engineering final year honours thesis project. An abstract of their project report is given in Appendix D.

6 Recommendations

6.1 The need for GIS

The methodology of designing a conversion system around the DXF format outlined above is sufficient for the foreseeable future. It is accepted that in the long term, pure spatial map information may be insufficient and more sophisticated levels of information required, to further improve the effectiveness of the electrification design process. As more and more information with greater accuracy become available, the development of more sophisticated optimisation tools would become useful.

It should be obvious that CAD systems are insufficient vehicles for conveying accurate thematic information for cadastral type maps. Sumic et al have stated [1995: Part1]: "*The GIS forms a repository from which relevant information is retrieved for design purposes. Therefore, any computerised tool aimed at automating electrical plat design must have provisions for operating within the GIS environment.*" An example of using a GIS would be the terrain modelling features provided which would make the terrain based router proposed by West [1996,1997], obsolete. More efficient routers would probably become available with the new GIS's as they evolve such as the automated primary router proposed by Yeh [1995].

6.2 Specific problems is utilising a GIS environment

However, the migration to GIS environments inevitably requires ASED to be customised for the various proprietary systems. This problem can be overcome through the use of a text file (with a well defined data format) containing relevant thematic information which can be used in conjunction with the MAP file by the optimisation routines to be developed in the future.

It must also be recognised that when the time comes to migrate to a GIS platform, all maps are likely to be in some GIS format or other for obvious beneficial reasons. Furthermore, an efficient form of GIS data exchange would hopefully be established (analogous to DXF format being used now) at that time and that format can be exploited to ensure that none of the major proprietary GIS users are excluded.

6.3 OLE Automation to solve the problem

6.3.1 OLE automation concepts

Holzner (1994:267) states that OLE automation is a powerful building block for the future of Windows based computing. Most modern applications are distinct self contained entities that provides the user with clearly defined functionality such as a spell checker. OLE automation makes it possible to integrate one program's function with those of another. Under OLE automation, both data and functions are shared between the OLE automation controller (clients) and server. When a OLE automation "exposes" certain data and functions, they become available to the controller. A controller can call the server's functions and use them, for example, an application can use a spell checker of another word processing application. Similarly, data can also be exposed which is then manipulated by an OLE automation controller. Exposed OLE automation data items are called properties and exposed functions are called methods (terminology defined by Holzner [1994:372]). The basic operating principles of OLE automation are shown graphically in Figure 18.

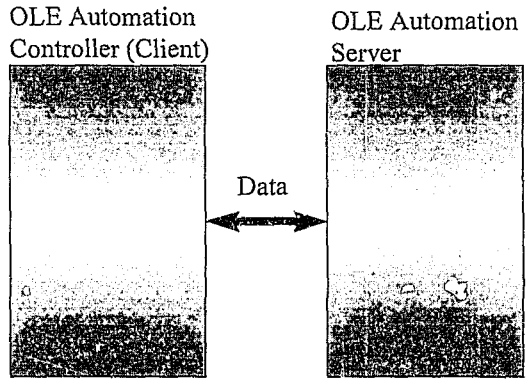


Figure 18 - OLE automation architecture

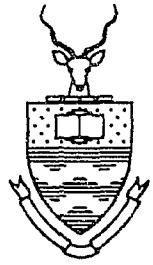
It should be noted that data can be exchanged in both directions. OLE automation communication is accomplished by actually writing to or reading from a common designated place in the physical memory. With the use of OLE automation, each application becomes a collection of tools, as opposed to singularly defined entities. The advantage of this is that it provides the user with greater flexibility and power enabling him or her to custom build applications using a variety such tools at his or her disposal.

6.3.2 OLE automation as a communication link

The majority of existing and certainly future GIS environments can be programmed to expose selected geographical and thematic data as OLE objects. OLE automation controllers, that will be incorporated within the optimisation routines, can then access and manipulate the exposed data. Additional data, that are calculated is then made available back to the OLE automation server so that it can be immediately reflected back to the user via the graphical user interface (GUI) of the GIS.

Thus, the use of OLE automation enables the creation of a seamless environment where the specific design tools, though not actually incorporated into the GIS system, appears to do so to benefit of the user. This communication mechanism is more elegant choice than the use of text based files, which are prone to accidental deletion by the user while performing other operations.

The drawback is that it requires some one to program the platform dependent modules of the program to identify and expose the correct layers of the map. This is important to ensure that only the relevant information is made available to the optimisation routines. That module must also be able to read back additional data produced by the optimisation routines so that it can be reflected to the user using its native GUI.



SADDIN

Problem Statement And Functional Specifications: Transformer Zoning

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Audience 1

1.4 Applicable Documents 1

2 Problem Statement 2

3 Specifications 5

4 Measuring Performance 6

4.1 Within Scope of Research 6

4.2 Long Term Evaluation 6

5 Conclusion 7

Change History

Configuration Control

Project:	SADDIN
Title:	Functional Specifications and Conceptual Design
Doc. Reference:	1996_34\TP\TB325
Created by:	T Rajakanthan
Creation Date:	31 August 1997

Document History

Version	Date	Status	Who	Saved as:
0.01	97\08\31	Draft	TR	\1995_10\TP\TB325.001
1.00	99\02\04	Approved	TR	\1995_10\TP\TB325.100

Revision History

Version	Date	Changes
0.01	97\01\10	New document created from TB002.100 (Document Creation Template)
0.02	98\01\05	A refinement algorithm was required and is described.
0.03	98\03\16	The most efficient way of determining the number of transformers to be used in a given area was finalised.
0.04	98\04\21	Changes as required by project manager was made
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	99/02/04	Approved	TB 533

Change Forecast

Will be updated as necessary.

1 Scope

1.1 Introduction

ASED (automated software for electrical distribution) is being developed for use within the drafting environment to optimise as much as possible the design of township electrification schemes. One particular aspect of the optimisation is determining the optimum transformer locations and consumer service areas within a given area of a township.

This document comprises of three sections. The background section provides the reader with an understanding of the manual practice involved in identifying optimum transformer locations, service areas and the inherent difficulties. This is the formal definition of the problem. The subsequent section presents the actual functional specifications which are derived from the problem defined in the former section. Lastly, a short discussion is provided on how the performance or effectiveness of this software can be evaluated.

1.2 Purpose

The purpose of this document is to essentially present the specifications and their reasons for their conception. In doing so, the motivation for undertaking this project is presented.

1.3 Audience

The project manager and developer, members of the SEAL management board, the external examiner and all other interested parties.

1.4 Applicable Documents

1.4.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.4.2 References

References and Bibliography (TB 140)

2 Problem Statement

Engineers, involved in the field of designing township electrification, often face the task delineating transformer zones and locating transformers at suitable positions. Delineation of transformer zones involves the determining of service areas within a township. Suitable transformer locations are defined as those that can feed the most number of consumers with the least cable lengths and thus the least cable costs. It is obviously difficult if not impossible to identify the exact optimum transformer location within practical time limits but it is necessary to get sufficiently close to that limit. Presently these locations are being determined manually by visual inspection for small areas and sometimes by trial and error for larger areas. In either case, it is standard practise to locate the transformers at road intersections (Figure 1) or mid-block (Figure 2) so that cables can be conveniently routed straight from the transformer in four directions.

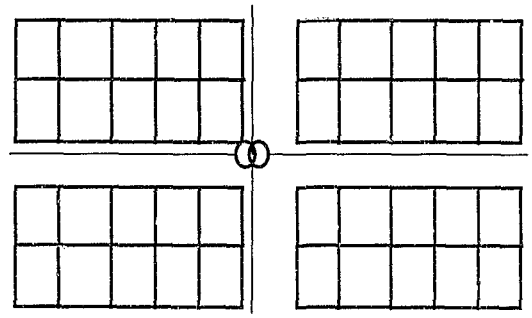


Figure 1 - Transformer located at an intersection

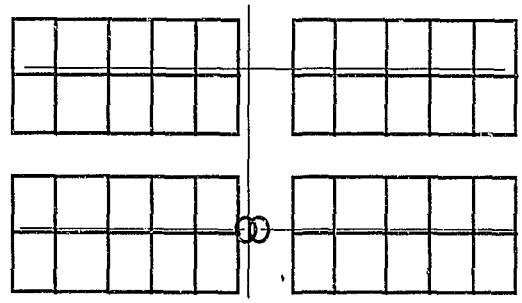


Figure 2 - Transformer located for mid-block routing

One of the methodologies currently employed by designers in determining transformer locations is outlined in steps below.

Determination of the total number of consumers in the township. This figure is usually provided with the map of the township. If this figure is unavailable, manually counting the consumers is the only way of ascertaining this figure.

Calculation of the After Diversity Maximum Demand (ADMD). The ADMD represents the simultaneous maximum demand of a group of homogenous customers divided by the number of customers and is expressed in kVA. Standard ADMD figures based on the general income of the township may be available to the designer and one is chosen for a particular scheme as deemed appropriate.

Selection of a transformer size. Most utilities standardise their use of transformers for obvious reasons. Typically a single model is used throughout the scheme and other sizes are used in cases where such a choice is economically justifiable.

Determination of the maximum number of consumers that can be connected by a single transformer. All transformers are capable of operating well above their rating for a short period of time. This is a very useful property of the transformer that is exploited, since peak demand is not longer than a few hours. Considering the overloading feature, the maximum number of consumers permissible per transformer is then calculated. The designer may actually choose a limit per zone that is less than the calculated maximum to allow for projected growth. This new figure is then regarded as the maximum number of consumers allowed per zone and will be referred to as the *maximum zone limit.*

Calculation of the minimum number of transformers required for a given township or part of it. Knowing the maximum number of consumers that can be supplied by a single transformer and the number of consumers that need to be supplied, the minimum number of transformers required for a particular scheme can be calculated. The following equation, which is self explanatory will yield the required answer.

$$\text{No. of transformers} = \frac{\text{Transformer rating + overload capacity}}{\text{number of consumers} \times \text{ADMD}}$$

Delineation of transformer zones. Knowing the number of transformers that need to be placed, the designer delineates the same amount of regions on the map using visual inspection. The designer then manually counts the stands within each zone. If any of the zones have more than the maximum zone limit then those customers are reassigned to adjacent zones as deemed appropriate. Sometimes a particular zone layout may be found to be inadequate, requiring greater number of transformers to be placed. The designer then has to start from scratch and repeat the zone delineation process. This is clearly a tedious and time consuming exercise.

Once the zones have been finalised the transformers are located at a convenient position at the load centre of gravity of each zone. It should be noted that once zones have been delineated the centre of gravity may not be obvious for the following reasons.

- Typical township settlements may have irregular shaped blocks.
- The blocks themselves are located irregularly within the township.

By using the term convenient it is meant that the position will probably be at a road intersection or as shown in Figure 1 and 2. It should be noted that often the locations of the transformers are further influenced by the exiting or planned MV line. Clearly, it is desirable to place transformers in such a way so as to minimise the distances to the MV line. This aspect takes into account the sharing of poles for both the MV and LV cabling. The transformer location may be further influenced by the main roads within the township as these are preferred routes for MV cables.

It should be noted that the methodology described above is fairly simple when working with maps that require 2 or 3 transformers. The degree of complexity increase substantially when considering areas that require more than 5 transformers to service over 500 consumers. Thus locations identified manually may be far from the optimum.

3 Specifications

The functional specifications as derived from the problem statement is as follows:

1. Delineate transformer zones and compute near optimum transformer location within a given area of a township. All the zones must be non-overlapping and must service every consumer in the entire area of the township.

2. Each of the zones must not contain more customers than the transformer can supply (maximum transformer capacity). The designer must be able to specify a zone limit that is less than the maximum transformer capacity.

3. For each of the delineated zones, the theoretically optimum transformer location must be computed and presented to the user based on the load.

4. A boundary must be defined for each zone allowing the designer to easily make a visual assessment.

5. This software must reduce transformer zone determination time significantly and produce improved zoning leading to the reduction of overall capital and operating costs. Alternate arrangements must also be produced for assessments, if necessary.

6. Provision must be made for in the design of this software unit so that it can interface with not only ASED but also with other similar software.

Furthermore, it must be feasible to use this unit to rapidly investigate the effects of increasing the number of transformers. Increasing the number of transformers to be placed in a scheme does not necessarily increase the cost. The increased use of transformers might lead to a reduction in cost since the total cable runs would be shorter. The ability for this unit to perform such an investigation so as to analyse the number of transformer versus total cable lengths could prove useful to a designer in ascertaining the most suitable number of transformers that should be utilised to further minimise capital costs.

4 Measuring Performance

As with the design of any system it is important to consider how one is to grade the performance of the system. The first aspect of this process is determining the validity of the operations of the software unit. In other words, determining if this product does what it was intended to do. This is a fairly simple task as it can be easily ascertained from visual inspection. If the formation of the zone boundaries are approximately correct, then the algorithms are functioning as designed.

The second aspect is evaluating how well this software unit performs with respect to the optimum requirements. One should realise that it is very difficult to grade the performance of this unit as there are no absolute designs to compare with, since there are no right or wrong answers. A possibility would be to do comparisons with designs done by consulting engineers or practitioners which may serve as an indication of this software unit's performance. However, as one would expect, transformer zoning could vary substantially from one designer to another. In the light of these difficulties two evaluation plans were devised and they are now described.

4.1 Within Scope of Research

Extensive testing will be done but not exhaustive due to practical time constraints. A sample of five townships will be evaluated intuitively, each with varying number of transformers and zone limits. For one suitable scheme which has already been completed, this software will be used to completely redesign a scheme using the generated transformer zones so that a comparison can be done.

4.2 Long Term Evaluation

A more useful way of measuring the performance is the degree by which this software unit aids designers, within a drafting office environment. Clearly, if the designers have to spend a substantial amount of time making significant changes to the zone layout then such a software unit is not useful. If, on the other hand, it offers the designer a good starting zone layout from which an acceptable solution can be evolved rapidly, then this tool is valuable. As stated earlier this aspect cannot be fully explored within the scope of this research but will be for commercial reasons.

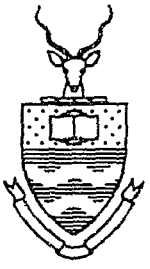
In addition, if it can be conclusively proven that the zones generated by this software unit leads to cheaper designs then the tool could be classified as really useful.

5 Conclusion

This document has presented the specifications for a unit that will be used for identifying optimum transformer locations and service areas within a given area of a township. It has been shown that the manual procedure for determining transformer zones is labour intensive and time consuming.

It is an accepted fact that good transformer zoning can improve efficiency in energy consumption and minimise cable usage. Thus, an efficient transformer zoning program is useful.

The Literature Survey is described in document B2 and the Conceptual Design is presented in document B3.



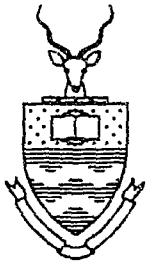
ASED

Literature Survey: Transformer Zoning

Management Product

Version 1.00

Document Status: Approved



ASED

Literature Survey: Transformer Zoning

Management Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Applicability 1

1.4 Definitions/Terminology 1

1.5 Audience 2

1.6 Applicable Documents 2

1.7 Assumptions 2

1.8 Requirements Traceability 2

2 Power reticulation concepts 3

3 Algorithmic Solutions 4

3.1 Review of existing techniques 4

3.2 Evaluation of algorithmic techniques 8

3.3 Description of the best surveyed Solution 9

4 Non-algorithmic solutions 12

4.1 Introduction to Artificial intelligence 12

4.2 Application of Expert systems in electrical distribution 13

5 Comparison between pure algorithmic systems and expert driven systems 17

5.1 Advantages and disadvantages of a pure algorithmic system 17

5.2 Advantages and disadvantages of a pure expert system 18

5.3 Knowledge based systems 19

6 Conclusions 20

Change History

Configuration Control

Project:	AUTOSED
Title:	Literature Survey
Doc. Reference:	D:\1996_34\TP\TB150.001
Created by:	T Rajakanthan
Creation Date:	12 November 1996

Document History

Version	Date	Status	Who	Saved as:
0.01	96\11\12	Draft	T R	\1996_34\TP\TB150.001
1.00	99\02\04	Draft	T R	\1996_34\TP\TB150.100

Revision History

Version	Date	Changes
0.01	96\11\12	Document was created from template TB002.100
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	99\02\04	Approved	TB 533

Change Forecast

Updates will be made as new material is covered.

1 Scope

1.1 Introduction

This project entails the application of computer based optimisation tools to automatically generate transformer zones as part of the design of electrical distribution networks. All previously published work on these and related subjects were reviewed through an extensive investigation. The findings of such a literature survey are discussed in this document.

It should be noted that the information deemed crucial for the development of the transformer zoning program are discussed in detail including relevant theories regarding the possible evolution of significant alternatives. Related information that is not crucial has not been included, although many books and papers were reviewed. Specific references are made to material discussing equivalent work regarding the subject matter at hand.

In this document a brief summary of electrical distribution is provided for the benefit of those people that are not familiar with the concepts and terminology. The main part of the literature survey is separated into two conceptual categories. Algorithmic solutions examines mathematical and analytical approaches while non-algorithmic means evaluates artificial intelligence techniques for solving the problem.

It should be noted that the terminology used when describing the papers will be native to those papers so that there are no ambiguities should they be referenced. However ambiguous terminology will be explicitly clarified in brackets so that the discussion is sufficiently clear in conveying the gist of the paper.

1.2 Purpose

This document presents the relevant findings from the literature survey.

1.3 Applicability

1.4 Definitions/Terminology

AM/FM: Automated Mapping/Facilities Management

GIS: Geographic Information System

HV: High voltage

LV: Low voltage: refers to two types of cabling often distinguished by its current carrying capacity. 1) The LV feeders are used to connect junctions to the transformer. 2) Service cables are used to connect the dwellings to the junctions.

MV: Medium voltage: referring to the cabling and its supporting furniture that extends from a substation typically to a transformer or transformers.

SCADA: Supervisory Control and Distribution Automation

AI: Artificial intelligence

Substation: HV/MV transformer.

Transformer: This refers to a relatively small transformer such as those used in rural electrification projects. Also referred to as a distribution transformer.

1.5 Audience

The project supervisor, namely Mr A Meyer

The project developer, namely T Rajakanthan

1.6 Applicable Documents

1.6.1 Specifications

1.6.2 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.6.3 References and Bibliography

All the references made in this document are in the *References and Bibliography* section.

1.7 Assumptions

1.8 Requirements Traceability

2 Power reticulation concepts

The fundamental principles of power distribution is summarised. The power aspects can be separated into Medium Voltage (MV) and Low Voltage (LV) domains (also referred to as primary and secondary distribution, respectively). Each of these domains can be further decomposed into numerous components.

In general, the MV starts at the power substations (HV/MV transformers) and extends all the way to the MV/LV transformers or load points scattered around in its service area. The definition of a service area is the region in which all load points are to be fed by a single substation/transformer. Similarly, LV starts at the MV/LV transformer and feeds numerous junctions within its service area. The junctions in turn supply the end consumers who would typically be located within the immediate vicinity of that junction.

In designing a distribution scheme, some of the tasks involved on the MV side include substation placement, MV/LV transformer placement, determination of the substation service area and the determination of the connectivity between the substation and the load points (ususally MV/LV) transformers. The tasks involved on the LV side include the determination of the service areas, cable routing, junction placement and determination of the connectivity between transformer and junctions.

It is also important to recognise that the procedures described above do not necessarily have to occur in that order, due to the numerous considerations that are involved in every action. For example, the transformer locations could be influenced by load centres and existing MV feeders.

Most of the papers evaluated dealt solely with the primary feeder routing as opposed to the secondary routing. However, routing between supply substations and load points (distribution transformers) in primary distributions is analogous to routing between distribution transformers and end consumers (load points) in secondary distribution. Voltage and current constraints are similar and therefore papers dealing in general with transformer/substation siting have been surveyed. However, there are some inherent differences which must be considered when applying MV techniques to solve LV problems.

Readers are alerted to the fact that the British refer to MV/LV transformers as substations while Americans refer to MV power stations as substations. All the work done adopts the American terminology. Those papers that use the British terminology will be indicated as appropriate.

3 Algorithmic Solutions

3.1 Review of existing techniques

Computerised techniques were applied to power distribution design from as early as 1959 by Knight et al. Using the computer it was possible to explore mathematical models for network design calculations, which were previously thought to be too time consuming to do by hand. Since then, a significant amount of work has been done to improve the network performance calculations, conductor optimisation and loss reduction of distribution designs.

There are numerous literature that deal with one or more aspects of distribution design optimisation. To the best of the author's knowledge, no techniques exist that automate and optimise all aspects of distribution design. It should be understood that all the literature discussed here refer to those that explicitly deal with transformer or substation siting.

Carson et al [1973] described a method that was developed to assist engineers in the design of low-voltage distribution systems. The method is based on an interactive use of a computer program that derives network configurations automatically, from a single-load flow computations. This methodology requires the specification of possible cable routes and possible substation sites.

Crawford et al [1975] proposed a technique that uses operations research methods to simultaneously optimise substation positions, determine size and service boundaries given alternative locations for substations and reliability constraints. In this methodology, a grid is overlaid on the map of the service area and broken up into sectors each measuring 0.22mi^2 (≅0.57km^2). Thus each sector is a load centres within this grid. Data requirements for this technique are the minimum feasible distance from each substation site or potential site to each sector. The load for each sector is known from the monthly billing data or predicted. Using the combination of Dijkstra's minimum path algorithm together with Ford and Fulkerson's transportation algorithm, substations are selected so that the feeder losses are minimised.

Hindi et al [1977] presented a branch and bound method operating in conjunction with a capacitated transshipment method for determining the optimal layout of a radial low-voltage-distribution networks. This technique requires the specification of all the optional substation locations. Costing aspects are modelled in the optimisation process.

Wall et al [1979] proposed a solution that uses a current fast upper bounded transshipment code to obtain radial network layout and

estimated costs. This model requires the possible feeder networks and the actual or potential transformer location.

Thompson et al [1981] used a branch and bound technique search method which utilises the shortest path table to obtain lower bounds and solutions from a transshipment linear programming models for upper bounds. This model approximates the load for a quarter mile area, requires actual or potential feeder networks and actual or potential sites for the substations.

Gönen et al [1981] developed a system planning tool using mixed integer programming to be applied particularly for determining optimum expansion planning. Pure-integer programming is used where all variables must take integer values; mixed-integer programming is used where only some variables are required to be integers [Adams:1974]. The algorithm requires the specification of existing substations, future (proposed) substations, demand centres, existing feeders and future feeders.

Kaplan et al [1981] presented a technique for assisting engineers dealing with the construction of new substations. The proposed technique limits the number of possible solutions using grapho-analytical methods to home in on the optimal solution. This technique requires a variety of data for the load points such as its location, its load profile[1], the necessary overhead and underground lines from the substation to the load, conductor sizes and the cost factors. Solutions are initially determined assuming idealised conditions which ignores topographical, land ownership and environmental considerations. When the solution has been generated, modifications have to be manually done to account for the local conditions.

Sun et al [1982] solved the optimal substation planning and primary feeder planning problem using the linear transshipment method. Again, this technique requires actual or approximated load densities and actual or potential transformer sites.

An algorithm for the static investment planning of large radial distribution systems using a branch-and-bound search mechanism is presented by Fawzi [1982, 1983]. The optimisation variables include number, rating and location of new substations [1983]. The number of substations required are calculated assuming all the potential sites to be occupied and eliminating the most expensive configuration encountered in every iteration until none can be further eliminated. The technique outlined is a

[1] The load profile includes the maximum power, power factor and the load curve over a 24 hour period for varying seasons.

non-linear optimisation technique that uses the left and right hand derivatives to determine the minimum of the cost function.

El-Kady et al [1984] presented a technique for optimal planning of substations and primary feeders. This technique focuses extensively on costing and is suitable for horizon planning studies. This technique requires the specification of actual or potential feeders and transformers.

Willis et al [1985a] evaluated some of the more popular computerised techniques which assist in the planning of distribution systems. They recognised that there was no system available at the time of publication that was capable of addressing all aspects of the distribution planning simultaneously as it would exceed the practical limitations of all known methods. Instead, most procedures attempted to solve only one particular aspect of the plan. Willis et al were of opinion that the siting models had an upper limit of 25 substations. Willis et al [1985b] then proceeded to define a method for the siting and sizing of substations which is highly oriented towards expansion planning. Amongst the vast amounts of data required, candidate sites for substations have to be specified.

Boardman et al [1985] described the application of a branch-and-bound method to a heuristic circuit-optimisation algorithm for electricity distribution planning. In the paper, the network model consists of a number of load and supply substations. The number of substations in the model are expected to grow and all the future geographical locations are known.

Gönen et al [1986] not only did an extensive review of the existing power distribution system planning models which had been published to date (of which all the related findings have been presented in this document) but also proposed their solution. Their technique works by minimising an objective function which was the result of a mixed integer programming model of the cost function. The cost function contains the variables that represent the cost of building a new substation and the proposed substation locations.

Ben Dov et al [1987] proposed an algorithm which finds the optimum size of cables, transformers and substations to yield an overall optimum reticulation system for a residential area. The cost function used is extensive including losses, maintenance and disturbance costs. The location and the sizing calculations for the transformers and substations are based on the assumption that the residential distribution is uniform (uniform housing density).

Ponnavaikko et al [1987] noted that most of the common techniques used for distribution planning (described above) are based on linear models whereby the non-linear cost functions of the substation and feeder arrangement have been linearised. In the technique that they have

proposed, the problem is formulated as an quadratic mixed integer programming problem. The problem is solved in two stages: firstly the quadratic programming problem is solved treating all the variables as continuous variables and secondly integerising the non-integer solutions obtained. The input to this model would be the potential substation location and the feasible feeder branch elements (conductor feeder routes).

Glamocanin et al [1993] proposed a tool aimed at assisting distribution system planning. This model focuses on improving the quality of supply, while minimising the cost in accomplishing that. The data required for open loop distribution planning consists of load data and description of existing and planned equipment. Thus, the possible substation locations and the potential feeder routes must be known in advance. Based on this information, the most suitable substation positions are obtained by using an iterative application of the minimal loop network algorithm for every available substation location.

Hsu et al [1995] proposed a heuristic algorithms for assisting substation planning problems, feeder planning problems and the sectionalising switch planning problem. In the substation planning problem, the locations of the substations are determined based on the specified number of substations. This is then followed by the determination of the feeder network based on the requirements of minimum investment cost and maximum available configurations in future system operation. Finally the sectionalising switches are placed at locations where feeder reconfiguration is most probable. The paper does not state whether the positions yielded are optimum or if any cost estimation features have been included.

Wong et al [1996] presented a feasible design method, using two mathematical techniques (Dijkstra's algorithm and Transportation algorithm) to optimise substation sizes and service boundaries given alternative locations for the substations, under reliability constraints.

Goswami [1997] reported of the development of a new algorithm based on the branch exchange technique for the planning of radial distribution systems. This technique has been implemented in two stages (i) to optimise the network configuration of each distribution zone and (ii) to find the optimal service area of each distribution zone. This technique requires the specification of all the probable substation locations and feasible feeder connections. It then assumes that all probable locations are valid and eliminates those that are two excessive in cost with each iteration. The process is halted when no more transformers can be eliminated. Thus the final solution has been reached where the number and locations of the transformers have been determined.

3.2 Evaluation of algorithmic techniques

Most of the algorithmic techniques described above have defined some form of an objective function or other to be minimised, containing variables that represent potential or future substation locations. When the processing is completed, those variables representing substation locations would have been assigned either a '1' or a '0' which represents the decision to build a substation or not.

The reason why most of the proposed techniques require the specification of potential transformer locations as opposed to leaving that an open ended problem has been inferred. In urban areas the potential sites are well established through master developmental plans as these sites have to cater for large transformers. In rural areas, the use of smaller transformers and, in most cases, pole-top transformers are suitable due to the low energy consumption of the consumers. Thus, techniques of specifying potential transformer sites in townships are not appropriate in rural electrification as there can be many potential locations, especially when considering the use of pole-top transformers. In most cases, many of the comparatively large number of existing cross roads within a township can qualify as potential locations for transformers. It would be better to have some technique that could propose ideal locations for the transformer, which can then be relocated to the nearest feasible location.

As stated earlier, most of these techniques described above relate to the primary side of the distribution and not the secondary side. The large amounts of data such as the specification of feasible routes for conductors are not practical to generate for rural areas to when dealing with hundreds of consumers, each having several routes to hundreds of potential transformer locations.

For most of the techniques described above, the security and quality of supply play an important role in the design of the distribution scheme. Thus, the algorithms are orientated to accommodate this primary condition. Quality of supply in rural areas are not as critical as in urban or industrial areas where there are many resources that require an uninterrupted power supply.

Willis et al [1985] performed a comparative evaluation between the most of the important techniques available at that time. They found that when the quality of the data was high, advanced methods were found to produce substantially better results over simpler methods. They also observed that as the data quality was reduced more advanced methods gave no better results than the much simpler methods. They also identified an upper limit to the number of substations that can be determined. In South Africa, the recent trend is towards the use of smaller

and greater number of transformers and therefore the number of cases where this upper limit is exceeded will not be uncommon.

It is appreciated that a substantial amount of work has been done by the international community and the criticism on the techniques proposed above are not in anyway indicative of their work. However, it must be understood that the problem definition is significantly different to urban electrification and unique to rural electrification, particularly on low cost schemes being built in third world countries.

During the survey, one solution stood apart from many of the other techniques, as it did not require the specification of potential sites. This technique has been described in the next section.

3.3 Description of the best surveyed Solution

Grimsdale et al [1960] propose a solution to calculate the initial transformer positions by an approximation procedure which ignores the road layout This involves the technique of locating *n* equi-spaced nodes in a Cartesian plane and will be described in the next paragraph. An iterative process is then used, which starts with the computed approximate substation positions. Consumer loads are first allocated to the nearest transformer, using the short (straight) line principle. The actual road routes are ignored. Better positions are located by moving the transformer from the approximate initial position to the calculated centre of gravity of load for each group of allocations (zones). The loads are reallocated to the nearest transformer and new centres of gravity are calculated for each zones. This process is repeated until there is no significant difference in transformer positions for two successive iterations (i.e. the change in distance for each transformer is less than 20m).

In order for this procedure to work initial substation positions are required that are sufficiently far apart. This is the problem definition of locating *n* nodes points (transformers) in a Cartesian plane (map of a township). Grimsdale et al proposed two methods for identifying initial transformer positions. The first is the square extraction technique and the second is the boundary division method. The concepts presented by Grimsdale et al are described in the following sections.

3.3.1 The Square Extraction Technique

According to Grimsdale description, the township area needs to be broken up into unit squares. Figure 1 shows the outline of the shape of the township with the area inside broken up into unit squares. The squares in this particular example measured approximately 63x63m (¼ acre unit area).

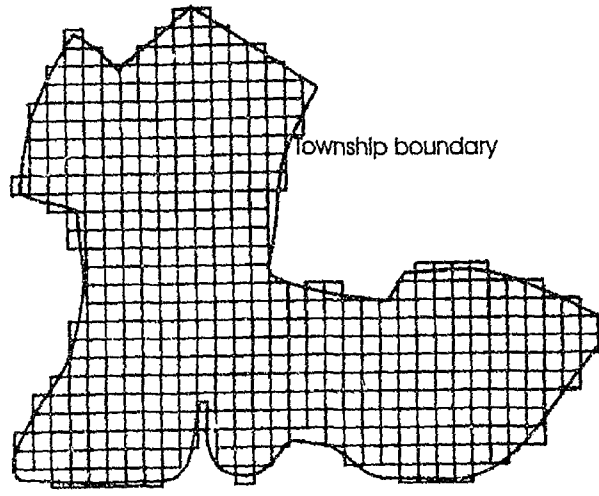


Figure 1 - Township map broken up into squares [Grimsdale: 1960]

Starting from the bottom right-hand corner and working to the left and upwards, the required number of transformer blocks (each measuring 3x3 unit squares) are positioned. In this particular example, five transformers are being placed. Each transformer block is increased by one unit square, until all the blocks can just be contained in that area. Figure 2 shows these steps.

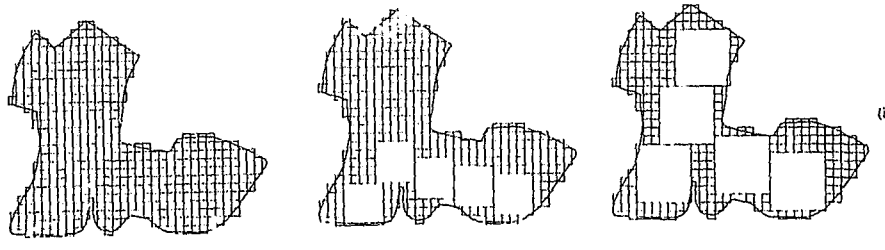


Figure 2 - Blocks of increasing size being placed [Grimsdale: 1960]

The centroid for each transformer block will be the initial location of the transformer. It can be clearly seen that for this example the initial transformer locations are sufficiently far apart. Supposing that the transformers were being placed in a township that resembles an elongated strip, then the transformers will be placed at one end of the township. However, the operation of the second part of Grimsdale's algorithm which deals with the consumer allocations will result in the

transformers migrating and being relocated at positions that are more evenly spaced.

3.3.2 Boundary Division Method

In this method the township border is scanned to locate five equally spaced points on the boundary. These points are then taken as the initial transformer locations.

3.3.3 Evaluation

According to Grimsdale et al [1960], a comparison of these methods indicated that the square fitting technique required far fewer iterations and produced fewer non-optimum solutions than peripheral starting points. Careful consideration of the logic in both techniques lead the author to the same conclusion and consequently that technique was utilised in this research. In order for either of these algorithms to operate, the user simply has to state the number of transformers that have to be placed.

This technique was the only encountered that starts off with a simple layout to produce *n* evenly placed nodes within the confines of the township. When this technique was tested it was found to be have a significant flaw (described in the Testing and Results (B6) document). The allocation process, operates without accounting for transformer overloading or under loading, resulting in a gross variation in the number of consumers allocated per zones. Townships in South Africa have been found to have the dwelling density varying from one area to another. This is primarily due to the fact that the housing settlements are informal and social restrictions determine land utilisation. The result was that the zones delineated, although moderately even in area, did not contain the same number of consumers. When trying to optimally design electrification scheme layouts, over loading or under-loading of transformers are unacceptable scenarios.

However the author was of opinion that it provided an adequate first draft at the solution and some other technique was required to improve the solutions to acceptable levels. Again, two possible methodologies were identified for improving the solution.
1. Use some form of algorithmic technique or
2. use non-algorithmic techniques to solve the problem.

Existing algorithmic techniques have been discussed above and the subsequent section evaluates non-algorithmic techniques. The fundamental principles of non-algorithmic techniques are investigated and some of the pertinent work that has been done up to date are investigated.

4 Non-algorithmic solutions

4.1 Introduction to Artificial intelligence

Non-algorithmic solutions refer to techniques involving artificial intelligence (AI), which is the field of study that is concerned with making computers behave intelligently. According to Gillies [1991], expert systems, neural networks, pattern recognition and robotics are all some manifestation of AI. Considering the techniques listed, expert systems have been identified as the most suitable regarding its application to solve this type of problem. The other techniques listed are involved in some form of pattern recognition which is clearly not applicable to an open ended problem such as transformer zoning.

The definition of an expert system is that it is a software system designed to mimic the reasoning of human experts in some particular field to provide solutions to real-world problems. An expert system typically poses and answers questions relating to information that has been taken from experts in a related field. Such systems utilise logic, decision making and knowledge processing to categorise, consult, analyse and diagnose.

Conceptually, an expert system consists of two major components; a *knowledge-base* and an *inference engine*. The knowledge-base may be further divided into a rule-base containing the reasoning knowledge (rules) and a database containing factual knowledge (data). The inference engine attempts to use the data together with the rules to infer further knowledge so that solutions may be found to solve particular problems. The inference process, or the reasoning mechanism, must be able to justify any decisions made which in some way models the human decision making process [Negotia;1985].

4.1.1 Database

The database contains a list of assertions which represents factual knowledge which is present in the system at a particular time. Typically a roster is maintained of all assertions that have been made regarding a particular problem. Assertions are dynamically added or retracted from the this roster as the solution to the problem progresses.

4.1.2 Rule-base

All the rules contained in the rule-base have the same basic format:

IF *if-clause* THEN *then-clause*

When the *if-clause* is true, the *then-clause* which may be an assertion or action, is triggered.

There are two basic types of rules. Deductive rules rely upon premises (*if-clause*), when true, proves a conclusion (then-clause) which are always assertions. Conversely, in reactive rules actions are triggered when premises are true which includes adding new assertions to the system, deleting existing assertions or executing routines that have nothing to do with the assertions.

4.1.3 Inference engine

Generally, expert systems use one of two inference methods. The goal driven process would start off with a hypothesis, assume that this hypothesis is true and then attempt to prove or disprove the validity of this assumption using rules. This technique is suitable for systems that employ deduction systems where rules are sought whose conclusions match an unconfirmed hypothesis.

On the other hand, data driven systems makes use of existing assertions to make inferences from rules and deduce new information. The data-driven, rule-based system starts with some initial knowledge (assertions) and tries all available rules over and over, making new assertions as it goes, until no more rules can provide a new assertion. The premises in the *if-clause* are used to identify appropriate situations for deducing new assertions, or to perform actions.

4.2 Application of Expert systems in electrical distribution

To put the information provided above in a practical context, a discussion is provided on institutions who are using expert systems to assist them with various aspects of the electrical distribution planning process. Suitability of these techniques are discussed within the relevant section.

4.2.1 Expert system in solution generation

In the past, expert systems have been used extensively in dealing with aspects of power generation and machines. Wong et al [1987] were the first to apply expert systems to distribution design, dealing with the automatic allocation of loads to the busbars of distribution substations. The rule-base for this expert system is compiled for use in areas that require ring circuits where the continuity of supply is of utmost importance. The constraints and rules are not suitable for rural load allocation where rings are not required and quality of supply though necessary is not crucial.

4.2.2 Expert system as a design tool

Shao et al [1991] use an expert system as a design tool for radial type secondary system, capable of accommodating a projected load growth over a specified time horizon. One of the powerful features boasted by this product is that it allows the designer to query the expert system as to "how?" or "why?" it came to a certain recommendation. This is a thorough system, which takes numerous constraints into account in the calculations. It does determine the transformer sizes but is not capable of determining suitable locations.

Rao et al [1992] have an enhanced system that not only allows the queries such as "how?" and "why?" to be answered but allows the designer to explore alternative scenarios by using the "what if?" option. The system is complex but based on the sample dialogue presented in the paper it is apparent that the modelling is accomplished for an evenly distributed area with an even load density. The output is restricted to the determinations of the number of transformers, the size of secondary line conductor and the size of the service drop conductor. In other words geographical data such as conductor routes or transformer positions are not calculated.

4.2.3 Expert system/algorithm hybrid as a design tool

Expert system, although valuable in its own right, is often an insufficient vehicle for solving power engineering problem. Knowledge based systems, which can be considered as marriage between the optimisation procedures of operations research and the artificial intelligence of expert systems (expert/algorithm hybrid), are preferred. This fact is true, in most of the pertinent research presented in this section.

Chen et al [1989] propose an expert system/algorithm hybrid for load allocation in expansion planning. This is accomplished using the heuristic rules in the knowledge-base together with the algorithms aimed at minimising power loss and investment cost which supports the inference engine to reach proper load allocation plans. This requires the specification of candidate substation locations and feasible conductor routes.

Hsu et al [1990] also used an expert system/algorithm hybrid for determining substation locations and feeder configurations of a distribution system. The location-allocation algorithm is employed where, given the service area of each substation, its position is determined such that the feeder losses are minimised. In order to determine the best location for the substation, the feasible routes from the substation to the distribution transformers, which takes into account physical constraints, are required. This algorithm is powerful but if the concept is adapted to transformers

and consumers (as opposed to substation and transformers) it will require all feasible routes from all the transformer to all the consumers in the township. Furthermore, the location allocation algorithm assumes a single figure as the unit resistance. In optimising the secondary cable runs it is practical to use various different cable sizes to minimise costs and improve efficiency. Thus, these constraints which have been qualified make this technique unsuitable for rural distributions design.

Ranjan et al [1996] have used a knowledge based expert system for distribution system planning. However, generalised algorithms are used for obtaining the optimal feeder path and the optimal location of substation based on minimum loss criteria. Conductor optimisation techniques are also employed as subroutines within the generalised distribution planning process.

4.2.4 Expert system integrated within a GIS environment

Sumic et al [1993a, 1993b] have developed an Intelligent Decision Support System (IDSS) for Automating the Electrical Plat Design (AEPD). The IDSS produces electrical distribution network designs via the application of artificial intelligence through which heuristic strategies and rules of thumb are automatically applied. The significant difference between other systems and this is that it operates within a geographical Information system (GIS) environment. The benefits of an expert system operating within a GIS environment is that it will have access to not only spatial information but also to thematic information such as load data. The paper points out that the reasons for not implementing such a system in the past was due to the fact a vast amount of data was required which may have not been available or have been expensive to gather. Furthermore, the data gathered may have been of suspect accuracy or difficult to model. A substantial portion of the GIS maps available in South Africa suffer from these shortcomings.

According to Sumic et al [1993a] the engineers are interested in finding satisfactory rather than optimal solutions simply because the cost of obtaining an optimal solution is unnecessarily prohibitive. The paper acknowledges that optimisation routines can still be used to solve some segments within the design process, but no optimisation routines have been actually incorporated into their system.

A complicated procedure is used to define transformer zones (clustering process). Prior to the clustering process, all sites that are suitable for transformer locations are defined during pre-processing. Each cluster starts to grow automatically from an identified origin such as the cul-de-sac or main street and the process stops before any of their numerous pre-defined constraints are violated. The transformer is automatically located at the nearest potential site which in turn is nearest to the centre

of gravity of that cluster. A great deal of information is necessary for the clustering process and for many other aspects of the design process such as all the possible equipment placement sites, routing corridors, street crossings and obstacles [1993b].

Lin et al [1993,1996] have also presented techniques for determining optimum substation locations, determining its service area and feeder planning based on the shortest path. The GIS provides them with an environment for building a database for both graphic and non-graphic facility data, that is utilised by these incorporated routines. Land use, value, zoning and as well as other physical restrictions are considered and accounted for in the calculations.

These technique are not feasible for use in rural areas, since sufficient non-graphic facility data is usually not available for the transformer or substation siting processes mentioned above. Gathering the required level of data would be prohibitively expensive to accomplish in the time available. Furthermore, as stated earlier, the specification of potential sites for some of the techniques is a tedious and time consuming task for designing engineers that work under time constraints.

Comparison between pure algorithmic systems and expert driven systems

This section compares the two techniques for solving the problem. Pure algorithmic solutions are created from operational research and their aim is to find an optimal solution to a given problem. On the other hand, the expert systems are based on artificial intelligence and it can provide answers as to "how?" and "why?" it obtained a particular solution.

5.1 Advantages and disadvantages of a pure algorithmic system

5.1.1 Advantages of a pure algorithmic system

A pure algorithmic technique, if designed and implemented correctly, will accomplish the task it was designed to perform to find an optimal solution to a well defined problem. Once the program has been provided with its input, it will proceed to process the information and to produce an optimal or near optimal solution without informing the user of its inner workings. The reason for not disclosing the inner workings, apart from protecting the copyright of the product, is because the information is often meaningless to the average user.

With the increasing availability of more powerful computers, computer models that were initially considered prohibitively time consuming to be implemented are being done so now. Optimisation procedures can be successfully applied to a variety of situations for which solutions are guaranteed, provided that a mathematical model has been correctly formulated to solve that problem.

5.1.2 Disadvantages of a pure algorithmic system

Sumic et al [1995a] point out that the number of possible design solutions that might satisfy a given set of spatial, technical and economical constraints, which define "the search space" are quite numerous for an averaged sized township. To apply conventional optimisation routines from operational research, a well defined objective function must be derived. It is difficult to quantify the numerous variables that comprise the objective function such as "customer satisfaction". Due to the ill-structured and open-ended nature of the problem, it is not possible to derive an objective function without sacrificing salient decision factors. Because of these facts, the application of pure operational research optimisation techniques is ill suited for solving the overall electrical distribution network problem.

The flip side of the advantage stated above is that should the user wish to know "how?" or "why?" a particular solutions was generated, an algorithmic technique will be unable to answer this query.

5.2 Advantages and disadvantages of a pure expert system

5.2.1 Advantages of a pure expert system

Shao et al [1991] has summarised the main advantages of using espert system based approach and they are listed below.

1. *It is easier to make incremental improvements as the rules are kept in a separate external database to the code.* New rules can be added or removed from the database as technology and design procedures evolve over time.

2. *It is easier for the system to explain what it is doing and why.* Since an expert system mimics the reasoning strategies of a real expert, it is possible for the system to backtrack and generate an explanation on how it reached a specific conclusion.

3. *It is easier for a human expert to determine what is incorrect or incomplete about the system's knowledge.* Since an expert system is capable of presenting its reasoning to the user it is possible to determine what, if anything, is wrong.

4. *It is easier to interactively use a human expert's abilities wherever the expert system falls short particularly in the development stage where the rule-base is still incomplete.* Since this system mimics a real expert, it can easily co-ordinate its efforts with the input from the user.

5. *It is much easier to automate the whole task as much as possible.* Once the performance of a specific task is evaluated and justified, the expert system can give good designs consistently, within a short period of time.

5.2.2 Disadvantages of a pure expert system

The most notable disadvantage is that they do not guarantee an optimal solutions but rather a satisfactory solution [Sumic:1993a]. An expert system is capable of intelligently narrowing the search space of that problem but can be compromised by the availability and accuracy of the data regarding that problem. The accuracy and applicability of the rules determine the degree of optimality of the solution.

Certain systems cannot be reduced to rules particularly algorithmic techniques. Thus, expert systems cannot solve all specified problems but provide an alternative method for the solution. Since an inference engine is not changeable, the performance of an expert system is highly dependent on the resourceful use of data and rules.

A vast amount of time would have to be spent on acquiring rules and compiling them in a database. Certain problems may not have so many rules but complex systems might require rules from many experts in the field. Time and effort is required to incorporate these rules carefully into the database, ensuring that none of them oppose each other.

5.3 Knowledge based systems

It can be noticed that the advantages of one system are the disadvantages of the other system and vice versa. Knowledge based systems embodies the advantages of the two aims. If an Expert/Algorithm hybrid is to be produced it is important to ensure that it embodies the advantages of both systems when a combination is created. Otherwise, one may be faced with the disadvantages of both disciplines.

6 Conclusions

The literature survey has presented three possible techniques for solving the problem, namely algorithmic, expert system and a algorithm/expert hybrid (knowledge based system). An expert system is rarely useful in its own right unless it incorporates some algorithmic optimisation routines.

A significant number of algorithmic techniques were evaluated and were found to be unsuitable for rural electrification in Africa. The majority of the techniques required the specification of potential substation or transformer sites, in addition to feasible feeder routes. Due to the increasing use of small pole-top transformers, any of the large number of cross roads or other suitable locations within a rural settlement can qualify as potential transformer sites. Furthermore, due to the existence of numerous pathways within a township, the specification of all feasible routes to all the potential transformer locations a prohibitively time consuming task. The reason for these shortcomings is that the majority of the work done by the International community focuses on the primary side of the distribution and not the secondary.

It should be obvious to the reader that it would be far better for the computer to calculate the most optimum transformer locations so that the designer can make the creative decision of identifying the most suitable locations. One algorithm, proposed by Grimsdale, was found not to require the specification of potential sites and is able locate any number specified transformers within the township limits. The algorithm has been described in this document. It will be shown that this solution is inadequate as it was originally designed for urban areas that have a fairly uniform consumer distribution. Judging by the numerous maps of townships available in South Africa, the consumer distribution varies significantly from one area to another.

Numerous, expert and knowledge based systems were investigated but, in general, were found to require a substantial amount of information in order for it to be able to perform some calculations. Typically, the level of information required, may not be readily available or expensive to generate manually, such as terrain and obstacle data. For this reason, many of the expert systems have been incorporated within GIS environments so that access to the required information is simplified. In accomplishing this, their systems operate on specific proprietary GIS environments, thus alienating many potential users.

After evaluating all the available techniques, it was concluded that Grimsdale's algorithm was simple, reliable and adequate at obtaining a 'first draft' of the solution. It is able to delineate zones within a township. However, devising a suitable technique for improving the solution

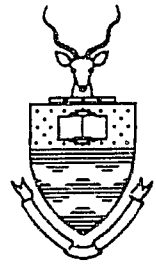
generate by Grimsdale's algorithm was necessary. The solution can be improved using algorithmic, artificial intelligence or expert/algorithm hybrid techniques.

According to Fox [1990] "*if an algorithm exists that optimises the solution, use it. On the other hand, for those problems that cannot be solved using optimisation algorithms, AI will provide a satisfactory solution but it does not guarantee an optimal solution.*" An algorithmic technique is preferable compared to the AI techniques due to the numerous advantages that it presents (described is section 5). Both algorithmic and knowledge based systems would nevertheless be investigated to determine its feasibility in being used to solve the transformer zoning problem.

Designing a complex algorithm is not necessarily desirable. Based on the comparative evaluation done by Willis et al [1985] when the quality of the data was high, advanced methods were found to produce substantially better results over simpler methods. However, as the data quality was reduced more advanced methods gave no better results than the much simpler methods. In South Africa and in most third world countries it is safe to assume that the quality of the data will be poor.

A important criticism on some of the previous work done was that optimisation was based on models linearised from non-linear cost functions [Ponnavaiko:1987]. It is important to ensure that any algorithmic technique developed must not suffer from this flaw as this problem is not linear.

The Conceptual Design document (B3) discusses the proposed solution in detail.



SADDIN

Conceptual Design: Transformer Zoning

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iv

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Audience 1

1.4 Definitions 1

1.5 Applicable Documents 1

2 Limitations and Assumptions 3

3 Interface: Input 5

3.1 Geographical Data Input 5

3.2 User input 5

4 Grimsdale's Algorithm 7

5 Solution Advancement 10

5.1 Proposed technique 10

5.2 Alternative techniques considered 14

5.3 Further refinement process 16

6 Drawing the boundary polygon 19

7 Interface: Output 22

7.1 Geographical Data Output 22

7.2 Output of Results 22

8 Costing 26

8.1 Costing versus zone limit variation 26

8.2 Costing versus use of more transformers 26

8.3 Cost Estimation Formulas 27

8.4 Determining cost profiles 30

9 Conclusion 32

Change History

Configuration Control

Project:	SADDIN
Title:	Transformer Zoning: Conceptual Design
Doc. Reference:	\1995_34\TP\TB328
Created by:	T Rajakanthan
Creation Date:	10 January 1997

Document History

Version	Date	Status	Who	Saved as:
0.01	97\01\10	Draft	TR	\1995_34\TP\TB328.001
1.00	99\02\04	Approved	TR	\1995_34\TP\TB328.100

Revision History

Version	Date	Changes
0.01	98\04\20	New document created from TB002.100 (Document Creation Template)
0.01	98\06\02	information from TB 330 was cut and paste here as the conceptual design was separated from the high level design.
0.02	98\07\29	The boundary definition section was added.
0.03	98\08\13	Minor revisions were made throughout the document.
0.04	98\10\07	Changes to the section cost determining formulas, section 8.3.
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	MSR Project Minute Reference
1.00	99\02\04	Approved	TB 533

Change Forecast

Will be updated as necessary.

1 Scope

1.1 Introduction

A detailed description of the conceptual framework and the various mechanisms that constitute the transformer zoning program as implemented is described in this document. It attempts to address some of the difficulties associated with developing this program. A high level description is provided with a brief explanations of some of the important conceptual algorithms and design architecture and hierarchy. Alternative options that were explored and are also briefly described. References are made to the Literature Survey (B2) and the reader is expected to have perused the document prior to reviewing this document.

Prior to the description of the conceptual design, all the important assumptions and limitations have been recognised and discussed.

1.2 Purpose

The purpose of this document is to present the conceptual framework of the transformer zoning program. Understanding this document is prudent, since it lays the foundation for the high level design and low level design of the implementation phase. It should be noted that the this document describes the basic steps that need to performed to achieve an adequate solution and in no way dictates the design or implementation methodology.

1.3 Audience

The project manager and developer, members of the SEAL management board, the external examiner and all other interested parties.

1.4 Definitions

Zone limit

Maximum zone limit

1.5 Applicable Documents

1.5.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.5.2 References

References and Bibliography (TB 140)

2 Limitations and Assumptions

Prior to defining the system, it is important to make some assumptions and identify limitations. To accurately determine the transformer locations, true cable lengths along practical routes from the transformer to every consumer have to be determined. This involves the use of a router module such as the one developed by West [1996a, 1996b, 1997]. The complexity of the algorithm employed makes the use of this module a time consuming process. (The router module developed by West, given a starting point, an ending point and additional data as described in TB 313 within a map of a township will be able to determine the optimum conductor route.) Thus for every suitable transformer location identified, the time taken to compute all the practical routes in order to identify the cheapest configuration would be prohibitively time consuming. In order to simplify the problem, an assumption is made that the transformer location would be the centre of load gravity of the customers its serving. The centre of load gravity will be calculated as the straight line distance from the transformer to every consumer in its service area. Clearly, this method is not accurate but the assumption is further based on the premise that total straight line distances have some determinable relationship to the actual distance, supposing that cables were routed along practical routes. An additional fact is that most townships have well interconnected pathways which can be used for conductor routing. The relationship between the straight line distances and actual conductor distances will be discussed in section 8.3.

The transformer positions calculated using this methodology may be unsuitable, for instance if it is located within privately owned or utilised premises. The designer is then expected to use judgement and experience to relocate these transformers to the nearest viable positions. Viable positions are dictated by the unique layout of every township and influenced by existing or proposed medium voltage cable routes. This aspect cannot be automated as there are no set rules for determining a transformer location within a township. In a problem domain where there are no clearly defined rules, it is impossible to create a system that can generate satisfactory solutions.

Thus these assumptions presented above have reduced this problem purely into a spatial domain without any constraints that will have to be imposed if practical distances were to be considered.

A further assumption is that only a single sized transformer can be specified for any given execution of the program. Alternative sizes can be specified in subsequent runs. This simplifies the solution search space and reduces computation time. As one would expect the rules that would be required to successfully determine the correct transformer size and the corresponding zones are numerous, as it becomes an open ended

2 Limitations and Assumptions

Prior to defining the system, it is important to make some assumptions and identify limitations. To accurately determine the transformer locations, true cable lengths along practical routes from the transformer to every consumer have to be determined. This involves the use of a router module such as the one developed by West [1996a, 1996b, 1997]. The complexity of the algorithm employed makes the use of this module a time consuming process. (The router module developed by West, given a starting point, an ending point and additional data as described in TB 313 within a map of a township will be able to determine the optimum conductor route.) Thus for every suitable transformer location identified, the time taken to compute all the practical routes in order to identify the cheapest configuration would be prohibitively time consuming. In order to simplify the problem, an assumption is made that the transformer location would be the centre of load gravity of the customers its serving. The centre of load gravity will be calculated as the straight line distance from the transformer to every consumer in its service area. Clearly, this method is not accurate but the assumption is further based on the premise that total straight line distances have some determinable relationship to the actual distance, supposing that cables were routed along practical routes. An additional fact is that most townships have well interconnected pathways which can be used for conductor routing. The relationship between the straight line distances and actual conductor distances will be discussed in section 8.3.

The transformer positions calculated using this methodology may be unsuitable, for instance if it is located within privately owned or utilised premises. The designer is then expected to use judgement and experience to relocate these transformers to the nearest viable positions. Viable positions are dictated by the unique layout of every township and influenced by existing or proposed medium voltage cable routes. This aspect cannot be automated as there are no set rules for determining a transformer location within a township. In a problem domain where there are no clearly defined rules, it is impossible to create a system that can generate satisfactory solutions.

Thus these assumptions presented above have reduced this problem purely into a spatial domain without any constraints that will have to be imposed if practical distances were to be considered.

A further assumption is that only a single sized transformer can be specified for any given execution of the program. Alternative sizes can be specified in subsequent runs. This simplifies the solution search space and reduces computation time. As one would expect the rules that would be required to successfully determine the correct transformer size and the corresponding zones are numerous, as it becomes an open ended

problem with a large number of suitable solutions. The time taken for a designer to sift through these to identify acceptable or practical solutions would in itself be a time consuming process, not mentioning prohibitive computation times. In addition, the complex algorithm which re-selects alternative transformers sizes and re-delineates zones could possibly lead to the non-convergence of a solution.

A final assumption for this research is that when experimenting with the placement of increased number of transformers, the transformer sizes will not be decreased correspondingly. In other words the testing will be done for varying quantities of a specific transformer rating.

3 Interface: Input

The input to the transformer zoning program has been separated into two categories since they are processed in different ways.

3.1 Geographical Data Input

The geographical data is essentially the representation of a map of the township. Usually, there is vast amounts of information but only that which is relevant must be input. As the software unit for creating "intelligent" maps is incomplete, some of the data extraction algorithms was employed to extract data directly from the CAD file. More specifically, a DXF file (Drawing eXchange File format) is accessed directly to extract the information listed below. If an "intelligent" map format is the source (as described in documents A1) then again only the relevant data would be extracted.

1. A list of the stand text labels and their respective co-ordinates of their text nodes are required. A CAD drawing file contains vector based features and therefore all text have a co-ordinate associated with them defining their location. These locations are assumed to coincide with the consumer supply point. The term *consumer supply point* is defined as the geographical location where the power cables are extended to and typically connected to a low cost distribution box (distribution board). The activities done to extend the power supply from this point into the dwelling does not concern the supply authorities. Since it is common practise to place stand labels within their stand boundaries, actual stand boundaries are not considered. It is accepted that the concurrency between the assumed and the actual supply point may vary. However, when considering a township with many consumers it is anticipated that the overall effect would be negligible.

2. A polygon defining the township boundary is also required. This boundary is to be utilised by Grimsdale's algorithm as explained in section 4. The polygon literally represents a boundary which traces the outline of the township. Accuracy in drawing the polygon is not prudent as it is being used merely for initial approximation purposes.

3.2 User input

A DOS based interface was developed for user input as it was the easiest to implement. Complicated windows based interfaces could have been developed but it would not aid in proving the concepts at hand and would have been time consuming. However, the interface structure must be versatile so that only the interface classes need to be modified in the

3 Interface: Input

The input to the transformer zoning program has been separated into two categories since they are processed in different ways.

3.1 Geographical Data Input

The geographical data is essentially the representation of a map of the township. Usually, there is vast amount of information but only that which is relevant must be input. As the software unit for creating "intelligent" maps is incomplete, some of the data extraction algorithms was employed to extract data directly from the CAD file. More specifically, a DXF file (Drawing eXchange File format) is accessed directly to extract the information listed below. If an "intelligent" map format is the source (as described in documents A1) then again only the relevant data would be extracted.

1. A list of the stand text labels and their respective co-ordinates of their text nodes are required. A CAD drawing file contains vector based features and therefore all text have a co-ordinate associated with them defining their location. These locations are assumed to coincide with the consumer supply point. The term *consumer supply point* is defined as the geographical location where the power cables are extended to and typically connected to a low cost distribution box (distribution board). The activities done to extend the power supply from this point into the dwelling does not concern the supply authorities. Since it is common practise to place stand labels within their stand boundaries, actual stand boundaries are not considered. It is accepted that the concurrency between the assumed and the actual supply point may vary. However, when considering a township with many consumers it is anticipated that the overall effect would be negligible.

2. A polygon defining the township boundary is also required. This boundary is to be utilised by Grimsdale's algorithm as explained in section 4. The polygon literally represents a boundary which traces the outline of the township. Accuracy in drawing the polygon is not prudent as it is being used merely for initial approximation purposes.

3.2 User input

A DOS based interface was developed for user input as it was the easiest to implement. Complicated windows based interfaces could have been developed but it would not aid in proving the concepts at hand and would have been time consuming. However, the interface structure must be versatile so that only the interface classes need to be modified in the

event that this software unit is required to communicate with any other system. The inputs that the user is expected to provide for this software unit are:

1. The name of the CAD file being read (source file).
2. The name of the CAD file that the appended information will be stored in (see section 7.1 for more details).
3. Specifications of the number of transformers to be placed in the township.
4. Specification of the zone limit (maximum number of consumers per zone).
5. Specification of the direction in which the software unit will be scanning (concept explained in section 5.1). This feature is to be used extensively in the testing phase to determine the sweeping behavioural characteristics. Based on the data obtained it would be possible to empirically evolved an efficient algorithm for the scanning process (described in section 5.1).This aspect will be automated when this product eventually becomes commercially available.

4 Grimsdale's Algorithm

Grimsdale's algorithm constitutes of two phases; namely square extraction and consumer allocation. Implementation details for both techniques were omitted from the paper but the basic principles have been described. Even if implementation details had been provided, it would have probably been procedural based (assertion based on the publication date of this paper) which would have been cumbersome to implement. Thus, a means of efficiently accomplishing this task was devised. Alternatives were evolved but none significant.

The conceptual framework for the implementation of the square extraction technique is now described. The township area must be broken up to contain squares (as shown in Figure 1). In order to do this the following two parameters will be required:-

1. the township boundary which is a polygon defining the shape of the township (geographical input) and
2. the number of transformers that have to be placed (user input).

It was established that the most efficient way to represent the squares on a map was to represent those as elements in a matrix. The location of a particular element in the matrix represents a specific square at the same relative location with respect to some common point of reference on the map.

In order to represent a matrix, a rectangular boundary must be superimposed onto the township. To do this, a temporary rectangular boundary must be computed so that the township boundary polygon can just be contained within as shown in Figure 1. This rectangle, referred as the *matrix boundary*, can now be divided into squares, each measuring 10x10m, starting from the bottom left corner. This allows the creation of squares in a systematic and regular fashion. If an entire square does not fit within the opposite end of the matrix boundary then they are omitted. For example if the township boundary is 1245m wide then only 124 squares can be placed ($1245 \div 10 = 124.5$). Knowing the height and width of the boundary, the size of the matrix required to hold the sampled data can defined.

The size of the squares used in this program are smaller than the ¼ acre (63x63m) size used by Grimsdale within the township. One reason for this is that the computation time is not affected significantly due to the availability of fast machines. The other reason is that better resolution is required in cases where the transformers zones being created are much smaller than those required for urban electrification as originally designed for by Grimsdale.

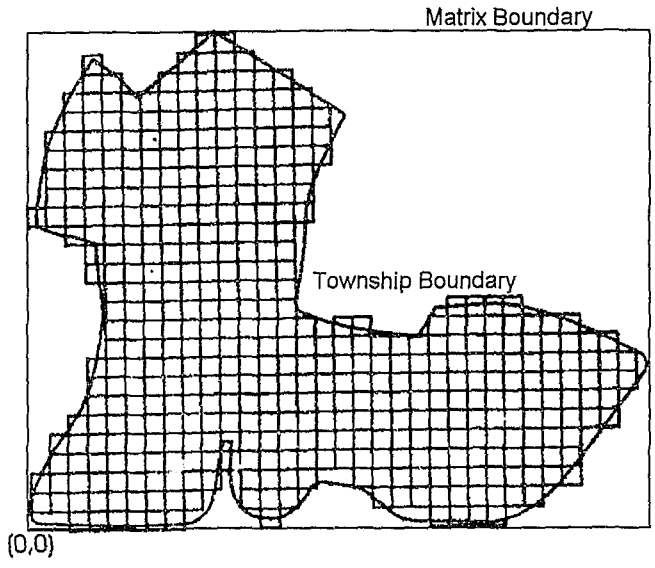


Figure 1 - Township shown with the two types of boundaries [Grimsdale:1960]

Once an area of evenly placed squares is obtained, a routine is required to differentiate between those squares that are within the township and those that are not. An algorithm exists (scanning algorithm [Newman: 1979]) that is used to determine whether the centroid of each unit square is located within the township boundary or not. A '0' is assigned to squares that are found to be outside the township boundary and a '-1' is assigned to the squares inside. Thus the map is sampled and is represented in a matrix and it would look like the sample shown in Figure 2 for the township shown in Figure 1. The element at the bottom left corner corresponds to the bottom left corner of the map. This is a 10x10 sub-matrix which has been extracted from the larger matrix representing the entire map.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \end{bmatrix}$$

Figure 2 - An extract of the Matrix representing the map of the township

The other major reason that the matrix form of map representation was employed was that it simplified the implementation of the actual square extraction process. This involves the placing of a 3x3 square matrices, representing transformer blocks, within the larger matrix over the appropriate elements. When a matrix is placed the elements over which they are placed are rewritten to ensure that subsequent matrices being placed would not be placed over occupied regions. If the placement procedure was completed successfully then the original map elements are restored and another attempt is made with a larger square matrix (to represent an increasing size of the transformer block). This process is repeated until the size of the square matrices cannot be increased any further. Using co-ordinate references, the exact location of the centroids of those transformer blocks are determined.

The second phase of Grimsdale's algorithm was the consumer allocation process and its conceptual framework is straight forward as described in the Literature Survey (B2).

When Grimsdale's algorithm was tested, it was found that it required a fairly uniform load density. The allocation process of the second phase, operates without accounting for transformer overloading or under loading, resulting in a gross variation in the number of consumers allocated per zones. (See the testing and Results document (B6) for more information.) This is a major shortfall in the algorithm proposed by Grimsdale since it was originally designed for use in urban areas where the housing density is somewhat even.

An alternative is to use different sized transformers but the problems with this ideology is as stated in the limitations (section 2). Thus, a means of consumer re-assignment is required to solve this problem and is described in the subsequent sections.

5 Solution Advancement

5.1 Proposed technique

The sweeping algorithm was developed with the intention of re-allocating consumers from overloaded zones to others so that all transformers become evenly loaded. This algorithm can be classified as a heuristic solution. For this aspect of the operation the zone limit must be specified by the user, since none of the zones allocated should exceed this zone limit. The specification of the zone limit has two logical constraints.

1. The zone limit cannot be greater than the maximum capacity that the transformer can supply (maximum zone limit). The maximum number of consumers that can be supplied is determined for the transformer size that is likely to be used predominantly for a given scheme (calculation details are given in the product functional specifications - B1).
2. The zone limit cannot be less than the average number of consumers per zone. This is the rounded up figure obtained when the number of consumers is divided by the number of transformers that are going to be placed. Specification of a zone limit lower than the average will clearly result in the non convergence of a solution.

The two constraints can be expressed mathematically by the following statement:-

Average zone limit $\leq$ *User specified zone limit* $\leq$ *Maximum zone limit*

In this prototype, the "sweeping" takes place in all four directions, but the algorithm is described below for the case when it is sweeping right. In theory, the sweeping can be done in any required direction provided that necessary code is added to program. Starting from the left of the map, an imaginary vertical line is moved towards the right. Transformers are processed in the order that this line encounters. When an overloaded transformer is encountered the following steps are executed:-

1. All transformers located within 1.3 times the distance of the nearest transformer, on the right hand side of the current (active) transformer, are identified as candidates for transfer. Candidate transformers are identified regardless of whether they are overloaded or not. A radius multiplier of 1.3 was intuitively selected and proved adequate in ensuring that there is more than one candidate where suitable. Transformers further away cannot clearly be considered suitable for transfer.
2. Each candidate is then requested to bid for the consumer currently assigned to the active transformer that is nearest to them. The consumer that measures the shortest distance to a candidate transformer is then reassigned.

3. The previous step is repeated until the transformer being processed has reassigned all its consumers who were overloading it.

The next overloaded transformer is identified and the above procedure is repeated. By transferring consumers from one transformer zone to another the consumer allocations will eventually even out. New centres of gravity for all zones are calculated at the end of the "sweep". The flowchart for the sweeping algorithm is given in Figure 3.

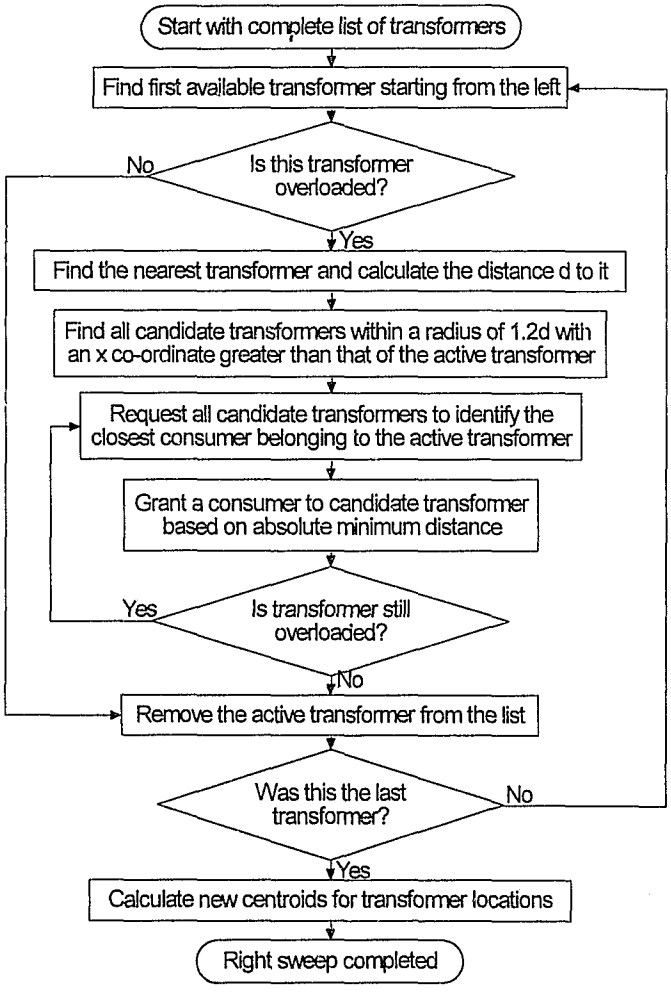


Figure 3. - Algorithm for a single "sweep"

Once this sweeping process is completed, one of two scenarios could exist. The scanning procedure either successfully reallocated the consumers or the consumers have been "piled up" onto one side of the map. The latter case occurs particularly if the last few zones being processed were overloaded. Alternatively, oddly shaped townships could

also give rise to this condition such as one shown in Figure 4. The former case is ideal but to resolve the latter case the scanning process must now be repeated from right to left, top to bottom, bottom to top or some combination of these. In other words, sweeping algorithm must be repeated in multiple directions until the consumers allocated per zone are even.

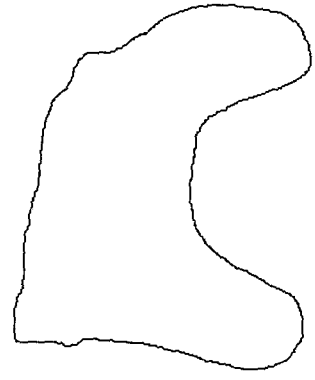


Figure 4 - An outline of an oddly shaped township

Since sweeping in multiple directions is necessary, a means of evaluating the effectiveness of a sweep is prudent. Furthermore, it would be useful if there is a means of determining the best direction for a subsequent sweep. Such evaluation techniques would allow sweeps in multiple directions to occur by leading the program along the most "promising" path and avoiding the combinatory explosion as inherent in the alternatives suggested in section 5.2.3.

One technique for evaluating the effectiveness of a sweep was to determine the number of consumers who were "piled up" at the one end the township. After testing it was found that cases occurred where sweeping in more than one direction resulted in the same number of consumers "piling up" at the ends of those sweeps. Therefore, the figure that was the total sum of all the straight line distances from every transformer to every consumer in its' service area was used as an additional factor for evaluation. It is believed that the this figure is a more accurate measure when comparatively evaluating the effectiveness of a sweep, since clearly this is the factor being minimised. This figure will be referred to the as the *total cable distance* (TCL) and can be mathematically represented as follows:-

$$TCL = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} abs\left|T_i\, c_j\right| \qquad \text{for } M \le \textit{zone limit}$$

Where T_i - (x,y) position of transformer i,

c_j - (x,y) position of customer j,
N - number of transformers,
M - number of consumers for a particular zone,
$|T_i c_j|$ - represents the scalar distance between T_i and c_j.

The TCL calculated immediately after the execution of Grimsdale algorithm is taken as the reference value. After a sweep in a given direction is conducted, it is easy to measure the increase or decrease in TCL as a percentage of the reference value. Thus the computer can numerically quantify the effectiveness of a sweep in a particular direction in this manner.

In order to measure the variation in TCL, the sweep actually has to be performed. This is accomplished through the use of simulation sweeps which involves the actual sweeping process but all the consumer re-assignments are reversed at the end the of the sweep once all relevant calculations have been made. Simulation sweeps are conducted in all four directions and the respective data is gathered and analysed in determining the best direction for the actual sweep. More specifically, at the end of every simulated sweep, the total number of consumers who are "piled up" at the ends and the percentage increase in TCL are computed. The direction that resulted in the least number of overloaded consumers "piling up" is selected for the actual sweep as clearly this would be the logical path to follow. Where there are more than one minimum number of overloaded consumers "piling up", the TCL is considered in determining the direction for the subsequent sweep.

Once the sweep in the best direction is completed then this is regarded as the starting point for subsequent sweeps. In other words, the process is repeated with the simulation sweeps followed by the actual until an acceptable solution is reached, whereby there are no zones with overloaded consumers. Substantially less sweeps are required using this methodology than the alternative considered in section 5.2.3, resulting in a more time efficient process since the best direction for each subsequent sweep is computed. The flowchart for this operation is given in Figure 5.

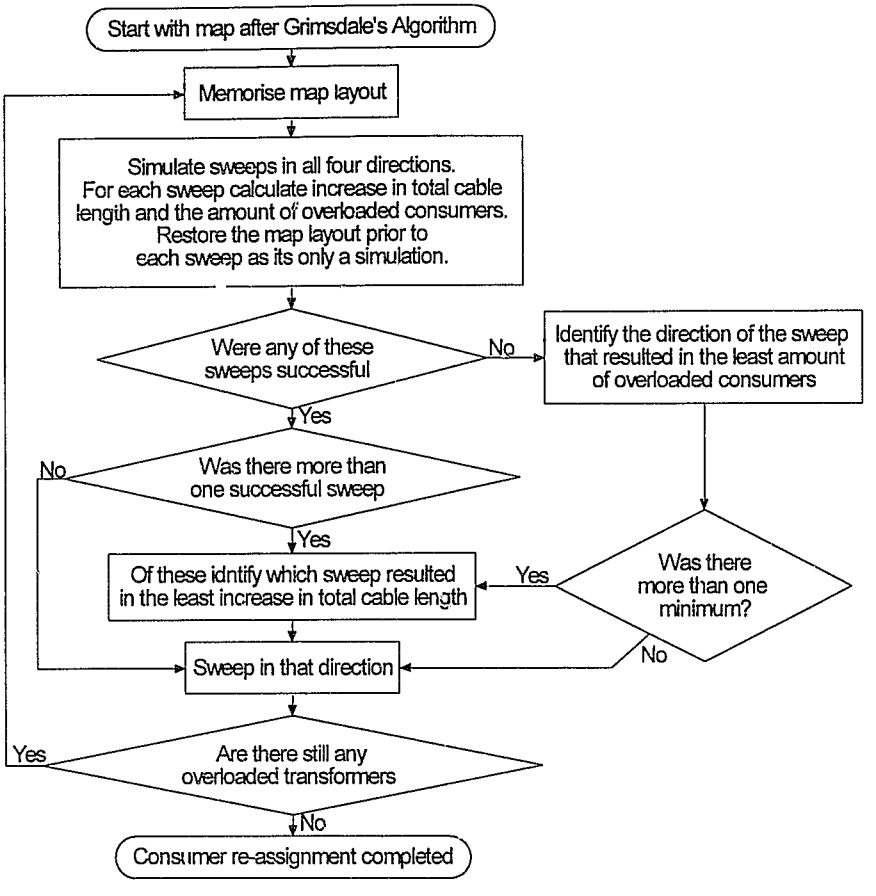


Figure 5 - Overall usage of the sweeping algorithm

It is important to note that the occurrence of two successive sweeps in the same direction is not permitted, as it will not contribute to improving the solution.

5.2 Alternative techniques considered

Section 5.2.1 discusses an alternative to the consumer re-allocation algorithm described above. Section 5.2.2 and 5.2.3 describe alternative consideration within the scope of the proposed problem. This is particularly on the matter of establishing the efficient usage of the sweeping algorithm.

5.2.1 Alternative technique for reallocating consumers

An alternative was to develop an expert system (concepts of expert systems is described in the Literature Survey - B2) for consumer re-

assignment. Such a system would envisaged to start from one end of a drawing and systematically transfer overloaded consumers to the nearest transformer or transformers using a list of rules as a guide. Similarly, in the cases where transformers are found to be under loaded, it will take on the nearest consumers belonging to neighbouring transformers. In this manner, consumer transference would proceed, while strictly adhering to a specific set of rules.

This solution methodology requires a suitable rulebase that would have to be compiled to ensure that consumers are efficiently redistributed. Certain shaped township layout were identified where incorrect transference could occur due to of the list of available rules. Transference occurring especially at the ends of a township were prone to erroneous or repetitive transference resulting in the non-convergence of a solution. Thus further rules had to be evolved and this resulted in the rulebase getting larger and more complex. More rules were then required as new possible situations were formulated. Furthermore, it was always possible to hypothetically conjure up township layouts where the existing rules would fail to converge at a solution. The only valid conclusion that could be made was that every township was unique requiring some different innovative solution to deal with each case. This technique was thus abandoned as it could not always guarantee a solution and the rulebase was getting complicated.

It should be noted that this expert system technique evaluated could, in most cases, produce a solution but it would neither be optimum or even near optimum. However, the algorithmic technique (sweeping algorithm) was found to be more robust and it would be shown that it would converge at a solution more reliably. This is due to the nature of the algorithm as it systematically processes all the transformers starting from one end of the township to the other. The solutions is also far more likely to be near optimal that the expert system discussed. As stated in the Literature Survey (B2), more complex expert systems can be evolved but vast amounts of information, that is not readily available, would be required.

5.2.2 Alternative One: technique for employing the sweeping algorithm

In this particular technique, sweeping would be done in all four directions. After every sweep, an accumulative list is compiled of the consumers that were reassigned and those that were not. It should be noted that the original consumer assignment would be restored prior to sweeps in subsequent directions. At the end of the sweeping process the accumulated lists are analysed. Those consumers that were not reassigned are clearly the ones that are located within the vicinity of the transformers and therefore their assignments are correct. Then, every transformer in turn will take on the nearest available consumer until all reach their specified capacity. When some transformers reach their limit,

the remaining will continue to take on consumers until all remaining have been assigned. The new centres of gravity of load are calculated for each transformer.

However when this algorithm was tested it found to have a fundamental flaw. The nature of this algorithm is such that consumers located especially in the outskirts of townships were not being assigned at all resulting in incomplete solutions. The reason for this is that as the transformers are taking on consumers, especially in densely populated areas, the transformer limits are reached far before consumers that are located on the outskirts have been assigned. In such case those consumer cannot be assigned to other transformers as they are too far away.

The other problem occurred in sparsely populated zones. All the consumers that are nearest to other transformers would have been absorbed thus resulting in no more consumers being available for absorption for transformer located in a sparsely populated region. Therefore the only consumers that can be absorbed are further than the nearest transformers and such an allocation will be deemed unacceptable.

This technique was abandoned due to the problems listed above.

5.2.3 Alternative Two: technique for employing the sweeping algorithm

This methodology was based on the premise that sweeping in all four directions must provide a solution. If after a particular session of sweeping (in four non-repetitive consecutive directions) a solution is not found then an alternative combination would be used. For example if in a particular session, sweeping occurred in the right, left, up and down and a acceptable solution was not achieved then another sequence would be used. Since there are four variables (directions) there can be sixteen combinations (2^4 combinations).

Based on the testing employing this technique it was found that in most cases there was redundant sweeping and in other cases insufficient sweeping. For both cases, the time taken to complete the sweeping process was excessive due to the 64 sweeps (16 sessions x 4 directions) for each run of the program. Thus the method proposed and described in section 5.1 above was devised and it was evolved from this technique to be more efficient.

5.3 Further refinement process

An unanticipated result of the repeated use of the sweeping algorithm described above was that the some of zones became stretched and warped. From visual inspection it was easy to ascertain that some

consumers were being supplied from transformers that were not the nearest. These conditions were occurring particularly along the borders of adjacent zones (See the Testing and Results (B6) for more details.) It was established that by swapping the allocations of two consumers in adjacent zones, the straight line distances from the transformer to the respective consumers can be reduced significantly. Clearly, a single operation will not make a difference to the TCL. However, it will be shown that a substantial amount of swapping would noticeably reduce the TCL.

5.3.1 Refinement algorithm

An algorithm was evolved to improve the consumer assignment and its basis of operation is described briefly. Suppose that there are two consumers x and y who are allocated to transformers A and B, respectively. If the following mathematical condition is true

$$(|Ay| + |Bx|) < (|Ax| + |By|)$$

then the allocations are swapped so that consumer x is supplied by transformer B while consumer y is supplied by transformer A. This checking process is not performed on all the consumers but only on those ones that have been identified as being assigned to transformers that are not the nearest.

It should be noted that the number of consumers allocated per zone would still remain the same, since this is merely a consumer assignment swapping process. Once again the percentage increase or rather decrease in TCL with respect to the reference will be used to measure the degree of improvement. The flowchart for this operation is shown in Figure 6.

Two different techniques for the usage of the refinement algorithm was devised and their effectiveness was evaluated (discussed in the Testing and Results document - B6). The two different techniques are described in the following subsections.

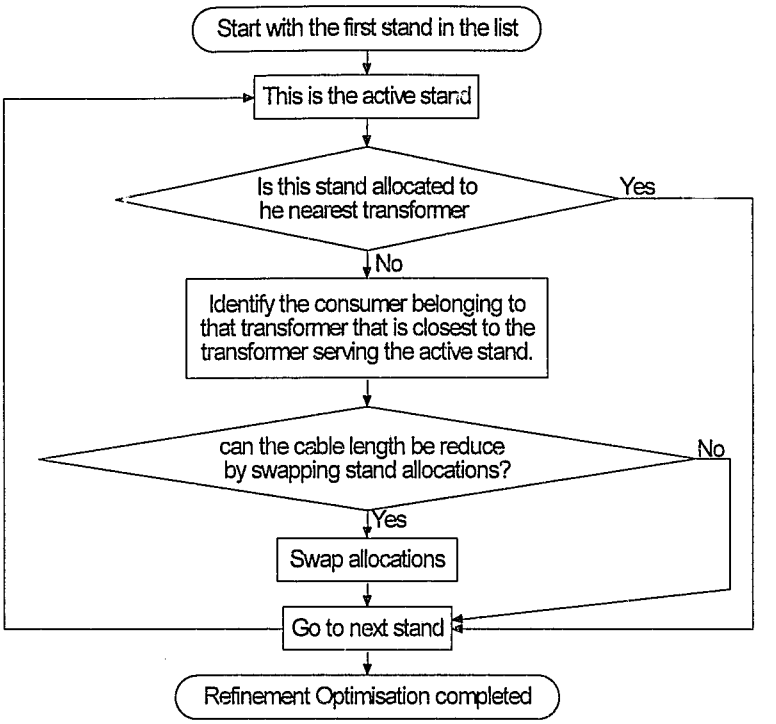


Figure 6 - Flowchart for the Refinement Algorithm

5.3.2 Post refinement process

In this methodology the refinement process is executed once scanning algorithm had completed its task. A single execution of the refinement algorithm is, in most cases, is insufficient. As consumers are exchanged between zones and the shape of the zones change those consumers may find themselves closer to other transformers. Therefore, this algorithm must be executed until no more consumer exchanges can be achieved.

5.3.3 Current refinement process

In this methodology, the refinement process is applied after every sweep. The refinement process is however not used after every simulation sweep in order to save time and is assumed that the simulation sweep would provide an adequate indication of the effectiveness of the sweep. Since this is a single application of the refinement algorithm, it is faster overall than the using the previous methodology. It would also be shown that this techniques is a great aid to preventing zones from severely distorting. For these reasons this technique is preferred.

6 Drawing the boundary polygon

As stated in the specifications it is necessary to define polygonal boundaries of the transformer zones that have been calculated. An algorithm was developed to accomplish this and it is described briefly. Assume that there are *n* nodes (representing consumers) located within a Cartesian plane as shown in Figure 7. Four quadrants are defined by the four extreme points labelled *N*, *W*, *E* and *S* as shown in the figure.

The basic principles of the algorithm are described for defining the boundary for the first quadrant starting from point *E* and ending at point *N*. An imaginary rectangular is defined where the points *N* and *E* are diagonally opposite each other. Within this quadrant, a point is identified which has the shortest horizontal distance to point E. The point that was just identified is noted as a vertex of the boundary polygon. This point is now assumed to be the new *E*, thus defining a smaller rectangle or quadrant than the previous. The same process is repeated to find the point that has the shortest horizontal distance to point *E*, and identified as a subsequent vertex of the boundary polygon. This process is repeated until point *E* coincides with point *N*, which indicates that the boundary has been defined for this quadrant.

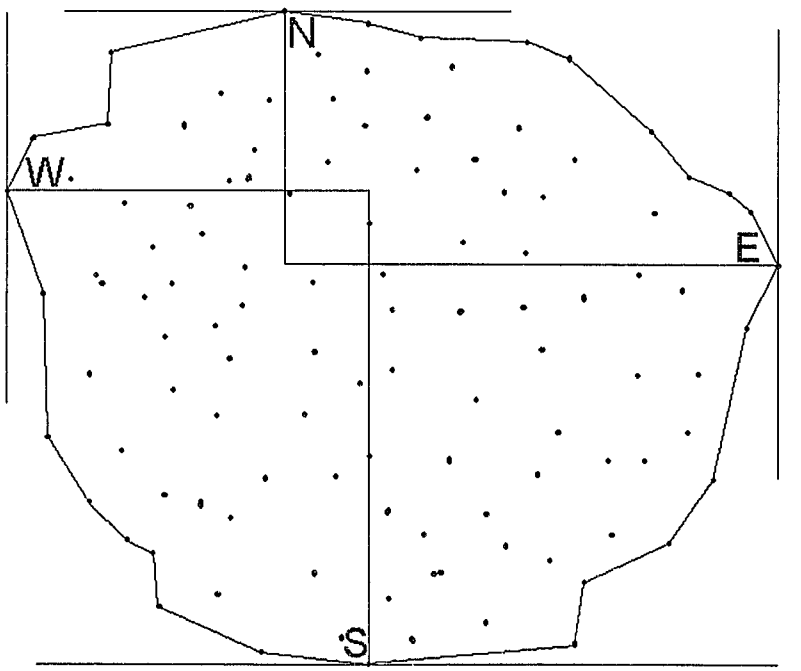


Figure 7 - Defining a crude polygon around a set of points

A similar process is used to define the boundaries for the rest of the quadrants and is clearly stated in the flowchart presented in Figure 9. It must be noted that the when defining the boundary from point *N* to *W* minimum vertical distances are considered and not horizontal distances. Similarly, when defining the boundary from point *W* to *S* horizontal distances and from point *S* to *E*, horizontal distances must be considered. Thus a boundary polygon for a group of points can be defined. The algorithm contain provisions to deal with points that are vertically or horizontally lined depending on which quadrant is being defined.

This algorithm developed and was chosen over other available algorithms such as Graham's scan [Preparata:1985, O'Rourke:1995:106] and Jarvis's march (O'Rourke:1995:110). Both algorithms are faster than the one proposed but define convex polygons around a set of points located within the Cartesian plane. The reason for using the algorithm proposed rather than available techniques is that in the zoning process, it is usual for the zones to interleave and fit like a jigsaw puzzle. Figure 8 shows a group of interleaving transformer zones and it can be noticed that they are concave polygons. The definition of convex polygons for this particular example will clearly result in overlapping convex polygons that can be easily misinterpreted during visualisation.

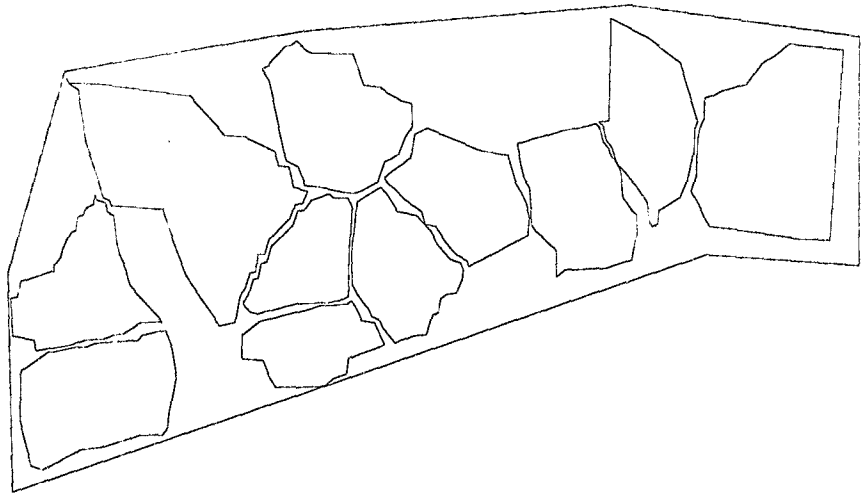


Figure 8 - Interleaving transformer zones within a township

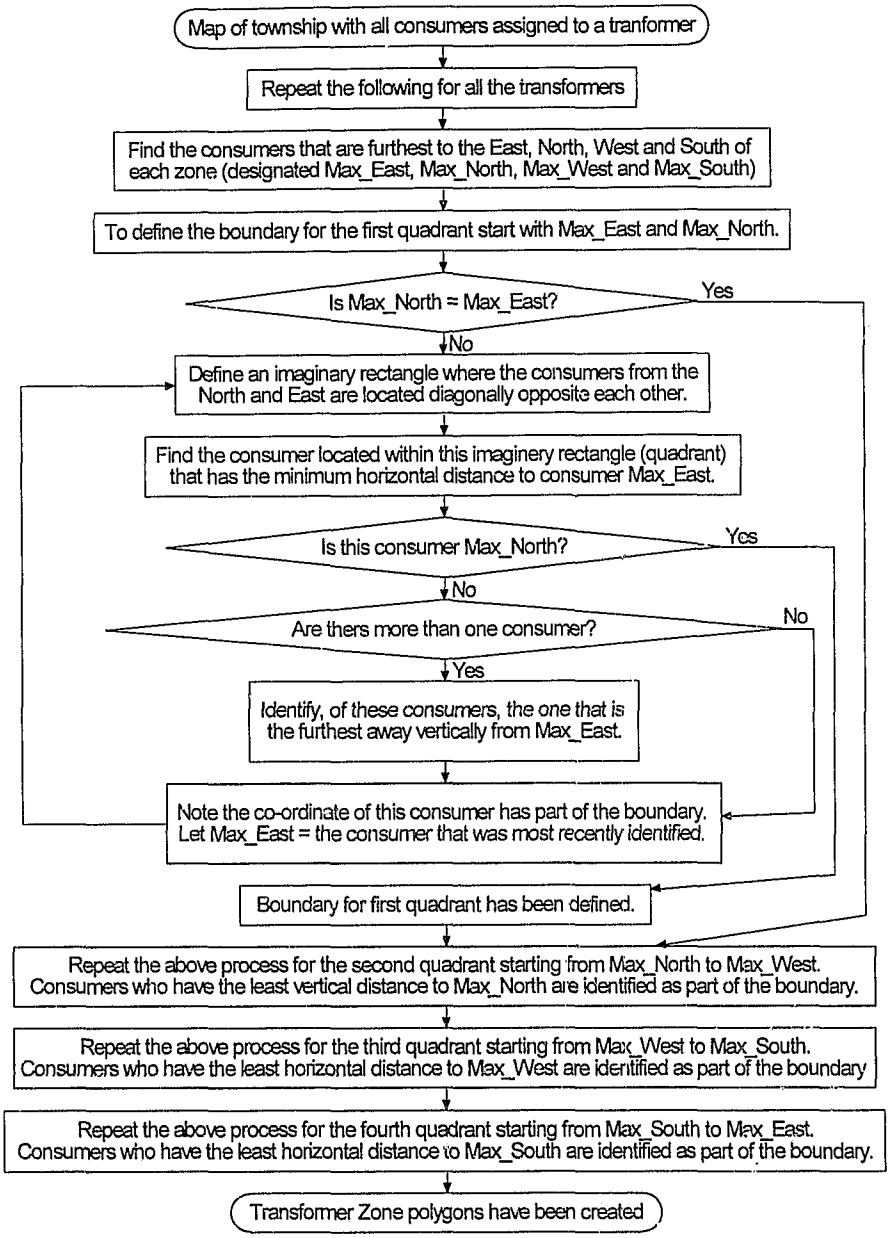


Figure 9 - Flowchart for defining polygons

7 Interface: Output

The output, like the Input, is also classified into two types since the are produced using different techniques.

7.1 Geographical Data Output

The nature of the process is such that visualisation of the results would minimise the evaluation time and allow more effective assessments to be made. It is possible to create a windows based graphical environment that could serve as a visualisation tool. However, this aspect is once again not crucial to proving the concepts at hand and will require a substantial time to develop. Furthermore, such a graphical interface will not have the powerful view control features that are typically available on most CAD packages. It would be more appropriate to use an existing graphical package (third party software). In order to do that it is necessary to create a file format that can be read by the selected graphical package.

The MicroStation™ package by Intergraph™ was used as the third party graphics package. It boasts powerful CAD features and tools which were used extensively in the testing phase. For example, tools that allowed measurements of two points on the map proved useful in gathering data for section 8.3.

Since a CAD file in DXF format is being used as an input, it would be logical to create a CAD file in DXF format as an output. Naturally, the output CAD file will contain appended information that would enable the graphics program to display all the transformer delineation. Since the number of consumers allocated per zone is of importance, it will be displayed within all the transformer zones. For testing purposes, the transformer location is also provided which will serve as some form of indication to the validity of this program's operation.

In addition, for testing purposes, the transformer number is displayed next to the transformer location so that the TCL per zone can be found by looking up a text file that is simultaneously produced in the computation process.

7.2 Output of Results

In addition to the graphical data output, other data such as the total cable lengths (TCL) and its percentage variation must be visible during the scanning process for testing purposes. This information will be displayed using the DOS based interface. The initial TCL is calculated after the completion of Grimsdale's algorithm and presented in metres. Simulation sweeps are done in every direction and the changes in TCL is presented

to the user as a percentage along with the number of consumers piled up at the end of those simulation sweeps. Data gathered in this manner will purely be for research and evaluation purposes and the version that will be commercially available will not incorporate this feature.

A typical display would be as shown by the text window in Figure 10. At the start, the user is informed of the original TCL in metres before performing any of the simulation sweeps to generate the required information. This line is followed by a line that specifies in a tabular form the direction to which the information below pertains. The first line following this indicates the number of consumer piled up after a sweep in that direction followed by the percentage variation in TCL on the next line. A different font is used to indicate the user input which in this case is the specification of the direction for the subsequent sweep. The reasons for the particular direction selected should be self explanatory. A exiting feature (q) was deliberately incorporated so that during testing the program can be terminated after any number of sweeps so that a post-mortem on the zone formation can be conducted if necessary. The reason for the existence of this feature is that if the program is forcefully terminated using Windows commands the program cannot then proceed to create the transformer zones. The final or overall increase in percentage variation after the scanning algorithm is specified on the last line.

```
Initial total cable length = 459223
Right       Left        Up          Down
28          104         79          40
3.7967      0.395934    3.04687     5.55139
please enter direction of sweep: r

28          18           5          28
3.7967      5.35052     3.89134     3.7967
please enter direction of sweep: u

0           5           5           5
3.28117     3.80953     3.89134     4.23919
please enter direction of sweep: r

0           0           0           0
3.28117     3.28117     3.28117     3.28117
please enter direction of sweep: q

Percentage increase in total cable length = 3.28117%
```

Figure 10 - Sample of The Scanning Algorithm User Interaction

If the post refinement process is utilised then the first part of the output dialogue will be similar to the dialogue shown in Figure 10. The second

part, similar to the dialogue shown in Figure 11, will be appended to the first part.

```
Percentage increase in total cable length = 4.35873%
counter = 234  193
Percentage change after refinement = -0.24834%
counter = 148  68
Percentage change after refinement = -0.795819%
counter = 131  36
Percentage change after refinement = -0.914348%
counter = 127  19
Percentage change after refinement = -0.975512%
counter = 122  9
Percentage change after refinement = -0.996913%
counter = 118  4
Percentage change after refinement = -1.03454%
counter = 117  4
Percentage change after refinement = -1.04278%
counter = 115  4
Percentage change after refinement = -1.05164%
counter = 115  2
Percentage change after refinement = -1.05669%
counter = 113  0
Percentage change after refinement = -1.05669%
```

Figure 11 - Sample dialogue window for the refinement process

It can be observed that after every line where the percentage increase in TCL has been listed (except for the last line), there is a line which starts with `counter =` and lists two integers. This aspect was used during the test phase and the first number lists the number of consumer that have been assigned to transformers that are not the nearest. The second number indicates the number of consumers that were successfully exchanged. Looking at those lines, it can be seen that this process is repeated until no more consumers can be exchanged whereby it is halted.

If the current refinement process is employed then the sample dialogues shown in Figure 10 and Figure 11 will be replaced by something that is similar to the one shown in Figure 12. It can be noticed that it is a marriage between the former two dialogue boxes, and the contents should be self evident.

```
Initial total cable length = 455651
Right   Left    Up      Down
135     180     214     195
6.22568 24.6873 8.60661 26.4696
please enter direction of sweep: r

counter = 139  84
Percentage change after refinement = 4.98809%
135     20      135     114
6.22568 2.86425 37.7369 15.1844
please enter direction of sweep: l

counter = 190  144
Percentage change after refinement = 0.885504%
0           20       20       0
-0.0630073  2.86425  2.59278  -0.0630073
please enter direction of sweep: r

counter = 126  68
Percentage change after refinement = -0.531581%
0            0             0             0
-0.0630073   -0.736926     -0.736926     -0.736926
please enter direction of sweep: q

Percentage increase in total cable length = -0.531581%
counter = 105  29
Percentage change after refinement = -0.637645%
```

Figure 12 - Sample dialogue for the current refinement process

8 Costing

The sections described up to this point have been on the technical aspects. However, a methodology is required to determine the most suitable transformer delineation. The evaluations methodologies are described followed by the discussion of the costing formula employed.

It should be noted that for this prototype, it was necessary to enter the various parameters and gather results manually. The gathered data is then entered into a spreadsheet. This aspect was not automated as part of this research and will be when a more time efficient algorithm is devised to search the problem space.

8.1 Costing versus zone limit variation.

For a given number of transformers that are to be utilised in a particular scheme, the zone limit can be varied from the maximum zone limit to the average zone limit. For every zone limit, an estimate can be made on the cost of the scheme. Cost calculation details are provided in section 8.3.

It is expected that as the zone limit is reduced from the maximum to the average, the cost for that scheme would increase due to the increase in cable lengths. The cable lengths are expected to increase since the zones would have to start including consumers belonging to other zones that were further away since they were not initially include. However, The increase in costs, if marginal, may warrant a particular configuration with a corresponding zone limit being selected at the designer's discretion.

8.2 Costing versus use of more transformers

As explained in the specifications, increasing the number of transformers does not necessarily increase the total cost. Shorter cable runs may then required thus possibly reducing the overall cost. This aspect can now be explored since it implies potential savings on the overall cost. (Cost calculation details are provided in section 8.3.) Suppose the process of varying the zone limit described above is repeated for varying quantities of transformers. It would then be possible to plot a curve of the estimated costs versus the number of transformers. There would be a minimum in the curve and that would represent the most suitable number of transformers for a particular scheme.

It is recommended, that the average zone limit be used in determining the number of transformers, since that represents the maximum cost for a particular configuration.

8.3 Cost Estimation Formulas

As with any work of this nature, a means of estimating the cost is prudent in assessing a particular zone layout. The cost estimate includes factors such as the number of transformers, total cable lengths and the zone limit. In this cost estimation, no provision has been made to account for the running costs that would be incurred due to the no-load losses of under loaded transformers, especially during the off-peak period. The reason for this is that in order for such an evaluation to be accurate, sufficiently accurate load curves and transformer characteristics would be required. This factor does not alter the costing equation presented but will be a factor that can be added to this equation when an acceptable relationship is established.

The costing aspect was not implemented as part of this program but rather in a separate spreadsheet. The reason for this is to keep the acquisition of raw data and the investigation of evolving the most appropriate costing formulas separate.

In order to obtain reasonable cost estimates it is important to model the relationship between the TCL and the zone limit, realistically. The TCL is not linearly proportional the actual cable lengths as initially suspected. In fact the ratio of the TCL to the actual feasible distance was found to increase in somewhat predictable manner with zones. Since the costing investigation described above involves the variation of the consumer zone limit, an acceptable formula was required to model the relationship between TCL and actual feasible distances.

The relationship between the TCL and actual cable lengths was determined empirically from a variety of test cases with varying zone limits. For every selected case, the TCL would be calculated by the program and the routing would be manually done, thereby establishing a ratio. The LV feeder routes were placed mid-block. The total service conductor lengths were estimated as the product of the approximated average (40m) and the number of consumers within that zone. The calculated TCL was found from the text file that is produced with every test run of the program. For the manual routing in all these test cases, the transformer was moved from the location calculated to the nearest feasible location. This change in distances for all transformers were minimum if not negligible. The actual cable distances were calculated by placing cable routes in the most logical way for a variety of zone limits. It is accepted that if the routing was done by other designers, the actual feasible distance may vary but it is anticipated that all variations would be a within a negligible range. The important aspect however, is to represent the general variation at a satisfactory level for cost assessment purposes.

Test results gathered in this manner allowed the modelling of the relationship of the TCL with respect to the actual cable length sufficiently accurately using a third order polynomial as shown in the graph in Figure 13. Matlab™ was used to generate a spline from which the coefficients of the second order polynomial was generated. Greater order polynomials as high as 10 were generated but the second order was found to model the behaviour of the measured data sufficiently. This modelling has a limited range of 20 to 190 consumers and its behaviour beyond these limits has not been examined for the simple reason that smaller transformers are being preferred in recent trends. However, this range represents the commonly used figures currently used by the electrification program in South Africa.

The graph presented below essentially implies that whenever a TCL is calculated, it must be divided by the factor for that particular zone limit.

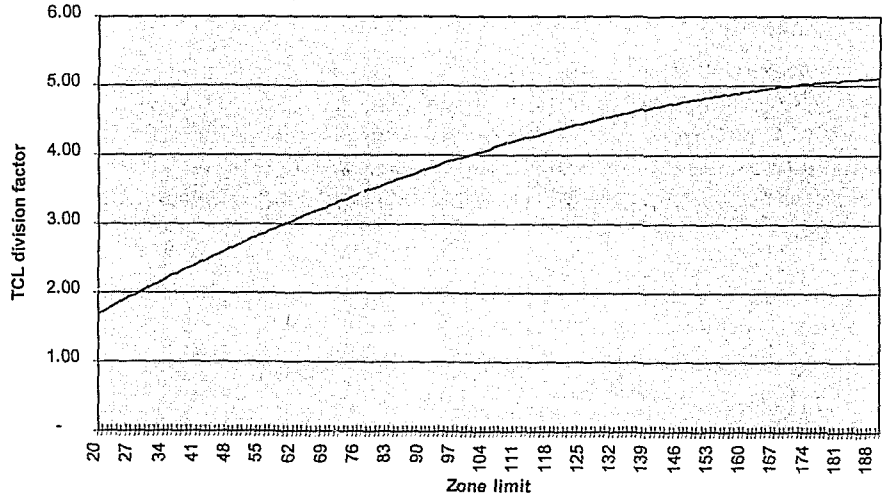


Figure 13 - Graph of TCL Division Factor vs Zone Limit

The equation modelling this graph is given below.

$$TotalCost = \left(\frac{TCL \times LVCCF}{1.12e-6(ZL)^3 - 2.9e-4(ZL)^2 + 3.84e-2(ZL) + 1.08} \right) + XFRS \times (TCF + MVCCF)$$

Where	*LVCCF*	Low voltage cable cost factor
	TCF	Transformer Cost Factor
	MVCCF	Medium voltage cable cost factor
	XFRS	Number of transformers

ZL Zone Limit

It should be noted that this cost formula can only be correctly applied only if the consumers contained in all the transformer zones are almost the same. Otherwise the results will not be valid since this equation assumes that *ZL* factor is representative of all of transformer zones.

The TCF constitutes the actual cost of the transformer, its associated fittings and labour. An additional fixed fee is added to the cost of the transformer to represent the cost of the MV cabling. It should be noted that there are two types of MV cabling.

1. Internal MV cabling - refers to the MV feeder network that supplies all the distribution transformers within the township.
2. External MV cabling - refers to the feeder that tees off from an existing MV line and extends to the settlement.

The external MV feeders are of no concern since those lengths are independent of the township layout. On the other hand, the internal MV feeders are of concern and are proportional to the number of transformers being placed. There are various factors that influence the cost of the MV feeders such as single phase, dual phase and three phase and is therefore difficult to establish accurate values. Nevertheless, an extra cost factor must be accounted for to improve the estimates so that it better reflects the cost of the scheme. This is represented by the medium voltage cable cost factor (MVCCF) which is regarded as a fixed addition to the cost of the transformer (TCF). This factor would typically be determined by the user depending on the type of systems which is planned on being installed.

Once the cost of the transformers and the MV cabling have been accounted for, the remaining cost of the scheme must clearly be for the low voltage cabling, associated street furniture and energy dispensers[1]. The low voltage cable cost factor (LVCCF) will be empirically determined so that the total cost would be similar to an actual designed electrification layout once all other costs have been accounted for, as mentioned above. The LVCCF is an average figure that accounts for the pole costs and for the different cables that are typically used for feeders and for service connections.

The total cost of the scheme was estimated as the product of the number of consumers and the "magic cost" for supplying a single consumer. The magic cost is established by the national electricity supplier as the

[1] Energy dispensers are located within the consumers' premises, supplied by the service connection, and has plug points. It requires a token or a PIN number to activate the supply so that power can be drawn from the energy dispenser.

maximum allowed per connection for any design that is being implemented, which has a value of R3,500[2] [Dwolatzky:1998]. The goal trying to be achieved as far as cost the cost per connection in the immediate future is R2,500 [Stephen:1998]

More accurate cost factors will be empirically determined after a few months of field usage. Accurately determined cost factors for the cost equation will only serve to guide the user in selecting the most suitable transformer zone configuration.

8.4 Determining cost profiles

Having described the cost equation, it is important to describe the most efficient way of determining the cost profile for a township. Once the cost profile has been determined for varying number of transformers, it would be possible to select the scheme with the optimum number of transformers. Unfortunately, at present the costing profiles have to be determined manually. The, TCL, average zone limit, and the percentage increase in TCL must be manually recorded into a spreadsheet where the costs can be determined using the equation defined above.

For testing purposes the values used for the TCF, LVCCF and MVCCF were 18000, 29 and 1000 respectively. The TCF was for a 3ø 50kVA transformer, pole fittings and labour. The latter two figures are approximations obtained from actual design work that was done on a real township (discussed in the Testing and Results - B6).

The process of determining the transformer zone values is as follows:

1. Establish the desired maximum zone limit for a particular type of transformer that is being planned on being used throughout the scheme.

2. Using this figure, establish the minimum number of transformers that the township would require.

3. A quick survey is done to evaluate the number of transformers that are most suitable for the design. This is accomplished by executing the program with the specification of the average zone limit and varying the number of transformer zones. To assist the user, the program will state the average number of consumers per zone. The number of transformers specified does not necessarily have to change in increments of 1 but it can be 2, 3 or more to rapidly converge at the

[2] All monetary values are in South African Rands. This includes the values assigned to the various factors comprising the cost equation.

required configuration. The designer must use his/her judgement for this purpose. Thus the cost for each execution with varying number of transformers can be established. Using this information it is possible to identify the layout which would produce the cheapest cost with a specific number of transformers.

4. The designer can then view this configuration that he is very likely to use. It must be noted that some of the configurations may be impractical due the specification of the restrictive zone limit. He may then, based on the layout, specify a zone limit that is greater than the average with the same number of transformers. That scheme can then be regarded as the first draft at the transformer delineation process.

This is substantially a more efficient search algorithm than investigating every single possibility to find an adequate solution.

9 Conclusion

The conceptual design for the transformer zoning program has been presented in this document, detailing its principles of operation. Alternatives, where considered have been discussed in the sections to which they pertain.

This document describes the implementation aspects of Grimsdale's algorithm. The shortfall in Grimsdale's algorithm was identified which was that the transformer zones generated had a gross variation in the consumer allocation. The sweeping algorithm was proposed with the intention of evening out the consumer allocations. However, the sweeping algorithm was found to have a flaw which was inherent in its design resulting in unacceptable warping or distortion of the transformer zones. This problem was overcome by a refinement algorithm which swaps consumer assignments in order to not only reduce the TCL but also to improve the zone layout. Thus these two sequential techniques have been devised in an attempt to achieve acceptable zone layouts.

The algorithm proposed can be considered as heuristic. Consumer assignment and transference is more efficiently accomplished by operating in a systematic manner from one side of the township to the other. Due to the systematic operations technique used, it was anticipated that this technique would converge at a solution. This algorithm does not try and minimise any linearised objective functions.

Artificial intelligence systems were considered, but it a system that could reliable converge at a solution could not be devised.

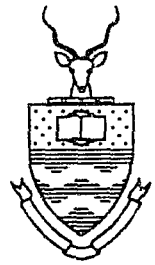
Considering the statement by Willis et al [1985] which stated that when the quality of the data was high, advanced methods were found to produce substantially better results over simpler methods. However, as the data quality was reduced more advanced methods gave no better results than the much simpler methods. In South Africa and in most of third world countries it is safe to assume that the quality of the data will be poor. At a time when more accurate levels of information are readily available, a suitably more sophisticated technique can be devised.

In developing the interface for this module, the fact that it may be required to interact with various other software was a key consideration. Furthermore, care was taken in the design of the overall algorithms to ensure that all software bugs were eliminated prior to coding stage.

Details of the high level design and the low level design are in the documents B3 and B4, respectively. It is important to have understood this document sufficiently in order to follow those supporting documentation. It

will be shown that as anticipated this software module is capable of producing a first draft at transformer zoning.

will be shown that as anticipated this software module is capable of producing a first draft at transformer zoning.



SADDIN

High Level Software Design

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History iv

Configuration Control iv

Document History iv

Revision History iv

Management Authorisation iv

Change Forecast iv

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Audience 1

1.4 Applicable Documents 2

1.5 Assumptions 2

2 Establishing a Design Methodologies 2

3 Object-Oriented Design and Development 6

3.1 Essential Basic Concepts 6

3.2 The Object-Oriented Design/Development Methodology 11

3.3 Booch Notation 15

4 System Architecture 20

4.1 Logical Architecture 20

4.2 Physical Architecture 33

5 Conclusion 39

Change History

Configuration Control

Project:	SADDIN
Title:	High Level Software Design
Doc. Reference:	D:\1996_34\TP\TB330
Created by:	T Rajakanthan
Creation Date:	20 April 1997

Document History

Version	Date	Status	Who	Saved as:
0.01	97\04\20	Draft	T R	\1996_34\TP\TB330.001.doc
1.00	99\02\04	Approved	T R	\1996_34\TP\TB330.100.doc

Revision History

Version	Date	Changes
0.01	97\04\20	New document created from TB002 (Document Creation Template)
0.02	97\10\07	Revisions were made in accordance with changes in Coding.
0.03	98\0305	Revisions were made
0.04	98\05\26	Changes were made as recommended by supervisor.
0.05	98\06\02	Conceptual Design was separated as single document TB 328
0.06	98\09\02	Further revision and refinement was made to the document
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	99/02/04	Approved	TB 533

Change Forecast

Will be updated if there changes made to the design.

1 Scope

1.1 Introduction

This document deals with the aspect of optimum transformer location and its implementation conforming to the specifications outlined in the Product Functional Specification document (B1). It serves to document design decisions and decisions concerning the methodology of solving this problem. In order to appreciate this document fully it is necessary to first read the Product Functional Specification so that the reader can understand what the nature of the requirements. The literature survey (document B2) will provide the reader with critical information on existing systems and solution methodologies. The overall conceptual design is described in document B3 in a concise manner and explains some of the major decisions made regarding the overall operation of this software product.

In this document, the adoption of a suitable design methodology and the basic concepts of object-oriented design are discussed. The system developed is then located within the broader context of the object-oriented development process. The system's architecture is described in detail, including both the logical and physical architectures. The logical architecture is illustrated mainly with the aid of class diagrams while the physical architecture is conveyed through the use of module diagrams.

1.2 Purpose

The purpose of this document is to give the reader a *high level* understanding of the implementation of the Transformer Zoning Program. The low level design and implementation is based on this document and therefore the reader has to grasp the concepts presented here.

1.3 Audience

The project manager and developer, members of the SEAL management board, the external examiner, and all other interested parties.

1.4 Applicable Documents

1.4.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.4.2 Definitions

Consumer Supply Point - This is the geographical location where the physical power cables are extended to and typically connected to a low cost distribution box.

1.4.3 References

1.5 Assumptions

2 Establishing a Design Methodologies

Before the analysis of the problem domain it is crucial to adopt a design methodology, since the methodology affects both the analysis of the problem as well as the design. Although other design methodologies exist (such as Top-down structured design and Data-driven design), they are not discussed as they are beyond the scope of this document. In dealing with this entire system, an object-oriented approach has been adopted because of its inherent advantages over other approaches. It will become clear in the subsequent sections that it is the most appropriate methodology to be used in the design of this system.

Object-oriented software development differs fundamentally from traditional functional or procedural approaches. A structured methodology focuses primarily on solving problems using algorithmic or functional decomposition. This is accomplished through the extensive use of procedures or subprograms, often referred to as procedural programming. Such a design would constitute a tree structure comprising a main procedure which utilises a number of subprograms, each of which further utilise other subprograms and so on.

On the contrary, object-oriented development allows the *decomposition* of a software system into key *abstractions* in the problem domain. Use of the term abstraction refers to the process of identifying a few well defined components which together represent a complex model. Thus, using data abstraction and decomposition, a software system can be represented by entities which model objects in the real world. According to Booch, this form of modelling complexity is intuitive and appeals to the human cognition.

The object-oriented paradigm has inherent advantages as it encompasses the concepts of abstraction, encapsulation, modularity and hierarchy within a single conceptual framework, called the Object Model. Booch [1994:20,77] has identified the following significant benefits of the use of object-oriented design and the Object Model:

- **Object oriented decomposition.** Object-orientation helps in managing the complexity which is inherent in software systems. Problems that need to be solved are often complex thus requiring a system to be broken up or decomposed into sub-systems which are in turn decomposed into smaller, more manageable parts. This division of a system's state space allows us to understand the system by examining a few parts of it at a time thereby reducing the complexity of each individual component.

- **Stability and Risk Reduction** Object-oriented systems are designed to evolve incrementally from smaller stable systems. A complex system built from scratch cannot be expected to work and cannot be easily patched up to do so. However a complex system can be evolved from a simple system that works, which reduces the risk which is normally associated with building and maintaining complex software systems. It is essential to employ a programming methodology that allows a system to evolve in order to match further requirements whilst the integrity and stability of the system are maintained. This is an essential criteria particularly for systems being developed in a research environment.

- **Software Reusability** An object-oriented approach provides an economy of expression through the reuse of common mechanisms. This results in smaller systems than can ever be designed using algorithmic decomposition. This implies that software systems are more maintainable as there is simply less code. In particular, aspects of the transformer zoning process that are repetitive or have similar properties that can be encapsulated in a class. For example a `stand` is a class as it can contain a co-ordinate and a text label representing the consumer supply point of a consumer. In a typical process there will be many instances of "stand"s each referring to a unique consumer within a township. Since classes have uniquely defined properties, it can be easily used in other projects.

- **Maintainability** The facilitation of maintainability is one of the key advantages of using object-oriented techniques. Due to their inherent ability to evolve incrementally over time from stable intermediate forms, and due to the high degree of reuse which is achievable, object-oriented systems are highly maintainable. This implies that properties of classes can be modified without making little or no changes to other related classes. The developer can enhanced some process classes within the software system to become exceedingly more complex, knowing that ramifications to the rest of the system are minimised.

- **Natural Conceptualisation** Booch suggests that object-orientation systems are a natural way of viewing software which appeals to the workings of human cognition as it models the real world. Abstractions that model the problem and solution domains in terms of entities that are expressive and intuitive lend themselves to effortless conception and comprehension. A purely functional decomposition of the system would hamper one's ability to effectively describe the domains being dealt with.

For various reasons that are listed below, the coupling between the system interface and the body of the system was minimised, with respect to the transformer zoning program. One of the functional specifications

was that provision must be made for the transformer zoning software to be platform independent so that it can be incorporated into other software systems. In addition, when developing a software unit, it is crucial that it can be tested independently without incorporating it into any larger program. This allows an evaluation to be made regarding the performance of this software unit. In order for different software systems to operate, clear interfaces have to be defined, designed and implemented. Object oriented design approach allows the interface classes to be defined separately to the process classes, thereby isolating the transformer zoning process from the system input and output. Thus dependencies on external factors (which may often be beyond the control of the developer) are encapsulated and contained within the system's interface classes. This further ensures that the interface can be tailored to integrate with any other software package by simply modifying the system's interface classes.

To conclude the design methodology selection process, the complexity of the system being designed advocates the choice of an object-oriented approach because an object-oriented view is simply "better at helping us organise the inherent complexity of software systems" [Booch, 1994:19].

3 Object-Oriented Design and Development

A cursory discussion of the basic concepts of object-oriented design and development follows. Details on these topics can be found in books by Booch [1994], Cantu and Tendon [1994], Rumbaugh [1991] and Stroustrup [1991], *inter alia*. Object-oriented design methodology has to be supported by an object-oriented programming language but the methodology is independent of the choice of implementation language.

3.1 Essential Basic Concepts

In order to understand the software architecture described in section 4 one has to have a fundamental understanding of object oriented design. Using various graphical notations, the desired behaviour of the system and the roles and responsibilities of the objects that carryout this behaviour can be defined. It should be noted that the notation shown is not complete but rather the most relevant notation that is applicable to this design are presented.

3.1.1 Classes

3.1.1.1 The Nature of a Class

Classes are abstractions and is the essence of an object. Objects are *instances* of classes and according to Booch [1994:83], the terms instance and object are interchangeable. Classes act as a repository of all the general information that defines or characterises that particular type of object. A class is in a sense a template that has certain properties and models specific behaviour. For example `Mammal` is a class which represents characteristics common to all mammals. An object, being a specific instance and therefore having those characteristics of this class, could be a `dog` or a `cat`.

Pragmatically, creating classes provide the programmer with a means for creating new *types* that can be used with the same convenience as the built-in types provided by the programming language.

Stroustrup [1993] suggests that there are two distinct kinds of classes in a software system. The first type are classes that directly reflect concepts in the problem domain and represent user-level abstractions and generalisations. The second type are those that are artefacts of the implementation and may describe entities such as hardware or software resources and data structures. While the latter type of classes do not aid the definition of the problem domain they are useful artefacts in the design

or implementation of the system. Examples of the former type are `XfrLocator` and `Stand` while the latter are `GenList`.

Representing a concept or entity in the application domain as a class in the software design of the system is a non-trivial task, and typically requires a significant amount of insight. Details concerning the implementation of classes in the C++ programming language can be found in Swan [1995], Cantu and Tendon [1994] and Stroustrup [1991], amongst others.

At this stage the definition of a frequently used term from this point forth will be useful. *Instantiation* is the process of creating a specific instance (object) of a particular class. This term refers to the relationship between an object and its class.

3.1.1.2 Relationships Among Classes

Classes do not exist in isolation, and are related to one another in various ways to form a class structure which represents the system model. For example an amplifier, CD player, speakers and other components (classes) constitute a hi-fi and must be linked (relationship) in order to operate is some useful manner. Similarly, within the framework of object-oriented design certain basic kinds of class relationships can exist and these are described below. Class hierarchies are used to illustrate classes and their relationships to one another.

Association is the weakest and most general form of relationship between classes and infers the existence of a semantic link or some dependency. For instance, a class "Hammer" may have an association with a class "Nail". These weak associations are refined as the design of the system becomes more mature, and in the final implementation of the system these relationships are typically resolved into either *inheritance*, *aggregation* or *using* relationships [Booch, 1994:108].

Inheritance is a mechanism whereby one class inherits the behaviour and structure of another class. Inheritance is used to capture commonality present in different abstractions. Booch [1994] calls this an "is a" relationship. For instance, a car *is a* type of transport vehicle and therefore the class "Car" would *inherit* from the class "Transport Vehicle". "Transport Vehicle" is known as the base class, superclass or parent whereas "Car" is the *derived* class, subclass or child. A subclass may augment, redefine or restrict the superclass from which it is derived, as can be seen in the above example. Designing a suitable inheritance hierarchy requires intelligent classification.

Multiple inheritance is also possible, through which a class may be derived simultaneously from a number of different superclasses. Clearly, such classes would inherit characteristics from each of its parent classes.

Polymorphism is made possible through the use of inheritance. Polymorphism provides a means whereby a single name may denote objects of different classes, but which are related by some common superclass [Booch 1994:72]. By varying the parameters used to invoke the object, different responses can be obtained depending on which actual object is being referenced. This concept is implemented through the use of virtual functions which are not discussed since its not being utilised.

Aggregation denotes a whole/part relationship between classes in which one class *contains* an instance of another class. An example of this would be a class "Car" containing a class "Engine". This would result in a "Car" object containing an "Engine" object. In other words, aggregation allows for the formation of composite objects. Containment can either be physical or by reference. Physical containment, or containment by value indicates that the existence of the contained class is dependent on the existence of the class of which it is a part of. Instantiation of the enclosing class causes the component class to be created; destruction of the enclosing class causes it to be destroyed. Containment by reference allows each of the classes to exist independently of the other, and additionally allows parts to be shared structurally. Thus, two different class instances may contain references to the same object. A containment relationship implies that the container is able to "use" the contained class (refer to the next section).

Using relationships occurs between peer classes where one class takes on the role of a client while the other takes on the role of a supplier. The supplier provides a service that is requested by the client. In programming language syntax, a using relationship occurs when the signature of a class's member function contains the supplier class as a parameter. A using relationship also manifests itself in the form of a member function of a client class creating a local instance of a supplier class.

Instantiation in this context has a slightly different meaning to the one defined previously. A *parametrized class* (also known as *generic class*) is one that serves as a template that may be parametrized by others classes (containment of other classes). The structure and behaviour of a parametrized class are defined independently of its formal class parameters. Parameterized classes are often used for implementing container classes or collections such as arrays, stacks, lists and matrices. According to this definition, instantiation refers to the process of obtaining a specific instance of a template class parameterized with another class.

For instance, a generic `GenList` class is defined and used extensively in this system. This class would typically provide read and write access to the contained list of elements. To use the `GenList` class, one would need to instantiate it with a particular type of element, for instance with the class `Stand`. Now a particular instance (object) of a parametrized class (instatiation) can be created containing a list of `Stand` objects. Detailed information about templates and generic programming for C++ can be found in Booch [1994, 184] and Stroustrup [1991: 255-291].

3.1.1.3 Designing Quality Classes

It is not possible to go from the initial conceptual abstractions to the final stage in one step. The solution would inevitably advance in numerous intermediate stages, each of which would be a further refinement of the previous. In order to save time and to streamline the solution evolution process, Booch [1994:136-137] suggests five metrics for measuring the quality of design abstractions and they are discussed below.

The right level of *coupling* is desired with the rest of the system. Coupling can be understood as the strength of association between different abstractions. Weak coupling is desirable between non-related classes but tight coupling between superclasses and subclasses ensures exploitation of the commonality amongst the abstractions. On the contrary, strong interrelation amongst the different elements of a system makes each element harder to understand, modify and correct due to the effect it may have on the rest of the systems.

Classes should be *cohesive* so that the elements of a class or module all work together to provide some well bounded behaviour. It is undesirable to have abstractions that are totally unrelated existing in the same class or module, as it is simply poor design. In other words, each element in a class or module is functionally cohesive, if its semantics is of a clearly delimited abstraction or concept.

Classes must be *sufficient* in that they must capture enough characteristics of the abstraction in order to make it meaningful and useful. Classes should also be designed so that they are *complete*, so that all the meaningful characteristics of the abstractions are captured. They must also provide a general interface which is usable by a broad range of clients.

Finally, Booch states that classes should also be designed to be *primitive*. Stated another way, the operations defined for a class should be seen as a set of "basis" operations from which all higher level operations can be constructed. Thus creating functions that use a combination of other basis functions is discouraged.

3.1.2 Objects

3.1.2.1 The Nature of an Object

An object is a tangible entity that has a *state*, exhibits some well-defined *behaviour* and has a unique *identity*; the structure and behaviour of similar objects are defined in their common class [Booch, 1994:83-97] From a software point of view, an object is a self-contained unit which comprises a collection of related procedures and data structures. The properties stated as discussed by Booch are explained below.

The *state* of an object refers to its properties and the values contained by each of those properties at any particular time. An object's properties are those characteristics instilled by the class definition, and will not vary. However, the values assumed by the properties may be time varying, thus being in a different state for each variation.

Objects do not exist in isolation. The way in which an object acts and reacts in terms of its state changes and message passing represents its *Behaviour.* An object acts on another object by passing messages or invoking operations. The state of an object affects its behaviour, since typically some operations are only permissible under certain state conditions.

An object's *identity* is unique and is what distinguishes it from all other objects. Thus it follows that every object must exist at a unique location in memory and its identity is preserved over the lifetime of the object, though its name may not be. More than one name (*aliases*) may be used to reference the same object.

3.1.2.2 Relationships Among Objects

Like classes, objects do not exist in isolation but collaborate with each to produce a desired behaviour or output. Booch [1994:97] has identified two categories of collaborative relationships amongst objects, which are described below.

A **link** is a physical or conceptual connection or association between objects, through which a client object invokes operations or services provided by a supplier object. Links represent peer-to-peer or client/supplier relationships between objects, and are embodiments of the "Using" relationship for classes. Message passing between objects is constrained to occur only across links. A key consideration with respect to links is the question of visibility. In order for a message to be passed from one object to another, the supplier object must be visible to the client object in some way.

Aggregation denotes containment whereby an object contains other objects comprising its various parts. Object aggregation occurs as a result of instantiating classes participating in an aggregation relationship. It is through this mechanism that objects may be encapsulated as part of other objects.

3.2 The Object-Oriented Design/Development Methodology

There is no definitive object-oriented design/development methodology. Design methodologies have been proposed by Rumbaugh [1991], Stroustrup [1991], Booch [1994] and Meyer (described in Cantu and Tendon [1991]), amongst others. Although some methodologies are detailed and formal, others are less precise and offer only guidelines. All of these methodologies are similar but differ mainly in the emphasis of various aspects of the design process.

Booch's [1994:169-290] methodology has been adopted for this project because it is very formal and complete. Furthermore, powerful design tools are available to support development using this methodology which was an additional incentive. The *Rational Rose* tool, produced by the Rational Object-Oriented Engineering Software company, was extensively employed in the design process. This tool allows rapid creation of class, object, module and interaction diagrams without burdening the designer with the notation. All the diagrams presented in this document were generated using this tool.

A brief discussion of what Booch's methodology entails follows. Booch [1994: 230] has identified two features that characterise successful object oriented software projects.

1. Strong architectural vision
2. An iterative and incremental development life-cycle

Strong architectural vision implies that the system has conceptual integrity and a well defined internal class and object structure. A well defined architecture is expected to have clearly defined layers of abstraction; a clear separation between implementation and interface; and simplicity in structure. Conceptual integrity in the system's architecture makes the system understandable, extensible and maintainable which reduces developmental risk.

Traditionally, the software design approaches have been exclusively *top-down* or *bottom up* which did not lend itself to the object oriented paradigm. A well structured iterative software development life cycle is required so that processes can be managed, controlled and measured. The system architecture, strategic and tactical decisions are successfully refined with every iteration. Most importantly, it must simultaneously

provide sufficient freedom for creativity and innovation. Booch's methodology endeavours to promote the above traits.

The methodology comprises of two collaborative processes; the micro development process and the macro development process. The micro development process is iterative representing the daily activities of the software developer. The macro process imposes a rational design process on the activities of a complete development process over the life-cycle of a project. The micro developmental process which allows for innovation and the exploration of alternative architectures repeats within the controlling framework of the macro process.

3.2.1 The Micro development Process

The stages that comprise the micro development process [Booch, 1994: 235] is illustrated in Figure 1.

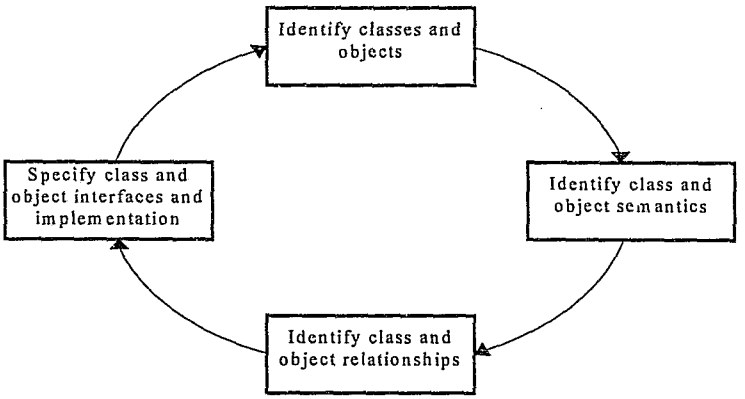


Figure 1: The Micro Process

The phases depicted are explained in detail in the following sections:

Identify Classes and Objects This is the task of identifying the components that define the vocabulary of the problem domain. Decomposition of the system is achieved by the identification of classes and objects which will serve to define the boundaries of the problem. During this phase, commonality among abstractions is identified and grouped so that it can be utilised advantageously in an inheritance hierarchy.

Identify Class and Object Semantics This refers to the process of defining the behaviour and attributes of the classes and objects which have been identified in the previous phase. Allocating responsibilities

defines and separates the roles of each abstraction, thus leading to the specifications of the interface of each class.

Identify Class and Object Relationships Identifying semantic dependencies between abstractions defines the way in which objects and classes must collaborate to achieve the required system functionality. Initially, this is accomplished through the process of identifying associations between classes.

Specify Class and Object Interfaces and Implementation In this stage, structures and algorithms are devised, which provide the desired properties and behaviours of the abstractions which have been identified. This process is a way of creating a more tangible representations of the abstractions identified which are then further refined as a result of the pragmatic consequences of implementation. Analysis of each abstraction will probably lead to the identification of new classes and objects, at a different level of abstraction, which will aid the implementation aspects of the abstraction currently under analysis. Thus this starts another iteration of the micro developmental cycle where more classes and objects are identified.

3.2.2 The Macro Process

The macro development process shown in Figure 2 [Booch 1994: 249] summarises the main activities which constitute this process.

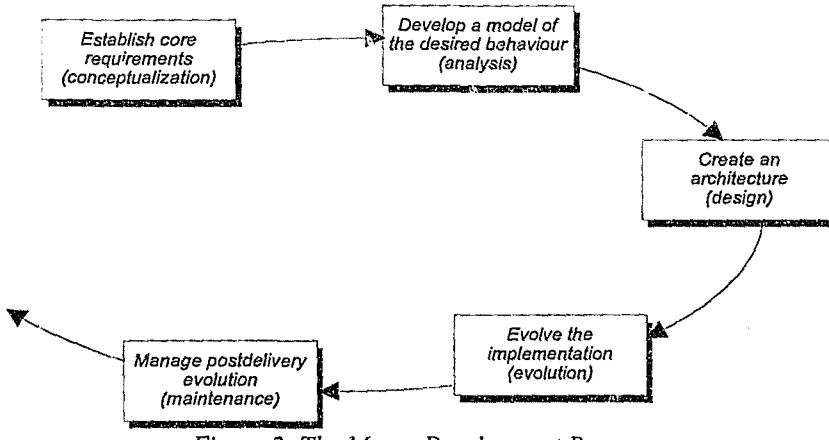


Figure 2: The Macro Development Process

The details of these processes are described in the following sections:

Conceptualisation Producing one or more prototypes is the primary product of this stage. In order to accomplish this, the core requirements of the system being developed must be establishing. Various ideas for the applications are explored and its assumptions validated. This is a

creatively intensive stage and therefore should not be restricted by rigid development rules.

Analysis The result of this stage is a comprehensive description of the problem followed by the derivation of the functional requirements of the solution. Emphasis is placed on the behaviour of the system rather than its precise form. During the analysis, the solution is modelled as classes and objects that constitute the vocabulary of the problem domain.

Design This refers to the process of designing the software architecture for the evolving implementation, and to establish the common tactical policies that must be used by disparate elements of the system. The logical architecture (section 4.1) is expressed through the use of class diagrams and object diagrams, while the physical architecture (section 4.2) is conveyed by means of module diagrams.

Evolution This evolutionary phase refers to the process of product development where successive refinement or optimisation of the software implementation ultimately leads to the production system. The evolution of an architecture is endeavouring to simultaneously satisfy various competing constraints, such as functionality, performance and physical resources. The primary products of this phase are a number of incremental executable releases of the of the software.

Maintenance This is concerned with the post delivery management aspects of the system produced. Evolution to the system will occur but at a localised level rather than a global level. Reasons for the continued system evolution are twofold; addition or modification of the existing system behaviour to meet new requirements and the elimination of latent defects (program bugs).

3.2.3 Defining the Project Within the Object Oriented Paradigm

A primary goal of this project is to deliver the first fully operational version (or "first major release"), consequently undergoing one complete cycle of Booch's macro development process. Booch [1994:249] notes that the macro process invariably repeats itself after major product releases. Therefore according to Booch, the project is expected to undergo several macro development process cycles as maintenance aspects conceived by those involved in this research program are implemented.

The system developed will be a prototype. Booch [1994: 250] states that "prototypes are by their very nature incomplete and marginally engineered", thus being utilised to merely validate concepts. To say that something is "marginally engineered" does not imply that it is badly engineered or ill defined. A marginally engineered product is one in which any aspect of design and development, which is not crucial to the proof of

concept, is either ignored or implemented minimally. All concerns peripheral to validating the central idea are not explored, but discussed for the purposes of integration with external or third party system. The system's *user interface* has been classified as peripheral concerns for the transformer location program because they are not essential for proving the concepts at hand. However, a flexible architecture for the interface must exist so that at a later stage it can be evolved to incorporate complex communication mechanisms. For this research, third party software is used for visualisation of the results (MicroStation™ produced by Intergraph).

3.3 Booch Notation

The Booch notation is used for this project since a well-defined and expressive notation is important to the process of software development Booch [1994:171]. Design decisions made in formulating the architecture of the system can be unambiguously communicated to others through the use of a standard notation. A further advantage of using this notation is that there are powerful tools available for consistency and correctness checking. As mentioned before, the *Rational Rose®* tool by Rational Software has been extensively used in this design process.

Booch's notation caters for a number of different views that can be used to comprehensively describe a software system. This notation provides facilities for describing both the logical and the physical architecture of the system. The logical view of the system is conveyed by means of class diagrams and object diagrams, while the physical view of the system is described through the use of module diagrams and process diagrams.

In addition, the notation provides mechanisms for describing the time-varying behaviour and dynamic semantics of the system, by means of interaction diagrams and state transition diagrams. The purpose of interaction diagrams and state transition diagrams is to indicate timing sequences of the behaviour demonstrated by some system. No state transition diagrams are given for this design as none of the system classes exhibit significant event-ordered behaviour. The graphical notation which is used in each of these diagrams is described in the following sections.

3.3.1 Class Diagrams

The purpose of the class diagrams is to show the existence of, and the relationships between the classes which make up the system. The notation used for class diagrams is summarised in Figure 3. It should be noted that relationship that are not concerned with this design are not represented below.

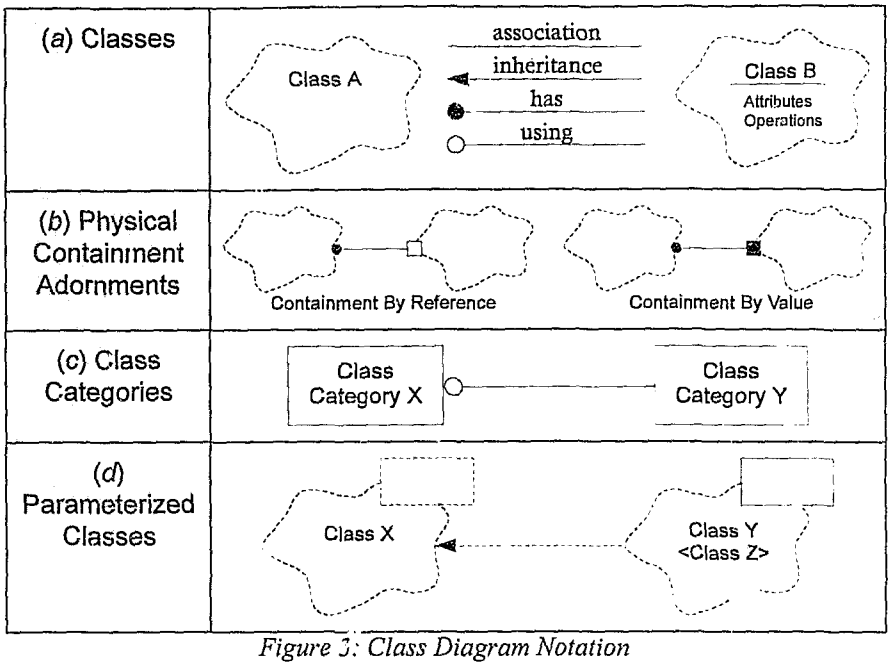


Figure 3: Class Diagram Notation

As shown in Figure 3*a*, classes are represented by an amorphous blob or cloud icon, drawn with a broken outline indicating that the class is an abstraction and must be instatiated to be usable. Every class should have a name written inside the icon which must be unique. Desired attributes and operations may be listed below the class name, beneath the separating line. The four types of relationships between classes, discussed in section 3.1.1.2, are shown in a class diagram by means of the four variations of connecting lines between classes (Figure 3*a*). Thus the diagram as shown would be interpreted as, class B is respectively associated with, derived from, part of, or used by class A. Adornments to the containment (aggregation) relationship connecting line may be used to show the nature of physical containment (Figure 3*b*).

When systems grow and become complex, the class becomes an insufficient vehicle for decomposition. In a system where there are many abstractions, it becomes possible to identify clusters of classes which are cohesive but can be loosely coupled to other clusters. The use of class categories allows the expression of complex high-level logical architecture of the system. Class categories are represented by rectangles and the only relationship which is shown between class categories is the *using* relationship (Figure 3c). A *using* relationship between class categories means that the client category can use, inherit from, contain or otherwise associate with classes in the supplier category.

A dashed rectangle adornment is used to represent a parameterized class. An instantiated parameterized class is depicted using a *solid* rectangle adornment, denoting the instantiated class' actual parameters. The instantiation relationship itself is indicated by a dashed arrow from the instantiated class to the parameterized class from which it was instantiated.

3.3.2 Object Diagrams

Mechanisms represent common collaborations or patterns of behaviour among the various objects in a system. These patterns may be discovered by considering scenarios typical to the system. A scenario is a sequence of events that represent aspects of system operation. Scenarios and mechanisms are illustrated through the use of object diagrams.

Object diagrams describe the objects and the interactions that take place between these objects during a slice of run-time. Interactions are illustrated through the use of messages (which represent an invocation of an object's function) that pass from one object to another. Object diagrams show the existence of, and the relationships between objects. The notation used for object diagrams is shown in Figure 4 [Booch 1994: 208].

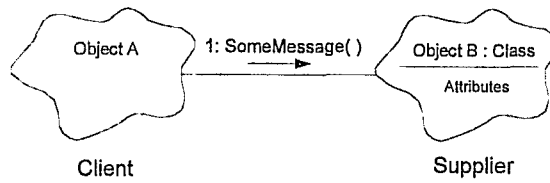


Figure 4: Object Diagram Notation

To represent objects, the amorphous blob, or cloud icon is again used, but the boundary of the icon is solid, indicating that the object is a concrete entity. The name of the object should appear inside the icon, and if necessary, followed by the object's class, separated by a colon. Selected attributes may be shown beneath a separating line. Links which exist between objects are indicated by a solid connecting line. Operation invocation or message passing is represented by a directed arrow, with the operation name and a sequence number for the message alongside it. Thus by following a sequence of numbers one can obtain an understanding of that particular scenario.

3.3.3 Module Diagrams

The physical architecture details the concrete hardware and software composition of the system's implementation [Booch, 1994: 175]. In other words, module diagrams show the allocation of classes to modules in the

physical implementation of the system. Modules represents files which contain portions of the implementation code which are the physical artefacts of a software system. The notation used for module diagrams is depicted in Figure 5 followed by a brief description.

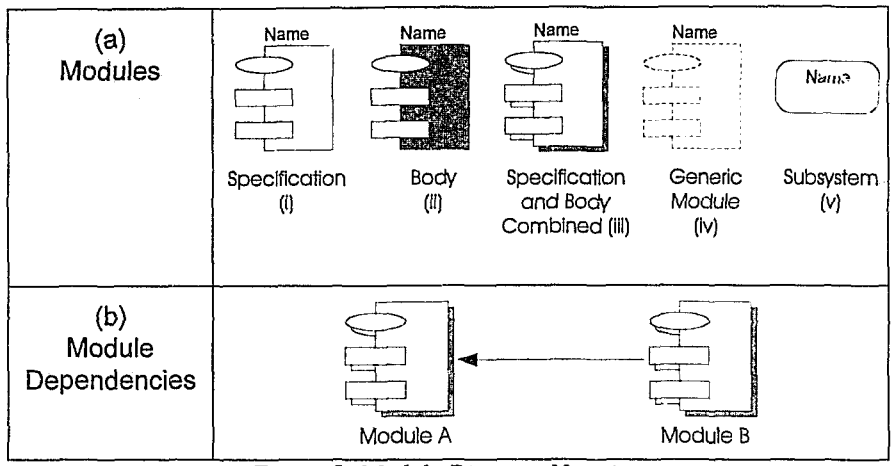


Figure 5: Module Diagram Notation

The icons used to depict the various types of modules are shown in Figure 5*a*. It is considered good practice to have a single file that contains the class declarations and a separate file that contains the definitions. The class declaration file contains the class specifications which defines the "contractual duties" that it is capable of performing if invoked. The definition file is usually not visible to the user as it contains the method implementation. A *specification* module (Figure 5*a*(i)) contains class declarations, while a *body* module (Figure 5*a*(ii)) contains class definitions (code implementations). In accordance with Booch's convention, these two files have the same filename, except for a distinguishing extension [Booch, 1994:220]. As a consequence of the system having been implemented in C++, the declaration files all have the "`.h`" extension, while the definition files have the "`.cpp`" extension. Together, these two files comprise a single module in the physical implementation of the system, referred to as a *package* (Figure 5*a*(iii)).

A *generic* module consists of a header file and a body file of a template class. Different notation is used for generic modules due to the fact that compilers treat generic classes differently to ordinary classes. Normal classes are usually pre-compiled but template classes are compiled only when they are instantiated. As a consequence of this, all clients of template classes must include both the definition and declaration of such classes since the pair form the generic package (Figure 5*a*(iv)).

Subsystems represent clusters of logically related modules (Figure 5*a*(v)) and other subsystems. Subsystems are analogous to the class categories

present in class diagrams. Subsystems may be dependent on modules or other subsystems. Every module in a system either belongs to a subsystem or exists at the top level.

Only one type of relationship exists between modules. A compilation dependency (as depicted in Figure 5*b*) in the C++ programming language is articulated through the use of the "#include" pre-processor directive. All dependencies shown should be taken to be dependencies between module *specifications*. For example Figure 5*b* would be interpreted as; module B is dependent on module A.

4 System Architecture

This section endeavours to describe the system architecture for the transformer zoning program. The logical architecture is first described in section 4.1, followed by the physical architecture in section 4.2. Booch notation [1994:171-229] is used extensively from this point forth and it is assumed that this notation is understood by the reader.

4.1 Logical Architecture

The logical architecture of the system describes the key concepts or abstractions that form the vocabulary of the system. The logical architecture of the system is explained here with the aid of the various types of Booch diagrams. No state transition diagrams are given as none of the system classes exhibit significant event-ordered behaviour. The following section is an aside detailing the source code contributions to the project. After this, the discussion of the system's class hierarchies begins.

Note: `Class names are indicated through the use of this font.`
Class categories are indicated through the italicisation of this font.

4.1.1 Source Code Contributions

The generic container classes `DlListElem` and `GenList` documented are not original. The code for the following classes was adopted from that produced by Stephen Levitt as part of another MSc project. These classes are purely being used for maintaining linked lists (explained in section 4.1.2.1.1.) and for its associated management purposes. This is an example of reusability, since it saves valuable time in writing and debugging code. Reasons for using a generic container are justified in section 4.1.2.1.

4.1.2 Class Categories

The diagram in Figure 6 shows the class categories. This architecture relies on the class category *Controlling Unit* to regulate or control the transformer zoning process. More specifically this class category controls the timing sequence of the entire process. The *Input* class category is responsible for acquiring all the required graphical and user data in order for the process to occur. The class category *Square Extractor* is utilised to implement the square extraction technique. An entire class category was allocated to this aspect of Grimsdale's algorithm as it became complex

after careful analysis (the reasons will become apparent in section 4.1.2.2). When the execution of the square extraction technique is completed, the computed initial transformer locations are used by the class category *Transformer Locator* to compute better or refined locations. This class category completes the rest Grimsdale's algorithm and then proceeds to execute all the refinement processes (scanning and refinement algorithm). It is anticipated that the bulk of the work will be in this class category. Finally, the class category *Output* will be employed to create an output of a CAD file with the computed transformer zones.

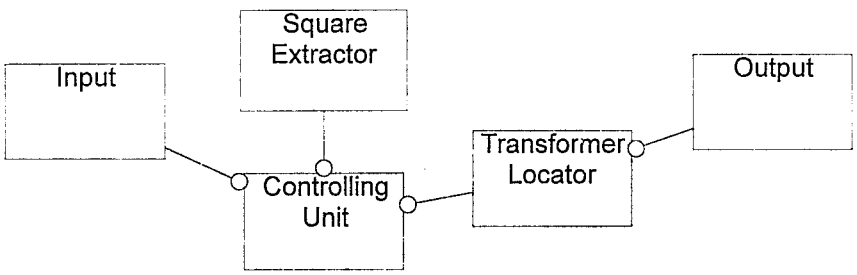


Figure 6 - Class Categories

The interface (*Input* and *Output*) class categories are separate from the rest of the process to ensure that any interface related modifications will have to be done to these without interfering with any of the other processes. This design is thus in conformance to the functional specifications.

Thus the overall logic of the operation of this software unit has now been described. The following sections discuss each class category in turn and for each category the following three subsections exist:

1. *Design Issues:* The purpose of this subsection is to provide strategic issues and the motivation for the decisions made in designing a particular class category. The tactical design choices and the strategic issues involved cannot be fully reflected in the class diagrams and their respective descriptions since they apply to the class category as a whole.

2. *Class Structure:* A class diagram is used to show the structure of the classes constituting each category. The relationships between the classes are illustrated and, where necessary, elaborated upon in the subsequent discussion of the responsibilities of each class.

3. *Class Responsibilities:* The roles and responsibilities, that characterise the behaviour of each class is defined. All the internal class details such as data structures and member function algorithms are described in the low level design of this project (doc no B5).

after careful analysis (the reasons will become apparent in section 4.1.2.2). When the execution of the square extraction technique is completed, the computed initial transformer locations are used by the class category *Transformer Locator* to compute better or refined locations. This class category completes the rest Grimsdale's algorithm and then proceeds to execute all the refinement processes (scanning and refinement algorithm). It is anticipated that the bulk of the work will be in this class category. Finally, the class category *Output* will be employed to create an output of a CAD file with the computed transformer zones.

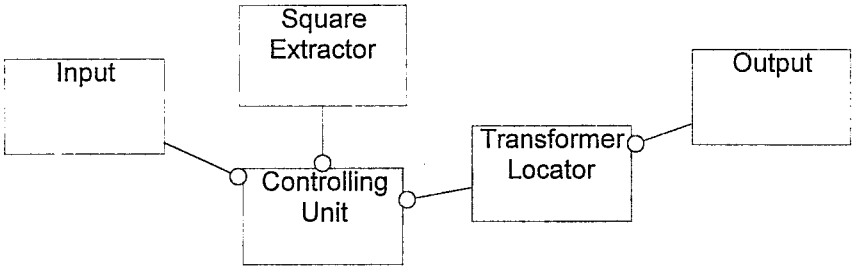


Figure 6 - Class Categories

The interface (*Input* and *Output*) class categories are separate from the rest of the process to ensure that any interface related modifications will have to be done to these without interfering with any of the other processes. This design is thus in conformance to the functional specifications.

Thus the overall logic of the operation of this software unit has now been described. The following sections discuss each class category in turn and for each category the following three subsections exist:

1. *Design Issues:* The purpose of this subsection is to provide strategic issues and the motivation for the decisions made in designing a particular class category. The tactical design choices and the strategic issues involved cannot be fully reflected in the class diagrams and their respective descriptions since they apply to the class category as a whole.

2. *Class Structure:* A class diagram is used to show the structure of the classes constituting each category. The relationships between the classes are illustrated and, where necessary, elaborated upon in the subsequent discussion of the responsibilities of each class.

3. *Class Responsibilities:* The roles and responsibilities, that characterise the behaviour of each class is defined. All the internal class details such as data structures and member function algorithms are described in the low level design of this project (doc no B5).

after careful analysis (the reasons will become apparent in section 4.1.2.2). When the execution of the square extraction technique is completed, the computed initial transformer locations are used by the class category *Transformer Locator* to compute better or refined locations. This class category completes the rest Grimsdale's algorithm and then proceeds to execute all the refinement processes (scanning and refinement algorithm). It is anticipated that the bulk of the work will be in this class category. Finally, the class category *Output* will be employed to create an output of a CAD file with the computed transformer zones.

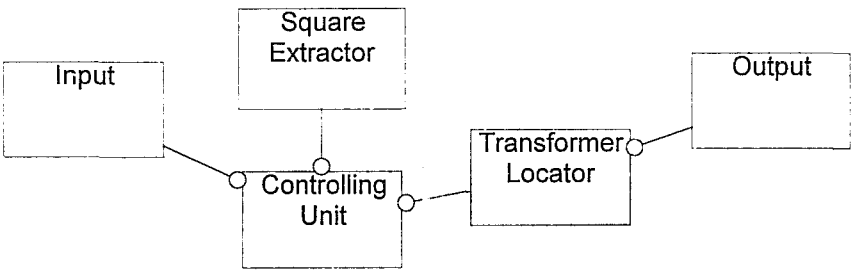


Figure 6 - Class Categories

The interface (*Input* and *Output*) class categories are separate from the rest of the process to ensure that any interface related modifications will have to be done to these without interfering with any of the other processes. This design is thus in conformance to the functional specifications.

Thus the overall logic of the operation of this software unit has now been described. The following sections discuss each class category in turn and for each category the following three subsections exist:

1. *Design Issues:* The purpose of this subsection is to provide strategic issues and the motivation for the decisions made in designing a particular class category. The tactical design choices and the strategic issues involved cannot be fully reflected in the class diagrams and their respective descriptions since they apply to the class category as a whole.

2. *Class Structure:* A class diagram is used to show the structure of the classes constituting each category. The relationships between the classes are illustrated and, where necessary, elaborated upon in the subsequent discussion of the responsibilities of each class.

3. *Class Responsibilities:* The roles and responsibilities, that characterise the behaviour of each class is defined. All the internal class details such as data structures and member function algorithms are described in the low level design of this project (doc no B5).

The next section, however, discusses generic containers as this is utilised throughout the system.

4.1.2.1 Generic Container

Generic classes encapsulate commonly used data structures (which are not specific to this application). The generic container on offer is used by other classes in the system which require their services. The purpose of using a container class is for containing a list of data objects and a full discussion follows in the subsequent section.

4.1.2.1.1 Design Issues

In writing programs, there is often the need to store various lists of data. For example, in this project, it is necessary to hold a container of `Stand` objects and a container of vertices (`Point`) defining a polygon. Storage of information of this nature can be easily accomplished through the traditional use of arrays. However, the use of arrays has a major drawback in that in can only be declared with a given size and cannot be dynamically resized at a later stage. It is unreasonable to define large arrays so that the situations being dealt with never exceed the maximum capacity, since this results in memory wastage.

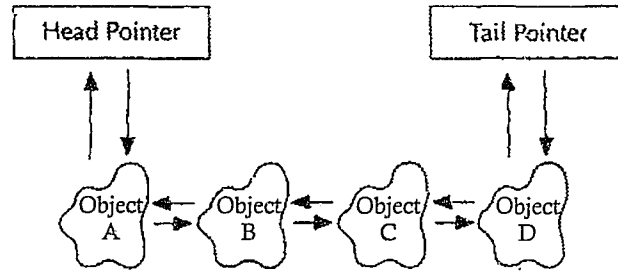


Figure 7 - Structure of a doubly linked list [Parsons: 1995: 211]

A more elegant solution is to use linked lists, which allow the size of the container (container containing the list of elements) to vary as objects are added or removed. A list can be implemented using pointers where one pointer references the first object in the list and then each object points to the subsequent object in that list. This list can only be accessed in one direction and sequentially. The doubly linked list contains two reference pointers to access both ends of the list (Parsons 1995: 211, shown in Figure 7). This simplifies certain operations such as element insertion and deletion. The linked lists are used extensively in this project and therefore

is being discussed. The primary reason for the usage of linked lists is that it ensures the efficient use of memory.

The main debate concerning the implementation of container classes is whether to use a polymorphic approach or a template-based approach. A polymorphic approach requires that the class of any object (which is to be contained) to be derived from the base class around which the container is designed.

Cantu and Tendon [1994:326] point out that the choice between the constructs of genericity and polymorphism is an open issue and that the decision should rest on which construct fits better into the design and can be more easily implemented.

In this design, templates are used to construct the container classes. Template container classes have unconstrained genericity as opposed to polymorphic genericity, which is limited to inheritance hierarchies. By "unconstrained genericity" it is meant that a container may contain and manipulate *any* type. The generic container functions by containing a list of nodes containing an object and two pointers. One pointer points to the next element while the other points to the previous element so that the nodes can be 'chained' into a linked list. The deciding factor for employing template classes is that they can work directly with arithmetic types. This property was the important as it was necessary to build a Matrix as explained in section 4.1.2.3. However, during the implementation phase, a new `Matrix` class was designed and implemented as complications were encountered with the use of generic classes.

In order to achieve the same effect with a polymorphic container, every type of matrix element would have to be derived from a base "matrix element class". This would require that arithmetic types to be wrapped up within a class, which is inelegant and adds unnecessary overhead.

Lastly, it is worth noting that achieving genericity and reusability comes at the cost of functionality. The methods that a container class can perform on its contained objects are, of course, very limited if any type of object can be contained. Specialised functionality cannot be built into a container because the functionality required cannot be anticipated as it is not known what will be contained. If more specialised containers are required then they can be derived from an instantiated template container class and other functionality added. For this project, no such additional functionality was not required and the only requirement was the ability to maintain elements in a linked list. All the elements, where necessary, will be manipulated externally.

is being discussed. The primary reason for the usage of linked lists is that it ensures the efficient use of memory.

The main debate concerning the implementation of container classes is whether to use a polymorphic approach or a template-based approach. A polymorphic approach requires that the class of any object (which is to be contained) to be derived from the base class around which the container is designed.

Cantu and Tendon [1994:326] point out that the choice between the constructs of genericity and polymorphism is an open issue and that the decision should rest on which construct fits better into the design and can be more easily implemented.

In this design, templates are used to construct the container classes. Template container classes have unconstrained genericity as opposed to polymorphic genericity, which is limited to inheritance hierarchies. By "unconstrained genericity" it is meant that a container may contain and manipulate *any* type. The generic container functions by containing a list of nodes containing an object and two pointers. One pointer points to the next element while the other points to the previous element so that the nodes can be 'chained' into a linked list. The deciding factor for employing template classes is that they can work directly with arithmetic types. This property was the important as it was necessary to build a Matrix as explained in section 4.1.2.3. However, during the implementation phase, a new `Matrix` class was designed and implemented as complications were encountered with the use of generic classes.

In order to achieve the same effect with a polymorphic container, every type of matrix element would have to be derived from a base "matrix element class". This would require that arithmetic types to be wrapped up within a class, which is inelegant and adds unnecessary overhead.

Lastly, it is worth noting that achieving genericity and reusability comes at the cost of functionality. The methods that a container class can perform on its contained objects are, of course, very limited if any type of object can be contained. Specialised functionality cannot be built into a container because the functionality required cannot be anticipated as it is not known what will be contained. If more specialised containers are required then they can be derived from an instantiated template container class and other functionality added. For this project, no such additional functionality was not required and the only requirement was the ability to maintain elements in a linked list. All the elements, where necessary, will be manipulated externally.

4.1.2.1.2 Class Structure

The classes which are derived from the template classes are depicted in Figure 8 (`Polygon`, and `LStands`) and are discussed in the respective categories.

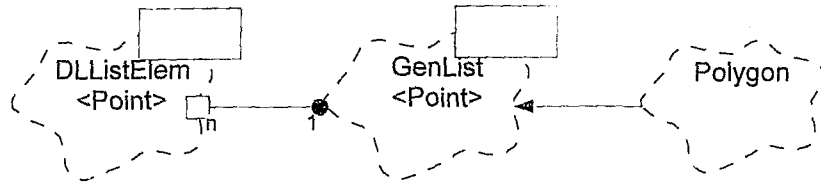


Figure 8 - Generic Container for Class <Point>

The class `Point` can contain two vales (*x* and *y* co-ordinate) that can be used to define a unique point in space. The generic container `GenList<Point>` is a container of a list of vertices (`Point`'s) that can be used to represent a polygon. `Polygon` was originally derived with the intention of adding extra functionality but the need was obviated in the implementation phase. However, `Polygon` was not dispensed with as it had not only a meaningful name but also enabled easier management of its pointers.

In addition, a linked list is being utilised to contain all the stand labels and their respective co-ordinates as shown in Figure 9. The class `LStands` was derived for the same reason as `Polygon`, mentioned above, and has no additional functionality. Class `Stand` is also derived from `Point` since this represents the co-ordinate of the text node. The class `Stand` contains a field that designates it to a transformer. It was decided that this would be the easiest way of maintaining track of the consumer to transformer assignment in the *Transformer Locator* class category.

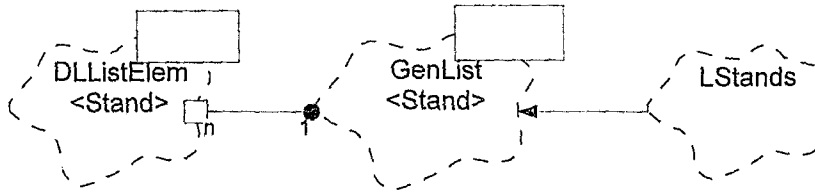


Figure 9 - Generic Container for Class <Stand>

4.1.2.1.3 Class Responsibilities

`GenList` class encapsulates the concept of a doubly linked list. The list size and status can be accessed. New elements can be appended onto the list, and elements can be removed from the list. As stated earlier, the most notable aspect is that this list is generic and thus may contain

elements of any type. Classes `LStands` and `Polygon` are the ones that are actively used in the design.

4.1.2.2 Class Category *Input*

The *Input* class category is responsible for acquiring all the required data in order for the processes to occur. This data involves not only the information regarding the map of the township but also the user specifications such as number of transformers and zone limits.

4.1.2.2.1 Design Issues

Primary consideration in the design of this class category was that any modifications necessary for this product to interface with others is minimised. In keeping with this ideology, this class category is straight forward consisting of only one class (`Input`). The definition of this class can be re-designed and implemented without affecting any of the other classes provided that its interface mechanisms are still maintained.

4.1.2.2.2 Class Structure

The class diagram in Figure 10 shows the class `Input` and other required classes.

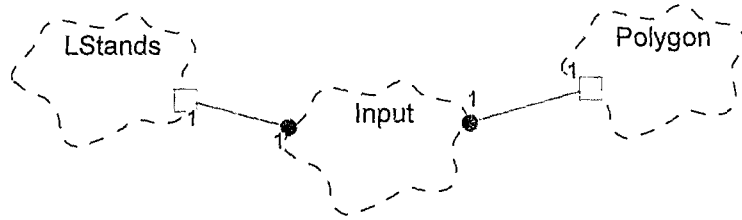


Figure 10 - Class Category Input

4.1.2.2.3 Class Responsibilities

`Input` obtains the source filename for the CAD file from the user. The CAD file is then accessed and two linked lists are created. Namely, a list of all the stands (`LStands`) and a list of all the vertices defining the boundary polygon (`Polygon`). Note that both containers are being contained by reference. This assures the continued existence of those lists once `Input`, which was the class that instatiated them, has been destroyed. The pointers to these lists are returned to the *Controlling Unit* class category so that it can then access both these lists. Other user inputs are also returned to the *Controlling Unit* class category.

4.1.2.3 Class Category *Square Extractor*

The class category *Square Extractor* is utilised to determine *n* nodes representing the starting transformer positions that are sufficiently far part. In effect, this class category encompasses all the logic and functionality of the square extraction technique described by Grimsdale.

4.1.2.3.1 Design Issues

As required by first part of Grimsdale's algorithm, the township area must be broken up to contain squares. This was accomplished by representing the map in the form of an matrix. In creating the matrix, the scan conversion algorithm is used and is defined as follows: it can be determined whether a point lies within a polygon or not by counting intersections of the boundary with an imaginary line extending from the point to some other point located at infinity to the polygon [Newman 1979:232]. In order to efficiently determine intersections, it was decided that it would be easier to work with a list of lines rather than manipulate the vertices in the `Polygon`. Thus class `Line` was conceived which has the added functionality of containing the calculated gradient and the y-intercept. This would alleviate a lot of repetitive calculations that would be necessary if the scan conversion algorithm operated on a list of vertices.

4.1.2.3.2 Class Structure

The class diagram in Figure 11 shows the class `Map` which is responsible for the square extraction process. All other class shown are there to assist class `Map`.

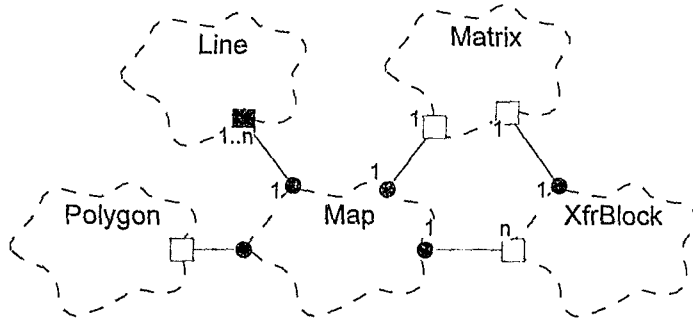


Figure 11 - Class Category Square Extractor

4.1.2.3.3 Class Responsibility

The name `Map` is not truly representative of the functionality of the class as it evolved substantially from conception to the form presented here.

`Map` is evoked with the two inputs (`Polygon` and `LStands`) as stated in 4.1.2.3.1 as the arguments and contains all the processing algorithms for the square extraction technique.

The first operation would be to create a list or an array of `Line`'s computed using the vertices from `Polygon`. Thus, `Line` contains two sets of co-ordinates or, more specifically, two instances of `Point`. Once the two pints are specified, `Line` automatically determines the gradient and y-intercept. These factors are necessary in determining the intersections with the imaginary line when using the scan convection algorithm. An array is dynamically declared to store all the `Line`'s. An array was used instead of a generic container as its quantity can always be easily determined by the number of vertices in `Polygon` and the size of the array would remain fixed for a single execution of the program. The scan converting algorithm is then used to create the matrix representing the map (`Matrix`). `Matrix` is contained by reference so that it can be passed to the class `XfrBlock` (abbreviation for *Transformer Block*).

`Map` instantiates as many `XfrBlock`'s as the number of transformers that have to be placed. The class `Map` then execute the square extraction technique described above. The class `XfrBlock` is responsible for placing itself in the `Matrix` without placing it over another instance of itself whenever it is called upon to do so by `Map`. Every instance of the class `XfrBlock` is unique and thus aware of the space its occupying within the township. `Map` will complete the square extraction technique and returns a pointer to the array of `XfrBlock`'s to be used in subsequent processes by other class categories. Each instance of the class `XfrBlock` is used to calculate and contain the co-ordinate of a particular transformer location.

4.1.2.4 Class Category *Transformer Locator*

The objective of the class category *Transformer Locator* is to firstly complete Grimsdale's algorithm and then to compute better or refined transformer locations. The reason for implementing the latter part of Grimsdale's algorithm in this class category and not in *Square Extractor* was because the centroid calculating routine could be located here so that it can also be utilised by the sweeping algorithm.

4.1.2.4.1 Design issues

The design aspects here are straight forward and all the processing algorithms are encompassed within one class `XfrLocator`. This simplifies the task of passing computed data from one class to another creating unnecessary overhead. There was no need for a complicated supporting class architecture as with the *Square Extractor* class category.

However a limited number of additional classes would be necessary to aid the process. For instance, this category has a class `XfrBoundary` that can group stands belonging to a specific transformer and define a boundary or polygon around that group. Thus every instance of this class is responsible for delineating its own boundary.

4.1.2.4.2 Class Structure

The class diagram in Figure 12 shows the class `XfrLocator` which is responsible for determining the most suitable consumer to transformer assignment configuration.

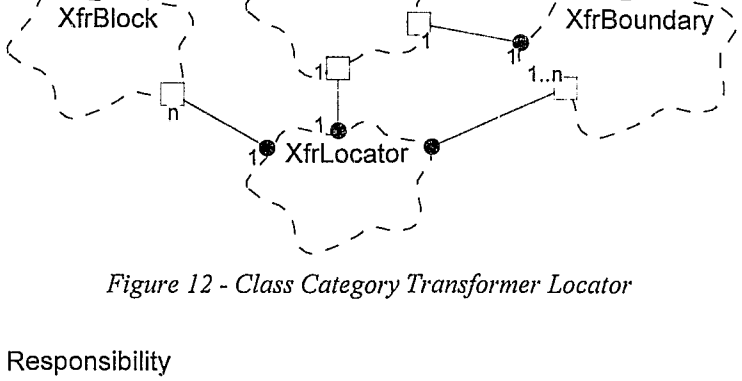


Figure 12 - Class Category Transformer Locator

4.1.2.4.3 Class Responsibility

The array of `XfrBlock`'s and `LStands` are passed as an argument to `XfrLocator`. All the processes of consumer to transformer allocation resulting from the operation of the sweeping and refinement algorithm are implemented in this class. Any modifications, or improvements to the algorithm that may have to be implemented in future are to be done in the class `XfrLocator`.

The class `XfrBoundary` is instantiated only at the completion of all the processing algorithms. The number of these objects instatiated will naturally correspond to the number of transformers that are being placed and will have a unique identifier corresponding to every transformer. On completion of the scanning algorithm, a reference to `LStands` is also passed to `XfrBoundary` as an argument. Each `Stand` object would have been designated to a particular transformer by then. Thus, every instance of `XfrBoundary` can identify those `Stand`'s assigned to itself and compute a polygon that can be used to encircle its consumers, thus defining its own zone boundary.

4.1.2.5 Class Category *Output*

The role of the class category *Output* is to create an output of a CAD file with the computed transformer zones.

4.1.2.5.1 Design Issues

As with the class category `Input`, it was important to develop a class category that can be easily modified so that this software unit can interface with other software products. Only one class `Output`, which encapsulated all the necessary behaviour is defined. In this research, the primary objective of this class was to create a copy of the CAD file being read with the polygons delineating the transformer zones appended.

Two alternatives existed in the usage of this class and they are discussed, since they influenced the communication mechanisms:

An option was to evoke the instance of the class `Output` and pass to it a pointer to the array of `XfrBoundary`'s. Then `Output` will access every instance of `XfrBoundary` and extract the vertices from it. This is typically a client server relationship where the one class requests the other class for a service or, in this case, data. Disadvantages are that the message passing is bi-directional and thus requires overhead to facilitate this particular communication mechanism. Furthermore, the algorithms in `Output` responsible for the extraction of the data must depend on the usage of the class `Polygon`. Thus any modifications to the class `Polygon` could significantly impact the class `Output`.

The other option was to pass a reference of an instance of the class `Output` to every instance of the class `XfrBoundary`. Then `XfrBoundary` can pass the co-ordinates of the vertices to `Output` defining the transformer zone polygon. There are advantages to using this communication mechanism for the following reasons. Firstly, the containment of a list of vertices in `XfrBoundary` implies that should the linked list management or any of its functionality change in future, it will not affect the process of data transfer. Secondly, should the implementation of class `Output` change then the data receiving mechanism does not have to be altered. When the communication mechanisms are simplified, it inevitably leads to a less complex yet more elegant and fail safe solution. For these reasons, the latter option was employed.

During the test phase, an instance of the class `Output` will be also passed to `XfrBlock` enabling the evaluation of the square extraction technique. The principles are the same as for class `XfrBoundary`. The results of the square extraction technique are irrelevant for the final output. However, it would be useful for testing the square extraction

technique to examine the effects of rotating the map of the township as described in the Testing and Results document (B4).

4.1.2.5.2 Class Structure

The class structure for `Output` is shown in Figure 13.

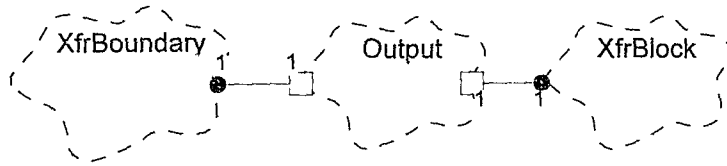


Figure 13 - Class Structure for <Output>

4.1.2.5.3 Class Responsibility

This class `XfrBoundary` is responsible for creating the transformer boundary or zone polygon. Every `Stand` in the list, by this stage, would have been assigned to a specific transformer. Every instance of `XfrBoundary` would have been designated to a unique transformer and each, considering all those stands allocated to itself, would define a polygon encompassing all those consumers. As the co-ordinates of the transformer are computed they are sent to the instance of `Output`. For testing purposes only, `XfrBlock` would also use `Output` in the same manner.

Class `Output` is responsible for all the outputs of this software unit. It accumulates the vertices that are passed to it and constructs a polygon and appends that information to the DXF file.

4.1.2.6 Class Category *Controlling Unit*

The task of the class category *Controlling Unit* is to control the transformer zoning process. This is being discussed last as it incorporates all the other class categories.

4.1.2.6.1 Design Issues

As required in the specifications, it is the ultimate goal to incorporate the transformer zoning program into a bigger software system. Therefore, it would be strategically useful in having a single class that can be instatiated by an external system which in turn will execute all the sequences, described above in their respective class categories, to complete the transformer zoning process. Thus a single class `Control` was defined to conduct all the control operations, as implied by the name.

The `Control` must be easily modifiable so that certain processes can be omitted if required. For example, the further refinement process may be omitted if is found to be too time consuming and making negligible improvement to the solution.

It should be noted that the class architecture could have been accomplished without the class `Control`. But in doing so, it enabled the creation of a clear control mechanism especially if it was to form part of other software systems.

4.1.2.6.2 Class Structure

The class structure for `Control` is shown in Figure 14 along with classes from the other class categories with which direct communication occurs.

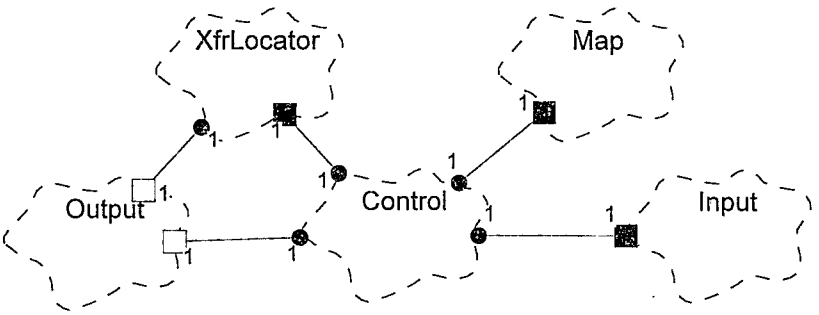


Figure 14 - Class Category Controlling Unit

4.1.2.6.3 Class responsibilities

`Control` regulates the entire transformer zoning process. In Figure 14 the inter-operation of the various classes is not clearly shown and this information can be more efficiently conveyed through the use of object diagrams as discussed in the subsequent section 4.1.3.

4.1.3 Mechanisms and Scenarios

A scenario for this system, that captures interactions at the highest level of abstraction, is "the transformer zoning process". This scenario is illustrated in Figure 15, and an explanation of the diagram follows.

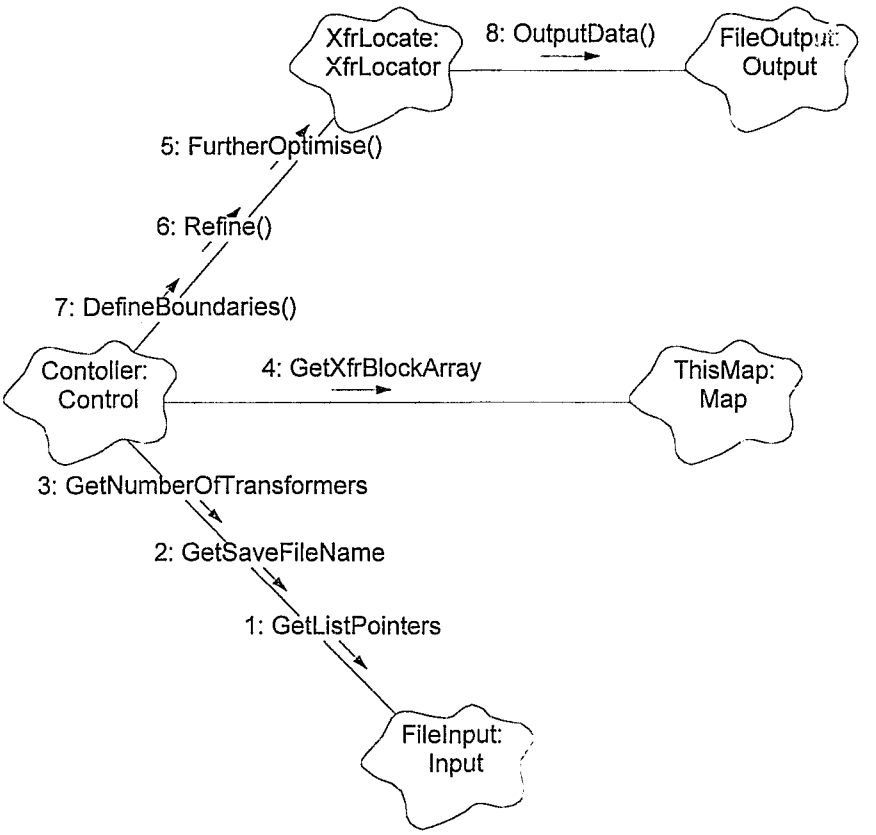


Figure 15 - High Level Object Diagram

An instance of `Control` (`Controller`) starts the transformer zoning process. `Controller` in turn instantiates `FileInput` which in turn acquires all the graphical and non graphical (user) data. Pointers to instances of the two linked lists `LStand` and the `Polygon` (see section 4.1.2.2) are returned to `Contoller`. In addition, other data such as the destination CAD filename and the number of transformers are also returned to the `Controller`. `Controller` passes the pointer to an instance of `Polygon` to `ThisMap` (instance of `Map`). `ThisMap` returns an array of `XfrBlock`'s to `Controller` after completing the square extraction technique. `Controller` then passes the pointers to `XfrBlock`'s along with the list of `LStand` and a pointer to an instance of `Output` (`FileOutput`) to `XfrLocate`. `XfrLocate` (instance of `XfrLocator`) automatically completes Grimsdale's algorithm. `Controller` then instructs `XfrLocate` to *FurtherOptimise* (Sweeping algorithm), to *Refine* (Refinement algorithm) and *DefineBoundaries* (drawing of polygon boundaries). The last operation results in the passing of the pointer pointing to `FileOutput` to an instance of `XfrBoundary` (not shown in the diagram). This instance in turn will pass the relevant co-ordinates to `FileOutput` to construct the transformer zone delineation.

Object diagrams can be represented in another way and they are referred to as *interaction diagrams*. Despite the fact that information presented in both these types of diagrams are virtually identical, the diagrams are structured differently. An interaction diagram, being in a tabular form of an object diagram, is able to clearly state the timing sequence of the messages that are passed between objects. The interaction diagram for the above is shown in Figure 16.

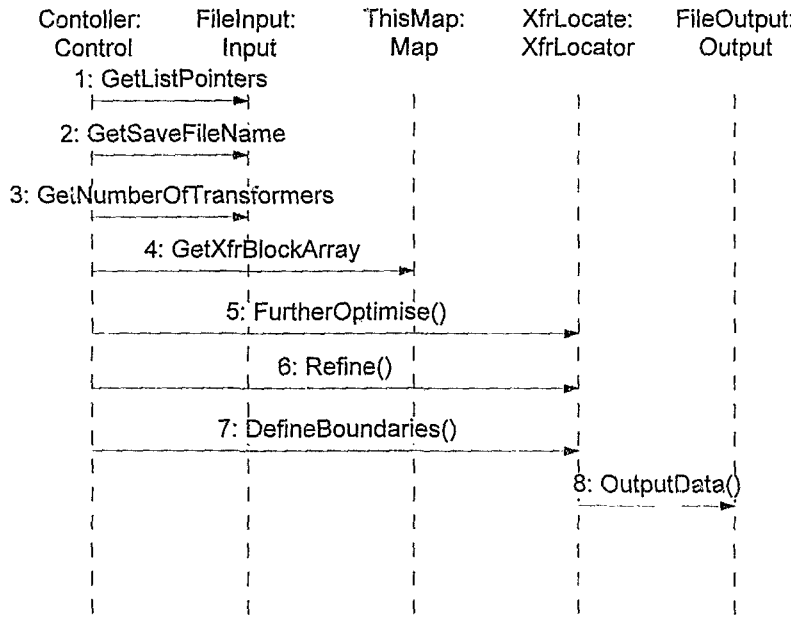


Figure 16 - Interaction Diagram

4.2 Physical Architecture

Before reading this section the reader must have a basic understanding of the concept of the physical architecture and the notation as described in section 3.3.3.

This section describes the concrete software composition of the system implementation: the allocation of classes and objects to modules, as well as the allocation of processes to processors. Module names are indicated through the use of a specific font, for example, LStand.

In accordance with Booch's convention [Booch, 1994:220], two files exist for every module or package with the same filename. All the declaration files all have the extension ".h", while the definition files have the extension ".cpp". However, there are some exceptions to this convention and they are described below.

- The class `OuputFile` is contained in the file "classes.h". This class was developed as part of a previous project (undertaken by the author) and is being utilised as an extra in this project mainly for testing and gathering results. This is yet another example of reusability
- The definitions of the Generic class `DLListElem` and `GenList` are contained in the file Dllist.hpp while the implementation is contained in Dllist.cpp.
- There are certain modules that only contain type definitions and/or preprocessor directives and do not contain classes. These modules are Comndata and Types which are purely associated with the generic classes.

The transformer zoning system presented here has been designed with a single processor in mind, and, thus, all the system's processes are sequentially allocated to the same processor. Details about the specific hardware/software platform chosen are given in the low level design.

It would appear from some of the module diagrams that there are certain redundant dependencies. For example, a diagram might indicate that module A depends on module B and module B depends on subsystem C, *and that module A also depends on subsystem C*. The last dependency appears to be redundant. The reason that it may have to be included is that B can be dependent on a different *module* in *subsystem* C to the *module* which A is dependent upon. In addition, dependencies on standard C/C++ libraries (for example the "maths" library) are not shown.

4.2.1.1 Top Level

Figure 17 offers a top-level view of the system's physical architecture. This diagram depicts the dependencies that exist between the various modules and subsystems that reside at the top level of the system. The main function also resides at this level. Note that the module Input and its dependencies constitute the *Input* subsystem. The dependencies for the modules Polygon and Lstands are described in section 4.2.1.2.

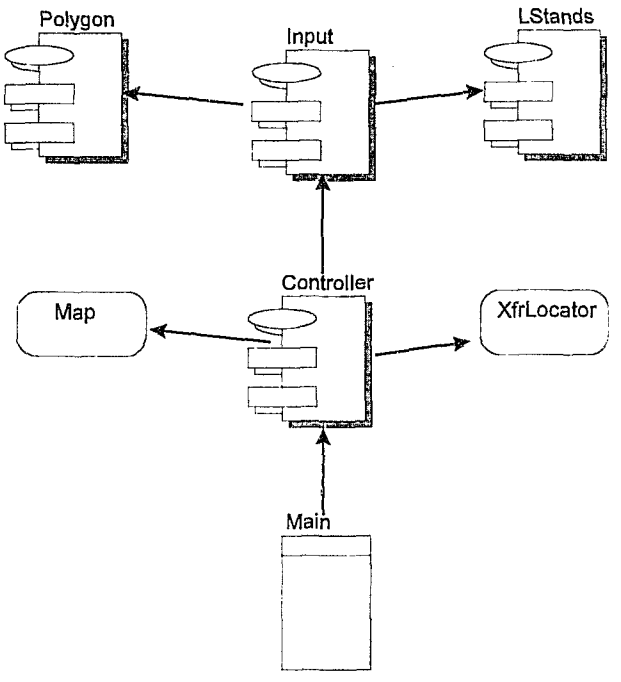


Figure 17 - Main Module Diagram

The reason for using the main function is to simulate the operation of this entire unit, or more specifically the class `Control`, being instantiated externally by another software system in which this unit will eventually be incorporated. XfrLocator subsystem has other dependencies such as the Output and XfrBoundary module. Therefore, by virtue of the physical structure, Controller also depends on XfrLocator's dependencies. The unique main function resides at the top of the hierarchy. The two subsystems shown in Figure 17 are described in the followir ɣ sections.

4.2.1.2 Generic Container Subsystem and dependencies

Both the modules Polygon and Lstands shown in Figure 17, depend on a generic package and other modules.

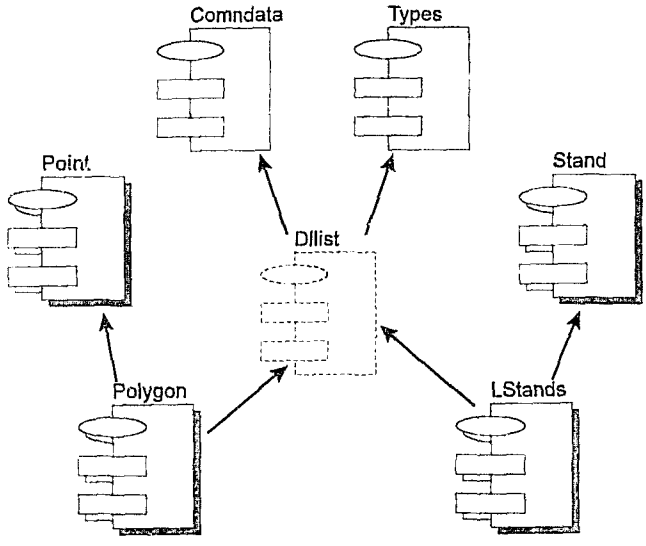


Figure 18 - Generic Containers and their relationships

Polygon depends on both the Dllist and Point Module so that the generic class can be instatiated with the class `Point` in order to create a linked list container. Likewise, Lstands also depends on both Dllist and Stand module so that the generic class can be instatiated with the class `Stand`. Clients needing to use this generic container need to include both the "Dllist.hpp" and "Dllist.cpp" files. Comndata contains type definitions specific to the generic packages while Types also contains some necessary definitions.

4.2.1.3 *Map* Subsystem

The *Map* subsystem is structured to implement the square extraction technique. The notable exception is that the *Xfrblock* subsystem is dependent on the *Output* subsystem (described in section 4.2.1.5). The reason for this dependency is so that the results of the square extraction technique can be output to the CAD file for visual evaluation.

It is important to note the similarities between the Map class category and this module. The similarity is due to the fact that the structure of the logical architecture can be expected to closely resemble that of the physical architecture.

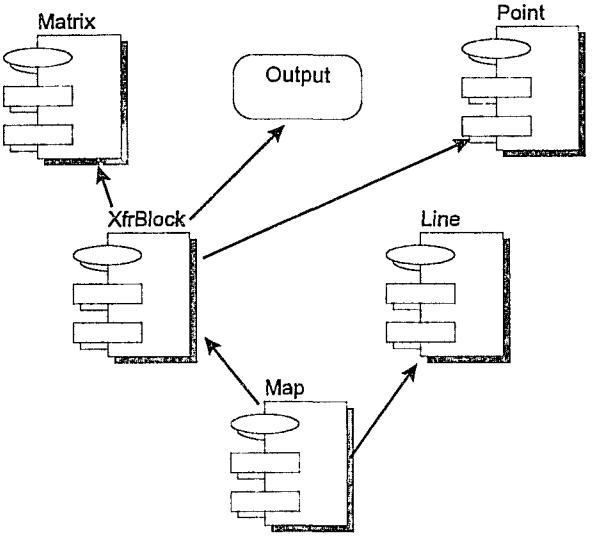


Figure 19 - Map Subsystem

4.2.1.4 *XfrLocator* Subsystem

This module is also straight forward and the similarities with the class categories can be noted.

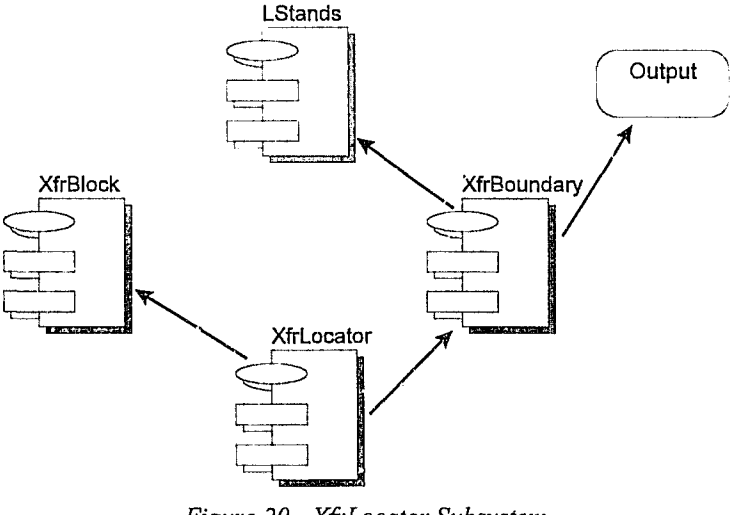


Figure 20 - XfrLocator Subsystem

4.2.1.5 *Output* Subsystem

The *Output* subsystem shown in Figure 21 has a dependency on polygon because there is a method in the Output Module that can define a polygon if an instance of `Polygon` is passed to it as an argument. This feature is not being used in the final product but was implemented originally with the intention of using it. Such a tactical choice, as discussed before, would make the `Output` class further dependent on the generic classes. The module Classes is being used to create text files, which is primarily being used for gathering additional data in the testing and evaluation phase.

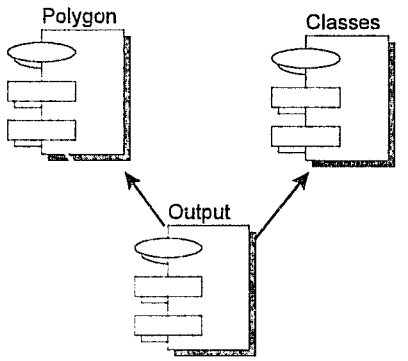


Figure 21 - Output Subsystem

Thus the complete physical architecture has been presented.

5 Conclusion

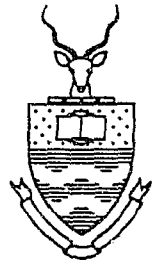
This document attempts to provide insight into the high level software design of the Transformer Zoning Program to be developed. The design methodology used has been described and justified. Class hierarchies of the system, the interaction and collaboration mechanisms between significant system objects and the physical system dependencies have been described in detail. Wherever appropriate, alternative designs have been discussed and choices justified.

The high level software design presented is object-oriented in that it employs some of the major techniques of object-oriented programming; namely, inheritance and encapsulation through the use of classes. Care has been taken in the design of each of class within each class category to ensure that each class has a distinct function with a clear division of responsibility. This minimises the likelihood of system wide changes occurring because any changes that need to take place has been encapsulated within a single class of that system. Consequently, the system is more flexible to change and less likely to be error prone in the evolution process.

When the need arises for this program to interface with or to be incorporated within a larger and more complex system, only the `Input` and `Output` classes need to be redesigned and implemented. In doing this, it is important to preserve the communication mechanisms so that it does not impact other components of the system that are dependent on it.

An object-oriented methodology allows for significant reuse. Reuse in this system is evident in the use of the generic container class.

The Low Level Design (B5) elaborates the design presented here in terms of the classes and its methods. Complex algorithms presented in the conceptual design (B3) are elaborated upon from an implementation point of view.



ASED

Low Level Software Design

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents ii

Change History v

Configuration Control v

Document History v

Revision History v

Management Authorisation v

Change Forecast v

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Definitions 1

1.4 Applicable Documents 1

1.5 Assumptions 1

2 Background 2

2.1 Notation 2

2.2 Common Tactical Policies 2

2.3 Class Declarations and Definitions 3

3 Generic containers 4

3.1 Module Comndata 4

3.2 Module Types 4

3.3 Generic Containers: Module Dllist 4

4 Utility classes 7

4.1 Class Point 7

4.2 Class Stand 7

4.3 Class Line 9

4.4 Class Matrix 10

4.5 Class Polygon 11

4.6 Class LStands 11

4.7 Class OutputFile (Module Classes.h) 11

5 Class Input 12

5.1 Member functions 12

5.2 Data members 13

6 Class Map 14

6.1 Member functions 14

6.2 Data members 16

7 Class XfrBlock 18

7.1 Member functions 18

7.2 Data members 20

8 Class Output 21

8.1 Member functions 21

8.2 Data members 23

9 Class XfrBoundary 24

10 Class XfrLocator 25

10.1 Member functions 25

10.2 Data members 29

11 Class Control 30

Change History

Configuration Control

Project:	SADDIN
Title:	Low Level Software Design
Doc. Reference:	E:\1996_34\TP\TB340.001.DOC
Created by:	T Rajakanthan
Creation Date:	13 September 1997

Document History

Version	Date	Status	Who	Saved as:
0.01	97\09\13	Draft	TR	\1996_35\TP\TB340.001
1.00	99\02\04	Approved	TR	\1996_35\TP\TB340.100

Revision History

Version	Date	Changes
0.01	97\09\13	New document created from TB002.100 (Document Creation Template)
0.02	98\09\04	Major changes and additions were made to this document.
0.03	98\0917	Editions were made to the entire document
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	Management Board Minute Reference
1.00	99/02/04	Approved	TB 533

Change Forecast

1 Scope

1.1 Introduction

The low level design deals specifically with the "nitty-gritty" implementation details such as the notation that is used, and the common tactical policies employed. The class declarations and definitions are described and details of certain algorithms are given. Wherever appropriate, algorithms are further explicated through the use of flowcharts.

This system was developed on an IBM compatible PC clone, using Microsoft Visual C++ compiler (ver 5.0) for the Windows 95 operating system. The transformer zoning program was designed to run in text mode as a 32-bit console application. In the development of this software, the Microsoft Foundation Classes (MFC) was deliberately not used. As a result, the system is not dependent on a particular operating system or compiler and is therefore portable.

1.2 Purpose

The purpose of this document is to present all the low level implementation details.

1.3 Definitions

1.4 Applicable Documents

1.4.1 Specifications

1.4.2 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.5 Assumptions

It is assumed that the reader of this document has a working knowledge of the C++ programming language. It also assumed that the reader has read the Conceptual Design and the High Level Software Design (documents B3 and B4 respectively). If this document is being evaluated with the intention of adding modifications to the program, he or she must have access to the source code of the project.

2 Background

2.1 Notation

This section is to inform the reader of the notation conventions applied to the coding of this project. These convention may not apply to source code that has been borrowed.

Mixed case is used for the naming of all classes and enumerated types. The first letter of all names is typically capital and may be of any length. The same principle applies to the objects, variables and class data members. All source code filenames are long filenames as supported by the Windows 95 operating system. Meaningful names have been used so that the code is user friendly.

Although long filenames in the Windows operating system are not case sensitive, they will be listed in the same format as used for naming the classes. The header files of Modules which contain template classes will have an "hpp" extension while the body has a "cpp" extension. The header files of modules which contain standard classes will have an "h" extension while the body has a "cpp" extension. The filenames of standard modules reflect the names of the classes that the contain.

2.2 Common Tactical Policies

Booch [1994:519] defines a tactical design decision as one that has local architectural implications. The common tactical policies that have been made during the design phase are discussed in this section.

This system designed has a single thread of execution control. The thread starts and terminates with the `Main` function which is only being used to instantiate the `control` object. As explained in the High level design `control` sequentially executes the entire transformer zoning processes.

Wherever possible memory allocation is optimally utilised with the use of linked lists. Memory allocation is done dynamically during run time using the C++ `new` and `delete` operators for linked lists and some of the variables. In this prototype, no verification techniques are being employed to determine whether the requested memory has been successfully allocated. It is important to be able to deal with failed memory allocation as it could result in the system "crashing". However, every effort has been made to ensure that memory leaks do not occur. This is accomplished by promptly destructing objects that are no longer required, which results in the memory allocated to that object being released. Furthermore, all other combinations of object termination are carefully considered to ensure that all pointers are deleted at the end of the process.

It is considered good practise to build in mechanisms to deal with failed memory allocations. It is the responsibility of an object's client to check for errors and act accordingly. This includes the destruction of an object if it could not be constructed successfully.

2.3 Class Declarations and Definitions

The subsequent sections give a class by class description of the source code. The reader should note that the source code of this project is well documented and fairly self-explanatory. Thus, this document describes overall algorithms for methods and does not give a detailed description of the source code. It is recommended that the this document be read in conjunction with the source code if one has to gain insight into the processes. However, it is not absolutely imperative to have access to the source code to obtain a general overview of the classes' functionality.

It should be noted that each of the algorithms, that have been described in the Conceptual Design or in this document, are not implemented by a single method. In order to simplify the complexity, a single algorithm has been accomplished through the use of several inter-operating methods. Wherever, appropriate, the user will be informed of the specific methods which interact to produce a combined effect that reflects a documented algorithm. However, in some cases, several algorithms may very well exist within a single method.

For completeness, every class in the system and its data members and member functions are listed but not every one of these elements is described. The descriptions of functions which are self evident from the descriptive titles or which have minimal functionality have been omitted, since the commented source offers adequate explanations.

3 Generic containers

This section contains the documentation for the generic containers which was obtained from the author [Levitt: 1998] who was responsible for their design and implementation. These template classes are documented for the sake of completeness and for the reason that they are highly reusable provided that their interface is understood.

3.1 **Module** Comndata

The following type definitions and compiler directives are contained in Comndata. The definitions in Comndata are used by classes which are part of the *Generic Containers* class category/subsystem (as described in the High Level Design.

`#define ALLOCATE_ERROR "Dynamic allocation error"` - used in conjunction with an error message string to indicate dynamic memory allocation errors.

`typedef char string80[81]` - a string of 80 characters, used in the declaration of an error message string.

3.2 **Module** Types

The following type definitions are contained in the module Types. The following type definitions simplify declarations of some built-in types.

```
typedef unsigned char  uchar
typedef unsigned short ushort
typedef unsigned long  ulong
```

3.3 Generic Containers: Module `Dllist`

3.3.1 Template Structure `DLListElem`

This template structure has the following data members:

`T element;` //the actual element contained by each node in the linked list.

`DLListElem<T> *next;` //a pointer to the next node in the list.

`DLListElem<T> *prev;` //a pointer to the previous node in the list.

3.3.2 Template Class GenList

3.3.2.1 Data Members

Protected:

```
ushort listsize;            //number of nodes in the list.

DLListElem<T>* head;        //pointer to the head of the list.

int status;                 //list status.

DLListElem<T>* current;     //pointer to the current node in the list.
```

3.3.2.2 Member Functions

Public:

```
GenList();                  //This constructor is used for creating an empty generic doubly linked list.

GenList(GenList<T>& source);

GenList<T>& operator= (GenList<T>& source);

virtual ~GenList();

bool IsEmpty() const;       //This member function is used for checking if the linked list is empty. Returns TRUE if the list contains no nodes.

ushort GetListSize() const;

int GetStatus() const;

DLListElem<T>* GetCurrent() const; //This function is used to access the list nodes. Returns a pointer to the current node in the list.

virtual int Append();       // This operation appends a dummy element onto the end of the list. One is still required to initialise this element through accessing it directly. The list pointers and list size are maintained. Returns '0' if the operation was performed successfully or '-1' if there was memory allocation error.

virtual int AppendElement(T& new_element); //A reference to the element can be passed and it would be appended to the list. This operation appends the referenced element onto the list. The list pointers
```

and list size are maintained. Returns '0' if the operation was performed successfully or '-1' if there was a memory allocation error.

`virtual int RemoveCurrent();` //This functions deletes the node pointed to by the `current` pointer and points `current` to the next node in the list. If the last node is removed `current` is set to point to NULL. The list pointers and the list size are maintained irrespective of the node removed. Returns '0' if the operation was performed successfully or '-1' if the current pointer was not pointing to an element. This may be due to the fact that the list is empty *or* that `current` is pointing to the end of the list.

`virtual void GoToStart();` //This function points current to the first node in the list.

`virtual int NextElement();` //This function advances the `current` pointer to the next node in the list. Returns '0' if the operation was performed successfully or '-1' if the `current` pointer was pointing to NULL (which implies that the operation could not be performed).

`virtual int PrevElement();` //This function sets the `current` pointer to point to the previous node in the list. Returns '0' if the operation was performed successfully or '-1' if the `current` pointer was pointing to NULL (which implies that the operation could not be performed).

`virtual T& GetElement();` //This function is used for directly accessing elements in the list. Clients of this function must be sure that the `current` pointer is not pointing to NULL. Returns a reference to the *element* contained by the node currently being pointed to.

4 Utility classes

The classes listed in this section are necessary to assist other classes in the system but do not perform any distinct tasks of their own. These classes are not complicated and are listed here, since many of the other classes listed in subsequent sections utilise these classes extensively.

4.1 Class `Point`

4.1.1 Member functions

Public:

```
void InsertValue(float x, float y);

Point(float x = 0, float y = 0);

const float GetY();

const float GetX();

virtual ~Point();
```

4.1.2 Data members

Private:

`float coord[2];` //an array of two floats is used to store the x and y co-ordinates.

4.2 Class `Stand`

This class is derived from `Point`.

4.2.1 Member functions

Public:

`void ResetProcessed();` //resets the variable "processed" - processed = FALSE

`bool IsProcessed();` //returns the status of the variable "processed"

`void Processed();` //sets the variable "processed" to TRUE

`void InsertValue(char * identifier);` //Allows a string to be added to the object.

`void MemoriseCurrentXfrAllocation();` //This method is used for simulation purposes whereby the current transformer assignment is noted and stored in the *memorise_xfr_allocation* variable.

`void Reset();` //Restores the previous transformer allocation and resets the "assigned" variable.

`bool IsAssigned();` //returns the status of the variable "assigned".

`void InsertValue(float x, float y, char *identifier);` //Can insert x, y co-ordinate and a string into a stand object.

`const unsigned GetXfrAssigned();` //Returns the transformer number that it has been assigned.

`void AssignToXfr(short xfr_num);` //Method to assign the transformer number.

`const char* GetString();` //returns the string that the object contains

`Stand(float x, float y, char *identifier);` //Constructor with all the parameters defining a stand.

`Stand();`

`virtual ~Stand();`

4.2.2 Data members

Private:

`bool processed;` //variable used to contain the status of whether this stand has been processed or not.

`short memorised_xfr_allocation;` //contains the transformer number.

`bool Assigned;` //variable used to contain the status of whether this stand has been assigned or not.

`short XfrAssigned;` //contains the transformer number to which this stand has been assigned.

`char string[20];` //holds the text string representing the stand label.

4.3 Class `Line`

4.3.1 Member functions

Public:

`void InsertValue(Point StartPnt, Point EndPnt);` //A line can be defined by passing two Point's to it as arguments

`Point GetEndPoint();` //returns the point with the greater x co-ordinate.

`Point GetStartPoint();` //returns the point with the greater x co-ordinate .

`void InsertValue(float x_start, float y_start, float x_end, float y_end);` //A line can be defined by purely passing in all the co-ordinates. The values are inserted ensuring that the starting x co-ordinate is greater than the y co-ordinate. This sorting is done as feature of the class and in no way assists or hampers the process involved in this system.

`Line(float x_start, float y_start, float x_end, float y_end);` //Constructor with the various parameters.

`const float Get_m();` //returns the calculated gradient

`const float GetY_Intercept();` //returns the calculated y_intercept

```
const float GetEndY();

const float GetEndX();

const float GetStartY();

const float GetStartX();

Line();

virtual ~Line();
```

4.3.2 Data members

private:

```
float c;        //used for storing the calculated y-intercept
float m;        //used for storing the calculated gradient
Point end;      //used for storing the endpoint co-ordinates
Point start;    //used for storing the start point co-ordinates
```

4.4 Class `Matrix`

4.4.1 Member functions

Public:

```
unsigned GetColumnSize();
```
//returns the number of columns in the matrix.

```
unsigned GetRowSize();
```
//returns the number of rows in the matrix.

```
short GetValue(unsigned RowNum, unsigned ColNum);
```
//returns the value located at a specific point in the matrix. A '-2' is returned if the location of the requested element exceed the matrix dimensions. The values typically expected in this operation are either a '1' or a '0'.

```
void InsertValue(unsigned RowNum, unsigned ColNum,
short Value);
```
//Inserts the `Value` at a specific location in the matrix. There are checking mechanism to ensure that the element being placed does not exceed the matrix dimensions.

```
Matrix(unsigned MaxRows, unsigned MaxCols);
```
//Constructs a matrix of the size specified.

```
virtual ~Matrix();
```

4.4.2 Data members

Private:

```
unsigned ColumnSize;

unsigned RowSize;

short* Rows;
```
//a single dimensioned array that will be used to store the matrix elements. It is not possible to dynamically create a two dimensional array and therefor a single dimensional array must be logically segmented to represent one.

4.5 Class `Polygon`

Class `Polygon` is simply a derivation of `GenList<Point>` and contains no additional functionality. A linked list is used to contain the vertices that define a polygon.

4.6 Class `LStands`

Class `LStands` is a derivation of `GenList<Stand>` and contains no additional functionality. A linked list is used to contain all details of the stand in the township.

4.7 Class `OutputFile` (Module `Classes.h`)

4.7.1 Data members

Private:

```
FILE *stream;
```
//stream used for the output of information

```
char string[20];
```
//to temporarily contain a string prior to output.

4.7.2 Member functions

Public:

```
OutputFile();
```

```
Initialise(char *filename);
```
//Opens an output file with the specified filename.

```
output(char *text);
```
//used for writing text only and moves to the next line.

```
output(float x, float y);
```
//used for writing out two floats and moves to the next line.

```
output(int x);
```
//used for writing out a single integer and moves to the next line.

```
~OutputFile();
```
//writes "ENDFILE" and closes output stream and thus the file.

5 Class `Input`

5.1 Member functions

Public:

```
LStands* GetStandListPtr();
```
//returns a pointer to a list of stands

```
unsigned GetXfrNum();
```
//returns the number of transformers that are required to be placed

```
char* GetSaveFileName();
```
//returns a string representing the destination CAD DXF filename as entered by the user

```
char* GetOpenFileName();
```
//returns a string representing the source CSD DXF filename as entered by the user

```
Polygon* GetBoundaryPtr();
```
//used to return a pointer to the boundary polygon

```
Input();
```
//All the relevant information is read and associated lists are created in the constructor. It gets the file source CAD filename from the user, opens the file and reads the relevant information from it. Incomplete mechanism are incorporated to check whether the file exists and if it is in the right configuration. The flowchart for the text extraction algorithm is given in Figure 1.

The algorithm for identifying the vertices of the polygon defining the township boundary is listed below:

1. Start at the top of the file and search for the word ENTITIES.
2. Once the word in found look for the word POLYLINE.
3. Every time POLYLINE is found, identify the layer it is on. If it is on the layer TERRITORY then proceed to step 4, otherwise loop back to step 2.
4. Look for VERTEX.
5. Once VERTEX is found look for the number 10. Read the number on the consecutive line. This will be the *x* co-ordinate.
6. This will be immediately followed by the number 20 and then the *y* co-ordinate.
7. Pass the extracted co-ordinates of the vertex to the object.
8. Repeat steps 5 to 8 until the word SEQEND is encountered. This will imply that all the vertices have been identified.

These steps will be stopped when ENDSEC is encountered, which implies that the end of the file has been reached.

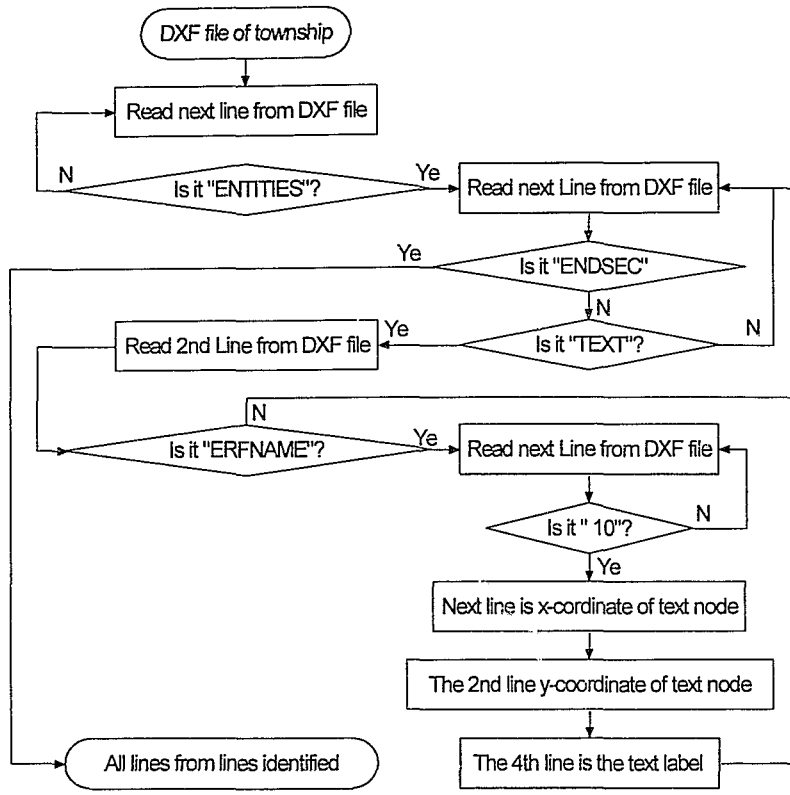


Figure 1 - Flowchart for extracting the stand labels from a DXF file

```
virtual ~Input();
```

5.2 Data members

It should be noted that all data members are contained by reference thus ensuring the continued existence even after the Destructor has been called (when `Input` goes out of scope). Clearly, this lists created are necessary and must be available for the most part of the transformer zoning process.

Private:

`LStands* StandList;` //is used to store the list of `Stands` read from the CAD file

`char * InputFilename;` //Name of the source CAD filename

`Polygon* Boundary;` //points to the `Polygon` containing the vertices of the boundary polygon

6 Class Map

6.1 Member functions

Public:

`XfrBlock *GetXfrArray();` //Returns a pointer to the array of `XfrBlock`'s.

`Map(Polygon *Boundary, Output *Outputfile, unsigned xfrs);` //Constructor with a pointer to the boundary polygon, output file and the number of transformers. The pointer to the output file will be used to output the transformer blocks during the testing phase. (The code has to be enabled to accomplish this.) All the steps to the square extraction process occurs in the constructor and some of the details are now given.

The extreme limits of the boundary are calculated so that a temporary artificial workspace boundary can be defined. The matrix representing the map is then constructed and the algorithm for this process is shown in

Figure 2. The scan conversion algorithm has been incorporated into this flowchart.

It should be noted that when the matrix has been created, a second copy of the original matrix is made. After an attempt, at placing the transformer blocks within the matrix, it will become unusable. Each instance of `XfrBlock` alters the appropriate elements within the matrix to represent the space that it is occupying. Thus, after all the instances of `XfrBlock` have placed themselves the matrix elements would have been modified. This matrix can then be restored from the copy before the next attempted placement. In order to better understand this process one must also examine the properties of the `XfrBlock` which is described in section 7.1.

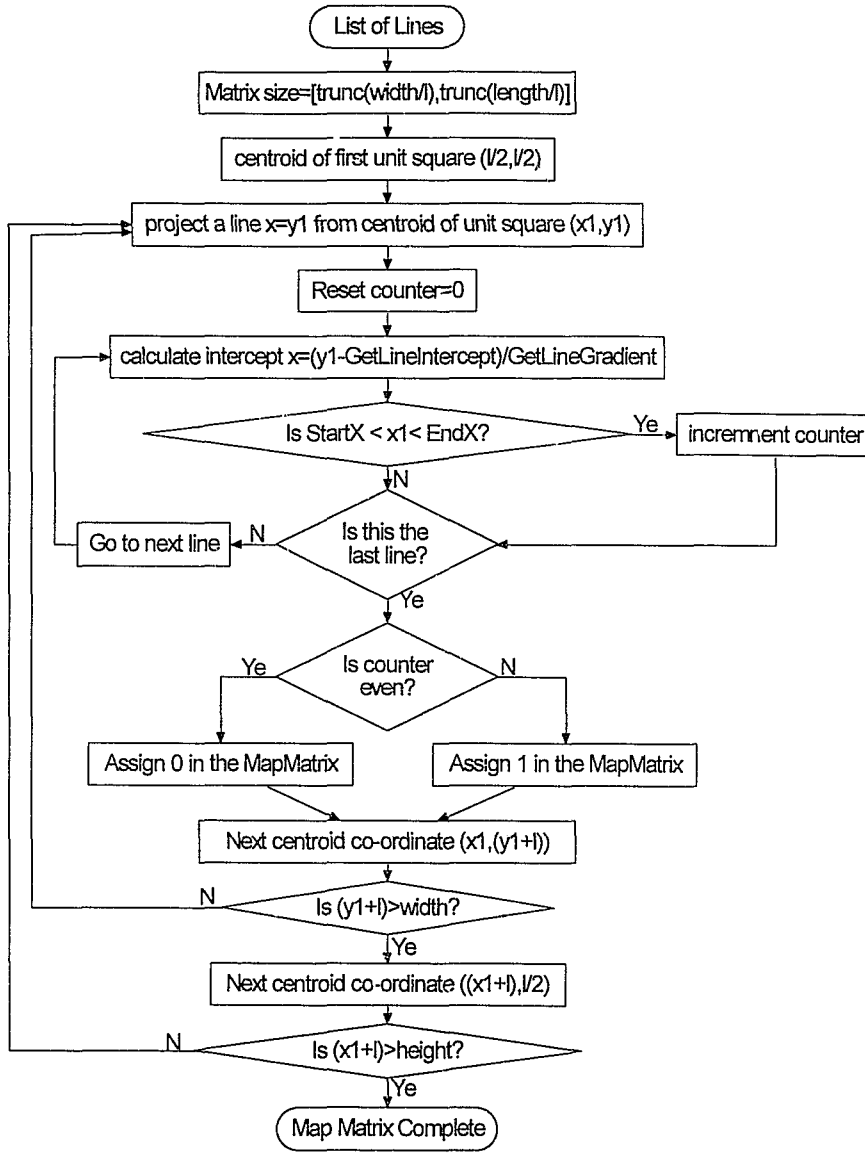


Figure 2 - Flowchart for creating the matrix representing the map

The transformer block placement algorithm is given in Figure 3 for the case where the side measures three unit squares. However, the same principle applies for placing transformer blocks of any size.

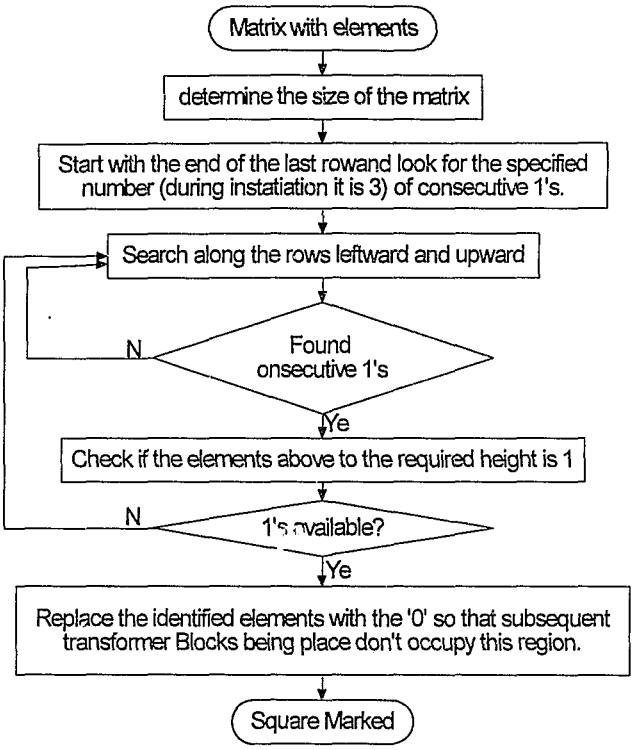


Figure 3 - Flowchart for the transformer block placement

The transformer block is passed various parameters such as the SideLength of the transformer block and the bottom left co-ordinate of rectangle. This will allow each transformer block to automatically determine the co-ordinates of its centroid, which is the location of the transformer. See section 7.1 for more details on calculating the centroid.

```
virtual ~Map();
```

6.2 Data members

Private:

```
XfrBlock * Blocks;
```
//to be set up into an array to contain the co-ordinates of the XfrBlock's

```
unsigned SideLength;
```
//represents the number of squares forming the side of a transformer block

```
int length;
```
//this is the actaul length of a side of unit square.

```
bool ScanConvert(float x_ref, float y_ref);          //A
```
method that executes the scan conversion algorithm. TRUE is returned if the point is located within the polygon.

```
Line *LineList;                    //to be set up into an array to contain
```
the list of Line's

7 Class `XfrBlock`

7.1 Member functions

Public:

`void MemoriseCurrentPosition();` //Copies the current position of the transformer from the variable `XfrPos` to the variable `Memorised_XfrPos`.

`void ResetPosition();` //Restores the current position of the transformer to the variable `XfrPos` with the previous position stored in the variable `Memorised_XfrPos`.

`unsigned GetXfrNum();` //returns the unique identifier assigned to the transformer which is stored in `BlockNum`.

`void ShowXfrPosition(Output *Outputfile);` //Outputs a "+" at the location where the transformer is currently situated. Using the location of the transformer as the basis, it adds and subtracts 20m horizontally and vertically to calculate four new co-ordinates. These co-ordinates are used to define two lines (+) to represent the location of the transformer.

`void SetXfrPosition(float x, float y);` //Defines the new location for the transformer.

`Point GetXfrPosition();` //Returns a copy of the object `XfrPos` which is the current location of the transformer.

`void Initialise(unsigned Num, Matrix *Map);` //Performs the exact same task as the constructor (with the same parameters).

`XfrBlock();`

`void ShowBlock(float x_ref, float y_ref, unsigned SideLength, Output *OutputFile);` //By passing in the bottom left corner of the rectangular workspace boundary, the actual side length of the unit square in metres and a pointer to the output file, it can determine the co-ordinates of the square defining the transformer block. That rectangle is then written to the output file. The following section calculates the relative position of the transformer block based on the information (stated above) that is provided. This method has to be executed in order for its relative position to be determined. That's why the code for actually displaying the transformer block has been deliberately commented out within this method. The block itself was displayed for testing reasons.

`bool PlaceBlock(unsigned side =3);` //Places a transformer block of the size specified by the `side`. If no `side` is specified a default of 3 is used.

`XfrBlock(unsigned Num, Matrix *Map);` //The block is constructed with the assignment of a unique identifier for the transformer and a pointer to the Matrix representing the map of the township. Since all instances of `XfrBlock` manipulate a single instance of `Matrix`, it has to have a pointer to it.

`virtual ~XfrBlock();`

The next six methods listed below are available to assist external management control and do not affect any of the internal processes. In other words, by setting variables clients can determine whether a particular object has been processed or not.

`void DeSelectCandidate();` //sets the `candidate` variable to FALSE.

`void SelectCandidate();` //sets the `candidate` variable to TRUE.

`bool IsCandidate();` //returns the status of the `candidate` variable.

`void Reset();` //sets the status of `processed` back to FALSE to imply that this block has not processed.

`void Processed();` //sets the status of `processed` to TRUE to imply that this block has been processed.

`bool IsProcessed();` //returns the status of `processed`.

Private:

`bool MarkBlock(unsigned row, unsigned col);` //Replaces the elements in the `Matrix` with a `row` x `col` sub-matrix containing '0's starting from the lower right corner. At present, this function just returns TRUE but provision has been made so that additional functionality for error checking can be added later if necessary.

`bool CheckBlock(unsigned row, unsigned col);` //Checks for consecutive elements in the matrix that are '1' starting from the lower right corner. TRUE is returned if the space is available.

7.2 Data members

Private:

`Point Memorised_XfrPos;` //variable used to store the location of the transformer whenever instructed by the method described above.

`bool candidate;` //This is used during the sweeping algorithm to make an `XfrBlock` (representing the transformer) either available for transfer (TRUE) or not (FALSE). Does not affect any processes internally.

`Point XfrPos;` //contains an up to date position of the transformer at any one time.

`unsigned BlockCol;` //This and the next variable are used to keep track of the lower right corner of the transformer block. This information, as stated earlier, is necessary to determine the actual co-ordinates of the transformer location.

`unsigned BlockRow;`

`unsigned BlockNum;` //contains the unique number representing the transformer number.

`unsigned BlockSide;` //represents the number of unit squares that make up the side of this transformer block.

`unsigned ColSize;` //represents the size (number of columns) of the matrix representing the map of the township.

`unsigned RowSize;` //represents the size (number of rows) of the matrix representing the map of the township.

`Matrix* MapMatrix;` //Used to point to the `Matrix`, which are accessed by all instances of `XfrBlock`'s.

`bool processed;` //Contains the status as to whether this `XfrBlock` has been processed or not.

8 Class `Output`

This class is a DXF file creating engine. Provided that its got a DXF file to read, it will make a copy of it and append specified drawing information. It should be noted that, if any modifications or extensions have to be made directly to the DXF file creation process, then it is highly recommended that one obtains an AutoCAD© 12 or better version manual. The information described in this section is sufficient for one to be able to use this engine easily.

Before proceeding it is important to understand the fundamental basis for the operation of this class; it uses a existing DXF CAD file to create a copy of it. In this duplication process, additional basic feature entities such as text lines and polygons are appended. Thus when viewing the new DXF file one would be able to see the original features and the new features that have been defined.

For management purposes, all the appended information are added to a new layer within the CAD file. In order to facilitate this process, additional functionality has been provided.

Some may consider this class restrictive in its present version. This refers to the ability to be able to define only one additional layer and writing to that layer. That view is correct as it was designed to solve the original specifications and it does so satisfactorily. It should be noted, that with relatively minor modifications the capability can be significantly extended, so that multiple layer can be created and features can be added to any layer. The ability to define various properties of the features, such as thickness and colour will add another dimension of flexibility to this engine.

8.1 Member functions

Public:

`void AddText(float x, float y, int num, float cab);` //Adds text (`num`) to the drawing at the specified co-ordintes. For testing purposes `cab` is also being passed as an argument but it is being written to another text file.

`void AddLine(float StartX, float StartY, float EndX, float EndY);` //Adds a line with the specified co-ordinates of the endpoints.

`void StartNewPolygon();` //The DXF file uses header information for every entity that it creates. Since this particular entity (polygon) can have any number of elements (vertices), the header and

footer information must be created separately to actually defining every vertex. This is unlike adding a line or text, which are singular entities, and therefore the header and footer information can (in some cases) are added every time this type of entity is defined. However, when defining a polygon, the `StartNewPolygon` method must be used to add header information before any of the vertices are passes to it using the `AddVertex` method. The `EndSequence` method must be used after the definition of the polygon so that the footer information can be added.

`void AddPolygon(Polygon *polyline);` //This method can define a complete polygon entity if a `Polygon` object is passed to it as an argument. Header and footer information are automatically added by calling the appropriate methods.

`void AddVertex(Point vertex);` //Adds a vertex to the polygon if an object Point is passed to it. It must be noted that the `StartNewPolygon` method must be invoked first.

`void AddVertex(float x, float y);` //As above but co-ordinates are passed in as floats

`void EndSequence();` //As explained above, after a series of vertices defining a polygon have been passed, then this method must be called to add footer information so that polygon entity is properly terminated.

`Output(char *OpenFileName, char *SaveFileName);` //Constructor requires the source and destination DXF filenames. It opens the source DXF CAD file and makes a identical copy of the contents into the destination DXF file. In order to add a new layer to an existing DXF file, the layer registry must be updated, which is listed before the entities section. The layer registry contains the list of layer names that are being used in that particular drawing file. This copying process is halted when the end of the layer registry is reached. Thus, the `SetLayerName` method must be evoked next before any other functions are executed.

`void SetLayerName(char* LayerName);` //Used to set the layer name on which the new drawing information will be appended. Unfortunately, the layer name can only be defined once and cannot be changed later. This implies that the information can be written to only one layer. No provision has been made at present to be able to write to existing layers.

Once the layer name has been defined, the process of copying the contents of the source file to the destination file will continue until the Entities section is reached. At this point this class is ready to receive feature entity information from clients.

`virtual ~Output();` //The destructor then continues to copy the rest of the entities section, now that all the necessary drawing elements have been defined, until the end of the file is reached. It then closes both files.

8.2 Data members

Private:

`char* string;` //Used to temporarily store character strings prior to output.

`FILE* Stream;` //Used to stream the output information to a file.

`OutputFile TextStream;` //Used to output the co-ordinates of every transformer zone to a text file.

`OutputFile Stats;` //Used to output additional text files of data. This is being used for gathering test data.

`ifstream *inf;` //stream used for reading in a file

9 Class `XfrBoundary`

This class has only public member functions and they are listed below.

`void Define(unsigned xfrs, LStands *StandList, Output *Outputfile, XfrBlock *Blocks);` Given the various parameters the boundary polygon is determined. `xfrs` is used to assign a unique identified to a particular instance of `XfrBoundary` in the same way as for `XfrBlock`. Thus, using `xfrs` those customers in the `StandList` who are assigned to this transformer can be identified. The algorithm calculates the vertices of the transformer zone boundary and sends it to the output file as each vertex is calculated. The drawback to this dynamic polygon defining technique, is that it is not possible to access and modify a specific vertex at a later stage.

The algorithm for defining the boundary is straight forward as described in the Conceptual Design (document B3). However an additional mechanism is in place to ensure that two coincident stand points (caused by human error or otherwise) will not be detected and defined as a polygon boundary more than once.

The TCL for each transformer zone is calculated in this method and output to the file together with the transformer number. Offsets with respect to the transformer location are specified so that the numbers do not appear over each other in the DXF file.

This methods contains extra code that can be used to draw straight lines from every consumer to the transformer that it has been assigned to. This feature was used when debugging the various consumer assignment algorithms. This section of the code has been disabled.

```
XfrBoundary();

virtual ~XfrBoundary();
```

10 Class `XfrLocator`

10.1 Member functions

Public:

`XfrLocator(LStands *Stand_List, Output *OutputFile, XfrBlock *Block_List, unsigned XfrNum);` Constructor with the various parameters. A pointer to `OutputFile` is passed so that it in turn can pass it to `XfrBoundary` which will use it when defining the transformer zone polygon. A pointer to `StandList`, the array of blocks and `XfrNum` are passed as arguments. `XfrNum` is necessary to inform the program of the number of `XfrBlock`'s in the array as it cannot be determined from any other means.

The constructor automatically performs the consumer allocation process in conformance to the second part of Grimsdale's algorithm. To assist the process of consumer allocation and location of new transformer positions, methods `StandAllocate` and `XfrCentroidLocate` are respectively employed.

```
virtual ~XfrLocator();
```

`void FurtherOptimise();` //Performs the scanning algorithm. It displays to the user the total number of stands, average zone limit in the township and the initial TCL. The simulation sweeps and the actual sweeps are controlled within this method along with the assistance of numerous private methods listed below. Thus the display includes results of the calculations, both from the simulation and actual sweeps. The preferred directions for all the sweeps are prompted for and obtained from the user. The refinement process is also evoked from here after every actual sweep that is performed (not after a simulation sweep). At the end of the process, the final increase in TCL is displayed.

There is a section of code within the main loop of this method which is incomplete and must be completed so that it can accomplish two things. Firstly, to automatically determine the best direction for every sweep based on the minimum number of overloaded consumers which have been determined after the simulation sweeps in all four directions. If there are more than one minimum, then the least increase in TCL percentage must be evaluated for those directions sharing that same numerical minimum. Furthermore, if there are more than one minimum in the TCL percentage then one of those directions must be chosen at random. Secondly, a suitable exit condition must also be defined, which must occur only when the number of overloaded consumers accumulated after the simulation sweeps is zero.

It should be noted that the interface which prompts the user for the direction of the subsequent sweep is case sensitive and all inputs must be in small letters. No checking mechanisms are in place and therefore entering an invalid character will result in the sweep occurring again but no damage to the process will occur. Furthermore, the ability to enter the direction was a feature used extensively during the testing phase, which will eventually be automated as explained above.

`void Refine();` //Performs the refinement algorithm. This method calls the private method `AttemptSwap` to assist in the refinement process.

`void DefineBoundaries();` //This method defines the transformer zones for all the consumer allocations. It delineates a polygonal boundary and passes the calculated vertices to `OutputFile`. See section 9 for more details on defining transformer zones.

Private:

The next two methods are used to complete Grimsdale's algorithm. They are called by the constructor.

`bool XferCentroidLocate();` //Computes the centroids of transformer zones. It also determines the distance between the previous and the new transformer positions. FALSE is returned if the distance moved by any of the transformers is greater than the specified limit (hard coded in the implementation - in metres). This return condition is used to control the loop in the constructor which executes the second part of Grimsdale's algorithm. The scanning algorithm also utilises this method but it is not concerned with the return value.

The distance migrated is calculated for each transformer and only if all of those transformers have migrated by less than the distance specified above, then `done` will return TRUE (return value).

`void StandAllocate();` //Assigns all consumers to transformers based on the shortest distance principle.

`bool AttemptSwap(int stand_num, int current_xfr, int best_xfr, float best_distance);` //This method works in conjunction with the `Refine` method. It attempts to perform a swap by passing the current "`stand`" that has been identified, the current transformer that it is assigned to and the transformer that it would be best assigned to. The distance from the stand to the new transformer is also passed. The algorithm in this method will determine the stand, currently

assigned to the "best" transformer, that can be swapped with the stand in question.

The following methods are present to assist the scanning algorithm

`void MemoriseLayout();` //Instructs the stands to note its current transformer assignment and transformers to note their current geographical position. This feature is used during the simulation sweeps.

`void RestoreLayout();` //Restores the previous stand to transformer assignment and the geographical location of the transformers. Used to undo the `MemoriseLayout` method.

`int CalculateMaxOverload();` //Calculates the total number of consumers in all the zones that are above the `Consumer_limit` (zone limit). It uses the `CalculateExcess` method listed below for each transformer and performs a summation.

`float CaculateTotalCableLength();` //Calculates the TCL.

`void ResetStandAllocation();` //Reset the transformer assignment of every stand to the previous. This method is used by `RestoreLayout` method.

`bool DownSweep();` //Used by the Sweeping algorithm when sweeping down. This algorithm identifies a list of suitable transformers as suitable candidates for transfer and then transfers them. Consumers are then reassigned using the `ConsumerTransfer` method listed below.

During the sweeping process, transformers located below the current transformer and those within a certain radius will be selected as candidates for transfer. The reason for having such a mechanism is to prevent transformers that are two far away from being selected as candidates for transfer. The algorithm only looks for candidates whenever an overloaded transformer is encountered. There is a factor which restricts the search for suitable transformers to within 1.2 (defined by `Search_radius_multiple` in the code) times the distance of the nearest transformer on the right hand side of the current transformer.

Consider the case where one of the overloaded transformer encountered is in a "compromising" location such as transformer A as shown in Figure 4. This diagram shows transformer zones within a township where the transformer locations are indicated with a '+' symbol. In this case the nearest transformer would be identified as C in conformance to the algorithm which identifies candidate transformers (assuming a sweep in the right direction). It can, however be clearly seen that the most suitable transformer for transfer is transformer B. With the refined algorithm to be

described, transformer zone A will remain so that it can be resolved by transferring in another direction. Also, due to the problem now described, this program will not permit more than one sweep in the same direction.

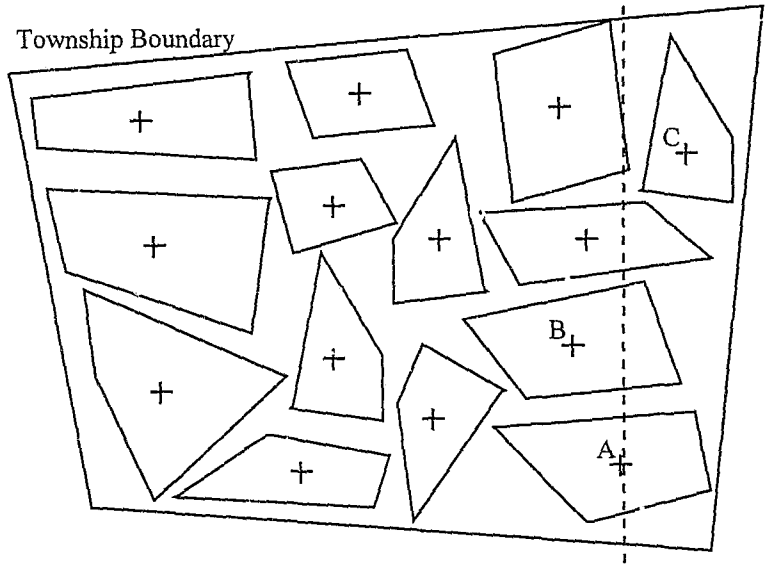


Figure 4 - Simulated transformer zones within a township

Thus the following procedures have been incorporated into the code to ensure that this bug does not occur (described for the case when sweeping to the right).

1. Find the first overloaded transformer.
2. Calculate, the nearest transformer on the right of this transformer and note the distance `distref`.
3. Set the maximum search radius for candidate transformers so that it does not exceed twice this limit under no circumstances. Thus, `nearest_xfr_distance_ref` is set to twice `distref`.

This process is done for the first overloaded transformer encountered and it is not repeated for subsequent transformers encounters. However, all the candidate transformers selected for transfer must be satisfy the following conditions:

- The transformer is less than `nearest_xfr_distance_ref`, and
- less than the product of `distref` and `Search_radius_multiple`.

It should be noted that the transformer is already located on the right hand side of the current transformer, by virtue of the selection process.

A more reliable but time consuming algorithm would have been to select all transformers located on the right hand side of the current transformer

as candidates. During the consumer transferring process only those candidates that are suitable will be assigned but this implies having to process greater number of transformers.

`bool UpSweep();` //As above but in the upward direction

`bool LeftSweep();` //sweeps to the left

`bool RightSweep();` //sweeps to the right

`void ConsumerTransfer(int xfr_num);` //Identifies the nearest consumer belonging to the specified transformer and transfers it to the nearest candidate transformer. It repeats this process until all the excess consumers for the specified transformer have been re-assigned. This is a universal method used by all "`Sweeping`" methods (described above) since it is independent of direction. This method only processes transformers (`XfrBlock`) that have been selected as candidates.

`bool IsXfrOverloaded(short xfr_num);` //Determines whether the specified transformer is overloaded or not. Returns TRUE if the transformer is overloaded.

`int CalculateExcess(short xfr_num);` //Calculates the excess number of consumers assigned to the specified transformer.

10.2 Data members

`float initial_cab_length;` //used to store the initial TCL for comparison purposes. This figure is used to compare the increase in TCL after sweeping or refinement.

`int Consumer_limit;` //contains the zone limit as specified by the user.

`Output* Outputfile;` //Used to point to the output file

`int ListSize;` //Used to hold the list size of `LStands` (object `StandList`)

`int xfrs;` //the number of transformers in the array of `XfrBlock's`.

`XfrBlock * Blocks;` //Pointer to the array of `XfrBlock's`

`LStands * StandList;` //Pointer to `LStands`

11 Class `Control`

This class only has the bare essential member functions; namely the constructor and destructor. It is an important class as it controls the sequence of events that have to be done to achieve transformer zoning. Thus, this class has been described in a bit more details so that one can not only understand the sequence of events but also the methods of the other classes which are being accessed directly.

```
control();          //All relevant operations are in the constructor.
```

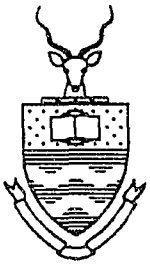
Since many of the variables are requested from some objects and passed to other objects, pointers are defined for storing those variables or pointers while in transit. Instantiating the class `Input` results in the source DXF file being opened and all relevant information being extracted. The information is stored in `StandsList` (list of stands) and `Boundary` (list of vertices). It obtains the source and destination CAD filenames from `Input` and instantiates `Output`. The destructor for `Input` is now called as it goes out of scope.

The layer name for the output file is defined here. This must be done prior to any feature being added to the output file using one of the methods. Class `Map` is then instantiated which automatically executes Grimsdale's square extraction algorithm. Instantiating `XfrLocator` results in Grimsdale's algorithm being completed. Calling the FurtherOptimise method executes the scanning algorithm. In its present configuration the refinement algorithm is executed after each successful sweep. The refinement algorithm is finally executed again after the scanning algorithm by evoking the `Refine` method directly.

Now that all the processing has been completed, the transformer zone boundaries can be defined. This is done by using the `DefineBoundaries` method.

```
virtual ~control();
```

Thus the controlling algorithm and all other aspects of the low level design has been described. It is further iterated that to completely understand a particular class or method one must study the Conceptual Design (B3), High Level Design (B4) and the commented source code. The commented source code is necessary if one has to be able to make changes or modifications to the program.



SADDIN

Testing And Results

Technical Product

Version 1.00

Document Status: Approved

Table of Contents

Table of Contents 2

Configuration Control 5

Document History 5

Revision History 5

Management Authorisation 5

Change Forecast 5

1 Scope 1

1.1 Introduction 1

1.2 Purpose 2

1.3 Definitions/Terminology 3

1.4 Audience 3

1.5 Applicable Documents 3

2 Townships used and their Statistics 4

3 Grimsdale's Algorithm 9

3.1 Square Extraction Technique 9

3.2 Consumer allocation algorithm 10

4 Scanning (Sweeping) Algorithm 13

4.1 Testing procedure 13

4.2 Preliminary findings 14

4.3 Properties of the scanning algorithm 15

5 Refinement Algorithm 25

5.1 Post refinement process 25

5.2 Current refinement process 30

6 Costing profiles 33

7 Complete sample design evaluation 41

7.1 Practical usefulness of this software 41

7.2 Cost equation comparison 42

8 Comparison between Grimsdale's and scanning algorithm 47

9 Evaluations 50

9.1 Grimsdale's algorithm 50

9.2 Scanning algorithm 50

9.3 Refinement algorithm 52

9.4 Costing issues 53

9.5 Complete sample designs 53

10 Conclusions 55

11 Recommendations 58

11.1 OLE automation 58

11.2 Interactive zone correction 58

11.3 Map inversion/rotation 59

11.4 Improved costing 59

11.5 Use of different transformers 60

11.6 Use of different sized transformers within a single execution 60

Change History

Configuration Control

Project:	SADDIN
Title:	Testing and Results
Doc. Reference:	\1996_34\TP\TB 345
Created by:	T Rajakanthan
Creation Date:	14 May 1998

Document History

Version	Date	Status	Who	Saved as:
0.01	97\03\01	Draft	TR	\1995_10\TP\TB345.001
1.00	99\02\04	Approved	TR	\1995_10\TP\TB345.100

Revision History

Version	Date	Changes
0.01	98\05\14	New document created from TB002.100 (Document Creation Template)
0.02	98\09\19	This document was completely revised making changes throughout the document.
0.03	98\10\25	Further changes were made prior to submission of first draft.
1.00	99\02\04	Approved

Management Authorisation

Version	Date	Status	MSR Project Minute Reference
1.00	99/02/04	Approved	Approved

Change Forecast

Will be updated as necessary.

1 Scope

1.1 Introduction

This document discusses the testing methods used and the results obtained, in an effort to qualitatively measure the effectiveness of the transformer zoning program. As is the case with any system designed to solve a particular problem, it would typically have well defined behavioural characteristics both desired and undesired. Clearly, the system that models the desired behavioural characteristics as closely as possible while distancing itself form the undesired is ideal. Thus, the testing phase serves to identify both these characteristics so that the level of practical applicability can be determined. It is essential to have read the Conceptual Design before reading this document.

For testing, a sample of five real townships located in South Africa were used. All of these maps were provided by ESKOM (the national supplier of electricity in South Africa) in MicroStation™ DGN file format. MicroStation™ is a third party (CAD) software that is utilised and is capable of exporting and importing drawing files in DXF format. It is extensively used as a visualisation and analysis tool in evaluating the transformer zoning program.

The number of consumers in the township varied from 300 all the way up to 6000 consumers. The shapes of the townships also varied significantly. The testing was conducted in a step by step manner examining common conditions that could occur with each of the algorithms that are involved in the transformer zoning process. Wherever appropriate processing times are also discussed. The aspects of the program tested are outlined below:

1. Complete examination and investigation of the effectiveness of Grimsdale's algorithm. Grimsdale also suggests that performing the square extraction on the mirrored image of the township provides vastly different or possibly better results. This aspect was investigated to see whether it is possible improve the zone layout. Although, not suggested, the inversion suggestion was taken further by assessing the performance of the square extraction technique when the township is rotated in 90° angles. (Section 3)

2. The use of the scanning algorithm was then scrutinised as a means to improve the solution. The level of improvement to Grimsdale's algorithm was also evaluated. Effect of map rotation and/or inversion were also investigated an attempt to improve the results. (Section 4)

3. The further refinement process was analysed. There were two methodologies in terms of the usage of the refinement algorithm and they were both examined. (Section 5)

4. The proposed costing formula was used to identify the optimum number of transformers that would provide the cheapest configuration for all the townships. Graphs and townships are provided with a short discussion on the effectiveness of the transformer zoning program. A costing profile was generated for one map in all its inverted and/or rotated states. (Section 6)

5. Complete sample designs were produced on a selected township using the transformer zones generated. The purpose of this exercise was to attempt to prove the validity of the proposed equation using the data from actual designs. (Section 7)

There are few other general comments that should be noted. All the transformer zone boundaries shown are as they appear in MicroStation. Since most maps are large, only selected information has been presented as so that the map is not cluttered. Furthermore, the level of information that can be displayed is physically limited by the resolution of the printer available. Typically every consumer will appear as a dot on the map of the township. It should also be noted that the maps are not represented to any scale but have been sized to best fit the space available. This ensures that the maximum possible details of the map is revealed. Where it was considered necessary to explain certain aspects, close-ups of the townships are provided with increased level of detail.

It should also be noted that an extensive study was done on various different maps but only the most notable or significantly different samples are presented. Notable cases are where specific errors occurred leading to improved techniques for achieving solutions or identified exceptions to the norm. Samples are provided merely as an aid to visually support the documentation. For the majority of the cases, a full discussion can be made with reference to the total cable lengths (TCL described in detail in the Conceptual Design) and very few diagrams can be expected.

It should be kept in mid that wherever computation timing issues are discussed and figures are presented, they are based on an Intel® Pentium 200MHz/MMX IBM compatible PC. This machine was using the Microsoft™ Windows '95 operating system.

1.2 Purpose

The purpose of this document is to evaluate the system designed through various case studies.

1.3 Definitions/Terminology

σ - symbol for standard deviation

Zone limit - All of the transformer zones created within a township must contain a total number of consumers, equal to or less than zone limit for each zone.

Larger zone limit - a relative terminology used to imply cases where the zone limit is close to the maximum zone limit

Smaller zone limit - a relative terminology used to imply cases where the zone limit is close to the average zone limit

1.4 Audience

The project manager and developer, members of the SEAL management board, the external examiner and all other interested parties.

1.5 Applicable Documents

1.5.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.5.2 References

References and Bibliography (TB 140)

Conceptual Design (A3)

2 Townships used and their Statistics

A total of five townships were used in the detailed testing process. These townships varied significantly in size and shape and were carefully selected to represent the various types of townships in South Africa. In order for the reader to be able to easily follow the documentation the townships are going to be referred to by an alias. The number of consumers are determined precisely by the transformer zoning program. It should be noted that other maps were tested but the maps selected here are those for the purpose of demonstrating not only the best case scenarios but also the worst.

The first township is Kwa-ratsiepane which is circular to square in shape accommodating 1113 consumers. This township will be referred to as Township A and is approximately 2.7 km horizontally and 2.3 km vertically.

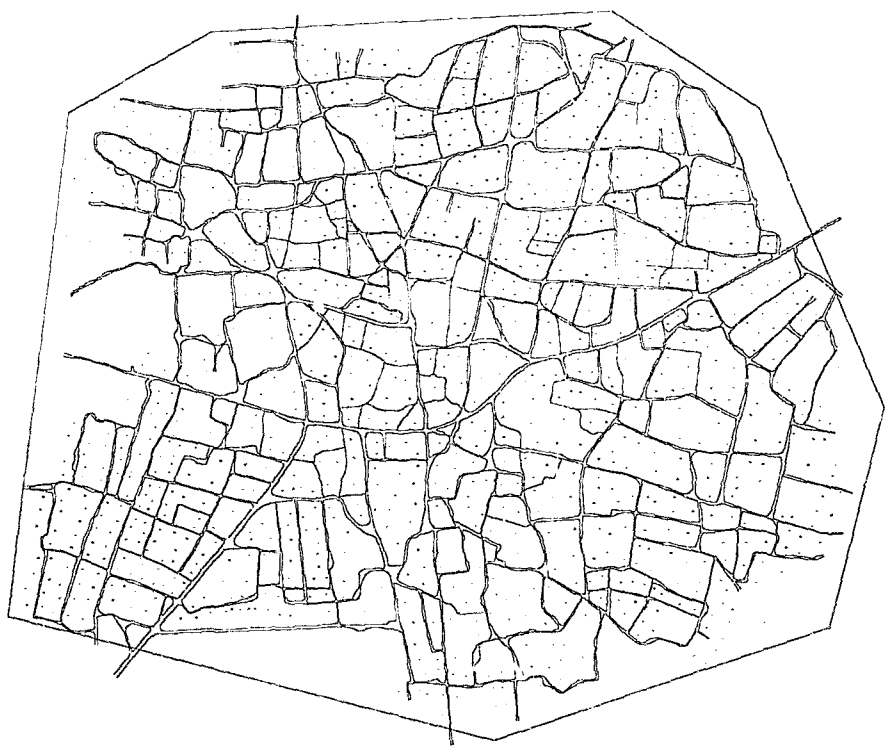


Figure 1 -Township A

The second township is Hebron which is classified as ribbon in shape and has 1770 consumers. Only the most important roads were explicitly drawn on the map. The many pathways that riddle the township can be easily deduced by looking at the stand boundaries (not shown). This township will be referred to as Township B and measures 4.5 km along its primary

axis (approximately 45° above horizontal clockwise) at a average width of 1.2 km.

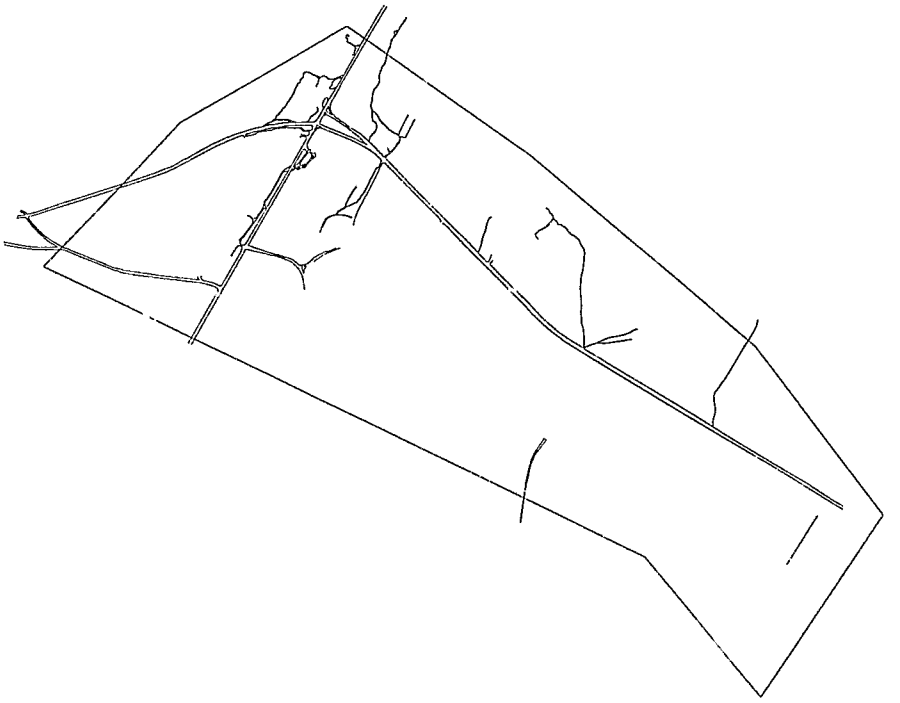


Figure 2 - Township B

The third township although not common in shape is unique and is thus included. This township has just over 300 consumers. It can seen in Figure 3 that at the Southern and Western locations consumers are scarcely populated while the areas on the East and North are relatively densely populated. This township will be referred to as Township C and is almost 2.5 km horizontally and 2.5 km vertically.

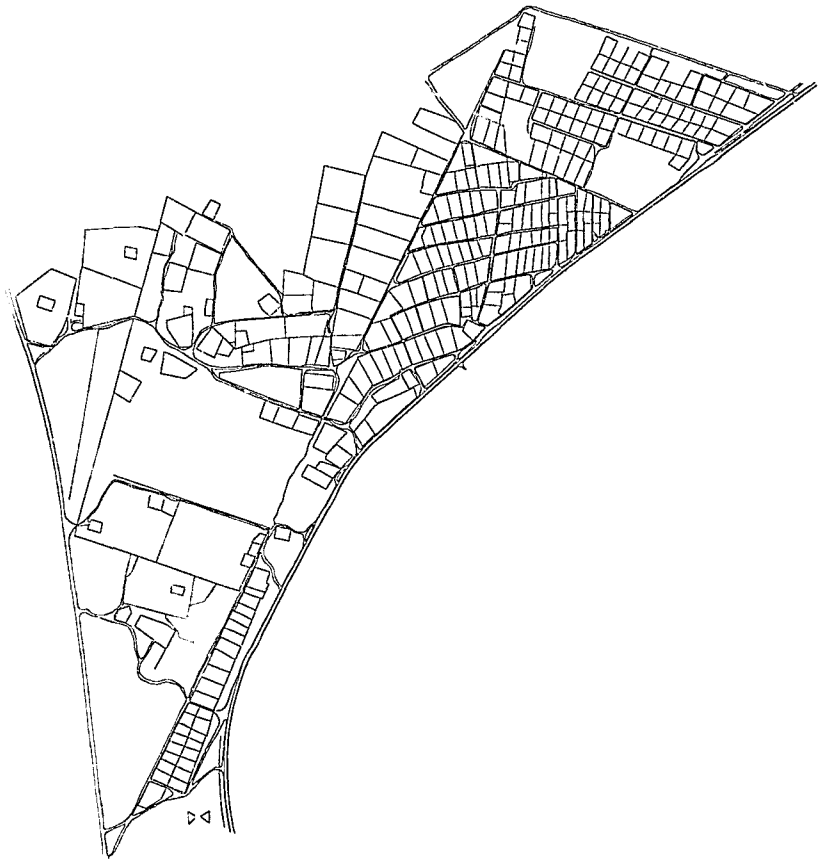


Figure 3 - Township C

The fourth township is also not common in shape but nevertheless unique as part of the township is divided by a marsh (represented by the shaded region in Figure 4) forming an obstacle. This township has almost 700 consumers. It can seen in that the there are certain areas that have no consumers such as the central and north-western region. These regions contain a park and a playing field respectively. This township will be referred to as Township D and is approximately 1.1 km horizontally and 750m vertically.

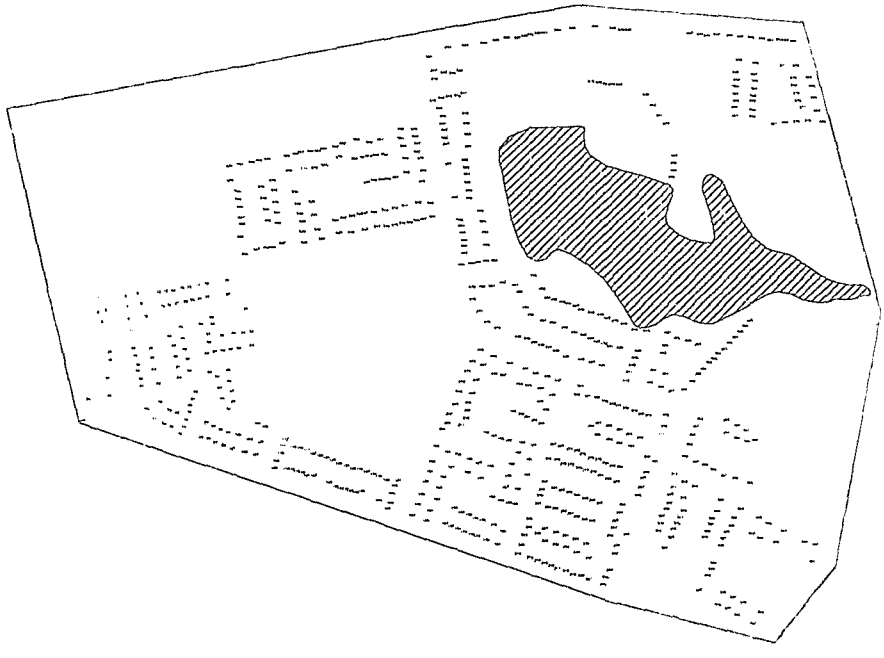


Figure 4 - Township D

The fifth township (as shown in Figure 5) not common but is included by virtue of fact that it is one of the larger townships currently being reticulated. The numerous pathways that run through the township are visible. This township has just under 5,900 consumers. Numerous roads are represented through the use of dashed lines but many of the smaller pathways have not been represented at all. It can be noted that the northern region has sparsely populated consumers. This township will be referred to as Township E and is approximately 3.5 km horizontally and 4.1 km vertically.



Figure 5 - Township E

3 Grimsdale's Algorithm

To simplify the evaluation, Grimsdale's algorithm has been separated into its two primary components and tested separately.

1. The square extraction technique is responsible for locating *n* transformers at positions within a township that are sufficiently far apart. The level of improvement in its performance, brought about by implementing the recommendations made by Grimsdale is also discussed.
2. The consumer allocation algorithm completes the entire process and thus the overall effectiveness is evaluated. In doing so the shortcomings of Grimsdale's algorithm are identified. In addition, the impact of the changes in the square extraction technique to the overall effectiveness of Grimsdale's algorithm is also discussed.

3.1 Square Extraction Technique

The square extraction technique was found to place the transformer blocks correctly in accordance with the algorithm. Grimsdale has suggested that inverting the map produces different if not better results. This concept has been taken a step further to consider the effects of rotating the map through 90° to see whether the results can be further improved. The measure of improvement using the square extraction technique is that if the map in one of its rotated or inverted states is able to accommodate larger squares than normal. Clearly then a better fit has been achieved. For testing purposes the maps were inverted and/or rotated externally using a third party CAD package prior to the square extraction process. Thus the results quoted below are a comparison of the size of the squares achieved on a map in a rotated and/or inverted state with reference to the original.

From the testing done, it was noted that the shape of the township determined whether the rotation or inversion would play a significant part in being able to place larger squares. Transformer locations were calculated and they were found to be notably different. However, based on the results it is impossible to predict the variation in transformer locations by simply examining the shape of the map.

Townships that are moderately symmetrical in shape such as Townships *A* and *E* were found to have an insignificant difference in the size of the squares being placed. For example the difference in the size of the squares was less than 5% for more than 50 transformers being placed in Township E. On the other hand, for asymmetrical townships such as Township B, it was found to vary as much as 10% with the placement of 15 transformers.

Naturally as the number of transformers being placed are increased the level of improvement becomes negligible or non-existent. As an example when considering Township E, the difference was as much as 13% for 20 transformers but reduces to zero as the number of transformers being placed exceeds 70. These results should not be taken at face value as they are merely, a ratio of the side of the square being placed to the other.

Thus, it can be concluded for this aspect of the testing that rotation and inversion of the maps allows the placement of bigger transformer blocks as part of the square extraction process. It was further observed that these effects became insignificant as the number of transformers being placed were increased. However, the consumer allocation algorithm (discussed in the following section) must be evaluated to determine whether improvement in the performance of the square extraction will lead to the overall improvement of Grimsdale's algorithm.

3.2 Consumer allocation algorithm

3.2.1 Evaluation

The speed at which the overall computation was done varied from township to township but in a maximum time of two minutes (for Township E) a possible solution could be achieved. The effect of rotation or inversion of the map (described in the previous section) was investigated by executing the complete Grimsdale's algorithm. The objective was to determine if it was possible to reduce the increase in TCL.

For smaller townships, it was found that as the number of transformers being placed was increased the improvement in TCL achieved was as much as 1.8% for 20 transformers for Township B. However, the improvement in TCL for 10 transformers was found to be significantly less (0.027%). Similarly for Township E, it was found that the improvement in TCL for 20 transformers was 1.14%, 50 was 0.44% and 80 was 2.47%. In general the improvement in TCL was significant as the number of transformers being placed were increased.

To conclude, it would appear that rotating and/or inverting the map improves the results by a noticeable amount. Unfortunately, there is no way of evaluating the best square fit for the maps in one of it rotated or inverted states when large number of transformers are being placed. The definition of "large number" implies more than 1 transformer per 0.23km^2 area. Thus, the Grimsdale's algorithm must be completely executed in order to determine improved transformer layouts.

3.2.2 Shortcomings

Grimsdale's algorithm was designed for use in urban areas where the consumer distribution is sufficiently even. This is not case in rural areas, which is obvious if one examines the consumer allocations in each zone as shown in Figure 6. This was evaluated by performing a statistical evaluation of the total cable lengths within all the individual transformer zones. For this purpose, Township E was used as it was a very large township. The average number of consumers per zone for 36 transformers was 164 but the zones varied from 105 to 229 (σ = 25). Likewise, the average number of consumers for 60 transformers was 98 but the zones varied from 53 to 160 (σ = 22.5).

However, a notable property was identified after extensive testing using this algorithm. Particularly in areas where there were obstacles or consumers grouped together, the algorithm tended to create transformer zones in a logical fashion. Suppose that an initial transformer position was located away from the main township near or amongst a cluster of several consumers. Then, the transformer zone will tend to form round those consumers regardless of the rest of the township.

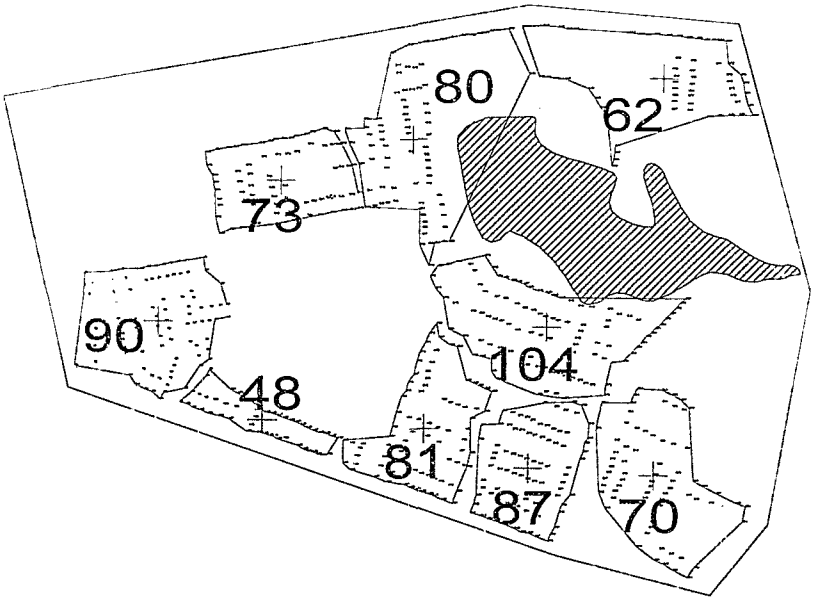


Figure 6 - Transformer zones generated by Grimsdale's algorithm

Figure 6 shown township D where the + sign indicates the ideal location for the transformer. For those zones defined, the number positioned next to the sign indicates the number of consumers located within the zone. It can be noted that the zones generated vary significantly in the number of consumers contained (min 48, max 104, avg 77, σ = 16.4). It can be noted

that the group of consumers located on the Northern side of the swamp have been grouped together. Similarly grouping patters can be noted with the consumers that are located on the Western half of the township.

4 Scanning (Sweeping) Algorithm

4.1 Testing procedure

The testing was done by calculating the minimum number of transformers for a particular township and then varying the zone limit form the maximum to the average. Suppose that 12 transformers are being placed in Township B, then the maximum zone limit is established as 17[illegible] (reasons for establishing this figure are irrelevant to proving the concepts at hand) and the average is 148. Thus 31 (178-148+1) executions of the program was done starting with the zone limit 178 all the way down to a zone limit of 148. This procedure will then be repeated with thirteen transformers, fourteen transformers and so on.

This test was done merely to evaluate the variation in TCL as the consumer limit was varied. It is important to note that wherever in the document an increase or decrease in TCL is discussed it is with respect to the absolute TCL measured immediately after Grimsdale's algorithm.

As explained in the conceptual design, attention was paid to evolving rules of thumb (heuristics) for determining the direction for the scanning algorithm. The program was designed in such a way that the computer will perform simulation sweeps in all four directions and present the user with the number of consumers piled up at the end of the sweep and the percentage increase in TCL. Using this information, various direction determining techniques were investigated to identify one that will converge at an acceptable solution in the least number of sweeps.

The direction determining technique that was used by default was assessing the number of consumers piled up at the end of a simulation sweep and assuming the direction with the least figure. In the cases where there were more than one direction with the same number of consumers piled up at the end of a simulation sweep, the direction with the least increase in TCL was selected. Other direction determining techniques investigated are discussed in subsequent sections.

The text window in Figure 7 shows a typical display as produced by the program with the relevant information. (Information about the display formatting has been discussed in the Conceptual Design - A3.) In the final scan it can be seen that sweeping in three directions gives the same number of consumers piling up at the end of the sweep but different increase in TCL. It can also be further noted that the percentage variation in TCL in the interim steps are sometimes greater than the final. This is due to the fact that the initial sweeping causes the zones to warp unreasonably, but as more sweeping occurs the zones become more refined. Note that a bold font is used to indicate the user input.

```
Initial total cable length = 459223
Right        Left       Up         Down
28           104        79         40
3.7967       0.395934   3.04687    5.55139
please enter direction of sweep: r

28           18         5          28
3.7967       5.35052    3.89134    3.7967
please enter direction of sweep: u

0            5          5          5
3.28117      3.80953    3.89134    4.23919
please enter direction of sweep: r

0            0          0          0
3.28117      3.28117    3.28117    3.28117
please enter direction of sweep: q

Percentage increase in total cable length = 3.28117%
```

Figure 7 - Sample of The Scanning Algorithm User Interaction

4.2 Preliminary findings

The scanning algorithm provided remarkable improvement to Grimsdale's algorithm. For example, when 36 transformers were placed in Township E, it was found that the minimum number of consumers per zone was 105, maximum was 167, average was 164 and σ = 11.6. However, the TCL increase was about 12% while the standard deviation almost halved. As expected, the standard deviation of the TCL, when considering individual zones, almost doubled. As far as Township D was concerned the minimum was 74, maximum was 78, average just over 77 and $\sigma \approx 1.6$. Township D is presented in Figure 8 with the transformer zone configuration that produced these results.

The number of sweeps required for the majority of the executions where the zone limit is high was usually a single or a double. However, as the zone limit approaches the average zone limit the number of sweeps required increases.

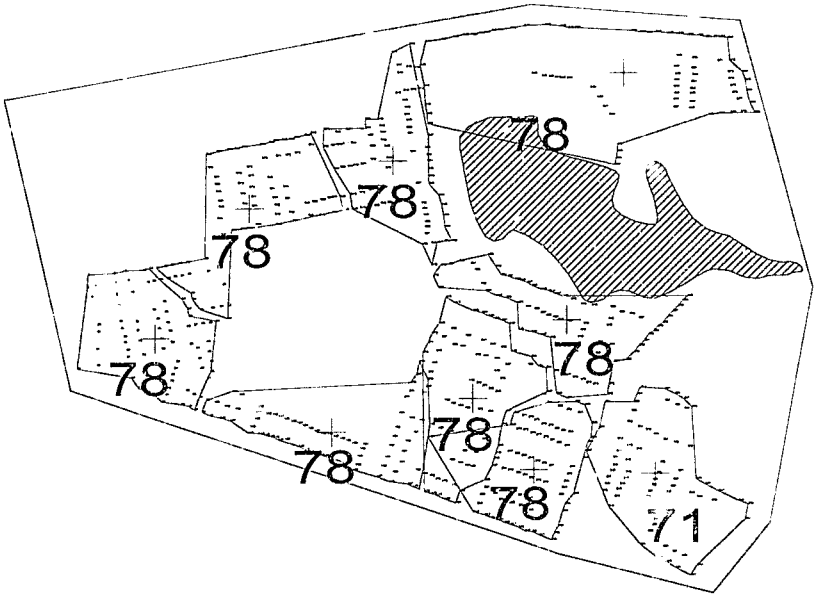


Figure 8 - Township D after the execution of the scanning algorithm

Although the allocations are even, some shortcomings are visible especially with the transformer zone located in the West of the township. It can be clearly seen that a consumer is being serviced across another zone. These shortcomings within this algorithm have been identified and they are discussed in this section.

4.3 Properties of the scanning algorithm

As stated earlier, the scanning algorithm technique though effective had several flaws and some interesting properties.

4.3.1 TCL versus zone limit

An unanticipated property of this algorithm was that the TCL did not increase linearly or in some predictable pattern as the zone limit was reduced from the maximum to the average. When Grimsdale's Algorithm was completed, the TCL was expected to be at a minimum, since this algorithm groups (see above) consumers logically. Thus, in order to distribute consumers evenly, some would have to be forced onto adjacent zones at a greater distance thereby increasing the TCL. However, in practise, often there is a general increase in TCL as the zone limit is reduced, but the pattern is unpredictable. It has also been found that the TCL can be less than 0% which suggests that Grimsdale's algorithm does not produce the minimum TCL. These properties are summarised by the

graph in Figure 9 for the placement of 12 and 13 transformers within Township B.

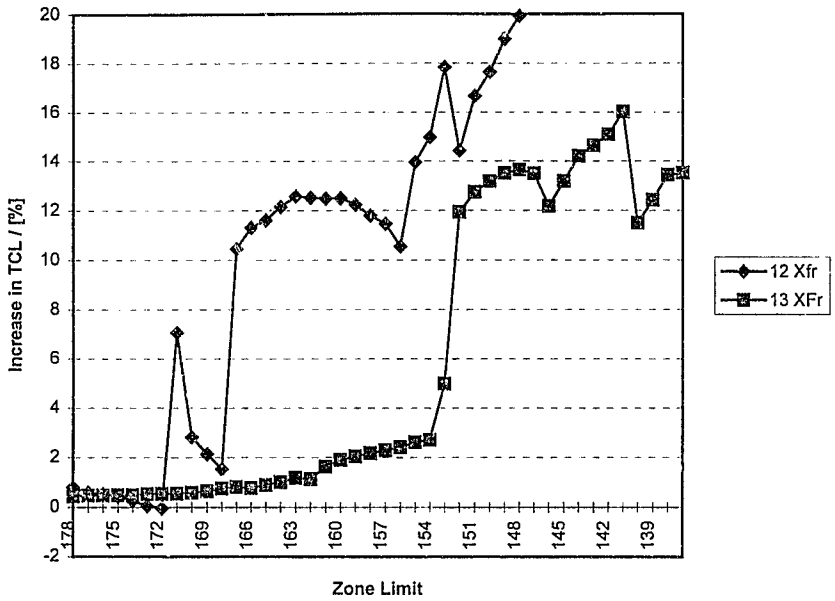


Figure 9 - Graph of Percentage Increase in TCL vs Zone Limit

The pattern represented by the graph shows some of the worst case scenarios and can only be described as erratic. The occurrence of this is due to the fact that the program is not as thorough in the reallocation process as when the zone limit is lowered. Stated simply, some consumers may have not been re-assigned until it was absolutely necessary to do so in order to achieve the specified zone limit. When the zone limit is lowered, it may result in the cascade process whereby a transformer reassign some of its consumers which results in that transformer having reassigning some of its consumer and so on. This process could result in the overall decline in the TCL. When the zone limit is high, sweeps in single directions are sufficient to generate a solution. As the zone limit is lowered, sweeps in multiple directions was required for this particular township. Inevitably, this results in the increase in TCL exceeding 10%.

4.3.2 Timing issues

The time taken for the operations were notable but not prohibitive. The computation time increases exponentially as the number of transformers being placed increases. Quantitatively stating, the time taken for the operations varied from seconds to 10 minutes, for each sweep of the

scanning algorithm (Township B and Township E respectively). This made testing every possible avenues impractical especially for larger townships. Consequently, careful samples were identified and tested so that the results would adequately represent the abilities of this algorithm. To this effect Township B was used for the testing as it was relatively large and had a sufficiently complex layout.

4.3.3 Problems with specific shaped townships

When the scanning algorithm is used on a ribbon shaped township such as Township B it was found that sweeping in a particular direction caused a problem. This is not a software bug but a flaw which was inherent in the evolution of this algorithm. It can be seen in Figure 10 that transformer zone D is severely deformed. It can also be further observed that the customers in the extended zone can be best served by transformers B and C.

The error is caused as result of the downward sweep. When transformer B is identified, the algorithm searches for candidate transformers that are located below this transformer. Although transformer C is more suitable than transformer D, the former is not recognised as its geographically located above the imaginary horizontal line[1] which now passes through transformer B. Thus, the algorithm forces the consumers of transformer B uneconomically on to transformer D. This problem does not exist with any of the other townships since all the transformers had a suitable transformers in the correct location that can take on excess consumers. The problem manifested itself due the direction of the primary axis of the township not coinciding with any of the sweeping directions. Thus the map of Township B was rotated by 45° in the anti-clockwise direction so that primary axis of the township was approximately horizontal. As anticipated, this error did not occur.

[1] An imaginary line is used to separate possible candidates from the entire list of transformers. If the algorithm is sweeping down, then transformers that are located below the imaginary line passing through that transformer are evaluated to determine whether they can be designated as candidates.

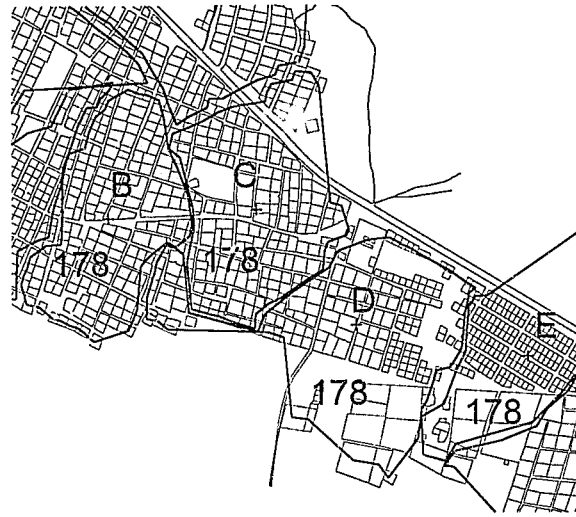


Figure 10 - Township with deformed transformer zone

4.3.4 Warping of transformer zones

A further undesirable property of the scanning algorithm was noted. After every sweep, new centres of gravity for the transformer locations are computed. As one would expect, this results in the transformers and their transformer zones migrating and changing shape. Unfortunately, sweeping in multiple directions often results in unacceptable warping of the transformer zones. Although, this effect is not obvious or pronounced in most cases, an extreme example is provided to demonstrate this effect, since clearly this is far from a suitable solution. The first image in Figure 11 shows the Southeast corner of Township B after Grimsdale's algorithm while the second image is after the scanning algorithm. Examining the transformer labels, one can trace how the transformers have migrated resulting in the warping of the respective zones. Another example of severe transformer zone distortion is shown in Figure 13.

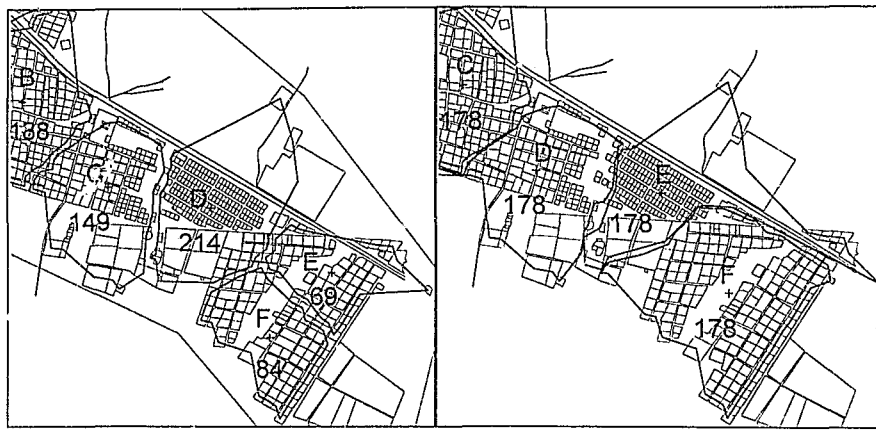


Figure 11 - Transformer zone deformation

A short description is provided on what causes this erroneous zone formation. When the first sweep occurs in the Southward direction, transformer D forces all the excess consumers it has accumulated from the other transformers to transformer E and F. Since transformer E is closer to transformer D, it takes most of the excess consumers. Then transformer E forces all its excess consumers onto transformer F. Predictably, the sweep downwards results in the extreme Southerly located transformer being overloaded. New centres of gravity are calculated for all transformer zones which results in the transformer positions migrating.

The subsequent sweep was in the upwards (Northerly) direction. Transformer F, now being overloaded, transfers its excess consumers to transformer E. Transformer E, being overloaded, transfers all the excess consumers to the nearest available transformer above its latitudinal location - which happens to be transformer D. Transformer D repeats the process to transformer C and so on. When re-assigning consumers from transformer F to E and from E to D the zones are extending in the upward direction. The consumer that is located on the extreme right in transformer zone E does not get transferred as it is clearly not near any of the other transformers. This results in the distortion of the transformer zone as shown in Figure 11. This inherent flaw in this algorithm was one of the reasons that prompted the development of the refinement process described in section 5.

However, in this particular case the latter two problems did not persist after the map has been rotated anti-clockwise so that the primary axis of the township was approximately horizontal. The reasons for this is twofold:

1. Transformer F is further to East than transformer E once the map has been rotated.

2. Sweeping occurred only along the primary axis eliminating the occurrence of the flaw described above. Sweeping along the primary axis implies that regardless of the number of consumers piled up at the ends after the simulation sweeps the direction chosen will be one of those along the primary axis.

These can be clearly seen in Figure 12 which shows the before[2] and after images of the same part of Township B after is has been rotated.

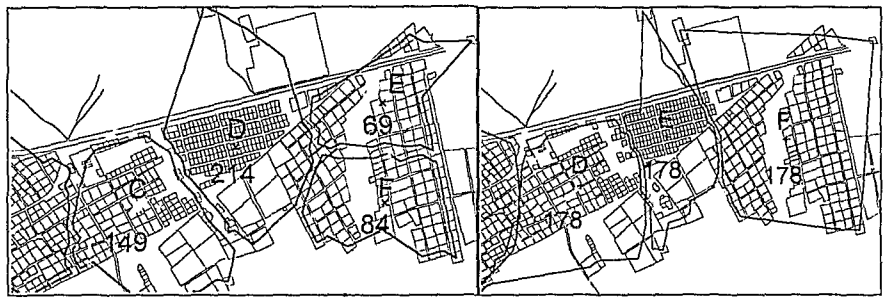


Figure 12 - Township B Rotated Anti-Clockwise

Based on this problem that was encountered with ribbon shaped townships, directions chosen for the sweeping algorithm must always be along the primary axis of the township. This heuristic produced better results and did not cause severe transformer zone deformation as with the township in its un-rotated or natural position.

4.3.5 Rapid convergence to non-con convergent cases

In the case of Township B, after it was rotated, sweeping was done in the East-West direction. It was found that more iterations were required than with the conventional sweeping method but the final solution obtained had significantly less increase in TCL. For example with Township B, the first three results presented in Table 1 were obtained by simply scanning along the primary axis of the township. The first column indicates the number of transformers while the second represents the zone limit. The absolute TCL is presented in metres and the directions of the sweep and the percentage increase in TCL as per normal scanning is presented. The directions for the specific scanning for ribbon shaped townships and its respective increase in TCL are provided in the subsequent columns. The improvement in percentage increase is clear. It should be noted that these are somewhat extreme examples and the occurrences of many sweeps are rare.

[2] The "before" state implies that the transformer zones generated were immediately after the execution of Grimsdale's algorithm.

no. of xfr	Zone limit	TCL	Normal	TCL%	Algorithmic	TCL%
12	154	446795	ldr	16.06	lrlrlrlrlr	9.80
12	150	446795	ldr	17.65	lrlr	13.86
12	148	446795	ldr	19.93	lrlr	8.30
13	146	418344	ldr	12.17	lrlrlrlrlrlrlrdr	9.06

Table 1 - Table of test results

It was found that in cases where excessive sweeping occurred along the primary axis, the rate of convergence may slow down or halt completely. When this occurs, a single sweep in the perpendicular direction and then resuming the sweeping along the primary axis can significantly speed up the convergence of a solution.

Empirically, the following rule was established that will in most cases be adequate for manual scanning process. If sweeping along the primary axis results in the rate of convergence slowing down then perform one sweep in the perpendicular direction. The rate of convergence in this case refers to the rate at which the number of consumers piled up at the end of a sweep are decreasing after every sweep. Then, continuing sweeping along the primary axis will lead to the rapid convergence of a solution. However, a certain number of sweeps have to be performed to determine the rate of convergence before initiating a perpendicular sweep. This is obvious with the last entry listed in Table 1, where it took 16 sweeps to establish that the number of consumers being piled up after the simulation sweeps had stagnated. Again, the number of cases where these conditions occur are rare, especially when considering that the need for sweeps in multiple directions are rare.

4.3.6 Pattern recognition for improved performance

The normal direction determining technique for the scanning algorithm is adequate, including the rule defined above for ribbon shaped maps. However an interesting aspect was noted that provided improved solutions. Suppose one is working with the scanning algorithm, selecting the direction for every sweep for transformer zones with zone limits varying from the maximum to the average. It is usual to note a pattern in the series of directions used for higher zone limits. In the case of Township B, it was noted that when working with larger zone limits, the first sweep was in the Eastward (right) direction. It was also noted that the first direction recommended changes for smaller zone limits. In other words, the starting direction of the sweep recommended by the program for larger zones are not the same for smaller zones.

It was found that if the recommended directions are ignored and the scanning algorithm is initiated with the same directions used with larger zones, better results may be obtained. In order to do this, the user is

required to identify a pattern in the direction specified by the scanning algorithm for larger zone limits and to apply that to smaller zone limits. The data shown in Table 2 is as with Table 1 and the increase in TCL with the conventional method and the intelligent method is quite obvious. The examples shown here are extreme but have been selected to demonstrate this point.

No. Xfr	Zone Limit	TCL	Normal	TCL%	Trial	TCL%
12	167	446795	l	10.45	rl	0.68
12	162	446795	lr	12.52	rl	-1.80
12	156	446795	lrlur	10.56	rl	2.60
12	153	446795	lrlrlrlr	17.84	rl	6.98
12	148	446795	ldr or lrlr	19.93	rlr	10.74
13	151	418344	lr	12.74	rl	5.35
13	146	418344	lrlrlrlrlrlrlrdr	12.17	rl	4.02
13	141	418344	lrlr	16.03	rl	8.40
19	110	515873	lrlrlrlr	3.44	rlrlrlrlrlrlrlrl	-9.38

Table 2 - Table of alternative trial figures

The reason for this occurrence is that as the zone limit varies, the consumer reallocation process may be more effectively accomplished by sweeping in other directions. The simulation sweeping process can only calculate the results of a single sweep in all four directions and not the entire process. Thus the direction recommended for the subsequent sweep may not be the best as far as the entire process is concerned.

4.3.7 Consumer misallocation

Consumers located near the boundaries of adjacent zones tend to be allocated to the wrong transformer. The reason for this is an inherent flaw in the scanning algorithm. This is demonstrated pictorially by the example in Figure 13 showing tow adjacent zones defined for the central part of Township E. Only the consumer numbers are shown but the stand boundaries and road information have been deliberately removed as they will only serve to clutter the map. The points at which the straight lines from the consumers convene are where the respective transformers located. The line located between the two transformers forms the equi-distant locus of those two points representing the transformers. The two different zone boundaries are shown by different lines.

Apart from these zones being severely warped, it can be noticed that the consumer allocations are not logical. Clearly, consumers served from one of the transformers can be better served from the other and vice versa. Thus the refinement algorithm described in section 5 was developed. It

should be understood that the refinement process is a necessity for the scanning algorithm and not an option in order to obtain valid solutions.

The zone boundaries in Figure 13 and Figure 15 were generated by the boundary polygon creation algorithm described in the conceptual design. It can be noted how the polygon fits the consumers like a glove.

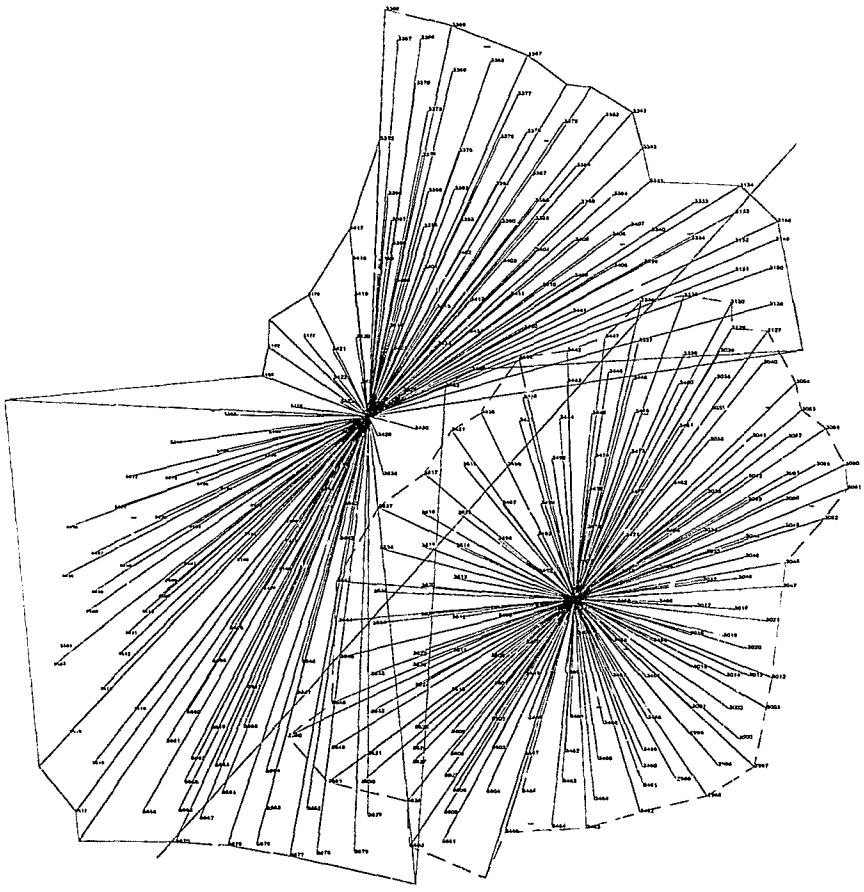


Figure 13 - Warping of transformer zones and consumer misallocation

4.3.8 Effects of map Inversion/Rotation

Using the scanning algorithm for certain maps the following results were gathered. For Township B, the improvements were marginal when the map was rotated or inverted for small number of transformers being placed (for 5 to 10 transformers the improvement was less than 0.2%). However, as the number of transformers being placed are increased the improvement becomes more pronounced to as much as a 9.6% reduction in TCL for 17 transformers. Similarly for Township E, the improvement for 20 transformers was 2.25% and for 50 transformers it was 9%. However,

for 80 transformers the difference was as much as 20% which is significant.

No technique for determining the best transformer zone fit that provides the least TCL for an inverted and/or rotated state at an earlier stage[3] has been established. Since it is not possible to predict the inverted and/or rotated state that would provide the best result after Grimsdale's algorithm, the sweeping algorithm will have to actually be performed.

[3] After the completion of Grimsdale's algorithm for a map in all its inverted and/or rotated states.

5 Refinement Algorithm

Two different techniques for the usage of the refinement algorithm were devised and tested. The effects of inversion and/or rotation were not investigated in this section.

5.1 Post refinement process

In this methodology, the refinement process was repeatedly applied after the scanning algorithm had completed its task to improve the consumer assignments. The time taken for this operation varied from 45 seconds for Township D to over 3 hours for Township E. It should be noted that the time taken for the computation increases exponentially, and not linearly, as the number of consumers involved increases.

From the sample dialogue shown in Figure 14, it can be established that a significant Improvement in the TCL can be achieved after a one or two iterations. After every line where the percentage increase in TCL has been listed (except for the last line), it can be noticed that there is a line starting with `counter =` and listing two integers. This aspect was used during the test phase. The first number lists the number of consumer that have been assigned to transformers that are not the nearest while second number indicates the number of consumers that were successfully exchanged. This process is then repeated until no more consumers can be exchanged and then this process is halted.

It can be also noted that that at the end there are still consumers that have been assigned to transformers that are not the nearest. This is to be expected, as that is the whole reason for the scanning algorithm which improves the somewhat "idealised" locations generated by Grimsdale's algorithm. In other words, at this stage the TCL cannot be reduced further by performing any more consumer exchanges across transformer zones.

```
Percentage increase in total cable length = 15.2998%
counter = 273  203
Percentage change after refinement = 10.1336%
counter = 162  56
Percentage change after refinement = 9.57368%
counter = 141  24
Percentage change after refinement = 9.39426%
counter = 130  6
Percentage change after refinement = 9.33506%
counter = 129  7
Percentage change after refinement = 9.30336%
counter = 126  5
Percentage change after refinement = 9.28412%
counter = 126  6
Percentage change after refinement = 9.26936%
counter = 123  2
Percentage change after refinement = 9.25763%
counter = 123  2
Percentage change after refinement = 9.25295%
counter = 122  0
Percentage change after refinement = 9.25295%
```

Figure 14 - A sample dialogue box

The exhaustive execution of the refinement algorithm as opposed to a single or a double will ensure the correct zone formation, even though the improvement in TCL might become negligible. A sample is presented in Figure 15 to demonstrate the effectiveness of the refinement algorithm. The refinement algorithm was applied to same area in the township as shown in the pictorial in Figure 13. The effectiveness of the transformer zone boundary generated is again visible in the pictorial.

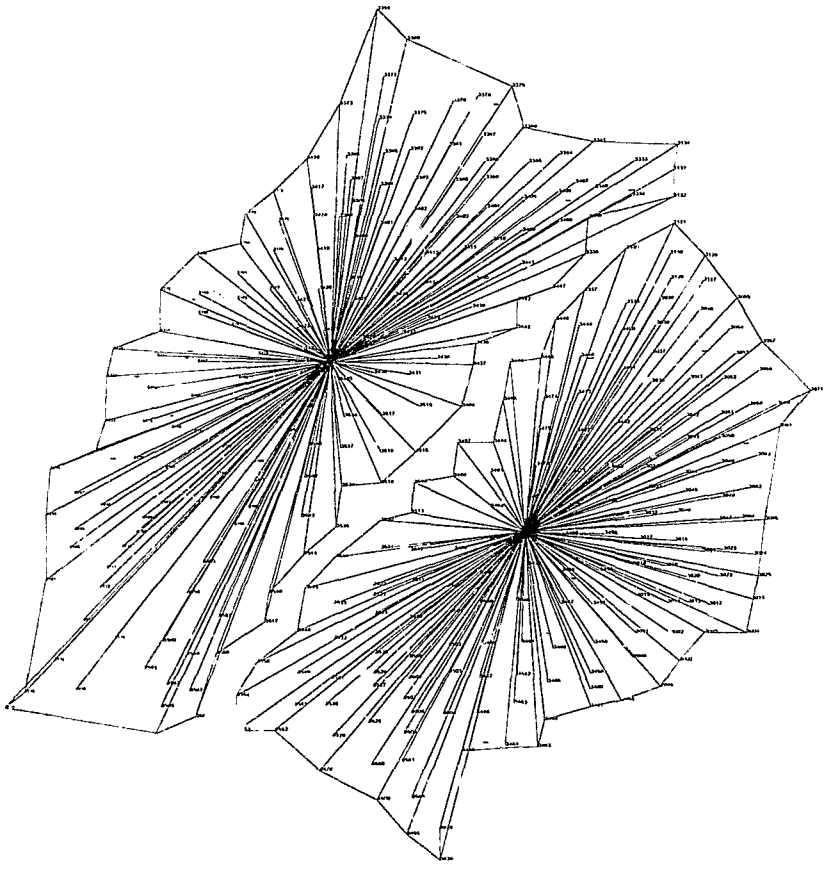


Figure 15 - Refined transformer zones

It can be clearly seen that the consumer allocations have been resolved between these two transformers. It must also be noted that allocations had been resolved between neighbouring transformer zones as well (not shown in the diagram).

The refinement algorithm does make an indisputable improvement to the TCL. The graph shown in Figure 16 is for Township A and the results obtained are for the placement of 7 transformers. In the graph in Figure 16, the general increase in TCL as zone limit is reduced can be clearly seen. It can also be observed that the improvement in TCL, though small, is undeniable and becomes more pronounced as the lower zone limit is reached.

Author Rajakanthan T
Name of thesis Software Modules To Further Assis The Rapid Production Of Optimised Electrical Reticulation Schemes Rajakanthan T 1999

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.