

https://dspace.org/

An approach to solving a scheduling problem arising in industry: A scheduling problem arising in a credit and risk call centre

Stoltz, Adham Raymond

2018

https://hdl.handle.net/10539/25625

Stoltz, Adham Raymond, (2017) An approach to solving a scheduling problem arising in industry, University of the Witwatersrand, Johannesburg, https://hdl.handle.net/10539/25625

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

An Approach to Solving a Scheduling Problem Arising in Industry

Adham Stoltz

(Student No: 0104009J)

Supervised by: Prof Montaz Ali

School of Computer Science and Applied Mathematics,

University of the Witwatersrand, Johannesburg, South Africa.

November 2017

Submitted in fulfillment of Master's of Sciences



WITS
UNIVERSITY

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly. This dissertation has been submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, South Africa, in fulfillment of the requirements for the degree of Masters of Science. The work has not been submitted before for any degree or examination to any other institution.



Adham Stoltz, 25 March 2018

Abstract

This work is a result of the necessity to resolve a real world problem: a call centre capacity problem. The result of the capacity challenge is a high cost of operations and large number of disgruntled customers. The research comprises two components: developing a new mathematical model as an extension to existing scheduling problems and then also the industrial engineering component of implementing this new scheduling model into the call centre operation in which the problem was identified.

The real world problem is introduced and discussed in detail, with particular emphasis placed on the unique features and events which give rise to the capacity problem.

The problem is shown to share many features with the class of problems known as Scheduling Problems. Particular attention is paid to the sub classes of Resource Constrained Project Scheduling (RCPSP) and Resource Levelling Problems (RLP). A review of the literature relating to these classes of problems, as well as the extensions and features that are relevant to the real world problem, is presented.

The real world problem is discussed in more detail and framed in terms of the conventions that have been established for RCPSP and RLP. Similarities and differences between the real world problem and established models from literature are discussed. The special and unique features of this problem are then shown to be: the need to perform scheduling on a batch basis every day, without knowledge of what will occur the following day and also the requirement that resources (call centre capacity) are fully utilized and not exceeded. There is also no terminal state which can be planned for, batch scheduling continues indefinitely. The result of the research is that a new mathematical model called the Critical Time Window Resource

Levelling Problem (CTWRLP) with the Continual Rescheduling method is developed and proposed as a method for solving the real world scheduling problem. This method addresses all the requirements of the real world problem.

An approach to solving the model in the practical environment is also presented. This involves additional pre-processing required to prepare all inputs for the scheduling model, namely creating sub-models for calculating resource consumption and resource availability. The global optimization technique of Simulated Annealing is then introduced as this is the method chosen to solve the optimization component of the CTWRLP.

The CTWRLP model is implemented in the call centre. The practical algorithm, numerical solution technique, results and success of the model are presented.

Acknowledgements

Thank you to my supervisor Prof. Montaz Ali for his continuous support and patience.

To my parents: Merle, Ray for your unwavering belief in me. To my darling daughters Sophia and Rebecca, thanks for letting dad sit and work while you guys played around my chair. To my beautiful wife Fulvia thank you for your support, love and care. I could not have finished this without you!

Special thanks to an incredible leader and boss, Linda Morgan, who trusted my vision and allowed me to use her world as a playground.

Dedication

Dedicated to:

Sophia and Rebecca - May your futures shine as the brightest stars.

Contents

Abstract ii

1 Introduction 1

1.1 Background to the Problem 1

1.2 The Field of Scheduling 2

1.3 Purpose 3

1.4 Thesis Overview 4

2 Detailed Introduction to the Problem 5

3 Literature Review and Motivation for Research 14

3.1 The Resource Constrained Project Scheduling Problem 15

3.2 Extensions and Variations of RCPSP 17

3.3 Resource Levelling Problem 22

3.4 Machine Scheduling Problems and the Flow Shop Problem 28

3.5 Summary and Research Objectives 31

4 Problem Characterisation and Research Findings 33

4.1 Scheduling models and application to the real world problem 33

4.2 The New Model and Features to Address Real World Problem 40

4.3 Summary 49

5 Implementation Approach 50

5.1 Summary 50

5.2 Solution Approach 53
5.3 Summary 71

6 Simulated Annealing **72**
6.1 Introduction to Simulated Annealing 72
6.2 Simulated Annealing Algorithm 74
6.3 Summary 76

7 Solving the Critical Time Window Resource Levelling Problem **77**
7.1 Model Calculations 79
7.2 Features of Scheduling Algorithm 94
7.3 CTWRLP with Continual Rescheduling 96
7.4 Practical Demonstration of Algorithm 98
7.5 Summary 107

8 Conclusion **108**
8.1 Conclusion and Discussions 108

References **117**

A MATLAB Code **118**
A.1 Objective Function (Critical Time Window) Computation 118
A.2 Algorithm Computation Steps 121
A.3 Candidate Solution Creation 123

List of Figures

2.1 High level process view of the collections department. 6

2.2 Illustrative Response Rates for Low Risk Customers. Lines are the collections actions, y axis shows response rate and x axis shows number of days since the collections action was performed. 11

2.3 Illustrative Response Rates for High Risk Customers. Lines are the collections actions, y axis shows response rate and x axis shows number of days since the collections action was performed. 12

3.1 Generic acyclic graph illustrating a nine activity project with precedence relationships between activities. 16

3.2 Feasible schedule for a permutation flow shop problem with $n = 4$ jobs and $m = 3$ machines. 30

4.1 Schematic of conceptual algorithm for solving the real world scheduling problem. 37

4.2 Resource utilization over 8 time periods in a resource levelled schedule 45

4.3 Resource utilization over 8 time periods in a CTW resource leveled schedule where the CTW is $\kappa = 3$ periods. 46

5.1 Profile of accounts going into arrears during a typical month. 56

5.2 Call profile as result of collections actions. 56

5.3 Call volumes per type for a typical month. 59

7.1 Typical profile of activities requiring scheduling in the collections environment. 99

7.2 Inbound Call Volumes in unscheduled environment. 99
7.3 Inbound Call Volumes in Scheduled Environment. 103
7.4 Day 2 - Unscheduled case response profile 104
7.5 Day 2 - Scheduled case response profile 104
7.6 Day 3 - Unscheduled case response profile 104
7.7 Day 3 - Scheduled case response profile 104
7.8 Day 4 - Unscheduled case response profile 104
7.9 Day 4 - Scheduled case response profile 104
7.10 Credit and Risk Call Centre Service level. *Extracted from Financial Year F12 Final Executive Report.* 105

List of Tables

5.1 Cohort response rate for 20,000 high risk direct debit customers 61
5.2 Activity days for collections strategies. 65
5.3 Call Centre planning input metrics. 66
5.4 Call Centre workforce break schedule inputs. 66
5.5 Call Centre workforce and capacity (service level, SvL) planning tool. 66

7.1 Results of CTWRLP Algorithm Scheduling 102

Chapter 1

Introduction

Call centre workforce scheduling is a challenge that every call centre operations manager faces. Much time and effort is spent using several commercially available tools to attempt to understand the pattern of inbound calls and ensure adequate resourcing of the call centre to meet the performance goals. This dissertation focuses on a particular call centre operation experiencing a severe capacity challenge, where the nature of the call centre and the industry in which it operates means that the currently available scheduling tools fall far short of providing assistance in solving the problem (other than demonstrating the call centre should increase staff complement by around 300% which is not an option due to cost constraints).

1.1 Background to the Problem

The credit and risk collections call centre in a South African telecommunications company experiences a massive influx of calls into the call centre at certain peak periods during the course of a month. There is limited capacity to handle these calls, due to staffing constraints. This results in large numbers of calls being abandoned, and a very low service level being achieved by the call centre (service level refers to the number of calls answered within a certain acceptable wait time - the target for this call centre being 80% of calls answered within 20 seconds. This is a commonly adopted target service level guideline across many

call centres). Some preliminary investigations were conducted, and it was found that the majority of calls into the centre occur as a direct result of collection actions which are taken against customers. The methods used by the credit and risk team to perform these collections include: SMS notifications, email notifications, physical letters sent to account holders as well as the suspension of some or all services provided to the customer. Most of these collections activities result in the account holders making telephone calls to the credit and risk department. The objective for this particular business area (credit and risk) is to reduce bad debt (by collecting on arrears accounts) as well as to provide excellent customer service. In addition to these business-critical objectives, the credit and risk department is required to operate at the least possible cost to the organisation as a whole. This is in line with the global drive to reduce costs in order to improve profit.

A significant portion of any organisation's cost is the labour or workforce cost. After some consideration of the dynamics involved in this real world problem, it became apparent that this problem can be contextualised as a scheduling problem.

Solving the scheduling problem will alleviate the strain on the call centre and allow the collections department to operate more efficiently (cost effectively) in general.

The problem considers the optimal assignment of activities (collections activities) such that there is sufficient resource capacity (call centre capacity) to handle the resulting phone calls, as well as ensuring that collections are performed timeously - in order to ensure bad debt is kept to a minimum (avoid the situation where non-paying customers roll over into the next billing period and therefore increasing the bad debt exposure).

1.2 The Field of Scheduling

Quite generally, in a scheduling problem, the task is to find an optimal order for activities (or jobs/tasks/projects) to be executed (or processed/performed) in order to satisfy some constraints or objectives. The constraints include resource constraints as well as precedence constraints and the objectives are typically to minimize the makespan (total duration) or to

meet some deadline criteria. There is also a special case in scheduling where the objective is to ensure smooth utilization of the resources across the project time-line, thereby minimizing wastage. This is particularly relevant in the field of construction project scheduling, where the resources are mostly labour and under utilization results in idle workers.

Scheduling is a topic which has been extensively studied for the last 50 years. In a state-of-the-art review of methods for short term scheduling of batch processes Mendez et al. (2006) list a total of 13 major categories, with several sub categories, for classification of batch scheduling problems. At the present time there is no single model which would be capable of handling all possible characteristics of an arbitrary scheduling problem.

A quite general scheduling model is the Resource Constrained Project Scheduling Problem (RCPSP) Brucker et al. (1999); Hartmann and Briskorn (2010). The RCPSP model provides a framework to describe scheduling problems in terms of activities, constraints and an overall objective Brucker et al. (1999); Hartmann and Briskorn (2010). When the overall objective is focussed on the smooth utilization of resources rather than exclusively minimizing the makespan, the RCPSP is then referred to as the Resource Levelling Problem (RLP) Demeulemeester and Herroelen (2002). This general framework lends itself to the problem at hand. Project scheduling is a broad field with many associated models Demeulemeester and Herroelen (2002). Upon reviewing key characteristics of the real world problem, it was decided to explore the RCPSP and RLP models as they catered for the need to address constraints around resources. In this dissertation, both RCPSP and RLP will be used variously to refer to the class of scheduling problems to which the real world problem belongs. The underlying formulation and structure of both RCPSP and RLP are the same, with the difference being the objective function being minimized. There are many extensions to RCPSP. The extensions relevant to our problem will be discussed and the applicability reviewed.

1.3 Purpose

The objectives of the study are as follows:

- Research the real world problem.
- Review various models for RCPSP and RLP and assess applicability to the problem.
- Develop a new model to fit the nature of the problem.
- Propose and test a solution approach for the new model.
- Implement the solution and resolve the capacity issue being experienced by the call centre.

1.4 Thesis Overview

The dissertation is organised in the following manner. The real world problem is described in detail in Chapter 2. We then review the literature relating to scheduling problems and summarise currently understood models in Chapter 3. Various extensions and enhancements to scheduling models are also discussed. In Chapter 4 the real world problem is characterised and defined in terms of the literature. The new scheduling model Critical Time Window Resource Levelling Problem (CTWRLP)

Chapter 2

Detailed Introduction to the Problem

In this chapter, the real world problem is presented in detail. The characteristics and features that make this problem interesting are highlighted. We begin to show why a scheduling model would be a good approach in attempting to solve this problem.

A credit and risk debt collection department for a major cellular telecommunications is responsible for collecting on invoices which have been issued to customers and which have not been paid.

A high level process view of how the department operates ('The System') is presented in Figure 2.1. Terminology such as 'self-cure' and 'soft-lock' are introduced. This terminology is industry specific jargon and will be defined and explained in this section.

The customer will first come into the system when they fail to make a payment on their agreed due date. Typically, this results in batches of accounts appearing in the system at unpredictable intervals over the course of the month. The number of accounts that appear is also unpredictable. For example, in one month a batch of 10,000 accounts may appear on a particular day, while at the same time of the month in the previous month, only 2,000 accounts entered the system.

Once an account enters the system, **COLLECTION ACTIVITIES** will be performed on that account, in an attempt to get the customer to pay. In this environment, the collection

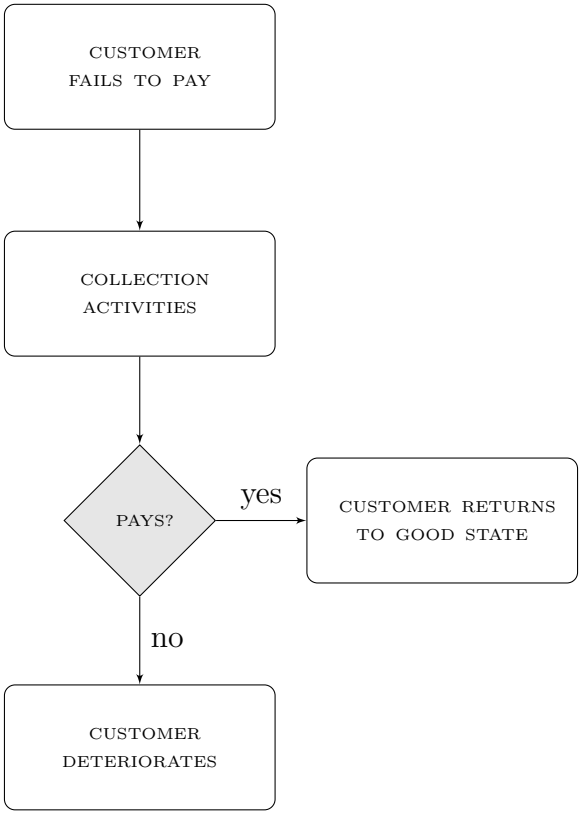

Figure 2.1: High level process view of the collections department.

activities are performed by an automated work flow system which has the ability to perform millions of actions concurrently. These modern work flow systems are excellent capacity multipliers and allow organizations to perform collection actions on many more arrears accounts than would have been possible if they had to rely solely on human credit controllers.

Some examples of collections activities could be:

- An SMS to the customer, requesting the customer to make payment.
- SOFT LOCK (SL) - the customer's cellular service is suspended.

With regard to the problem which formed the basis of this study, there are 3 distinct activities which are performed (precedence relationship: ACTIVITY 1 < ACTIVITY 2 < ACTIVITY 3). This means that the collections actions need to follow in a specific order, and a late stage collections activity (say soft lock for instance) may only occur once the second sms (SMS 2) has happened, which in turn needs to be preceded by the first sms (SMS 1).

For example, if a customer does not respond to the first 2 collections activities (which are SMS notifications) then activity 3 will be performed (SOFT LOCK). We now define a **RESPONSE** as an event which occurs after a customer has been subjected to a collection action.

A **RESPONSE** (reaction) can take three possible forms:

- The customer can make a payment, without contacting the collections call centre (SELF CURE)
  - The workflow system will recognise the payment and remove the account from the collections environment.
  - No further collections actions will need to be performed.
- The customer can phone the collections call centre, in order to negotiate or to make a payment.
  - The customer's account will be removed from the early collections work stream - no further collections activities will need to be performed.

- The customer will ignore the collection action. Technically, this is not a response at all, but it will be categorised as such for the purposes of this study as the likelihood of this event occurring will need to be modelled and included in the mathematical model.
    - Further collection actions will need to be taken against the account.

In the telecommunications company, the vast majority of customers pay via direct debit (customer's monthly bill is paid automatically directly from their bank account to the telecommunications company). This means that if the payment fails (for example: insufficient funds in the designated bank account) the customer either does not know how to pay the telecommunications company directly, or the customer will need to negotiate a payment arrangement to settle the arrears amount on their account. In either case, the customer needs to phone the collections contact centre in order to request that a direct debit resubmission is performed by the telecommunications company, or to negotiate an arrangement to pay the money he owes if he/she cannot afford to do so at that point in time. This characteristic introduces the capacity constraint: The capacity of the call centre to handle incoming calls from customers. It is this call centre capacity which needs to be taken into account when taking collections actions on accounts in arrears.

If too many SMS notifications are sent out, the influx of calls in response to the SMS could be in excess of the available capacity on that day. If it were as straightforward as staggering the release of SMS notifications over a period of days, this problem would be very straightforward to solve. In other words, if we were able to take an approach where we were able to schedule the start of the collections treatments in some way so that the notifications sent out on a given day would not result in too large an influx of calls, then this would be a straightforward problem. The issue is that the downstream effects need to be taken into account and the combinatorial nature of the problem adds significant complexity, making this approach unrealistic. Another characteristic, which adds a significant level of complexity to the problem, is that for this telecommunications company, customers do not all pay on the same day. In fact, customers' payment due dates can be any day of the month. This situation is further exacerbated by the fact that there may be, and often is, a delay in the information about the rejected payment getting to the telecommunications company from

the bank. These two factors combined mean that to a large extent, the number of customer accounts appearing in the collections system on a given day is unpredictable. There are peaks and troughs (most customers prefer to have their payment due dates at the beginning or end of a month) but accounts may enter the system up to seven days after they missed their payment due dates. At this point we would like to point out that a scheduling model which is flexible and will allow for reactive rescheduling of batches is clearly needed. A scheduling model which relies on *a priori* knowledge on the number of batches to be scheduled on a given date will not suit this environment.

We now introduce the concept of a collections "strategy". A collection strategy is a group of activities which are taken against an account which has been identified as requiring debt collection intervention. Collections strategies are typically designed by business subject matter experts and are based on experience and opinion, rather than data or analysis. A typical collections strategy could be as follows:

1. On the first day where the payment is overdue, an SMS will be sent to the customer notifying him that he has missed the payment [Day 1 of collections strategy].
2. Two days later another SMS will be sent [Day 3 of collections strategy].
3. Three days later another SMS will be sent, significantly more harshly-worded than the previous SMS [Day 6].
4. Five days after the last SMS, the customer's service will be suspended (SOFT LOCK) [Day 11].

In the credit risk collections industry, it is an usual practice to use segmentation models in order to segment the customer base into high, medium and low-risk categories. These scorecards make use of statistical methods to provide a probability score for each customer, which will indicate their propensity for becoming a bad debt customer. Customers which exhibit similar risk characteristics are grouped together into segments. The collections strategy described above would usually be applied against a high risk customer. A lower risk customer might be allocated to a collections strategy which would only see them reaching the soft lock stage 40 days after their (missed) payment due date.

A fact which has been completely ignored in the design of traditional collections strategies, is that in addition to having a particular bad debt risk profile, each segment of the customer base has its own "response" profile. This means that each customer segment has a different propensity to respond to the various collections mechanisms. Typically, and perhaps intuitively, the lower-risk or better-behaved segment (those customers with good payment history and account status) have significantly higher response rates to the first SMS notifications, while the high-risk (poorly-behaved) segment have low response rates to SMS notifications and only start responding once they have been soft locked.

In the un-scheduled environment, the customer base was segmented into 5 distinct segments (based on their behavioural score). This was the approach taken by the debt collections prior to the interventions presented in this dissertation. It was based on the experience and gut feel opinions of the management team. The output of the behavioural scorecard Anderson (2007) calculation (the segmentation model discussed above) was used to allocate a customer to each of the five segments. The segments are referred to by their risk characteristic, namely : Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH). The H and VH segments would be exposed to quicker collections strategies, the M segments to slightly longer, and the L and VL even longer still. There was no consideration given to how the overlap of these strategies and the varying response rates of customer segments to collections actions, may negatively impact call centre service level.

We conducted an investigation to establish whether there was merit to the idea that different customer segments would demonstrate distinctive response behaviours. The methodology is presented in detail in Chapter 5. The information in Figure 2.2 and Figure 2.3 is based on data collected and analysed for the call centre systems. Average responses were calculated based on three months worth of transactions. All transactions (system driven collections actions and inbound calls) were analysed. A combination of SAS and MS Excel were used to perform the data preparation and analysis. Here, in Figure 2.2 and Figure 2.3, we will present a simple illustration of the findings. In Figure 2.2 and Figure 2.3 the response rates of Low Risk and High Risk customers to the various collections actions are respectively illustrated. The figures show the following: The x axis shows a categorical grouping, representing the

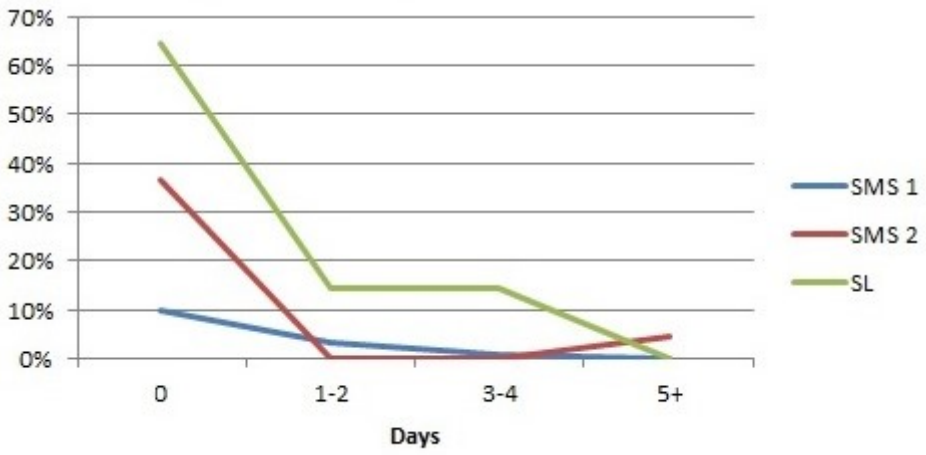

Figure 2.2: Illustrative Response Rates for Low Risk Customers. Lines are the collections actions, y axis shows response rate and x axis shows number of days since the collections action was performed.

number of days after the collections action which customers respond and the y axis shows the percentage response of the total customers who were exposed to a collections action. From Figure 2.2 and Figure 2.3 we see that there are significant differences in the response profiles of customers from differing risk segments. Low risk customers have a higher response to the initial SMS messages, and a lower response to the more harsh action of soft lock (SL), higher risk customers still respond to the initial messages, but in much fewer numbers than low risk customers, while far more high risk customers respond to the soft lock (SL). (We speculate this reduced response is a feature of low risk customers - when low risk customers - who are habitually well behaved customers - do not pay and then ignore the initial SMS messages, it may be likely due to some unforeseen financial or other disruptive event, the account will never be settled. Conversely, higher risk customers are habitual late payers who know the collections process and wait for the soft lock to delay as long as possible before paying). It is important to note that even though customers respond to a collections action, that does not mean that he/she is now completely dealt with and no longer needs to be considered in the collections environment, i.e.. he/she will not exit the system. They will continue to receive collections treatment according to the collections strategies until they make payment.

We now will briefly summarize the information on the problem presented earlier:

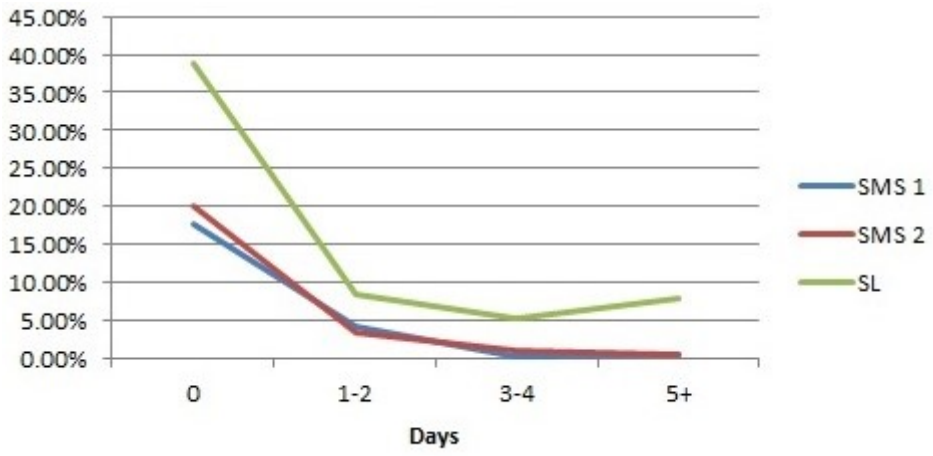

Figure 2.3: Illustrative Response Rates for High Risk Customers. Lines are the collections actions, y axis shows response rate and x axis shows number of days since the collections action was performed.

- Customer accounts can enter into the system on any day of the month.
- The number of accounts entering the system on a given day is unpredictable.
- The accounts entering the system will be distributed across the five segments mentioned above. Each segment has its own average cure rate (the proportion of account holders who will restore themselves to good standing after receiving notifications but without having to engage in any further interaction with the collections center) and response rate to collection activities.
- Collections strategies will be applied to all accounts which entered the system on that day.

The additional characteristics or features that are overlooked by the telecommunications company in its current debt collection operations are now presented. In order to illustrate the effect of an overlap consider a simple example: Suppose 10,000 accounts entered the collections system just after the middle of a given month, say on day 17 or 18 of the month. Suppose 5000 of these accounts are high risk accounts. Based on the collections strategy presented above, we know that the majority of these accounts will be soft locked 11 working days from now, which will be the beginning of a new month. Now, at the beginning of the

month, we know too that we can expect large volumes of accounts to be entering into the system. We will experience a scenario where perhaps 4000 of the accounts which entered 11 days ago are being soft locked, in conjunction with say 10,000 SMS notifications being sent out to low-risk customers who have just entered the system on that day. This will result in high volumes of inbound phone calls, perhaps too many for the call centre to handle.

It is this characteristic - collections actions overlapping and resulting in customer responses - which creates an inbound call load too great for the call centre to handle. Our approach to solving this problem is to create data driven collections strategies: Collections actions are performed dynamically according to available capacity, business rule constraints and the objective of minimizing bad debt. The problem then becomes an exercise in managing the scheduling of collections actions so that the resulting phone calls never exceed the capacity threshold of the call centre, as well as ensuring that actions are not too delayed which could result in additional bad debt. We will formulate this business problem in the context of a scheduling model. We will show how certain features align with existing models for scheduling problems in the literature, and will present new models to deal with the features of this problem that have not yet been addressed in the literature. We will present these extensions in Chapter 4.

Chapter 3

Literature Review and Motivation for Research

Scheduling problems and applications are a well known class of applied problems that occur in many forms /citet10. Some examples could be the scheduling of labour resources in a construction project, the scheduling of operations by the CPU of a desktop computer, the scheduling of manufacturing jobs in a production line or creating a roster of flight crew duties for an international airline operator. There is extensive literature relating to scheduling problems and considerable work has been done in the field in order to abstract the models to a level where a scheduling model could be applied to many practical problems. Due to the diversity of the practical problems, there is no single scheduling model that can be universally applied. In Chapter 2 we have shown that the real world problem being considered in this study can be characterised as a scheduling problem. In this chapter, we will present a summary of the literature relating to the real world problem. After considering the elements of the thr problem at hand, notably the features of needs to schedule batches of jobs, or the resource constrained nature of the problem, particular focus will be given to three categories of scheduling model: Resource Constrained Project Scheduling, Job / Flow Shop Scheduling and Resource Levelling Problems. We will occasionally make reference to certain features of the real world problem as various concepts from literature are introduced. The problem will then be fully explored later in Chapter 4. This chapter will conclude with the motivation

for this research and research objectives being presented.

3.1 The Resource Constrained Project Scheduling Problem

The standard RCPSP Brucker et al. (1999) can be stated as follows: The RCPSP concerns a project with J activities which will be scheduled. The duration of a given activity j, which is also referred to as the 'processing time' (the number of time periods required to complete activity j given sufficient amount of required resources), is denoted as d_j. In the general case of RCPSP, an activity may not be interrupted. Once started, it cannot be stopped until its end state has been reached. There may also be precedence relationships between some activities. They arise out of practical considerations; for example, one cannot disconnect a customers cell phone service until you have contacted them twice already via SMS. These precedence relationships are represented by sets of predecessors P_j, which are used to ensure an activity j may not be started until each of its predecessors $i \in P_j$ has been completed. There are also two additional activities: $j = 0$ and $j = J + 1$ which represent the start and end of the project. These are dummy variables which have 0 duration and 0 resource requirement. A project is thus represented by an acyclic graph where nodes represent the activities and arcs represent the finish - start precedence relationships. A generic example of an acyclic graph is presented in Figure 3.1. This figure represents a project with nine activities, and precedence relationships as can be observed by the directed edges.

Activities are constrained as follows:

- Precedence constraints require activity i to be scheduled after all preceding activities to i have been completed.
- Performing activities consumes resources. Resources have limited capacity.

Each activity requires resources in order to be completed. Three types of resource categories are identified, Demeulemeester and Herroelen (2002):

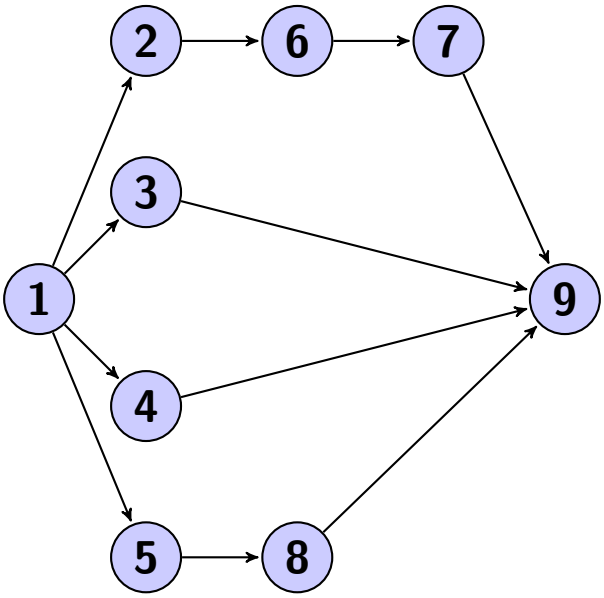

Figure 3.1: Generic acyclic graph illustrating a nine activity project with precedence relationships between activities.

Renewable resources which are renewed from period to period (e.g. call centre capacity available in a day).

Non - renewable resources which are resources available on a total project basis (For example, in the context of the real world problem, one could investigate creating a dummy non - renewable resource to represent bad debt risk).

Doubly - constrained resources which share the characteristics of both of the above resource types.

For the standard RCPSP there are K renewable resources labelled $k = 1, \ldots, K$. For each resource k, the availability of that resource is assumed to be constant over time. This resource availability (or capacity) is represented by the set R_k. The resource consumption rate of an activity j can be represented as r_{jk} units of resource k for each time period of its processing duration. The typical objective of RCPSP is thus to find a schedule which organises all the activities J such that they are completed, while satisfying resource constraints R_k. Activities need to be scheduled in such a fashion as to ensure the earliest possible completion of the project, i.e. minimize the makespan. 'Makespan' refers to the total duration of the project

Demeulemeester and Herroelen (2002). This may be stated as assigning start time s_j to the activities $j = 0, \ldots, J+1$ such that the total duration of the project is minimized.

3.2 Extensions and Variations of RCPSP

There have been a number of extensions to RCPSP which have been developed in order to cater for practical problems. The reader is referred to Hartmann and Briskorn (2010) for a recent and thorough overview of all the various extensions which have been studied. In this chapter, we discuss the extensions which are relevant to the basic problem studied in this dissertation.

3.2.1 Resource consumption varies with time. In the standard form of the RCPSP, the resource requirement rate is constant over time. The RCPSP can be generalized by resource requests varying with time, see Hartmann and Briskorn (2010). This can be denoted as $r_{jk}(t)$, which is the requirement of activity j for resource k at time period t, where t is an element of the duration of j. Approaches to dealing with this extension of RCPSP have been studied in Cavalcante et al. (2001) amongst others. For our problem, resource consumptions could be considered to vary with time. For example, in Figures 2.2 and 2.3, we see that the response rate on day one of an SMS action is much higher than that on the following days. In other words, on the first day the SMS is sent, we see a certain response from the customers who receive that SMS. Then, on the following day, without having received any further SMS's, there will be additional responses from those same customers (who received the SMS on the previous day). This could be charaterised as an activity (SMS) having a variable resource consumption requirement (Higher on the first day, progressively lower thereafter).

3.2.2 Multiple modes. The standard form of RCPSP considers the case where each activity can only be executed in a single fashion, determined by fixed duration and fixed resource requirements. This extension of RCPSP can be stated as follows: An activity j must be performed in one of its modes m where $m = 1, \ldots, m_j$ with M_j being the number of modes. Once started in a particular mode, an activity must remain in that mode until the end of

its duration. The processing time of activity j which is processed in mode m is given by d_{j_m}. The resource requirement of activity j, processed in mode m for resource k is given as $r_{j_m k}$, or $r_{j_m k}(t)$ for the case where resource consumption varies with time. The Multi-Mode RCPSP has also been extensively studied in literature, see references Brucker et al. (1999); Hartmann and Briskorn (2010); Yang et al. (2001). In our problem, a collections action (activity), an SMS for example, will be performed against different risk segments. An SMS activity against the high-risk customers will result in a lower number of inbound calls than an SMS activity against the low risk customers. So in essence, the same activity, an SMS, will result in different resource consumption rates depending on the segment against which it is applied. This could be modelled by using a multiple mode approach.

3.2.3 Resource capacities varying with time. In standard RCPSP, the capacity of renewable resources is assumed to be constant, i.e. the capacity is the same in every period. To cater for resource capacities changing with time, the capacity or availability of renewable resource k in period t can be denoted as $R_k(t)$. Time-based variable resource capacities have been studied by Hartmann and Briskorn (2010). We have already indicated that in our real world problem, the resources, call centre capacity (and bad debt risk), will change per time period based on external factors.

3.2.4 RCPSP with rescheduling. This is referred to as reactive scheduling. Several methods of dealing with rescheduling have been researched by Hartmann and Briskorn (2010). This is perhaps the least obvious extension of RCPSP which is studied as part of this dissertation. Rescheduling is necessary when for some reason, a schedule of activities needs to be recalculated after the start date of the project. By this point, some activities may already have been completed and may be ignored, while others might be in progress and should be considered as fixed, i.e. the effect they have on resources needs to be taken into account in the constraints. Typically, a scheduling solution is created once at the outset of the project. The objectives and constraints are defined (project deadline and resource capacities respectively) and then the optimal schedule of activities or jobs is created. The project is then executed according to the schedule. In some instances though, unforeseen circumstances necessitate the adjustment of a schedule. This is where the flexibility of the

model to adapt to new conditions or constraints becomes critical. A simple example of this could be to consider a schedule for a construction project. If there are some delays during the first stage of the project, the schedule will need to be updated to take this into account. All activities that have been performed cannot be changed and a new optimal schedule will need to be created based on the current state of the project. We have already indicated that in the real world problem environment, there is no particular project end date. Activities requiring scheduling are continuously introduced into the system. Every time period t, new accounts enter into the system, which will necessitate the rescheduling of all the activities j which have already had start time assigned s_j, $s_j \geq t_{now}$, where t_{now} is the current or active time period. This extension of RCPSP is especially relevant to the problem at hand, a mathematical model that takes into account the need for dynamic rescheduling will need to be developed.

3.2.5 Multiple projects. The multiple project extension of RCPSP refers to the case when multiple projects are involved, which are all dependant on some common resources, need to be scheduled simultaneously. Several approaches to dealing with this extension have been discussed in the literature (Hartmann and Briskorn (2010); Kumanan et al. (2006)). The most common approach is to consider the multiple projects as a large super network, with an associated resource which supplies the entire network. Kumanan et al. (2006) use a genetic algorithm to schedule a multi-mode, multiple project problem. We discuss this further once we discuss the other elements which are included in the problem formulation.

3.2.6 Formulation of the Renewable Resource, Multi - Mode RCPSP (MRCPSP). In this section we present a formulation of the general Multi-Mode RCPSP as per Yang et al. (2001). A project is a series of activities which may be represented in an activity-on-node graph: $G(N, A)$, where A is the set of pairs of activities between which a finish-start precedence relationship with minimal time lag of 0 exists and N is the total number of activities in the graph. A set of J activities, numbered from dummy start node 0 to dummy end node $N + 1$ i.e. $J = N \cup \{0, N + 1\}$ are to be scheduled on a set of K renewable resources. The schedule of activities needs to adhere to the constraints of precedence (activities need to occur subject to their precedence relationships) and resource

availability. Each activity $j \in N$ can be performed in one of m_j various execution modes. The duration of an activity j, executed in mode m_j is defined as d_{jm_j}. Each execution mode of each activity will require the use of some renewable resource units $k \in K$. The resource requirement of activity j executed in mode m_j of renewable resource k is denoted as r_{jm_jk}. The availability of each renewable resource $k \in K$ is denoted as R_k and is constant for every time period of the project duration. A schedule S is defined by a vector of the activity start times s_j and their corresponding time of completion f_j. A schedule is considered to be *feasible* if all the precedence and resource constraints are satisfied. The objective of the MRCPSP is to minimize the total duration of the project, and thus to find an *optimal* solution where all constraints are satisfied and all activities occur in such a way as to achieve the shortest possible project duration. This is achieved by finding a start time s for the final dummy activity $J + 1$ such that the overall makespan is the shortest. Due to precedence relationships, this means that start times for all the preceding activities will need to be established such that this overall minimization can be achieved.

The MRCPSP may thus be formulated as follows:

Minimize

$$s_{J+1} \tag{3.2.1}$$

subject to the following constraints: *Precedence constraint*

$$s_i + d_{im_i} \leq s_j, \qquad \forall (i,j) \in A, \tag{3.2.2}$$

where d_{im_i} is the duration of activity i executed in mode m_i. s_i is the start time for activity i and s_j is the start time for activity j. A is the set of pairs of activities between which finish-start precedence relationship with minimal time lag of 0 exists. *Renewable resource capacity constraint*

$$\sum_{j \in S(t)} r_{jm_jk} \leq R_k, \qquad \forall k \in K, \forall m_j \in M_j, \tag{3.2.3}$$

where $S(t)$ denotes the set of activities in the time period $[t-1, t)$ multi-mode execution constraint is restricted by $m_j \in M_j, j \in N$, where N is the set of all activities.

Each activity j has to be performed in exactly one mode m_j. *Project start time constraint*

$$s_0 = 0, \tag{3.2.4}$$

the project must start at time 0. *Non - negativity constraint*

$$s_j \in \mathbb{Z}^+, \qquad \forall j \in N, \tag{3.2.5}$$

activity start times must be non-negative integer values. This is the general formulation of a MRCPSP. We could consider the process of scheduling collections activities against arrears accounts as the projects to be scheduled.

3.3 Resource Levelling Problem

The Resource Levelling Problem (RLP) is a class of scheduling problems which differ to RCPSP in that RCPSP considers resources to be constraints while trying to minimize makespan where RLP is concerned with creating a schedule such that the utilization of resources is efficient according to some criteria (mostly that resource utilization is levelled or as closely levelled as possible over the project time line).

Like RCPSP, RLP has been studied for quite some time. In Petrović (1968) we see a consideration of RLP in the 1960's. The methods of solving RLP are similar to RCPSP with exact, heuristic and meta-heuristic procedures being used depending on the complexity and size of the problem being considered, Hegazy (1999). The methods and algorithms used to solve RCPSP all extend to RLP Hegazy (1999).

RLP inherits most of the features and variations of RCPSP, the difference in the models is the formulation of the objective functions. The resource levelling problem has been shown to be NP - hard even if only one resource is considered, Bagheri and Hessam Mahmoudi (2012).

3.3.1 Formulation of the Resource Levelling Problem (RLP). In RCPSP the objective is to always minimize the makespan of the project. In RLP, there are various objective functions which all attempt to minimize resource consumption fluctuations. We now present the model for the standard deterministic RLP with the objective functions currently studied in literature. As in RCPSP the model considers a project as a series of activities which may be represented in an activity-on-node graph: $G(N, A)$.

The following formulation differs from RCPSP in that it considers the project deadline in the constraints, where RCPSP minimizes makespan (deadline).

In Ponz-Tienda et al. (2017) a model for RLP is demonstrated as follows:

1. The set N of activities which need to be executed, with dummy activities $\{j_0, j_{n+1}\}$ as start and end of the project:

$$N = \{j_0, j_1, \ldots, j_n, j_{n+1}\}, \tag{3.3.1}$$

where n represents the index of the activities.

2. The set D of activity durations:

$$D = \{d_1, \ldots, d_n\}. \tag{3.3.2}$$

3. The set T of times in which the activities must be performed:

$$T = \{0, 1, \ldots, t\}, \qquad t \leq \bar{T}, \tag{3.3.3}$$

where $\bar{T}$ represents the deadline of the project.

4. Resources are represented as the set R of resources, with k total resources:

$$R = \{r_1, \ldots, r_k\}. \tag{3.3.4}$$

5. Resource requirements (k total resources) for each activity (n total activities) are shown as RQ:

$$RQ = \{\{rq_{11}, \ldots, rq_{k1}\}, \ldots, \{rq_{1n}, \ldots, rq_{kn}\}\}, \tag{3.3.5}$$

where rq_{kj} represents the resource requirement of activity j for resource k.

6. The set U of resource requirements for each period t:

$$U = \{u_1, \ldots, u_t\}. \tag{3.3.6}$$

7. The set C of costs associated to each resource:

$$C = \{c_1, \ldots, c_k\}, \tag{3.3.7}$$

where k is the total number of resources.

8. The set S_j denoting the sequence of starting times for each activity over the project time line T:

$$S_j = \{ss_0, \ldots, ss_{n+1}\}, \qquad es_j \leq ss_j \leq ls_j, \tag{3.3.8}$$

where es_j and ls_j denote respectively the earliest and latest possible start time for each activity.

9. The objective function can be represented as $c_i f[r_i(S,t)]$. This functions returns the cost c_i of resource r_i multiplied by the associated resource consumption of resource r_i during time t for feasible schedule S, for all resources k. The function f represents the specific objective function being considered (relating to the various RLP models), the options of which are presented in subsection 3.3.2.

The model for addressing the resource levelling problem (for an entire project) is then as follows:

$$\min \sum_{i=1}^{k} c_i f[r_i(S,t)] \tag{3.3.9}$$

subject to:

Deadline condition

$$ss_{n+1} \leq \bar{T}, \qquad and \tag{3.3.10}$$

Activity Precedence condition

$$ss_i + d_i + \gamma_{ij} \leq ss_j, \qquad \forall (i,j) \in A, \tag{3.3.11}$$

where γ_{ij} is the total slack between i and j (amount of time between the two activities).

3.3.2 Objective Functions for the Resource Levelling Problem. In the RLP model presented in subsection 3.3.1 the objective function has been represented as $c_i \cdot f[r_i(S,t)]$. This is to show that the objective function may be considered as the product of the cost of each resource, and some function which returns the consumption of each resource r_i for each time t of a given schedule S. Unlike RCPSP, which has a single objective across all problem types - minimize makespan - RLP has many possible objective functions. In Damci and Polat (2014) a review of the objective functions for RLP is presented. Objective functions are mostly concerned with minimizing fluctuations in resource consumption from time period to time period. The results of the study in Damci and Polat (2014) is that nine objective functions have been identified in the literature for RLP. More recently, in Ponz-Tienda et al. (2017) some additional objective functions have been proposed. We will now present a summary of these findings.

The most commonly occuring objective function for RLP looks to minimize the variance of resource consumptions by minimizing sample variance or mean square error over the time

horizon of the schedule. In Ponz-Tienda et al. (2017) this is referred to as the Minimum Squares Optimization or Sum of Squares Optimization (SSQR) method. It may be presented as follows:

$$\min \sum_{i=1}^{k} c_i \cdot \sum_{t=1}^{\bar{T}} \hat{u}_{it}^2 \tag{3.3.12}$$

where c_i is the cost of each resource and $\hat{u}_{it}$ the utilization. The optimization of the SSQR objective function will result in creating a schedule which sees completely flat resource utilization histogram where the sampling variance is minimized.

The specific objective functions for RLP as studied in Damci and Polat (2014) are as follows:

1. Minimization of sum of absolute deviations in daily resource usage:

$$\min \sum_{i=1}^{\bar{T}} |Rdev_i| \tag{3.3.13}$$

   where $Rdev_i$ is the deviation between resource utilization from time period i to time period $i+1$.

2. Minimization of the sum of only the increases in resource utilization from one time period to the next:

$$\min \sum_{i=1}^{\bar{T}} |Rinc_i| \tag{3.3.14}$$

   where $Rinc_i$ is the increase in resource utilization from time period i to time period $i+1$.

3. Minimization of the sum of absolute deviations between resource utilization in each time period and the overall average resource utilization:

$$\min \sum_{i=1}^{\bar{T}} |R_i - A_{rr}| \tag{3.3.15}$$

   where R_i is the resource requirement at time period i and A_{rr} is the average resource utilization.

4. Minimization of the maximum resource utilization:

$$\min[\max(R_i)] \tag{3.3.16}$$

   where R_i is the resource consumption at time period i.

5. Minimization of the maximum deviation in daily resource utilization:

$$\min[\max|Rdev_i|]. \tag{3.3.17}$$

6. Minimization of the maximum absolute deviation between resource utilization in each time period and the overall average resource utilization:

$$\min[max|R_i - A_{rr}|]. \tag{3.3.18}$$

7. Minimization of the sum of the squares of resource utilization per period:

$$\min \sum_{i=1}^{\bar{T}} R_i^2. \tag{3.3.19}$$

8. Minimization of the sum of the squares of deviations in resource utilization per period:

$$\min \sum_{i=1}^{\bar{T}} Rdev_i^2. \tag{3.3.20}$$

9. Minimization of the sum of the squares of deviation between resource utilization in each time period and the overall average resource utilization:

$$\min \sum_{i=1}^{\bar{T}} (R_i - A_{rr})^2. \tag{3.3.21}$$

The objective functions offer different mechanisms to deal with the requirement of smoothing the resource consumption profiles over the planning / scheduling period of the project.

Further to the above objective functions, additional objective function metrics are proposed in El-Rayes and Jun (2009) as mechanisms to minimize the impact of resource fluctuations in practical scheduling scenarios. These objective function models begin to build a foundation which attempts to address the practicalities of scheduling when the resources are workforce or labour related. In such situations (similarly to the real world problem studies in this dissertation) having idle workers is highly undesirable. The following objective function models are proposed in El-Rayes and Jun (2009):

1. Resource Idle Days (RID) metric quantifies the total number of idle and non-productive days which are a result of fluctuations in resource utilization levels. In addition to the Resource Idle Days objective function, El-Rayes and Jun (2009) add a Maximum Resource Demand (MRD) value which results in a multi-objective function model, which is adjustable by inputting weighting values at the time of execution. The Resource Idle Days and Maximum Resource Demand (RID-MRD) model is as follows:

$$RID = \sum_{i=1}^{k} c_i \cdot \sum_{t=1}^{\bar{T}} [min\{max(u_1, u_2, \ldots, u_t), max(u_t, \ldots, u_{\bar{T}-1}, u_{\bar{T}})\} - u_t] \quad (3.3.22)$$

$$MRD = \sum_{i=1}^{k} c_i \cdot max(u_1, u_2, \ldots, u_{\bar{T}-1}, u_{\bar{T}}) \quad (3.3.23)$$

$$RID - MRD = W_1 \cdot RID + W_2 \cdot MRD \quad (3.3.24)$$

where W_1 and W_2 are user defined weights input at execution times to establish the relative importance for RID and MRD, respectively. These weights would be established by setting up simulations and assessing the outcomes using various values. After a few trials, the user would be able to estimate the relative values of the weights.

2. The Release and Re-Hire (RRH) model was also introduced by El-Rayes and Jun (2009). This model attempts to cater for situations where labour resources may be adjusted temporarily when not needed, and brought in again at a later stage when the activities require additional resources in order to be processed. The Maximum Resource Demand metric as in Equation 3.3.23 is included once again to arrive at a configurable multi-objective function model:

$$RRH = \left[\frac{1}{2}\left(u_1 + u_T - \sum_{t=1}^{\bar{T}-1} |u_t - u_{t+1}|\right) - MRD\right], \quad (3.3.25)$$

$$RRH - MRD = W_1 \cdot RRH + W_2 \cdot MRD, \quad (3.3.26)$$

where W_1 and W_2 are weightings as described for Equation 3.3.24.

Additional objective function models for RLP have been introduced by Florez et al. (2009) in order to account for social objectives relating to the workforce, in addiction to the usual core objectives of cost and time in scheduling. The following two objective function models

are presented as models which aim to create a schedule such that the project maintains a stable workforce:

1. Sum of Differences of Consecutive Daily Resources (SDCDR) minimizes the sum of absolute variations of resources across the project planning timeline:

$$SDCDR = \sum_{i=1}^{k} C_i \cdot \left(u_1 + \sum_{t=1}^{\bar{T}-1} |u_t - u_{t+1}| + u_{\bar{T}} \right). \tag{3.3.27}$$

2. Sum of Squares of Differences of Consecutive Daily Resources (SSDCDR) utilizes the sum of squares computational measure as opposed to absolute measure in an attempt to create a more level / rectangular profile:

$$SSDCDR = \sum_{i=1}^{k} C_i \cdot \left(u_1^2 + \sum_{t=1}^{\bar{T}-1} (u_t - u_{t+1})^2 + u_{\bar{T}}^2 \right). \tag{3.3.28}$$

The above is a review of the objective functions used in RLP models in order to achieve various resource utilization profiles, depending on the goal of the decision maker and the requirements of the project. There is no one objective function capable of satisfying the requirements of all RLP problems and as such the RLP may be considered as a non-regular class of problems. The application of RLP to the real world problem addressed in this paper will be presented in Chapter 4.

3.4 Machine Scheduling Problems and the Flow Shop Problem

Machine scheduling problems form a well-known group of scheduling problems which have been extensively studied. Many of the advances made in the field of scheduling have occurred through the study of machine scheduling problems. At the simplest level, these problems are concerned with assigning jobs to machines such that they are completed as soon as possible while not exceeding any capacity or resource constraints. Machine scheduling problems present themselves as special cases of RCPSP where the resources correspond to machines.

The basic machine scheduling problems can be extended to a class of problem known as shop scheduling problems. In shop scheduling problems, the jobs (activities) consist of several operations which need to be processed in some order on several machines. The extensions and variations to RCPSP as discussed in a previous section are all still applicable to machine scheduling problems.

3.4.1 General shop scheduling. The formulation of the general case of this problem is taken from Brucker and Knust (2006). In general shop scheduling problems, there are jobs $j = 1, \ldots, n$ and m machines $M_1, \ldots, M_m$. Job j is made up of n_j **operations** $O_{1j}, \ldots, O_{n_j j}$. Operations of the same job may not be processed simultaneously and a machine can process at most one operation at any time. Operation O_{ij} must be processed for d_{ij} time units on a dedicated machine $\mu_{ij} \in \{M_1, \ldots, M_m\}$. Precedence constraints may exist between operations, depending on the nature of the problem. The general shop scheduling problem may be presented as an RCPSP as follows: we define $k = m + n$ renewable resources and $R_k = 1$ for $k = 1, \ldots, m + n$. The resources $k = 1, \ldots, m$ represent the machines and $k = m + 1, \ldots, m + n$ are used to model the different operations of job j and the fact that they may not be processed simultaneously. The latter set of resources can be denoted as $m + j$ where $j = 1, \ldots, n$. The activities to be scheduled are the $\sum_{j=1}^{n} n_j$ operations O_{ij}. Each operation O_{ij} requires one unit of machine resource μ_{ij} and one unit of job resource $m + j$. Special cases of the general job shop problem are the job shop, flow shop and open shop problems. We now introduce the flow shop problem, as there are elements of flow shop scheduling which will be useful in understanding how to formulate the real world problem.

3.4.2 Flow shop scheduling. A flow shop problem is a special case of the general job shop problem with $n_j = m$ operations for $j = 1, \ldots, n$ and $\mu_{ij} = M_i$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$. This means that an operation O_{ij} must be processed on machine M_i and on no other machine. In a permutation flow shop problem, jobs have to be processed in the same order on all machines. A permutation flow shop scenario is presented in Figure 3.2.

3.4.3 Parallel machine scheduling. A further generalization of the machine scheduling problems, is the case where multi-purpose machines could potentially handle many different jobs. In this situation, each job j is associated with a subset of machines $\mu_j \subseteq \{M_1, \ldots, M_m\}$.

Job j can be processed on any machine in this set. If job j is processed on machine M_k then the processing time is d_{jk}. This problem can be formulated as a multi-mode RCPSP with m renewable resources where each job j has μ_j modes corresponding to the machines on which j may be processed. The machine scheduling problem may be extended to a flow shop scheduling problem with parallel, non uniform machines. In this case, rather than the job j being associated with a certain subset of the machines, it is the operations $O_{1j}, \ldots, O_{n_j j}$ which are associated with a subset of the machines.

3.4.4 Multi-capacitative machines. In general shop scheduling, certain assumptions are made about the nature in which the machines perform their operation, or process the various jobs. Once an operation is started, it is processed without interruption and in addition to this, machines may not perform more than one operation at a time. In order to solve our problem, we need a scheduling algorithm which allows for multiple operations to be performed simultaneously on a single machine. We discuss this requirement in more detail in the next section. Nuitjen and Aarts (1996) introduce the concept of a multi capacitated job shop scheduling problem as a generalisation of the job shop problem. They allow each machine to process several operations simultaneously, provided that the capacity of the machine is not exceeded by the total size of the operations it is performing. There does not seem to be any further study of this multi-capacitative feature in literature regarding RCPSP.

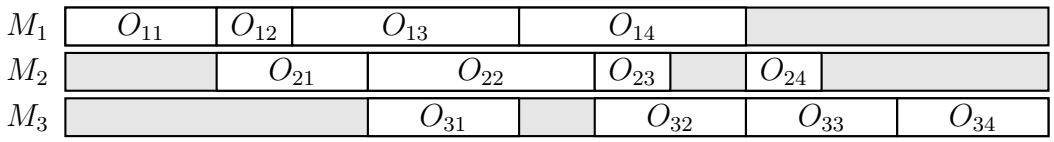

Figure 3.2: Feasible schedule for a permutation flow shop problem with $n = 4$ jobs and $m = 3$ machines.

3.5 Summary and Research Objectives

In this section, we have presented three main classes of scheduling problems: Resource Constrained Project Scheduling Problems, Resource Levelling Problems and Machine Scheduling Problems. We have also presented certain extensions, features and objective function models for these problems. At this point it is now possible to contextualise the research and work required in order to address the goal of this dissertation, namely solving the real world problem as presented in Chapter 2.

3.5.1 Research objectives. The objectives of this research are as follows:

- Review the research on scheduling techniques and define the real world problem in terms of the available research.
- Develop a mathematical formulation for a new model, incorporating aspects of RCPSP and RLP to address the real world problem.
- Use optimization techniques to solve the above model.
- Create a practical implementation of the scheduling algorithm so that the real world problem may be solved.
- Implement a practical solution to the industrial problem in a live business environment and observe the outcome.

3.5.2 Research methodology.

- Create a model of the real world problem in terms of existing RCPSP and RLP frameworks and extensions.
- Make use of the Simulated Annealing computational optimization method.
- Data mining - extensive data mining and data analysis is undertaken in order to provide data for the modelling of the industrial problem. Certain models need to be developed as part of a pre - processing work stream in order to compute inputs for resource capacity and resource utilization. These models are supplementary and do not form part of the main component of my research.

- Solve the problem: Create a scheduling algorithm which dynamically assigns collection activities to batches of accounts according to available capacity, risk and deadlines.

We have begun to discuss and demonstrate certain features of the real world problem that are similar to aspects of the various scheduling models discussed in this chapter. From the next chapter and going forward we will begin to focus on how to specifically address the requirements of the real world problem. We will begin to specifically define the concepts of 'activities' and schedules in the context of our problem. In Chapter 4 we will present the real world problem in more detail. We will discuss the problem features making reference to the literature review conducted in this chapter and provide a formulation of a model which can be used to address the real world problem.

Chapter 4

Problem Characterisation and Research Findings

In Chapter 3 we have presented the concepts relating to the field of scheduling. In this chapter, we will now frame the real world problem in terms of these concepts and discuss the features of the problem in detail, demonstrating which features may be addressed with existing models and which features will require a new model in order to be addressed. The various extensions and objective function models will be considered and their applicability to the real world problem assessed. We will demonstrate that the real world problem has unique features not covered in literature and additional extensions and objective function models will need to be developed. We will also present the new objective function we have developed to extend the RLP class of problems in order to address the requirements of the real world problem studied in this dissertation.

4.1 Scheduling models and application to the real world problem

The problem, as presented earlier in Chapter 2, can be contextualized in the framework of the RCPSP and RLP models from Chapter 3. Scheduling problems consider inputs in terms

of:

- The Project: An enterprise or endeavour planned to achieve a particular outcome.
- Activities: The actionable elements of the project which need to be performed in order to achieve the project outcome.
- Resources: Tangible or Intangible materials which will be consumed as activities are performed.
- Batches: A quantity of project input elements against which activities need to be performed at one time.

For the real world problem, we define the following:

- The Project: Execute collections activities against non-paying accounts in order to collect outstanding monies owing the organisation, in such a fashion that call centre service level is never exceeded.
- Activities: Collection actions taken against the delinquent accounts (according to the collections strategy of the organisation subject to precedence and resource constraints) are the project activities. In the implementation of our model, we use the collection strategies (defined processes involving the specific collection actions) applied against batches of delinquent accounts as the elements requiring scheduling. This will be explained in more detail in subsection 5.2.6.
- Resources: The call centre capacity will be the daily renewable resource considered for this project. It may be possible to include another 'dummy' secondary resource to represent risk of bad debt. This dummy resource will be necessary in the mathematical model so that when an optimization algorithm is implemented to optimize the model, the collection strategies on high risk customers are carried out ahead of those applied to the low risk customers, even if it results in a longer makespan. This feature will be addressed in the Chapter 8 where future work is described.
- Batches: A group or cohort of accounts entering the collections environment on a given day can be considered as a batch. The scheduling requirement will then be to schedule

the start date of collections strategies for each batch of accounts such that the resource consumption constraints / objectives are achieved.

4.1.1 Terminology and Concepts. We briefly define some terminology which will be important in fully understanding our problem. These will be reinforced later as we begin to refine and specify additional details needed to create a model for this problem.

- **Time point**: The term 'time point' relates to a specific time point t, usually with reference to the time when a specific schedule is created. In a practical sense, this would typically be before the start of a working day (say for example at 04:00 in the morning) when no other factors could occur which influence or change the input parameters.
- **Time period**: The term 'time period' relates to the interval over which collections actions or other activities are performed, responses are received and therefore capacity is consumed. This is the entire period from time point t until the beginning of the next time period $t+1$, i.e. the interval $[t, t+1)$. A scheduling phase is made up of several time periods.
- **Scheduling phase**: The scheduling phase refers to the set of time periods (and time points) over which all previous and future scheduled activities will have an impact on resource consumptions, and for which an optimal schedule needs to be created.

4.1.2 A conceptual formulation of the real world problem in terms of RCPSP and RLP. Consider a simple case of the real world problem: Four batches (as defined in Chapter 3) of accounts which need to undergo collections treatment.

In the real world problem, accounts enter into the collections environment every day. These accounts need to be allocated to batches. The batches are the jobs j which need to processed. The collections activities: SMS1, SMS2, SL are the operations which need to be performed on every batch and are analogous to the machines in FSP. This will need to be extended to a multiple mode case, since each operation could be processed in one of several modes. For example: An SMS activity could be performed such that its duration is two, three, four or five time units. Each of these options will be represented as a distinct mode.

The multiple project approach could be used to cater for the characteristic that each segment of the customer base has a unique resource consumption profile. For example, an SMS activity performed on a batch of accounts from the very low-risk segment will result in a high number of responses, while the same activity performed on the very high-risk segment will result in very few responses.

We present a conceptual approach which could be adopted to formulate the real world problem:

- Accounts requiring collections treatment are segmented according to their risk/behavioural profile.
- Each segment becomes one project in a multi-project scheduling problem
- Within each distinct project, the accounts are distributed into batches. These batches are the jobs which will be scheduled.
- Batches need to be scheduled such that makespan is minimized subject to capacity constraints and with the following features:
  - A batch of accounts may not need all operations to be processed. Once a response is received from one operation, further operations are unnecessary. The batches in each segment (project) will have a certain likelihood of requiring each operation to be considered processed. This characteristic could be included in the modelling of the problem in order to ensure that resources are fully utilized.
  - Variable batch size. The batches will be adjusted during every time period and the scheduling of machines according to capacity. If new information is received, indicating that additional capacity is available, batch sizes need to be adjusted accordingly.

The diagram in Figure 4.1 provides a visual summary of the above approach.

4.1.3 Notable differences between the real world problem and RCPSP / RLP studied in literature. In this dissertation, we have presented a real world problem and also discussed the classification of the problem and its similarities to RCPSP / RLP (and

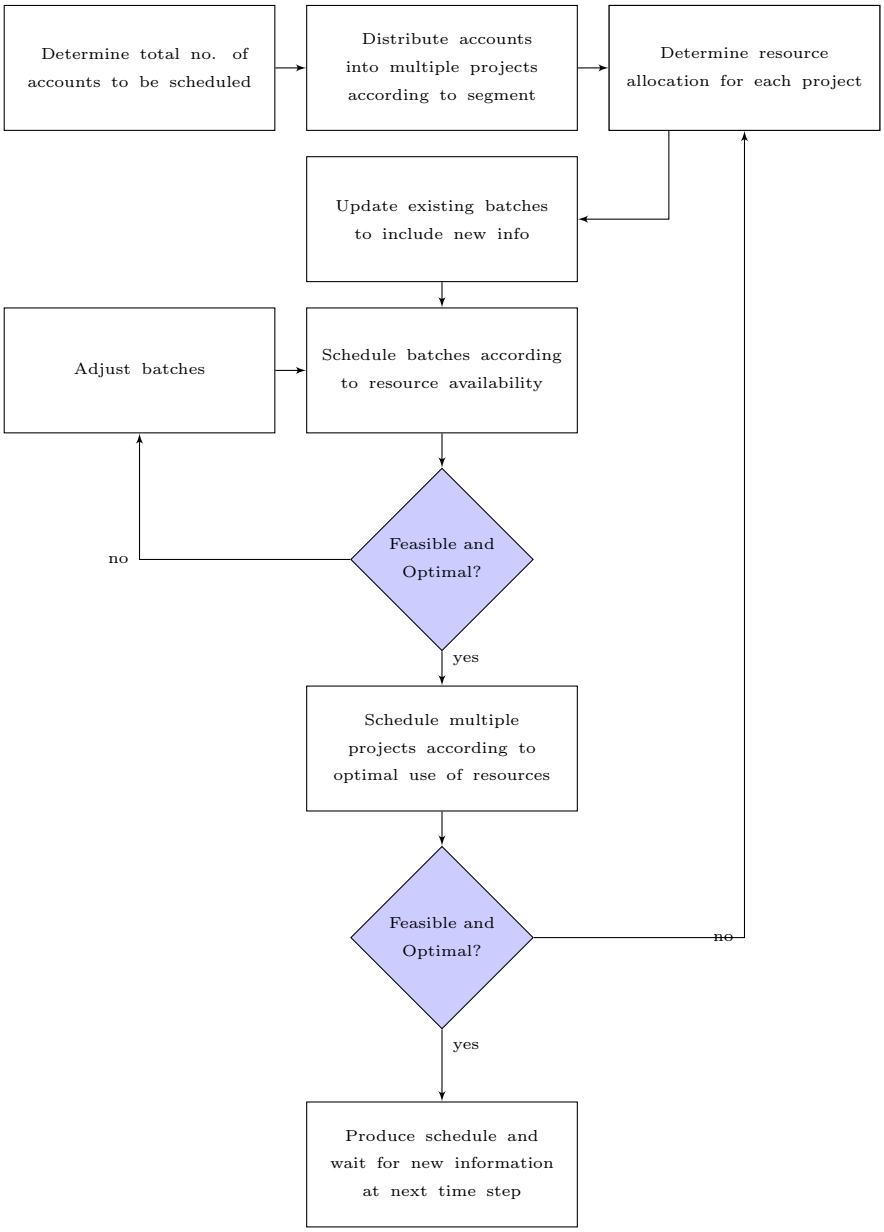

Figure 4.1: Schematic of conceptual algorithm for solving the real world scheduling problem.

extensions) which have been studied in literature. The industrial problem, and generic class of problems which it represents, seems to be a novel problem and does not appear in any of the literature that was reviewed. In this section, we highlight the aspects of the problem which are different to RCPSP / RLP in literature.

1. In standard RCPSP models, it is assumed that full knowledge of the number of activities to be scheduled is available. In our problem, no such *a priori* information exists, the number of activities requiring scheduling will change during every time period (e.g. at the beginning of each day). An RCPSP model which is capable of reactively rescheduling existing jobs (according to new information received) will need to be developed to cater for the features of our problem.

2. In a typical RCPSP, once a project or activity has begun, it cannot be adjusted. An extension to the RCPSP needs to be developed to cater for the scenario where activities can be removed from a job after already being scheduled. This is to cater for the feature of the real world problem where customers will no longer require collections treatment (due to returning to good standing). We propose to address this characteristic by two separate approaches: During the initial scheduling phase, a model could be created to calculate the chance of each job requiring the various operations to be performed against it. Alternatively, a worst case scenario approach could be adopted, where all accounts are scheduled with unadjusted resource consumption profiles and then during the daily rescheduling phase, the schedule is updated to account for all accounts that have exited the system due to self cure. In the practical implementation of our solution, we cater for this feature by taking the self cure into account in the resource consumption modelling as described in the first approach above.

3. In RCPSP, the goal is to schedule activities subject to available resources while minimizing makespan. In RLP the goal is to create a schedule such that resource consumption is levelled or nearly levelled over the project time horizon. In the real world problem, the resource constraint relates to call centre capacity. A satisfactory solution to the real world problem could be achieved by scheduling such that the capacity constraint is not violated. However, it would be better to ensure that all

available capacity was fully utilised at every time point. This requirement stems from the fact that call centre capacity incurs labour costs and having workers sitting idle is a waste of resources. We develop an extension to RCPSP / RLP where the FULL consumption of all renewable resources is targeted at each time period. This extension also ensures that any schedule created attempts to fully utilize all available resources in the time periods closest but ahead of (i.e. in the future) the current scheduling period. It is not necessary for full resource utilization to occur at a time far in the future of the scheduling time. This ensures that any schedule can act as an initial optimal schedule for the next period, where an optimal scheduling may be carried out again.

4. While the real world problem shares similarity with Fuzzy Resource Levelling Problems as discussed in Bagheri and Hessam Mahmoudi (2012), a key differentiation is that at each scheduling phase, it is not necessary to create a schedule that uniformly utilizes resources over the make-span of the project. Instead, it is necessary at that scheduling phase to create a schedule such that resources are fully utilized within the next few time periods (critical time window) with a relaxation on the resource utilization further along the time line. Thus a schedule is created in such a fashion that the call center will always have optimal utilization within the next few days while insuring there is capacity to handle the additional volumes which will enter the system at later dates.

5. In FSP and RCPSP, machines have their own resources. In the real world problem, all machines share resources.

6. Typically, rescheduling is considered to be a contingency. Once a schedule has been developed, rescheduling will occur if there are some changes required due to unforeseen events. In order to solve the real world problem, we propose a model where rescheduling is a key feature. The model will be built around the concept of rescheduling at every time step.

4.2 The New Model and Features to Address Real World Problem

We have shown in this chapter that the current state of RCPSP and RLP literature will not adequately satisfy all the features of the real world problem. In this chapter we will summarize the specific features that characterise the real world problem, we will then propose a new model to address the requirements of our problem and the broader group of problems which are introduced by the existence of this particular problem.

4.2.1 Specific features of the real world problem. A review of the features of the specific problem being studied (based on the information in previous sections) will now be presented. The features will be discussed in general terms, with specific reference made to the context of the real world problem being addressed in this dissertation. The features are as follows:

- There exists some environment with known availability of renewable resources at each time period. The resource availability is not constant and fluctuates from time period to time period according to a known distribution. The resources need to be fully utilized at every time period. (In the real world problem, the resources are the call centre capacity).
- Every time period, there are new activities entering the system which will need to be scheduled, in addition to the activities which have already been scheduled in a schedule from the previous time period. (Reactive rescheduling of new batches of accounts entering the collections process on a daily basis).
- There is no terminal state for the scheduling system. Unlike traditional scheduling environments, there is no end condition. The scheduling model needs to cater for this feature by ensuring optimal resource consumption on an ongoing basis.
- There is a risk in delaying the start of activities, which differs from activity to activity. (Delaying collection actions against accounts may result in increased bad debt exposure. This will be more significant for high risk accounts.) Risk could be modelled

as an additional resource; delaying the start of collections actions on an account will increase the consumption of this resource. The model which will be introduced only considers a single resource, the dummy resource feature will be addressed in future research.

- Resource Consumption Vary With Time - An activity will consume a variable amount of renewable resource per time period over the duration of its execution window. (Collections actions taken on accounts will result in responses over a number of days, with the intensity of the response varying from day to day - usually decreasing).

4.2.2 Critical Time Window Continual Rescheduling Model. Based on the criteria presented in the previous subsection, we define the class of problem represented by the real world problem as a sub-class of the Resource Levelling class of problems. We propose a new model to address the features and requirements of this class of problems which has two significant features:

- Critical Time Window Resource Levelling.
- Continual Rescheduling.

We call the model the Critical Time Window Continual Rescheduling Model. The model addresses the challenges posed by the real world problem in the following manner: The lack of a terminal state for the real world problem creates a challenge. Traditional scheduling models require some end event or deadline to be met in order to provide either an objective function or constraint input into the scheduling model. In order to address this feature of the real world problem, where new accounts are entering into the collections environment on a daily basis, we propose a continual rescheduling approach.

In order to build up this model, we will introduce the concept of four related sets, CT, L_+, L_- and H_t such that:

$$CT \subseteq L_+ \subset H_t \tag{4.2.1}$$

and

$$L_- \subset H_t \tag{4.2.2}$$

where CT refers to the *critical time window*, L_+ refers to the *optimization time-line*, L_- is the *relevant history time-line* and H_t refers to the *relevant time horizon* centred around time point t. Why these time sets are important and how they are applied in the formulation of our proposed model will be described in detail soon in this section.

We begin by discussing the concept as to how continual re/scheduling will be addressed:

- The scheduling model will be optimized at every time point, not only once at the 'beginning'.
- The optimization time-line is defined as

$$L_+ = \{t, t+1, \ldots, t+\tau\}, \tag{4.2.3}$$

  the time period over which all future activities may be scheduled and also the time period during which we wish to control the consumption of resources.

- The relevant history time-line is

$$L_- = \{t-\tau, t-\tau+1, \ldots, t-1\}, \tag{4.2.4}$$

  the period during which activities have been previously performed and still have an impact on resources in current and future time periods. This means that activities which were performed during, say, time interval $[t-\tau, t-\tau+1)$ will still have an impact on resources during (at minimum) time period $[t, t+1)$ and there may be an impact on resources in future time periods depending on what was scheduled to begin during the time periods following $[t-\tau, t-\tau+1)$.

- The scheduling phase will refer to the relevant time horizon relating to the current time point t so that

$$H_t = L_+ \cup L_- \tag{4.2.5}$$

  Conceptually, the scheduling phase is entered around the current time point t. A relevant time horizon H_t will be considered at each scheduling phase. The relevant time horizon will be defined to be the minimum number of time periods τ such that if the current scheduling phase is being executed at time t then the resource consumption

requirements (responses) of activities that were initially performed during $[t-\tau-1, t-\tau)$ are zero at $[t, t+1)$ and for all future time periods. By symmetry, the resource requirements of all activities scheduled during the current scheduling phase at period $[t, t+1)$ will then also be zero for all future time periods starting at $t+(\tau+1)$. The *relevant time horizon* is thus defined as:

$$H_t = \{t-\tau, t-(\tau-1), \ldots, t, \ldots, t+(\tau-1), t+\tau\}^{1}. \tag{4.2.6}$$

We demonstrate a relevant time horizon by example: Suppose that the time, ϕ, of the last observed response to any collections action is $\phi = 15$ days after the start of the collections strategy and also the length of the scheduling horizon is $\tau = 15$ periods. This means that from day 16 onwards after a collections action has been performed, no further responses are observed. Thus for scheduling purposes, when a new schedule is being created, only the responses from accounts that had collections actions performed against them up to 15 days ago need to be considered, and the response profile of accounts being acted on today only needs to be considered for up to 15 days in the future. The relevant time horizon is 30 days and the response time-line and scheduling time-horizon are of equal length.

Note that the relevant time horizon must be at minimum the amount of time periods necessary to plan for the resource consumption requirements of any activity that once initiated, would still have an impact on resources during t. The relevant time horizon could also be longer than this defined minimum though. Say for example $\tau = 15$ and $\phi = 13$. This could be done to allow, for example, the scheduling system to schedule the start of certain activities far into the future (i.e. delay the start). This could be done in certain cases where the activity load is so significant that it is not possible to create a schedule (over a shorter time horizon) that still does not violate the capacity availability. This would be something that is expert knowledge / judgement based and can be adjusted depending on the nature of the problem at hand.

[1] Equation 4.2.6 shows the discrete time points in the continuous interval $[t-\tau, t+\tau]$ over which the various actions of scheduling, receiving responses, updating data etc. are performed

- During each scheduling phase, the response profile of all activities performed in the past, which still have resource consumption impact in the current relevant time horizon will be taken into account. All activities which were scheduled during previous scheduling phases but have not yet been executed (start times occurring in $\{t+1,\ldots,t+\tau\}$ where t is time of currently active scheduling phase) will be available for rescheduling along with new activities that entered into the system at time t.
- A new schedule is then created during each scheduling phase such that the scheduling criteria (presented below as critical time window feature) are met.
- This procedure is executed on an ongoing basis - a Continual Scheduling Model. The relevant time horizon becomes a rolling scheduling window.

In terms of the relevant time horizon terminology, consider a schedule that is created at time point $t-1$, relating to scheduling phase H_{t-1}.

$$H_t = \{t-\tau-1, t-\tau, \ldots, t-1, t, \ldots, t+\tau, t+\tau-1\}. \tag{4.2.7}$$

Scheduling phase H_{t-1} is centred at $t-1$, while the following scheduling phase H_t is centred at t.

In phase H_{t-1} at time point $t-1$, 'scheduling' means that the start date of activities (i.e. collection strategies assigned to the batches of accounts) is set to be $t-1$. The execution of these strategies occurs over multiple time periods, which results in an impact on resources in periods $[t-1,t);[t,t+1);\ldots;[t+\tau-2,t+\tau-1)$. Now, due to the continual rescheduling approach, we do scheduling (i.e. execute the optimization algorithm - Simulated Annealing - until completion to obtain an optimal schedule at a given time point) the next day at time t for the associated phase H_t (Equation 4.2.6). At time t of phase H_t the activities to be performed by the workflow system during period $[t-1,t)$ have now been completed and as such, become part of the unalterable 'history' that any schedule, created for phase H_t, must take into account. Conversely, the activities from the optimal schedule created for phase H_{t-1} that were set to begin during periods $[t,t+1),\ldots,[t+\tau-2,t+\tau-1)$ have not yet been carried out. Now, rescheduling at time point t of phase H_t means that one needs to take into account:

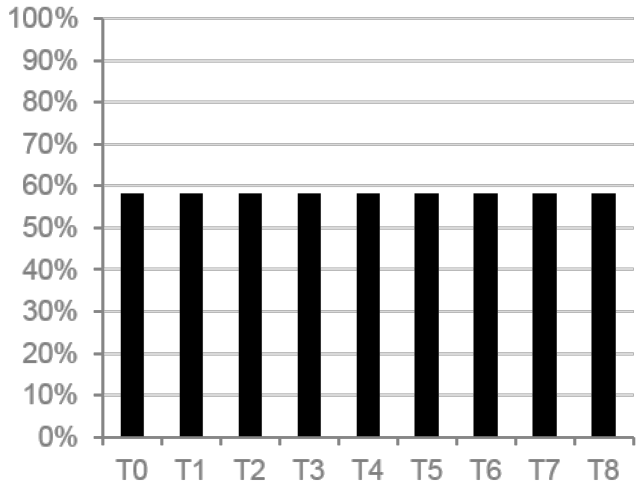

Figure 4.2: Resource utilization over 8 time periods in a resource levelled schedule

- The old (historical) information of H_{t-1} for calculating impact on resources in the period $[t, t+\tau-1)$.
- The new information (eg. number of new accounts entering the system) accumulated at t over the period $(t-1, t]$ for phase H_t.
- The existing information relating to activities that were scheduled to begin over $[t, t+\tau-1)$ which are now free to be rescheduled based on new information.

This concept is demonstrated practically in section 7.1.

We now discuss the second feature which relates to the resource consumption profile. As a result of the continual scheduling approach discussed above, traditional resource levelling objectives will not be applicable. By creating a schedule such that resource consumption is levelled over the relevant time horizon, a situation will likely emerge that there will be insufficient resource availability at certain times, when new accounts entering the collection system far exceed the capacity of the system over the relevant time horizon. The regular resource levelling problem seeks to create a schedule such that the makespan in minimised or some deadline is achieved and the utilization of resources is consistent at every time during the project time line. Figure 4.2 demonstrates this concept.

In order to address the features of the real world problem, we propose a Critical Time

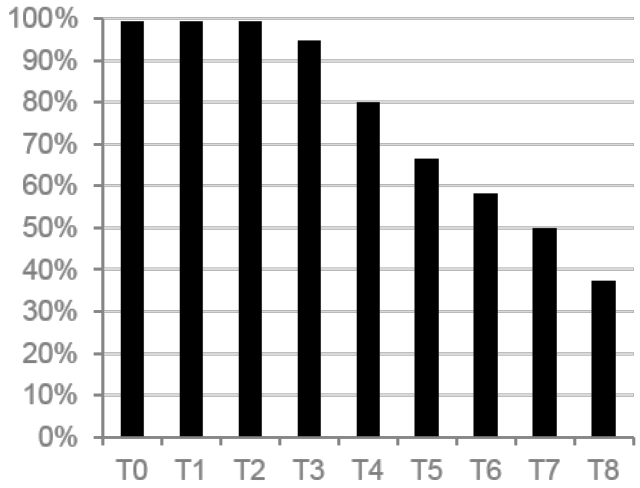

Figure 4.3: Resource utilization over 8 time periods in a CTW resource leveled schedule where the CTW is $\kappa = 3$ periods.

Window resource levelling approach, in conjunction with the continual scheduling approach. We define a critical time window as

$$CT = \{t, t+1, \ldots, t+\kappa\} \tag{4.2.8}$$

where t is the time of current scheduling phase and κ ($\kappa \leq \tau$) is the number of time periods defined by analysis or expert knowledge. The objective is to ensure that during each scheduling phase, a feasible and optimal schedule is created such that the resource (call centre capacity) is fully utilised within the next few time periods (e.g. the next κ periods). Figure 4.3 demonstrates this concept for a resource where the critical time period is 3 periods and the scheduling phase is at time $T0$.

With the critical time window approach we propose a method where scheduling is carried out at the beginning of each day i.e. at time point t, considering information over the period $[t-\tau, t)$. The objective is then to schedule the start date of the collections strategies (which need to be applied to the arrears accounts) over the period $[t, t+\tau)$ i.e. τ days, with the reinforcement of full (or some other desired control condition) utilization of resources over the period $[t, t+\kappa)$. The concept of a critical time window may also be considered by way of explanation: In the collections call centre environment, the schedule of accounts to be released into the collections strategies is created such that the call centre capacity is

fully utilised over the next 3 days. This means that regardless of what new information is received during the next time period (the following day over the period $(t, t+1]$) the call centre efficiency will be maximised by the current schedule. If, on the one extreme, there are no new accounts that enter the system on the next time period, then the schedule will be created using the activities which were scheduled to be performed starting at time point $t+1$ during the previous scheduling phase but not yet performed. Once again the structure will be such that the utilisation is maximised over the next 3 days. On the other extreme, if there is a large number of accounts entering the system, the model will still seek to maximize the resource utilisation in the periods closest to the scheduling phase, and have a more relaxed requirement for resource utilization further away. This is an important feature as the collections strategies have differing resource consumption profiles and depending on the number of accounts in each strategy, the resulting response profiles (and the related resource consumptions) will overlap and in a certain way to create the total resource demand. For some collections strategies, the demand peaks later on in the collections cycle (for example the high risk accounts treatment strategy) whereas other collections strategies result in a more immediate response (for example the low risk strategies). In creating a critical time window model, the optimization algorithm will select the best combination of accounts to schedule against the various collections strategies such that the call centre capacity is fully utilised but not exceeded during the critical time window.

We recall that the optimization time-line L_+ is defined where L_+ is the subset of H_t containing all time periods from the current point t up until the last future time point in the rolling time horizon H_t. That is:

$$L_+ = \cup_{i=1}^{\tau}[t + (i-1), t + i) \tag{4.2.9}$$

or

$$L_+ = \{t, t+1, \ldots, t+\tau\}. \tag{4.2.10}$$

L_+ is the interval over which the optimization is performed.

In summary: The rolling schedule horizon H_t is the time-line over which all activities need to be considered where activities may still have an impact on resources during the current

and future time periods i.e. in time-line L_+. The critical time window CT is the set of time periods in L_+ where there is particular focus on ensuring the complete (or some other condition) utilization of resources more strictly than during the remaining periods in L_+.

By combining the critical time window approach with the rolling scheduling window of the continual scheduling model described above, we propose that we may achieve a result whereby the call centre capacity is always fully utilised in our real world problem.

4.2.3 The objective function for Critical Time Window Continual Rescheduling Model. Our formulation of the critical time window resource levelling problem thus introduces a new objective function:

Minimize

$$\sum_{k \in K} \sum_{t \in L_+} \omega(t).(r_k(S,t) - R_k(t))^2, \tag{4.2.11}$$

where $\omega(t)$ is a vector of weights corresponding to the importance of each time window at time t (enforce critical time window criteria). $r_k(S,t)$ is the resource consumption of resource k at time t across all activities for schedule S. $R_k(t)$ is the resource availability of resource k at time t, for all time periods in rolling schedule horizon H_t.

The optimization of the objective function in Equation 4.2.11 is subject to:

Deadline condition

$$ss_{n+1} \leq \bar{T} \tag{4.2.12}$$

Activity Precedence condition

$$ss_i + d_i + \gamma_{ij} \leq ss_j \qquad \forall (i,j) \in A \tag{4.2.13}$$

where γ_{ij} is the total slack between i and j.

The conditions and components in the model relate to the RLP model presented in subsection 3.3.1.

4.3 Summary

In this chapter, we have now shown that the real world problem can be formulated mathematically as a scheduling problem. We have also shown that the real world problem cannot be completely modelled by the RLP from literature and we thus introduced the new continual scheduling method and critical time window objective function RLP model. In the next chapter we will discuss the other elements necessary before the real world problem can be solved optimally.

Chapter 5

Implementation Approach

In this chapter, we will describe how the information and concepts from the previous chapters are consolidated and used to address our problem. We will begin the chapter with an overall summary, pulling together the ideas and suggestions from previous chapters and describing the exact approach used to capture the nuances, and address the features of our real world problem. Following this, we will then go into the detail of each of these features, describing the exact process or steps which were taken.

5.1 Summary

In this section we provide a high level summary of the approach that was taken.

In Chapter 4 we described the industrial problem in detail, highlighting the features which would need to be addressed in any attempt to create an optimal solution to the problem. In Chapter 3 we described the approaches to scheduling that have been studied and began to explore the industrial problem in terms of these concepts. We now describe exactly how the problem will be dealt with:

- The collections actions will be consolidated into specific collections strategies (This can be thought of as a process comprising groups of collections actions set out in a

specific order). A more detailed explanation of this will be supplied in subsection 5.2.6. Effectively the notion of 'activity' precedence is no longer required for this model as the collections strategies become predefined processes which are not modified. There is also no deadline condition for this environment, the schedule will be created with the single purpose of controlling the resource consumption according to the criteria which have been detailed and will be discussed in Chapter 7.

- A scheduling system will be implemented in the call centre environment. This will be the technical implementation of the algorithm described in Chapter 7.
- The scheduling system is a stand-alone system which will perform the optimization procedure to obtain a feasible and optimal schedule based on information delivered from the work flow system and other sources.
- The workflow system is a separate system. This is an operational system which performs the collections strategies against the delinquent accounts. That is, the workflow system performs tasks in some predefined manner based on user defined parameters and settings.
- Information pertaining to the number of new accounts entering the system over the time period $[t-1, t)$ prior to the current time point t is delivered from the work-flow system to the scheduling system.
- The scheduling system will schedule the start dates of the collections strategies (which will be applied to the arrears accounts). In doing this, the scheduling algorithm will need to take into account collections strategies / processes started in the past which still have an impact on resources during current and future time periods, as well as the resource consumption profiles of the various collections strategies over time.
- The scheduling system will run before the start of each working day (time point t) to create a schedule based on the information pertaining to the current scheduling phase H_t. Again, the schedule provided will be a schedule of start dates of the various collections strategies. So for example, the work-flow system will have information that there are 10 new accounts requiring treatment, as well as 100 accounts that are

already in the system but have not yet begun collection treatment. Thus the total of 110 accounts for this particular collection strategy is then passed to the scheduling system, which will then (as part of the optimization process) calculate some optimal way of scheduling the start dates for that particular collections strategy against the 110 accounts over L_+.

- Information relating to the number of collections strategies which will need to be started (against arrears accounts) in the current time period $[t, t+1)$ will then be fed back into the work-flow system from the scheduling system.
- There is a flow of information between the scheduling and work-flow system. The work-flow system receives and controls information relating to the number of accounts entering the system, as well as the state of accounts currently undergoing collection treatments. Some of this information is passed to the scheduling system, which then in turn calculates a new schedule (the start date of the collections strategies). The number of collection strategies which need to be started during time period $[t, t+1)$ is then passed back to the work-flow system, which then begins those strategies.
- The actual work-flows relating to executing the various steps in the collections strategies will be handled by the existing work-flow system in the credit and risk environment.
- The work-flow system will automatically (based on expert knowledge / management defined control parameters) perform the various activities over the course of the day (i.e. the period $[t, t+1)$ for example). Practically this means that the automated system will perform the actions in some spread out way over the course of the day.

From this point onwards in the dissertation the notion of a scheduled 'activity' in the context of project scheduling for the real world problem should be understood to relate to the idea of collections strategies applied to arrears accounts.

5.2 Solution Approach

In this section, we describe the process that was followed in performing the study that resulted in this dissertation. The study arose as a result of a problem that was identified in an industrial setting. After an initial exploration of the problem, it was decided that the problem presented sufficient complexity and novel attributes and would be a non-trivial problem for an MSc research. Various components that are described below all relate to the work that was performed in developing a body of knowledge and other prerequisites which were necessary to allow the formulation and optimization of the mathematical model. We will be presenting how the research and solution process evolved for the problem. The solution refers to the all the elements necessary to address the requirements of the industrial problem, which include many aspects in addition to the core mathematical model and computational technique related to the underlying scheduling problem. The section contains detail of the following actions:

- Research and problem discussion - business problem identification.
- Data analysis and exploration.
  - Understanding call volumes and patterns.
  - Creating customer segments.
  - Mapping response rates.
- Problem Framework - identifying solution elements.
  - Data-driven collection strategies.
  - Workforce management.
  - Work-flow optimization (Optimization in the industrial engineer sense of performing tasks more efficiently).
- Implementation of solution.

5.2.1 Business problem identification. The Credit and Risk collections call center had an ongoing problem with inbound service levels, particularly during the periods at the beginning and end of the month. This was a well-known phenomenon and various methods had been trialled in order to address the problem.

Most customers use the last working day or first working day of the month as their preferred date for the direct debit of their bank account for the funds owing to service provider. It was widely assumed that the inbound calls were directly related to the debit order dates of the customer, (and not significantly influenced by the SMS's that were being sent to the customers). As such, the prevailing opinion of the management team was that there was limited opportunity to reduce the volume of inbound calls.

Any method of alleviating the strain on the call center would be as a result of improving customer behaviour (resulting in a reduction in the number of non payment events) or through workforce management solution such as employing extra temporary call center workers during the busy periods.

There were also some attempts made to control the volume of calls on a daily basis through manual SMS 'throttling'. This entailed staggering the release of SMS's to customers over the course of the day. For example, if 10,000 SMS messages were going to be sent, 2,000 would be sent first thing in the morning, then an hour later another 2,000 would be sent and so on until all SMS messages had been sent. Similar tactics were applied to the other collections activities too. This demonstrates that there was some understanding that SMS messages and collections actions would trigger a response from the customers.

The problem of increasing call volumes had been ongoing for many years. As the customer base of the service provider increased, so too would the number of customers who went into arrears each month. In order to address the issue and ensure that service levels could be met, the number of employees working on the call centre was constantly being increased. It reached a point, however, where a management decision was taken to limit the total number of employees to 100 call center agents. This was to ensure that operating costs were kept to reasonable levels.

In the year of 2009, the service level had been particularly bad for three consecutive months (between 55% - 65%). In January 2010, the head of the collections department called an emergency day-long workshop session to discuss this crisis and generate ideas for how to address underlying problems and improve the service level, while not incurring additional cost and ensuring bad debt targets were met.

In preparation for this workshop session, the author of this dissertation extracted data from the source systems (call center system and work-flow management system) and performed some analysis on collections activities, debit order cycles and call volumes. During the course of this analysis, the author of this dissertation noticed that call volumes were directly correlated to collections activities. SMS, soft-lock and other collections actions were probably resulting in customers calling on to the call center.

In order to verify these findings, the author of this dissertation set up some experiments with the manager of the call center. A randomly selected a sample of customers was chosen from a batch of accounts. We delayed collections actions on these accounts by two days (compared to the main population). The result was that 91% of calls from the sample batch were received 2 days after calls made by members from the main population (control group). From this we were able to empirically establish that collections actions were driving inbound call behaviour.

We used this information to develop the following hypothesis, which we then presented at the workshop session:

Collections strategies were not designed in an optimal fashion. Collections strategies were causing overlapping waves of inbound calls due to being executed in an unscheduled environment. We hypothesized that collections strategies could be re-designed based on the research into response rates and then executed in optimal fashion based on some scheduling solution. This would result in better utilization of call centre resources, throughout the month, without needing to bring in additional headcount.

Figures 5.1 and 5.2 are a visual representation of the problem. These figures describe the situation facing the call centre based on an unscheduled / non - optimal implementation of

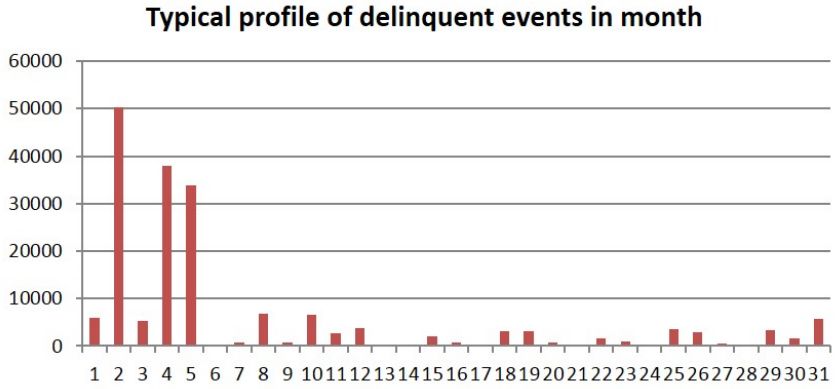

Figure 5.1: Profile of accounts going into arrears during a typical month.

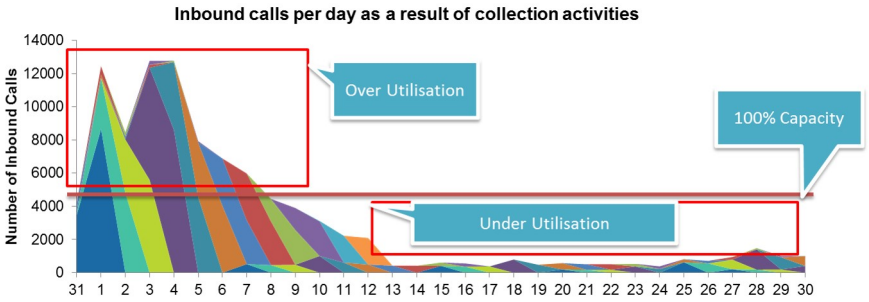

Figure 5.2: Call profile as result of collections actions.

collections strategies (all collections strategies are started on the same day that a delinquent account enters the system). In Figure 5.1 we see the number of accounts (represented on the y axis) going into arrears on a given day (represented on the x axis). In Figure 5.2 we see the resulting number of inbound calls (represented on the y axis) into the call centre as a result of collections actions on the accounts. This is also shown again later in Figure 7.2. The different colors in the chart show the number of calls as a result of collections actions which were taken on a certain date. So for example the blue color shows the number of calls received on a given day as a result of the collections actions taken that day. The teal color shows the number of calls that were received as a result of actions taken the previous day, and so on.

5.2.2 Data analysis and exploration. The first task was to perform a more rigorous and detailed analysis of available data. We extracted the complete data from the work-flow system and call center system for the previous 12 months. We developed an analytical data

set which we used to model and understand the behaviour of customers. For each inbound call we had the following information:

- Time and date of the call.
- Date of start of collection episode (date of the event which triggered entry of account into the collections process).
- Date on which any collections activities were performed against the account.
- Segmentation (based on credit and behavioral score) information of the account.

Based on the above analytical data set, we performed a data mining and analysis exercise, the results of which would form inputs into the final solution and mathematical scheduling algorithm.

As a result of these activities, it was decided that an MSc by research would be performed, the basis of which would be the industrial problem faced by the call centre and the associated mathematical scheduling problem.

5.2.3 Understanding call volumes and patterns. The first challenge of the research would be to create a model to understand inbound call volumes into the call center. We extracted data from the workflow management system as well as the call center call management system. We immediately encountered a challenge in that we had to match inbound calls with the respective billing account. (The call management system tracked the phone number of the dialing party, while the collections workflow system only had the account number). In approximately 60% of the cases it was possible to match customers to their billing accounts via the number they used to call into the call center. This was via a relatively straightforward join to the customer management system where it is possible to look-up the dialling number against customer account. The remaining 40% of cases were more challenging. Many customers call in to the call center from phone numbers not associated to the billing account in question. This is mostly attributable to customers who have reached the stage of the collections process where their cell phone account has been suspended due to non-payment. We solved the problem of identifying which billing account was related to the call in the following way:

- Both the call center management system and the collections workflow system tracked the ID of the user (call center agent) who handled the call and worked on the account.
- The call center management system was on an IBM DB2 database and the workflow system used Oracle SQL 11i. Both databases were synchronized and used the same time index.
- We created a matching algorithm which matched the inbound calls to the accounts on the workflow system based on common user identification fields and overlapping time stamps.

The result was that we were able to match every inbound call with the associated billing account.

Once this data was prepared we began to analyze call patterns and extract meaningful information to begin understanding the nature of inbound calls. The first step was to establish which inbound calls were as a result of collections activities and which were as a result of other factors. We performed further data mining and investigation and used the result of the previous step to obtain information relating to when collections actions were performed against the billing accounts of customers who have called in to the call center. If any collections activity was performed within the five days prior to a phone call, we flagged the reason for the phone call as being related to collections actions. The remaining phone calls which were not linked to any collections activities were categorized as 'noise', but they did follow a distinct periodicity. After performing the analysis, we found that inbound call volumes followed a very distinct pattern over the course of the month. Figure 5.3 illustrates the typical profile of 'noise' vs collections generated calls during a month. The result of the analysis was that we were able to examine all of the inbound call events on a given day and understand how many of the calls were related to collections activity, and how many were related to other, uncontrollable factors. We found that there were high numbers of inbound calls not related to collections actions towards the beginning and end of the month. This was linked (by judgement and information obtained when interviewing a sample of customers) to the times in the month where customers would relieve their payment from employers as well as when their account payments were due. Customers were pro actively calling in to

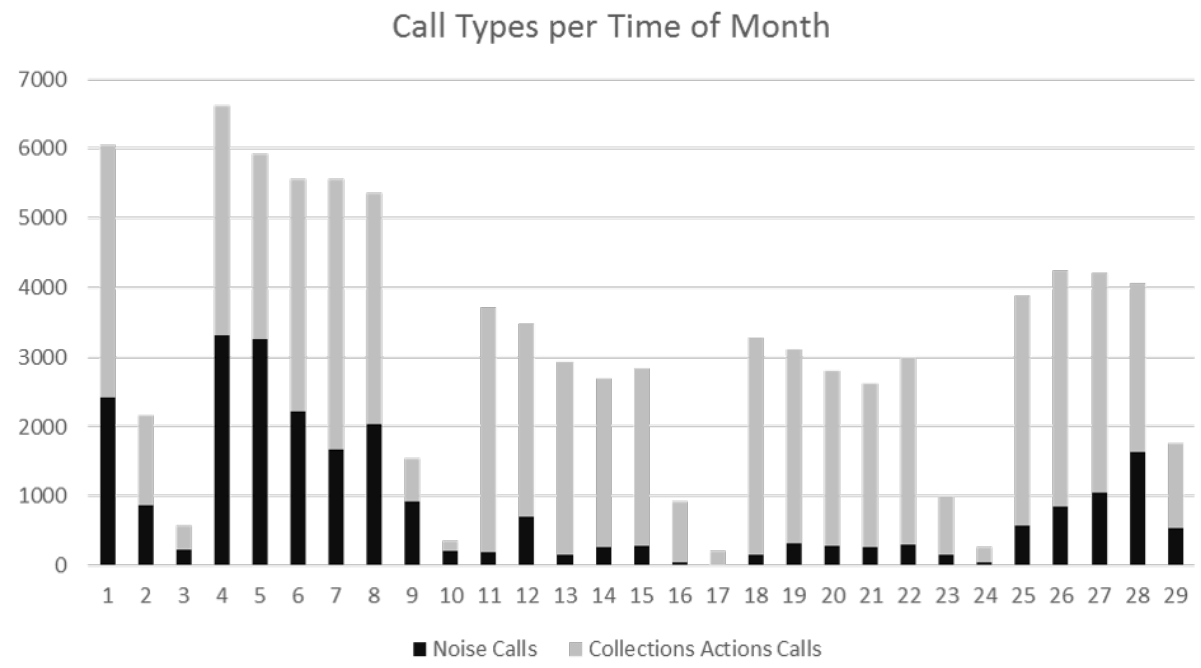

Figure 5.3: Call volumes per type for a typical month.

discuss payment terms or other options before actually triggering any collections actions through failure to pay. Identifying which calls were as a result of collections actions and which were not was an important step as it proved that the majority of calls were as a result of collections activity, and the bulk of the calls occurred within the first 10 days of the month. From this we concluded that there was a strong causality relationship between inbound calls and collections events, and that it was possible to schedule collections events in such a way that total number of calls handled in the month could be maintained and spread out over the month in such a way as to optimize service level.

5.2.4 Creating customer segments. Once the underlying patterns were detected in the inbound call volumes, we established that it would be important to understand the characteristics of the customers and how they reacted to the collections activities. We made use of an existing behavioral scorecard (a credit risk scorecard) to distribute customers into one of three risk segments: Low, Medium or High Risk. The behavioural scorecard is a tool that uses historical information based on certain well-defined attributes - mostly related to customer payment behaviour - to predict (using techniques in the well established field

of consumer credit scoring) a specific outcome. In this case, the outcome is how likely a customer who is currently in good standing will default within the next three months. Low risk indicates very low likelihood of imminent default, high risk indicates high likelihood of default. We made use of the behavioural segmentation, as well as customer payment method (cash or direct debit) to segment the customer base into 6 segments. The reason for splitting the base between direct debit and cash customers was due to the fact that cash customers would know how to settle their accounts, while customers paying direct debit would need to call in for instructions. We speculated that this would mean cash-paying customers might have a slightly different response behaviour than debit order customers. This assumption was later validated through analytical investigation; the six segments identified became the basis of the segments for the scheduling algorithm.

5.2.5 Mapping response rates. Once the six segments were identified, we began to analyse the response behaviour of each of the segments. It quickly became apparent that there was a strong relationship between behavioural risk rating and response. We found that the high risk customers typically only responded to the more onerous collections activities (the strongly-worded SMS notifications or suspension of service) while the lower risk customers had a much higher (in relative terms) response to the early SMS's and in fact, most of the lower risk customers would have phoned the call centre by the time the collections strategy progressed to the soft lock phase. Based on this analysis, we used some regression techniques (in MS Excel) to model response curves for each of the six segments based on the historical performance information which was available. The result was a response model which gave a good indication of the number of customers in each segment who would respond on a given day after being targeted for collections actions. Table 5.1 illustrates the response model for a cohort of 20,000 accounts entering collections on a given day, all of whom are in one segment. (In this case the High Risk segment of Direct Debit Customers).

Time Period	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_10	y_11	y_12	y_13
Activity	SMS1	(SMS1)	(SMS1)	SMS2	(SMS2)	(SMS2)	SOFTLOCK	(SOFTLOCK)	(SOFTLOCK)	(SOFTLOCK)	(SOFTLOCK)	(SOFTLOCK)	(SOFTLOCK)
ACCOUNTS IN BATCH	20000	18300	17477	17127	16185	15699	15699	11461	9971	9372	8998	8818	8729
RESPONSE RATE	8%	4%	2%	5%	3%	0%	22%	10%	5%	3%	2%	1%	1%
CURE RATE	0,50%	0,50%	0%	0,50%	0%	0%	5%	3%	1%	1%	0%	0%	0%
Capacity Requirement	1600	732	350	856	486	0	3454	1146	499	281	180	88	87

Table 5.1: Cohort response rate for 20,000 high risk direct debit customers

Table 5.1 shows a simple collections strategy made up of two SMS notifications and a soft lock action, respectively SMS1, SMS2 and Soft Lock, the activity names with brackets around them eg. (SMS1),(SMS2),(SOFTLOCK) refer to days on which no collections actions are being performed, but the responses from the previously performed actions are still being dealt with. Accounts in batch relate to the number of customer accounts who still require collections actions to be performed against them. This population reduces as time goes on as the customers either self cure out or call in to the call center. Either of these events result in the account exiting the population requiring further treatment (and scheduling). The start of a collections strategy against the batch of accounts, and the call centre capacity as the resource are some of the key factors that the scheduling algorithm needs to take into account. Table 5.1 also demonstrates the calculated capacity requirement based on the response to the collections action. The highest response is always on the same day as the activity, with the response reducing over time. Each combination of customer segment and collections activity has its own calculated response profile. The length of time that the collections strategy will have an impact of resources is 12 time periods (13 time points). Recall that (as explained in subsection 4.2.2 the duration / number of time periods of the collections strategy is effectively orthogonal to the time-line relating to H_t. For example, if we consider the single collections strategy, say CS_1 in Table 5.1, we see that there is an impact on resources for 12 time periods. This means that for the batch of accounts against which CS_1 was scheduled to begin during $[t, t+1)$, the last observed response (impact on capacity) would be realised during $[t12, t13)$. For a batch of accounts which CS_1 was scheduled to begin during $[t+1, t+2)$, the last observed response (impact on capacity) would be realised during $[t+13, t+14)$ and so on for all the various collection strategies.

5.2.6 Data driven collections strategies. The collections strategies, as they are applied to arrears accounts are the 'activities' which will be scheduled. The collections strategies are comprised of multiple collections actions (as defined in Chapter 4) which have been set out in some specific order. Collections strategies will have some specific duration, and associated length of time that they will have an impact on resources. The specific impact on resources during a given time period t will depend on which collections action occurs during that time period in the collections strategy. For example, a collections strategy may

take 10 days to execute, with a resource consumption impact over, say $t_{final} = 15$ days (depending on the length of time). The process of defining the collections strategies is now defined in this section. Once an understanding of how customers responded to collections actions was achieved through analysis of response rates, we began to design new collections strategies which were aligned with actual effectiveness determined by observation of customer responses. Previously, collections strategies were purely judgement-based, with no insight or sensitivity to the fact that the collections activities were overlapping and causing severe strain on the call center capacity.

Based upon the analysis of the response rates, we designed collections strategies which were aligned to the response rates of the customers segments. For example, from observation it was clear that 93% of responses to the first SMS would occur within 3 days of that activity. Therefore waiting longer than 3 days would be unnecessary and the next collection activity could occur on day 4. These start date of these collections strategies would become the projects which the scheduling algorithm would allocate against batches of accounts in some optimal way, on a daily basis. The collections strategies were designed as follows:

- The strategies were based on quantitative as well as qualitative factors.
- Qualitative influences were issues such as generally accepted behaviour in the collections industry as well as a sensitivity to the fact that even though the customers being dealt with are potentially delinquent customers, they are still consumers and most of them will remain paying customers. This means there has to be consideration given to courteousness and customer satisfaction responses. The result of this is that it was deemed necessary to have at least two SMS messages sent to the customer, with a minimum of one day between messages to allow the customer to respond, and at least a 5-day gap after the last message before a more harsh collections action, such as suspension of the customer's service.
- Quantitative factors were the results of the study of response rates, namely empirical models which would be used to define resource consumption characteristics (call centre capacity requirements) of the collections strategies.

- There was a clear distinction between cash-paying customers versus those paying by direct debit. We speculate this was a result of cash-paying customers knowing how to settle their arrears by making a direct payment, while customers paying via direct debit needed to call in to find out how to settle their accounts. This resulted in the first main bifurcation for segmentation (Cash-paying vs Non Cash-paying customers).
- Another clear distinction was that responses were well-aligned to the customers' credit risk scores. High risk customers did not have a very high response to the initial SMS notifications and were most likely to respond (if at all) to the more harsh treatment of having their service suspended (soft lock). Low risk customers had a much higher prevalence of self cure (that is, left alone having not been subjected to collections actions, they spontaneously made payment to resolve the arrears episode) at all stages of the collections strategy (regardless of most recently performed action). They also exhibited a much higher response to the initial SMS messages (most likely they were not aware there was an issue and when they received an SMS they called in quickly to resolve). We also observed that very few low risk customers progressed to the stage where the soft lock action was taken against them. Medium risk customers, as expected demonstrate a mix of the two extremes, with moderate responses to early stage messages with higher responses to the soft lock.
- Based on the analysis of the responses in the high, medium and low risk categories, as well as sensitivity to qualitative factors, the author created 3 generic collections treatment strategies. One each for high, medium and low risk customers.

The three collections strategies would be applied against customers in the direct debit or cash paying category. Depending on the payment type, there is a different response rate to the collections treatment. There are only three collections actions which were included in the collections processes: SMS 1, SMS 2 and Soft Lock actions.

The longest time from the start of a collections strategy until the last collections action (soft lock) is performed is 7 days (in strategy 1) and the last time an observable response is triggered is 6 days later, so the total observation length of time that the collections strategies could have an impact on resources after their start time period is 12 time periods. In table

Table 5.2 we show the days (after the start time period of a collections strategy) on which the collections actions are performed.

Strat / Activity Action Day	SMS 1	SMS 2	Soft Lock
Low Risk Strat	1	4	7
Med Risk Strat	1	3	6
High Risk Strat	1	2	4

Table 5.2: Activity days for collections strategies.

This concept is shown in Table 5.1 where Strategy 2 from Table 5.2 is demonstrated against a batch of 20,000 accounts. The last collections action in the strategy (Soft Lock) is actioned at time point y_7 while the responses (resource consumption) is observed until y_{13}.

5.2.7 Workforce management. Another component that would need to be addressed is the management of the capacity of the call center. The capacity of the call center (ability to handle inbound phone calls) is a function of the number of call center agents available at a given time. This may be understood by means of an example:

Suppose, from a study of available data, it is found that it takes a call centre agent working at a particular call centre three minutes to handle an inbound call (on average). Consider also that the call centre operates for 8 hours every day. We also know that the call centre agents have an efficiency of say 80%. We then calculate the total capacity per agent to be: $(0.8 \times (8 \times 60)/3 = 128$ or 128 calls per agent per day.

Practically, there are additional considerations to be taken into account, such as the number of call centre staff working per hour of the day (taking into account break times for example) as well as the distribution of inbound calls throughout the day, certain peak periods see higher call volumes than the rest of the day. All these dynamics are taken into account when building the call centre capacity model and will be briefly described below.

This capacity constraint therefore becomes a workforce management problem. It is necessary to understand how the number of agents available in the call center in a given hour translates into capacity to handle inbound calls.

Fortunately, call centre workforce management is a well-studied field. An implementation of

the Erlang C model can be used for multiple purposes. It can predict call centre load (given certain parameters such as expected call volumes). It can also, given a targeted service level, number of call centre agents and certain expected call durations, be used to calculate the number of inbound calls which would allow is to satisfy a targeted service level. For instance, given a certain availability of agents, the total call volume for the hour and for the day can be ascertained. We would then use this calculated inbound call volume as the resource capacity constraint in the scheduling model.

Parameter	Value
Expected (No. Calls)	4000
Call Duration (s)	325
Acceptable Waiting Time (s)	20

Table 5.3: Call Centre planning input metrics.

Planning Metric	Tea	Lunch
Break Shifts	12	3
Staff	65	65
People Per Shift	5	22
People on Floor	60	43
Change Over Efficiency	98%	95%
Actual people Available	58	41

Table 5.4: Call Centre workforce break schedule inputs.

Call Centre service level planing tool

Interval (Seconds)	Time	Distr.	Expected Call Volumes	Staff	SvL	(Traffic Intensity)	Duration	Occupancy	Erlang C
3600	" 8 -9"	15%	600	65	95%	54.166667	325	0.833	0.105
3600	"9 - 10"	13%	520	58	96%	46.944444	325	0.809	0.080
3600	"10 - 11"	13%	520	58	96%	46.944444	325	0.809	0.080
3600	"11 - 12"	12%	480	58	99%	43.333333	325	0.747	0.022
3600	"12 - 13"	12%	480	41	0%	43.333333	325	1.057	1.499
3600	"13 - 14"	11%	440	41	28%	39.722222	325	0.969	0.778
3600	"14 - 15"	9%	360	41	94%	32.500000	325	0.793	0.105
3600	"15 - 16"	8%	320	58	100%	28.888889	325	0.498	0.000
3600	"16 - 17"	7%	280	65	100%	25.277778	325	0.389	0.000

Daily Service Level	**78%**
Total Calls Handled	4000

Table 5.5: Call Centre workforce and capacity (service level, SvL) planning tool.

In Table 5.3 - Table 5.5 we demonstrate an implementation of the call volume planning tool which we developed in Microsoft Excel to perform the capacity calculation. On a monthly

basis (based on a frequency decided by management), the call centre manager would capture the number of call centre staff who would be working on a given day (taking into account planned leave). In the Excel tool, we then took into account unplanned leave (for example, sick leave) as well as the normal pattern of occupancy (given an 8-hour day, the number of hours agents are effectively available based on the number of breaks they take, and general human efficiency).

We also took into account the pattern of inbound calls within a day. Most calls arrive in the first hour of the day, with additional busy periods at later stages in the day. These peak periods are when maximum occupancy is necessary in order to ensure service level is maintained.

All of these dynamics translate into a total available capacity for the call centre. The typical result would be a capacity of around 4000 inbound calls on a week day, 2000 calls on a Saturday and 1000 calls on a Sunday.

In Table 5.3, Table 5.5 and Table 5.4 the service level calculation for a single day is demonstrated by way of example.

Table 5.3 shows the input parameters, namely the total expected call volumes for the day, the average call duration (from observation) and the target acceptable waiting time (in this case, 20 seconds). These parameters are then used for calculations throughout the model.

Table 5.4 shows some of the planning considerations which are then taken into account finally in Table 5.5. In Table 5.4 the Break Shifts row indicates how many shifts are used to rotate the call centre staff through their tea and lunch breaks (in this case 12). The Staff row indicates the total staff compliment that needs to be allocated to break shifts. People Per Shift refers to the number of staff that will go off together on a break shift. People on Floor is the calculated number of staff available at a given time during the lunch and tea break shifts (lunch breaks typically being an hour and tea breaks only fifteen minutes). Change over efficiency is the loss of productive time due to practicalities such as standing and sitting, logging into systems, setting up workstations etc. The actual people available is the net result of all the parameters and calculations, and will be used to calculate the call

centre capacity during the various break shifts.

In Table 5.3 and Table 5.4 we have input parameters for the model. In Table 5.5 we show how a given day may be broken up into one hour sections and the overall service level (SvL) calculated for each hour based on certain parameters, namely the distribution (Distr.) of calls throughout the day (i.e. percentage of total expected calls for the day which occur during that specific hour) which allows the calculation of the Expected Call Volumes. Staff refers to the total staff compliment available during that hour (taking into account breaks and efficiencies as set out in Table 5.4). Service Level (SvL) is then calculated based on the Erlang C function which takes Traffic Intensity (the ratio of call volumes divided by interval (Seconds) length multiplied by the call duration (Seconds)), Occupancy (Traffic intensity divided by number of staff) and the number of staff available to handle the call.

Now that the capacity of the call centre is understood, the problem becomes creating a schedule such that the calls resulting from collections action do not exceed the available capacity on a given day.

5.2.8 Workflow optimization. Once we had arrived at an understanding of the elements involved in the capacity problem facing the call centre, we were ready to begin addressing the problem of how to initiate collections activities in such a way as to maximally utilize the available call center resources, without exceeding the service level constraints. We deduced that the work flow optimization would take the form of a scheduling problem, where the collections actions would need to be scheduled against accounts in arrears in such a way that all arrears accounts would need to be treated, but the actions could be spread out in such a way as to optimally make use of available capacity. Instead of having huge peaks of activity during the beginning of the month, and suffering severely low call center service levels, followed by lulls or periods of low activity in the middle of the month with 100% service level, collections actions could be scheduled in such a way as to achieve a consistent service level throughout the month. Our conceptual approach is as follows:

- Create 3 collections strategies which comprised of collections activities occurring at specific intervals. The rationale behind the 3 strategies would be to have a differentiated treatment for low risk, medium risk and high risk accounts respectively. High risk

accounts would face accelerated treatment. This was based on the fact that the research into response rates showed that high risk account holders were least likely to respond to the initial collections SMS actions and would only begin to respond (if at all) to the soft lock activity. They were also most likely (compared to the medium and low risk) to default again on their next payment cycle. It is important to make sure the collections strategy deals with these accounts quickly so that they could progress to the late stage collections process, where their billing would stop. This is also the phase where specialist collections agencies could attempt to get hold of the account owner once the earlier automatic collections actions failed.

- Create a scheduling algorithm which would run daily and decide which accounts would be released into the collections processing environment (begin collections strategies against the accounts).
- Update the pool of eligible accounts on a daily basis to accurately reflect the total number of accounts requiring treatment (some customers would 'self cure' while in the queue pending the start of collections actions).
- Use a scheduling algorithm to ascertain which accounts from the holding pool could be released into the production pool based on available call centre capacity.

5.2.9 Implementation of solution. Given that this was a practical problem being experienced in a real world call centre, we needed to use existing systems and procure additional software packages for implementing the various mechanisms necessary to solve the overall problem. The following is an overview of the key tangible and intangible systems and processes which we used to implement the scheduling algorithm and overall solution:

- Workflow management system: The call centre already made use of a workflow management system. This is a commercial software package which operates over an Oracle SQL database. The workflow management software is responsible for implementing and keeping track of collections actions that are performed automatically as well as those performed by human agents. The author of this dissertation engaged with the system administrators and the project office to implement the new collections

strategies. This entailed writing a detailed business requirement specification to outline the collections strategies, as well as working with the system administrators to design the system interface which would allow the workflow system to make use of the results of the scheduling algorithm.

- Workforce and capacity planning tool: As mentioned in subsection 5.2.7, this is a tool designed in Microsoft Excel which makes use of the Erlang C function and intelligence derived from the call centre managers to calculate the total available capacity on a given day. A process was implemented where the capacity was updated on a monthly basis. The result of the capacity calculation was captured as a text file and loaded to a network location on a server.

- Scheduling Tool: We designed the scheduling algorithm using base Matlab (to be detailed in the next section) and using the Matlab deployment module, created a standalone executable software package which would execute the scheduling algorithm on a daily basis.

- Integration: On a daily basis, the work-flow system would produce a file which listed the number of accounts in the holding pool. These accounts were split in the 6 categories, depending on whether they were cash or direct debit customers and if they were in the high, medium or low risk categories. The Matlab package would then import this information as well as the capacity file on a daily basis and perform the scheduling optimization. An output file was then produced which the Work-flow programme then picked up and used to release into the collections environment on that specific day (based on the result from the schedule which indicates how many collections strategies should be initiated in the current time period).

- Ongoing enhancements: The system was continually enhanced and we updated the software package as practical considerations needed to be taken into account. For example, we added a routine where working days and non working days were overlaid on top of the capacity calculation to force no releases on certain days (such as public holidays) when there were no call centre staff working.

5.3 Summary

In this chapter, we have presented and reviewed the steps taken to arrive at the point where we would be ready to consolidate the various inputs and use the CTWRLP approach to create a schedule which would address the requirements of the real world problem experienced by the call centre. In the next chapter, we will introduce the Simulated Annealing optimization method, which is the component required to perform the optimization necessary for arriving at a solution.

Chapter 6

Simulated Annealing

Simulated Annealing is the algorithm chosen to solve the real world problem. In this chapter we will provide an overview of the algorithm. A description and explanation of the method will be provided and reasons for choosing simulated annealing as the method for solving the real world problem will be discussed. A detailed review of the method may be found in Rutenbar (1989), we will summarise the important features in this chapter.

6.1 Introduction to Simulated Annealing

The Simulated Annealing (SA) method or technique was developed by Kirkpatrick et al. (1983). It is a technique developed specifically for the field of combinatorial optimization problems. This field of optimization is a practical field and has considerable focus due to the combinatorial nature of many real-world problems. The unique feature of SA is that it is based on the observation of the statistical nature of annealing in solids. This is the physical process where a solid that has a uniform crystalline structure under certain conditions is heated up (or raised to a high enough energy state) such that the crystalline structure dissolves (dis-ordered chemical state) and then cooled according to some careful schedule until the material once again coalesces into a highly ordered crystal structure, Rutenbar (1989). The SA combinatorial optimization technique seeks to replicate this

physical phenomenon in the field of combinatorial optimization in order to solve complex combinatorial problems.

6.1.1 Principles of Simulated Annealing. The SA method is set up in such a way as to simulate the physical annealing process. In the annealing process, some solid material is heated or raised to a high enough energy state such that its constituent atoms are in a state of unstructured flux. The material is then subjected to cooling in a structured and deliberate manner, such that it reaches thermal equilibrium at each temperature. This means that the atomic structure of the material has settled into a state commensurate with the current temperature level. If the temperature changes, so too will the structure. The more regular the configuration of particles, the less energy is required. As the system is cooled to lower and lower levels, so too will the arrangement of particles begin to achieve an optimal configuration which minimizes overall energy requirements (cost). The SA algorithm seeks to mimic the physical process by achieving an optimal solution to a combinatorial problem through iterative improvement.

The SA method takes into account the need to have varying energy states (temperature levels) as well as the statistical mechanical nature in which the annealing process occurs. This refers to the fact that as a solid anneals, (cools down) there is a certain likelihood that the chemical components will settle into a regular (optimal) crystal structure. It is not guaranteed that as the temperature reduces, the solid will arrange in this crystalline form. The SA method addresses this through a cooling schedule approach. Regardless of the objective function E being optimized, the SA technique utilises a random search technique during each epoch of the cooling schedule. The algorithm will then accept candidate solutions which are more optimal than previous solutions as well some candidate solutions which are less optimal according to some probability defied as $p = \exp(-\delta E/T)$ where δE is the change in objective function E (analogous to the change in energy in the physical example, hence being represented by E) and T is the temperature control parameter T (relating to the cooling schedule in the physical annealing process). In simple terms, the algorithm starts with a random solution and tests further candidate solution against the following criteria: If the solution reduces the energy, accept the solution. If the solution increases the energy,

accept solution with probability p as defined above.

An important concept relating to the Simulated Annealing method is the cooling schedule. This refers the the algorithmic mechanisms by which the method mimics the reduction of temperature in the physical annealing process. The cooling schedule is generally comprised of four main features:

- T_0, the initial temperature of the system.
- ϵ, the terminal condition for the cooling schedule (once the temperature is reduced below this value, stop iterating the algorithm.
- M, the Markov Chain length condition, which controls the number of solution search iterations performed at each temperature T_t.
- α, the factor by which the temperature T_t is reduced to generate T_{t+1}.

Selecting values for the parameters of the cooling schedule becomes important when tuning the algorithm for performance, and for ensuring the required tolerance level is achieved for the solutions.

6.2 Simulated Annealing Algorithm

We now present conceptual steps in the Simulated Annealing Algorithm.

6.2.1 Applicability of Simulated Annealing to the industrial problem. We first present our motivation for the choice of SA. The Simulated Annealing algorithm is a robust and practical algorithm which can be used to address a wide range of combinatorial optimization problems, without having to first uncover deeper insight or obtain detailed understanding of the mathematical structure which underlies the problem, Rutenbar (1989). We have shown that the real world problem exhibits characteristics of a complex combinatorial problem. The SA algorithm is a stochastic algorithm which is able to locate a good solution quickly. Moreover, the implementation of SA does not require any mathematical properties of either the objective function or the constraints. The problem

Algorithm 1: Simulated Annealing Algorithm

Begin Simulated Annealing Optimization Algorithm.;
Input Parameter: M **= number of iterations to attempt at each temperature**;
Input Parameter: T_0 **= initial temperature (high energy state)**;
while *Cooling schedule end condition not met for* T *(*$T_{t+1} \geq \epsilon$*)* **do**
    for $m = 1$ **to** M **do**
        Generate random candidate solution;
        Evaluate the change in energy δE (difference between active and candidate objective function values);
        if $\delta E < 0$ **then**
            Accept candidate solution as new active solution and update configuration;
        else
            Accept candidate solution with probability and update configuration $P = \exp(-\delta E/T)$;
        end
    end
    Update (reduce) temperature parameter T in line with cooling schedule. $(T_{t+1} = \alpha T_t)$
end

could also be considered to be somewhat ‘messy’ in nature in that an exact deterministic solution is not required, but rather a reliably ’good enough’ solution that can be produced quickly is the goal. The nature of the objective / cost function and constraints are also fluid and could change over time - for example the introduction of additional resources or constraints, and a finely tuned heuristic would need to be re-developed to cater for these while the SA algorithm would be able to incorporate these changes relatively easily. The practical nature of the problem and also the requirement that the solution be robust, repeatable and easily implementable were also significant considerations in selecting the SA method as the chosen solution approach.

6.3 Summary

In this chapter we briefly introduced the Simulated Annealing method. The steps of the SA algorithm have been summarised in algorithm 1 In the next chapter we will now demonstrate the SA method used with the CTWRLP model to address the requirements of the real world problem.

Chapter 7

Solving the Critical Time Window Resource Levelling Problem

In this chapter, we will describe the approach taken to obtain a solution to the Critical Time Window Resource Levelling Problem, formulated specifically to address the industrial problem, the objective function of which has been proposed in Equation 4.2.11. We have described the industrial problem and shown that it exhibits many characteristics of the resource levelling and resource constrained project scheduling class of problems. We have also developed the extensions to these fields, specifically the introduction of the notion of the critical time window method. The Simulated Annealing algorithm has also been described and positioned as the meta-heuristic which will be used to perform the optimization of our problem. We now present the steps taken to consolidate the research and understanding into a practical solution of the real world problem. We will also discuss how the solution was implemented and also the results after implementing the solution framework practically in the call centre.

In this section, we will present the detailed steps necessary to implement the Critical Time Window Resource Levelling Problem with Continual Rescheduling. We begin by reviewing some key concepts.

The idea of Continual Rescheduling may be thought of as the conceptual 'shell' or container

of the method, within which the actual computational optimization is performed. This describes the process where reactive rescheduling is performed at every time period (every day in the case of our problem). Within the shell is the problem, represented by Equation 4.2.11 which is optimized at each time period. Finally, in order to perform the optimization, certain calculations are required to obtain the inputs necessary to perform the computation in Equation 4.2.11. Namely: the calculation of the resource consumption of each resource k for all activities for a schedule S during each time period in optimization time-line L_+. This calculation is specific to the nature of the underlying problem. In this chapter we will demonstrate the approach that was taken in addressing the industrial problem.

In this chapter we will present the following:

- The computational steps necessary to generate an initial input schedule S_0.
- The process to calculate the resource consumption requirement $r_k(S,t)$ for Equation 4.2.11 from a given schedule S.
- The principles for calculating a candidate solution S_1 from S_0 and calculating the associated resource consumption profile $r_k(S_0,t)$ and $r_k(S_1,t)$.
- The process of establishing the weight function $\omega(t)$ in order to enforce the critical time window condition.
- The evaluation of the objective function given by Equation 4.2.11 with respect to a schedule S, weight function ω and resource availability R_k.
- The use of the Simulated Annealing optimization algorithm to find an optimal solution to Equation 4.2.11.
- The continual rescheduling approach to deal with the features of the real world problem.

We conclude the chapter by providing an algorithm which encapsulates all of the above points.

7.1 Model Calculations

In this section, we discuss the calculations that are necessary to formulate the problem in such a way that it may be addressed by a global optimization algorithm.

7.1.1 Initialization of first solution (schedule): $\mathbf{S_{H_{t-1}}}$. The starting point for the calculations of the arguments for Equation 4.2.11 is an initial schedule which we will refer to as S_0. This is effectively a matrix which represents the activities which are required to be scheduled during the active scheduling phase. We will first demonstrate the pre-processing that will be necessary to obtain S_0 from a set of inputs. Namely: historical information relating to activities which have been performed in the past and still have an impact in the optimization time-line L_+ and information relating to the new activities entering the system at the current time period which will need to be taken into account when generating a new optimal schedule.

If we are performing the optimization during phase H_t then the initial schedule will comprise activities which were not yet started during the previous scheduling (H_{t-1}) phase as well as any new activities which entered the system during the current scheduling phase. So, for example, if in the previous scheduling phase H_{t-1}, 10000 activities were scheduled but only say 1000 activities were scheduled to start at time $t-1$ during phase H_{t-1} (which becomes t during active phase H_t) then 9000 activities will not yet have begun during the current active scheduling phase and will be eligible for rescheduling based on current information. This is in line with the continual rescheduling feature introduced in subsection 4.2.2.

We demonstrate the concept now by way of an example:

Consider the case where there are three categories of activities (collections strategies / processes) with independent resource consumption profiles requiring scheduling. The rolling schedule horizon H_t is 6, 3 periods of history and 3 periods of current at t and future at L_+, in terms of Equation 4.2.6 we see that in this case $\tau = 3$. The longest duration a scheduled activity may have an impact on resource consumption is 3 time periods.

The result of the schedule execution during a particular active phase H_t is a 3×7 matrix

(where the rows represent the activities and columns are the time intervals):

$$S_{H_{t-1}} = \begin{bmatrix} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}$$

In this example, for simplicity, the number of activities requiring scheduling is the same in each class per time period. So, if we consider row 1 which would represent a specific collections strategy, say collections strategy A, in the case of our problem, then today would be represented by column 4, yesterday would be column 3 and three days ago would be column 1. Similarly, tomorrow would be column 5 and three days from now would be column 7. Today, collections strategy A is scheduled to begin against 4 accounts. Yesterday it was scheduled to start against 5, three days ago it was scheduled to start against 7 and so on. Recall that the notion of 'activities being scheduled' relates to the start date of the collections strategies applied against batched of accounts. The way this matrix is interpreted is as follows:

- During the active scheduling phase H_t, the start time for the schedule, t, is the 4th column. This is refers to the column of $S_{H_{t-1}}$ containing the values

  $$\begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix}$$

  in the example. (Each of the three collections strategies - represented by the rows of the matrix - should be started against 4 accounts at time period t.)

- Each previous column represents the activities released during $[t-1, t)$ of previous scheduling phases. So for example the activities represented in $t-3$ of H_t are the activities that where scheduled to be performed at time t of H_{t-3}. This refers the the column containing the values

  $$\begin{bmatrix} 7 \\ 7 \\ 7 \end{bmatrix}$$

  in the example.

We obtain the information about the new activities on all three categories over the period $(t-1, t]$ of H_{t-1} which need to be scheduled in the next phase, H_t. This represented by the vector:

$$J_t = \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

These new activities in J_t entering the system, as well as the schedule $S_{H_{t-1}}$ from the previous active phase H_{t-1} are the activity inputs into the scheduling algorithm. From these inputs we will now calculate S_0 which will be our initial solution for the current active scheduling phase H_t.

7.1.2 Align inputs for optimization initialisation. In the previous section, we presented the matrix $S_{H_{t-1}}$, the result (schedule) of a certain scheduling phase, H_{t-1}. The first step in initialising the components is to update the scheduling phase - H_t is now the active phase (we advance one time period forward along the scheduling horizon).

The result $S_{H_{t-1}}$ of the schedule from the previous step, H_{t-1}, will be used along with new information for the current active phase, namely the new inputs in J_t.

We know that the activities from the previous schedule $S_{H_{t-1}}$ which have yet to be performed are represented in columns 5:7 in $S_{H_{t-1}}$, the schedule result of phase H_{t-1}, namely the values:

$$\begin{bmatrix} 3 & 2 & 1 \\ 3 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix}$$

The total amount of activities per class requiring scheduling during active phase H_t is then $\hat{J}$ (summing across the rows of J_t and $S_{H_{t-1}}(5:7)$):

$$\hat{J} = \begin{bmatrix} 16 \\ 26 \\ 36 \end{bmatrix}$$

7.1.3 Initialise first trial solution schedule S_0. We now need to prepare the inputs in order to create S_0, the initial schedule for phase H_t against which the first candidate

solution S_1 will be tested. We do this by discarding the values from $S_{H_{t-1}}$ in column 1, as these activities no longer have an impact on resource consumption in the current and future time periods i.e. in L_+. We then shift all the values in columns 2,3,4 across to columns 1,2,3 respectively. These are the activities which have been scheduled in previous scheduling phases which still have an impact on resources in current and future time periods. Finally we insert the calculated activities from $\hat{J}$ into S_0 in column 4. Thus we obtain the initial schedule at t:

$$S_0 = \begin{bmatrix} 6 & 5 & 4 & 16 & 0 & 0 & 0 \\ 6 & 5 & 4 & 26 & 0 & 0 & 0 \\ 6 & 5 & 4 & 36 & 0 & 0 & 0 \end{bmatrix} \tag{7.1.1}$$

7.1.4 Calculate resource consumption requirements $r_k(S,t)$. In order to compute Equation 4.2.11, we will need to calculate a value for the resource consumption requirements of problem being considered. In the case of our real world problem, there is only a single resource, i.e. $k = 1$ therefore we will refer to the resource consumption values for a given schedule S calculated at time t as $r(S,t)$.

The first step in calculating this value is to calculate the expected responses for the active and candidate schedules. We now introduce the resource consumption profile input, relating to the activities being scheduled in a given schedule S. We will specifically describe the calculation $r(S_0,t)$ for the initial schedule S_0.

As described, each activity has a resource consumption requirement for at most the duration of the optimization time-line i.e. over L_+ (suppose for this example: 3 time periods). An activity scheduled to be performed during t, will have a certain impact on resources over $L_+ \subset H_t$ where $L_+ = \{t, t+1, t+2, t+3\}$. Similarly an activity that was performed during $t-3$ will have a resource requirement in $\{t-3, t-2, t-1, t\} \subset H_t$. The resource consumption requirements of the activities can be shown as a 3×4 matrix C, say for example:

$$C = \begin{bmatrix} 0.5 & 0.3 & 0.15 & 0.05 \\ 1 & 0.8 & 0.5 & 0.3 \\ 1.5 & 1.3 & 0.9 & 0.5 \end{bmatrix} \tag{7.1.2}$$

where c_{ij} represents the resource requirements of activity i (collections strategy i as applied

to a single account) at time period j.

Recall (from subsection 4.2.2) that it is not necessary for the time dimension of C (ϕ) to be equal to that of L_+ (τ), but L_+ should be at least the same length of time as the time dimension of C (and possibly longer).

So, if 1 unit of activity 1 was performed during t, then it would consume 0.5 units of resources during t and 0.3 units of resource during $t+1$ and so on. For the purposes of this demonstration, we have values in C which are greater then 1. These values were chosen only for the purpose of providing a numerical example and do not necessarily reflect the practical situation where it is (generally) not expected to have more than one response per account. The approach followed to calculate the values of C in our real world problem was discussed in subsection 5.2.5.

This will depend on the maximum duration for which an activity performed in a given time period will continue to have an impact on resources in future time periods.

We now have:

- A schedule S with rows denoting the various collections strategies and columns being the scheduling horizon H_t.
- We denote (from Equation 4.2.5) $T = 2\tau + 1$ to be the column dimension of S.
- n represtens the number of collections strategies.
- Thus $S \in \mathbb{R}^{n \times T}$.
- $C \in \mathbb{R}^{n \times \phi}$, ϕ is the number of periods of responses observed after the start of a collections strategy.

The total response profile for all activities in the schedule is then calculated by performing the calculation:

$$C^\intercal S \tag{7.1.3}$$

where S is a candidate feasible schedule for the problem at hand.

We are now able to compute a value for $r(S,t)$ for our example using S_0. The calculation of $r(S_0,t)$ is as follows:

This is achieved by performing matrix multiplication. For an $n \times T$ matrix of activities to be scheduled, S_0 and an $n \times \phi$ matrix of responses / resource consumptions C, the total response / resource consumption is $\bar{C}_{Tot} = C^\intercal S_0$ where $\bar{C}_{Tot}$ is an $\phi \times T$ matrix.

We may demonstrate this step using the examples defined above, namely:

$$S_0 = \begin{bmatrix} 6 & 5 & 4 & 16 & 0 & 0 & 0 \\ 6 & 5 & 4 & 26 & 0 & 0 & 0 \\ 6 & 5 & 4 & 36 & 0 & 0 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0.5 & 0.3 & 0.15 & 0.05 \\ 1 & 0.8 & 0.5 & 0.3 \\ 1.5 & 1.3 & 0.9 & 0.5 \end{bmatrix}.$$

Then for $\bar{C}_{Tot}(S_0) = C^\intercal S_0$ we obtain:

$$\bar{C}_{Tot}(S_0) = \begin{bmatrix} 18 & 15 & 12 & 88 & 0 & 0 & 0 \\ 14.4 & 12 & 9.6 & 72.4 & 0 & 0 & 0 \\ 9.3 & 7.75 & 6.2 & 47.8 & 0 & 0 & 0 \\ 5.1 & 4.25 & 3.4 & 26.6 & 0 & 0 & 0 \end{bmatrix},$$

$\bar{C}_{Tot}(S_0) \in \mathbb{R}^{\phi \times T}$

(Note that the initial schedule S_0 has 0 values in the last three columns, but new candidate solution matrices S_i will have values). The way this matrix should be interpreted is now explained.

The different classes of activities are no longer represented in the matrices, these fall away in the computation of the total responses.

The columns of the total response matrices still represent the schedule time periods and the rows now represent the time periods during which the activities scheduled to be performed will still have a resource consumption requirement.

By way of example, consider column 1 of the matrix $\bar{C}_{Tot}(S_0)$.

This column represents the activities which were scheduled to be performed at time $t-3$ (which would have been the centre time point during schedule phase H_{t-3}).

The total resource consumption requirements for all the activities scheduled for action during $t-3$ can be obtained by reading down the rows. So for the activities scheduled to be performed during $t-3$

- The total resource requirement during $t-3$ is 18 units of resource.
- The total resource requirement during $t-2$ is 14.4 units of resource.
- The total resource requirement during $t-1$ is 9.3 units of resource.
- The total resource requirement during t is 5.1 units of resource.

The total resource consumption requirements for all the activities scheduled for action during $t-2$ can be obtained by reading down the rows. So for the activities scheduled to be performed during $t-2$

- The total resource requirement during $t-2$ is 15 units of resource.
- The total resource requirement during $t-1$ is 12 units of resource.
- The total resource requirement during t is 7.75 units of resource.
- The total resource requirement during $t+1$ is 4.25 units of resource.

And so on for each column in $\bar{C}_{Tot}(S_0)$.

Finally, we now calculate the total resource consumption requirements of a schedule S_0 at given time period t i.e. $r(S_0, t)$.

Based on the above explanation, it is evident that the resource consumption during a given time period depends on the activities that have been performed in the past. Referring once again to $\bar{C}_{Tot}(S_0)$, the total resource consumption during t, i.e. $r(S_0, t)$ as defined in Equation 4.2.11, depends on the activities performed in all the previous time periods of the schedule.

In our example, the resource requirement on a given day is the sum of the 'same day' responses of the activities performed on that day added to the 2nd day responses of the activities performed the previous day, plus the 3rd day responses of the activities performed 2 days prior, plus the 4th day responses of the activities scheduled 3 days prior.

(Day 1 responses are counted on the same day that the activities are performed).

In this case, $r(S_0, t_0) = 5.1 + 7.75 + 9.6 + 88 = 110.45$.

More generally, for an $\phi \times T$ activity response matrix $\bar{C}_{Tot}$ the resource consumption of resource k (including the multi resource case in this definition for the purpose of generality) across all activities for schedule S is $r_k(S)$ where:

$$r_k(S,t) = \begin{cases} \sum_{i=0}^{\phi-1} \bar{C}_{Tot}(i+1, t-i), & \text{if } i < t. \\ 0, & \text{otherwise.} \end{cases} \qquad \forall t \in H_t \tag{7.1.4}$$

where l in the sum represents the row dimension of the matrix $\bar{C}_{Tot}$.

For our example, $r(S_0, t)$ (where t represents the centre of the scheduling horizon, column 4 in this example) is then:

$$r(S_0, t) = r(S_0, 4) = \bar{C}_{Tot}(1,4) + \bar{C}_{Tot}(2,3) + \bar{C}_{Tot}(3,2) + \bar{C}_{Tot}(4,1)$$

i.e.

$$r(S_0, t) = \begin{bmatrix} 18 & 29.4 & 33.3 & 110.45 & 82.85 & 51.2 & 26.6 \end{bmatrix}$$

This concludes the explanation of how to obtain the resource consumption calculation values necessary to compute Equation 4.2.11. We now address the next step which relates to obtaining a candidate solution schedule S_1 from an initial schedule S_0

7.1.5 Generate candidate schedule S_1 from S_0. Combinatorial optimization techniques seek to optimize an objective function by testing various combinations of the underlying input value. We now discuss the method for calculating a candidate solution S_1 from initial solution S_0 that we developed for our problem and scheduling model. In the context of the CTWRLP model, the considerations for generating the candidate solution are as follows:

- The candidate solution S_1 is generated using S_0 as the basis.

- The values in S_0 representing the activities which have been performed in the past that still have impact on resources in the present and future must not be altered. These activities have already been performed and their impact on resources is effectively permanent (until they are phased out as the scheduling horizon advances). These values are in columns 1:3 of S_0 (Equation 7.1.1) in our example.
- The values in S_0 representing the activities (i.e. the start date of the collections strategy i is at the time point represented by that particular column j, for that number of accounts, shown in cell S_{ij}) which have been scheduled to be performed in the current and future time periods must be permuted in some way in order to create a new candidate solution. These are the values in columns 4:7 of S_0 (Equation 7.1.1) in our example.
- The permutation needs to be done in such a way such that the total sum of activities for each class of activities is conserved across S_0 and S_1.

The process to create a candidate solution S_1 from S_0 is as follows: Very simply, we seek to create a new candidate solution in some random way such that the total number of activities to be performed stays the same, but the number of activities scheduled to start during each time period is changed compared to the initial solution. For each row n in the $n \times T$ matrix S_0, permutations of the values in row n will be performed as follows:

1. Step 1: Select a random integer between 1 and the total number of columns T in S_0, which will be the total number of columns to be modified. (This will change during each iteration of the Simulated Annealing algorithm runtime).
2. Step 2: Repeat the following until the total number of adjusted columns is equal to T defined in step 1:
    - The value in each cell may be increased only once and decreased only once.
    - Select a random column in the active row of S_0.
    - Reduce the value in that column by some random amount.
    - Select another random column in the active row of S_0.

- Increase the value in that column by the same amount as the value which was used to reduce the amount in the earlier step.

This may be represented mathematically as: suppose we have $x_T = x_1 + x_2 + x_3 + \ldots + x_n$. Then for a fixed value of x_t we wish to vary (randomly) the values of each x_i for $i \in [1 : n]$.

This is achieved as follows: Let $\rho_i \in [0, 1]$ be random for each i. Then for each i we define

$$\hat{x}_i = \left(\rho_i / \sum_{i=1}^{n} \rho_i \right) x_T. \tag{7.1.5}$$

In order to add an additional degree of randomness into the process of generating a candidate solution (schedule), the number of values to be included in the calculation of Equation 7.1.5 is randomly selected. So, for example, if there are 4 columns in the matrix, a number of columns between 1 and 4 could be chosen to be modified. Selecting 1 will result in no change, selecting 4 will result in all columns being modified and so on.

A practical demonstration of the implementation of this pseudo-code is provided in the Appendix, in section A.3. The key point to take into account is that generating a random solution means creating schedules where the total number of activities remains constant across various candidate solutions. The order in which the activities are performed (or the number of activities performed on a given day of a schedule) varies.

Note in matrix C the values c_{ij} are percentages, that is they indicate the percentage of the number of accounts which will respond in a given time period. For example, in Equation 7.1.2 $c_{11} = 0.5$. This means that 50% of the number of accounts against which collection strategy 1 is applied, will respond on the same (first) day. Say, from schedule S_0, collections strategy 1 (row 1) was scheduled to start against 6 accounts at time point $t - 3$ (column 1). The from c_{11} we see that 50% of those 6 accounts, i.e. 3 accounts are expected to call in to the call centre, i.e. 3 calls. Since we are using percentages to calculate the expected responses, it is not necessary for the values in a schedule S to be integers, we can use fractions. In the practical implementation we are dealing with thousands of accounts per time period per strategy, and the percentage responses of those accounts.

For the purposes of demonstration, a candidate solution S_1 will be generated from the initial

trial solution S_0. Recall:

$$S_0 = \begin{bmatrix} 6 & 5 & 4 & 16 & 0 & 0 & 0 \\ 6 & 5 & 4 & 26 & 0 & 0 & 0 \\ 6 & 5 & 4 & 36 & 0 & 0 & 0 \end{bmatrix}$$

The values which will be used to generate the solution is the subset $\hat{S}_0$ of S_0:

$$\hat{S}_0 = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 26 & 0 & 0 & 0 \\ 36 & 0 & 0 & 0 \end{bmatrix}$$

Performing the candidate solution generator step on $\hat{S}_0$ returns the result $\hat{S}_1$ where;

$$\hat{S}_1 = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0.2963 & 0 & 0 & 25.7037 \\ 3.0844 & 0 & 0 & 32.9156 \end{bmatrix}$$

Recall that the selection of the number of values to be adjusted is based on the selection of a random number which could be 1. This has occurred in the above calculation, for example in row 1, where we see that row 1 has remained unchanged. While in both rows 2 and 3 the number of columns randomly selected to be modified was 2 and in both instances the column selected for modification was column 4. As the optimization algorithm iterates through many candidate schedules, the values will change until no 0 values remain (unless they are selected by the optimization engine as favourable values for optimal solutions).

In section A.3 we provide an additional example to demonstrate several iterations of this candidate solution generating routine to demonstrate how the values are changed each iteration while the total sum across each row of the matrix remains constant.

The complete candidate solution S_1 is then obtained by concatenating $\hat{S}_1$ with the values in S_0 which are the activity history and cannot be altered. Thus we obtain S_1 where

$$\hat{S}_1 = \begin{bmatrix} 6 & 5 & 4 & 16 & 0 & 0 & 0 \\ 6 & 5 & 4 & 0.2963 & 0 & 0 & 25.7037 \\ 6 & 5 & 4 & 3.0844 & 0 & 0 & 32.9156 \end{bmatrix}$$

This concludes the explanation on the method to obtain a candidate solution S_1 from initial solution S_0. For the purposes of continuing the worked example and by applying the same techniques presented in subsection 7.1.4 where the resource consumption profiles were calculated for S_0, we obtain the modelled response for candidate solution S_1, $\bar{C}_{Tot}(S_1) = C^{\intercal}S_1$:

$$\bar{C}_{Tot}(S_1) = \begin{bmatrix} 18 & 15 & 12 & 12.9229 & 0 & 0 & 75.0771 \\ 14.4 & 12 & 9.6 & 9.0467 & 0 & 0 & 63.3533 \\ 9.3 & 7.75 & 6.2 & 5.3241 & 0 & 0 & 42.4759 \\ 5.1 & 4.25 & 3.4 & 2.4311 & 0 & 0 & 24.1689 \end{bmatrix}$$

And total resource consumption $r(S_1)$ is similarly (from Equation 7.1.4) calculated as:

$$r(S_1, t) = \begin{bmatrix} 18 & 29.4 & 33.3 & 35.3729 & 19.4967 & 8.7241 & 77.5082 \end{bmatrix}$$

We will now begin to present the approach to obtaining objective function values based on the above inputs.

7.1.6 Resource availability input (capacity). The resource availability is also an input for the computation of Equation 4.2.11. This is a vector R of values which are the resource availabilities per time period. For example (values selected for purposes of providing numerical example):

$$R(t) = \begin{bmatrix} 80 & 80 & 80 & 80 & 80 & 80 & 80 \end{bmatrix}$$

7.1.7 Critical time window resource levelling feature and initialization of penalty function $\omega(t)$. We now discuss the implementation of the new feature of critical time window resource levelling. This is applied in the objective function calculation phase of the SA execution to steer selection of candidate solutions towards solutions that provide better resource consumption profiles in the short term (Critical Time Window).

The selection of the critical time window phase is based on expert knowledge and judgement. The critical time window phase is the subset of the continual schedule time horizon during which activities need to be scheduled such that all resources are (in the case of the real world problem) fully utilised. The mechanism by which this is implemented in the Simulated

Annealing algorithm is to introduce a weighting which ascribes relative importance to the various time periods. A detailed overview is provided in Chapter A specifically in section A.1.

A simple explanation is as follows: Consider two candidate solution schedules generated by the Simulated Annealing method. These two schedules will be the result of a random permutation of the underlying inputs. We desire the algorithm to select the solution that more optimally uses the available resources, with the additional condition that the solution which better utilises the resources within the predefined critical time window be given preference, even if the overall resource consumption profile is not as good.

As a discussion point to clarify the idea of the usefulness of the Critical Time Window (CTW) Reactive Scheduling method, we could think about this in the context of the following example (relating to the real world problem): Suppose we have a call centre where the inbound call handling capacity is 5000 calls per day - this implies the capacity of the call centre over a 15 day scheduling horizon is notionally 75000 calls. We define our critical time window as 1 time period (which means we ascribe the most importance to optimal resource allocation during the active scheduling phase only). Consider a proposed schedule that does not use the CTW approach. The schedule results in a resource consumption profile that is uniform over the entire scheduling horizon, say 3000 calls per day for 15 days - a total of 45000 calls to be handled. Importantly, there is surplus capacity on day 1 (and all other days less relevantly) of 2000 calls. This surplus capacity will never be recovered (one could think of the resource being destroyed if not utilised). Consider another proposed schedule, which was created using the CTW approach, where the resource consumption profile was 5000 calls on day one, 0 calls from day 2 through 14 and 40000 calls on day 10. The important feature of this schedule is that there is no underutilisation of capacity on day 1! The second schedule is actually the more favourable solution and we require that the objective function be structured such that this type of solution is favoured.

Practical considerations as to why this CTW approach is desirable are as follows:

- The number of new accounts entering the system during each time period are unknown.
- By ensuring that resources are always fully utilised at every time step, we are able

best cater for the situation where there may be an increase in the number of activities requiring scheduling in the future.

- Providing the number of activities in the buffer (in other words not scheduled for release in the active time period, but rather for some time in the future) does not exceed the notional capacity of the call centre over the future scheduling horizon, all activities will be able to be performed without impacting resource constraints.

The critical time window feature of the SA algorithm is thus designed to ensure that selection of candidate solutions favours solutions which provide more optimal utilisation of resources according to the defined critical time window.

Finally, in order to perform the objective function calculation of f for each of the candidate solutions, we define the critical time window penalty function $\omega(t)$ for $t \in CT$ where CT is defined in Equation 4.2.8. This is the factor that will ensure that the optimal consumption of resources is given priority in certain time periods, compared to others.

In our example, the current and future time periods (in columns 4:7 of the schedule matrices) are the only time periods where the resource consumption is a concern i.e. time periods in the optimization time-line L_+, the columns 1:3 occur in the past. For purposes of example we define $\hat{\omega}$ to be:

$$\hat{\omega} = \begin{bmatrix} 1000 & 800 & 500 & 100 \end{bmatrix}$$

By multiplying the values of $\omega(t)$ against $(r(S,t) - R(t))$ we then create a differential weighting (or penalty function effect) for the various values of $(r(S,t) - R(t))$. As the algorithm iterates through candidate solutions, those solutions that provide better results in the time periods carrying higher weightings will be selected. This enforces the critical time window requirement.

Now that all the necessary elements have been computed or initialised, we are able to evaluate the objective function with respect to S_0 and S_1.

7.1.8 Objective function calculation. We recall that in order to perform the calculation in Equation 4.2.11, the following needs to be in place:

- The resource consumption requirements $r(S_0, t)$ and $r(S_1, t)$ of S_0 and S_1 need to be calculated based on the activity resource consumption profiles.
- The resource capacity per time period, $R(t)$ needs to be established (based on an input).
- Some penalty condition ω must be initialised.

For this example, we now calculate the values of the objective function f for schedules S_0 and S_1 with corresponding resource consumption requirements $r(S_0)$ and $r(S_1)$.

We recall from the definition of the CTW objective function in Equation 4.2.11 for a single resource:

$$f(S_0) = \sum_{t \in L_+} \omega(t).(r(S_0, t) - R(t))^2 \tag{7.1.6}$$

$$f(S_1) = \sum_{t \in L_+} \omega(t).(r(S_1, t) - R(t))^2 \tag{7.1.7}$$

For this example, there is only a single resource so $k = 1$. Also, in order to align the dimensionality of all the calculation elements, we define ω as:

$$\omega = \begin{bmatrix} 0 & 0 & 0 & 1000 & 800 & 500 & 100 \end{bmatrix}$$

Then performing the calculation for $f(S_0)$ and $f(S_1)$ for the values provided in this example, we obtain the results: $f(S_0) = 4199980.5$ and $f(S_1) = 7466435.833$. The Simulated Annealing algorithm will then select either S_0 or S_1 based on the criteria described in algorithm 2, line 12 - line 17. By performing many iterations, the algorithm will select schedules such that the results aligned with the desired resource consumption profile.

This concludes the presentation of the calculation steps in our model. We will now provide an overview of the chosen optimization algorithm Simulated Annealing and the Continual Rescheduling approach.

7.2 Features of Scheduling Algorithm

Now that we have covered the computational steps necessary to calculate the objective function value for various trial solutions, we discuss in detail the design of the scheduling method. This brings together the theoretical concept of the critical time window resource levelling with continual rescheduling problem, with the practical requirement of scheduling of activities for the collections department in the real world problem.

7.2.1 Optimization Method. The method selected to perform the optimization computation is Simulated Annealing (SA), as discussed in Chapter 6. The SA process is run on a daily basis. The new information is input into the algorithm, the optimization is then performed to obtain the optimal schedule based on current and historic information and the new schedule of actions is calculated. The specific features of the algorithm are discussed later in this chapter. We now describe how the SA algorithm presented in Chapter 6 is applied to our problem. In Chapter 6 we described the elements of the generic Simulated Annealing algorithm. We specifically recall the parameter inputs, namely cooling schedule elements::

- T_0, the initial temperature of the system.
- ϵ the terminal condition for the cooling schedule (once the temperature is reduced below this value, stop iterating the algorithm.
- M the Markov Chain length condition, which controls the number of solution search iterations performed at each temperature T_t.
- α the factor by which the temperature T_t is reduced to generate T_{t+1}.

When selecting the values for the cooling schedule, several factors need to be taken into consideration. In the implementation of the CTWRLP to address the needs of the real world problem, a significant consideration was that the algorithm should not take too long to execute. This is fundamentally informed by selection of cooling schedule parameters. By means of suitable numerical study, we have chosen values for the cooling schedule to satisfy the degree of accuracy as well as run time duration of the algorithm as follows:

- $T_0 = 10000$.
- $\epsilon = 0.001$.
- $M = 200$.
- $\alpha = 0.97$.

These values for the cooling schedule have been used in the implementation of the CTWRLP Simulated Annealing model.

7.2.2 Continual rescheduling feature. The scheduling algorithm is designed in such a way that a new schedule is created daily, incorporating the most recently-available information relating to call center capacity and the number of accounts requiring scheduling in the collections environment (broken down into cash-paying and direct debit customers and then into the high, medium and low risk categories. A total of 6 categories).

In practical terms, the concept of continual rescheduling is addressed as follows: Every scheduling phase (daily), the previous schedule is discarded. Only the history is retained - i.e. the accounts that have already been released into the collections strategies which will have an effect on the capacity resource in the current and future time periods.

The critical time window resource levelling feature is used to ensure that the schedule optimally utilizes (and does not exceed) the capacity in the closest time periods of the schedule, while capacity in the future may be over or under utilized.

Effectively, a practical requirement in the implementation of the continuous rescheduling feature is that the algorithm should keep and update a history. For example, if on a given day t the schedule indicates that 1000 accounts should be released into a particular collections strategy, then on the following day, the algorithm needs to know that in the previous time period $t-1$ 1000 accounts were released and the response impact of those accounts needs to be taken into consideration when creating the schedule for the current day.

The scheduling horizon will be defined by length of time it will take after which no further responses (activities requiring resources) occur as a result of performing actions at the current scheduling phase t. In the real world problem, the longest response profile is 15 days from the

start of the collections strategy, therefore the algorithm will keep 15 days worth of 'history' and discard the data older than 15 days. It will also consider a forward looking forecast of 15 days, The scheduling horizon is thus 30 time periods.

During each scheduling phase, the latest information about the activities requiring scheduling is captured. The previous schedule is completely discarded, retaining only the structure of the activities that have already been scheduled. A new critical time window resource levelled schedule is then created.

7.3 CTWRLP with Continual Rescheduling

In this section we now present the algorithm for the CTWRLP using the Simulated Annealing Method. In order to perform some of the computations relating to the practical implementation of this algorithm, several specific computational procedures were developed. These are described and presented in Chapter A and are referenced where required. In preparation of performing the computation, the following need to be defined:

- The scheduling horizon H_t - how many historic and forward looking periods need to be taken into account. All inputs and calculations will be informed by this dimension.
- The Critical Time Window penalty function $\omega(t)$ - the relative importance of the critical time window within the scheduling horizon.
- Resource constraints R - the capacity and availability of the various resources over the scheduling horizon.

In presenting the CTWRLP with Continual rescheduling method, we refer to the Simulated Annealing method presented in algorithm 1. We present a pseudo-code algorithm more specifically tailored to addressing the features of our problem.

7.3.1 Simulated Annealing for CTWRLP.

Algorithm 2: Critical Time Window Resource Levelling using Simulated Annealing

For each active scheduling phase H_t in rolling scheduling horizon H:;

Input: Activities J requiring allocation of resources (yet to be scheduled).

Input: Resource Consumption profiles C of each activity.

Input: Result from previous scheduling phase output $S_{H_{t-1}}$.

Input: Capacity (resource) availability R.

Output: Activities which need to be performed during the active time period and a provisional forward looking schedule for the remainder of the scheduling horizon.

Result: An optimal, critical time window resource levelled schedule S_{H_T} based on available data.

Begin Simulated Annealing Optimization Algorithm.;

Input Parameter: M **= number of iterations to attempt at each temperature**;

Input Parameter: T_0 **= initial temperature (high energy state)**;

Begin Simulated Annealing Optimization Algorithm.;

Align inputs so that historical schedule information, active schedule phase and critical time window periods are positioned correctly.;

while *Cooling schedule end condition not met for* T $(T_{t+1} \geq \epsilon)$ **do**

    Initialise first schedule trial solution S_0;

    for $m = 1$ **to** M **do**

        Generate random candidate solution S_1;

        Evaluate objective function according to CTWRLP objective function Equation 4.2.11;

        if $\delta E < 0$ **then**

            Accept candidate solution as new active solution and update configuration;

        else

            Accept candidate solution and update configuration with probability $P = \exp(-\delta E/T)$;

        end

    end

    Update (reduce) temperature parameter T in line with cooling schedule $(T_{t+1} = \alpha T_t)$.

end

7.4 Practical Demonstration of Algorithm

In this section, we demonstrate a worked example of the execution of the algorithm. This is a simulation of how the scheduling solution is applied in practice. The simulation will be based on data (averaged) for one month's worth of information. Six months worth of data was sampled and then averaged to create the data for the simulation.

7.4.1 Context for practical demonstration. The first step in assessing the practical implementation is to gain understanding of the context in which the scheduling algorithm will be applied. Figure 7.1 demonstrates the activities which are scheduled over the course of this simulation. Total accounts against which collections strategies will need to be scheduled is 186623 for the month. Figure 7.1 shows the amount of new activities that will require scheduling on each day of the month. It is important to note that in the real world implementation, no forward knowledge of events is available. We have information of the entire amount of activities that will need to be performed over the course of the execution of the schedule, however for the purposes of the simulation we execute each scheduling phase based only on the information from the corresponding state as shown in Figure 7.1. For the purposes of this simulation, it is assumed that there is no history (no collections actions already in progress). It is also be assumed that the daily capacity is uniform at 5000 calls per day.

Figure 7.2 illustrates the simulated inbound call volumes experienced over the course of the sample month (as a result of the activity profile in Figure 7.1). These simulated inbound calls are calculated based on the collections arrears events profile in Figure 7.1 multiplied by the modelled response rates (based on observation and data collection as discussed in subsection 5.2.5. It clearly shows that the 5000 calls per day capacity is significantly exceeded until day 18, where-after there is surplus underutilized capacity. This profile demonstrates the opportunity that could be realised if the inbound calls could be rebalanced over the entire scheduling period.

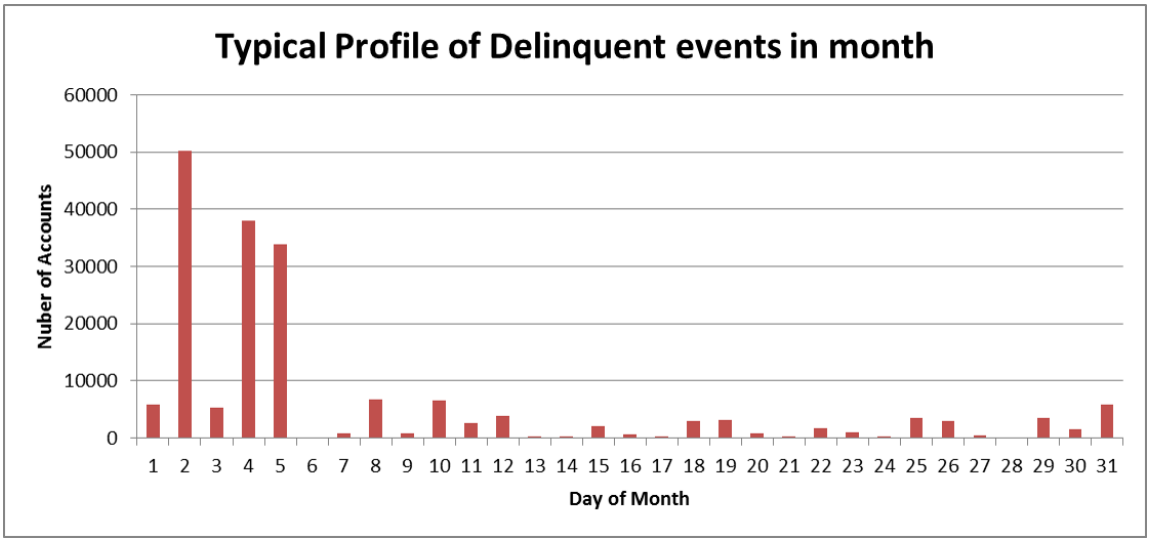

Figure 7.1: Typical profile of activities requiring scheduling in the collections environment.

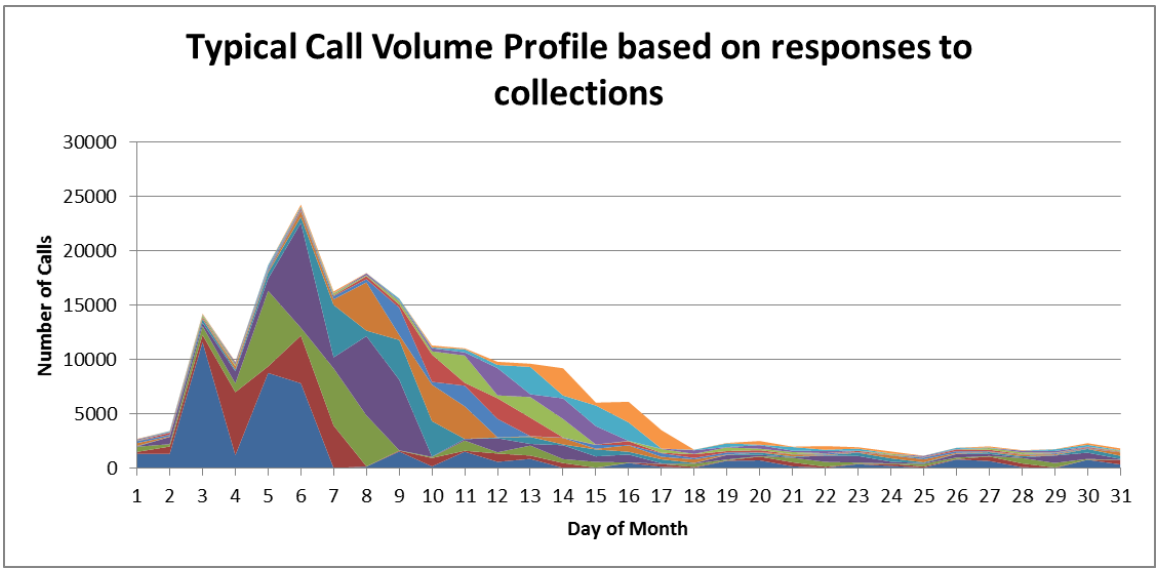

Figure 7.2: Inbound Call Volumes in unscheduled environment.

7.4.2 Results of CTWRLP algorithm implementation. The CTWRLP algorithm is now applied to the scenario described in the previous section. As a recap: The following is a summary of how the scheduling process is implemented for this simulation:

- In practice, scheduling takes place at the start of each business day. The output of the scheduling algorithm will be used to decide on the number of accounts to be released for collections actions on that day. The resulting schedule of activities to be released is captured and the modelled response rates will be used to calculate simulated inbound call volumes. The simulation will then advance to the next phase as if it is the following business day, and so on.
- The scheduling method is initialised using a scheduling horizon of 30 days, $\tau = 14$ (15 days historic, 15 days forward looking).
- The Critical Time Window is initialised such that the current scheduling phase (active day) is weighted most important.
- The call centre inbound call capacity is set at 5000 calls per day i.e. $R(t) = 5000$.

The resource consumption of each of the three collections strategies (Table 5.2) being scheduled for start is:

$$C = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.2 & 0.1 & 0.05 & 0.05 & 0.15 & 0.05 & 0.01 & 0.01 & 0.01 \\ 0.3 & 0.15 & 0.25 & 0.05 & 0.05 & 0.3 & 0.1 & 0.05 & 0.01 & 0.01 & 0.01 & 0 \\ 0.1 & 0.1 & 0.1 & 0.25 & 0.15 & 0.1 & 0.1 & 0.1 & 0.05 & 0.05 & 0 & 0 \end{bmatrix}$$

In this instance, the time dimension of the response matrix is less than the scheduling horizon (recall from subsection 4.2.2 that this may be the case).

The values for ω, $\kappa = 9$ were selected by suitable numeric study and were defined to be:

$$\omega = \begin{bmatrix} 1000 & 800 & 600 & 600 & 600 & 500 & 500 & 500 & 400 & 400 \end{bmatrix}$$

This ensures that the algorithm gives precedence to ensuring resource utilization is maximised for the current time period and successively reducing importance to time periods progressively further away from t_0.

The values for the cooling schedule were defined as described in subsection 7.2.1.

The results of the scheduling are presented in Table Table 7.1. Some measurements for the performance of the optimized vs. unoptimized case are also presented below the table. These illustrate the improvement in overall resource utilization in the optimized case. In Table 7.1 we see the unscheduled (un-optimised) scenario which corresponds to the visual representation of the inbound call volumes represented in Figure 7.2. Recall that if the total number of inbound calls on a given day exceeds 5000, this means all inbound calls which occur after the 5000 threshold are unanswered. In the optimized scenario, the scheduling algorithm has been run at the start of each day. In this case there are still some days where the total capacity is violated, but the degree of violation is very small and will not result in the catastrophic failure of service and low service levels which are experienced in the unoptimised case. We also see that the total amount of un-utilized capacity in the optimised case is less than in the unoptimised case, which points to a more efficient utilization of a non-renewable resource. The average capacity violation (calculated as the average of the number of calls which exceed 5000 on the number of days on which the capacity was exceeded) is also significantly improved in the optimized case, 202 versus 7872 in the un-optimized case.

		Un-Optimised			Optimized			
Schedule Phase	New Activities To Be Scheduled	Activities Performed (Un-Optimized)	Total Inbound Calls	Resource Surplus	Activities Performed (Optimised)	Activities Deferred (Buffer)	Total Inbound Calls	Resource Surplus
Day 1	5793	5793	2720	2280	5793	0	2720	2280
Day 2	5834	5834	3448	1552	5834	0	3448	1552
Day 3	50226	50226	14241	-9241	14357	35869	3994	1006
Day 4	5270	5270	9824	-4824	5186	35953	3666	1334
Day 5	37987	37987	18712	-13712	1146	72794	3972	1028
Day 6	33867	33867	24279	-19279	6034	100627	4869	131
Day 7	184	184	16287	-11287	6805	94006	4973	27
Day 8	735	735	17988	-12988	720	94021	3652	1348
Day 9	6735	6735	15636	-10636	1855	98901	3615	1385
Day 10	789	789	11317	-6317	13418	86272	4073	927
Day 11	6660	6660	11053	-6053	6733	86199	4595	405
Day 12	2631	2631	9816	-4816	4680	84150	4838	162
Day 13	3803	3803	9637	-4637	2918	85035	4864	136
Day 14	242	242	9223	-4223	2377	82900	4882	118
Day 15	208	208	6056	-1056	10659	72449	5928	-928
Day 16	2134	2134	6138	-1138	3160	71423	4746	254
Day 17	660	660	3521	1479	3500	68583	4929	71
Day 18	286	286	1724	3276	1931	66938	5032	-32
Day 19	3040	3040	2346	2654	3572	66406	4390	610
Day 20	3204	3204	2525	2475	15239	54371	4909	91
Day 21	837	837	2008	2992	4227	50981	4603	397
Day 22	280	280	2059	2941	5172	46089	5117	-117
Day 23	1685	1685	1954	3046	8142	39632	4794	206
Day 24	986	986	1565	3435	4303	36315	4932	68
Day 25	335	335	1171	3829	12092	24558	5232	-232
Day 26	3576	3576	1903	3097	4115	24019	5013	-13
Day 27	2972	2972	2031	2969	3925	23066	4755	245
Day 28	507	507	1679	3321	2931	20642	4552	448
Day 29	126	126	1772	3228	16084	4684	5113	-113
Day 30	3444	3444	2327	2673	4938	3190	5133	-133
Day 31	1587	1587	1848	3152	4204	573	5047	-47

	Un-Optimized	Optimized	Notes
Root Mean Squared Measure	6631	766	
Average Capacity Violation	-7872	-202	Average of capacity violation only
Average Capacity Underutilization	2847	619	Average of surplus capacity only
Capacity Violation as % Of Capacity	-157%	-4%	
Total Slack (Underutilised) Capacity	48399	14227	

Table 7.1: Results of CTWRLP Algorithm Scheduling

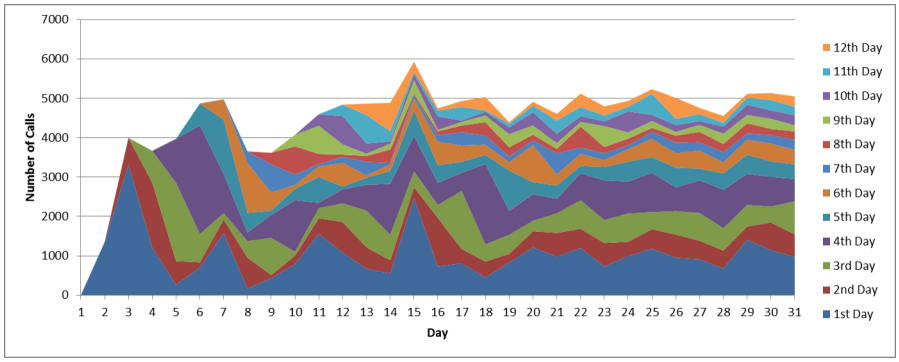

Figure 7.3: Inbound Call Volumes in Scheduled Environment.

In Figure 7.3 we see a graphical representation of call volumes from the scheduled / optimized scenario. The total call volumes track closely to the target 5000 calls per day. The different colours in the chart show the number of calls as a result of collections actions which were taken on a certain date. So for example the blue color labelled '1st Day' shows the number of calls received on a given day as a result of the collections actions taken that day. The '2nd Day' color shows the number of calls that were received as a result of actions taken the previous day, and so on. When we compare the results shown in Figure 7.3 it is clear to see that the capacity utilisation is more uniform than in Figure 7.2 and the violation (exceeding) of available capacity has been improved.

Finally, Figures 7.4 - 7.9 show the comparative resource consumption per schedule phase (for three consecutive scheduling phases on Day 2 - 4) of a scheduled vs. unscheduled environment, based on the same inputs. We see that the scheduling system works to control the resource consumption and keep it close to the 5000 mark where the unscheduled environment sees huge resource availability violations.

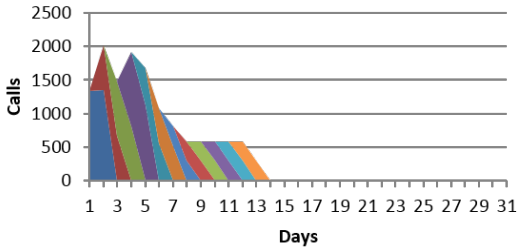

Figure 7.4: Day 2 - Unscheduled case response profile

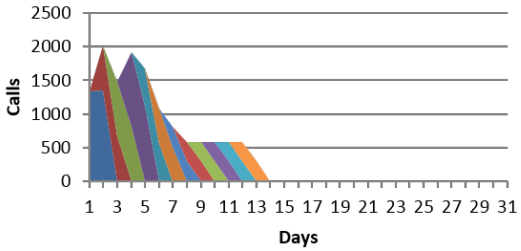

Figure 7.5: Day 2 - Scheduled case response profile

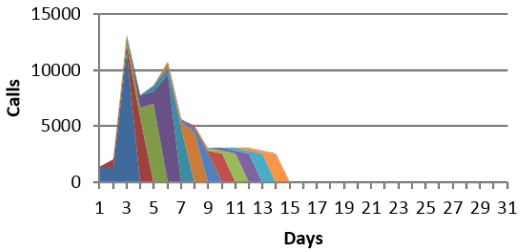

Figure 7.6: Day 3 - Unscheduled case response profile

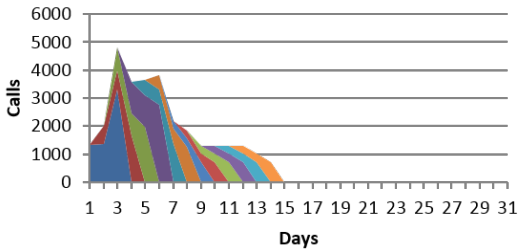

Figure 7.7: Day 3 - Scheduled case response profile

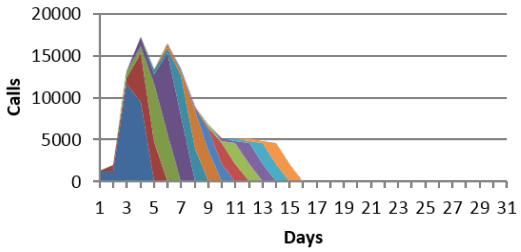

Figure 7.8: Day 4 - Unscheduled case response profile

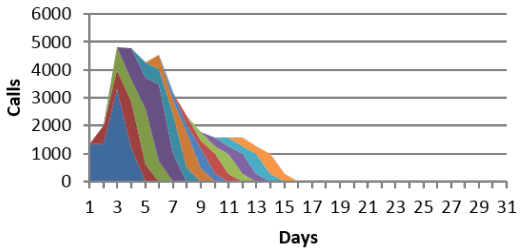

Figure 7.9: Day 4 - Scheduled case response profile

7.4.3 Results of implementation. The CTWRLP algorithm was implemented as a solution to the challenges experienced by the credit and risk team in a South African telecommunications service provider. It was implemented in April 2011. It has been in operation in that environment ever since then and to the time this document has been written. The most dramatic improvement can be observed in Figure 7.10, which shows the

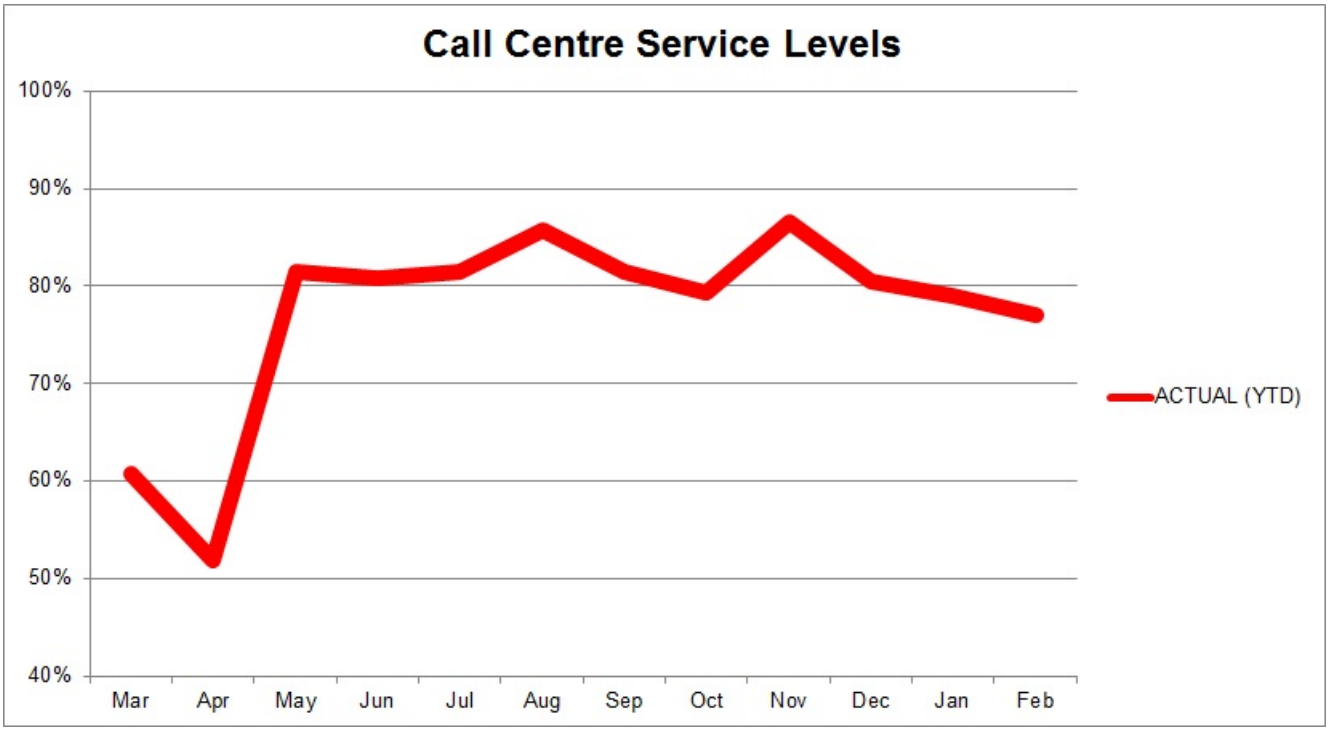

Figure 7.10: Credit and Risk Call Centre Service level. *Extracted from Financial Year F12 Final Executive Report.*

improvement in service level immediately post implementation.

We can clearly see that the service level improves immediately post implementation. The chart shows the actual (year to date) call centre performance for the the environment subsequent to the implementation of the optimization method. Considerable effort was taken to ensure the necessary process and systems were implemented to support the ongoing success of the scheduling system. A team of analysts was set up who were responsible for maintaining the response models, the workforce planning models and the call centre capacity models. Since then, several enhancements have been made to the collections environment with the introduction of self-help services in the form of an SMS portal, an on-line channel, smart phone applications and automatic voice response systems. The algorithm has proven resilient to these changes, with only updates to the response rates required in order to adapt to the evolving environment. An additional benefit was realised in that as call volumes were reduced and load balancing was implemented through scheduling, it was possible to reassign human resources to the late stage manually intensive collections operation. This resulted in

a reduction in bad debt!

7.5 Summary

In this chapter, we have described how the practical problem was addressed using the CTWRLP model and continual scheduling method. We gave an overview of the algorithm, discussed the optimization method that was used to address the scheduling model and also provided a review of the results of a practical implementation of the scheduling method. In the next chapter we will provide the conclusion of this dissertation, including some discussion around points such as future research and other potential applications of this method.

Chapter 8

Conclusion

In conclusion of this dissertation, we will discuss some ideas relating to where this new scheduling technique may be applied. We also conclude the dissertation with a summary and review of the various aspects which have been covered.

8.1 Conclusion and Discussions

8.1.1 Conclusion. This research arose out of the need to discover a solution to a practical problem experienced by a credit and risk call centre. The objective of the research was to study the problem and explore the attributes of the problem (in the context of available literature and research relating to problems of a similar nature).

The focus of the research was then to formulate the problem in mathematical terms (by developing a new model or extend existing models), and then apply an optimization technique to solve the problem. The final phase of the research was to implement the model in the call centre and observe the effects.

The practical problem relates to the inability of the call centre to meet service level requirements, resulting in a large number of calls being dropped (unanswered) and associated issues of poor customer service and potential bad debt exposure. The problem is made

complex by the uncertainty surrounding the number of activities which will enter the call centre system on a given day, and the ongoing interactions of the activities which have previously been performed with current and future resources. Further investigation into the problem revealed a complex system of interacting components namely: Collections activities and strategies, customer response rates, work force planning and call centre capacity planning. These elements have been shown to align with the components of scheduling problems.

A review of the literature relating to scheduling problems was conducted and the literature and research pertaining to scheduling problems has been found to be extensive. Scheduling is a well-studied field due to the fact that optimised processes provide significant cost benefit in many industries. Scheduling techniques are used to address a broad array of practical applications, ranging from logistics and construction through to the scheduling of processes on integrated circuits in electronics such as computer central processing units. As a result there has been continuous focus on research into scheduling and associated methodologies for over 50 years.

Based on our literature review findings, we have shown that there are two main classes of scheduling problems which exhibit strong similarities to the practical problem being studied in this research; namely the Resource Constrained Project Scheduling problem (RCPSP) and the Resource Levelling Problem (RLP).

In RCPSP the objective is to minimise overall project duration, given a set of activities with certain precedence constraints and resource consumption requirements. There are many extensions and variations of RCPSP which deal with the way in which activities are performed, how resources are treated and objective function models to satisfy various requirements of the RLP family of problems.

In RLP the main consideration is the creation of a schedule which results in a uniform or flat resource utilisation over the time-line of the schedule. The relevant attributes of these models formulations have been discussed and the practical problem has been shown to exhibit many similarities to these, but was not of a structure that would lend itself to complete formulation using either of these classes of models.

The main finding of the literature review is that there is currently no complete model or extension of RLP or RCPSP which could be used to address the practical problem in the call centre. This informed the research objective of developing a new model to address this specific problem.

In line with the research questions and objectives, an extension of RLP was then developed in order to address the requirements of this problem. We presented and introduced a new objective function model called the Critical Time Window Resource Levelling Problem and also the introduction of a new method and approach called Continual Rescheduling. These two features were presented as extensions to the resource levelling problem.

The Simulated Annealing optimization method was introduced and the reasons for choosing Simulated Annealing as the optimization method for solving the CTWRLP model were listed.

We have then shown that the CTWRLP model with continual rescheduling method could be successfully used to schedule the collections activities to achieve an optimal service level profile. Finally we showed the successful implementation of the method to solve the practical problem faced by the call centre. The dramatic improvement in service level - from below 60% to above 80% - provides convincing evidence of the success of the implementation. The call centre now meets service level requirements more efficiently, and this in turn has improved the businesses customer service and reduced its bad debt exposure too.

8.1.2 Discussion. This research featured an academic exploration of scheduling models as well as an industrial engineering study into workforce planning and optimization, culminating in results in both areas.

In the academic sense, a new model was developed and introduced to the field of mathematical scheduling. And in the industrial engineering study, a practical solution was developed and applied to the environment (i.e. the credit and risk department), resulting in a significant improvement to performance.

The key finding has been that a scheduling method could indeed be developed to address specific features of the real world problem. The fact that it was used to solve the problem

and address the practical issues validate the author's hypothesis that a scheduling method could be used to smooth call volumes over the course of a month without sacrificing the number of activities which needed to be performed.

The research considered the service level target only. Minimisation of bad debt (which is the second main objective of the credit and risk department) was only implicitly considered, in that we sought to take action on all accounts as quickly as possible, given certain call centre capacity limitations.

This gives rise to possible further study in extending the critical time window resource levelling problem to a multi-objective or rather multiple resource model. The concept would be to develop a model that takes into account all the features of a critical time window model yet is also able to handle additional targets such as minimizing bad debt. This could be achieved by modelling bad debt as an additional resource which would need to be consumed or alternatively dealt with as a completely separate objective function in the optimization computation. It may also be relevant to further study the optimization method used to create the schedule. Alternate methods such as genetic algorithms or particle swarm optimization could potentially achieve faster or more accurate solutions which would be beneficial in the practical implementation of a scheduling solution.

In terms of the industrial engineering problem, further research could be done to identify areas which may benefit from the practical application of the critical time window resource levelling method. We could speculate where there may be opportunity and first recall the key features of the algorithm:

- The method is applicable to an environment where the main concern would be the optimal interaction of activities to be performed and the resources required for these activities, during a particular phase of the scheduling period. In our problem, it was most important that the resources were fully consumed in the period close to the time where the scheduling was performed, but there could be other applications where the resource consumption in the future was more important than in the near term.
- The method is less concerned with minimizing total duration of a project and more

concerned with optimally allocating resources, provided that there is overall sufficient capacity in the system to address all the activities within a given time (this is implicit in the set up and structure of the method). There is some capability to postpone or buffer the start of activities according to certain macro time-line conditions. For example, in the case of the collections environment, there is sufficient capacity over the period of a month to deal with all activities that require scheduling. Exception reporting could also be used to monitor the number of activities awaiting release for processing and how long they have been waiting. Special campaigns or short term staff could be brought in to address periodic backlogs.

- The method applies to an environment where there is some set resource availability per time period. If the resource is not fully utilized in each period then it is wasted.
- The method applies to an environment where there is no specific start or end to the scheduling horizon, the schedule needs to roll continuously in an environment where the demand comes in continuously. The most optimal utilization of resources needs to be established each day, based on the new activities requiring scheduling and the activities that have already been scheduled in the past which are still consuming resources.
- The method caters for a situation where activities which have been performed in the past continue to consume resources in current and future time periods.

Given the above features, we propose that there may be opportunity to explore the CTWRLP method in the following practical fields:

- Construction project scheduling, where there is a fixed workforce and variable supply of construction materials; the construction activities could be scheduled based on the workforce and currently available building materials. As new materials are delivered, a new schedule is created to optimally utilize all available resources (based on precedence and reasonableness constraints). Implicit in this approach is that the project makespan will be minimised as a result of all resources being fully utilized.
- High-Tech manufacturing: In modern high-tech manufacturing facilities, multiple SKUs are produced at the same site using both specific and common resources, as

well common manufacturing equipment (3D-Printing for example) and workforce. The demand is variable and based on the inflow of orders. A scheduling algorithm which ensures maximum utilisation of all resources will benefit the manufacturing company in ensuring cost efficiency.

- Clinical Pharmacology and Medicine: There may be an application for the method in the medical field. If the interaction of certain drugs with each other, as well as with the patient are well known and reliably modelled (factors such as peak plasma concentration, half-life, etc) then the CTWRLP method may work in assisting the creation of a dosing schedule for a patient where the interaction of various drugs may need to be taken into consideration.
- Exercise Science and Professional Athlete training: With the advent of modern technology and technology assisted training for professional and recreational athletes, there is now a capability to measure and model the effects of various exercise and physical stimulus on the body in terms of the amount of recovery needed for the athlete, as well as the increase in performance (cardiovascular and strength for that athlete). If we consider that the body has a finite 'recovery' resource and the impacts of the various exercise types are well known (say for example the impact of running versus lifting heavy weights) then the CTWRLP method may be used to create a training schedule for an athlete such that performance is maximised while never exceeding a certain recovery threshold.

Finally specific improvements to the solution implementation in the credit and risk call centre could be made.

Applications of machine learning could be used to create models that automatically update the models which inform the response rates of customers to collections actions. Machine learning techniques could also be used to deduce and quantify the inbound calls which are 'noise'-related rather than related to actions taken by the collections team. Additional work could be done to refine the segmentation to achieve better results with these models.

The idea of introducing bad debt as an additional non-renewable resource could also be

explored. This would mean an extension of the model to a multi-resource case. The idea would be to model the risk associated to bad debt exposure of each of the delinquent accounts as a resource. The scheduling algorithm would then be steered towards prioritising actions to be taken against accounts which would have a bigger impact on the bad debt resource. The balance between ensuring optimal utilization of the call centre resources and minimizing bad debt exposure would need to be managed.

The model also only operates at a daily level and certain times of day present challenges, such as the hours around lunch and tea breaks. Further research could be conducted to assess the feasibility of modelling responses at an hourly (intra-day) level and performing the scheduling at this level.

This model and example illustrate a successful practical adaptation of scheduling methods and the implementation of a scheduling solution. Within the context of the challenges of scale and the adoption of digitisation to solve business problems, this solution presents a glowing example of how the application of innovative scientific methods can create efficiency and optimise costs.

References

E. Aarts and J. Korst. *Simulated Annealing and Boltzman Machines: A stochastic approach to computing*. Wiley, 1989.

R. Anderson. *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*. OUP Oxford, 2007. ISBN 9780191527654.

M. Bagheri and S. Hessam Mahmoudi. Resource leveling in fuzzy project scheduling. In *Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*, Istanbul, Turkey, 2012.

K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource constraine project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149:268 – 281, 2003.

P. Brucker and S. Knust. *Complex Scheduling*. GOR-Publications, 2006.

P. Brucker, A. Drexlb, R. Mohringc, K. Neumannd, and E. Pesche. Resource constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112:3 – 41, 1999.

C. Cavalcante, C. Carvalho, de Souza, M. Savelsbergh, Y. Wang, and L. Wolsey. Scheduling projects with labor constraints. *Discrete Applied Mathematics*, 112(1):27 – 52, 2001. Combinatorial Optimization Symposium, Selected Papers.

A. Damci and G. Polat. Impacts of different objective functions on resource leveling in construction projects: a case study. *Journal of Civil Engineering and Management*, 20: 537 – 547, 2014.

E. L. Demeulemeester and W. S. Herroelen. *Project scheduling: a research handbook*. Kluwer Academic Publisher, Boston, 2002.

K. El-Rayes and D. Jun. Optimizing resource leveling in construction projects. *Journal of Construction Engineering and Management*, 135:1172 – 1180, 2009.

L. Florez, D. Castro-Lacouture, and A. L. Medaglia. Sustainable workforce scheduling in construction program management. *Journal of the Operational Research Society*, 64: 1169–1181, 2009.

S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource constrained project scheduling problem. *European Journal of Operational Research*, 207:1 – 14, 2010.

T. Hegazy. Optimization of resource allocation and levelling using genetic algorithms. *Journal of Construction Engineering and Management*, 125:167 – 175, 1999.

P. Jernstrom. Multi-product multi-purpose machine scheduling in dynamic environments. Technical report, Process Design laboratory, Department of Chemical Engineering, Faculty of Technology, Abo Akademi University, Finland, 2006.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942 – 1948, Perth, Australia, 1995.

C. Kirkpatrick, D. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

G. Koole. *Call Centre Mathematics, A scientific method for understanding and improving contact centres*. MG books, Department of Mathematics, Vrije Universiteit Amsterdam, 2013.

S. Kumanan, G. Jegan Jose, and K. Raja. Multi project scheduling using an heuristic and genetic algorithm. *International journal of Advanced Manufacturing Technology*, 31:360 – 366, 2006.

C. Mendez, J. Cerda, I. Grossman, I. Harjunkoski, and M. Fahl. State of the art review of optimization methods for short term scheduling of batch processes. *Computers and Chemical Engineering*, 20:913 – 946, 2006.

M. Mika, G. Waligora, and J. Weglarz. Simulated annealing for multi mode resource constrained project scheduling. *Annals of Operations Research*, 102:137 – 155, 2001.

A. Najafi. Using multi objective particle swarm optimization for bi-objective multi mode resource constrained project scheduling problem. *World Academy of Science, Engineering and Technology*, 78, 2011.

W. Nuitjen and E. Aarts. A computational study of constraint satisfaction for multi capacitates job shop scheduling. *European Journal of Operational Research*, 90:269–284, 1996.

B. Odedairo and V. Oladokun. Relevance and applicability of multi-objective resource constrained project scheduling problem review article. *Engineering, Technology and Applied Science Research*, 1, 12 2011.

R. Petrović. Optimization of resource allocation in project planning. *Operations Research*, 16(3):559–568, 1968.

J. Ponz-Tienda, A. Salcedo-Bernal, E. Pellicer, and J. Benlloch-Marco. Improved adaptive harmony search algorithm for the resource leveling problem with minimal lags. *Automation in Construction*, 77(Supplement C):82 – 92, 2017.

R. Rutenbar. Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine*, 5:19–26, 1989.

B. Yang, J. Geunes, and W. J. O'brien. Resource-constrained project scheduling: Past work and new directions. Technical report, Department of Industrial and Systems Engineering, University of Florida, 2001.

Appendix A

MATLAB Code

In this appendix, detailed descriptions and examples for the computational steps and concepts in the dissertation are provided. These examples will include excerpts of Matlab code, as well as the results of certain computational steps performed in Matlab.

A.1 Objective Function (Critical Time Window) Computation

The Critical Time Window Objective function computation is now described. The implementation of the computation for the critical time window computation is presented in this section. We demonstrate this using a scheduling scenario where 3 (represented by matrix dimension m) distinct categories will need to be scheduled and their respective resource consumption profile is modelled over 12 (represented by matrix dimension l) periods. A single resource (capacity) will be considered, represented by C a $1 \times n$ vector. The rolling schedule time horizon will be 30 (represented by matrix dimension n) time periods, 15 for history and 15 for the new schedule.

The first step in calculating the objective function is to calculate the expected responses for the active and candidate solutions. This is achieved by performing matrix multiplication.

For an $m \times n$ matrix of activities to be scheduled, V and an $m \times l$ matrix of responses / resource consumptions R, the total response / resource consumption is $R_{Tot} = R'V$ where R_{Tot} is an $l \times n$ matrix.

Now the expected daily volumes need to be calculated. This is done by performing a summation over R_{Tot} so that the total responses per day are calculated. The matlab code to perform this operation is as follows:

```
function [ G ] = ExpectedDailyVolumes(RT)


%Calculate Daily Responce Volumes for each Event and Response Type
%   RT is the total response matrix
G = zeros(1,size(RT,2));


for i = 1:size(RT,2) %30
    for j = 0:size(RT,1)-1 %12
        if mod(i-j,size(RT,2))==0
            col = size(RT,2);
        else
            col = mod(i-j,size(RT,2));
        end
        V = RT(j+1,col);
       G(i) = G(i) + V;
    end
end
```

The output of this step is a $1xn$ vector G of total expected daily volumes. The final step is to now calculate the objective function value for the activity schedule matrix V. This is done with the following function:

```
function [ OF_val , wSoln , Soln ] = Obj_Fnc( RT,CAP )
%The objective function to calculate the effectiveness of a distribution of
%volumes
```

```
%   This objective function take the expected response rates,
%   compares it to the available capacity with certain weighting and criteria,
%   in an attempt to ensure that the inbound volumes match capacity as
%   closely as possible for the following few days.

%RT is the expected volume of inbound calls
%Capacity is the capacity vector

%OF_val is the calculted objective function value
%Soln is the solution vector of expected call - capacity

%define critical time window weighting:
w1 = [1000,800,600,600,600,500,500,500,400,400,300,300,100,100,100];
w = cat(2,zeros(1,15),w1);
wSoln = zeros(1,size(CAP,2));
Soln = wSoln;
%Initialise objective func val:
OF_val = 0;

for i = 1:size(CAP,2);
    if RT(i) == 0
        wSoln(i) = 0;
        Soln = 0;
    else
        wSoln(i) = (abs(CAP(i) - RT(i)))*w(i);
        Soln(i) = (abs(CAP(i) - RT(i)));

    end
    OF_val = OF_val + wSoln(i);
end
```

```
end
```

The objective function value is calculated for the active and candidate schedule matrices and the winner is selected based on the criteria presented in the algorithm description.

A.2 Algorithm Computation Steps

The computation of the various components of the algorithm are presented in this section. In order to arrive at the calculation of the objective function value the following steps need to occur:

1. Generate a candidate schedule of activities.
2. Calculate the resource consumption profiles corresponding to activity schedules.
3. Calculate the total resource consumption per time period.
4. Compute the objective function value for each of the resource consumption matrices.

The algorithm detailing the computational steps involved in implementing the SA algorithm is as follows:

Algorithm 3: Critical Time Window Resource Levelling using Simulated Annealing

Begin Simulated Annealing Optimization Algorithm.;
Set Search Parameters - Markov Chain Length and Cooling Schedule Criteria.;
while *Cooling schedule end condition not met for* T **do**
    repeat
        Create Candidate Activity Matrix V_{cd} - refer to section A.3
        Calculate the resource consumption matrix R and R_{cd} for the original activity matrix V and the candidate activity matrix V_{cd} respectively.;
        Calculate total resource consumption per time period.;
        Calculate objective function values O and O_{cd} for each of the resource consumption matrices R and R_{cd};
        Test fitness of objective function solution - refer to section A.1
        if $O_{cd} < 0$ **then**
            Set $V = V_{cd}$ Candidate schedule becomes active schedule when candidate objective function value is better than active objective function value.
        else if $X < \exp(-(O_{cd} - v)/T)$ *where* X *is a random variable uniformly distributed on* $[0, 1]$ **then**
            Set $V = V_{cd}$ Candidate schedule becomes active schedule if candidate objective function value is worse than active objective function value but Boltzmann probability condition is satisfied.
        else
            Retain V as current active schedule
        end
    until *Markov chain length criteria met*;
end

A.3 Candidate Solution Creation

In the algorithm description, there is a step where a candidate solution needs to be created based on an active solution. The design of the algorithm relies on the ability to generate a random solution and test the cost (objective function result) against other solutions. Practically, generating a random solution means creating schedules where the total number of activities remains constant across various candidate solutions, but the order in which the activities are performed (or the number of activities performed on a given day of a schedule) varies.

We achieve this by creating a Matlab function called *randwalkma*. The function takes as input an $m \times n$ matrix X and produces an $m \times n$ matrix Y as output, where the sum of the values of each row in X and Y are the same, but the values of the elements have been adjusted in some random way. The Matlab code for the function is as follows:

```
function [Y] = randwalkma(X)
%Function to randomly distribute values across rows of matrix.
%Used to create candidate schedule solution.
A = size(X);
%Determines dimensions of matrix X.
for n = 1:A(1)
%Repeat for each row of X,;
    X_old(n,:) = X(n,:);
%Create record of original values.
    flag = zeros(A(2));
    dims = ceil(A(2)*rand);
%Select a random number of columns between 0 and total columns in X;
%This is the total number of columns that will be adjusted;
    delta = zeros(dims,1);
    for i = 1:dims
        r = ceil(A(2)*rand);
```

```
        while flag(r) ~= 0;
           r = ceil(A(2)*rand);
        end
        flag(r) = 1;
        delta(i) = rand*X_old(n,r);
        X(n,r) = X_old(n,r)-delta(i);
    end
%Select a random column in the active row of X.
%Reduce the value in that column by some random amount.
%Repeat until the total number of columns adjusted is equal to the "dims"
    %parameter.
    flag = zeros(A(2));
    for i = 1:dims
                r = ceil(A(2)*rand);
        while flag(r) ~= 0;
           r = ceil(A(2)*rand);
        end
        flag(r) = 1;
        X(n,r) = X(n,r)+delta(i);
    end;
%Select a random column in the active row of X.
%Increase the value in that column by the value
    %which was used to reduce the values in the previous step.
%Repeat until the total number of columns adjusted is equal to the "dims"
%parameter.
    Y = X;
end
end
```

We demonstrate the effect of the function on a 3×5 matrix. Command line execution as follows:

```
A = [1000,0,0,0,0;1000,0,0,0,0;1000,0,0,0,0]
A1 = randwalkma(A)
A2 = randwalkma(A1)
A3 = randwalkma(A2)
A4 = randwalkma(A3)
A5 = randwalkma(A4)

SumA = sum(A,2);
SumA1 = sum(A1,2);
SumA2 = sum(A2,2);
SumA3 = sum(A3,2);
SumA4 = sum(A4,2);
SumA5 = sum(A5,2);

Check = [SumA,SumA1,SumA2,SumA3,SumA4,SumA5]
```

Outputs as follows:

```
A =

        1000           0           0           0           0
        1000           0           0           0           0
        1000           0           0           0           0



A1 =

   1.0e+03 *

    1.0000         0         0         0         0
    0.7164         0         0         0    0.2836
```

```
    0.4248         0    0.5752         0         0


A2 =

  488.8936         0         0  511.1064         0
  627.7900         0   88.6146         0  283.5954
  113.0550  311.7614  575.1836         0         0


A3 =

  488.8936         0         0  511.1064         0
  627.7900         0   85.9216    2.6929  283.5954
  325.0230   99.7934   85.8870         0  489.2966


A4 =

  488.8936         0         0  511.1064         0
   93.7752         0   81.0402  540.1269  285.0578
  350.6874   99.7934   60.2227         0  489.2966


A5 =

  488.8936         0         0  511.1064         0
  143.2122   47.6189  381.8060  239.3610  188.0018
  578.9501   99.7934   30.8683         0  290.3882
```

```
Check =

   1.0e+03 *

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

Each of the matrices A, A1, A2, A3, A4 and A5 would be candidate schedules. Objective function values for each candidate schedule matrix are calculated and the fitness is assessed as per the algorithm description.