SCHEDULING COAL HANDLING PROCESSES USING METAHEURISTICS

by

David Gideon Conradie

Submitted in partial fulfilment of the requirements for the degree of

MASTER OF ENGINEERING (INDUSTRIAL ENGINEERING)

in the

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY

UNIVERSITY OF PRETORIA
PRETORIA
SOUTH AFRICA

April 2007

Executive Summary

SCHEDULING COAL HANDLING PROCESSES USING METAHEURISTICS

by

David Gideon Conradie

Supervisor: Johan W. Joubert
Department: Department of Industrial and Systems Engineering
Degree: Master of Engineering (Industrial Engineering)

The operational scheduling at coal handling facilities is of the utmost importance to ensure that the coal consuming processes are supplied with a constant feed of good quality coal.

Although the Sasol Coal Handling Facility (CHF) were not designed to perform coal blending during the coal handling process, CHF has to blend the different sources to ensure that the quality of the feed supplied is of a stable nature. As a result, the operation of the plant has become an extremely complex process. Consequently, human intelligence is no longer sufficient to perform coal handling scheduling and therefore a scheduling model is required to ensure optimal plant operation and optimal downstream process performance.

After various attempts to solve the scheduling model optimally, i.e. with exact solution methods, it was found that it is not possible to accurately model the complexities of CHF in such a way that the currently available exact solvers can solve it in an acceptable operational time.

Various alternative solution approaches are compared, in terms of solution quality and execution speed, using a simplified version of the CHF scheduling problem. This investigation indicates that the Simulated Annealing (SA) metaheuristic is the most efficient solution method to provide approximate solutions.

The metaheuristic solution approach allows one to model the typical sequential thoughts of a control room operator and sequential operating procedures. Thus far, these sequential rules could not be modelled in the simultaneous equation environment required for exact solution methods.

An SA metaheuristic is developed to solve the practical scheduling model. A novel SA approach is applied where, instead of the actual solution being used for neighbourhood solution representation, the neighbours are indirectly represented by the rules used to generate neighbourhood solutions. It is also found that the initial temperature should not be a fixed value, but should be a multiple of the objective function value of the initial solution. An inverse arctan-based cooling schedule function outperforms traditional cooling schedules as it provides the required diversification and intensification behaviour of the SA.

The scheduling model solves within 45 seconds and provides good, practically executable results. The metaheuristic approach to scheduling is therefore successful as the plant complexities and intricate operational philosophies can be accurately modelled using the sequential nature of programming languages and provides good approximate optimal solutions in a short solution time. Tests done with live CHF data indicate that the metaheuristic solution outperforms the current scheduling methodologies applied in the business.

The implementation of the scheduler will lead to a more stable factory feed, which will increase production yields and therefore increase company profits. By reducing the amount of coal re-handling (in terms of throw-outs and load-backs at mine bunkers), the scheduler will reduce the coal handling facility's annual operating cost by approximately R4.6 million (ZAR).

Furthermore, the approaches discussed in this document can be applied to any continuous product scheduling environment.

Keywords: Scheduling, Simulated annealing, Metaheuristics, Approximation algorithms, Multiple-objective programming, Stochastic programming, Industry application, Coal handling, Coal blending, Coal homogenization

Nomenclature

Bleed-in Coal from strategic stockpiles thrown on live heaps with stackers.

Blend The planned heap composition referring to the percentage of coal from each mine required on that heap (to ensure a good quality coal composition).

Blending The process of stacking and reclaiming coal in such a fashion that the different qualities of coal is mixed with each other to form a combined product with a homogenous composition.

Bypass Coal from strategic stockpiles fed directly to the factory (via the factory belts).

Cooling schedule The combination of the initial temperature, the number of iterations and the temperature reduction function that governs diversification and intensification.

CHF The Sasol *Coal Handling Facility* studied.

CVC The *Coal Value Chain* which includes coal handling, processing and gasification.

FEL *Front End Loaders* are used for manual coal handling processes.

GA The *Genetic Algorithm* is a metaheuristic based on evolutionary principles.

GP *Goal Programming* aims to achieve certain goals by minimizing the amounts by which these goals are not achieved.

Homogenization See Blending.

Incumbent solution The best solution found to date.

MES *Manufacturing Execution Systems* are information systems that control, plan track and/or assist with manufacturing activities.

Metaheuristic A high-level strategy to guide heuristic methods to find approximate optimal solutions.

Mixing See Blending.

Neighbourhood of solutions Solutions closely related to the current solution, which are generated by changing only one aspect of the current solution.

Prop-chute Coal from mine trajectories sent directly to the factory, bypassing the stacking and reclaiming processes.

Reclaiming Picking up coal stacked on heaps and sending it to the factory belts.

SA The *Simulated Annealing* metaheuristic solves optimization problems with principles analogous to the physical annealing process of materials.

Throw-out When a mine's bunker is full, any mine production tons are thrown out on a stockpile next to the mine's bunkers.

Stacking Throwing mine production tons onto heaps on one of six stockpiles.

TS The *Tabu Search* metaheuristic solves optimization problems by preventing cycling back to previously visited solutions.

Contents

List of Figures

List of Tables

Chapter 1

Research Problem

In industry, coal is mostly used to generate electricity or to produce liquid fuel products (Collings, 2002). In the liquid fuels industry, processed coal is gasified to produce gas that is then transformed into fuel. Coal quality varies at every source, and also between sources. Since a stable quality coal mix is required for the gasification process, the scheduling of coal arriving from the sources is an intricate challenge for management.

The processes that consume coal are sensitive to the quality of the supplied coal as well as the variance in the quality of the coal feed. As coal from different sources have varying qualities, the different sources are homogenized (blended) in order to provide coal with consistent qualities to the consuming processes (Hydro, 1983:4-2; Collings, 2002). Zhong et al. (2005) define a coal blend as the required coal *quality mix* for optimal gas and fuel production.

A complex scheduling problem results when one attempts to optimize gasification yields by blending and processing coal correctly. According to Longwell et al. (1995) and Van Dyk et al. (2006), the benefits associated with good coal blending and processing are:

1. Maximizes the probability of success in any proposed gasification venture.

2. Optimizes the operation (increases efficiency) and profitability (reduces gasification operating cost).

3. Minimizes the variance in ash content and the diameter of the coal provided as these variables affect gasification efficiency.

4. Prevents breaks in coal feed that amounts to significant losses.
5. Prevents stoppages in coal production that impacts the mine's profitability negatively.

The importance of properly scheduling coal handling, processing and gasification should not be underestimated as a minor problem in the input side of liquid fuel production will often cause major disruptions in the supply of petrol, diesel and jet-fuel. As demonstrated during the South African fuel crises at the end of 2005, fuel shortages do not only cause major inconveniences to the public, but also has a significant impact on business; and therefore the country's economy (Furlonger, 2005:24–25).

1.1 Coal to Liquid Fuel Scheduling

The combination of coal handling, processing and gasification is often grouped under the term Coal Value Chain (CVC).

Traditionally, the *coal handling* processes are used to transport coal over vast distances and to build inventory supply buffers. The coal handling processes extract coal from mine bunkers and move the coal to stacking locations. Thereafter, the stacked coal is reclaimed from the stockpiles and conveyed to the coal processing area. The coal handling facility may also be used to perform coal homogenization (mixing or blending).

The *coal processing* facility removes coal particles that are too fine to be used in gasification. Fines restrict the flow through the gasifier which lowers production rates. The diameter of the coal let through during the wet sieve processes, as used in coal processing, is one of the variables that critically affects the efficiency of the gasification process.

During *gasification*, the coal is gasified under pressure in the presence of oxygen and steam. The gasification process is sensitive to the fine coal fraction and to the quality parameters such as total carbon, ash, sulphur, nitrogen, hydrogen, oxygen and moisture content.

The challenge in the CVC environment is to provide the gasification process with a continuous good quality feed of coal, given the erratic supply of coal from mines. The gasification process runs 24 hours a day and 7 days a week, but the mines produce at different rates for 18 hours a day during the week and 9 hours on Saturdays. These production rates and their effect on the coal handling facility's inventory levels are shown in Figure 1.1.

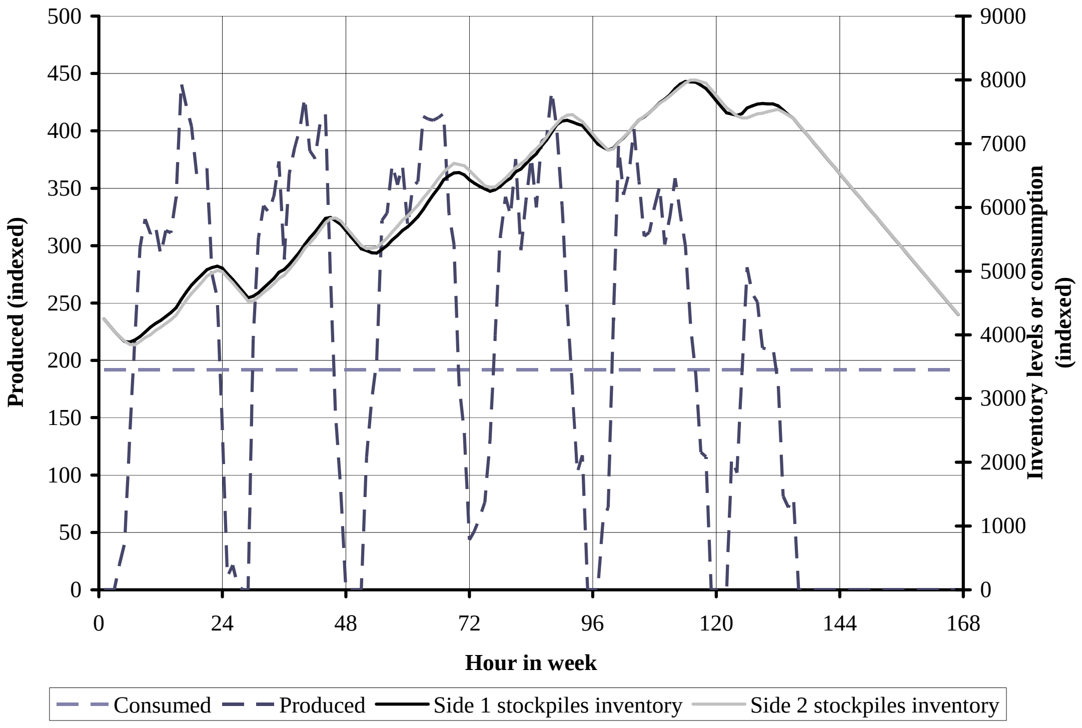


Figure 1.1: The effect of consumption and production rates on inventory levels

Coal has to be stockpiled during the periods that the mines produce so that the gasification process can be supplied with a 24 hour, 7 days a week feed. Any stoppage of feed to the gasification process shuts down the entire downstream process. To start up all the processes takes a few days, during which the company experiences losses in the order of millions of dollars (US) per day. These losses are due to loss of sales and depletion of inventory. The petrol, diesel and jet-fuel supply to the country will also be disrupted during this start-up period.

Mine production may also not be stopped as this has a negative impact on the mining company's profitability. As the mining company can only sell a certain amount of coal, costs have to be kept at a minimum and the maximum amount of time must be spent on production. A mine stoppage therefore represents a permanent loss in sales as it is not possible to catch up the lost production. Mine stoppages occur when the inventory buffers (mine surface bunkers and stockpile space allocated to the specific mine) are full.

As current approaches are not able to successfully address the coal scheduling need, this dissertation will develop and apply a model for scheduling the coal handling processes. This model will ensure minimum disruptions in the gasification process and mine production, whilst also minimizing the variance in the quality of the supplied coal. The coal processing and gasification processes are considered, but will not be scheduled as their operation depends on the actual real-time process values of the factory.

1.1.1 Coal Handling

Various collieries (coal mines) supply coal to the coal handling facility, which then combines and blends the various coal sources. The coal handling facility must act as a supply buffer between the collieries (which exhibit variable production rates) and the factory (which requires constant blends and exhibits uncertain demand). A simplified coal handling facility layout is shown in Figure 1.2.

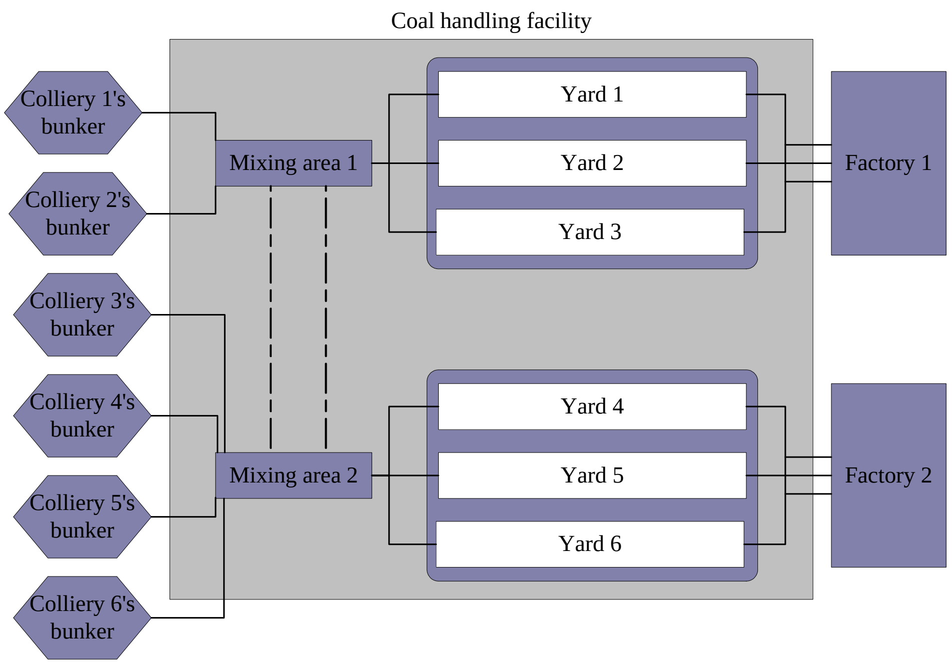


Figure 1.2: Simplified coal handling facility

Each colliery feeds its production into a surface bunker. Coal is then pulled from the bottom of the colliery's surface bunker to the coal handling facility, where the coal is conveyed to a coal storage area. This area is often referred to as yards, stockpiles or heaps (Swart, 2004:4).

A more detailed illustration of one of the yards in Figure 1.2 is shown in Figure 1.3. In this example there are two heaps on the yard. There can be between 0 and 5 heaps per yard. Each yard has both a stacker and a reclaimer. Stackers are used to stack (throw in layers) coal onto the heaps in the stockpile area. Reclaimers then pick up the stacked coal and send it to the coal processing facility (factory). The combination of the stacking and reclaiming processes are responsible for the blending of coal.

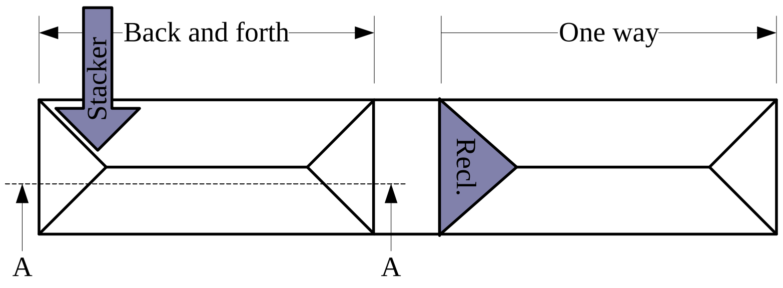


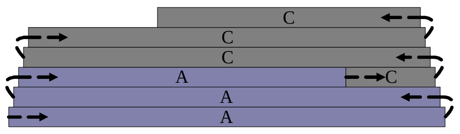
Figure 1.3: A stockpile yard shown with two heaps, a stacker and a reclaimer

Figure 1.4 illustrates the simplified blending process, by showing a heap being stacked and reclaimed, from cross-section view A–A (as indicated in Figure 1.3).

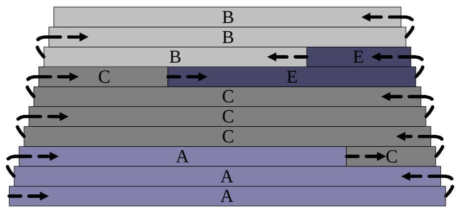
Figure 1.4(a) shows a stacker creating a heap by throwing various layers of coal onto each other in a specific area on the yard. These layers are thrown from side to side with different coal sources, as the stacker moves backwards and forwards along the length of the heap, until the heap is full. The heap's blend is enforced by throwing a certain number of layers, and fractions of layers, of each source.

When these layers have been stacked, the heap will exhibit the required blend when it is fully stacked (see Figure 1.4(b)). The contribution of each source to a fully stacked heap must be within ±3% of the heap's planned contribution for the respective colliery's coal. Once a stacker has filled up a heap, it will either start a new heap or fill up a heap that is not fully stacked.

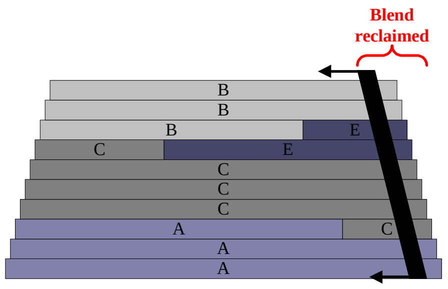
Only when the heap has been completed (fully stacked) it can be reclaimed.



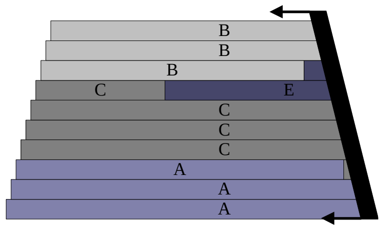
(a) ***Heap being stacked in a layered fashion with coal from mines A and C***



(b) ***Fully stacked heap with coal from mines A, B, C and E***



(c) ***Blending performed by reclaimer slicing heap***



(d) ***Heap being reclaimed further***

Figure 1.4: Blending process performed by stacking horizontal layers and reclaiming vertical slices as seen from cross-section A–A

Reclaiming is done using a large rake, which is angled at the same angle as the edge of the heap (as illustrated in Figures 1.3 and 1.4(c)). Because horizontal layers are stacked and vertical fractions are reclaimed, the reclaiming process actually performs the blending. Note that in practice a large number of thin layers are thrown on the heaps. Therefore, layers consisting of different sources will not have such a large impact on the reclaimed blend as it might seem in Figure 1.4(c) (as in this figure it might seem that source make-up of the first slice reclaimed differs drastically from the source make-up of the last slice reclaimed).

Figure 1.4(d) shows a heap of which 20% has already been reclaimed. The reclaimer will move through the heap until all the coal in that area of the yard has been reclaimed. The reclaimer can then either continue in the same direction or turn around to start reclaiming another fully stacked heap. If no fully stacked heap is available for reclaiming, the reclaimer must wait until the heap is full. Two reclaimers at a time must be reclaiming simultaneously to ensure that each of the factories is supplied with enough coal. Therefore, only one reclaimer may be waiting per factory at a time.

1.1.2 Coal Processing

The coal processing facility receives the reclaimed (blended) coal, washes it and splits this stream into three parts (Maree, 2006):

- Fine coal
- Normal coal
- Large coal

In the liquid fuel industry, the fine coal is discarded as it reduces the gas yield. However, some of the fine coal can be used for electricity generation.

The large and normal coal is fed to gasification. The way in which these two streams are placed on the conveyors has been found to impact gas yields, as it influences the gasifiers' efficiencies.

1.1.3 Gasification

The processed coal is dumped into gasifiers. The gasifiers react coal with steam and oxygen in a pressure vessel at a high temperature. This approach, called Sasol-Lurgi fixed bed dry bottom gasification, produces crude gasses CO and H_2 (Van Dyk et al., 2006) as well as ash sludge (Collings, 2002:24). The crude gas is then turned into liquid fuels and chemicals via downstream Fischer-Tropsch processes.

1.2 Research Design

This project will investigate the conjecture that complex, highly constrained scheduling problems can be successfully solved using metaheuristics. Based on Page and Meyer (2000:5), Mouton (2001:176) and Manson (2006:159), this then suggests that this project deals mainly with theory-building research, although some theory-testing research will also be required to prove the validity of the theory and methods developed.

The problem statement of this dissertation is formulated as a research question to help focus the problem (Mouton, 2001:53).

> Can a metaheuristic be developed to efficiently (i.e. solution time of less than two minutes) provide a schedule to minimize blend deviations without disrupting either demand or supply, whilst considering multiple-objectives, stochastic variables and the highly constrained environment?

Mouton (2001:56) states that the research design focuses on the deliverables of the research by summarizing what kind of study is planned to address the research questions adequately. To answer this dissertation's research question, the following research design is formulated:

- Based on the existing metaheuristic theories and literature, metaheuristic theory needs to be adapted to address complex, highly constrained, continuous product scheduling problems.
- To determine whether the developed theory is efficient, it needs to be applied in industry.

Based on Manson (2006:158)'s approach, the research design indicates that research into Operations Research (OR) is executed to advance the practice of OR as this dissertation will:

- Develop an improved *model* for the coal scheduling problem.
- Present a new methodology for *choosing* solution approaches for complex problems.
- Apply an *improved solution approach.*

1.3 Research Methodology

The research methodology communicates the steps of how the research process was undertaken (Mouton, 2001:56; Manson, 2006). As it focuses on how the research is executed, the document structure represents the research methodology of this dissertation.

To start addressing the research goal, the next chapter provides a literature review. It firstly attempts to classify the scheduling problem faced. As this is a practical problem, some guidelines on how to model applicable practical challenges are included. To provide a basis for determining which solution method should be used to solve the scheduling model, including the practical considerations, an overview of various solution methods is also given.

Chapter 3 provides an overview of all the solution approaches that have been used to solve the applicable scheduling problem. As all these methods have failed, an approach for choosing solution methods is suggested. This approach is then applied by setting up a simplified, yet representative, problem of the full-size scheduling problem. The simplified model is solved exactly to provide a benchmark for other solution approaches. Various approximate solution methods are applied to the same simplified model. The most appropriate solution method is identified by comparing both solution approach effectiveness (quality of solution) and efficiency (speed of execution).

A more detailed discussion of the chosen technique is given in Chapter 4. The basic version of the selected algorithm is then adapted into a solution approach to solve the studied scheduling problem.

In Chapter 5, the conceptual algorithm developed is applied to the practical scheduling problem. The algorithm's behavioural variables are adjusted to ensure the best possible performance of the developed algorithm in this application. A brief discussion of how the algorithm was practically implemented and integrated with the company's business processes is then given. The quality of the developed approach is evaluated by comparing the algorithm's output with the outputs generated by the previous manual approach applied in industry.

Chapter 6 concludes the document by summarizing the research done and its successful practical application. Finally, various suggestions for further research are discussed.

Chapter 2

Literature Review

To further investigate the scheduling problem faced, it is necessary to determine where this specific problem instance fits into the current scheduling body of knowledge. As a practical instance of this problem need to be solved, some practical modelling challenges must also be investigated. Once the problem has been classified and enhanced to consider practical challenges, one must determine how this model can be solved efficiently.

2.1 Problem Classification

2.1.1 Scheduling Problem

Scheduling, in its widest sense, is concerned with the allocation of scarce resources to tasks over time (Rardin, 1998:602; Dorigo and Stützle, 2004). The large scheduling body of knowledge entails various different applications; including workforce, timetabling, project and production (machine and shop) scheduling (Winston, 1994; Rardin, 1998; Taha, 2003).

Production Scheduling

Various different types of production scheduling models exist. The simplest scheduling problems in this category are *single machine* scheduling problems (Potts and Kovalyov, 2000; Allahverdi et al., 2006). Additional series machines can then be added to the single machine to form a multi-stage scheduling problem (Little and

Hemmings, 1994). Model complexity further increases as identical *parallel machines* are added to a single machine (Potts and Kovalyov, 2000).

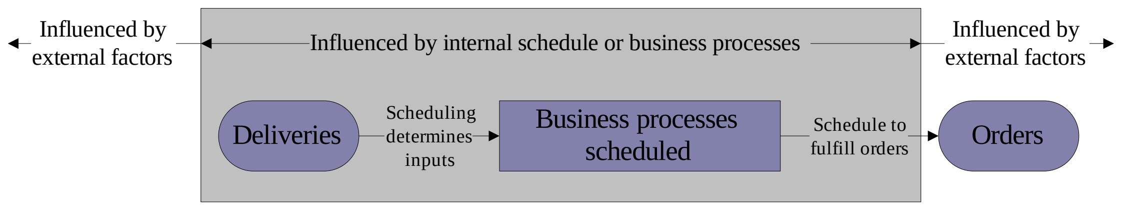
Flow shop problems further enhance parallel machine problems by allowing one or more machines at each stage in production (Little and Hemmings, 1994; Linn and Zhang, 1999; Allahverdi et al., 2006). *Open shop* problems consider environments where each job must be processed by every stage of production, in any order.

When each job can has a different route through the production facility, the problem is known as a *job shop* problem (Little and Hemmings, 1994; Allahverdi et al., 2006). In a job shop environment, a specific job is not necessarily processed by all machines and may be processed more than once by the same machine.

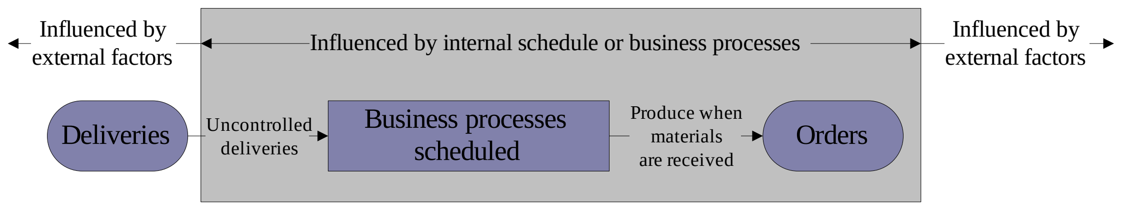
The coal handling scheduling problem has multiple stages and includes non-identical series and parallel “machines”.

The majority of research in the scheduling environment assumes that setup time is either negligible or included in job processing times (Potts and Kovalyov, 2000; Allahverdi et al., 2006). Although this assumption simplifies the scheduling model, it adversely affects the solution quality in environments where setups play a more important role. Research considering setup times started emerging in the mid-1960s, with a larger interest in this area in the past 10 years (refer to the 10 recent applications discussed in Allahverdi et al. (2006)). The coal handling scheduling problem requires explicit attention to setup times (in the form of mine trajectory, stacker and reclaimer change-over times) as these downtimes reduce the time available for pulling mine coal, stacking coal or sending coal to the factory.

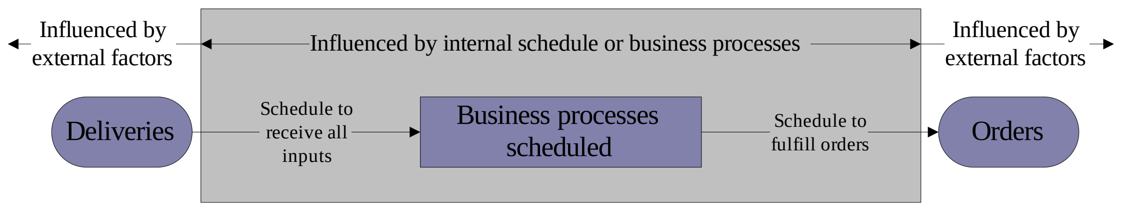
Where most scheduling problems are aimed at satisfying the outbound (or inbound) needs by determining when which inbound (or outbound) products should be procured and processed (as applied by Rardin (1998), Chunfeng et al. (1999), Potts and Kovalyov (2000), Sabuncuoglu and Bayõz (2000), Winston and Venkataramanan (2003), Chen and Chao (2004), Joubert and Conradie (2005), Allahverdi et al. (2006) and Bellabdaoui and Teghem (2006); amongst others), in the coal scheduling environment one cannot influence either inbound or outbound product delivery (Swart, 2004; Coetzer and Harmse, 2007). The coal handling processes become complex to schedule as one has to satisfy both the outbound needs and the inbound deliveries (production). The difference between typical scheduling prob-


Influenced by external factors
Influenced by internal schedule or business processes
Influenced by external factors
Deliveries
Scheduling determines inputs
Business processes scheduled
Schedule to fulfill orders
Orders

(a) ***Typical pull system (backward) scheduling problems***


Influenced by external factors
Influenced by internal schedule or business processes
Influenced by external factors
Deliveries
Uncontrolled deliveries
Business processes scheduled
Produce when materials are received
Orders

(b) ***Push system (forward) scheduling problems***


Influenced by external factors
Influenced by internal schedule or business processes
Influenced by external factors
Deliveries
Schedule to receive all inputs
Business processes scheduled
Schedule to fulfill orders
Orders

(c) ***Studied push and pull scheduling problem***

Figure 2.1: Push-pull system classification of scheduling problems

lems and the problem concentrated on in this dissertation is illustrated in Figure 2.1.

Given all the problem variations discussed, the coal handling scheduling problem can be more or less classified as an enhanced job shop-type problem (although it is not a pure job shop problem), including setup times in a push-pull environment. The candidate therefore agrees with Linn and Zhang (1999), who state that "it is apparent from the literature that there is a gap between theory developed and practice applications".

Static and Dynamic Scheduling

Scheduling problems can be split into two categories, namely static and dynamic scheduling models (Suresh and Chaudhuri, 1993; Goddard, 2006). Static scheduling models are models that provide a strict schedule for a certain period of time.

When in practice one deviates from this schedule, the entire scheduling model is run again with new input values. A disadvantage of this approach is production instability as the schedule provided by the new scheduling run may differ vastly from the previous run's schedule (Aytug et al., 2005).

To address this potential problem and to prevent having to wait for the entire model to be run again, dynamic scheduling systems are used. Dynamic scheduling models adjust the current schedule as soon as deviations from the planned schedule occur (Suresh and Chaudhuri, 1993; Sabuncuoglu and Bayõz, 2000). The candidate suggests that static scheduling models can also provide stable solutions by adding constraints to the static model that will ensure that the current schedule being executed is changed as little as possible in the new run.

Having reviewed scheduling problems, some practical considerations with regard to scheduling need to be considered. Allahverdi et al. (2006) state that both multiple objective and stochastic problems have received less attention in studies since they are more difficult than single objective and deterministic problems respectively. As multiple objectives and stochasticity are often found in practical problems, these practical considerations are further investigated.

2.1.2 Conflicting Objectives

When modelling a problem, it is not always possible to define its goal as a single objective function. For example, some problems are aimed at finding a solution that provides a good balance between multiple conflicting objectives (e.g. maximizing return, whilst minimizing risk), instead of only maximizing or minimizing a single objective (Ramesh and Cary, 1989; Rifai, 1996). Allahverdi et al. (2006) state that this is especially the case when dealing with practical problems.

In multiple objective optimization models it becomes difficult to identify the feasible solution that is best at addressing all of the objectives (Ehrgott, 2005). Especially since a feasible solution may be very good at addressing one of the objectives, but may be extremely poor at addressing the other objectives.

To be an "optimal" solution in a multiple objective optimization model, the feasible solution should not be able to be improved in any of its objective function values without decreasing one ore more of the other objective function values. This kind of feasible solution is known as an *efficient point* (Rardin, 1998:379; Tamiz et al., 1998). As multiple objective problems can have numerous efficient points, the collection of efficient points is known as the *efficient frontier*.

When solving multiple objective optimization models, the optimal solution should be one of the efficient points on the efficient frontier. To determine which efficient point on the efficient frontier is the optimal solution to a multiple objective problem, various ways of solving multiple objective problems are possible. Each method has different advantages and disadvantages. Table 2.1 compares the different multiple objective solution methods (based on Tamiz et al. (1998) and Rardin (1998:384–400)).

In Table 2.1, note that GP is the acronym for Goal Programming. When using GP to address multiple objectives in a problem, the model does not optimize (maximize/minimize) the objective directly, it attempts to minimize the deviations between desired goals and the realized results (Rifai, 1996). Goals are expressed as equations (i.e. soft constraints) and a deficiency variable is assigned to each of these; the aim is now to minimize the deficiency for each goal. This leads to the following provisional objective function and soft constraints.

Table 2.1: Multiple objective optimization solution approach comparison

Approach	**Optimal solution efficient?**
Pre-emptive	Yes
Weighted sums of objectives	Yes
Weighted deficiencies with GP	No
Pre-emptive GP	No
Pre-emptive GP with weighted deficiencies	No
Modified GP	Yes
Report all efficient solutions	Yes

The pre-emptive approach solves the problem with the most important objective. Then it puts the first objective's value into the model as a constraint and solves for the second objective and then repeats this process. Consequently, the disadvantages of this approach are:

- It places too much emphasis on the first objective
- All the solutions obtained are alternative optima of the first stage solution

Reporting all feasible solutions will provide feasible solutions, but proves tedious in practice, as too much information needs to be reported and more involved user intervention is required.

Given these disadvantages of pre-emptive and reporting all efficient point approaches, either weighted sums of objectives or modified GP should be used to solve multiple objective problems. Although the weighted sums of objectives approach is mostly used (Choobineh et al., 2005), the final decision between these two approaches depends on the specific application as some sources state that weighted sums of objectives might work poorly for certain problems (Ehrgott, 2005:98).

2.1.3 Stochastic Elements

Most OR practitioners use average values of the processes they are modelling as input values to their optimization models (Little and Hemmings, 1994; Chunfeng et al., 1999; Linn and Zhang, 1999; Potts and Kovalyov, 2000; Chen and Chao, 2004; Conradie and Joubert, 2004; Jansen and Mastrolilli, 2004; Swart, 2004; Choobineh et al., 2005; Bellabdaoui and Teghem, 2006). This approach proves popular for two reasons:

1. In most OR courses, single deterministic input values are assumed to be sufficient.
2. It is, in most cases, difficult to get hold of information (and even more difficult to get reliable information) in practise. With this difficulty, the "average value approach"[1] proves valuable as it is the best information one has about the system.

Although nearly all processes will exhibit some variability, the "average value approach" often does prove to be sufficient if the variability of the process is of

[1]The mode of the distribution is also sometimes used as people familiar with the process are often asked what the *typical* value would be.

such a nature that the solutions suggested by the optimization model are still practically executable or better than previously available solutions.

When one has knowledge about the distribution of the variable parameter, the reliability of the "average value approach" solution can be calculated. This is done using the suggested solution in conjunction with various realizations of the parameter distribution. The percentage of instances where the suggested solution is still feasible without requiring unplanned corrective actions (recourse), given the different distribution realizations, is known as the *reliability* of the solution. The reliability of solutions can be seen as a measure of feasibility, or the probability of being feasible (Kall and Wallace, 1994:12,13).

The theoretical reliability of deterministic average value optimization models is in the order of 25% and varies from application to application (Kall and Wallace, 1994:12). This indicates that working with deterministic estimates of the input parameters could produce solutions that would not be fully executable in 75% of cases (as recourse would be required).

One could use the worst case values of input parameters in optimization models. This approach provides a solution that is rather safe, and is often referred to as a "fat solution", but that will be extremely costly in practical applications. This approach also does not apply optimization over the variability of input parameters, but sees them as hard constraints.

In some instances the inherent system variability has a major impact on the type of solutions that would be feasible in practice and/or executable without recourse (i.e. the reliability of deterministically determined solutions would be low) (Ramesh and Cary, 1989; Suresh and Chaudhuri, 1993; Kall and Wallace, 1994; Birge and Louveaux, 1997; Sabuncuoglu and Bayõz, 2000; Aytug et al., 2005; Joubert and Conradie, 2005; Morison, 2005; Philpott, 2006; Henrion, 2006; Coetzer and Harmse, 2007). In these cases, extra time and effort should be spent to gather enough information about the critical process so that a probability density function can be used to consider variance in the model.

Simulation modelling is often used to determine what the effect of variability will be on the current solutions or actions (Suresh and Chaudhuri, 1993; Little and Hemmings, 1994; Sabuncuoglu and Bayõz, 2000). This impact of variability is compared using alternatives supplied by the user. Although this process may prove valuable, the two-phase approach only considers variability; it does not specifically include it in the alternative generation or optimization stage.

There are two approaches that can be used for optimization that considers the variability of the input parameters in a single model.

Robust Optimization This approach is used when the varying parameter values are only known within certain bounds (Philpott, 2006). The goal of robust optimization is to find a solution that will be feasible for any realization of the parameters in the given bounds, whilst trying to ensure that the solution is close to optimal for all the different realizations. In a scheduling environment this means that Robust Optimization focuses on "creating a schedule which, when implemented, minimizes the effect of disruptions on the primary performance measure of the schedule" (Aytug et al., 2005).

Stochastic Programming This is a modelling approach used to model optimization problems that need to consider the effect of variability (Philpott, 2006). Practical optimization problems are most likely influenced by some kind of uncertainty and in Stochastic Programming the known (or estimated) probability density function is included in the optimization process; rather than seeing it as a constraint, which would provide a "fat solution".

To understand how Stochastic Programming maximizes the expectation of a function of both the suggested decisions and the applicable probability distributions, the two different ways of implementing Stochastic Programming should be considered.

Two-Stage Recourse Programming

Two-Stage Recourse Programming is the most popular Stochastic Programming technique and is often referred to as the "classical Stochastic Programming modelling paradigm" (Birge and Louveaux, 1997:155; Philpott, 2006).

In practice there is often the opportunity to take some corrective actions, or *recourse*, when the effect of variability has become known (Sen and Higle, 1999:38). For example, in a situation with variable productivity it might happen that not enough units for an order have been produced in the planned production time, forcing the production facility to reactively start working overtime to finish order production in time (i.e. they take recourse).

For different realizations of a distribution, different amounts of recourse may be required (Birge and Louveaux, 1997:54). In the above example, if productivity is high, no recourse would be required as production will be completed before the order is due. Only a little recourse (i.e. overtime) will be required if the productivity was slightly lower than planned. However, large amounts of overtime will be required if productivity happens to be much lower than expected. As each realization of a distribution has a different probability of occurring, the probability of the amount of recourse required for different realizations is known.

When using recourse one ensures that, given any combination of the applicable distributions' realizations, the provided solution is feasible. However, recourse programming requires the costs of recourse (compensating decisions) to be known (Henrion, 2006), as recourse programming typically minimizes the sum of total first-stage costs and the cost of the expected recourse (Kall and Wallace, 1994:10; Birge and Louveaux, 1997:155).

When continuous probability functions are used, the problem becomes large and non-linear (i.e. difficult to solve). Therefore, the distributions are approximated by their discrete counterparts. Given these discrete approximations, Kall and Wallace (1994:31) show that recourse programs are convex under mild assumptions (i.e. easier to solve than the continuous non-linear models).

Chance Constrained Programming

In Chance Constrained Programming (CCP), also known as Probabilistic Constrained Programming, a certain probability of the optimal solution being feasible (reliability level) for all distribution realizations is achieved, rather than achieving feasibility for all realizations as in stochastic recourse programming (Birge and Louveaux, 1997:103; Sen and Higle, 1999:46; Philpott, 2006).

Chance Constrained Programming uses an approach where a solution is seen as feasible (in a CCP stochastic sense) when the solution is feasible with a high probability and only a low percentage of distribution realizations cause the solution to be infeasible (lead to constraint violations) (Henrion, 2006). This approach is based on the fact that it is almost impossible to plan for unexpected extreme situations; especially since the solution would be extremely expensive (have a poor objective function value).

CCP is often useful when only a certain level of adherence to a certain constraint is required (Sen and Higle, 1999) or when one needs to ensure that constraint violations do not occur too frequently (Birge and Louveaux, 1997:35). For instance, when a target of satisfying 95% of customer demand is enough to maintain the customer base (Kall and Wallace, 1994:14).

Method Comparison

Recourse models transform the variability into an expected parameter, but CCP "deal[s] more explicitly with the distribution itself" (Kall and Wallace, 1994:xi). Consequently, CCP models might be more difficult to solve, but they deal with more of the information contained in the probability distribution.

In practice, neither method is "better" or more valid than the other (Kall and Wallace, 1994:xi). The method that is most appropriate is therefore solely dependent on the application itself.

Now that the complexities of modelling scheduling practical problems have been reviewed, the next section investigates how one can go about solving these practical problems.

2.2 Solution Approaches

Due to the nature of scheduling problems, many scheduling problems require the use of a large number of integer and/or binary variables, making these problems more difficult to solve. It is consequently important to also consider the solution approach to be used for scheduling models. The different schools of thought with regard to the solution of scheduling problems can be categorized based on the level of optimality that the solution method attempts to achieve. These categories are as follows:

Focus on optimality The typical Operations Research approach used in scheduling is to apply exact optimization techniques to optimally solve scheduling problems. This approach should be used when the size of the scheduling model and the number of integer variables enable one to solve the model optimally within the time limit provided by the specific practical application. Conradie and Joubert (2004) and Joubert and Conradie (2005) successfully applied this approach to practical problems, whilst Bellabdaoui and Teghem (2006) successfully applied this approach to a simplified practical problem. Swart (2004) and Coetzer and Harmse (2007) could not successfully apply this approach to the practical problems they attempted to solve.

Focus on feasibility Even simple scheduling problems, such as problems that assign various tasks on identical parallel machines, are known to be *NP-hard* problems (Jansen and Mastrolilli, 2004). Given this complexity of scheduling models, the notion of focussing on feasibility rather than optimality when solving these problems has proven to be "very fruitful". As approximation algorithms ensure feasibility, but not necessarily optimality, they are often used to perform optimization in scheduling environments in an acceptably short time. As most scheduling problems need to be solved in an operational environment, the short time allowed for problem solution also supports the use of approximation techniques.

The following two sections deal with exact solution methods (which focuses on optimality) and approximation techniques (which focuses on feasibility), respectively.

2.2.1 Exact Solution Approaches

Allahverdi et al. (2006) reviewed 227 scheduling articles relating to different types of machine-type scheduling problems. The most popular exact solution method applied is the branch-and-bound algorithm (42% of the articles). However, Winston (1994:526) states that this technique requires large amounts of computer time. In operational scheduling problems one does not have large amounts of time available for problem solution and consequently practical scheduling models are often not solved using branch-and-bound. For a detailed discussion on branch-and-bound as well as branch-and-cut as applied in mixed integer scheduling or planning models, the reader is referred to Pochet and Wolsey (2006).

Dynamic Programming was applied in 20% of cases. The problems where Dynamic Programming were applied are the simpler scheduling problems. One should also be aware that the size of the Dynamic Programme drastically increases as model size increases. This attribute causes Dynamic Programming to be too computationally expensive for problems of moderate size and complexity (Winston, 1994:1041; Winston and Venkataramanan, 2003:785,797); and is known as the "curse of dimensionality" (Goldberg, 1989:5; Taha, 2003:425).

A total of 33% of the problems used unspecified exact mathematical programming solution methods, whilst 5% of cases were solved using other miscellaneous solution techniques.

The various articles reviewed by Allahverdi et al. (2006) also include approximation approaches. Their findings, with regard to solution methods, can be summarized as follows in Table 2.2.

Table 2.2: Solution approaches of scheduling problems

Scheduling problem	**Exact**	**Approximation**	**% exact**
Single machine non-batch setup time	7	12	36.8%
Single machine batch setup time	29	14	67.4%
Parallel machines non-batch setup time	7	25	21.9%
Parallel machines batch setup time	6	19	24.0%
Flow shop non-batch setup time	15	43	25.9%
Flow shop batch setup time	5	16	23.8%
Open shop	0	3	0.0%
Job shop	5	12	29.4%

Table 2.2 lists the studied scheduling problems in increasing order of complexity. As complexity increases, exact solution approaches are used in fewer applications. One may therefore conclude that, in the scheduling environment, approximation approaches are mostly used when model complexity increases beyond that of simple theoretical problem instances.

Scheduling problems often exhibit multiple-objectives, which makes the problem more difficult to solve. This also supports the reason why so many applications use approximation solution techniques. Solving stochastic programming models also requires more effort than solving deterministic models. Stochastic models are consequently often solved using approximation approaches.

2.2.2 Approximation Techniques

When using approximation techniques to solve a problem, one estimates the optimal solution by using efficient heuristic rules to attempt to produce a solution that is close to the global optimal solution.

The various different types of approximation techniques are now reviewed.

Heuristics

Various complicated models are solved with heuristics. Winston (1994:526) defines a heuristic as a method to solve a problem by trail and error when an algorithmic approach is impractical. Goldberg (1989:2) claims that heuristics can be divided into three categories. The first two categories are classified as follows (Conradie, 2004:24-25):

Calculus-based These methods have been studied in depth. This class of heuristics finds local optima by moving in the direction which has the steepest increase in objective function values. Problems with a single peak are easily solved by the "hill climbing" approach of calculus-based methods. A major problem with calculus-based methods is that they tend to "get stuck" at local optima instead of finding the global optimal solution. These methods would find any one of the peaks as its solution, not necessarily the highest peak. The peak that will be produced as the heuristic's answer depends on where the search started, i.e. the initial solution provided to the heuristic.

Enumerative Here the objective function value of each and every point in the solution space is analyzed. The highest value is then produced as the global optimal answer, as this method considers all solution points and consequently cannot "get stuck" at local optima. The problem with this method is its lack of efficiency due to the enormous amount of computations that need to be done. Even for moderate size problems, this approach loses its possibility for practical application.

An additional category employs intelligent random-based techniques to search the solution space to avoid getting stuck at local optimal solutions. These methods are referred to as *metaheuristics*, which can be defined as "master strategies which uses intelligent decision making techniques to guide algorithms to find global optimal solutions" (Conradie, 2004:26); by allowing the algorithm to move to solutions that will temporarily decrease the objective function to ensure that the algorithm does not converge to local optimal solutions.

A short review of each of the most popular metaheuristics is given below.

Tabu Search

Winston and Venkataramanan (2003:815) state that Tabu Search (TS) is a metaheuristic procedure based on intelligent decision making strategies. Intelligent decision making emulates the rules that humans use in daily decision making.

The Tabu Search algorithm temporarily forbids moves that would generate recently visited solutions. It is therefore a memory-based search algorithm as it keeps a Tabu list of certain forbidden moves in the solution space. This Tabu list prevents cycling between possible solutions and helps to find global optima by moving away from local optimal solutions (Rardin, 1998:687–688). To ensure that TS finds high-quality solutions, aspiration criteria are defined to override the Tabu list as required. An aspiration criterion evaluates the quality of possible moves in the solution space and is consequently used to guide moves made by the TS metaheuristic.

To find good solutions, TS is highly dependent on the quality of initial solutions (Chunfeng et al., 1999; Van Breedman, 2001; Chen and Chao, 2004). TS has previously been successfully applied in various situations to arrive at good approximate solutions.

Simulated Annealing

Simulated Annealing (SA) is rooted in the resemblance between the physical annealing of solids (formation of crystals) and combinatorial optimization problems

(Morison, 2005:41). In the physical annealing process a solid is first melted and then cooled very slowly going through an array of temperatures, spending a large amount of time at low temperatures, to finally obtain a lattice structure that corresponds to the lowest energy state (Busetti, 2005).

In the annealing process, the temperature of the metal is firstly increased and then gradually reduced. Increasing the temperature randomly scatters atoms by breaking down the atomic matrix structure of the atoms. With randomly positioned atoms, the metal is at its highest energy state. By lowering the temperature of the metal, the atoms gradually settle down into a certain stronger matrix-like order (Rardin, 1998:690). The "better" the order in which the atoms settle, the lower the energy state of the metal will be. By reducing a metal's energy state, its yield strength is increased.

The optimality of a solution is analogous to the energy state of the metal. With the temperature of the metal used as the objective function value, the temperature of the metal is gradually reduced at each stage of the heuristic. The temperature reductions cause SA to randomly search for better solutions in the current solution's surroundings (Winston and Venkataramanan, 2003:805). Possible solutions are represented by the order in which atoms settle. This order is tested to determine whether its associated energy state (i.e. level of optimality) is lower than others. The energy levels of solutions are determined by a formula referred to as Boltzmann's function.

According to Michalewicz (1992:16), SA eliminates most of the disadvantages of hill-climbing solution methods in that:

- SA is not highly dependent on the quality of initial solutions (unlike TS).
- Solutions found by SA are usually closer to the global optimal solution.

Genetic Algorithms

Living organisms adapt to their changing environments through biological evolution. These organisms have chromosomes that contain their genetic information (with each chromosome consisting of various genes). The way in which the genes are ordered and re-ordered within chromosomes is seen as a key factor in the survival of the species of organisms (Winston and Venkataramanan, 2003:808). The natural phenomenon of survival of the fittest and natural selection also play an important part in the successful evolution of organisms.

These concepts are used by Genetic Algorithms (GA) to solve optimization problems. Holland (1975) initiated the ground breaking work in the field of GAs by applying the principles of a population of chromosomes, mating based on natural selection, production of offspring by applying crossover to the parent chromosomes, mutation (genetic errors to ensure diversity) and survival of the fittest Goldberg (1989:1–2).

The first step in solving optimization problems using GAs is to find a way to represent a possible solution in a constant length chromosome string. When this representation has been determined, the following steps are followed to search for a good approximate global optimal solution (Winston and Venkataramanan, 2003:809):

Generation Randomly generate various chromosomes to form the initial population (generation 1).

Evaluation Determine the fitness (objective function) of each chromosome.

Selection Choose parents from the population for mating. Ensure that the fittest chromosomes have the highest probability of being selected.

Reproduction Combine the chromosomes of the parents, using crossover, so that offspring chromosomes are created (each offspring consisting of some parts from both parents).

Mutation After crossover has been applied, some chromosome genes are randomly altered.

Repeat The *Evaluation* to *Mutation* steps are then repeated for a predetermined number of generations.

GAs, based on the above mentioned principles, have been successfully applied to scheduling, routing and various other combinatorial optimization problems (Michalewicz, 1992:15). Additional theoretical proof of GA's good performance is provided by the Schemata Theorem and the Minimum Deceptive Problem (MDP), as set out by Goldberg (1989:27-57).

The Schemata Theorem provides mathematical proof that GAs are powerful tools when used to solve optimisation problems by tracking how key parts in the chromosome will be affected by the GA operators. It proves that short parts of the chromosome that represent good objective function values will be used in exponentially increasing fashion in the following generations of a Genetic Algorithm.

To support this mathematical proof, Goldberg (1989:46) formulated problems which are designed to deceive GAs so that they may not converge to the global optimal solution. The smallest problems than can deceive GAs are referred to as minimal deceptive problems (MDP). Goldberg (1989:54) found that GAs are often not deceived by the MDP.

Having reviewed possible solution approaches for practical problems, the following section shows how the most appropriate solution approach for the coal handling scheduling model is chosen.

Chapter 3

Choosing a Solution Approach

This section discusses the methodology applied to determine which solution method should be used to solve the Sasol Coal Handling Facility (CHF) scheduling problem. It starts with an overview of a method that failed to solve the complex coal handling scheduling model and then moves on to find an approach that will be able to solve the problem. The chosen method must be able to solve the problem within two minutes, as this is the longest time that the control operator can afford to wait for a new schedule whilst performing his/her operational duties.

3.1 Exact Solution Attempts of the Problem

The original objective of the first CHF scheduling model was to optimize daily operations by solving a Mixed Integer Non-Linear Programme (MINLP) scheduling model using continuous time points.

Continuous time points were chosen after a detailed study (Swart, 2004) indicated that, in this environment, deterministic time point models provide much worse objective function values than continuous time point models.

Swart (2004) developed an MINLP model for the coal scheduling problem with unit specific continuous time points. However, the model did not solve within 72 hours in an operational scheduling environment where solution times should be less than two minutes. The model was solved using GAMS 21.1 with the Dicopt solver (which uses Conopt as a Non-Linear Programme (NLP) solver and Cplex 9.0 as the Mixed Integer Programme (MIP) solver).

Thereafter the model was improved by using Special Ordered Sets of Type 1 (SOS1) variables. The SOS1 variables reduce the number of binary variables in the model. Restructuring of the continuous time and duration constraints was also done to further simplify the model.

The non-linear equations were linearized using approximate linearization techniques. The approximate linear model is solved first. The solution to this model then acts as the starting point for the proper non-linear model.

After reducing the scheduling horizon to 4 hours and not considering all the CHF infrastructure complexities, the model solved within two minutes. Although this solution time might be acceptable in practice, the too short scheduling horizon and simplifying assumptions make the model impractical.

Swart (2004) concludes that the "unit specific event based MINLP continuous time formulation method . . . is not robust enough to be applied to an operational industrial sized scheduling model such as the CHF problem".

The candidate and Marinda Swart (see Swart (2004)) attempted to make the MINLP model both practical and solvable by restructuring various constraints. This process did not achieve significantly better results.

Further attempts to develop a practical model were made by splitting the scheduling model into daily and shiftly scheduling models. The overall daily schedule is fed to the shiftly scheduler which then performs the actual detailed scheduling. However, neither of these two smaller models solved. The candidate developed the detail shiftly scheduling model and improved it to use a 24 hour scheduling horizon. It is found that the highly binary constraints in the model cause the solution time escalation, even before the non-linear constraints are added to the model.

Given these unsuccessful exact solution attempts, the next section deals with the selection of an alterative solution approach.

3.2 Simplified Model

To be able to compare the various solution methods with each other, a simplified version of the full-scale problem is formulated[1]. The characteristics of the coal

[1]Sections 3.2 through 3.5 represent joint work with Leilani E. Morison.

handling scheduling environment is included in the simplified model to ensure that the simplified model is representative of the full-scale problem. The candidate also suggests that the major shortcomings of the previous exact solution models should be included in the simplified model to ensure that the methods used to address these shortcomings will not cause erratic behaviour and adversely influence algorithm performance (after a solution method has been chosen based on an unrepresentative simplified model).

Consequently, the major shortcomings of the previous model are identified.

Operational rules Including all the operational philosophies and infrastructure constraints into the model makes the model unsolvable using exact solution methods.

Multiple objectives The previous exact model simply added the different conflicting objectives in the objective function. As this does not ensure that the solution is an efficient point, a different method of addressing the multiple objectives should be considered.

Stochasticity The previous formulation of the model does not enable one to consider the entire distribution of input variables when solving the model, but only an average value or point estimate.

To address the first issue in the simplified model (not in the full-scale model), the following simplifying assumption is made to ensure that the model will be optimally solvable and still be fairly representative of the actual case:

> Each of the yards is seen as a tank and consequently the model does not include stockpiles on yards or stacker-reclaimer heap logic[2] (this assumption is also made in the operational weekly blend planning model).

Given this assumption, the Mixed Integer Programme (MIP) for the simplified version of the coal scheduling problem without stochastic demand is formulated as follows, with:

[2]Discussed in Section 4.5.2

Input parameters:

cap_m = Capacity of bunker m, where $m \in \{1, \ldots, 6\}$

$fines_m$ = Fraction of fines for mine m, where $m \in \{1, \ldots, 6\}$

ash_m = Fraction of ash for mine m, where $m \in \{1, \ldots, 6\}$

$trcap_m$ = Trajectory capacity of mine m, where $m \in \{1, \ldots, 6\}$

$prod_{mt}$ = The production of mine m in time t, where $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

ls_m = Starting level for bunker m, where $m \in \{1, \ldots, 6\}$

ys_y = The starting stock level of yard y, where $y \in \{1, \ldots, 6\}$

Decision variables:

x_{ymt} $\triangleq$ Tons from mine m to yard y in time period t, where $m \in \{1, \ldots, 6\}$, $y \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

l_{mt} $\triangleq$ Bunker level of bunker m at time t, where $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

to_{mt} $\triangleq$ Throw-outs for mine m, in time t, where $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

lb_{mt} $\triangleq$ Load-backs for mine m in period t, where $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

$otons_{mt}$ $\triangleq$ Tons outside bunker m at the end of period t, where $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

$recl_{yt}$ $\triangleq$ Tons reclaimed from yard y in time t, where $y \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 24\}$

$$active_{ymt} \triangleq \begin{cases} 1 & \text{if coal from mine } m \text{ is moved to yard } y \text{ in period } t, \\ & \text{where } m \in \{1, \ldots, 6\},\ y \in \{1, \ldots, 6\} \text{ and } t \in \{1, \ldots, 24\} \\ 0 & \text{otherwise} \end{cases}$$

$$isrecl_{yt} \triangleq \begin{cases} 1 & \text{if coal is being reclaimed from yard } y \text{ in period } t, \\ & \text{where } y \in \{1, \ldots, 6\} \text{ and } t \in \{1, \ldots, 24\} \\ 0 & \text{otherwise} \end{cases}$$

Model:

$$\text{Maximize} \sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} \tag{3.1}$$

$$\text{Minimize} \sum_{m=1}^{6}\sum_{t=1}^{24} otons_{mt} \tag{3.2}$$

$$\text{Maximize} \sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} \qquad \forall y \in \{1,\dots,6\} \tag{3.3}$$

$$\text{Minimize} \sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} \times ash_m \qquad \forall y \in \{1,\dots,6\} \tag{3.4}$$

$$\text{Minimize} \sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} \times fines_m \qquad \forall y \in \{1,\dots,6\} \tag{3.5}$$

Subject to:

$$l_{m,t-1} + prod_{mt} - to_{mt} + lb_{mt} - \sum_{y=1}^{6} x_{ymt} = l_{mt} \quad \forall m \in \{1,\dots,6\},\ t \in \{1,\dots,24\} \tag{3.6}$$

$$l_{m0} = ls_m \qquad \forall m \in \{1,\dots,6\} \tag{3.7}$$

$$otons_{mt} = otons_{m,t-1} + to_{mt} - lb_{mt} \qquad \forall m \in \{1,\dots,6\},\ t \in \{1,\dots,24\} \tag{3.8}$$

$$l_{mt} \leq cap_m \qquad \forall t \in \{1,\dots,24\} \tag{3.9}$$

$$\sum_{y=1}^{6} x_{ymt} \leq trcap_m \qquad \forall m \in \{1,\dots,6\},\ t \in \{1,\dots,24\} \tag{3.10}$$

$$\sum_{m=1}^{6} x_{ymt} \leq 1,800 \qquad \forall y \in \{1,\dots,6\},\ t \in \{1,\dots,24\} \tag{3.11}$$

$$\sum_{m=3}^{6}\sum_{y=1}^{3} x_{ymt} \leq 1,800 \qquad \forall t \in \{1,\dots,24\} \tag{3.12}$$

$$\sum_{m=1}^{2}\sum_{y=4}^{6} x_{ymt} \leq 1,800 \qquad \forall t \in \{1,\dots,24\} \tag{3.13}$$

$$x_{ymt} \leq 1,000 active_{ymt} \qquad \forall y \in \{1,\ldots,6\},\ m \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.14}$$

$$\sum_{y=1}^{6} active_{ymt} \leq 1 \qquad \forall m \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.15}$$

$$\sum_{m=1}^{6} active_{ymt} \leq 1 \qquad \forall y \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.16}$$

$$\sum_{y=1}^{3}\sum_{t=1}^{24} recl_{yt} = demand_1 \tag{3.17}$$

$$\sum_{y=4}^{6}\sum_{t=1}^{24} recl_{yt} = demand_2 \tag{3.18}$$

$$recl_{yt} \leq \sum_{u=1}^{t-1}\sum_{m}^{6} x_{ymu} + 0.1ys_y - \sum_{u=1}^{t-1} recl_{yu} \qquad \forall y \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.19}$$

$$recl_{yt} \leq 1,800 \qquad \forall y \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.20}$$

$$recl_{yt} \leq 10,000 isrecl_{yt} \qquad \forall y \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.21}$$

$$\sum_{y=1}^{3} isrecl_{yt} \geq 2 \qquad \forall t \in \{1,\ldots,24\} \tag{3.22}$$

$$\sum_{y=4}^{6} isrecl_{yt} \geq 2 \qquad \forall t \in \{1,\ldots,24\} \tag{3.23}$$

$$active_{ymt}, isrecl_{yt} = \{0,1\} \qquad \forall y \in \{1,\ldots,6\},\ m \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.24}$$

$$x_{ymt} \geq 0 \qquad \forall y \in \{1,\ldots,6\},\ m \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.25}$$

$$l_{mt}, to_{mt}, lb_{mt}, otons_{mt} \geq 0 \qquad \forall m \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.26}$$

$$recl_{yt} \geq 0 \qquad \forall y \in \{1,\ldots,6\},\ t \in \{1,\ldots,24\} \tag{3.27}$$

The multiple objectives of the model are given in (3.1) through (3.5). Objective function (3.1) aims to maximize the sum total of all coal stacked on all the yards from all the mines in all time periods. This will ensure that coal is not left in the mine bunkers, but is moved to the coal handling facility. The amount of excess tons produced which is not moved to the stockpiles and for which there is no space in the mine's bunker, that will need to be thrown-out at the mine bunkers, are minimized in function (3.2). The third objective function, (3.3), ensures that coal is stacked to every yard in near equal quantities to ensure a stock balance between the two parts of the coal handling facility. Finally, to meet blend requirements, ash content (3.4) and fines content (3.5) need to be minimized. This will improve the gasification efficiency of the coal moved to the factory.

Equation (3.6) is responsible for the material balance throughout all of the time periods. It determines new bunker levels (for each time period) based on load-backs from outside stock piles, throw-outs to outside stockpiles as well as mine production and the quantity that is extracted and stacked from the mine bunker to any of the yards. In (3.7) the bunker levels in period 0 are set equal to the starting bunker levels. Equation (3.8) represents the material balance of coal lying outside each of the mine bunkers, by considering the previous amount of coal outside the bunker, coal thrown out of the bunker in the current period and coal loaded back into the bunker during the current period. Constraint (3.9) enforces the bunker capacity for each mine.

Constraint (3.10) ensures mine conveyor capacity for each mine to all possible yards. The stacker on each yard has a maximum stacking capacity for hour t, this stacking capacity is enforced by constraint (3.11). Additionally, (3.12) and (3.13) enforce conveyor capacity on all coal from mines on side 2 going to side 1's yards and all side 1 mines going to side 2's yards, as only two conveyors are utilized specifically for this cross stacking.

Equation (3.14) ensures that if any coal is stacked from mine m to yard y during period t, the $isactive_{ymt}$ variable takes on the value of 1. Using equation (3.15) it is ensured that coal from a mine cannot be stacked on more than one yard in a time period. Furthermore (3.16) ensures that only one mine can stack on a specific yard in any time period.

Equations (3.17) and (3.18) imply that enough coal needs to be reclaimed to cover an assumed deterministic demand. Constraint (3.19) enforces that only coal that is actually available (i.e. has previously been stacked on a yard) can be reclaimed; assuming that only 10% of starting inventory may be reclaimed in any period. Equation (3.20) further ensures that reclaiming capacity is not violated for any yard.

Constraint (3.21) ensures that if any reclaiming takes place from any yard y in a time period t, the variable $isrecl_{yt}$ takes on the value of 1. Equations (3.22) and (3.23) ensure that two or more reclaimers are assigned as active to both side 1 and side 2. Equations (3.24) through (3.27) define variable types and boundaries.

Although this simplified model shows correspondence to some scheduling problems investigated in literature (Brucker, 2004; Pochet and Wolsey, 2006), the full-scale model has very little representation to studied problems as it combines an extremely complex scheduling problem in a unique setting with batch sizing or lot sizing (Potts and Kovalyov, 2000; Brucker, 2004; Pochet and Wolsey, 2006) and blending problems often seen in refineries (Ierapetritou and Floudas, 1998; Jia and Ierapetritou, 2004; Méndez et al., 2006).

3.2.1 Incorporating Multiple Objectives

MIP's deal with solving problems based on the optimization of a single objective, subject to several constraints. Although MIP is used widely in decision-making processes, it has a major limitation which restricts the users of the technique to narrowing their problems to a single objective function (Ramesh and Cary, 1989; Rifai, 1996).

In many modern day optimization problems it becomes difficult to maximize a well defined utility function as many conflicting interests cannot be reduced to a common scale of cost or benefit (Rardin, 1998:373). It is thus conjectured that within this complex decision making environment the aim is rather to attempt to achieve a set of pre-established targets (or goals) (Tamiz et al., 1998).

To deal with the different objectives as identified for the coal scheduling problem, Goal Programming (GP) is proposed. GP is an extension of the MIP and is the preferred method as it reduces complex multiple objective trade-offs to a standard single objective program (Rardin, 1998:390).

The GP model does not optimize (maximize/minimize) the objective directly, it attempts to minimize the deviations between desired goals and the realized results (Rifai, 1996). Goals are expressed as equations (i.e. soft constraints) and a deficiency variable is assigned to each of these; the aim is now to minimize the deficiency for each goal. This leads to the following provisional objective function and soft constraints.

Additional input parameters:

γ_i = Weight for objective function i, where $i \in \{1, \ldots, 5\}$

Additional variables:

$d_i \triangleq$ Deficiency of goal i, where $i \in \{1, 2\}$

$sd_y \triangleq$ Deficiency of stacking goal per yard y, where $y \in \{1, \ldots, 6\}$

$ad_y \triangleq$ Deficiency of ash goal per yard y, where $y \in \{1, \ldots, 6\}$

$fd_y \triangleq$ Deficiency of fines goal per yard y, where $y \in \{1, \ldots, 6\}$

Multiple-objective model:

$$\text{Minimize } \sum_{i=1}^{2} \gamma_i d_i + \sum_{y=1}^{6} \Big[\gamma_3 sd_y + \gamma_4 ad_y + \gamma_5 fd_y \Big] \tag{3.28}$$

Additional constraints:

$$\sum_{y=1}^{6} \sum_{m=1}^{6} \sum_{t=1}^{24} x_{ymt} + d_1 \geq 120,000 \tag{3.29}$$

$$\sum_{m=1}^{6} \sum_{t=1}^{24} otons_{mt} - d_2 \leq 0 \tag{3.30}$$

$$\sum_{m=1}^{6} \sum_{t=1}^{24} x_{ymt} + sd_y \geq \sum_{y=1}^{6} \sum_{m=1}^{6} \sum_{t=1}^{24} x_{ymt}/6.5 \qquad \forall y \in \{1, \ldots, 6\} \tag{3.31}$$

$$\sum_{m=1}^{6} \sum_{t=1}^{24} x_{ymt} \times ash_m - ad_y \leq 5,833 \qquad \forall y \in \{1, \ldots, 6\} \tag{3.32}$$

$$\sum_{m=1}^{6} \sum_{t=1}^{24} x_{ymt} \times fines_m - fd_y \leq 5,833 \qquad \forall y \in \{1, \ldots, 6\} \tag{3.33}$$

Objective function (3.28) minimizes the deficiency between the goals for each of the objectives and their realized values. The goal values used are actual values based on service agreements in the coal scheduling environment. Each of the constraints calculates the difference between the realized value and the goal set for each of the multiple objectives.

The following expression, which includes a small fraction of the sum of the original objectives, must be added to the objective function value to ensure that an efficient solution is obtained. The small fraction is chosen small enough that, for the expression added does not significantly impact on the overall objective function values.

$$0.0001\Bigg[\sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} + \sum_{y=1}^{6}\sum_{t=1}^{24} x_{ymt} - \sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} \times ash_m - \sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} \times fines_m - \sum_{m=1}^{6}\sum_{t=1}^{24} otons_{mt}\Bigg] \tag{3.34}$$

3.2.2 Incorporating Stochastic Elements

Because varying factory demand has a major influence in the upstream operations of Sasol's Coal Value Chain (CVC), the only stochastic variables in the simplified model are the daily tons consumed by each of the factories.

The Two-Stage Recourse Programming technique is used as it is perfectly applicable in this environment as corrective actions can be taken after the extent of the uncertainty is known (Joubert and Conradie, 2005). When the planned schedule does not send enough coal to one of the factories to meet its demand, the coal handling business (CHF) needs to take recourse. This means that CHF will have to load coal back from its strategic inventory sources (dead stockpiles) onto the conveyor belts going to the factory, which needs more tons than was planned for in the original schedule. However, this process requires the use of earthmoving equipment known as Front End Loaders (FELs) which represent additional costs. The loading back process also generates additional fine coal, which has a negative effect on both coal processing and gasification.

The coal scheduling problem can be characterized by two stages. The first stage being the extraction, stacking and reclaiming of coal to and from normal stockpiles. The second stage is concerned with the corrective action (recourse) taken to compensate for uncertain customer demand. Corrective action takes the form of additional load in from "dead" or strategic stockpiles, as discussed earlier.

It is therefore necessary to appropriately model both first stage and possible second stage actions to provide a robust coal handling schedule. To model the stochastic nature of demand, equations (3.17) and (3.18) are expanded to (3.35) and (3.36):

Stochastic input parameters:

ϑ_j	$=$ Stochastic demand for coal at plant j, denoted by $\tilde{\zeta}_j$, where $j \in \{1, 2\}$
R	$=$ Number of demand realizations for both plants
ζ_{1r}	$=$ Realization r of $\tilde{\zeta}_1$ at plant 1, where $r \in \{1, \ldots, R\}$
ζ_{2s}	$=$ Realization s of $\tilde{\zeta}_2$ at plant 2, where $s \in \{1, \ldots, R\}$
p_{1r}	$=$ Probability of realization r at plant 1, where $r \in \{1, \ldots, R\}$
p_{2s}	$=$ Probability of realization s at plant 2, where $s \in \{1, \ldots, R\}$

Stochastic decision variables:

$dead_{1r}$	$\triangleq$ Dead stock loaded back as recourse for realization r at plant 1, where and $r \in \{1, \ldots, R\}$
$dead_{2s}$	$\triangleq$ Dead stock loaded back as recourse for realization s at plant 2, where and $s \in \{1, \ldots, R\}$

Adapted constraints:

$$\sum_{y=1}^{3}\sum_{t=1}^{24} recl_{yt} + dead_{1r} \geq \zeta_{1r} \qquad \forall r \in \{1, \ldots, R\} \tag{3.35}$$

$$\sum_{y=4}^{6}\sum_{t=1}^{24} recl_{yt} + dead_{2s} \geq \zeta_{2s} \qquad \forall s \in \{1, \ldots, R\} \tag{3.36}$$

Equations (3.35) and (3.36) ensure that every realization of plant demand is met by either reclaiming from normal, or a combination of normal and dead stockpiles. Possible reclaiming from dead stockpiles is included in the GP objective function as follows.

$$\begin{aligned}\text{Minimize } & \sum_{i=1}^{2}\gamma_i d_i + \sum_{y=1}^{6}\Big[\gamma_3 sd_y + \gamma_4 ad_y + \gamma_5 fd_y + \Big] \\ & - 0.0001\Big[\sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} + \sum_{y=1}^{6}\sum_{t=1}^{24} x_{ymt} - \sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} ash_m \\ & \qquad - \sum_{y=1}^{6}\sum_{m=1}^{6}\sum_{t=1}^{24} x_{ymt} fines_m - \sum_{m=1}^{6}\sum_{t=1}^{24} otons_{mt}\Big] \\ & + 100\sum_{r=1}^{R}\sum_{s=1}^{R} p_{1r} \times p_{2s}\Big[dead_{1r} + dead_{2s}\Big] \end{aligned} \tag{3.37}$$

The third part of (3.37) represents the expected reclaiming value from dead stockpiles with respect to the distributions of $\tilde{\zeta}_j$.

The distributions of the random demand variables seen in practise for the two factories are given in Table 3.1, using notation based on Kelton et al. (2002:585-598), with $N(\mu;\sigma)$ a Normal distribution (with mean μ and standard deviation σ) and $W(\alpha;\beta)$ a Weibull distribution (with shape parameter α and scale parameter β). These are the distributions that provides the best fit to the actual factory demand data when represented in histogram format.

Table 3.1: Simplified case demand distributions

Factory	j	**Variable**	**Distribution**
Side 1	1	$\tilde{\zeta}_1$	$N(57,100;1,200)$
Side 2	2	$\tilde{\zeta}_2$	$51,100 + W(1.84;2,860)$

These distributions can be approximated by discrete distributions such that $\{(\zeta_{1r}, p_{1r}), r = 1, ..., R\}$, with $p_{1r} > 0$, and $\{(\zeta_{2s}, p_{2s}), s = 1, ..., R\}$, with $p_{2s} > 0$.

To minimize distribution approximation errors, the number of subintervals (R) should be as large as possible, considering that computational workload drastically increases with the number of subintervals (Kall and Wallace, 1994:13). A relatively large $R = 15$ provides an acceptable compromise in this instance.

Samples are generated for each distribution of the demand parameters ζ_j^{μ}, for all factories $j = \{1, 2\}$ and $\mu = \{1, 2, ..., K\}$, with the sample size $K = 10,000$.

Each distribution sample is partitioned into 15 equidistant subintervals. For each of these subintervals an arithmetic mean $\overline{\zeta}_{1r}$ and $\overline{\zeta}_{2s}$ is calculated for the sample values ζ_1^μ and ζ_2^μ, respectively, in a particular sub-interval. This arithmetic mean provides an estimate for the conditional expectation of ζ_j for factory j (Joubert and Conradie, 2005).

For each of the subintervals the relative frequency (probability), p_{1r} for $\overline{\zeta}_{1r}$ or p_{2s} for $\overline{\zeta}_{2s}$ is calculated using

$$p_{1r} = \frac{k_{1r}}{K}, r \in \{1, ..., R\} \qquad \text{or} \qquad p_{2s} = \frac{k_{2s}}{K}, s \in \{1, ..., R\} \tag{3.38}$$

where k_{1r} and k_{2s} denotes the number of values for ζ_{1r} and ζ_{2s}, contained in the associated subinterval, respectively. These relative frequencies provide the probability that a specific instance of demand will realize. A specific demand instance is given by the arithmetic mean of a sub-interval.

3.3 Exact Solution as a Benchmark

The simplified model is solved with GAMS using the Cplex 9.0 solver within 4 seconds with an Intel Centrino 1.86GHz processor with 1GB of RAM. As the model is a simplified instance of the generic problem, the short solution time is expected to increase *dramatically* in the actual practical model.

The reliability of the schedule based on stochastic demand is compared with the reliability of the deterministic model. Both deterministic and stochastic schedules are compared against randomly generated demands (based on the demand distributions given in Table 3.1) to determine if recourse is necessary. The average number of times that recourse occurs when testing each of the schedules against the generated demands is calculated. The deterministic schedule, although producing a smaller objective function value, has a reliability of only 49.86%, whilst the proposed stochastic schedule yields a reliability of 91.09%. The increased reliability will eventually negate the smaller objective function value since less crisis management (i.e. recourse) will be required and this will consequently produce a lower actual "cost".

Although the simplified case study solves optimally in a reasonable time, the aim of this simplified model is to provide a generic platform for large-scale multi-objective, stochastic scheduling in the real-life industrial environment. With this in mind solution techniques for larger applications are investigated.

3.4 Approximate Solution Evaluation

The model solved using exact optimization is now solved with three metaheuristic algorithms. This enables one to compare both solution quality and execution times in the coal handling scheduling environment.

3.4.1 Simulated Annealing

The SA principles applied to solve simplified model are discussed below:

Neighbourhood The neighbourhood is split into two parts. The first is for stacking and is thus a $m \times y \times t$ matrix of tonnages to indicate how many tons is stacked from mine m to yard y in period t. In this case $m = \{1, ..., 6\}$, $y = \{1, ..., 6\}$ and $t = \{1, ..., 24\}$. The second part is for reclaiming and is thus a $y \times t$ matrix of tons, where $y = \{1, ..., 6\}$ and $t = \{1, ..., 24\}$, to be reclaimed from yard y in period t. A value in either matrix is increased or decreased (with an equal probability) with a specific amount of tons. The specific amount is the step size, which is an input parameter and is chosen as 1,000.

Stepping If a tentative move to a random neighbour has been made, the SA always accepts that neighbour if it has a better objective function value. However, the SA will also accept a neighbour with a worse objective function value than the current solution with a probability of $p = e^{\frac{\delta}{\tau(t)}}$, where δ is the change in objective function value (will be negative for worsening moves) and $\tau(t)$ is the current temperature at iteration t.

Initial temperature An initial temperature of 500 gives the best results for this specific case.

Temperature reduction The temperature is slowly reduced by $t(i+1) = t(i) - 0.5(\frac{\tau(t)}{t^{max}})$, where t is the current iteration number, $\tau(t)$ is the temperature for iteration t and t^{max} is the iteration limit (1,000 in the case study).

Ensuring feasibility The SA is only allowed to consider moves that are feasible. Feasibility is consequently tested for when the SA wants to move to a specific neighbour.

3.4.2 Tabu Search

The Tabu Search (TS) metaheuristic uses the same neighbourhood representation as the SA metaheuristic. The TS also uses most of the SA logic. The differences between the applied TS and the SA are summarized below:

- No temperatures or temperature reductions are included in the TS.
- Each iteration of TS chooses the best of *all* the immediate surrounding solutions (neighbours).
- The TS metaheuristic is not allowed to move back to recently visited solutions. These forbidden moves represents the Tabu List. The 10 most recently visited solutions are included in the Tabu List.
- Once a solution (not recently visited) is chosen, this move is accepted irrespective of whether it is an improving or worsening move.

3.4.3 Genetic Algorithm

The Genetic Algorithm (GA) is made up of the following major components and uses the same neighbourhood representation as the SA and TS:

Population Each iteration has a population of various solutions.

Elites The best solutions of the previous iteration are copied to the next generation.

Generating populations After cloning elites from the previous iteration, the rest of the population is generated by combining two of the solutions from the previous iteration.

Selection probability Each solution is assigned a probability for selection; the better the solution's objective function value, the higher its selection probability.

Selection process The two solutions are selected using a biased random method, giving fitter (better objective function) solutions a better chance of selection.

Crossover Two solutions are combined using a crossover procedure that swaps the stacking and reclaiming matrices of each solution to form two new solutions (offspring).

Mutation In a population of results, certain randomly selected solutions are randomly altered by changing either the stacking or reclaiming matrices to prevent convergence to local optimal solutions.

3.5 Selecting the Most Appropriate Approach

An Intel Centrino 1.86GHz processor with 1GB of RAM was used to solve the problem using the solution methods discussed above. The average results are summarized in Table 3.2 for different problem instances.

Table 3.2: Solution method comparison

Solution approach	**% above global optimal**	**Solution time (sec.)**
Exact optimization	0.00%	3.5
Simulated Annealing	0.19%	13
Tabu Search	0.09%	54
Genetic Algorithm	33.71%	34

Both the SA and TS metaheuristics come close to the global optimal solution. However, SA solves quicker than the TS. The GA does not provide a good solution as it is difficult to represent the problem solutions in such a way that a GA can improve it (refer to Goldberg (1989)'s Schemata Theorem). SA is therefore identified as the most promising of the metaheuristics.

However, for the case study SA takes more than twice as long as the exact solution approach. As exact solution time will increase drastically with problem size, SA is expected to be faster for the real-life model as its solution time is not expected to increase nearly as much as the exact solution time.

Consequently, the Simulated Annealing (SA) metaheuristic is proposed as the solution technique for larger applications of the model. According to Silva (2003), SA is easy to implement and is capable of handling almost any optimization problem and any constraint. Busetti (2005) states that SA was specifically developed for highly constrained problems, which makes it applicable to the practical model.

An overview of the SA metaheuristic is presented in the next section to provide a foundation for the development of an SA algorithm for the CHF scheduling model. The developed metaheuristic's design is then presented.

Chapter 4

The Simulated Annealing Approach

The Simulated Annealing (SA) method is one of the solution approaches that received a lot of attention in the past two decades. The first major breakthrough in applying the SA of solids to optimization was published by Kirkpatrick et al. (1983), who applied the approach suggested by Metropolis et al. (1953). In this approach the energy state of the metal being annealed corresponds to an objective function being minimized (Eglese, 1990).

SA is a metaheuristic solution approach for solving complex optimization problems (Rutenbar, 1989; Eglese, 1990; Silva, 2003; Busetti, 2005) and is often used to solve *NP-hard* combinatorial problems.

SA searches through the solution space $\boldsymbol{S}$, which is a set of all the possible solutions to the optimization problem. The objective function values of the solutions in $\boldsymbol{S}$ are defined by $\boldsymbol{z}$, where $\boldsymbol{z} = \{z_i\}_{i \in \boldsymbol{S}}$. The aim of the SA search is to find the best possible solution $i \in \boldsymbol{S}$, that minimizes the objective function $\boldsymbol{z}$ over $\boldsymbol{S}$ (Eglese, 1990), whilst ensuring feasibility of the suggested solution. This solution of SA is shown in (4.1), with the solution $z^* \in \boldsymbol{z}$ approximating the global optimal solution.

$$z^* \approx \min_{i \in \boldsymbol{S}} z_i \tag{4.1}$$

4.1 Annealing Analogy

The SA algorithm builds on the analogy between the physical annealing process of solids and the solution of combinatorial optimization problems. The physical annealing of solids refers to “the process of finding low energy states of a solid by initially melting the substance, and then lowering the temperature slowly” (Eglese, 1990).

Rutenbar (1989) and Eglese (1990) use an example where a solid is melted by applying high temperatures. The substance’s particles are randomly scattered in the molten form. The lowest possible energy configuration of the particles can be achieved by very slowly cooling the substance (Busetti, 2005). The minimum energy configuration is typically a perfect crystal, called the ground state of the solid (Rardin, 1998:690). However, if the cooling process is not done slow enough the molten substance will not achieve the ground state, but will settle into a “meta-stable” structure (i.e. only locally optimal), such as glass.

The analogy is further explained in Table 4.1.

Table 4.1: Analogies between annealing and optimization

Physical concept	**Optimization concept**
Different substance structures	Feasible solutions
Energy of a structure	Objective function
Melting temperature	Initial control variable (temperature)
Cooling rate	Temperature reduction speed
Rapid quenching	Descent algorithm
Ground state	Global optimal
Meta-stable structures	Local optima

Without allowing worsening moves, the SA becomes a descent algorithm. This descent algorithm is analogous to the rapid cooling down, or quenching, of the molten material into a locally optimal crystal structure, but not into the ground state of the substance.

4.2 The Basic SA Algorithm

The traditional, basic SA algorithm is illustrated in Figure 4.1 (adapted from Eglese (1990)).

The SA algorithm starts by choosing an initial solution, $i \in S$, either randomly or heuristically. The initial temperature is set to a large value. The SA then attempts to move to a random neighbour j of the current solution i. The change in objective function from the current solution to the selected neighbour is then calculated by $\delta = z_j - z_i$. If $\delta < 0$ (i.e. the selected neighbour has a better objective function value than the current solution), the algorithm accepts the move to solution j and therefore sets j as the new current solution. If $\delta \geq 0$, the attempted step to the selected neighbour is rejected if a random number, $0 \leq r \leq 1$, is less than a certain probability. The worsening move may be accepted as the current solution if the following equation is adhered to, with Γ the SA control parameter which is analogous to the *current* temperature of the annealing process.

$$r < e^{-\frac{\delta}{\Gamma}} \tag{4.2}$$

The SA is allowed to accept these worsening moves to prevent the algorithm from being trapped in local optima. Based on inequality (4.2), the SA has a lower probability of accepting moves that are much worse than the current solution (i.e. with a large δ). However, moves which are only slightly worse than the current solution (i.e. with a small δ) have a higher probability of being accepted. The control variable Γ is gradually reduced as the temperature of the metal being annealed would start decreasing.

In some applications the denominator in (4.2) (Γ) becomes $k_B\Gamma$, with k_B being the Boltzmann physical constant (Eglese, 1990; Peyrol et al., 1992). Most applications customize this constant to suit the specific application.

Some SA algorithms also place a limit on the number of times the internal loop is allowed to repeat without accepting a move. If the number of repeats is reached without a move being accepted, the current solution is kept and the algorithm moves on to the next iteration.

When only improving moves are accepted (by not considering equation (4.2)), the algorithm becomes a simple local search descent algorithm. The descent al-

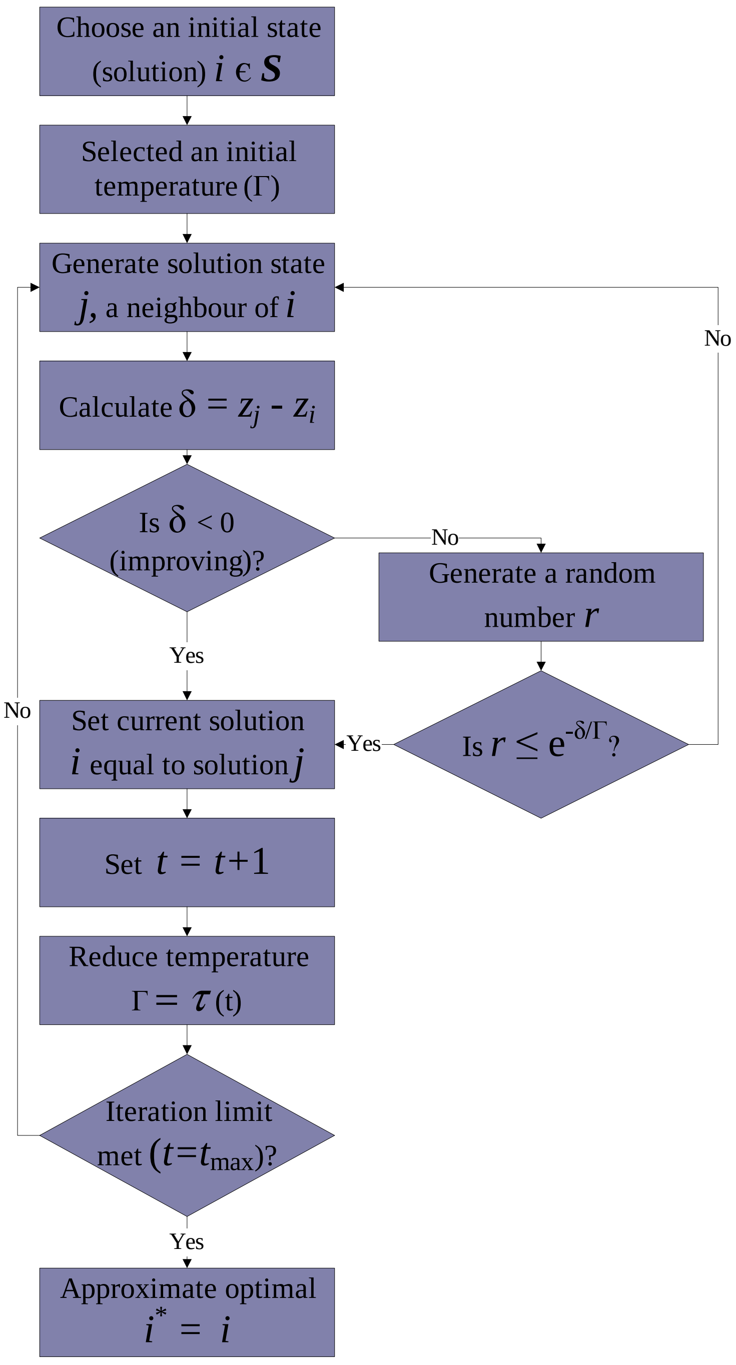

Choose an initial state (solution) $i \in S$
Selected an initial temperature (Γ)
Generate solution state j, a neighbour of i
Calculate $\delta = z_j - z_i$
Is $\delta < 0$ (improving)?
No
Generate a random number r
Is $r \leq e^{-\delta/\Gamma}$?
Yes
No
Yes
Set current solution i equal to solution j
Set $t = t+1$
Reduce temperature $\Gamma = \tau(t)$
Iteration limit met ($t = t_{max}$)?
No
Yes
Approximate optimal $i^* = i$

Figure 4.1: Traditional SA algorithm

gorithm will get stuck at local optimal solutions, as no further improvement is possible in any direction from local optimal points.

4.3 Applying SA to Solve Problems

The first important decision required when applying SA to a specific problem is to decide how the neighbourhood structure will be defined. This is an important decision as the effectiveness of the SA will depend on the neighbourhood structure. Eglese (1990) and Fleischer (1995) suggest that, in general, a smooth neighbourhood structure with shallow local minima is preferred to structures with many deep local minima.

A neighbourhood is seen as smooth when the typical steps made from one neighbour to another does not change the objective function value significantly (based on the problem context). If big jumps in objective function values occur between neighbours, many deep local minima will be present. This will require one to reduce the step size between neighbours or to apply a different neighbourhood structure.

The second set of choices that has to be made involves the cooling schedule. The cooling schedule of a SA metaheuristic is defined by determining various parameter values (Fleischer, 1995):

- Temperature reduction function $\tau(t)$, with t the iteration number.
- Initial temperature ($\tau(1)$).
- The temperature for iteration t, given by Γ, is equal to the function value of $\tau(t)$ at point t.
- Number of iterations (N).

The initial temperature is typically chosen large, corresponding to high temperature where the entire material is in a fully molten state (with a randomly dispersed structure). A high initial temperature ensures that nearly all moves are accepted during the beginning stages of algorithm execution.

The temperature reduction function is often used as a simple function, where the temperature of the next iteration is a fraction of the previous iteration's temperature. For example, $\tau(t+1) = \alpha\tau(t)$ (as illustrated in Figure 4.2), typically with $0.8 \leq \alpha \leq 0.99$ (Eglese, 1990; Rutenbar, 1989; Kirkpatrick et al., 1983). In

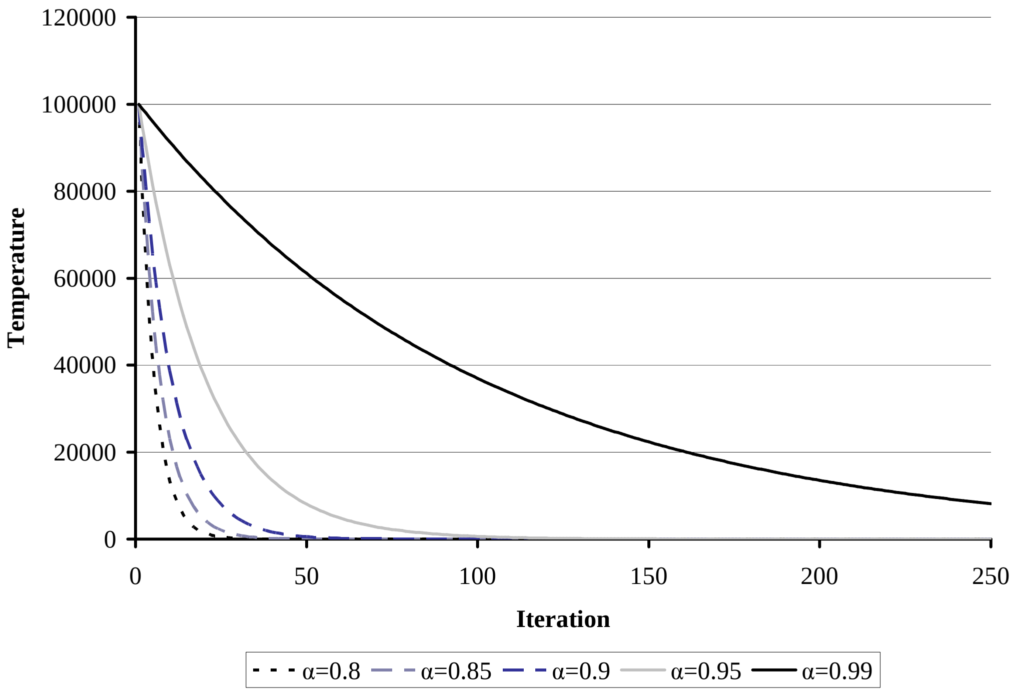


Figure 4.2: Temperature reduction function $\tau(t+1) = \alpha\tau(t)$

some applications, the temperature is fixed for a certain number of iterations before it is reduced in a stepwise manner (Peyrol et al., 1992), as shown in Figure 4.3. The function $\tau(t) = \frac{k_1}{1+tk_2}$ shown in Figure 4.4 has been used to ensure that the SA performs enough intensification (Eglese, 1990). Note that all k values are constants. Fleischer (1995) suggests a log-based function illustrated in Figure 4.5: $\tau(t) = \frac{k_3}{log(1+t)}$.

Most of these functions cause large temperature reductions in the earlier iterations with low temperature reductions later on. In applications where more diversification is required, it will be necessary to apply different temperature reduction functions to ensure that the temperature stays higher for longer periods.

The number of iterations is chosen so that enough neighbourhood steps are evaluated and so that the temperature is sufficiently close to 0 at the iteration

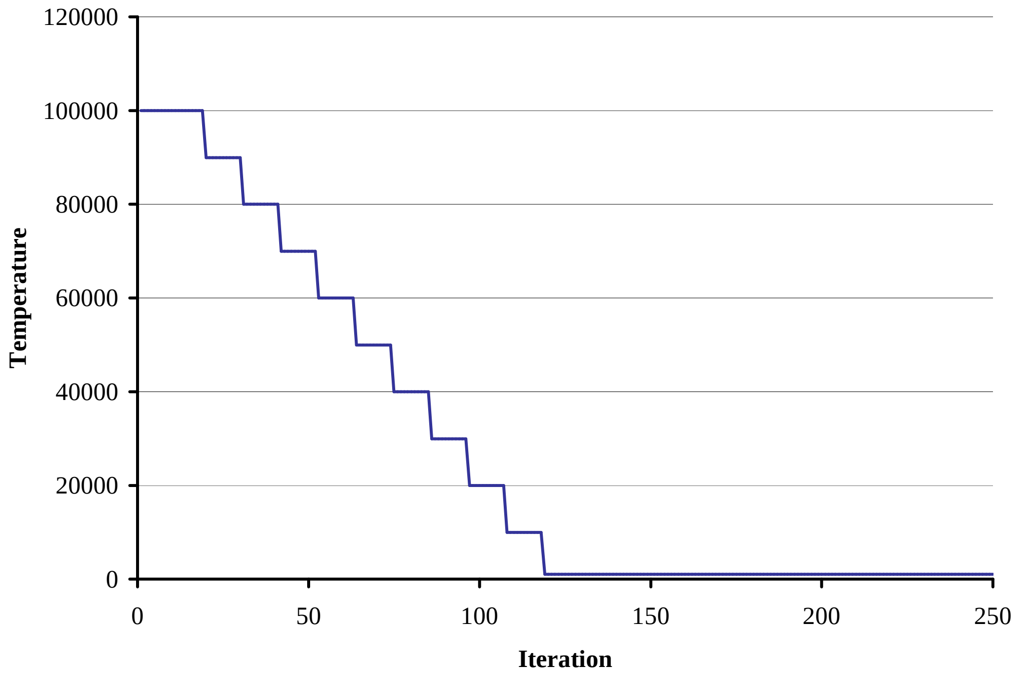


Figure 4.3: Step-wise temperature reduction function

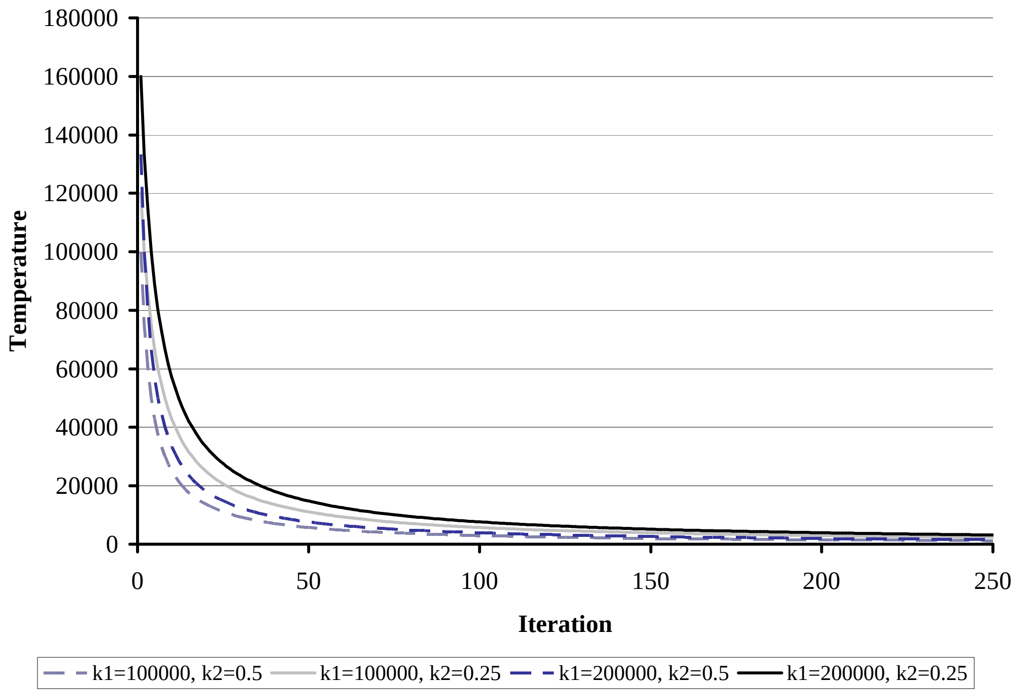


Figure 4.4: Constants based temperature reduction function

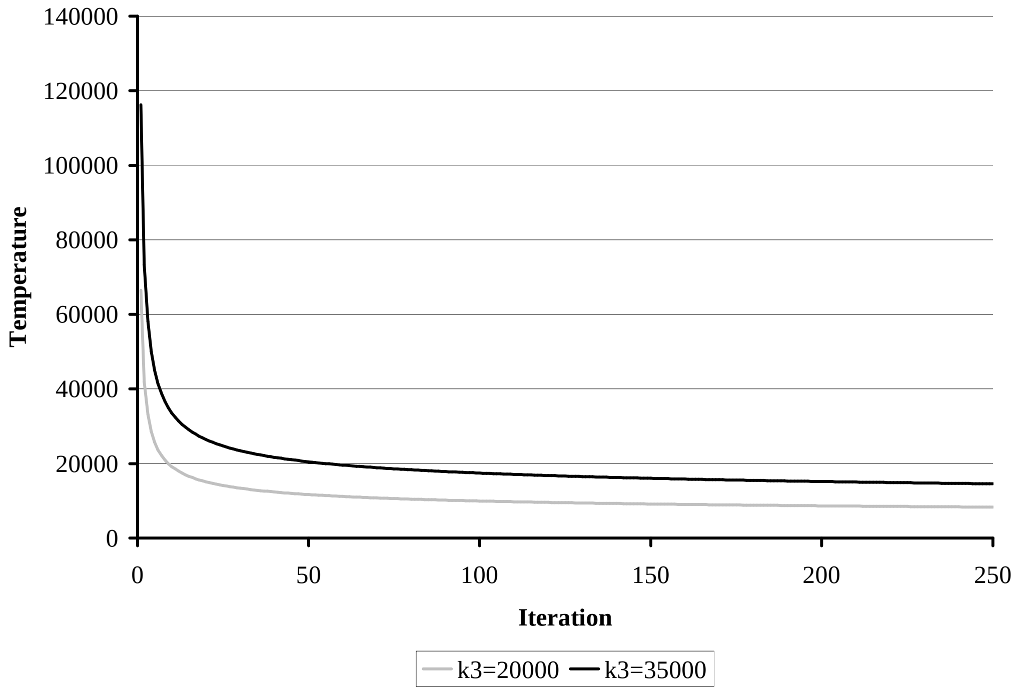


Figure 4.5: Log-based temperature reduction function

limit.

Eglese (1990) states that applying the traditional basic SA methodology, as discussed up to now, either very long execution times are required or poor solutions are produced if the number of iterations are limited. Various adaptations of the pure SA algorithm have consequently been implemented, which moves the algorithm further away from the annealing methodology it is based on, but which makes the algorithm more efficient. The most important of these modifications are discussed below.

Incumbent solutions Since the pure SA may accept worsening moves, it is possible that the solution of the final iteration is worse than a previously found solution. To address this problem, one simply keeps track of the best ever solution to date (the incumbent solution) to ensure that very good solutions are never lost in the SA execution. Glover and Greenberg (1989) claims that this approach enables shorter execution times as the pure SA algorithm has to perform a lot of intensification at the end to ensure that a good solution is reported.

Local optimization After the SA algorithm has executed, local optimization can be employed to improve the final solution.

Prevention of multiple neighbour visits At the end of the algorithm, a lot of computational time is spent on trying to improve moves (Eglese, 1990). This means that the SA starts evaluating moves that have already been evaluated in the current iteration. To prevent these repeated visits to neighbours, a rule can be applied that prevents the SA from visiting the same neighbour twice in one iteration. This approach guarantees that the final SA solution is at least a local optimal solution if the last n iterations did not provide improving moves, given a neighbourhood of size n.

Alternative acceptance functions In some applications the traditional probability of acceptance function, $e^{-\frac{\delta}{T}}$, is customized to suit the application.

Good initial solutions SA is quite sensitive to the quality of the initial solution. Therefore, many applications start with generating a good initial solution, rather than simply picking a random starting point. With a good initial solution, the initial temperature should also be reduced to prevent the SA from diverging so much that the time and effort spent on generating a good initial solution is negated.

Parallel computing Parallel computing can be used when multiple neighbours are evaluated at the same time (Rutenbar, 1989).

Alternative cooling schedules The temperature reduction process is sometimes customized to fit specific applications (Fleischer, 1995).

Hybrid metaheuristics The most applicable concepts from Genetic Algorithms and Tabu Search is often combined with SA to create an effective and efficient hybrid solution method.

SA has various advantages which makes it quite popular (Eglese, 1990; Rutenbar, 1989; Peyrol et al., 1992):

- SA is easy to implement as the SA part of the algorithm is short (which is essential when the problem environment is extremely complex). Only the neighbourhood structure might cause difficulty during implementation.
- SA is widely applicable.
- SA can provide good solutions in reasonable time, given that the neighbourhood is properly defined.
- Infeasible solutions may be allowed to ensure easier transition to good areas in the solution space by using penalty functions.
- SA is efficient when "good" solutions are acceptable and a very good approximate to the global optimal solution is not as important.

One should, however, keep in mind that just as some materials resist annealing, some problems just cannot be represented in a neighbourhood structure that will make SA applicable to that problem (Rutenbar, 1989). SA should therefore be used as one of the various methods to solve difficult problems, not as a universal tool to solve all optimization problems.

Having reviewed SA theory, an SA algorithm is developed for the CHF scheduling problem.

4.4 SA for Scheduling Coal Handling

The conveyor belt system at CHF that needs to be scheduled, as shown in Figure 4.6, illustrates the scope of the scheduling model. The scheduling model starts from the mine surface bunkers at the production mines, all the way through CHF and into the Coal Processing (CP) bunkers (which form the start of the Synfuels production process).

The SA scheduler was coded in Microsoft® Visual Basic for Applications (VBA), which makes it easy to update model inputs using Microsoft® Excel spreadsheets. Only Microsoft® Office Excel is required for the SA, making it globally accessible without incurring additional licence costs. The SA together with inputs and graphical outputs (for easy interpretation) are embedded in one spreadsheet file.

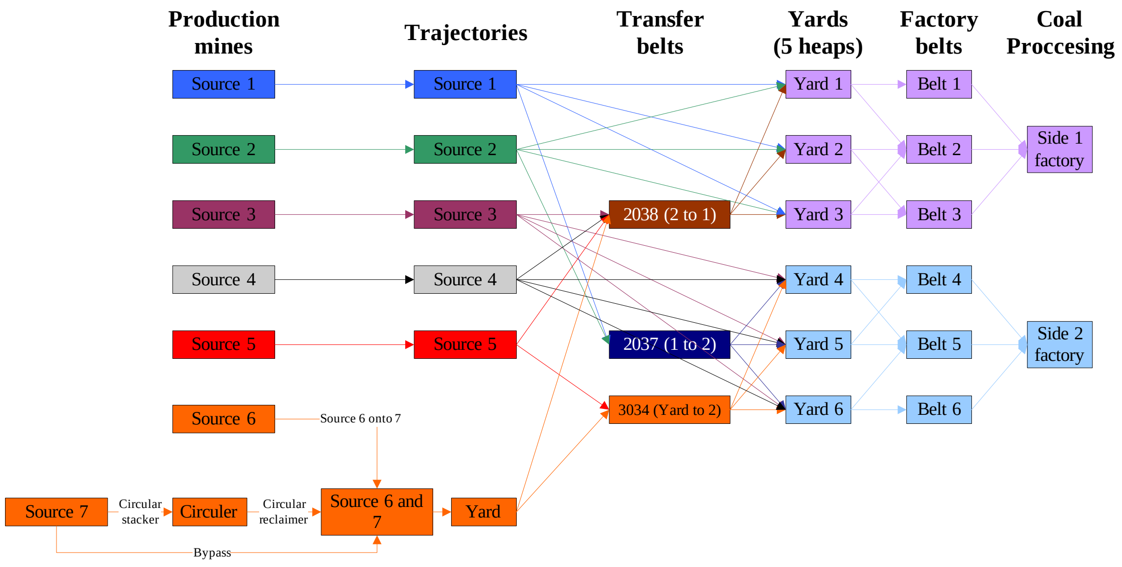


Figure 4.6: Elements included in the SA model of the coal handling facility

Notes on how to access the code of the developed program, along with an explanation of variable definitions and subscripts used, can be found in Appendix A. The steps required to run the SA scheduler are given in Appendix B.

Most of the input data required for the scheduling model is available on a live basis. This data is then pulled from the relevant database or process value historian into the *Input_rev12.xls* input workbook. Other operational rules are also captured in this input sheet. The *Input_rev12.xls* workbook serves as the master input file for the SA scheduler as well as for the weekly blend planning model and the blend validation simulation model used by CHF. *Input_rev12.xls* also includes various macros to fix and/or validate plant data that might include errors made by control room operators.

From *Input_rev12.xls*, the information required for the scheduler is pulled through to the *SA scheduler input.xls* workbook. All the data pulled from *Input_rev12.xls* is included in *SA scheduler input.xls* in the format required by the SA. This file then serves as the direct input into the SA metaheuristic (*Simulated Annealing for CVC v8.xls*).

The typical SA approach is used in this scheduling application. The approach used in this application is summarized in Figure 4.7.

Firstly an initial solution is generated. Thereafter the algorithm attempts

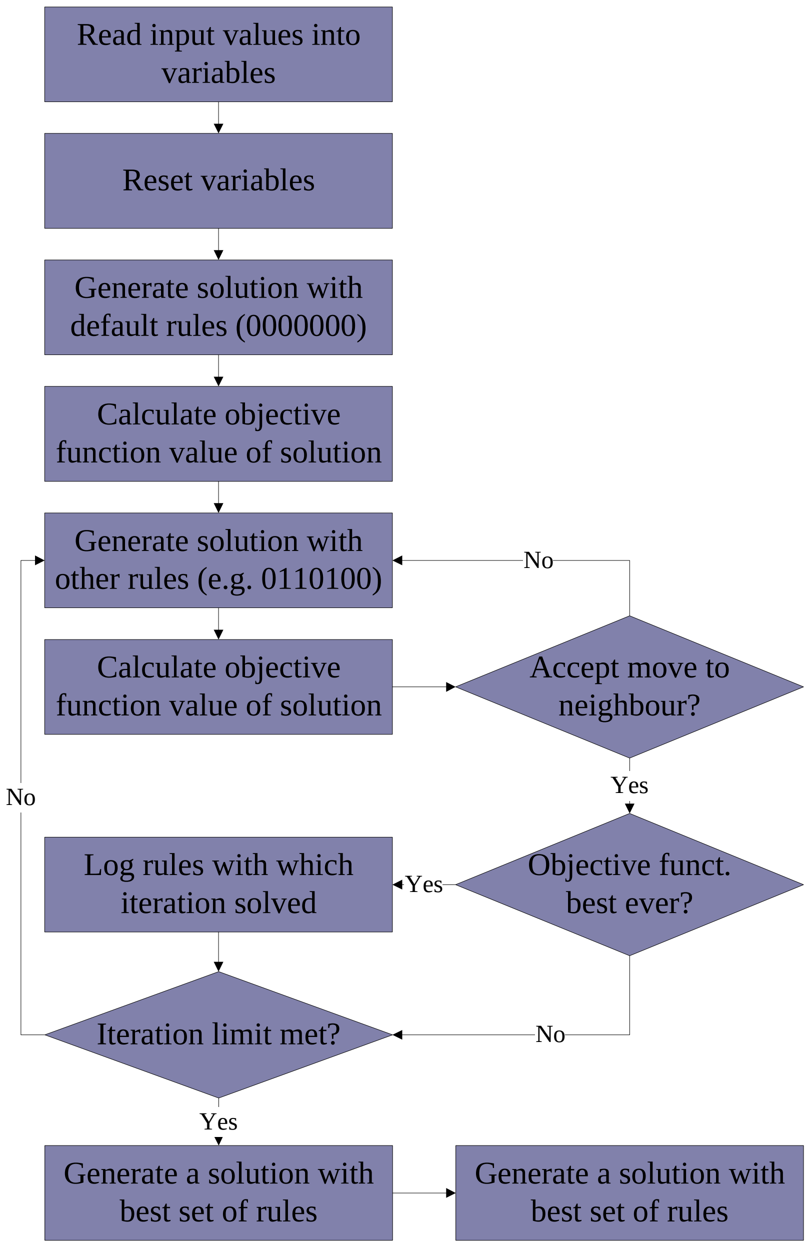

Read input values into variables
Reset variables
Generate solution with default rules (0000000)
Calculate objective function value of solution
Generate solution with other rules (e.g. 0110100)
No
Calculate objective function value of solution
Accept move to neighbour?
Yes
No
Log rules with which iteration solved
Yes
Objective funct. best ever?
Iteration limit met?
No
Yes
Generate a solution with best set of rules
Generate a solution with best set of rules

Figure 4.7: SA approach used in this application

to move to a randomly selected neighbour. If this is an improving move, the algorithm accepts this move as the next iteration's solution. However, if the move is not improving, it may be accepted with a certain probability. This probability is reduced from iteration to iteration to ensure good diversification at the start of the algorithm and more intensification at the final iterations.

To compliment the SA overview, it is necessary to explain the manner in which time points are implemented.

4.4.1 Continuous Time Points in a Discrete Environment

Swart (2004) found that continuous time point scheduling models provide better objective function values than discrete time points in the CHF environment, hence the SA scheduler should utilize continuous time points. In a metaheuristic environment this is extremely difficult as metaheuristics are typically aimed at solving combinatorial optimization problems (Eglese, 1990; Rutenbar, 1989), i.e. problems in a discrete environment.

Continuous time points are often estimated by using very small discrete time buckets. In the CHF environment, one would require time buckets of less than 5 minutes each. As a schedule is required for a 48 hour horizon, the model would become extremely large (more than 576 time points, with different usages of these time points for every piece of CHF equipment). Consequently, alternative ways of implementing continuous time points using a deterministic environment have to be developed.

This dissertation suggests the following method of providing a continuous time schedule using discrete time points. The SA works with hourly buckets (i.e. 48 time points in the scheduling horizon). This would provide an impractically crude schedule. To overcome this, more than one coal movement should be allowed to occur within an hour bucket. A change from one movement to another may occur at any point in the hourly time buckets. This approach will effectively produce continuous time points (refer to Figure 4.8, illustrating 5 time buckets), although it complicates the solution generation process drastically.

With this approach, only two movements can occur within one hour. Although this is a simplifying assumption, it matches CHF practice (as having three move-

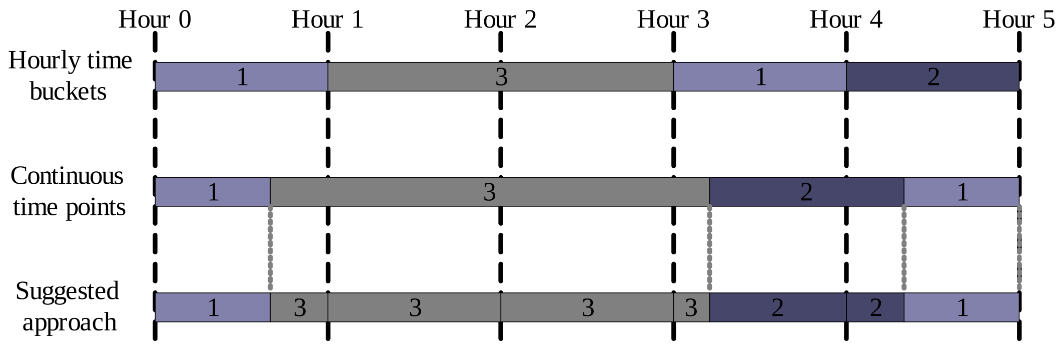


Figure 4.8: Creating continuous time points using discrete time buckets

ments per hour is unrealistic, since this would cause too small movements in an environment where the smallest movements are typically more than 2,000 ton - given typical CHF capacities of 1,800 ton/hour).

To start drilling deeper into the developed SA algorithm, the selected neighbourhood structure is now explained.

4.4.2 Neighbourhood Solutions

A slightly different interpretation of operational scheduling rules will generate different solutions, although these solutions will be closely related. The neighbourhood of a solution is therefore seen as a schedule generated by slightly changing one of the rules used to generate solutions. The rules used to define neighbouring solutions are shown in Table 4.2, for rule categories $\bar{\rho} = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, \rho_7\}$.

The set of rules used by the scheduler is denoted by $\bar{\rho}$. For example, the default solution (initial solution) is generated by using rule 0 in all the rule categories, i.e. $\bar{\rho} = \{0, 0, 0, 0, 0, 0, 0\}$.

The above mentioned neighbourhood structure is decided upon as this structure is smoother than other neighbourhood structures (such as swapping scheduling actions around). It is nearly impossible to work with a structure where swapping is done, due to the effects that a small change in the planned actions have on the entire system. A neighbourhood defined by swapping will therefore require inordinate amounts of computational time and will cause various deep local minima. The chosen structure (rules-based) is much easier implemented and provides a

Table 4.2: Schedule development rules

Category	Category description	Rule 0	Rule 1	Rule 2	Rule 3	Rule 4
ρ_1	Reclaiming rule	Based on quality	Based on yard space	-	-	-
ρ_2	Bunker level to start new heap	70%	60%	50%	90 %	80%
ρ_3	Yard stacking priority	Reclaimable tons	Space on yard	-	-	-
ρ_4	Mine stacking priority	Bunker fullness	Blend space for mine	-	-	-
ρ_5	Split between reclaimers	100% first	75% first	60% first	50% first	-
ρ_6	Bunker level for over-blending	85%	80%	75%	90 %	95%
ρ_7	New heap length factor	80%	90%	95%	70 %	60%

smoother neighbourhood with mostly shallower local minima (which Eglese (1990) states will lead to a more effective algorithm).

Given this neighbourhood, the SA algorithm has to choose a random neighbour of the current solution. This process is now discussed.

Firstly a random rule category is selected by generating a random number ψ, with $\psi \in \{1, 2, 3, 4, 5, 6, 7\}$. Once a value for ψ has been determined, a single step is taken (in a random direction) within the applicable set (ρ_ψ), where each set represents all the possible values in each rule category. With ρ_ψ, related to the categories and rules listed in Table 4.2, defined as follows.

$\rho_1 \in \{0, 1\}$

$\rho_2 \in \{0, 1, 2, 3, 4\}$

$\rho_3 \in \{0, 1\}$

$\rho_4 \in \{0, 1\}$

$\rho_5 \in \{0, 1, 2, 3\}$

$\rho_6 \in \{0, 1, 2, 3, 4\}$

$\rho_7 \in \{0, 1, 2, 3, 4\}$

If the initial solution is generated with $\bar{\rho} = \{0, 0, 0, 0, 0, 0, 0\}$, the next iteration will be generated as follows. Firstly random rule ψ to be perturbated is selected, after which a random direction of movement is selected. If $\psi = 2$ is selected and the direction chosen is increasing, the value for ρ_2 will now increment from 0 to 1. The following iteration will consequently be generated using the following set of rules: $\bar{\rho} = \{0, 1, 0, 0, 0, 0, 0\}$.

Once a neighbour for the following iteration $(t + 1)$ has been selected, its objective function (z_{t+1}) is evaluated. Improving moves $(z_{t+1} > z_t)$ are accepted and the SA algorithm accepts this neighbour as the following iteration's solution. However, non-improving moves $(z_{t+1} \leq z_t)$ are also sometimes accepted, based on the probability (κ) calculated in (4.3), with z_t representing the objective function value at iteration t and with $\tau(t)$ the temperature at iteration t:

$$\kappa_t = e^{\frac{-(z_{t+1}-z_t)}{\tau(t)}} \tag{4.3}$$

$\tau(1)$ is set equal to the initial temperature, as input by the user. The manner in which the current temperature Γ of iteration t, with $\Gamma = \tau(t)$, is reduced over iterations is now discussed.

4.4.3 Cooling Schedule

The cooling schedules found in literature do not enable the user to control when and how rapidly the algorithm changes from diversification to intensification and tend to start intensifying the search too early during algorithm execution. Therefore this dissertation proposes a new cooling schedule which addresses both of these shortcomings.

The temperature at each iteration $\tau(t)$ is defined by the function in (4.4), with $\tau(1)$ the initial temperature, N the iteration limit, w the time when sudden temperature reduction starts and k the speed of sudden temperature reduction.

$$\tau(t) = \frac{\tau(1)}{\pi}\left(- tan^{-1}(w \cdot k \cdot N - k) + \frac{\pi}{2}\right) \tag{4.4}$$

This function is easier understood in graphical format (with $w = 2$, $k = 6$ and $\tau(1) = 100,000$), as shown in Figure 4.9 for 150 iterations.

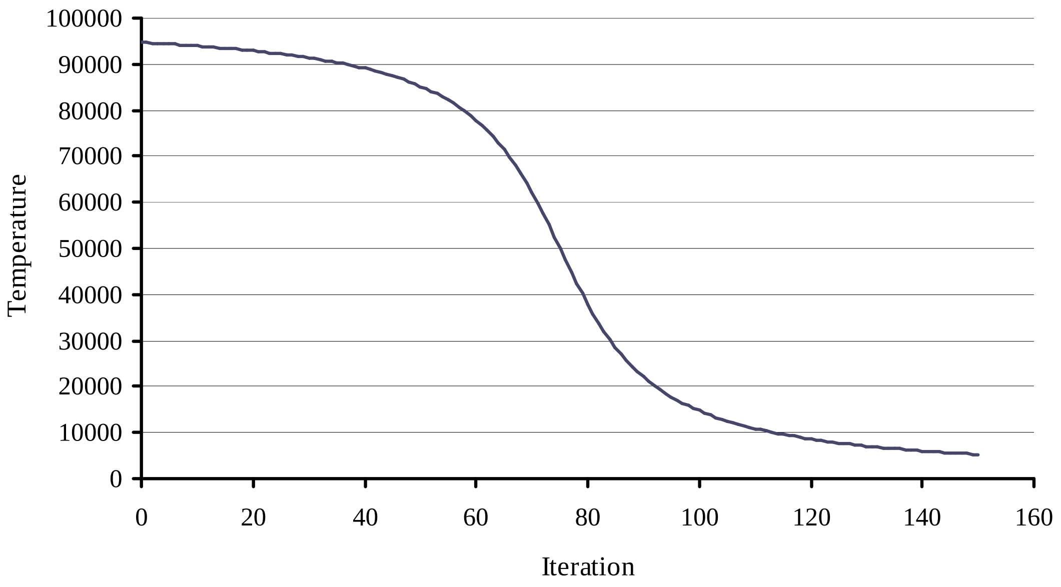


Figure 4.9: Temperature τ reduction over iterations t

From Figure 4.9 it can be seen that the temperature suddenly starts declining half way through the iterations. The speed with which this sudden decrease occurs, is set by the variable k, as shown in Figure 4.10 (with $w = 2$, $\tau(1) = 100,000$ and 150 iterations).

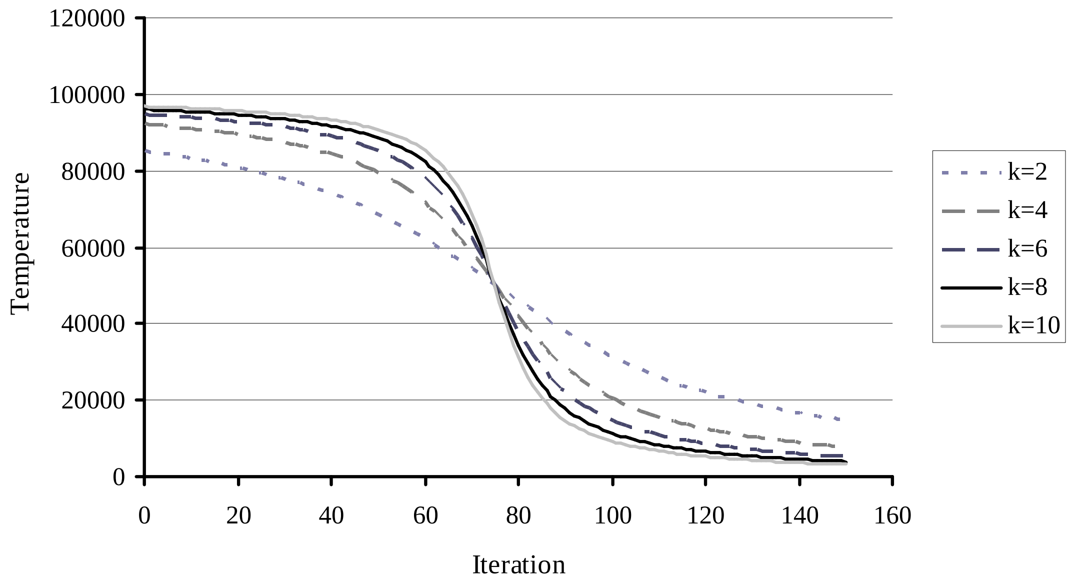


Figure 4.10: Temperature τ reduction with varying k

The w variable determines at what stage the sudden temperature decrease occurs. The effect of changing w is explained in Figure 4.11, with $k = 4$, $\tau(1) = 100,000$ and 150 iterations.

The variables w, k and the initial temperature have an effect on the temperature value $\tau(t)$ per iteration t. The delta between objective function values also has an effect on the probability of accepting the worsening move. This is shown in Figure 4.12, with $k = 4$, $w = 2$, $\tau(1) = 100,000$ and 150 iterations.

It is consequently seen that the initial temperature has to be varied according to the typical delta values that will occur during execution. With high objective function values, where the delta value can be as high as 100,000, the initial temperature should be large (see Figure 4.12); typically larger than 100,000 for this example. However, when smaller objective function values exist, the delta values will also be smaller. Therefore a smaller initial temperature is required to ensure that similar probabilities of acceptance are applicable when the delta value is the same fraction of the initial solution.

With delta values of less than 10,000, the initial temperature will have to be much smaller. The smaller initial temperature will ensure that the probability

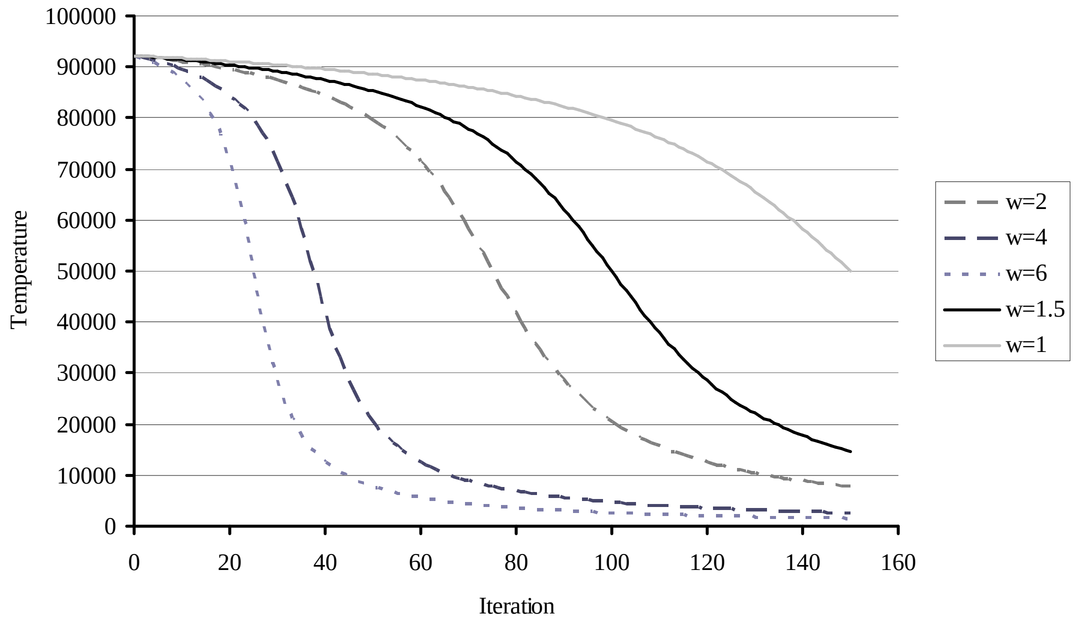


Figure 4.11: Temperature τ reduction with varying w

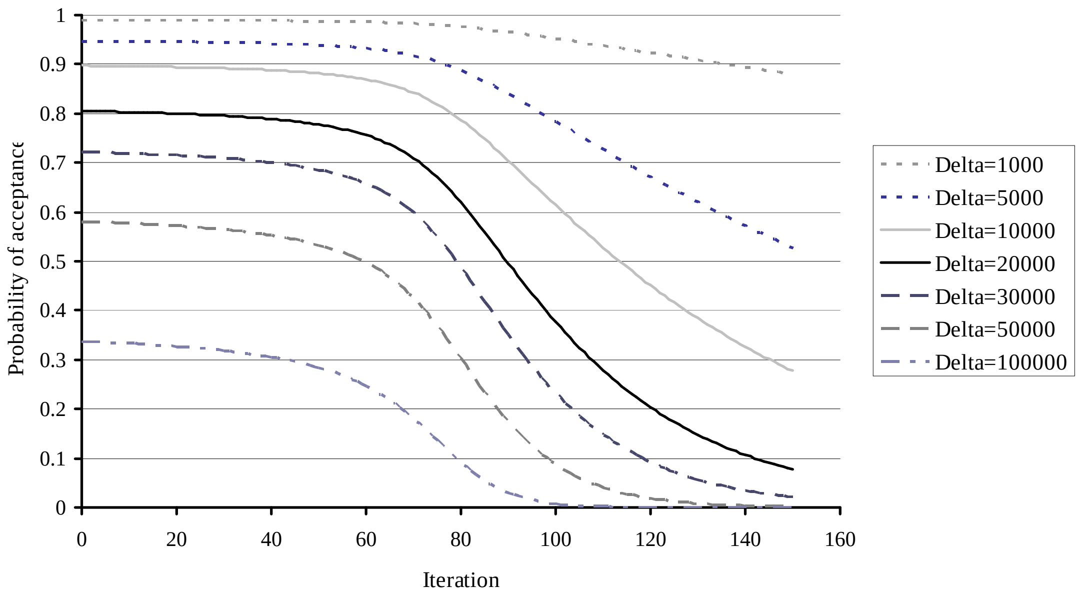


Figure 4.12: Probability of acceptance with varying delta for $\tau(1) = 100,000$

of accepting moves that worsen the objective function by a certain fraction, is the same for small delta values in an instance with low objective functions and for higher delta values (the value is higher, but it is the same fraction of typical objective function values) in an instance with higher objective function values. Figure 4.13 shows the same information as given in Figure 4.12, but with an initial temperature of 15,000.

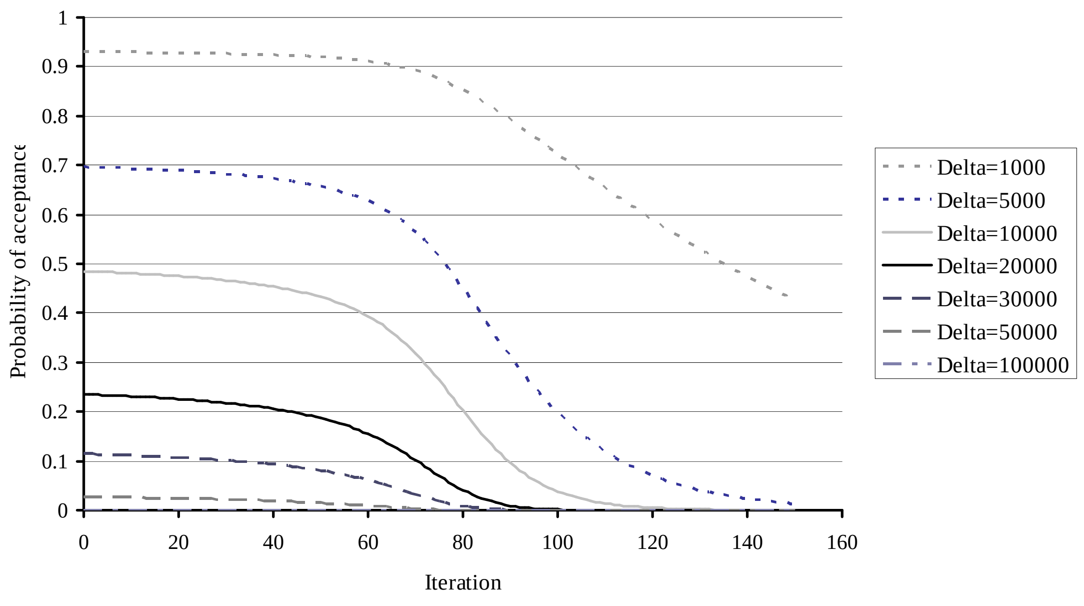


Figure 4.13: Acceptance probability with varying delta for $\tau(1) = 15,000$

The initial temperature should therefore be set as a multiple of the objective function of the initial solution (i.e. the solution with $\bar{\rho} = \{0, 0, 0, 0, 0, 0, 0\}$). This specific multiple value is determined in Section 5.1.

Now that the SA metaheuristic design has been discussed, it is necessary to determine the model's objective function value. This is done by discussing how the SA will incorporate multiple objectives and stochastic elements as these elements determine the model's objective function.

4.4.4 SA Objective Function

The CHF operational scheduling environment represents multiple conflicting objectives. Incorporating stochastic elements into the model adds an additional term to the objective function, as the expected value of recourse also needs to be minimized.

Multiple Objectives

The objective function used in the model is based on the weighted sum of the deficiency by which each objective did not achieve its goal (i.e. the modified GP approach, as discussed in Section 2.1.2). The following conflicting objectives are applicable.

- Total throw-outs at all mine bunkers for the 48 hours, with a target of 1,000.
- Total tons prop-chuted during the 48 hours, with a goal of 0.
- 5 times the number of times that 3% over-blending was allowed due to full mine bunkers, with a goal of 0.

These objectives require the following decision variables to be defined:

d_{to}	$\triangleq$ Deficiency of the throw-out goal, for all mines in all periods
d_{pc}	$\triangleq$ Deficiency of the prop-chute goal
$d_{+3\%}$	$\triangleq$ Deficiency of the over-blending goal
to_{bt}	$\triangleq$ Throw-out at bunker b in hour t, where $b \in \{1, \ldots, 8\}$ and $t \in \{1, \ldots, 48\}$
pc_{mt}	$\triangleq$ Prop-chute of mine m in hour t, where $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 48\}$
oc_{yhmt}	$\triangleq$ Count of over-blending allow on heap h on yard y of mine m in hour t, where $h \in \{1, \ldots, 5\}$, $y \in \{1, \ldots, 6\}$, $m \in \{1, \ldots, 6\}$ and $t \in \{1, \ldots, 48\}$

The objective function used to minimize the weighted deficiencies is shown in (4.5).

$$\text{Minimize } d_{\text{to}} + d_{\text{pc}} + 5d_{+3\%} \tag{4.5}$$

The values of the deficiencies are calculated by including the following constraints in the model:

$$\sum_{b=1}^{8}\sum_{t=1}^{48} to_{bt} - d_{\text{to}} \leq 1,000 \tag{4.6}$$

$$\sum_{m=1}^{6}\sum_{t=1}^{48} pc_{mt} - d_{\text{pc}} \leq 0 \tag{4.7}$$

$$\sum_{y=1}^{6}\sum_{h=1}^{5}\sum_{m=1}^{6}\sum_{t=1}^{48} oc_{yhmt} - d_{+3\%} \leq 0 \tag{4.8}$$

To guarantee that this GP approach will produce an efficient solution, a small fraction of the sum of the different objectives should be included in (4.5). The model's objective function, shown in (4.9), is based on the modified GP approach which guarantees efficient solution points (since the small fraction is included in the objective).

$$\begin{aligned}\text{Minimize } & d_{\text{to}} + d_{\text{pc}} + 5d_{+3\%} \\ & + 0.001\left(\sum_{b=1}^{8}\sum_{t=1}^{48} to_{bt} + \sum_{m=1}^{6}\sum_{t=1}^{48} pc_{mt} + \sum_{y=1}^{6}\sum_{h=1}^{5}\sum_{m=1}^{6}\sum_{t=1}^{48} oc_{yhmt}\right)\end{aligned} \tag{4.9}$$

Stochastic Elements

Although both the production from mines as well as the demand from plants are stochastic in nature, for the purpose of the scheduler only the stochastic demand will be considered, as mine bunkers are specifically aimed at buffering the fluctuations in mine production. Kall and Wallace (1994) state that stochastic demand can be appropriately modelled as an uncertain parameter that is characterized by a probability distribution, assuming that the probability function is known.

The amount of recourse is only calculated for the first 24 hours of the schedule, since CHF reschedules every 24 hours. This is done to reduce the execution time. Recourse is applied in this model by calculating the total corrective actions required in the first 24 hours of the schedule, given a specific realization of daily factory demand and the total planned reclaiming tasks. Both factories have unique demand distributions and therefore will require different amounts of recourse (given the different demands and different amounts of total coal reclaimed in the first day, per side).

The same approach as applied in the simplified scheduling model is used to incorporate stochasticity in this SA application. The only difference is that the demand distribution is now applied as a fraction of the planned factory demand (per side). The demand factor distributions are the best-fit distributions to actual factory consumptions seen in practice and are given in Table 4.3 with $N(\mu;\sigma)$ a Normal distribution (with mean μ and standard deviation σ) and $W(\alpha;\beta)$ a Weibull distribution (with shape parameter α and scale parameter β).

Table 4.3: Practical case demand distributions

Factory	j	**Variable**	**Distribution**
Side 1	1	$\tilde{\zeta}_1$	$N(1.0043;0.0392)$
Side 2	2	$\tilde{\zeta}_2$	$0.921+W(1.55;0.0805)$

Based on these distributions, the probability of occurrence (and the function value of the distribution at this point) is calculated for 15 realizations, exactly the same as explained in Section 3.2.2. The value of the distribution at each realization point is calculated with the formula given in (3.38).

The expected value of the recourse required for both factories, for all possible combinations of the factory demands, are added to the objective function of this model. Given the following notation, the model's objective function now becomes (4.10).

Input parameters:

p_{1r} $=$ Probability of realization r at plant 1, where $r \in \{1,\ldots,R\}$

p_{2s} $=$ Probability of realization s at plant 2, where $s \in \{1,\ldots,R\}$

Stochastic decision variables:

$dead_{1r}$ $\triangleq$ Dead stock loaded back or thrown out as recourse for realization r at plant 1, where and $r \in \{1,\ldots,15\}$

$dead_{2s}$ $\triangleq$ Dead stock loaded back or thrown out as recourse for realization s at plant 2, where and $s \in \{1,\ldots,15\}$

$$\begin{aligned}\text{Minimize } & d_{\text{to}} + d_{\text{pc}} + 5d_{+3\%} \\ & + 0.001\left(\sum_{b=1}^{8}\sum_{t=1}^{48} to_{bt} + \sum_{m=1}^{6}\sum_{t=1}^{48} pc_{mt} + \sum_{y=1}^{6}\sum_{h=1}^{5}\sum_{m=1}^{6}\sum_{t=1}^{48} oc_{yhmt}\right) \\ & + \sum_{r=1}^{15}\sum_{s=1}^{15} p_{1r} \times p_{2s}\left(dead_{1r} + dead_{2s}\right) \end{aligned} \tag{4.10}$$

Expression (4.10) represents the model's final objective function which the SA optimizes, given the CHF infrastructure and operational philosophy constraints.

The previous sections provide an overview of the SA, its neighbourhood definition, the cooling schedule used and the objective function of the model. These elements work together to approximate the global optimal solution. However, the SA must also generate each of the solutions used in all iterations. The next section explains how this neighbourhood solution generation process works.

4.5 Coal Handling Solution Generation

The SA provides a set of rules $\bar{\rho}$ for a specific neighbourhood solution. These rules are used to generate a feasible solution for that neighbour so that the objective function value of the neighbourhood solution can be evaluated. The SA then uses that objective function value to determine which neighbourhood solution should be considered next. This section describes how the detailed feasible solution associated with a set of rules, provided by the developed SA logic, is generated.

For every iteration, the SA algorithm considers a neighbourhood solution. Each of these solutions is made up of actions scheduled in each of the 48 hourly time buckets (wherein there can be continuous time points). The combination of the 48 buckets provides one solution, for which an objective function value can be determined.

Each solution is generated by consecutively developing the continuous tasks required in the 48 buckets. Much effort is required by the SA solution generation code to ensure that each bucket considers its impact on other buckets and the complete final solution.

For each of the hourly time buckets a feasible sub-solution has to be generated. Each bucket's feasible sub-solution must link with the other buckets' solutions to generate a feasible 48 hour schedule. This section discusses how the hourly solutions are created. The overview of the solution generation process is given in Figure 4.14. This process is repeated for every iteration.

The first step in Figure 4.14 performs the calculations for the initial setup required for some of the Microsoft® Visual Basic for Applications (VBA) input values.

For the first hour, the *time* variable is set to 1 and the first set of rules is set to default, i.e. $\bar{\rho} = \{0,0,0,0,0,0,0\}$. For subsequent hours, the *time* is incremented by one and a random neighbouring $\bar{\rho} = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, \rho_7\}$ is selected.

To start generating the current hour's sub-solution, the reclaiming part of the schedule is solved first. The operational rules used for creating the reclaiming actions are determined by $\rho_1 \in \{0,1\}$ (the rule used to decide which yard's reclaiming heap to reclaim) and $\rho_5 \in \{0,1,2,3\}$ (the fraction of capacity the two reclaimers per side is run at). The reclaiming part of the solution generation processes decides which of the six reclaimers (of which there are one per yard) should be run at which rate to keep the two factory bunkers full to ensure that the liquid petroleum production process is never interrupted. The reclaiming part also decides which heaps on which yards should be reclaimed.

Some calculations are then performed to adapt the required variables to the reclaiming actions decided upon.

The stacking part of the solution generation process decides which mine should be stacked on which heap, on which yard, at what rate, whilst ensuring that the blend on that specific heap is adhered to and considering hourly mine production rates. The effect that the stacking actions will have on the mine bunkers is also considered to ensure that the mine bunkers do not run empty or overflow (this helps ensure that one global optimal solution is found by preventing two local optima from occurring). The stacking actions are then created based on the following rules:

- Bunker level to start new heap: $\rho_2 \in \{0,1,2,3,4\}$
- Yard stacking priority (which yard is first stacked): $\rho_3 \in \{0,1\}$

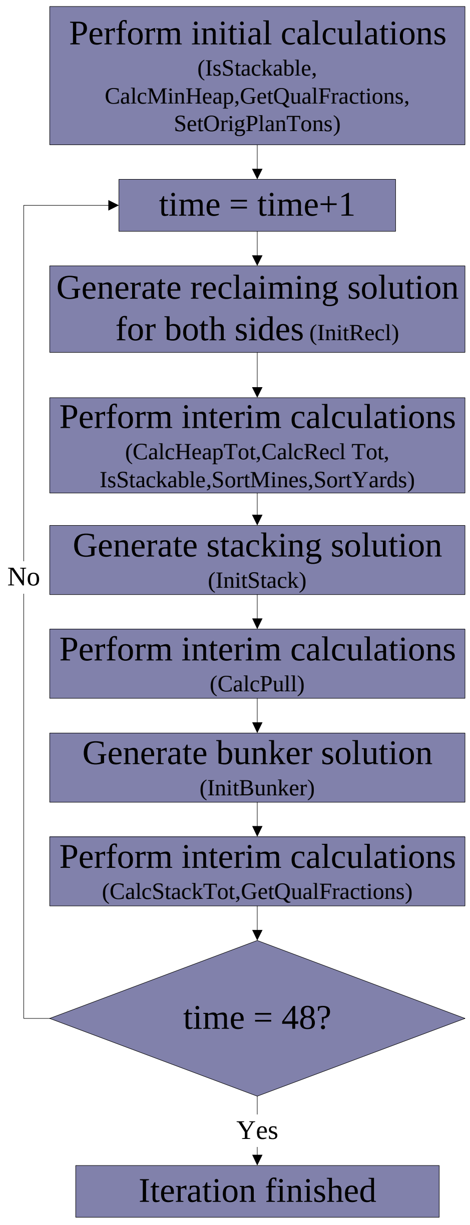

Perform initial calculations
(IsStackable, CalcMinHeap,GetQualFractions, SetOrigPlanTons)
time = time+1
Generate reclaiming solution for both sides (InitRecl)
Perform interim calculations
(CalcHeapTot,CalcRecl Tot, IsStackable,SortMines,SortYards)
Generate stacking solution
(InitStack)
Perform interim calculations
(CalcPull)
Generate bunker solution
(InitBunker)
Perform interim calculations
(CalcStackTot,GetQualFractions)
time = 48?
No
Yes
Iteration finished

Figure 4.14: Initial solution logic

- Mine stacking priority (which mine is first stacked): $\rho_4 \in \{0, 1\}$
- Bunker level for over-blending: $\rho_6 \in \{0, 1, 2, 3, 4\}$
- New heap length factor: $\rho_7 \in \{0, 1, 2, 3, 4\}$

Some calculations are then performed to adapt the required variables to the stacking actions decided upon.

The mine bunker solution determines how the mine bunkers and the CHF yard bunker must be operated to ensure that the planned stacking actions can be executed. Additional calculations are performed to adapt the required variables to the bunker actions decided upon.

This process is repeated until all 48 time buckets have been completed (i.e. all 48 hours have been scheduled). Note that although hourly time buckets are used, the SA algorithm has been written in such a manner that more than one material movement can occur within one hour. This ensures that the scheduling model can provide continuous time point solutions.

The detail of the processes used by the SA metaheuristic to setup the reclaiming, stacking and bunker solutions, for each hour, is now discussed in more detail.

4.5.1 Reclaiming

The approach used to generate reclaiming solutions for an hourly time bucket is shown in Figure 4.15 and summarizes the *InitRecl* sub-procedure.

First of all, the rates at which each factory is expected to extract coal from their respective Coal Processing (CP) bunkers are calculated. These hourly rates represent the tons that Sasol Coal Handling Facility (CHF) must reclaim, per side, to ensure that the CP bunker is at the same level at the end of the hour as it was at the beginning of the hour.

If a CP bunker is not 100% full, the rate at which CHF must reclaim on that CP bunker's side is increased slightly to ensure that the bunker is filled up to 100% within the next one to two hours.

Planned bypass tons (coal loaded onto factory belts from the strategic dead stockpiles — the recourse required for when factory demand cannot be met using normal reclaiming) are deducted from this increased rate, as the bypass tons sent

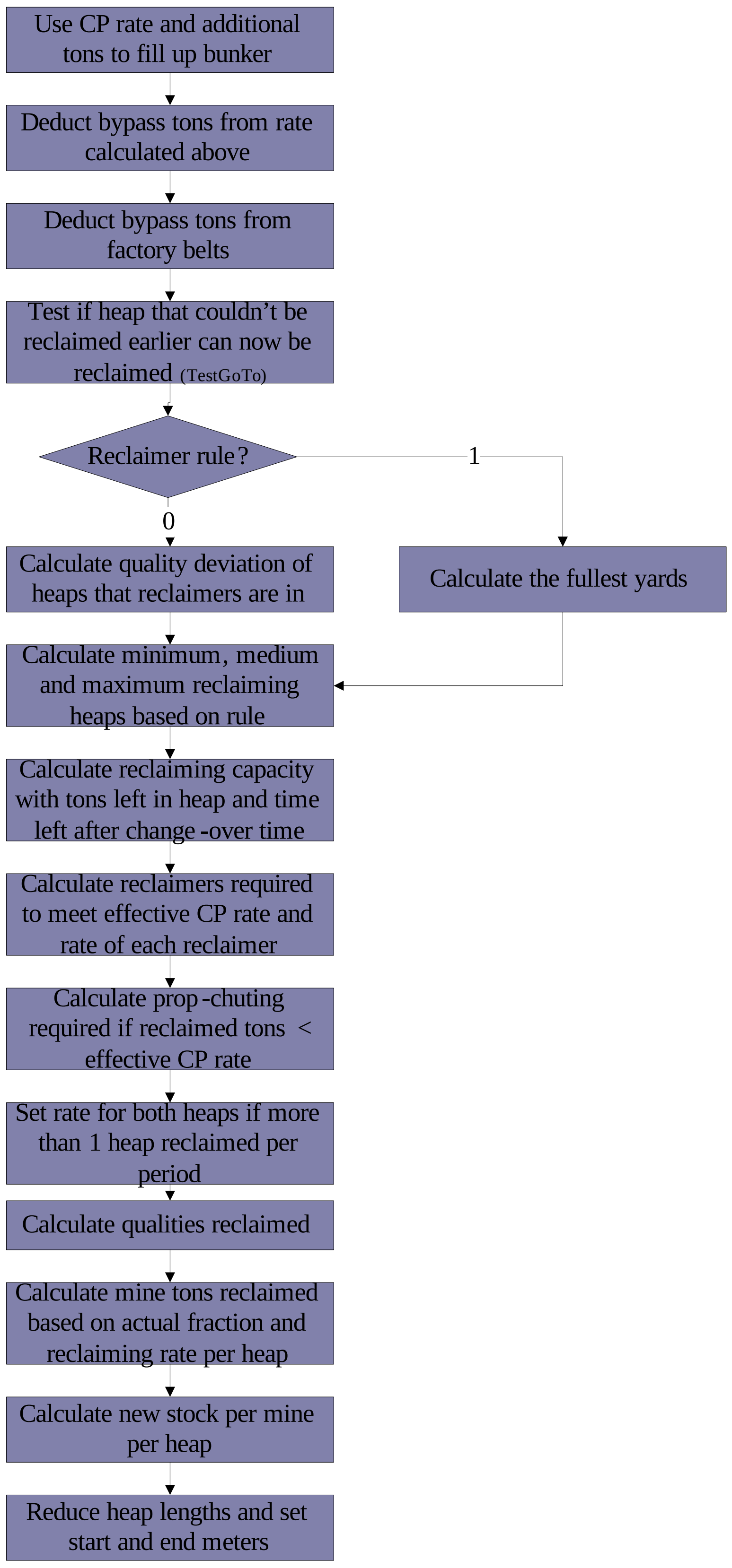

Use CP rate and additional tons to fill up bunker
Deduct bypass tons from rate calculated above
Deduct bypass tons from factory belts
Test if heap that couldn't be reclaimed earlier can now be reclaimed (TestGoTo)
Reclaimer rule?
1
0
Calculate quality deviation of heaps that reclaimers are in
Calculate the fullest yards
Calculate minimum, medium and maximum reclaiming heaps based on rule
Calculate reclaiming capacity with tons left in heap and time left after change-over time
Calculate reclaimers required to meet effective CP rate and rate of each reclaimer
Calculate prop-chuting required if reclaimed tons < effective CP rate
Set rate for both heaps if more than 1 heap reclaimed per period
Calculate qualities reclaimed
Calculate mine tons reclaimed based on actual fraction and reclaiming rate per heap
Calculate new stock per mine per heap
Reduce heap lengths and set start and end meters

Figure 4.15: Reclaiming logic

to factory means that CHF has to fill up the remaining tons demanded. This final rate is referred to as the effective required reclaiming rate.

The factory belt capacities are also reduced by the amount of bypass planned. This is done to ensure that the CHF reclaimed tons, together with the planned bypass tons, do not overload the conveyor belts going to the respective factories.

If a reclaimer is waiting for a heap to be finished stacking so that it can reclaim that heap, the algorithm tests whether such heaps have been filled up to a reclaimable level in the previous hour. If such a heap has been fully stacked, the model is allowed to start reclaiming that heap, if required. Otherwise, the model may not reclaim such a heap since it has to wait for the heap to be filled up first.

The priority of a yard's reclaimer is calculated based on the reclaiming rule (ρ_1) applicable to this specific iteration's solution. The different rules are applied as follows:

Reclaim based on qualities With $\rho_1 = 0$, the heap with the best quality is given highest priority and the lowest quality heap is given the lowest priority (this is done for both sides). This rule is the operational philosophy that should be applied most of the time at CHF.

Reclaim based on fullest yards With $\rho_1 = 1$, the fullest yard is given highest priority and the least full yard is given the lowest priority. This is done for both the side 1 and side 2 three reclaimers. The $\rho_1 = 1$ rule will typically work well when the stockpiles are quite full, as the fullest yards must then be reclaimed first to create stacking space for new heaps. That will prevent any of the mines being stopped (when there is no space on the stockpiles, CHF cannot pull coal produced by the mines and this will cause mine bunkers to fill up – which means the mines will be stopped if the bunkers get full). Stopping a mine has a major impact on Sasol Mining's profitability, since the mining environment consists of mostly fixed expenses and depends on the variable profit (based on produced tons) to recover these costs.

For each side, the reclaiming heap on the highest priority yard is reclaimed together with the reclaiming heap on the lowest priority yard. The first heap is reclaimed at a higher rate than the second heap. With $\rho_1 = 0$ this ensures that coal from the good quality heap offsets coal from the poor quality heap. It also ensures that heaps with poor quality do get reclaimed (i.e. prevents them from being left unreclaimed forever).

Each reclaimer's effective capacity is then calculated. This effective capacity is based on the following:

Machine rate The actual maximum reclaiming rate (capacity) of each reclaimer (i.e. 1,800 ton/hour).

Tons remaining on current heap If less tons are left on the heap than the reclaimer capacity, the effective capacity is the tons left on the heap instead of the maximum machine rate.

Change-over time If the current heap is finished in this hour and a second heap can be reclaimed, the change-over time of the reclaimer between the two heaps is deducted from effective reclaiming rate (i.e. less than one hour is available for reclaiming). If two heaps lie against each other, no change-over time is applicable.

Based on the effective capacity of each reclaimer, the number of reclaimers required per side is determined. As the factory typically uses around 2,500 ton/hour and the reclaiming rate is 1,800 ton/hour per reclaimer, the situation where only one reclaimer need to be run at a time will never occur. When two reclaimers are required on a side, the reclaimer with the highest priority is reclaimed at maximum reclaiming rate ($\rho_5 = 0$). The remaining required reclaiming tons are then taken from the lowest priority yard. If the first two reclaimers cannot reclaim enough coal to meet the effective factory demand, the third reclaimer (with the in-between priority) is also utilized.

Note that different reclaimer split rules (ρ_5) change the reclaiming rate split between the first two reclaimers. The rate at which the highest priority heap's reclaimer is run is set out below. The rest of the required tons are then reclaimed from the lowest priority heap's reclaimer.

- With $\rho_5 = 0$, the rate of the first reclaimer is 100%
- With $\rho_5 = 1$, the rate of the first reclaimer is 75%
- With $\rho_5 = 2$, the rate of the first reclaimer is 60%

- With $\rho_5 = 3$, the rate of the first reclaimer is 50%

If all three reclaimers cannot supply the required rate (mostly due to maintenance or the fact that, on some yards, no heaps are reclaimable), the additional required tons are supplied by prop-chuting (i.e. a mine's trajectory tons are fed directly onto the factory belts, in other words, the stockpiling process is bypassed).

Throughout the above mentioned processes, the metaheuristic keeps track of whether the reclaimer on a specific yard finishes reclaiming that heap in this period and moves on to start reclaiming the next reclaimable heap on that yard.

Once all the reclaiming movements have been determined, the quality of the reclaimed coal is calculated. The tons of each mine's coal reclaimed from the heaps are also calculated. The reduced number of tons left on the heaps is calculated per mine. The reduced heap lengths can then be determined.

4.5.2 Stacking

The stacking movements for each hour are created using the process outlined in Figure 4.16, which also summarizes the *InitStack* sub-procedure.

First, the bunker level of each mine is evaluated. If this bunker level is above a certain percentage, as determined by ρ_6, coal from the mine is allowed to be stacked on the current stacking heaps up to 3% above the percentage set out by the blend plan. The bunker percentage associated with different values of ρ_6 is set out in Table 4.2. This over-blending is used at CHF to prevent throw-outs at mines and this approach has been agreed upon by the gasification business unit.

The following four subsections deal with the four different stacking processes, in the order that the SA applies these different processes. Each of these processes selects a heap on which the applicable mine should be stacked.

When a heap has been selected, the SA attempts to stack on that heap if the infrastructure, mine bunker level and blend allow it. This process of attempting to stack a source is explained after the four different stacking processes are dealt with.

Throughout this section, the reader should keep in mind that as much stacking as possible must be done at all times. This is to ensure that the mine bunkers do not fill up. When bunkers are full, any further mine production will be thrown

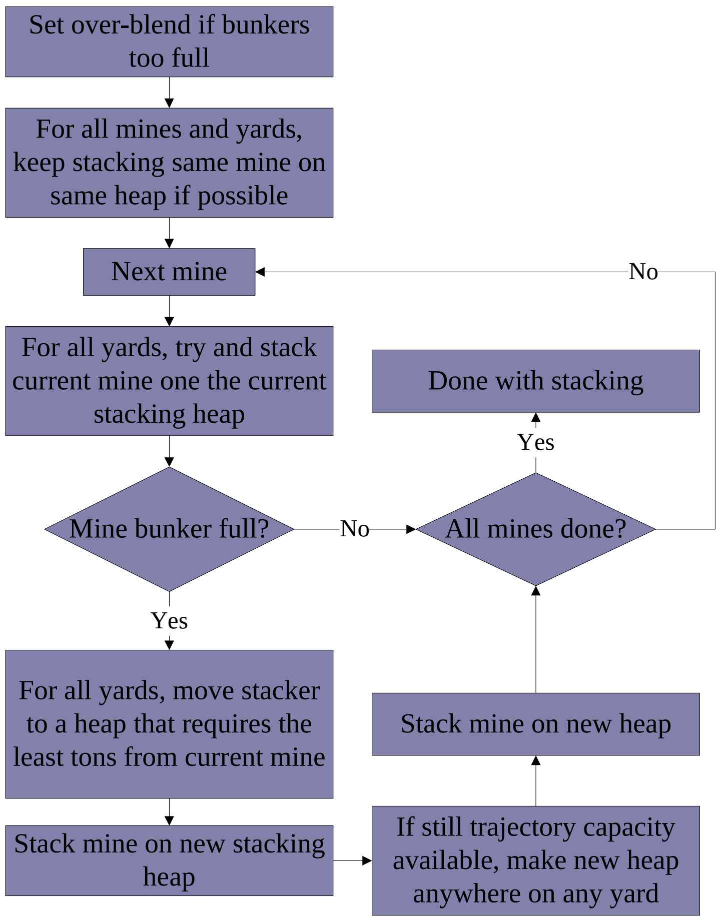

Set over-blend if bunkers too full
For all mines and yards, keep stacking same mine on same heap if possible
Next mine
No
For all yards, try and stack current mine one the current stacking heap
Done with stacking
Yes
Mine bunker full?
No
All mines done?
Yes
For all yards, move stacker to a heap that requires the least tons from current mine
Stack mine on new heap
Stack mine on new stacking heap
If still trajectory capacity available, make new heap anywhere on any yard

Figure 4.16: Stacking logic

out at the mine bunkers (which generates additional fines and costs). When no more space is available for throw-out at the mines, the mine is stopped causing enormous monetary losses for Sasol Mining.

Continue From Previous Period

The first attempt at stacking is done by continuing the stacking performed in the previous hour. Each of the mines keeps stacking on the same yard as in the previous period, starting with the mine with the highest priority. The mine priority is either determined by bunker fullness ($\rho_4 = 0$) or by the amount of space available for this mine on the heaps ($\rho_4 = 1$), given the blend plan and the previously stacked tons.

The model will attempt to stack coal from the selected mine on the same heap it was stacked in the previous period, *only* if coal from that mine was being stacked in the previous hour. This continuation of stacking ensures stability between hours and stability between scheduler re-runs (which ensures that plant instability is minimized).

Normal Stacking

During this stacking process, coal from all remaining mines are stacked in their order of priority. The model attempts to stack coal from the selected mine on the heap where the stacker is currently positioned, for every yard. Stacking is attempted on the yards with the highest priority first. This yard priority is either determined by the least reclaimable tons left on the yard ($\rho_3 = 0$) or by the open space left on the yard ($\rho_3 = 1$).

Move Stacker

After normal stacking has been completed and coal from the current mine (as selected in normal stacking) has still not been stacked up to the full hourly stacking or trajectory capacity, the model attempts stacking on different heaps (i.e. other heaps than the current stacking heaps – filling up other heaps that have not been fully stacked). This process starts on the yard with the highest priority and tries to stack on the heap on this yard that requires the least tons from that mine (as compared to other heaps on that yard).

Start New Heaps

When the currently selected mine's bunker is full and the mine's trajectory is not fully utilized for the current hour, new heaps are started to accommodate the selected mine. Once again, the model first attempts to start new heaps on the yard with the highest priority.

Stacking on the Selected Heap

This section describes the *StackPerMine* sub-procedure.

When a mine, yard and heap has been selected in the above four subsections, the model attempts to stack as much as possible from the selected mine on the selected heap by ensuring that the following rules are adhered to.

- The stacker capacity is adhered to. The stacking amount will be reduced until the stacker capacity is adhered to.
- Trajectory capacity is enforced. If the attempted stacking amount is more than the trajectory capacity, the stacking amount is reduced to an amount that the trajectory can handle.
- The side 1 to 2 and the side 2 to 1 transfer belt capacities are adhered to. Only one of the mines can be taken across the transfer belts at a time, i.e. only Source 1 or Source 2 can be taken to side 2 in an hour and only one of the side 2 mines can be taken to side 1 in an hour.
- Stacker movements between different heap positions cause stacker change-over times, which effectively reduce stacker capacities.
- Changing the current stacking mine to a different mine on the same heap causes mine source change-over times, which also effectively reduces the stacker's capacity.

- The amount of tons than can still be stacked from the selected mine on the selected heap, as determined by the blend plan, may not be exceeded. However, 3% over-blending may occur if required, as discussed previously.

When all the above rules can be adhered to, a stacking movement is finalized and the inventory levels and other variables are adjusted accordingly.

Throughout the stacking process, the Microsoft® Visual Basic for Applications (VBA) Simulated Annealing solution generation code (from hereon simply referred to as "code") provides the opportunity for Source 6-7 coal (i.e. coal from the yard bunker) to be stacked on side 1, side 2 or on both. This is possible because the yard bunker can be pulled onto two different conveyor belts at the same time – one to side 1 and one to side 2, as illustrated in Figure 4.17. The code also allows Source 5 to be stacked with Source 6-7 on either side 1 or side 2. CHF does this to prevent the small amount of Source 5 tons produced to occupy an entire stacker on its own.

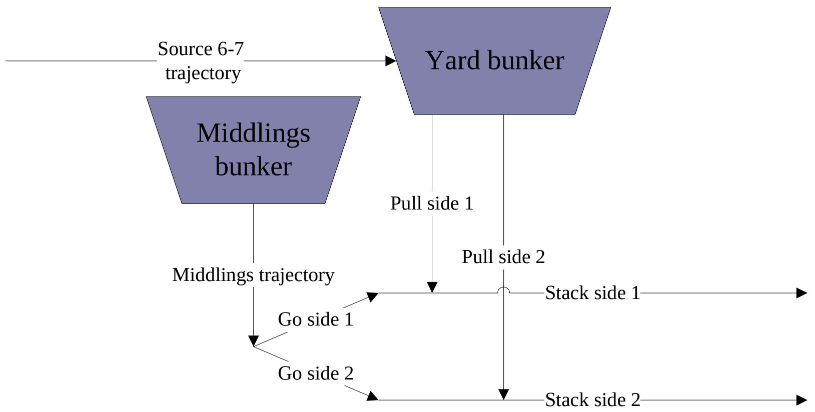


Figure 4.17: Source 6-7 and Source 5 staking possibilities

The Source 6, 7, 5 and yard bunker rules stated in the previous paragraph simplify CHF operations, but were extremely difficult to include in the SA. The impact that these Source 6 rules have on bunker operations is addressed later in Section 4.5.3.

The previous five subsections explain how the model determines the stacking actions. However, the model needs to consider the impact that the reclaiming and stacking solutions have on each other. Heap logic rules govern the impact that these two elements have on each other. These rules are now discussed.

Stacker-Reclaimer Heap Logic

One of the difficulties with the stacking and reclaiming processes is the interaction between the two. This is referred to as heap logic. CHF must operate the stockpiles so that they do not block the reclaimer between the end of a yard and a half stacked heap or between two half stacked heaps (as this means that no tons can be reclaimed, even though the factory must always be supplied with a continuous feed of reclaimed coal).

It is ensured that the stacking heaps will never block the reclaimer by limiting the heap length of newly started heaps. The code ensures that the new heap lengths are short enough so that the heap can be fully stacked before the reclaimer on that yard has finished reclaiming other heaps and needs to start reclaiming the new heap.

When a new heap is started, the reclaimable tons left on the yard is calculated. Thereafter the last reclaimable heap is determined. It is then determined how long it will take to reclaim all the reclaimable tons and how long the change-over time from the last reclaimed heap to the current heap will be. This time is the maximum time allowed for the heap to be fully stacked. When this time is short, the model can only start short new heaps to ensure that they can be fully stacked before they need to be reclaimed.

The maximum allowable heap length of the new heap can be calculated using equation (4.11), with R the reclaimable tons on the applicable yard, R_{cap} the reclaiming capacity for that reclaimer in that specific time period in ton/hour, R_{co} the reclaimer change-over time from the last reclaimed heap in hours, S_{cap} the stacking capacity in ton/hour and $S_{\text{t/m}}$ the stacking ton/meter amount.

$$\text{Maximum heap length} = \left(\frac{R}{R_{\text{cap}}} + R_{\text{co}}\right)\left(\frac{S_{\text{cap}}}{S_{\text{t/m}}}\right) \tag{4.11}$$

Equation (4.11) is easier understood by means of an example with t for ton, h for hour and m for meter. Say there are 20,000 tons reclaimable coal on a yard (i.e. $R = 20,000t$) and that the reclaiming capacity of that yard's reclaimer is 1,800 ton/hour (i.e. $R_{\text{cap}} = 1,800t/h$). If it would take the reclaimer 30 minutes to move from its current position to the heap that needs to be reclaimed, the R_{co} would equal $0.5h$. The typical stacking capacity is 1,800 ton/hour ($S_{\text{cap}} = 1,800t/h$)

and the typical stacking ton/meter is 200 ($S_{\mathrm{t/m}} = 200t/m$). The maximum heap length of the new heap can be calculated as follows.

$$\begin{aligned}\text{Maximum heap length} &= \left(\tfrac{20{,}000t}{1{,}800t/h} + 0.50h\right)\left(\tfrac{1{,}800t/h}{200t/m}\right)\\ &= \big(11.11h + 0.50h\big)\big(9m/h\big)\\ &= 11.61h\big(9m/h\big)\\ &= 104.49m\end{aligned}$$

The reclaiming time calculation assumes that the reclaimer runs at full capacity. However, this is not always the case due to lower reclaimer rates, no tons being reclaimed off that yard, maintenance and/or change-over times. The stacking time is set equal to the reclaiming time. This stacking time is used together with the stacking rate to determine how long the new heap can be. The stacking rate is also assumed to be equal to stacker capacity. However, this is also not practical due to maintenance, stacker change-over times, mine source change-over times and stacker idle times. To compensate for these two factors, the maximum allowed heap length is reduced by a certain factor. This factor is ρ_7 and varies between 60% and 90%, as set out in Table 4.2.

In the previous example, with $\rho_7 = 1$, the factor is 80%. The maximum length of the new heap is $104.49 \times 0.8 = 83.59m \approx 84m$.

If the calculated heap length is longer than the minimum allowed heap length (an input value, typically in the region of $100m$), a new heap has successfully been created. Otherwise, the new heap cannot be started and another position for starting a new heap has to be found.

The heap logic rules not only ensure feasibility, but also help to obtain reclaiming and stacking solutions that combine to form a good overall solution. Combining two locally optimal solutions without heap logic rules produces a poor solution and in the case of reclaiming and stacking solutions, the combination will not even be feasible (i.e. not practically executable).

The candidate would like to thank Marthi Harmse, from Sasol Technology — Operations and Profitability Improvement (OPI) department's Operations Research group, for suggesting this approach. After various people's attempts to properly address heap logic failed, her analytic thought process which identified

the root cause of heap logic problems (i.e. that all heap logic problems can be a avoided by governing new heap lengths) proved invaluable.

4.5.3 Bunkers

This section explains the *InitBunker* sub-procedure code. The process applied to generate the bunker part of a solution is summarized in Figure 4.18.

The most difficult part of generating the bunker solutions is to schedule the Source 6, Source 7, circular stockpile, circular stockpile bypass belt and yard bunker operations. Figure 4.19 provides the layout of this part of CHF, to enable the reader to better understand the reasoning behind this part of the bunker solution process.

A fraction of the Source 6-7 trajectory must be assigned to each source. If one of the sources has no production for the current hour, the other source is assigned the full belt capacity of 2,800 ton/hour. When both sources are producing in the hour in question, the fraction assigned to the belt for each source is equal to the fraction of its production fraction of total Source 6 and Source 7 production.

Only if there is enough space left in the yard bunker to pull coal from Source 7 and Source 6, these sources are pulled onto the Source 6-7 trajectory in the fraction calculated above.

The circular stockpile bypass is assigned a fixed percentage (based on historical data) of the Source 6-7 trajectory. This percentage is subsequently deducted from the Source 6 fraction to calculate the tons required from the circular stockpile (which effectively operates like a normal mine bunker).

All the above mentioned actions consider whether the bunkers have at least as much coal left as the model wants to pull from it in this hour. If not, the amount pulled is reduced to the tons left in the bunker. If mine bunkers would overflow, the tons which the bunker cannot accommodate are thrown outside the bunker.

Finally, all bunker levels at the end of this period (i.e. at the start of the next period) are determined using material balance equations.

The steps required to run the SA scheduler are given in Appendix B.

As the SA and solution generation logic have been discussed, the customization and implementation of the metaheuristic will be addressed in the next the chapter.

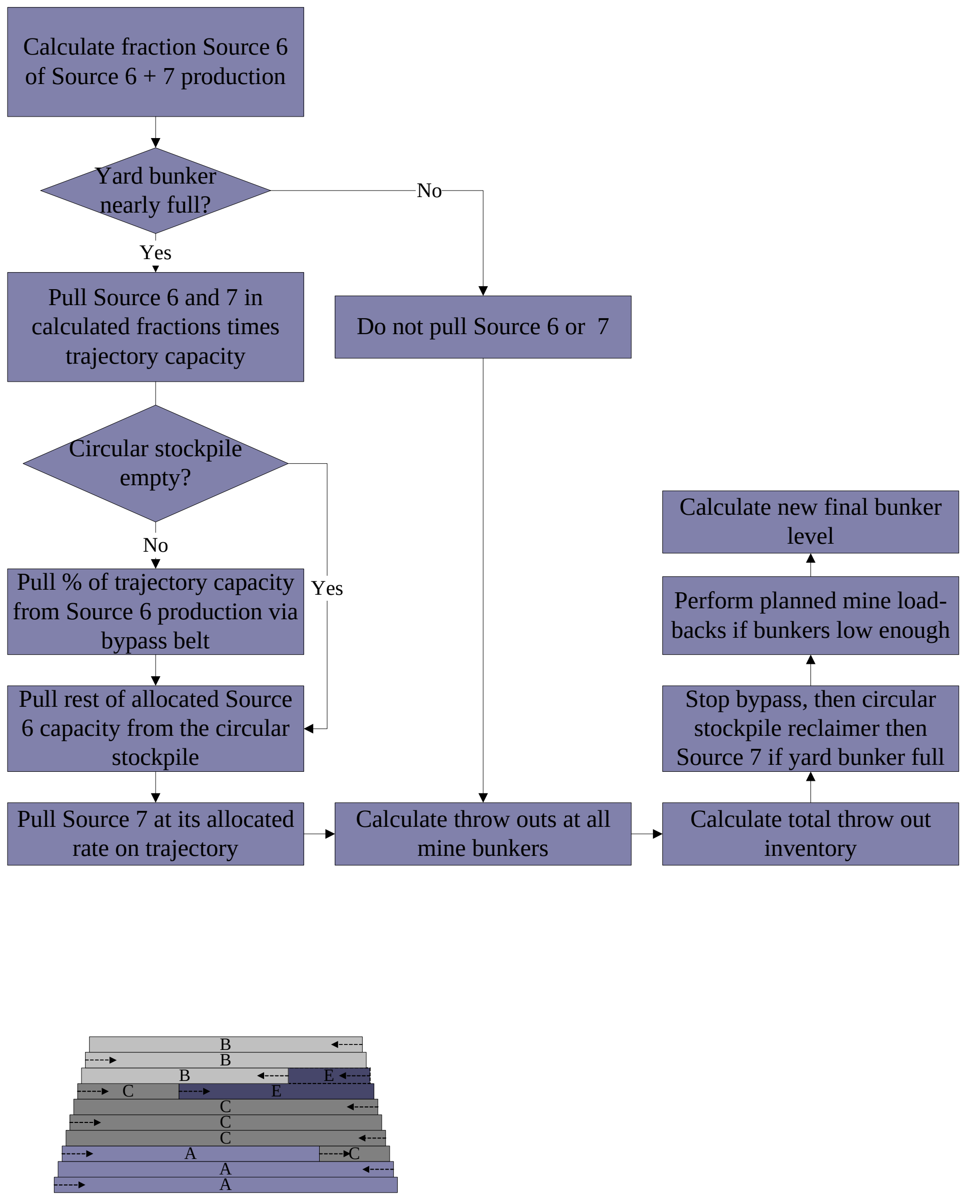

Calculate fraction Source 6 of Source 6 + 7 production
Yard bunker nearly full?
No
Yes
Pull Source 6 and 7 in calculated fractions times trajectory capacity
Do not pull Source 6 or 7
Circular stockpile empty?
No
Yes
Pull % of trajectory capacity from Source 6 production via bypass belt
Pull rest of allocated Source 6 capacity from the circular stockpile
Pull Source 7 at its allocated rate on trajectory
Calculate throw outs at all mine bunkers
Calculate total throw out inventory
Stop bypass, then circular stockpile reclaimer then Source 7 if yard bunker full
Perform planned mine load-backs if bunkers low enough
Calculate new final bunker level
B
B
B
E
C
E
C
C
C
A
C
A
A

Figure 4.18: Bunker logic

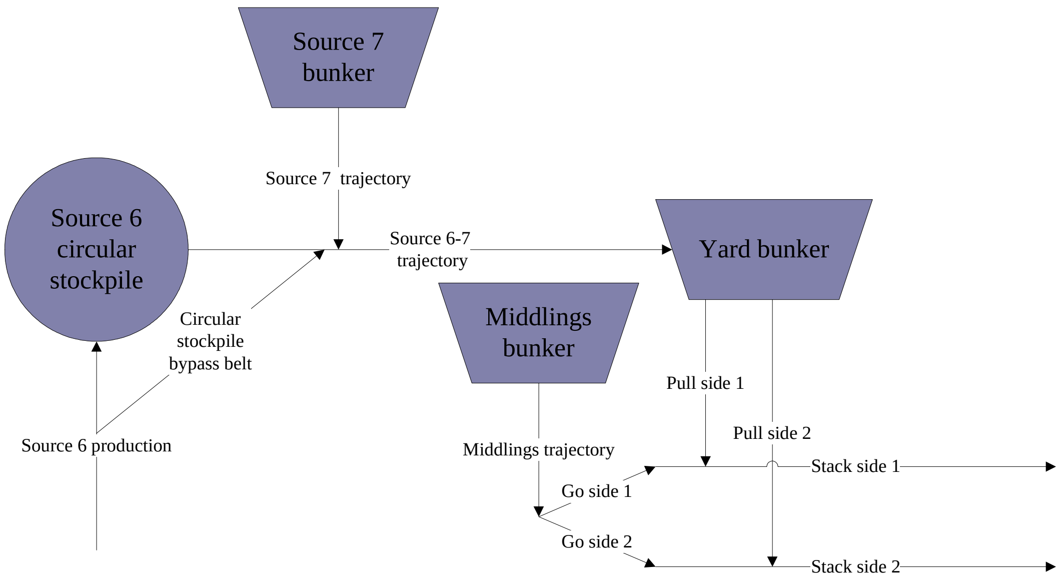


Figure 4.19: Source 6 and Source 7 operations

Chapter 5

Algorithm Customization and Implementation

To customize the SA to this specific application, the SA input variables can be refined to enhance algorithm performance. Once the algorithm has been customized to its application, it must be integrated with the company's business processes to ensure successful algorithm implementation. Once the SA has been integrated into the business, the performance of the developed approach can be evaluated by comparing the SA output to previously applied methodologies.

5.1 Input Variable Refinement

To enable successful implementation of metaheuristics, it is required to refine the metaheuristic parameters input values and to integrate the metaheuristic in the company's business processes. These two implementation points are discussed below.

The SA related input parameters have a major impact on how the SA will perform. It is necessary to find a combination of these SA input parameter values that will enable the SA to provide good solutions to any real-life CHF situation that need to be scheduled (i.e. *any* possible problem instance).

The SA parameters for which widely applicable values must be found are:

- Number of iterations (N)

- Initial temperature ($\tau(1)$)
- When sudden temperature reduction occurs (w)
- Speed of sudden temperature reduction (k)

To find the values of these parameters that will enable the SA to provide good answers for any problem instance, various trial runs for different problem instances are done. Fixed values that perform well for all problem instances are required as the CHF scheduler runs in the background and only displays the resulting schedule on the control room screens (i.e. no user intervention takes place).

5.1.1 Comparative Fractions Used for Evaluation

To find parameter values that will work well for all problem instances, one has to compare different instances to each other. However, the objective functions of different instances typically vary between 1,500 and 100,000. Therefore it would not make sense to simply compare the objective functions of different instances to each other. To overcome this problem, the objective function values of different instances are compared to each other using comparative fractions (ϖ). As various runs with different values of the input parameters are done, the minimum objective function for a specific problem instance (with any input parameter values) is used as the denominator in the fraction. The numerator of the fraction is the actual objective function value for the current run with its specific input parameter values.

For example, consider two different problem instances. Various runs of each instance have been performed with different combinations of input parameter values. Table 5.1 shows the objective function values for different runs (shown for 150 iterations).

The bold numbers in Table 5.1 represent the best objective function found for the specific instance during the various runs. The comparative fractions for the first line in Table 5.1 are calculated as follows. For the first instance, the fraction is $\varpi = \frac{5{,}783}{5{,}312} = 1.089$, and for the second instance $\varpi = \frac{16{,}876}{16{,}152} = 1.045$. The comparative fractions of very different problem instances can now be compared to each other.

Table 5.1: Example of calculating the comparative fraction (ϖ)

$\tau(1)$	k	w	**Instance 1**	**Instance 2**
50,000	2	1.5	5,783	16,876
50,000	2	1.5	5,689	17,293
⋮	⋮	⋮	⋮	⋮
100,000	8	4	**5,312**	17,012
100,000	8	4	5,452	**16,152**

The ϖ value therefore indicates how close the SA can get to the best ever solution for the specific problem instance. The ϖ value should be as low as possible, since a ϖ value of 1 represents the best ever solution for a problem instance.

5.1.2 Number of Iterations (N)

Various runs for each different combination of input parameter values for different problem instances are required to determine which values will enable the SA to perform well for all problem instances. The following problem scenarios were used for trial runs to enable input parameter value refinement:

1. A typical CHF situation with objective function values around 4,000.
2. A more difficult typical CHF situation with objective function values around 10,000.
3. A very good CHF scenario with objective function values in the region of 2,000.
4. A situation where a lot of mine throw-outs will occur with objective function values around 12,000.

The problem scenarios represent extreme and normal CHF situations and should be a good representation of all CHF problem scenarios.

Figure 5.1 shows the average objective function value over 5 runs for each of the 270 combinations of parameter values considered. The following ranges of

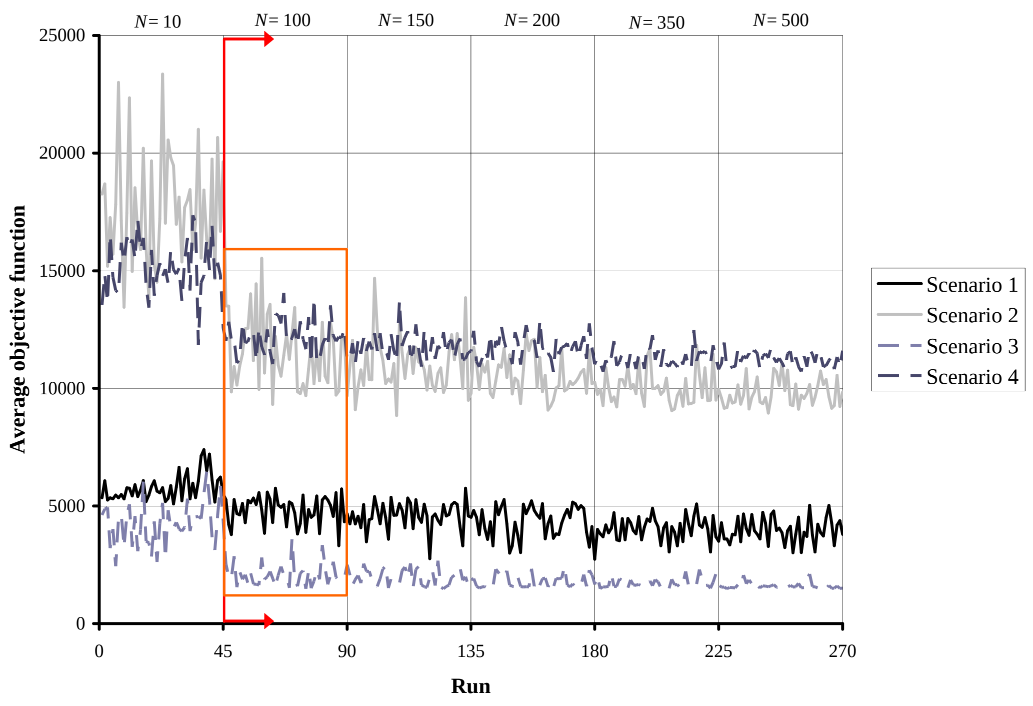


Figure 5.1: Average objective function values for SA parameter combinations

parameter values were used for this specific figure (with z_1 the objective function value of the initial solution):

- $N = \{10; 100; 150; 200; 350; 500\}$
- $\tau(1) = \{10,000; 100,000; 1z_1; 10z_1; 20z_1\}$
- $w = \{0.5; 2; 8\}$
- $k = \{1; 10; 20\}$

To be able to compare different runs to each other, Figure 5.1 is changed to Figure 5.2 to using average comparative fractions (ϖ) instead of average objective function values.

In Figure 5.2, runs 1 through 45 represent runs done with $N = 10$ for different combinations of $\tau(1)$, w and k. The lower comparative fraction values that occur from runs 46 onwards, represent runs done with $N = \{100; 150; 200; 350; 500\}$, i.e. $N \geq 100$. One should therefore use more than 100 iterations, but should consider the play-off between solution quality and solution time.

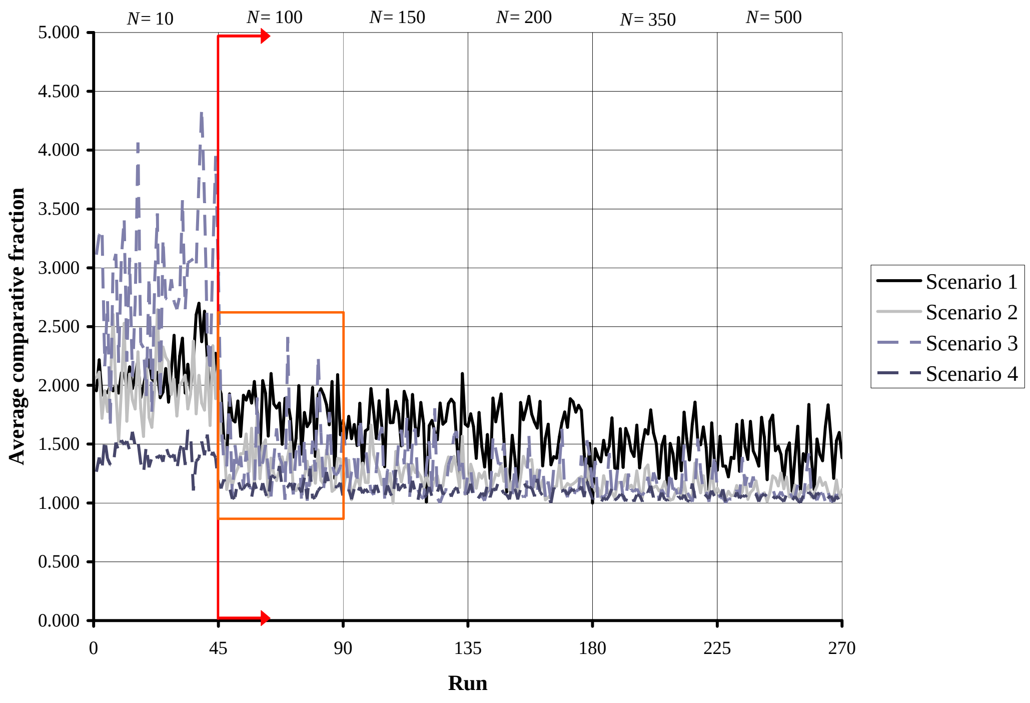


Figure 5.2: Average ϖ for SA parameter combinations

Although 100 iterations will cause faster execution times, runs with 100 iterations vary too much from each other (see runs 46 through 90, in the square, in Figure 5.2 and refer to the standard deviation of the comparative fraction for each scenario in Table 5.2). Using 100 iterations prevents the SA from visiting enough neighbourhood solutions to provide consistently good solutions. With 200 iterations, the SA provides answers that are only slightly better than with 150 iterations. Iterations limits larger than 200 are not considered as these require solution generation times greater than the time allowed in the CHF coal handling industry. Table 5.2 compares the standard deviations (σ) of the comparative fractions for various runs with different iteration limits.

Table 5.2 shows the poor performance of the SA with 100 iterations since too much variance for runs of the same problem scenario with different w and k values indicates that the SA will not necessarily always find good solutions in a single run in practice. The slight difference in performance between 150 and 200 iterations can also be observed. One therefore accepts 150 iterations as this would lead to shorter execution times than would be observed with 200 iterations.

Table 5.2: Iteration limit determination

N	**Standard deviation of the comparative fraction (σ)**	**Solution time (in seconds)**
	Scenario 1	
100	0.221	23.9
150	0.214	36.4
200	0.200	45.5
	Scenario 2	
100	0.170	24.0
150	0.130	36.8
200	0.092	47.3
	Scenario 3	
100	0.306	20.7
150	0.227	29.3
200	0.177	41.0
	Scenario 4	
100	0.068	21.4
150	0.050	34.4
200	0.049	42.0
	Scenarios combined	
100	0.127	22.5
150	0.101	36.7
200	0.101	42.7

5.1.3 Time of Temperature Reduction (w)

Having fixed the iteration limit at 150, the other SA input parameters can be investigated. To visually identify patterns in the graph, the average comparative fractions for runs with the same input parameter values are plot against the run numbers. With $N = 150$, various runs for the following sets of input parameters are done.

- $\tau(1) = \{10,000; 50,000; 100,000; 1z_1; 2z_1; 8z_1; 10z_1; 20z_1\}$
- $w = \{0.5; 2; 4; 8; 10; 14\}$
- $k = \{1; 4; 8; 10; 15; 20\}$

The results of these runs provide a busy graph as shown in Figure 5.3. With the above mentioned input parameter values, the initial temperature changes every 36 runs. No trend of pattern could be identified in Figure 5.3.

To simplify the graph, the average comparative fractions for the different scenarios are shown in Figure 5.4, with the average comparative fraction for each step in $\tau(1)$. Once again, no pattern or trend can be observed.

The data used to draw Figure 5.4 is now sorted according to the k input parameter, i.e. according to the speed at which the change from diversification to intensification occurs, and is illustrated in Figure 5.5. This is done to determine whether any patterns or trends associated with k exist. However, no trends or consistently repeating patterns can be observed.

Sorting the data according to the time at which the change from diversification to intensification occurs w, yields Figure 5.6. In this figure, a definite decreasing trend can be observed for all the different scenarios, with all scenarios reaching their lowest values between $w = 8$ and $w = 10$.

From Figures 5.4 through 5.6 one can see that the only parameter that affects the quality of solutions independent of the other parameter values is w. A specific value for w can be determined by investigating the data in more detail. Table 5.3 shows the average values for all runs with different $\tau(1)$ and k values done for different values of w, with the minimum value for each scenario indicated in boldface. Additional runs for $w = 9$ is done as Figure 5.6 shows that the lowest comparative fractions can be found in the region of $w = 8$ to 10.

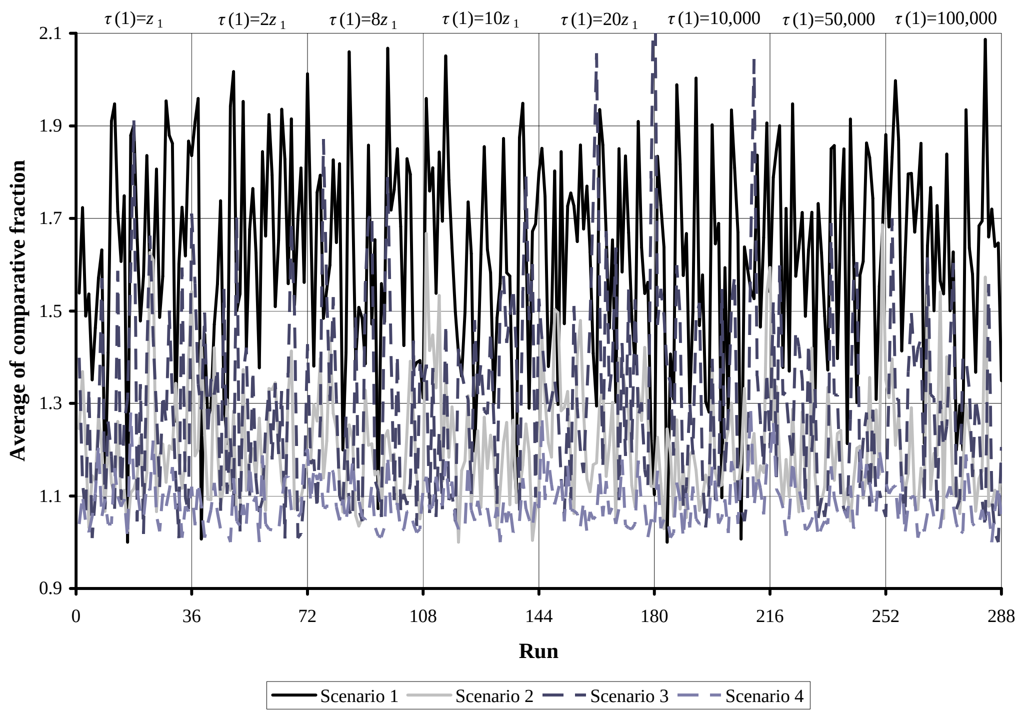


Figure 5.3: Detail of average ϖ for increasing $\tau(1)$ with varying w and k

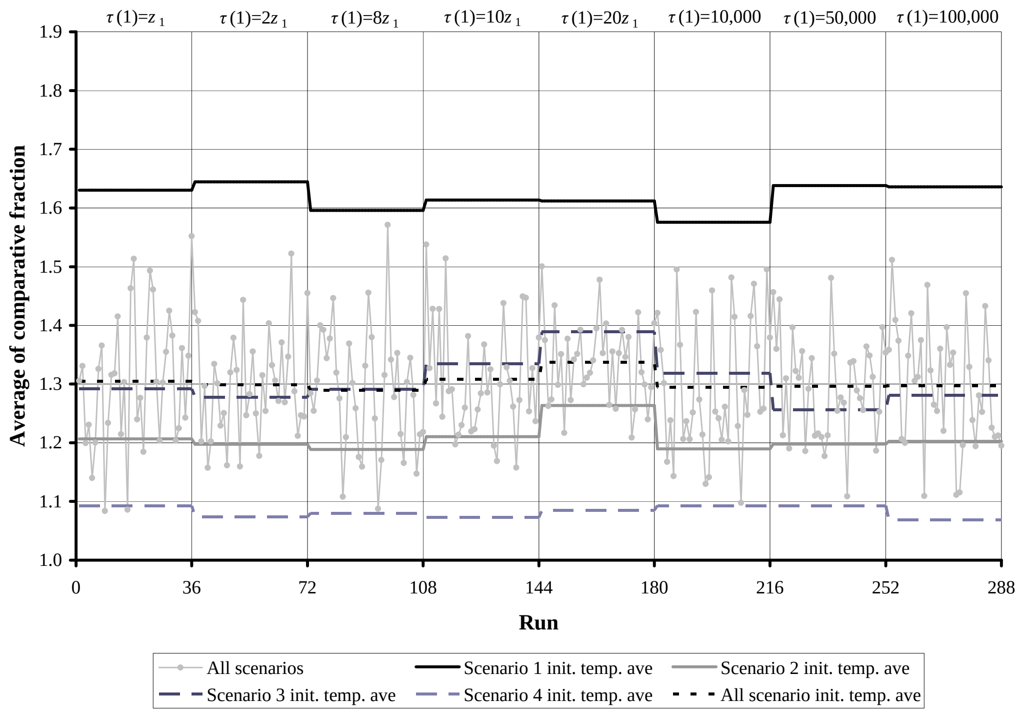


Figure 5.4: Average ϖ for increasing $\tau(1)$ with varying w and k

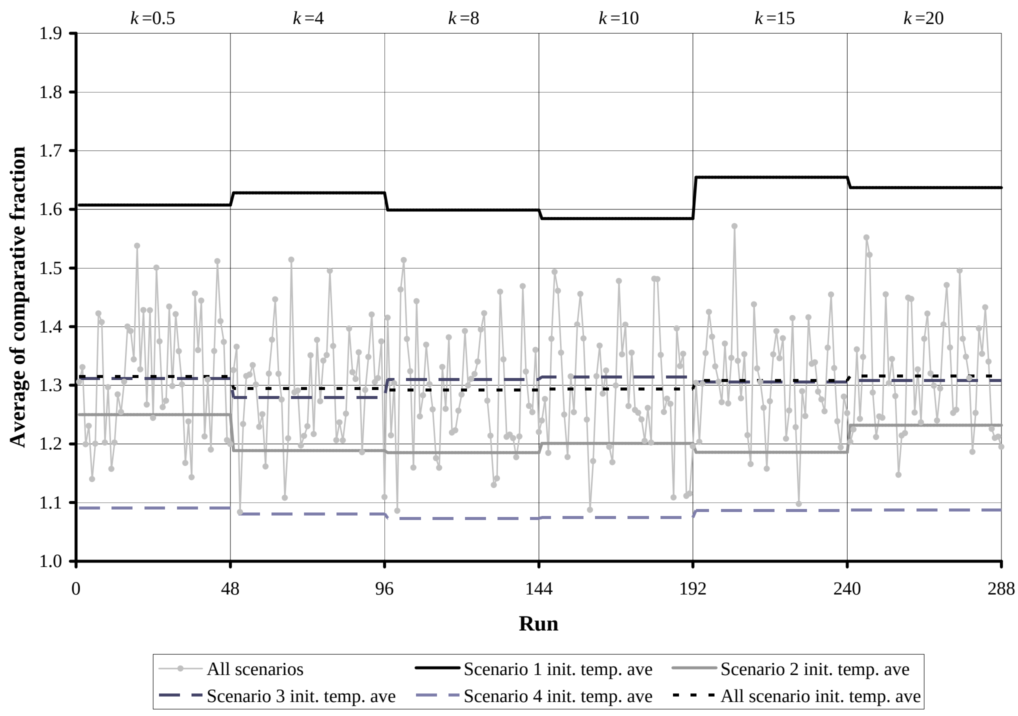


Figure 5.5: Average ϖ for increasing k with varying $\tau(1)$ and w

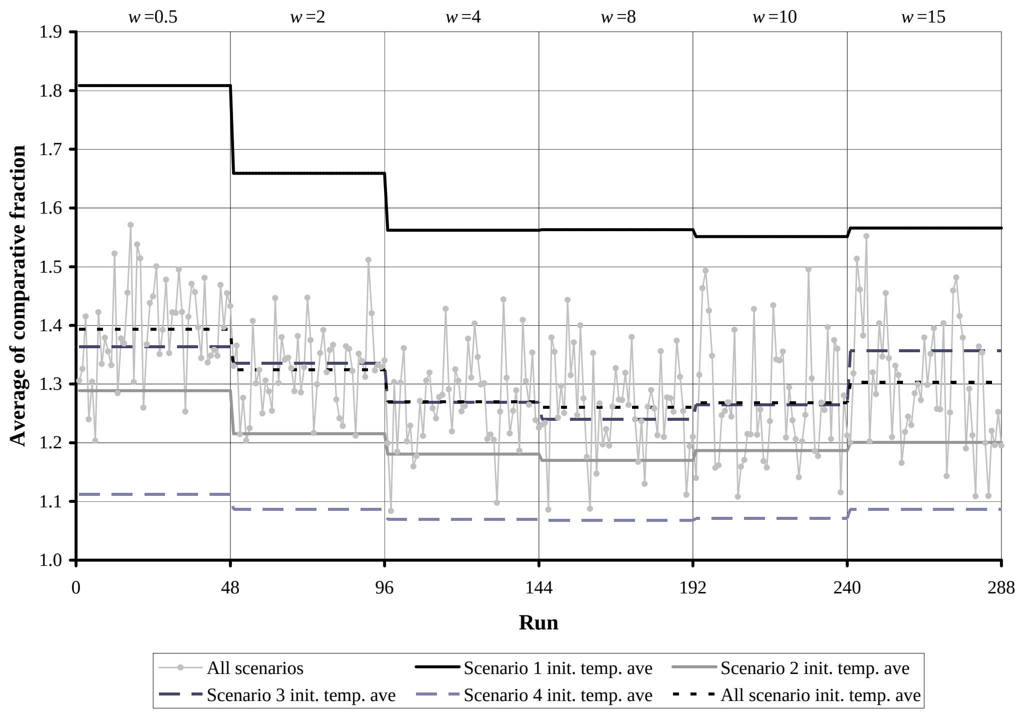


Figure 5.6: Average ϖ for increasing w with varying $\tau(1)$ and k

Table 5.3: Average ϖ for increasing w with varying $\tau(1)$ and k

w	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**All scenarios**
0.5	1.808	1.289	1.363	1.112	1.393
2	1.659	1.215	1.336	1.086	1.324
4	1.562	1.181	1.270	1.069	1.270
8	1.563	**1.170**	**1.240**	**1.068**	**1.260**
9	**1.557**	1.195	1.282	1.071	1.276
10	1.582	1.187	1.265	1.071	1.268
15	1.566	1.201	1.356	1.087	1.302

Table 5.3 shows that $w = 8$ will cause better performance than any other value of w for most scenarios. Although $w = 9$ performed the best for Scenario 1, $w = 8$ performed nearly as well as $w = 9$. Therefore $w = 8$ should be used in conjunction with 150 iterations for the SA scheduler.

5.1.4 Speed of Temperature Reduction (k)

The next two input parameters that need to be refined are $\tau(1)$ and k. Table 5.4 shows the performance of the algorithm with $N = 150$ and $w = 8$ for varying $\tau(1)$ (z_1 is the objective function value of the initial solution).

No specific $\tau(1)$ value outperforms all other $\tau(1)$ values for the different runs. Therefore, Table 5.4 is repeated for k in Table 5.5. Additional runs for $k = 9$ are included as many of the best values fall in the region between $k = 8$ and $k = 10$.

For most of the scenarios, $k = 9$ performs the best in conjunction with the previously determined input parameter values. For Scenario 2, $k = 8$ performs best and for Scenario 3, $k = 10$ performs best. With $k = 9$ being the best for most scenarios and $k = 9$ being a good average for the other scenarios, $k = 9$ should be used when running the algorithm.

Table 5.4: Average ϖ for increasing $\tau(1)$ with varying k

$\tau(1)$	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**All scenarios**
$1z_1$	1.532	**1.122**	1.387	1.051	1.252
$2z_1$	1.667	1.205	1.291	1.069	1.303
$8z_1$	1.544	1.142	**1.225**	1.065	1.229
$10z_1$	**1.386**	1.212	1.264	1.051	1.236
$20z_1$	1.521	1.200	1.318	1.062	1.268
10000	1.499	1.191	1.272	1.076	**1.222**
50000	1.521	1.196	1.310	1.098	1.260
100000	1.600	1.175	1.242	**1.034**	1.253

Table 5.5: Average ϖ for increasing k with varying $\tau(1)$

k	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**All scenarios**
1	1.505	1.222	1.307	1.077	1.278
4	1.615	1.158	1.242	1.053	1.267
8	1.493	**1.141**	1.237	1.050	1.230
9	**1.359**	1.162	1.278	**1.036**	**1.209**
10	1.502	1.177	**1.190**	1.076	1.236
15	1.651	1.185	1.320	1.084	1.310
20	1.613	1.216	1.447	1.067	1.241

5.1.5 Initial Temperature ($\tau(1)$)

The final parameter value that needs to be determined is the initial temperature $\tau(1)$. Table 5.6 shows the average comparative fraction values for various runs done with $N = 150$, $w = 8$ and $k = 9$ for different $\tau(1)$ values (z_1 is the objective function value of the initial solution).

Table 5.6: Average ϖ for increasing $\tau(1)$

$\tau(1)$	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**All scenarios**
$1z_1$	**1.000**	**1.024**	1.091	**1.010**	**1.031**
$2z_1$	1.953	1.373	1.081	1.081	1.372
$8z_1$	1.508	1.035	1.068	1.091	1.175
$10z_1$	1.197	1.168	1.480	1.046	1.223
$20z_1$	1.627	1.109	1.475	1.065	1.319
10000	1.304	1.145	**1.032**	1.039	1.130
50000	1.625	1.088	1.095	1.031	1.210
100000	1.728	1.187	1.066	1.035	1.254

From the low values in Table 5.6 one can see that the previously determined input parameter values generate solutions very close to the best solution ever found for each scenario. For most scenarios $\tau(1) = z_1$ provides the best performance.

To summarize the variable refinement process, Table 5.7 shows the input parameter values that will perform well for any problem instance. These values for the input parameters will produce answers that are on average only 1.1% above the best possible solution for normal problem scenarios (i.e. Scenarios 1, 2 and 4) and only 3.1% above the best possible solution for any given scenario.

5.2 Business Process Integration

The following sections shortly discuss how the scheduler is implemented in a practical environment.

Table 5.7: Selected SA input parameter values

SA input variable	Value
N	150
$\tau(1)$	z_1
k	9
w	8

5.2.1 Model Output Display

The model writes a log file of the detailed actions required by the incumbent solution. This log file shows reclaiming, stacking and bunker actions as well as numerous other outputs required to fully understand the incumbent solution. Please refer to Appendix C.1 for a detailed discussion of this output.

Various output graphs are employed to illustrate specific sections of output typically reviewed by users. These graphs are explained in Appendix C.2.

The best way for CHF to understand the model output is to see the details of how the yards change in a stockpile "picture", as used by CHF during normal operations. The "picture" is therefore adapted in such a way that the detail stockpile status for each hour can be displayed. Appendix C.3 provides more details on this popular output.

5.2.2 MES Integration

Manufacturing Execution System (MES) integration is done to be able to show the control room operators which tasks they must do and when they must do it. The approximate optimal solution produced by the SA is therefore fed into the MES and displayed in a user-friendly manner.

Comma Separated Value (CSV) files, containing the suggested schedule, are written to a server after model execution. These files are then read by the MES and displayed in the same program as used by the operators to control coal blending.

5.2.3 Tracking

When the operators start executing the suggested schedule, constant tracking is done to compare the actual plant status to the plant status expected by the scheduler. This is necessary to determine when it is required to re-run the scheduler if the variance in mine production, belt rates and factory consumption become so much that the current schedule become infeasible. Operator errors that cause large enough deviations will also require rescheduling.

The planned plant status and actuals are compared in the following areas:

- Mine bunker levels
- Factory bunker levels
- Tons on stockpile (for both stacking and reclaiming heaps)

Each 15 minutes the planned bunker and stockpile levels are compared with the plant actuals and it is then tested whether these differ by more than a predetermined threshold. If this is the case, the operator is prompted (and then forced by the MES if he/she does not react within an acceptable time span) to re-run the scheduling model with the current plant status as input variables. Note that the model's input values are automatically updated via the MES.

The obvious next step to improve this process is to incorporate real-time scheduling into the model. This will also reduce the differences and step changes that may occur between two consecutive runs.

The SA logic overview, solution generation process and SA implementation chapters explain how the SA scheduler for the CHF coal handling problem operates. Although proper SA logic design and solution generation are required, the importance of implementing the model and making the developed metaheuristic practically applicable should never be underestimated.

5.3 Evaluation of the Developed Approach

To ensure that research done in the Operations Research environment adds to the current body of knowledge, one has to evaluate the quality of the suggested approach (Manson, 2006). The developed Simulated Annealing algorithm is evaluated to show that it produces good quality solutions.

No CHF solution technique attempted in the previous 5 years has been able to produce solutions that match the quality of work done by control room operators. To evaluate the quality of the algorithm, the suggested output for a certain period is compared with the actual tasks executed by the control room operators without the use of the model. The following key performance areas are compared:

- Throw-outs at mine bunkers
- The number of over-blending that occurs
- The amount of prop-chuting required
- Recourse required (i.e. tons loaded back from strategic inventory sources)
- Stability of the supplied feed in terms of ash for both sides of the factory

The model output and actual events that took place are compared for two recent scenarios. Table 5.8 shows performance data for the first scenario.

Table 5.8: Evaluation of first scenario algorithm performance

Performance criteria	**Model**	**Actual realization without model**
Mine throw-out tons	1,470	1,213
$5 \times 48 \times 3\%$ over-blending	0	1,680
Prop-chuting tons	0	0
Total	1,470	2,893
Expected recourse (load-back tons)	1,265	0
Quality stability side 1 (% range)	0.75%	1.15%
Quality stability side 2 (% range)	0.55%	0.95%

From Table 5.8 one can see that the model throws out nearly the same amount as the control room operators did. However, the model manages not to over-blend, whilst the control operators did over-blend a lot. Less over-blending will result in a more stable feed being reclaimed to the factory. The total row indicates that the model's suggested scheduled outperformed the control room operators significantly.

The actual recourse experienced was 0, whilst the model had to include an expected value of 1,265. This is because the actual plant demand was less than the amount that would have required recourse. The model supplied a much stabler blend to both sides since the range between the highest and lowest value is lower than what was seen in practice. The stabler blend supplied by the model will ensure that the factory can produce at higher production rates than what can be achieved with the use of manual control room operator scheduling.

The second scenario's data is summarized in Table 5.9.

Table 5.9: Evaluation of second scenario algorithm performance

Performance criteria	**Model**	**Actual realization without model**
Mine throw-out tons	2,428	5,258
$5 \times 48 \times 3\%$ over-blending	1,920	4,080
Prop-chuting tons	1,679	0
Total	6,021	9,338
Expected recourse (load-back tons)	3,436	0
Quality stability side 1 (% range)	0.30%	0.50%
Quality stability side 2 (% range)	0.25%	0.40%

The second scenario shows that the model threw out and over-blended less than the operators — which will save costs and result in a stabler product being sent to the factory. The model does perform a small amount of prop-chuting, whilst in practice no prop-chuting was required. The model therefore sacrifices a bit of prop-chuting to reduce both throw-outs and over-blending by more than 52%. The model's total row is consequently much less than what was seen in reality.

As in the first scenario, the model once again provides a stabler blend (with a lower range in quality measurements) to the gasification venture on both side 1 and side 2.

From the scenarios analyzed one can see that the developed approach exceeds the performance currently seen in the control room. As the control room operators have an enormous amount of experience (typically more than 15 years) they do not perform too badly. It is therefore a good feat for the algorithm to meet and exceed their performance levels, especially as previous models either did not even consider all the CHF plant intricacies or could not be solved.

Chapter 6

Conclusion and Review

6.1 Conclusions

This dissertation addresses the operational scheduling of coal extraction, stacking and reclaiming under the added complexity of multiple, conflicting objectives such as blend requirements (influencing over-blending and throw-outs) and stochastic demand (influencing prop-chuting). This dissertation presents novel approaches for both modelling and solving the coal scheduling problem, as previous attempts at scheduling the coal handling processes were unsuccessful — even though these attempts did not incorporate all of the complicating practical factors.

A comparative study between solution methods indicates that the Simulated Annealing (SA) metaheuristic is best suited for this specific model. An SA metaheuristic is therefore developed to solve the complex real world scheduling problem.

The metaheuristic solution approach allows one to model the typical sequential thoughts of a control room operator and sequential operating procedures. Thus far, these sequential rules could not be modelled in the continuous solution environment required for exact solution methods (i.e. using simultaneous equations).

Because of the complex coal handling processes, the development of a practical solution took more than half of the total algorithm development time. It is, however, only possible to start implementing intelligent metaheuristic improvement strategies once a practically executable feasible solution can be generated.

Instead of the actual solution being used for neighbourhood solution representation, the developed SA indirectly represents neighbours by the rules used to generate neighbourhood solutions. Additionally, the cooling schedule is specifically customized to support larger diversification during the first stages of execution and better intensification at the final stages. Initial temperatures are determined based on the objective function value of the initial solution to prevent unfair biases to diversification when large objective function values are applicable for certain problem instances.

In Section 1.2, this project's problem statement is formulated as a research question:

> Can a metaheuristic be developed to efficiently (i.e. solution time of less than two minutes) provide a schedule to minimize blend deviations, considering multiple-objectives, stochastic variables, non-linearities and the highly constrained environment?

This dissertation successfully addresses this research question as an efficient metaheuristic is developed to schedule CHF operations (whilst considering stochasticity) in such a manner that as little as possible blend deviations occur, whilst also minimizing throw-outs and prop-chuting (the conflicting objectives). All the points in the research design, as listed in Section 1.2, were addressed to ensure that this dissertation successfully addresses the research question.

Various previous attempts at scheduling the coal handling environment over the past 5 years have not been implemented. This is beacuse they did not include all the rules and complexities inherent in this environment; which can be ascribed to the fact that the problem was not properly understood and that the solution methods used in previous attempts could not solve the intricate scheduling problems in an acceptable time. This dissertation therefore goes beyond the traditional solution approaches and adapts metaheuristic theory to derive a new approach that efficiently solves scheduling models including all the rules and complexities excluded in or missed by previous attempts.

As shown by the live tests analyzed in Section 5.3, less over-blending occurs and more stable qualities are reclaimed. This will ensure that the factory can produce at higher yields than those previously achievable and will therefore generate additional profits (which are not quantifiable at this stage). The reduction in throw-outs will also reduce operating cost of the coal handling facility by approximately R4.6 million (ZAR) per annum. Over and above the increased gas and fuel production and the cost reductions, the recourse risk that the business is exposed to is also reduced as the scheduled tasks already considers variable factory consumption and its negative impact on the coal handling business.

With old age and illness, many operators need to be replaced by inexperienced people who cannot schedule coal handling as well as experienced operators. The model developed in this dissertation therefore also creates sustainability, with regards to proper coal scheduling, in the coal handling business.

The approaches discussed in this document universally apply to any continuous product environment. The following points will illustrate that the continuous coal handling entities used can be directly translated into any continuous (typically chemical) environment:

Conveyor Pipes transporting fluids (as both moves continuous product).

Bunkers Tanks with simultaneous outflow and inflow points (as coal can be thrown into and pulled out of a bunker at the same time).

Heaps on stockpiles Tanks that can only be used for either inflow or outflow at any time (as stacking coal onto heaps (inflow) or reclaiming coal from heaps (outflow) cannot happen at the same time).

Blending The mixing of raw materials required in chemical environments (chemical reactions, blending or homogenization).

Mine production Outside suppliers or upstream processes with variable rates of supply (as in essence the mines are merely suppliers).

Factory consumption Any use of continuous product exhibiting a stochastic nature such as selling to customers or use in downstream production process (as the factory is CHF's customer).

6.2 Suggestions for Future Work

The work done in this project provides the following opportunities for future work.

6.2.1 Scheduling of Coal Handling Processes

Although this work provides the best practical schedules available up until now, some further attention still needs to be given to the CHF scheduling environment.

During the model implementation, a few programming "bugs" in the model have been corrected. Further implementation will most probably help identify any small remaining issues in the model to make it 100% practical. A process of continuous improvement is currently being setup to address further debugging by negotiating with Sasol's Operations Research group. This negotiated process is also required so that the model can be adapted to future changes in the physical infrastructure or in the operational philosophies of the business.

If some future techniques enable one to accurately model CHF in a simultaneous equation environment, one can initially use exact solution methods (to assist initial solution generation) or at the end (for further local optimization) of the metaheuristic.

As hybrid metaheuristics are often used to apply the "best of both worlds" in solving difficult problems, it should be investigated whether some Tabu Search or Genetic Algorithm principles can be incorporated into the SA metaheuristic. This hybrid metaheuristic might be able to efficiently supply even better approximate global optimal solutions.

6.2.2 Real-Time Scheduling

A major challenge at CHF is to start applying real-time scheduling, as this is required to enable automatic plant control. The SA metaheuristic can be used as a foundation when applying real-time scheduling principles. At CHF, real-time scheduling principles will avoid the need of a full re-run of the scheduler every time that the actual actions deviate too much from the scheduled actions and will reduce the number of human (operator) errors. This approach will also improve with plant stability as full re-runs of the schedule are avoided.

6.2.3 Solution Method Selection Guidelines

Operations Researchers often battle when a solution method has to be chosen for a practical model.

This document suggests first attempting exact solution methods. These methods will start failing rather early in the development process if the model is too complex for these methodologies. When exact solution methods fail, a simplified model that is representative of the characteristics of the real problem should be developed that can be solved optimally using exact solution methods. The simplified model should then be solved using alternative solution methods. The exact and alternative solution methods can then compared to each other in terms of quality of solution, distance from global optimal and solution times.

However, this approach requires an Operations Research expert familiar with all the different solution methods. This approach also demands time, a luxury not often available in industry.

A detailed study is required to provide guidelines as to when which solution methodology should be applied. These guidelines would probably refer to the number of binary variables, the number of constraints, non-linearity, convexity, complexity theory analysis of the model, required goodness of solution and allowable execution time.

Bibliography

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2006). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, Article in press.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., and Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161:86–110.

Bellabdaoui, A. and Teghem, J. (2006). A mixed-integer linear programming model for the continuous casting planning. *International Journal of Production Economics*, 104:260–270.

Birge, J. R. and Louveaux, F. (1997). *Introduction to stochastic programming.* Springer-Verlag, New York.

Brucker, P. (2004). *Scheduling Algorithms.* Springer, Berlin, fourth edition.

Busetti, F. (2005). Simulated annealing overview. In *http://www.geocites.com/francorbusetti/sa.* Accessed 04 April 2005.

Chen, M. and Chao, D. (2004). Coordinating production planning in cellular manufacturing environment using tabu search. *Computers & Industrial Engineering*, 46:571–588.

Choobineh, F. F., Mohebbi, E., and Khoo, H. (2005). A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, Article in press.

Chunfeng, W., Hongyin, Q., and Xien, X. (1999). Optimal design of multiproduct batch chemical processes using tabu search. *Computers and Chemical Engineering*, 23:427–437.

Coetzer, R. L. J. and Harmse, M. F. (2007). Report on the research project regarding scheduling technologies in Sasol. Research report, based on work by Conradie, D.G., Morison, L.E., Swart, M., Langley, I.J. and Kotze, L. Sasol.

Collings, J. (2002). *Mind over matter: The Sasol story, a half-century of technological innovation*. Sasol (Pty) Ltd, Johannesburg.

Conradie, D. G. (2004). A genetic algorithm for vehicle routing problem with multiple constraints. Final year project, Department of Industrial and Systems Engineering, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria, South Africa.

Conradie, D. G. and Joubert, J. W. (2004). Workforce sizing and scheduling for a service contractor using integer programming. *South African Journal of Industrial Engineering*, 15(2):133–139.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Massachusetts.

Eglese, R. W. (1990). Simulated annealing: A tool for Operation Research. *European Journal of Operational Research*, 46:271–282.

Ehrgott, M. (2005). *Multicriteria optimization*. Springer, Berlin, second edition.

Fleischer, M. (1995). Simulated annealing: Past, present and future. In Allexopoulos, C., Kang, K., Lilegdon, W., and Goldsman, D., editors, *Proceedings of the 1995 Winter Simulation Conference*, pages 155–161.

Furlonger, D. (2005). Clean switchover fueled the chaos. *Financial Mail*, 23 December 2005:24–25.

Glover, F. and Greenberg, H. J. (1989). New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European Journal of Operational Research*, 39:119–130. Referenced in Eglese (1990).

Goddard, S. (Accessed: 30 August 2006). Real-time systems: Introduction. http://www.cse.unl.edu/goddard/Courses/RealTimeSystems.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, USA.

Henrion, R. (Accessed: 6 October 2006). Introduction to chance-constrained programming. http://stoprog.org/.

Holland, J. A. (1975). *Adaption in natural and artificial systems*. University of Michigen Press, Ann Arbor.

Hydro, B. C. (1983). *Hat Creek project: 800MW coal preparation report*. Hat Creek, Canada.

Ierapetritou, M. G. and Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling. 2. Continuous and semi-continuous processes. *Industrial & Engineering Chemistry Research*, 37(11):4360–4374.

Jansen, K. and Mastrolilli, M. (2004). Approximation schemes for parallel machine scheduling problems with controllable processing times. *Computers & Operations Research*, 31:1565–1581.

Jia, Z. and Ierapetritou, M. (2004). Efficient short-term scheduling of refinery operations based on a continuous time formulation. *Computers and Chemical Engineering*, 28:1001–1019.

Joubert, J. W. and Conradie, D. G. (2005). A fixed recourse integer programming approach towards a scheduling problem with random data: A case study. *ORiON (Journal of the Operations Research Society of South Africa)*, 21(1):1–11.

Kall, P. and Wallace, S. W. (1994). *Stochastic Programming*. John Wiley & Sons, New York, first edition.

Kelton, W. D., Sadowski, R. P., and Sadowski, D. A. (2002). *Simulation with Arena*. McGraw-Hill, second edition.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680. Referenced in Eglese (1990).

Linn, R. and Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering*, 37:57–61.

Little, D. and Hemmings, A. (1994). Automated assembly scheduling: A review. *Computer Integrated Manufacturing Systems*, 7(1):51–61.

Longwell, J. P., Rubint, E. S., and Wilson, J. (1995). Coal: Energy for the future. *Progress in Energy and Combustion Science*, 21:269–360.

Manson, N. J. (2006). Is Operations Research really research? *ORiON (Journal of the Operations Research Society of South Africa)*, 22(2):155–180.

Maree, A. (2006). Sasol Synfuels coal processing. In *http://secabweb.sasol.com/cp/*. Accessed February 2006.

Méndez, C., Gorssmann, I., Harjunkoski, I., and Kaboré, P. (2006). A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. *Computers and Chemical Engineering*, 30:614–634.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 53:1087–1092. Referenced in Eglese (1990).

Michalewicz, A. (1992). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, New York.

Morison, L. E. (2005). Optimizing the Sasol Oil fuel distribution network. Final year project, Department of Industrial and Systems Engineering, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria, South Africa.

Mouton, J. (2001). *How to succeed in your master's and doctoral studies*. Van Schaik, Cape Town.

Page, C. and Meyer, D. (2000). *Applied research design for business and management*. McGraw-Hill, New York.

Peyrol, E., Floquet, P., Pibouleau, L., and Domenech, S. (1992). Scheduling and simulated annealing: Application to a semiconductor circuit fabrication plant. In *European Symposium on Computer Aided Process Engineering*, pages 39–44.

Philpott, A. (Accessed: 30 August 2006). Official COSP stochastic programming introduction. http://stoprog.org/.

Pochet, Y. and Wolsey, L. (2006). *Production planning by mixed integer programming*. Springer series in Operations Research and Financial Engineering. Springer, NY.

Potts, C. N. and Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120:228–249.

Ramesh, R. and Cary, J. M. (1989). Multicriteria jobshop scheduling. *Computers & Industrial Engineering*, 17:597–602.

Rardin, R. L. (1998). *Optimization in Operations Research*. Prentice Hall, New Jersey.

Rifai, A. K. (1996). A note on the structure of the goal-programming model: assessment and evaluation. *International Journal of Operations & Production Management*, 16(1):40–49.

Rutenbar, R. A. (1989). Simulated annealing algorithms: An overview. *IEEE Circuits and Device Magazine*, pages 19–26.

Sabuncuoglu, I. and Bayõz, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126:567–586.

Sen, S. and Higle, J. L. (1999). An introductory tutorial on stochastic linear programming models. *Interfaces*, 29(2):33–61.

Silva, J. D. L. (2003). *Metaheuristic and multi-objective approaches for space allocation*. PhD thesis, School of Computer Science and Information Technology, University of Nottingham.

Suresh, V. and Chaudhuri, D. (1993). Dynamic scheduling – a survey of research. *International Journal of Production Economics*, 32:53–63.

Swart, M. (2004). A scheduling model for a coal handling facility. Master's thesis, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria., Pretoria, South Africa.

Taha, H. A. (2003). *Operations Reseach: An introduction.* Prentice Hall, NJ, seventh edition.

Tamiz, M., Jones, D., and Romero, C. (1998). Goal programming for decision making: An overview of the current state-of-the-art. *European Journal of Operational Research*, 111:569–581.

Van Breedman, A. (2001). Comparing descent heuristics and metaheuristics for the vehicle routing problem. *Computers & Operations Research*, 28(4):289–315.

Van Dyk, J. C., Keyser, M. J., and Coertzen, M. (2006). Syngas production from South African coal sources using Sasol-Lurgi gasifiers. *International Journal of Coal Geology*, 65:243–253.

Winston, W. L. (1994). *Operations Research: Applications and algorithms.* Duxbury, California, third edition.

Winston, W. L. and Venkataramanan, M. (2003). *Introduction to mathematical programming*, volume 1 of *Operations research.* Duxbury, California, fourth edition.

Zhong, Z., Ooi, J. Y., and Rotter, J. M. (2005). Predicting the handlability of a coal blend from measurements on the source coals. *Fuel*, 84:2267–2274.

Appendix A

Variable Definition Notes

The following steps need to be executed to access the actual code embedded in the program. To run the model, please refer to Appendix B to prevent execution errors due to incorrect paths.

- Make sure that Microsoft® Excel's security settings allows one to access and use macros. This can be verified using the following menu path: Tools, Options, Security and then Macro Security. Newer versions of Microsoft® Excel must be set to either Medium or Low.
- Open the *Simulated Annealing for CVC v8.xls* file located on the attached CD.
- Click "Enable Macros" to ensure that the embedded code can be accessed.
- Click either "Don't Update" or "No" to make sure that links to other files and databases are not updated.
- Ensure that the *MainMenu* sheet is active.
- Right-click on the "Run SA metaheuristic for the CVC scheduling problem" button, select "Assign Macro" and click the "Edit' button.
- Microsoft® Visual Basic will now open. The code can be found in the 5 modules shown in the Project Explorer.

The code is also included in text format on the CD for persons who do not use Microsoft® products.

This rest of this section explains the formulations that are used to define the variables with. All the subscripts used in the model are explained below.

Side subscripts (1 to 2):

1. Side 1
2. Side 2

Mine subscripts (1 to 6 / 1 to 7 / 1 to 8):

1. Source 1
2. Source 2
3. Source 3
4. Source 4
5. Source 5
6. Stacking: Source 6-7 stream
   Reclaim: Source 6-7 stream
   Bunkers: Source 6 circular stockpile
7. Bunkers: Source 7
   Stacking: Bleed in
   Reclaiming: Bleed in
8. Bunkers: Yard bunker

Yard subscripts (1 to 6):

1. Yard 1
2. Yard 2
3. Yard 3

4. Yard 4
5. Yard 5
6. Yard 6

Heap subscripts (1 to 5):

1. Live 1
2. Live 2
3. Live 3
4. Live 4
5. Live 5

Time/hour subscripts (0 to 48 / 1 to 48):

- Every hour represents a time point
- Hour 0 is the hour before the hour in which the scheduler is run
- Hour 1 is the first hour for which model is solving

Transfer belt subscripts (1 to 3):

1. 2037 (Side 1 to 2)
2. 2038 (Side 2 to 1)
3. 3034 (Yard bunker coal to side 2)

Also note that most actual values at the beginning of the scheduling period are saved as time 0 in the variable arrays. The letter “l” in front of a variable name indicates that this variable is a local variable.

Appendix B

Executing the CHF SA Scheduler

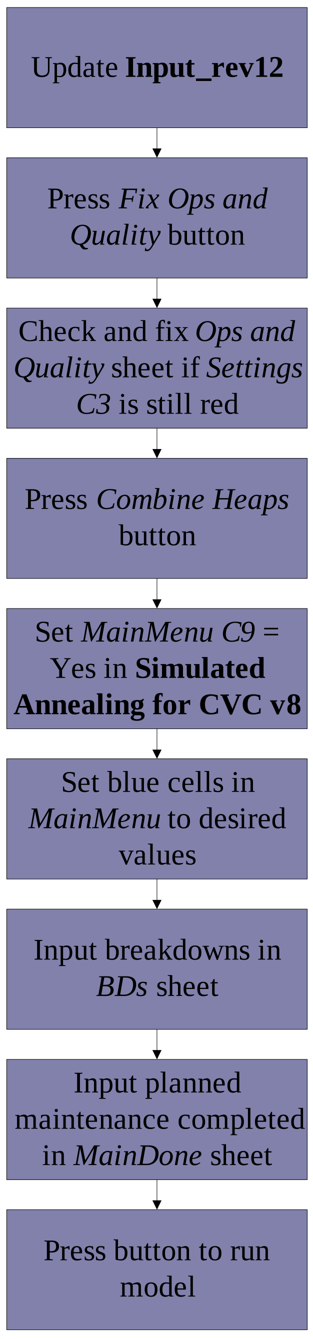
The run the scheduling model, the steps shown below should be followed. Refer to Appendix A for instructions on how to access the developed programming code.

- Copy the entire *Metaheuristic* folder on the CD to the C drive (C:\CVC\). The files in *Metaheuristic* the folder on the CD must be located on a writable hard-drive in the folder *C:\CVC\Metaheuristic*.
- Make sure that Microsoft® Excel's security settings allows one to access and use macros. This can be verified using the following menu path: Tools, Options, Security and then Macro Security. Newer versions of Microsoft® Excel must be set to either Medium or Low.
- Open the *Simulated Annealing for CVC v8.xls* file located in *C:\CVC\Metaheuristic*.
- Click "Enable Macros" to ensure that the embedded code can be accessed.
- Click either "Don't Update" or "No" to make sure that links to other files and databases are not updated (only click "Update" or "Yes" when running the scheduler on a live Sasol network).
- Ensure that the *MainMenu* sheet is active.
- Set all the options and input parameter values required on the *MainMenu* sheet.
- Click the "Run SA metaheuristic for the CVC scheduling problem" button to run the model.

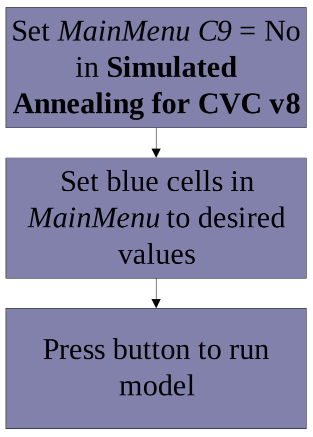
- After model execution various output graphs are included in the coloured sheets, as explained in Appendix C.

The steps shown in Figure B.1(a) should be followed when running the scheduler at CHF (note that Sasol Intranet network access is required). The process listed in the bullets above required to run the scheduler in an off-line environment (i.e. outside of Sasol), is summarized in Figure B.1(b).

Whilst executing, the graph shown in Figure B.2 will be displayed and updated. The graph shows the objective function value for the solution generated for each of the iterations (in blue) as well as the objective function value of the incumbent solution (in green).


Update **Input_rev12**
Press *Fix Ops and Quality* button
Check and fix *Ops and Quality* sheet if *Settings C3* is still red
Press *Combine Heaps* button
Set *MainMenu C9* = Yes in **Simulated Annealing for CVC v8**
Set blue cells in *MainMenu* to desired values
Input breakdowns in *BDs* sheet
Input planned maintenance completed in *MainDone* sheet
Press button to run model

(a) ***Running the scheduler at CHF***


Set *MainMenu C9* = No in **Simulated Annealing for CVC v8**
Set blue cells in *MainMenu* to desired values
Press button to run model

(b) ***Running the scheduler off-line***

Figure B.1: Process to run the SA scheduler

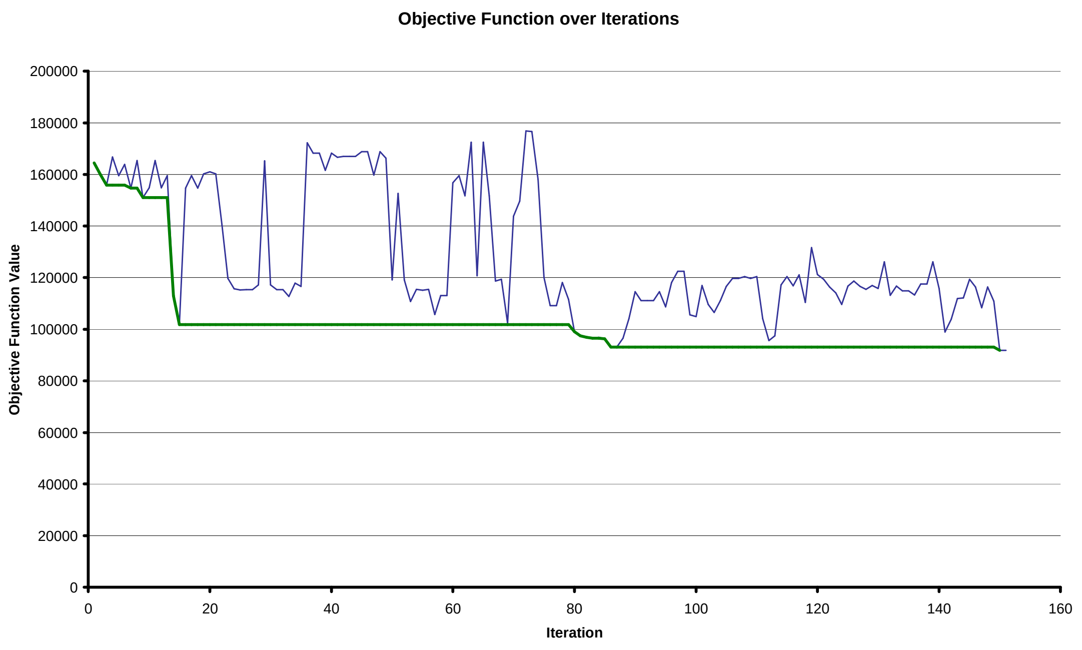


Figure B.2: Objective function over iterations

Appendix C

Model Outputs

The model has various different outputs to illustrate and describe the best solution found during execution. Each type of output of the best solution is now discussed separately.

C.1 Log Output

For debugging during development and to be able to investigate the solution in detail, a log file of the best solution is written out during execution.

Output is divided into 3 categories, each with different possible logs:

- Reclaiming
  - Reclaiming a not full heap
  - Reclaiming heap emptied
  - Prop-chute of x tons required but no source available
  - Prop-chute of x tons required
  - CP bunker below 65%
  - Reclaiming action of x ton/h
  - Reclaiming action2 of x ton/h (two heaps reclaimed in one period)
  - New heap length calculation caused division by 0 error
  - Reclaimer waiting for stacker

  - Reclaimer turned around
  - New reclaiming heap found

- Stacking
  - Tried to stack on reclaiming heap
  - New heap started
  - Over-blend 3%, bunker full
  - Stacker change-over time of x h
  - Mine source change-over time of x h
  - Stacking of x ton/h
  - Change-over time of x h carries over
- Bunkers
  - Bunker empty
  - Throw-out of x tons at mine bunker

Each log also writes out the relevant details required to understand exactly when coal must move a source to a destination:

- Time
- Yard
- Heap
- Side
- Mine

C.2 Output Graphs

Various output graphs are shown to illustrate the outputs in an understandable way:

- Expected bunker levels
- Expected circular operations
- Expected yard bunker operations
- Expected throw-outs
- Expected throw-outs per day
- Expected load-backs at mine bunkers
- Expected transfer belt operations
- A graph for the expected yard operations for every yard
- Expected stacking locations of all mines
- Expected CP bunker levels
- Expected reclaiming side 1
- Expected reclaiming side 2
- Expected reclaiming per day
- Expected ash and fines reclaimed
- Expected Carbon reclaimed

C.3 Detail Stockpiles Picture Output

When one has chosen that the model must write out the status of the stockpiles at the end of each hour in picture format, one can view the planned stockpile situation for every hour using the normal CHF picture.

To do this, open the *Prentjie SA output init.xls* file from the directory where all the metaheuristic scheduler files are located (*C:\CVC\Metaheuristic*).

- Click "Enable Macros".

- Click “Update Links” or “Yes” (depending in the type of prompt enquiring whether Microsoft® Excel should update links or not).

- Click “Continue” if a prompt appears that indicates that some links could not be updated. This might happen if the file is being run outside the Sasol network, but does not cause any problems in this workbook.

- Select the *Data* sheet.

- Type in the hour for which the picture of the stockpiles is required. If 1 is input, then the picture will shows the heaps as planned for at the end of the 1st hour (i.e. exactly one hour after the start time of the model). To view the current heaps as fed to the model as input data, select time 0.

- Then click the “Get heap data from model” button to update blend and quality information for that period. The file will automatically update the data on the Data sheet from the *Prentjie output data init.xls* file.

- Thereafter click the “Update Data” button.

- Also note that it is extremely difficult to calculate the planned percentages and some rounding errors might occur.

- Repeat the process from the fourth bullet onwards to view the stockpile status for a different hour.

Figure C.1 shows an example of the stockpiles picture output. A similar output is used by CHF to report information regarding the plant status on a daily basis. The output used at CHF was modified so that the model’s predicted output can be shown to the users in a familiar format.

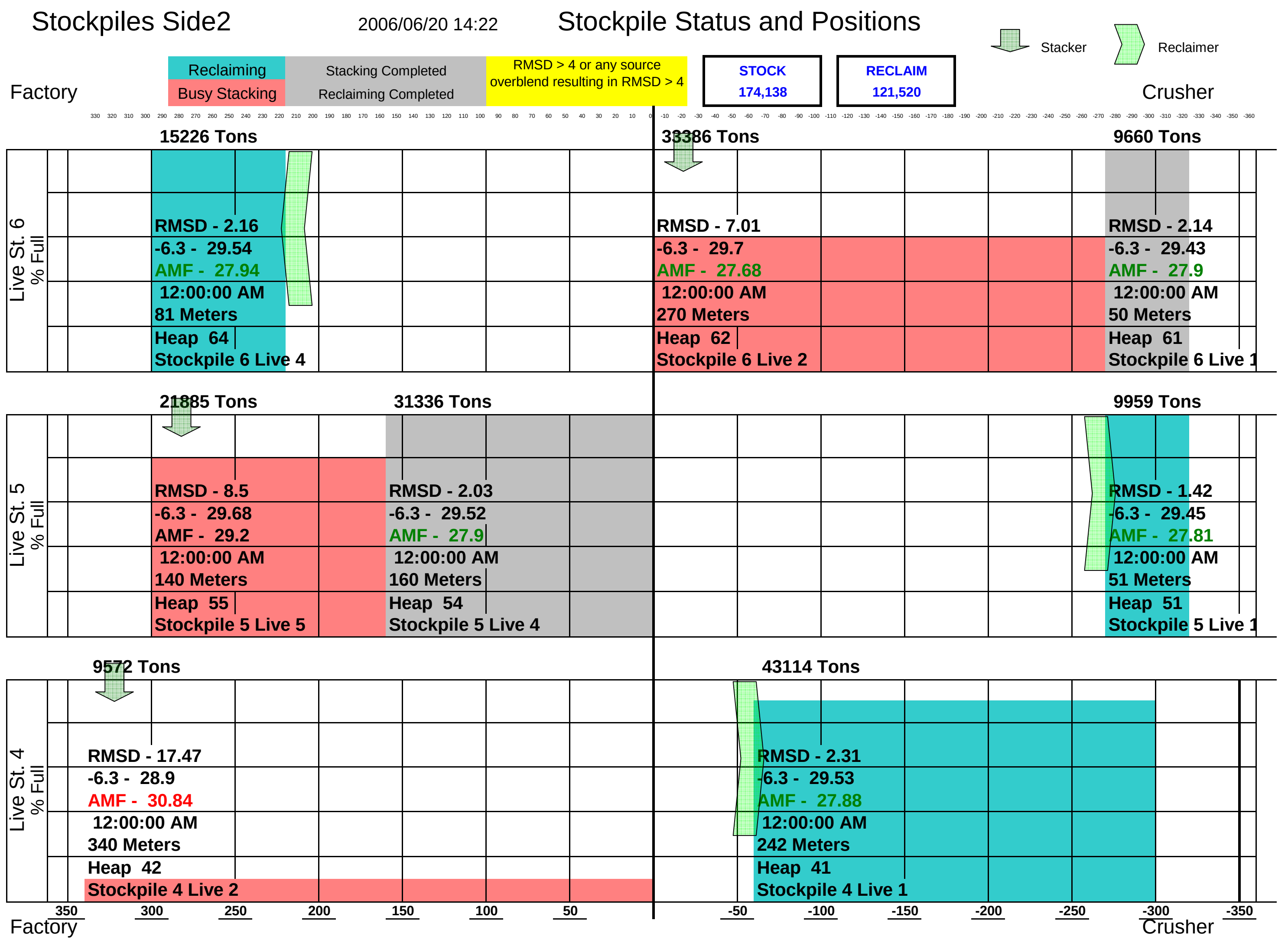


Figure C.1: Stockpiles picture output