School of Science & Engineering
Department of Mathematics & Statistics

Optimised Decision-Making under Grade Uncertainty in Surface Mining

Francois Grobler

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

September 2015

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature:

Date:

"Although our intellect always longs for clarity and certainty, our nature often finds uncertainty fascinating."
- Carl von Clausewitz

"Information is the resolution of uncertainty."
- Claude Shannon

Acknowledgements

Firstly, I would like to thank my supervisor, Lou Caccetta for his support and encouragement during this journey.

I also want to thank my associate supervisor, Jose Saavedra-Rosas, for his inputs and guidance around the thesis structure, mathematical modelling and programming in Python as well as acting as a constructive critic for my often exotic and tangential ideas - he usually managed to reel me back in although not always without a lively exchange of thoughts...

Most importantly, to my wife Céline, for tolerating the last four years and for being a willing (mostly) sounding board for new ideas, sometimes responding with a stifled yawn but always with a supporting ear.

I would also like to thank CAE Mining for providing the conditional simulation data sets which have been used for a major part of this thesis.

Abstract

Planners and managers tasked with strategic open pit scheduling in real industrial situations are faced with two key dilemmas.

Firstly, finding optimal answers for the Open Pit Production Scheduling Problem requires the solution of large combinatorial optimisation problems. Unfortunately, and despite a lot of effort, progress has been slow in finding ways to solve these large problems satisfactorily (or at all) in a reasonable amount of time through mathematical programming approaches.

The second issue is that until recently, most analysts and planners have relied on deterministic inputs to their models and have ignored parameter uncertainty. In doing this they have assumed in effect that information related to geological and grade variability, as well as the future behaviour of commodity prices and other economic factors (amongst many others), are known at the time that the decision is made. In reality these inputs are highly uncertain and depending on which specific instances are selected, could lead to vastly different results in pit designs and mining schedules.

With reference to the first issue, the size problem is commonly dealt with by either reducing the size of the input data set or by implementing clever formulation and solution strategies.

From the data perspective, a quick and easy strategy with drastic effects is to remove such data from the model upfront which are superfluous from a geometric point of view. In other cases, an attempt is made to lift the burden of dealing with large models by re-blocking or aggregating blocks into data sets with a smaller more manageable number of blocks. This makes it easier to solve the problem but potentially dilutes the resolution of the input data and results.

From the solution angle, heuristic methods are often used to solve large models faster than exact mathematical methods but this sacrifices accuracy and the ability to prove optimality. Some encouraging advances have been made over the last decade using mathematical programming methods such as mixed integer linear programming (MILP) in the formulation and solving of the problem without jeopardising the resolution of the input data. Key improvements include intelligent data pre-processing, removing obsolete variables with respect to earliest and latest start times, the use of branching algorithms with relaxed integer constraints, exploiting strong branching characteristics of the problem, the introduction of improved cutting planes, and decomposition methods such as Lagrangian relaxation.

The second problem (input uncertainty) has received less attention. A promising simulation method (Conditional Simulation) has enabled decision makers in the mining industry to generate multiple realisations representing the spatial variability of geological features and attributes such as metal grades and densities of a deposit. When such simulations are considered together they provide insights into the estimation of the uncertainty inherent in the geological or grade model due to limited information. Similarly, the stochastic behaviour of commodity prices resulting in limitations in accurate forecasting has been captured by methods such as Monte Carlo simulation and Real Option Valuation.

Unfortunately many of these methods, although bestowing massive improvements compared to historical deterministic models, still result in multiple outcomes (or scenarios) which ultimately leave the decision maker with the dilemma of having to select the best option.

In this thesis we first draw on some basic advancements made in the field of mathematical programming to construct a simplistic MILP formulation for a small instance of the Open Pit Production Scheduling Problem. We then introduce multiple Conditional Simulation data sets which are successively solved using the MILP formulation generating candidate solutions to the problem. We also introduce an interpretative framework which compares average values against accompanying standard deviation. We combine these two metrics into a single indicator, the coefficient of variation (CV), which we propose to be used to find a suitable trade off between risk (standard deviation of values) and return (average of values). This metric can then be used to make sense of candidate solutions and

optimisation results derived from multiple grade instances based on Conditional Simulations. This framework enables the identification of a single or a select number of most 'attractive' options by considering the trade off between risk and return (expressed as the CV).

We then use a version of the MILP formulation to incorporate a Scenario Optimisation approach which was developed in the 90's and applied to projects in power generation and financial portfolio optimisation. The approach can be used to solve a stochastic model based on a particular method for combining scenario solutions into a single feasible and 'robust' strategy. The results from this approach are then similarly tested for viability by using the interpretive framework.

This thesis combines advancements made in exact mathematical methods with a probabilistic scenario optimisation approach which incorporates and considers grade uncertainty while arriving at a single 'best' or most attractive solution (although not necessarily highest value or lowest risk). The resultant methodology is tested on a case study of 40 conditional simulations.

The three key contributions of this thesis are:

1. Scenario Optimisation

   By combining an exact deterministic MILP optimisation approach based on Caccetta and Hill (2003) with multiple conditional simulation inputs we utilised a modified two-stage scenario optimisation approach to generate a unique solution to an open pit scheduling case study.

   This approach has an initial stage during which optimised solutions are generated using a MILP objective function maximisation. This is then followed in a second stage by finding a solution which minimises the sum of the absolute differences between the original solutions derived in stage one and the current solution applied to the equivalent simulation data set.

   This two-stage scenario optimisation approach was modified from Dembo (1992) who used it in financial portfolio optimisation and in hydroelectric power generation. Our research indicates that this is the first time this approach has been used in a mining context.

2. Interpretive Framework using Coefficient of Variation

In order to incorporate both risk minimisation and value maximisation into the decision making criteria when comparing various 'competing' candidate solutions, we developed an interpretive framework which utilises the coefficient of variation (CV) as a measure which includes both factors. The framework based on the CV enables comparison of multiple optimisation results and their associated statistics when compared against the underlying data. Further, it provides an intuitive and easy way of identifying those solutions that offer a favourable trade-off between risk and return.

Although the coefficient of variation is a well known statistical metric for normalising and comparing different scenarios and options with different means and standard deviations, our research indicates that this is the first time the CV has been used in such an interpretive framework and certainly the first time it has been used in the mining context for comparing optimisation solutions derived from conditional simulations.

3. Data Perturbations

We conducted early exploratory research with a concept that we call 'data perturbation' and that we believe has a lot of merit for further research and investigation.

Generally, a number of ways can be utilised to generate valid candidate mining schedules. These can be as basic as starting from the top of a deposit and mining naively downwards bench by bench, and as sophisticated as following a true holistically optimised solution.

As long as certain requirements are fulfilled, such as adherence with predetermined precedence constraints (or pit slope angles) and compliance with minimum and maximum annual production capacity constraints, any such schedule will qualify as a 'valid' solution. Of course it could be a particularly poor economic solution (it might even have a negative NPV) but it would be permissible as a solution.

One way of generating such a permissible solution is to use a slightly perturbed version of the underlying conditional simulation data set to generate an optimised solution. This can for instance be achieved by applying a small random modifying factor (for example a uniformly selected factor that varies between -5 and +5 percent) on the true block value.

Although such a modification is not permitted in generating valid conditional simulation data sets during geo-statistical estimation, it could nevertheless be used as the basis for an optimised schedule. When such a solution is then compared against the underlying true condition simulation data, it might lead to improvements in the CV and therefore provide a more attractive solution than those derived directly from solving conditional simulation datasets, or those derived via scenario optimisation approach.

We believe this approach is a unique way of generating a large number of 'inexpensive' candidate solutions which can be assessed for incremental improvements in value or CV. As this method arose as an outflow of our research relatively late in the thesis time-frame we could only conduct cursory exploration on it but we believe that it merits further investigation and research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the context for the rest of the thesis. Within this chapter we lay out in basic terms:

(a) a brief description of the problem and the environment in which it occurs (Section 1.1),

(b) the key issues that we would like to address in the thesis (Section 1.2),

(c) some of the main classical and recent research literature generated on the key topics of interest (Section 1.3),

(d) the objectives that we aim to achieve through this research (Section 1.4),

(e) what is the motivation for this line of research (Section **??**),

(f) the methodologies that will be followed to answer the questions raised (Section 1.5),

(g) areas where the findings and conclusions from this research can find practical or commercial application (Section 1.6), and

(h) what we plan to achieve from each subsequent chapter in this thesis (Section 1.7).

1.1 Overview

Mining decisions related to large open pits mined over tens or even hundreds of years can be massively complex and involve a myriad of different components and considerations. There are many areas within such operations where Operations Research in general and optimisation techniques specifically can be successfully introduced to gain the maximum benefit. These techniques are mostly implemented on the development and exploitation stages of mining projects and operations. Areas of application include ore-body modelling and ore reserve estimation, the design and scheduling of optimum pits, determining optimum blends and cut off policies, equipment maintenance and fleet optimisation studies to name but a few.

This thesis however occupies itself with the Open Pit Mining Production Scheduling Problem. A strategic solution to this particular problem involves the decision on:

(a) which of the available blocks in a mining block model to extract by surface mining methods (i.e. mine them or leave them in the ground),

(b) when (in which time period) should the extraction take place, and

(c) what to do with them (send a mined block to waste, to the processing plant for treatment or to a stockpile for later consideration).

All these decisions need to be made with the objective of maximising the economic value of such an exercise while satisfying safe wall-slope (precedence) restrictions, production capacity, market factor and environmental constraints.

Problems such as these are routinely formulated as mathematical models with pre-determined input values, decision variables and technical or capacity constraints. Such problems are then solved as linear programming (LP) or integer/mixed-integer programming (IP or MIP) formulations using commercial software.

One of the key limitations of these models is the high level of uncertainty in the input values (e.g. prices and ore/metal grades) and assumptions used to construct the models originally as well as the extent to which this is taken into account or ignored. In some cases this can be further exacerbated due to the size of the models and the number of variables required to be solved for models reaching anything close to realistic sizes.

1.2 Problem statement

The issue that we are trying to address in relation to the uncertainty problem can be stated as follows:

Many of the current mine scheduling optimisation approaches ignore the uncertainty around input data (e.g. grade values, density) thus failing to assess the inherent risk or robustness of derived solutions. Often when they do incorporate risk through methods like Conditional Simulation, they provide no method to interpret results or isolate a single or few most suitable options.

1.3 Previous research

Over the last few decades the Open Pit Mine Production and Scheduling Problem has been approached from many angles. Sequential methods divide the problem into components by first generating a solution to the ultimate pit problem, followed by generating periodic phases (also called 'push backs') and only then finding schedule solutions to the individual subcomponents. Pioneering work using a sequential approach and focussing on solving the ultimate pit problem was done by Lerchs and Grossmann (1964) in the 60's using Graph Theory, Dynamic Programming and Network Flow methods. Numerous researchers subsequently built on 2-D and 3-D versions and modifications of Lerchs and Grossman's original work. Original literature utilising a Network Flow method was proposed by Picard (1976) and expanded on by various others. Heuristic methods have also been used to solve the ultimate pit problem with one of the most well known being the Moving (Floating) Cone Method. Early solutions using the Moving Cone Method were proposed by David et al. (1974) and Lemieux (1979) and modifications to the method has been proposed by many others.

More holistic or integrated methods considered the problem as a whole and attempted to solve the problem without subdivision into parts. These methods include linear (LP) and mixed integer linear programming (MILP) methods combined with solution strategies such as branch and bound and cut generation, as well as decompositions methods such as Lagarangian relaxation. Key literature from within this genre include Caccetta and Hill (2003), Gaupp (2008), Askari-Nasab et al. (2010), Darby-Dowman and Wilson (2002), Ramazan and Dimitrakopoulos (2004), Gershon (1982) and Weintraub et al. (2008).

Compared to the volume of work done on deterministic mine planning and schedule optimisation, research done on introducing uncertainty into mine planning has been relatively sparse. Since the early 90's Dimitrakopoulos and various contributors have been exploring the use of Conditional Simulation as a way to incorporate uncertainty into the ultimate pit design and production scheduling problem e.g. Journel and Huijbregts (1978), Dimitrakopoulos (1990), Dimitrakopoulos (1994) and Armstrong and Dowd (1994).

An interesting approach for incorporating uncertainty using Scenario Optimisation was suggested by Ron Dembo (Dembo, 1992) in the 90's but never applied to the mining industry. In this thesis we build on his approach.

1.4 Research Objectives and Motivation

The primary objective of this study can be stated as:

To devise an effective method which enables the inclusion of geological (grade) uncertainty into the problem formulation to derive a risk-adjusted 'robust' solution (i.e. a solution that involves and acceptable trade off between risk and return) and to develop an interpretive framework to make sense of the results as an aid to decision making.

The motivation for this research is that grade variability and uncertainty is routinely ignored in the compilation of mine planning models. In the cases where grade risk is introduced through conditional simulations, the exercise involves the generation of multiple simulations with limited sense-making occurring in the interpretation of the results often still leaving the decision maker with multiple options without the capacity to select the 'best' or most attractive solution(s).

1.5 Research Methodology

The research methodology is as follows:

(a) Review literature and approaches used in the past to deal with the primary problem under review.

(b) For a suitable base metal deposit, formulate a deterministic MILP model for a small to medium size block model (1,000 to 50,000 blocks) based on previous research and solve the problem using a commercial solver (CPLEX). Perform pre-processing on input data in order to reduce the size of the block model and data inputs as much as possible.

(c) Generate or obtain a block model which includes a suitable number of conditional simulations representing multiple equi-probable interpretations of the block model data.

(d) Generate a probabilistic modelling framework enabling iterative MILP optimisations of conditional simulations recoding schedules derived

(e) Develop an interpretive framework with which to conduct comparative analysis of derived options (schedules) in order to find a most suitable ('robust') solution which considers both value and risk.

(f) Investigate other approaches to incorporate model uncertainty in the solution (e.g. scenario optimisation) - add these solutions to the interpretative framework to see how they compare with schedules derived through optimising conditional simulation data sets.

1.6 Areas of Application

The key components of the mine planning process are illustrated in the following flow diagram (see Figure 1.1). Five areas have been identified within the optimisation set-up and solution process where it is believed that enhancements to the process can add significant value to the quality, solution time and interpretation of an optimisation.

1. Area A - during the pre-processing stage (see Chapter 4.3):

   (a) the size of the problem (data) can be significantly reduced by considering (and excluding) redundant blocks which can structurally not be part of the solution due to their geo-spatial location.

   (b) some waste blocks can be excluded from the bottom layers of the block model since they do not add value to the solution and will therefore never be selected anyway.

2. Area B - during the model generation stage (see Chapter 4.4):

   (a) blocks could be excluded during the variable declaration stage or as added constraints based on earliest and latest available start times.

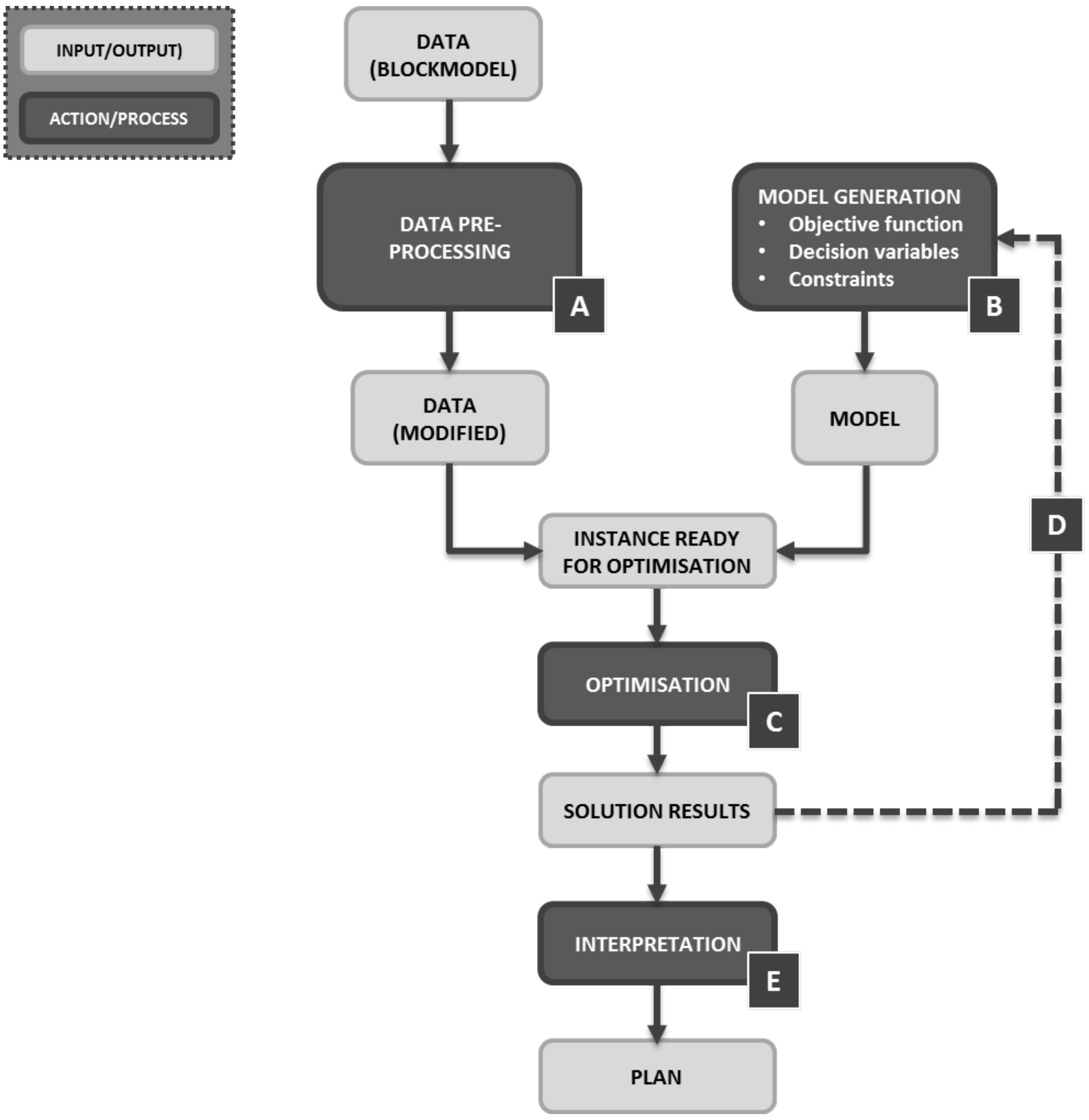


Figure 1.1: Flow diagram

(b) such constraints are possible as a result of the combination of annual maximum/minimum mining capacity constraints and precedence constraints.

3. Area C - during the optimisation (solve) stage (see Chapter 4.4):

   (a) select realistic MIP gap settings (e.g. 1% to 5%) for the optimiser (e.g. CPLEX)

   (b) select realistic CPU time or iterations as termination criteria for the optimiser

   (c) select realistic memory capacity allocation

4. Area D - during an iterative stage between solution and redefining the model:

   (a) during an iterative approach such as branch and cut where the optimiser is used to solve only the relaxed LP (and not the MIP) strategies can be introduced to reduce the solution space via additional constraints as the optimisation results are returned.
   (b) using call back facilities additional constraints, cuts, node priorities etc. can systematically be introduced.

5. Area E - during interpretation of results (see Chapter 5.4):

   (a) sense making going beyond just basic descriptive statistical summaries of the generated outputs
   (b) developing an interpretive framework which clearly produces a way to disseminate and visualise results
   (c) an interpretive framework which assists in identifying a single (or a select few) most favoured option.

1.7 Organisation of the thesis

The rest of this thesis is organised as follows:

Chapter 2 defines general mining terminology and then focuses on concepts specific to the open pit environment, geo-statistics and block modelling.

Chapter 3 provides a literature review focussing on research related to open pit optimisation following sequential and integrated approaches. This section includes various mathematical ('exact') and heuristic methods that researchers have developed to formulate and solve the Open Pit Production Scheduling Problem with special focus on ways to deal with the model size problem. A second section of this chapter reviews literature related to the introduction of variable and parameter uncertainty via probabilistic or stochastic methods.

Chapter 4 details the formulation of a MILP model and applies a number of known strategies to improve the manageability or solvability of the model. These include data preprocessing strategies to reduce the size of the data set, as well as model formulation and optimiser settings. A model is built in the Python (Pyomo)/CPLEX environment and solved, validated and interpreted.

Chapter 5 considers the introduction of grade uncertainty into the open pit optimisation problem through the use of Conditional Simulations. The chapter proceeds to develop an Interpretive Framework which enables a decision-maker to make sense of results derived from developing schedules from Conditional Simulations and testing them against the range of Conditional Simulation input data. The approach defines a strategy for integrating risk and return (the risk/return trade-off) by using the Coefficient of Variation of the resultant solution values as a metric thus proposing a method to identify the most attractive candidate solutions.

Chapter 6 considers a number of scenario optimisation techniques for including uncertainty, building on the research of Dembo (1992) and others. The chapter expands on various ways to develop models which recognise the variability introduced by conditional simulations. The two scenario optimisation approaches tested derive unique solutions which respectively minimises the average deviation from the expected value and maximises the average value of the solution. This method is tested against the Interpretive Framework developed in the previous section.

Chapter 7 provides a summary of the results and conclusions of the thesis and proposes some interesting areas for further research.

Chapter 2

Open Pit Production Scheduling

2.1 Introduction

This chapter reviews key concepts and definitions relevant to the mining environment in general and the Open Pit Production Scheduling Problem in particular.

The chapter starts off by describing key mining activities and physical processes on an open pit mine (Section 2.2) defining and illustrating the most important concepts to the Open Pit Production Scheduling Problem.

In Section 2.3 the flow of data is described from the initial drilling and sampling to the finalisation of a block model.

In Section 2.4 geostatistics is introduced together with the key concepts relevant to conducting resource estimation.

2.2 Mining operations

2.2.1 Mining methods

The mining of hard-rock mineral deposits occurs either via surface mining methods, underground methods or a combination thereof. An open pit mine utilises a surface mining method, which is suitable when ore material is close to the surface of the earth. This method of exploitation has been around for much longer compared to underground mining methods and is more productive. Surface mining methods include strip mining and open-pit mining and is a broad category of mining in which soil and rock overlying and waste interspersed with the mineral deposit are removed (see Figure 2.1). In comparison, in an underground mine the overlying rock is left in place, and the material of interest is removed through shafts or tunnels.

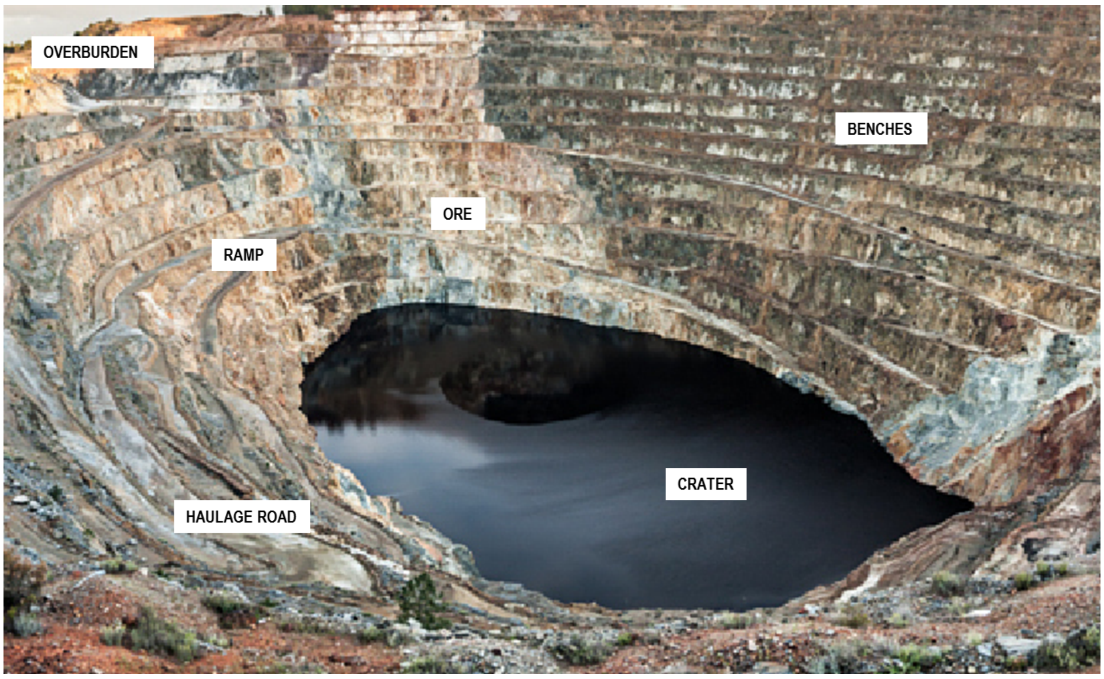


Figure 2.1: Open pit mine

In order to commence mining of valuable mineralised material (ore), large amounts of overburden (waste) must often be removed. This might include both unconsolidated material, removed relatively easily (and inexpensively), or harder material that requires drilling and blasting with explosives.

2.2.2 Mining activities

Surface mining operations typically employ the use of trucks and shovels.

Shovels (or excavators) are large mechanical diggers with various sized buckets, working at the exposed faces (called benches) in the open pit.

In the most conventional configuration, haul trucks transport ore and waste from the bottom of the open pit to the surface where it is dumped as waste or stockpiled for later treatment as ore. Material to be processed immediately is deposited into large crushers which reduce the blasted fragments to more manageable sizes. Other forms of transport from the pit to the processing facility might include conveyor belts or piping.

Ore is further processed in a treatment (or processing) plant by means of physical or chemical processes which separate the valuable material from waste or tailings to produce a liberated or concentrated product. This product might be sold as it is or undergo further processing (e.g. smelting, refining, beneficiation) before it is sold to consumers in the market.

2.3 Data collection and interpretation

The profitable extraction of mineralised ore and waste requires the development of a pit design (outline) and an extraction schedule for the operations. The development of an optimised design and mining schedule requires the effort of a number of role players and the collection, calculation, modification and compilation of a large amount of data. This process starts already during drilling and sampling of drill cores and compositional assay of samples by a qualified laboratory in order to obtain sample grades and densities. This information is then compiled into a database for use by the geologist to create a geological and grade model.

2.3.1 Drilling, sampling and geological modelling

The process of converting information obtained from a series of drill holes into a block model is explained by Froyland et al. (2004). A number of long, narrow holes are drilled into the ore body (or where it is expected to be) and the cores are removed, logged and analysed ('assayed') for mineral concentrations and other material characteristics. Geologists use drill core logs and their location/spatial information to create a 3-D geological or lithological model by interpreting the spatial continuity of geological units - see Figure 2.2.

Important information obtained from such analyses are the material density and metal content or concentration (the grade) contained in the drill cores. Drill hole cores however only provide a sparse and limited set of data from which a full 3-D model of rock and metal characteristics needs to be constructed. Such a model construction is commonly done using geostatistical methods.

2.4 Geostatistics

Geostatistical estimation is a deterministic approach used in the generation of estimates for block values (e.g. metal grade) in the absence of perfect information. Various well known classic textbooks Journel and Huijbregts (1978), Clark (1979), Deutsch (2002) and guidelines are available that expand on the concepts, benefits and limitations relevant to geostatistics.

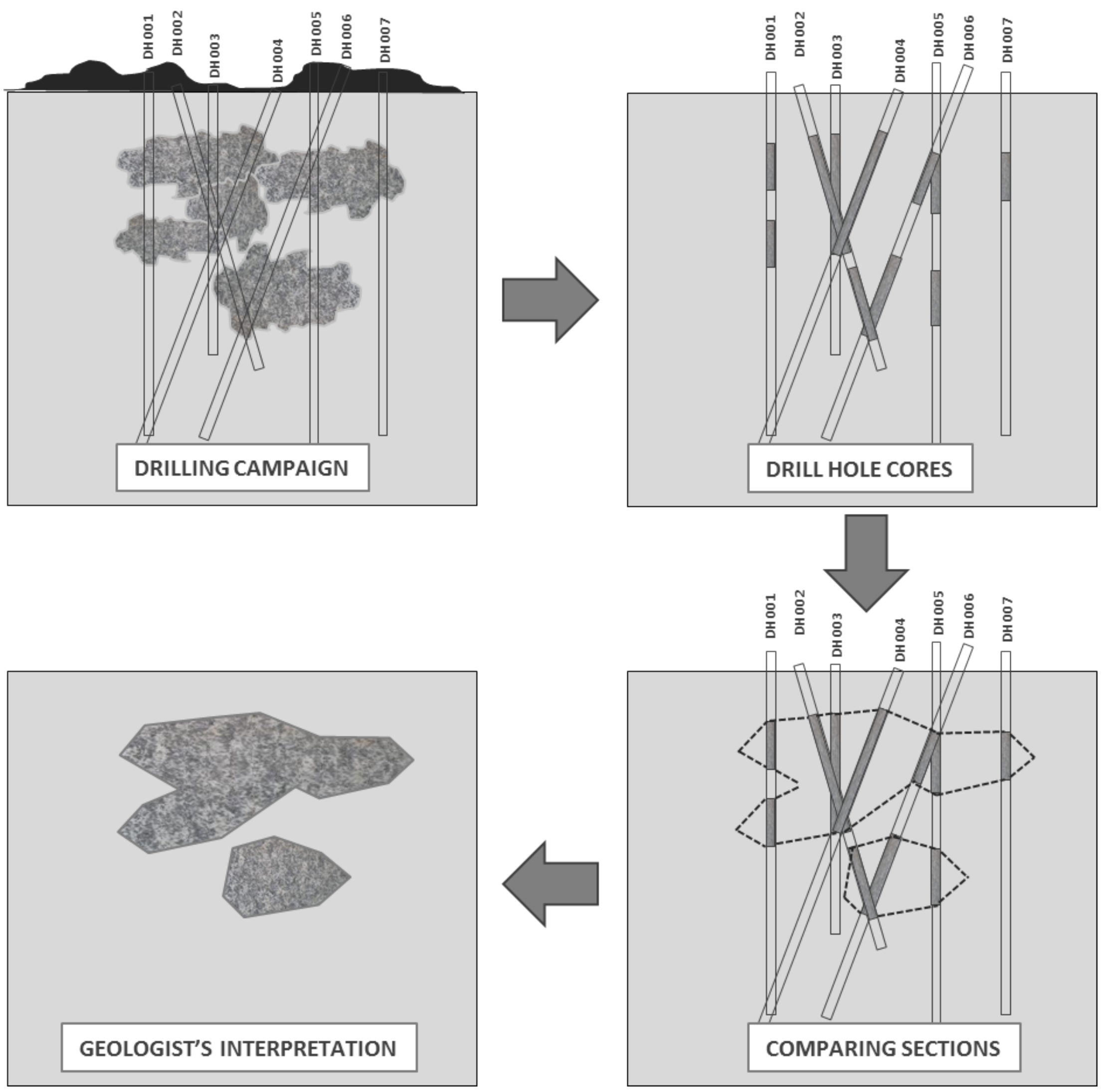


Figure 2.2: Geological interpretation from drill hole data

Whereas classical statistics hold that individual members of a population are positionally unrelated or random, in geostatistics a fundamental assumption is that individual members of a population within the mineralisation are positionally dependent and related. It is this dependence that enables the development of a mathematical model of a mineral deposit. This relationship is the basis of assumptions and predictions made of the size and grade of a mineral deposit.

A key geostatistical concept is that the values of samples located inside a block to be estimated are related to the value of the block and that the values of samples located closest to the block to be estimated are similarly related to the value of the block. Based on this assumption, a mathematical relationship (function) can be developed between the values of the samples and the value of the block based on distance to the centre of the block. This mathematical relationship is represented as a graph called the 'variogram'.

The (semi) variogram - There are numerous different types of variograms (relative variogram, semivariogram, log transformed variogram, etc.) The standard spherical variogram is depicted in Figure 2.3.

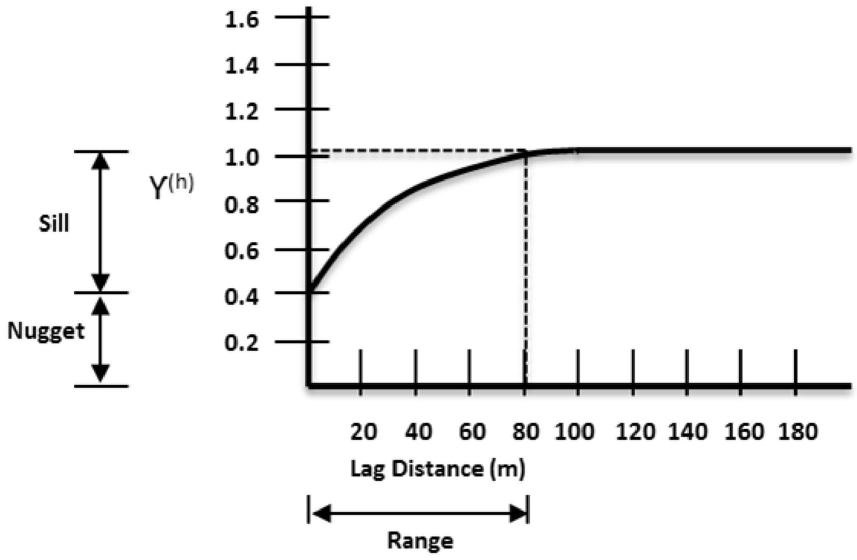


Figure 2.3: Typical (idealised) Spherical Variogram

A number of basic concepts are relevant to the interpretation of the variogram:

(a) the vertical axis represents the grade variability $\gamma^{(h)}$ between individual points in the sample population.

(b) the Nugget represents the inherent (natural) variability of a sample - the result of inherent geologic variability or it could also be an 'apparent' effect due to sampling, sample preparation, and/or assaying error.

(c) the Sill indicates the point at which the variability between individual points in the sample population becomes uncorrelated.

(d) the Lag Distance (horizontal axis) is the average distance for grouping points for variogram calculation (i.e. minimum distance between sample points).

(e) the Range (shown on the horizontal axis in Figure 2.3) indicates the maximum distance of influence which permits the correlation of values of samples and the value of the estimated block. At distances beyond the range, the mathematical relationship becomes random.

Kriging is a geostatistical estimation technique which has been commonly used for a number of decades. A number of different types of kriging variations are used depending on the circumstances (i.e. block kriging, point kriging, rankorder kriging, multiple-indicator kriging, etc.)

Kriging is believed to provide the best linear unbiased estimator of a point and the best linear weighted moving average of a block. Kriging minimizes the grade estimation variance when calculating the block grade from individual sample values. One of the well known limitation of block kriging is that it tends to over smooth the estimated values (Vann et al. (2002) and Journel and Huijbregts (1978)) resulting in estimated high-grade values that are less than the original sample values, and estimated low-grade values that are higher than the original sample values. By comparison, Conditional Simulation (explained later in this chapter) generates more representative values (see Figure 2.4).

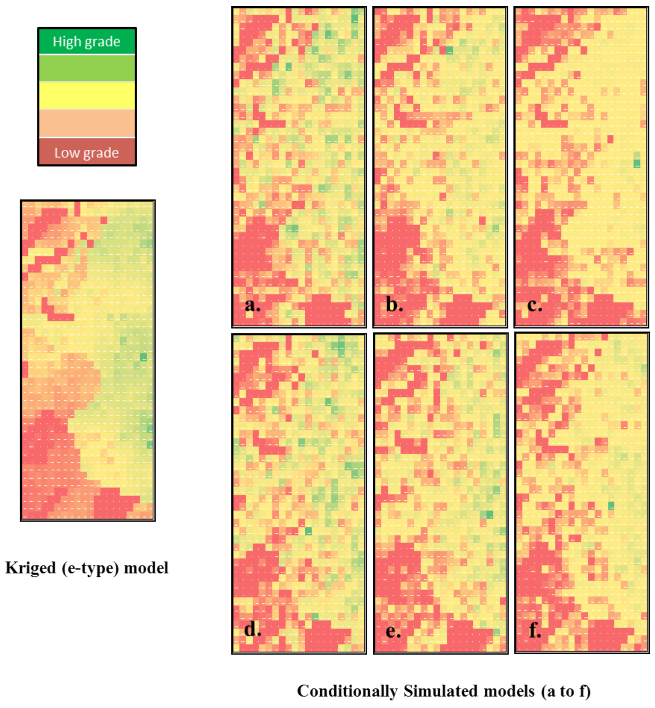


Figure 2.4: Kriged vs. Conditionally Simulated models illustrating smoothing effect

2.4.1 Creating a block model

Mineralised ore bodies are modelled using a collection of blocks called a block model (see Figure 2.5). This type of model commonly consist of a three-dimensional accumulation of blocks which are characterized by their spatial location, weight and associated contained value (Journel and Huijbregts (1978) and Hustrulid and Kuchta (2006)).

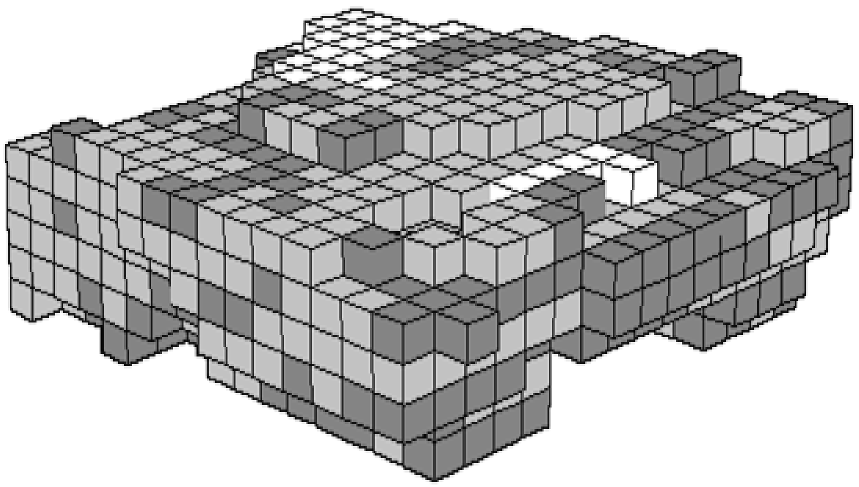


Figure 2.5: Three-dimensional geologic block model representation of blocks (cubes) in a deposit

Block considerations - The regular three-dimensional fixed-block model is commonly used in block modelling since it is ideally suited to the application of computerised optimization techniques Gignac (1975). In addition to blocks being equally sized, the actual orthogonal dimensions of the blocks will be determined by a number of additional factors. These include:

(a) the physical characteristics of the mine - the orientation and dip of the deposit, structural features (faults, fractures, fold) which in turn influence safe pit slope requirements

(b) the variability of grade and metallurgical attributes - the more 'massive' the features the large the blocks, whereas intricate convoluted pockets of mineralisation require smaller block sizes in order to access the ore

(c) the size and capacity of mining equipment used - large scale mining equipment does not require small block sizes

The physical size and dimensions of blocks in a model and their effect on selectivity and value is discussed by Jara et al. (2006) and Hekmat et al. (2013) respectively.

Wharton (1996) cautions that decisions on block model size depend on the purpose of the modelling exercise. For instance this could be for (a) delineating the ore body, (b) for reserve value calculation, (c) for designing a pit or (d) for sensitivity analysis.

Block value - If a block contains sufficient valuable material compared to its total weight then the block can be classified as *ore*. A block that does not contain sufficient value to justify its extraction and processing expenses is classified as *waste*. Whether a block is classified as ore or waste will determine the profit (or loss) that the mining of a block might yield. However this decision is often not a straightforward decision and dependant on many factors.

The decision on whether a block should be classified as ore or waste is typically determine by the '*cut off grade*'.

According to Rendu (2008) the cut off grade is the minimum amount of valuable product or metal that one metric ton (1000 kg) of material must contain before this material is considered for processing (and therefore extraction of the valuable product).

The direct profit or loss (or utility) $U_{dir}(x)$ expected from processing one metric ton of material of grade x is $U_{ore}(x)$, which is expressed as follows:

$$U_{ore}(x) = x \cdot r \cdot (V - R) - (M + P + O) \tag{2.1}$$

where:

- x is the average grade of valuable product contained in a metric ton of material (also called the 'ore')
- r is the metal recovery, or proportion of valuable product recovered from a metric ton of mined material
- V is the value of one unit of recovered valuable product (the 'grade') contained in a metric ton of material
- M is the mining cost incurred in mining a metric ton of material
- P is the processing cost incurred in processing a metric ton of material

- O is the overhead cost cost incurred in mining and processing a metric ton of material
- R is the refining, transportation and other costs incurred per unit of valuable product contained in a metric ton of material

In order for values to be attributed to individual blocks, a number of techno-economic factors need to be decided. These include:

(a) Commodity price forecast for the metal(s) for which value needs to be assigned. These could be static prices which remain constant over the scheduling period or period specific prices if that level of detail is available or required.

(b) Mining operating costs associated with e.g. drilling and blasting, digging/excavation, hauling, primary crushing.

(c) Processing or treatment operating costs associated e.g. secondary (or tertiary) crushing, leaching, floatation, de-watering, concentration, comminution etc.

(d) Metal recovery% related to different types of rock and their interaction with mining and processing equipment and processes.

(e) Discount factor, which is related to the deteriorating effect of time on value. Blocks scheduled for mining later will be more severely discounted ('penalised') than blocks mined earlier. A decision on the discount rate (which determines the discount factor) is important and is a function of the company's funding regime (lending, equity or owner's capital) plus a risk premium.

(f) Capital expenditure (CAPEX) associated with the mining and processing equipment and facilities. CAPEX is often excluded from a schedule optimisation unless the objective requires the resolution of variables associated to capacity levels and their associated CAPEX.

When the point is reached where a block model is developed with spatial locations (x,y,z coordinates), block dimensions (block size in the x,y,z directions) and key attributes such as block density (or tonnage), metal grade, ore/waste classification and recovery then further economic evaluation and analysis can be performed.

Economic Evaluation - Accumulating the values of these blocks over a time horizon together with the capital expenditure required to provide production and processing capacity to the process enables the derivation of a discounted cash flow (DCF) and net present value (NPV). The ultimate objective of any mining evaluation is typically to maximizing the NPV of mined material and processed ore given a certain set of assumptions (e.g. prices, costs, capital expenditure).

A number of factors need to be understood and taken into account during the planning of the open-pit operation to determine the shape and size of an open-pit. Some of these factors include according to Armstrong (1990):

(a) geology, lithology, topography, property boundaries

(b) extent of the deposit, localisation of mineralisation

(c) mining method, mining rate, processing rate, bench height

(d) cut off grade, pit slopes

(e) material and metallurgical characteristics, ore recovery

(f) mining and processing costs

Some of these considerations can be seen illustrated in Figure 2.6. For example, the bench height, which is the vertical distance between each horizontal level of the pit, should be set as high as possible within the limits of the size and type of equipment selected for the desired production.

The pit slope is one of the major factors that determines the amount of waste to be removed so as to mine the ore. The spatial location of blocks is important since blocks on lower levels of the pit cannot be accessed until those blocks above it have been removed in order to maintain the integrity of pit slopes. Removing of blocks should therefore honour certain spatial requirements (precedence constraints).

The two examples show these aspects for two different types of deposit. The one on the left has a tabular sub-vertical orientation, whereas the one on the right has a more lensoidal and disseminated nature.

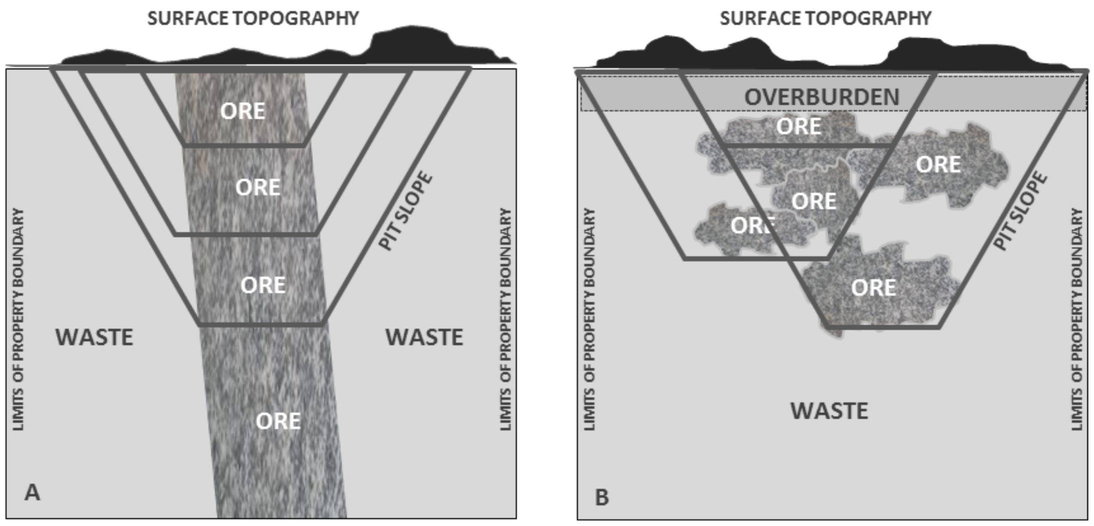

SURFACE TOPOGRAPHY
ORE
ORE
ORE
ORE
PIT SLOPE
WASTE
WASTE
LIMITS OF PROPERTY BOUNDARY
LIMITS OF PROPERTY BOUNDARY
A
SURFACE TOPOGRAPHY
OVERBURDEN
ORE
ORE
ORE
ORE
ORE
PIT SLOPE
WASTE
LIMITS OF PROPERTY BOUNDARY
LIMITS OF PROPERTY BOUNDARY
B

Figure 2.6: Example open pit configurations for two general types of deposit

At the end of the modelling process, mine planners have to decide whether, and if so when, blocks in a block model should be extracted and which blocks need to be discarded or sent to the treatment plant for further processing as ore. This is done through mine planning and scheduling.

For more explanations of mining terms, see Hustrulid and Kuchta (2006).

2.5 Geological and Mining Uncertainty

In the mining industry, decisions linked to millions and even billions of dollars are made based on limited information both from the point of view of our understanding of the deposit which we hope to excavate profitably, and our ability to forecast future prices, costs and economic trends that will materialise only as time passes. Despite this lack of information, most mining companies still stick to traditional deterministic decision making methods. Fortunately over that last few years, probabilistic modelling approaches such as Conditional Simulation which take account of risks and uncertainties has seen an increase in their application.

Before defining Conditional Simulation it is worth looking at some general sources of risk and uncertainty in mining.

2.5.1 General Sources of Mining Risk

Mining projects are invariably designed on the basis of highly uncertain variables arising due to the inherent nature of the variables as well as the prohibitive cost of improving the level of knowledge. Variables related to geological, metal content and material characteristics are quantified based on sparse drilling and sampling data. Dowd and Pardo-Iguzquiza (2002) notes that since such programmes typically provide data on a relatively large (regional) scale, the levels of resolution is understandably at an order of magnitude greater than the scale required for modelling, prediction and risk assessment. Key sources of mining risk and uncertainty are:

(a) **Geological Uncertainty** - the availability of drilling and sampling data in mining projects and operating mines are generally limited. This introduces a high level of geological uncertainty around the qualitative (what does it contain?), quantitative (how much does it contain?) and positional (where is it located?) characteristics of mineral deposits. These characteristics include the physical extent of the deposit, location of geological boundaries and contacts, geo-technical features (e.g. faults, fractures and folds) and geo-metallurgical properties (e.g. material density and metal recovery). Dunham and Vann (2007) expand on the importance of including geo-metallurgical uncertainty around issues such as characterisation of metallurgical recovery and throughput.

(b) **Financial and Revenue factors** - another major contributing factor to the total uncertainty of a mining project is the commodity price and exchange rates. Depending on the type of commodity these are typically also highly uncertain. Metal prices are influenced by many contributing factors associated with supply (other producers) and demand (consumers). Prices are further influenced depending on whether commodities are sold based on long term contracts or spot market prices, seasonality in demand, level of substitution, demographical growth etc.

(c) **Operating Costs** - mining and processing operating costs (OPEX) are also uncertain since they are associated with unpredictability in seasonal and growth trends of inflation rates, as well as availability and cost (supply and demand) of key production factors such as labour, tyres and power).

(d) **Capital Expenditure** due to the complexity of the cost estimation process, as well as the long duration between the time of estimation and the time of expenditure, capital expenditure (CAPEX) is similarly uncertain.

(e) **Efficiency and Performance factors** - factors such as metal recovery, dilution, production and processing efficiency are uncertain since they are either tied to geological uncertainty or related to future performance of human resources and equipment.

(f) **Other** - more subjective risks such as political, country related, social and environmental is difficult to quantify and often ends up in some form built into the project specific discount rate as a risk premium.

The combined effect of all these levels of uncertainty is that geologists and mining engineers have a difficult time compiling evaluations and business cases for mineral projects. Typically, uncertainty in estimation and forecasting is prohibitive on the accuracy of assumptions made in feasibility studies. This fact is evident from confidence intervals that decision makers typically put around different levels of study. Research conducted by Hall (2007) proposes confidence intervals for various types of studies from around 30% for conceptual studies, typically 20-25% for pre-feasibility studies and 10-15% for feasibility studies. In addition to these confidence intervals, Bullock (2011) indicates that a range of contingencies are also typically added to the bottom-line expected cost of the estimate depending on the level of accuracy.

Arguably the two most important areas of uncertainty influencing the accuracy and validity of mining case studies are those associated with geological (grade) and price uncertainty.

Within this thesis we only focus on the quantification of grade uncertainty but a lot of work has been done on various ways to incorporate price uncertainty into the mine planning and scheduling process. Some of the key methods used are Monte Carlo Simulation and Real Option Valuation.

Chapter 3

Literature Review

3.1 Introduction

This chapter reviews key literature relevant to the Open Pit Production Scheduling Problem. It starts off by looking at research related to deterministic methodologies and then follows with a review of research which introduces grade uncertainty into the optimisation problem via probabilistic and stochastic modelling.

The next section (Section 3.2) considers the most relevant modelling and optimisation methods (deterministic) and related literature. This is divided firstly into Sequential Approaches (Section 3.2.1), which break the problem up into ultimate pit optimisation, push back generation and then scheduling. We then secondly consider Integrated Approaches (Section 3.2.2) which solve the problem as a whole.

In Section 3.3 we introduce risks related to mining with a focus in Section 3.3.3 on the use of Conditional Simulation methods. Section 3.3.4 discusses various Stochastic approaches and Section 3.3.5 covers Scenario Optimisation approaches.

3.2 Deterministic Mine Planning and Scheduling Methodologies

A strategic solution to the open pit production scheduling problem involves decisions pertaining to:

(a) which of the available blocks in a mining block model to extract (i.e. mine them or leave them in the ground),

(b) when (in which time period) should the extraction take place, and

(c) what to do with the extracted material (send mined block to waste, to the processing plant for treatment or to a stockpile for later consideration).

Open pit mining production scheduling has been traditionally approached via two different angles. In the one case (the more traditional approach) the development progresses sequentially through three stages starting with the determination of the ultimate pit limits, followed by the development of a number of push backs or mining pits. This is then followed by the solution of optimised schedules for each push back. This is called the **Sequential Approach** in this thesis.

The other approach, called the **Integrated Approach** here, takes a holistic view of the problem to determine the optimal block extraction sequence, avoiding sub-optimal mine schedules that are inherently created using the sequential approach.

3.2.1 Sequential modelling approach

During the Sequential Approach, the mine design and schedule is generated in three distinct stages.

(a) Firstly, Ultimate Pit Limits are determined by applying an expected commodity price to the model and solving essentially a single period (i.e. no discounting) optimisation. This step requires that the cut off grade (the grade separating ore from waste) is fixed.

(b) In a second step, the blocks contained in the Ultimate Pit is subjected to a parametrisation procedure which generates a set of Nested Pits each corresponding to different price or revenue factors. Nested pit shell are then grouped together in such a way as to generate a limited number of Push Backs which allow reasonable ore to waste ratios see Figure 3.2.

(c) In a final step the blocks in each push back or grouping is solved separately as a Schedule allowing the sequencing of blocks in time.

Generating the Ultimate Pit

The Ultimate Pit is the design outline which delimits the extremities of the material targeted for mining (or the optimal boundary). The set of blocks contained within this design maximises the total profit of the pit so that adding or removing any blocks from this design will lead to a loss in value. The problem of finding the set of blocks that should be removed in order to maximise the total profit from a mine, subject to the constraints on pit slopes, is known as the **Ultimate Pit Limit Problem (UPLP)**.

Over the years, various types of methods have been employed to solve the UPLP. These include Graph Theory and Network Flow algorithms, Dynamic Programming, heuristics such as the Moving (Floating) Cone method, parametrizing functions and exact optimization approaches such as Linear Programming.

A large volume of literature is available on the solution to the UPLP. Overviews of various vintages on the applicable literature are provided by Kim (1979), Kim et al. (1988), Laurich and Kennedy (1990), Dincer and Golosinski (1993), Dagdelen (2001) and Newman et al. (2010). Traditionally the UPLP has most often been solved using versions and modifications to Lerchs and Grossmann's Graph Theory algorithm (Lerchs and Grossmann, 1964) or by solving Picard's Network Flow formulation (Picard, 1976).

Graph Theory was applied by Lerchs and Grossmann (1964) to model an open pit mine as a weighted, directed graph. In the same paper they also used dynamic programming to solve two-dimensional versions of the UPLP. In this model where vertices represent blocks and arcs represent mining restrictions (i.e. sequencing constraints) ultimate pit limits are determined by solving for the maximum closure of this graph. The problem of generating optimal mine schedules was raised and researched by Johnson (1968) who developed an LP formulation for optimising the timing of extraction of regularised resource blocks within a mine, with a maximum NPV objective. Johnson and Sharp (1971) presented a three-dimensional dynamic programing method for determining the optimal ultimate open pit limit which is an extension of the two-dimensional Lerchs-Grossmann approach. Koenigsberg (1982) reviews the state of the art of dynamic programming and demonstrates the connection between linear programming (and network flow) models and the dynamic programming method. Caccetta and Giannini (1988) build on the graph-theoretic approach of Lerchs and Grossman through the use of a dynamic programming technique to 'bound' the optimum. A similar approach to Lerchs and Grossman with some modifications has been employed by Zhao and Kim (1992) in which they claim to solve problem instances faster than the method proposed by Lerchs and Grossman. Huttagosol and Cameron (1992) extended the optimizing model based on the maximum closure of a graph developed by Lerchs and Grossmann for solving ultimate pit limit problems. Yegulalp and

Arias (1992) builds on the work of Johnson and Barnes (1988) to offer a modification of the push-relabel maximum flow problem (the excess-scaling algorithm). They show however that Whittle Programming Pty. Ltd.'s implementation of the Lerchs and Grossman algorithm has faster computation times than their own implementations of the excess-scaling algorithm. Seymour (1995) proposes a modified 3D Lerchs-Grossmann graph tree algorithm that generates a complete set of nested maximum valued pits in a single run. Underwood and Tolwinski (1998) build on the work of Lerchs and Grossman by developing a network flow algorithm based on the dual formulation to solve the same problem. Their results adds insight to the LG algorithm and provides proposals for efficiency improvements. Hochbaum and Chen (2000) makes a detailed comparison between the LG algorithm and the maximum flow 'push-relabel' algorithm and proposes an efficient implementation of the push-relabel algorithm adapted to the features of the open-pit mining problem. Askari-Nasab and Awuah-Offei (2009) explores the validity of a theorem that a pit outline determined by an optimal long term schedule algorithm is constrained by the conventional Lerchs and Grossmann's optimised pit outline. This hypothesis was investigated through a case study using an intelligent open pit simulator founded on agent based learning theories. Despite the fact that the Lerchs-Grossmann algorithm provided an important tool for mine design over the years, its use for scheduling is restricted to mines having a very short life (up to 3 years) as time is not an input parameter (Caccetta and Hill, 2003).

Network flow algorithms have been used to model the ore body as a directed node-weighted graph which is associated with the 3-dimensional block model. A network of nodes represents the blocks in the mine, connected by arcs which represent the sequencing constraints between the blocks. The network model of the ore body is constructed by addition of two nodes to its graph model: a source node and a sink node. Arcs are then generated from the source to all positive nodes, and from all negative nodes of the graph to the sink. The goal of such a model is to maximize the flow from a source node to a sink node. Arcs originating at the source which have excess capacity upon termination of the algorithm indicate to profitable blocks that should be mined in the optimal solution. Such profitable blocks together with any blocks required to be mined to reach them represent the ultimate pit limits.

Network flow algorithms have been utilised by Picard (1976), Caccetta and Giannini (1988), Giannini et al. (1991), Yegulalp and Arias (1992) and Hochbaum and Chen (2000) to derive solutions for the ultimate pit limits problem.

In the **Dynamic Programming** approach, the UPLP is usually decomposed into smaller sub-problems and most favourable decisions for the sub-problems are to be made at each stage. Namely, the decisions are optimised at subsequent stages rather than simultaneously. With this approach a trade off is made between the desire to obtain the highest possible value at the present stage against the implication this would have on value at future stages (Saavedra Rosas, 2009).

Dynamic programming methods have been used to solve two-dimensional versions of the UPLP (Lerchs and Grossmann, 1964) while three-dimensional versions of the problem are solved by Koenigsberg (1982), Wilke and Wright (1984), Wright (1987), Caccetta and Giannini (1988), Sevim and Lei (1994) and Yamatomi et al. (1996).

A number of heuristic methods have been used to solve the UPLP. One of the most well known is the **Moving (Floating) Cone Method** which has been widely used due to its simplicity. The Moving Cone Method assumes a block has a reference point for expanding the pit upward according to given pit slope rules. This upward expansion containing all blocks (whose removal is necessary for the removal of the reference block) forms a cone whose economic value can be determined. Early solutions using the Moving Cone Method were proposed by David et al. (1974), Lemieux (1979) and modifications were proposed by Yamatomi et al. (1996). A more recent variation of the floating cone algorithm called the floating stope method was developed for underground by Alford (Alford and Hall, 2009) but it fails to render optimal ultimate pit limits.

Approaches based on exact optimization are also common ways of solving the ultimate pit limits problem. **Linear programming methods** are used by Meyer (1969),Gershon (1982) and Huttagosol and Cameron (1992) to solve the ultimate pit limits problem. Gershon (1983) identifies and discusses computational difficulties arising in the solution of linear programming models as a mitigation against their widespread use in mine scheduling. In particular he notes with respect to Johnson's LP formulation that it mined partial blocks producing infeasible solutions as sequencing constraints are violated. Due to the fractional solution provided by the LP formulation, integer constraints are needed to produce feasible solutions. While the MIP formulation is theoretically robust, the solution of large scale problems was not practical with the computer hardware of the 80's and 90's (Gershon, 1983) but this changed with the research of Caccetta and Hill (2003).

Creating push backs

Following the Sequential Approach, the optimization of open pit mine design consists primarily of defining the ultimate pit limits and then dividing up the pit into manageable volumes of material often referred to as push backs (also called cutbacks, or phases).

A popular technique for producing push backs is to utilise an algorithm that produces the ultimate pit and rerun it multiple times over the orebody model while the economic block values are scaled down ('parametrized') by a series of decreasing factors - e.g. see Figure 3.1 showing a length-wise section through the block model.

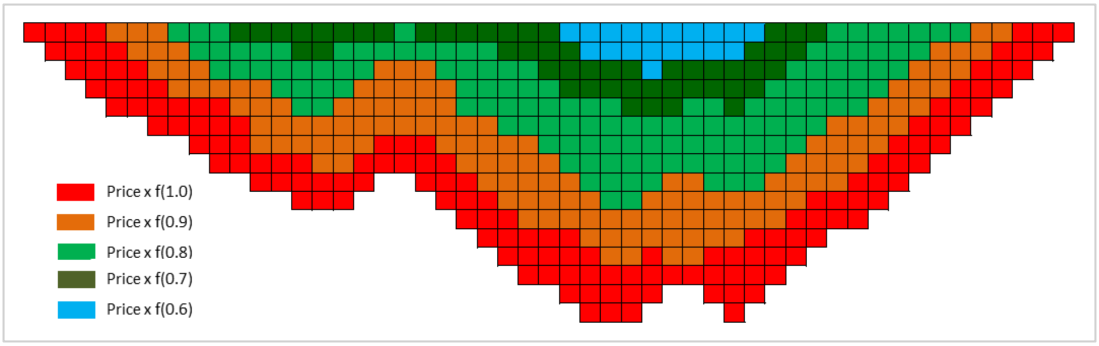


Figure 3.1: Nested pits illustrating parametrisation - section view length-wise through block model

Push backs, allow the mine designer to develop short term schedules for a smaller more manageable sub-set of the blocks considered. They also contribute to the yearly production schedules so one can apply an economic discount rate when calculating the Net Present Value of the mine. Push backs are produced from the sections of the orebody model that remain within the ultimate pit limits. Push backs and benches are often combined to generate bench-push back units - see Figure 3.2.

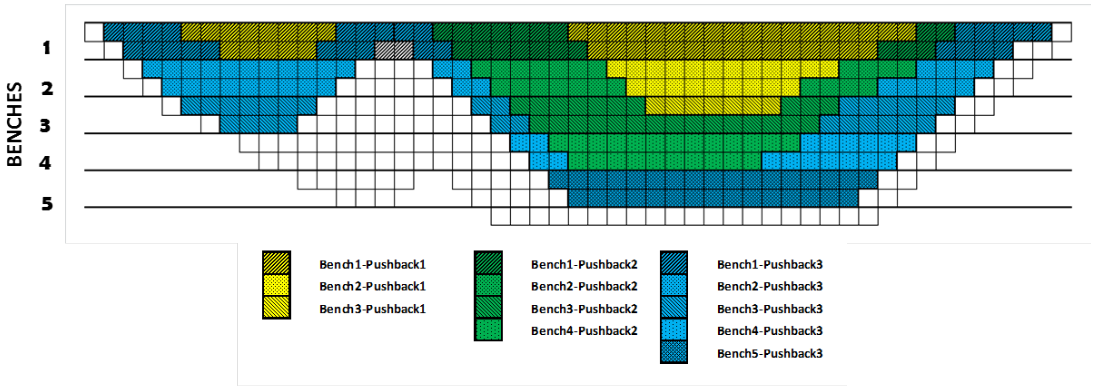


Figure 3.2: Bench-push back units - section view length-wise through block model

The concept of parametrization was introduced by Lerchs and Grossmann (1964) as a way to generate an extraction sequence. Using an undiscounted model, they proposed varying the economic value of each block by penalising it with an increasing factor. This leads to a set of nested pits which can be used to produce a production schedule. Dagdelen and Francois-Bongarcon (1982) present development of a double parameterization algorithm and Dagdelen and Johnson (1986) describe the fundamentals of a new algorithm based on lagrangian relaxation and parameterization. Modifications to the original work has also been proposed by Caccetta et al. (1998).

A number of authors have expanded on the theme of nested pit or push back generation. Mathieson (1982) discusses an overall mine planning methodology with emphasis on the design of an optimal sequence of pit expansions or phases. Kim and Zhao (1994) reviews methodologies for developing the pit sequence or staggered pit phases. Gu et al. (2010) presents a model in which a sequence of geologically optimum pits is first generated and then dynamically evaluated to simultaneously optimize the number of phases, the geometry and location of each phase-pit (including the ultimate pit), and the ore and waste quantities to be mined in each phase. The objective is to maximize the overall net present value.

Scheduling

As the final step in the Sequential Approach Optimised schedules are derived within and are bounded by the predetermined volumes defined as push backs. Traditional production scheduling methods are commonly performed using push backs designed to maximize the economic value, or metal content within each incremental push back in a greedy fashion.

Issues with the Sequential Approach

Some of the common issues with the Sequential Approach that lead to sub-optimal production schedules are:

(a) not considering requirements in grade and ore quality parameters;

(b) inability to incorporate mining and processing capacity constraints;

(c) inability to incorporate blending targets in the calculation of the ultimate pit boundaries (Stone et al., 2007);

(d) ignoring the deposit grade uncertainty;

(e) unsuitability of the LG shells as a basis for mining phase design in more complex operations with:

- multi-dimensional blended ore targets,
- multiple products with market constraints,
- multiple processes with capacity constraints,
- sinking rate constraints
- exposed ore constraints

(f) large variations in size of the push backs, or the so-called ‘gap problem’ leading to impractical results;

(g) not considering discounting during the optimization;

(h) assuming that a greedy approach will maximize discounted value.

3.2.2 Integrated modelling approach

The second approach is comprehensive and attempts to solve the scheduling problem in a single stage without preconceived assumptions about ultimate pit limits, push backs or stages.

A more integrated approach using mixed-integer linear programming (MILP), allows for the simultaneous definition of optimal mine production schedules within their ultimate pit limits without having to determine push back phases. MILP methods have been used for solving the mine production scheduling problem using a branch and cut strategy together with linear relaxation (Caccetta and Hill (2003), Bley et al. (2010)).

Current integrated production scheduling methods can be roughly divided into Heuristic methods and Exact methods.

Heuristic methods

Heuristic methods include Local Search algorithms such as Hill Climbing and Simulated Annealing, Tabu Search, Particle Swarm Optimization, Ant Colony Optimization and Genetic Algorithms.

Local Search algorithm paradigms aim to find an optimal solution for a problem by means of selecting the next potential solution from a set of neighbours of the current solution (hence the name local search). A general characteristic of local search algorithms is that it is 'memoryless' meaning that decisions taken are not influenced by decisions taken in the past or future considerations.

The **Hill-Climbing Algorithm** is a variation on the local search heuristic theme. In this approach the decision to progress is governed by each successive solution considered being an improvement to the previous (i.e. lower cost or greater value). The algorithm can be 'randomized' if it selects prospective moves randomly from neighbours, or 'greedy' if it only selects from the best next moves. Hill-climbing methods often end up at local optima but benefit from simplicity and solution speed as favourable arguments especially when applied to large problems, as an approximation or for rapid exploration of the solution space.

The **Metropolis and Simulated Annealing** algorithms are modifications on the theme of hill-climbing algorithms. They employ a probabilistic scaling on improvements similar to a temperature parameter. Simulated Annealing allows for varying temperature.

Yun et al. (2005) briefly covers the so-called field of 'computation intelligence' citing the use of genetic algorithms, genetic programming, evolutionary programming, artificial neural networks and ant colony algorithms as recent applications in solving mining problems.

Amaya et al. (2009) describe a scalable IP-based methodology for solving very large (millions of blocks) instances of this problem by embedding standard IP technologies in a local-search based algorithm they obtain near-optimal solutions to large problems in reasonable time.

In the **Tabu-Search Algorithm** the local search algorithm contains a mechanism which tries to avoid certain possible loops captured on a tabu list aiming to provide a type of memory regarding recently visited unfavourable states not to be chosen during future state updates. Lamghari and Dimitrakopoulos (2012) presents a meta-heuristic solution approach based on Tabu search for the open-pit mine production scheduling problem with metal uncertainty.

The **Particle Swarm Optimization** technique was derived from computer simulations of animal flocking and schooling behaviour. It expands on the concept of social sharing of information as an evolutionary advantage.

Ant Colony Optimization developed from observations in the behaviour of ants. It is used as a metaheuristic to find approximate solutions to difficult problems - see Sattarvand and Niemann-Delius (2008) and Sattarvand (2009).

Genetic algorithms are founded on principles of natural selection. These probabilistic algorithms exploit information gathered during exploring of the search space. The algorithms incorporate natural and evolutionary concepts such as chromosomes, selection criteria, reproduction, roulette selection, populations, mutation and crossover.

Simulated Annealing and Genetic Algorithms are discussed or utilised by Clement and Vagenas (1994),Thomas (1996), Pendharkar and Rodger (2000) and Sattarvand and Niemann-Delius (2008).

Kumral and Dowd (2005) use simulated annealing to solve a mine production scheduling problem. After the Lerchs-Grosmann algorithm is used to define the ultimate pit limits, production scheduling optimization is done in two stages: Lagrangian parametrization, resulting in an initial sub-optimal solution, and multi-objective simulated annealing, improving the sub-optimal schedule further.

Guo et al. (2010) presents intelligent optimization methods including genetic algorithm (GA), particle swarm optimization (PSO) and modified particle swarm optimization (MPSO) are used in optimizing the project scheduling of a coal mine in China.

Exact methods

Exact methods include linear programming (LP) and Integer Programming (IP) and Mixed Integer Linear programming (MILP) as well as hybrid methods of these which include heuristics (e.g. Lagrangian Relaxation) and search algorithms (e.g. branch and bound / cut).

Linear programming (LP) is the most widely used technique in open pit mine scheduling optimisation.

Goal programming is a branch of multi-objective optimization, which is an extension or generalisation of linear programming to handle multiple, normally conflicting objective measures. Each of these measures is given a goal or target value to be achieved.

Esfandiari et al. (2004) attempts to derive a production schedule by using a multiple criteria decision-making model and zero-one non-linear goal programming as effective ways to achieve higher profits in a pre-assumed time span and under specific economic and technical considerations.

Mixed Integer Linear Programming (MILP) - Darby-Dowman and Wilson (2002) provides an excellent overview of recent development in linear and integer programming methods.

Caccetta and Hill (2003) critically examine and point to limitations of techniques typically used in the mining industry for production scheduling. They then present a mixed integer linear programming model for scheduling problems together with a Branch and Cut solution strategy. Computational results for practical sized problems are discussed.

A number of authors have proposed MILP formulations to various mine scheduling problems e.g. Gershon (1982), Darby-Dowman and Wilson (2002), Caccetta and Hill (2003), Ramazan and Dimitrakopoulos (2004), Weintraub et al. (2008) and Askari-Nasab et al. (2010). Smith (1998) notes that the two major limitations associated with mathematical programming-based production scheduling are the use of integer variables and the representation of operational constraints as linear equations of these integer variables. Finding solutions for MIP or MILP problems are difficult due to the intense computational effort required by the branch and bound algorithm compared with an equivalent Linear Programming formulation. This seriously limits the size of scheduling problems that can be tractably solved using MIP/MILP approaches.

Chicoisne et al. (2012) propose a new decomposition method for the precedence-constrained knapsack problem, and show how we can use this to solve the LP relaxation of large production-scheduling formulations. Moreno et al. (2010) present an algorithm for solving the LP relaxation with and LP-based heuristic to obtain feasible solutions for the mine scheduling problem claiming to solve mining instances with millions of items in minutes and obtaining solutions within 6 percent of optimality.

Through the use of **Lagrangian Relaxation** it is possible to move certain constraints to the objective function and in this way 'relaxing' the original formulation (Caccetta et al., 1998). The violation of the constraints is assigned a cost in the objective function that has to be optimal (Saavedra Rosas, 2009). Some suggestions for Lagrangian dual heuristic and time aggregation approaches for the open-pit scheduling problem has been suggested by Amankwah (2011).

Fundamental Tree Algorithms (as proposed by Ramazan et al. (2005)) are used to reduce the number of binary variables and constraints required in MIP formulations. The algorithm uses linear programming to combine blocks into Fundamental Trees where a set of combined blocks is classed as a Fundamental Tree if the combined blocks satisfy specific conditions. After generating Fundamental Trees for a deposit, a modified MIP model formulations is used to generate annual production schedules.

Due to the combination of potentially large numbers of blocks, numerous possible time periods (10 to 20 or more) as well as multiple possible destinations, this could result in an extensive number of variables when formulated as a MILP. Add to this an equally large number of capacity and precedence constraints and such a model can easily 'explode' into a problem that becomes impossible to solve within a reasonable time frame. One of the ways of dealing with the exponential explosion of variables are by grouping ore blocks (aggregation) into larger units for mine scheduling. This assists in reducing the total number of variables but at the cost of precision and resolution of the result.

Aggregation methods have been proposed which reduce the number of blocks and therefore the number of variables. Johnson et al. (2002) proposes a methodology which combines, or aggregates, mining blocks in order to decrease the number of integer variables in scheduling without losing the optimality. Their method involves a fundamental tree concept to combine the blocks. Many of these methods use aggregation on mining decisions but still allow for processing decisions to be made at block level Stone et al. (2007), Gleixner (2008), Weintraub et al. (2008) and Boland et al. (2009).

Early elimination of obsolete blocks and data reduction strategies are discussed by Elevli et al. (1990), Hochbaum and Chen (2000) and Gaupp (2008).

Klotz and Newman Klotz and Newman (2013) present ways to assess optimizer performance on MIP problems and demonstrate methods for improving that their through creative formulation and algorithmic parameter tuning.

Bley et al. (2010) present a strengthened integer programming formulation for the open pit mine production scheduling problem by adding inequalities derived through the combining of precedence and production constraints. This allows them to exploit the special structure of the sub problems, deriving several additional constraints whose addition to the standard formulation leads to tighter linear relaxation and subsequent improvements in solution times.

Bienstock and Zuckerberg (2009) present a new iterative Lagrangian-based algorithm for solving the LP relaxation of the precedence constrained production scheduling problem. They propose that the algorithm can be proven to converge to optimality even for problems with millions of variables and tens of millions of constraints and that convergence to proved optimality is usually obtained in under 20 iterations, with each iteration requiring only a few seconds to solve with current computer hardware.

Fricke (2006) present a new approach for determining facets of the precedence constrained knapsack polyhedron based on clique inequalities.

Alvarez et al. (2011) continuous functional analysis improving on the need to construct binary decision variables which traditionally give rise to large-scale combinatorial and Mixed Integer Programming problems.

Cullenbine et al. (2011) propose a sliding time window heuristic for approximately solving an integer-programming formulation of the open pit mine block sequencing problem. The method is based on solving a sequence of mixed-integer programs that have fixed variables in early time periods, a full model representation in at least one middle period, and a relaxed representation in later periods and uses Lagrangian relaxation.

Kumral (2012) relaxes block destination as a decision variable and finds that in comparison with the conventional planning approach of pre-defining cut-off grades (thus classifying blocks as ore or waste) this approach could lead to significant value improvements albeit at the cost of additional variables.

3.3 Probabilistic and Stochastic methods

3.3.1 Background

A large body of research has been completed over the last few decades focussing specifically on the various forms of risk and uncertainty in mining and mine planning. Monkhouse and Yeates (2005) lists key sources of uncertainty that affect mine planning and Topal (2008) provdes an overview of some of the main approaches to include risk into mining evaluations.

Mining projects are invariably designed on the basis of highly uncertain variables arising due to the inherent nature of the variables as well as the prohibitive cost of improving the level of knowledge. Variables related to geological, metal content and material characteristics are quantified based on sparse drilling and sampling data. Since such programmes typically provide data on a relatively large scale, the levels of resolution is understandably at an order of magnitude greater than the scale required for modelling, prediction and risk assessment (Dowd and

Pardo-Iguzquiza, 2002).

Key sources of risk and uncertainty are:

(a) **Geological Uncertainty** - orebody uncertainty has been identified (Dimitrakopoulos, 1998) as one of the main contributors to overall uncertainty within mining projects . Due to the generally limited availability of drilling and sampling data in mining projects and even operations there is a high level of geological uncertainty around the qualitative (what does it contain?), quantitative (how much does it contain?) and positional (where is it located?) characteristics of mineral deposits geological. Morley et al. (1999) states four processes as contributors to resource and reserve uncertainty namely ore definition, geological interpretation, resource estimation and ore reserve estimation (mine planning).

Factors that complicate these processes are the physical extent of the deposit, location of geological boundaries and contacts, geo-technical features (e.g. faults, fractures and folds) and geo-metallurgical properties (e.g. material density and metal recovery). Dunham and Vann (2007) expand on the importance of including geo-metallurgical uncertainty around issues such as characterisation of metallurgical recovery and throughput.

(b) **Financial and Revenue factors** - another major contributing factor to the total uncertainty of a mining project is the commodity price and exchange rates. Depending on the type of commodity these are typically also highly uncertain. Metal prices are influenced by many contributing factors associated with supply (other producers) and demand (consumers). Prices are further influenced depending on whether commodities are sold based on long term contracts or spot market prices, seasonality in demand, level of substitution, demographical growth etc.

(c) **Operating Costs** - mining and processing operating costs (OPEX) are also uncertain since they are associated with uncertainty in seasonal and growth trends of inflation rates.

(d) **Capital Expenditure** - due to the complexity of the cost estimation process, as well as the long duration between the time of estimation and the time of expenditure, capital expenditure (CAPEX) is similarly uncertain.

(e) **Efficiency and Performance factors** - factors such as metal recovery, dilution, production and processing efficiency are uncertain since they are either tied to geological uncertainty or related to future performance of human resources and equipment.

(f) **Other** - more subjective risks such as political, country related, social and environmental is difficult to quantify but often ends up in some form built into the project specific discount rate as a risk premium.

The combined effect of all these levels of uncertainty is that geologists and mining engineers have a difficult time compiling evaluations and business cases for mineral projects. The fact that uncertainty in estimation and forecasting is prohibitive on the accuracy of assumptions made in feasibility studies is evident from confidence intervals that decision makers typically put around different levels of study. Research conducted by Hall (2007) indicates confidence intervals for various types of studies from ±30% for conceptual studies, typically ±20-25% for pre-feasibility studies and ±10-15% for feasibility studies. In addition to these confidence intervals, a range of contingencies are also typically added to the bottom-line expected cost of the estimate depending on the level of accuracy Bullock (2011).

Arguably the two most important areas of uncertainty influencing the accuracy and validity of mining case studies are those associated with geological (grade) and price uncertainty.

Within this thesis we only focus on the quantification of grade uncertainty but a lot of work has been done on various ways to incorporate price uncertainty into the mine planning and scheduling process. Some of the key methods used are Monte Carlo Simulation and Real Option Valuation - see for instance Cortazar et al. (2001), Samis et al. (2003), Nicholas et al. (2006), Cobb and Charnes (2007), Abdel Sabour et al. (2008), Akbari et al. (2008), Davis and Newman (2008), Akbari et al. (2009), Armstrong et al. (2009) and Sabour and Wood (2009).

3.3.2 Geological uncertainty

Geo-statistical estimations and block models are typically generated from sparse drilling data representing initial information about a mineral deposit. Such block models are then used as input to the open pit and schedule optimisation process. In reality, attributes of the actual deposit are only partially known and the properties of interest such as grades and ore material types are inferred.

As mentioned already before, numerous studies have shown that schedules based on optimal estimation methods such as kriging tend to underestimate or overestimate the real value of the ore body (Dimitrakopoulos et al., 2002) and this phenomenon has been interpreted and explained by the observation that kriging estimates tend to misrepresent the true variability of a deposit. As a result, kriging estimates produce 'smoothed' representations of the ore body which do not honour its spatial variability (Goovaerts (1999) and Chiles and Delfiner (2009)). In Figure 3.3 this phenomenon is illustrated by the fact that the kriged estimate in (a.) appears smoothed or 'averaged out' more so than the three simulated estimates in (b.), (c.) and (d.) which still retain their inherent variability.

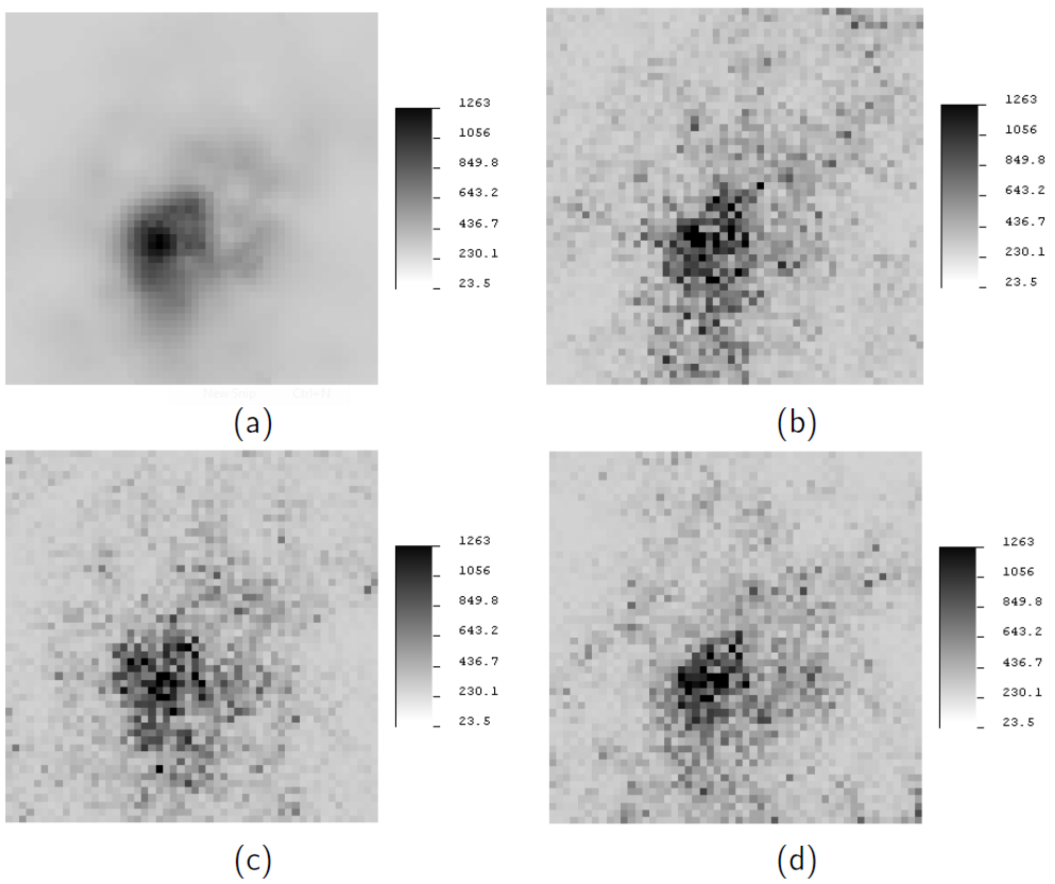


Figure 3.3: (a) Kriged estimate v.s. (b to d) Simulations - source: Saavedra Rosas (2009)

3.3.3 Conditional Simulation

Rendu (2002) highlights geo-statistical simulation as playing a crucial role in evaluating geological risk. The basic concepts include producing a best estimate model of a reserve by means of an optimal estimation method such as kriging and then developing a mine schedule using the standard procedure retaining the schedule for the base-case. A number of conditional simulations are generated for the deposit and the base case mine schedule is then tested against each alternative simulation model in turn, in this way producing a series of alternatives showing how actual production might vary if the base-case schedule was used.

These days, geo-statistical conditional simulation is a fairly standard and accepted technique for incorporating geological risk into block models with several researchers expanding on the methodology used (e.g. Ravenscroft (1992) and Dowd (1994)).

Conditional Simulation is a class of Monte Carlo techniques proposed by Halton (1970) to deal with ore body uncertainty in the generation of open pit schedule optimisation (see also Journel and Huijbregts (1978), Dimitrakopoulos (1990), Dimitrakopoulos (1994) and Armstrong and Dowd (1994)).

The majority of research and case studies applying conditional simulation to geological and grade variability has been focussed on open pit mining (Dimitrakopoulos (1998), Dimitrakopoulos et al. (2002), Dimitrakopoulos et al. (2007)).

There are several conditional simulation approaches. Parametric-simulation techniques assume that the data have a Gaussian distribution, so that a transform of the data typically is prerequisite. The steps of parametric simulation include a normal-score data transform from the z -space to the y -space, calculation of the variogram (covariance) on the y- normal scores, multiple simulations of the y-normal scores on a grid or within a volume and then a back-transform procedure from the simulated y-normal scores to the simulated z-values.

Some of the theoretically sound and practically tested conditional simulation approaches include the:

(a) Turning Bands simulation,

(b) Sequential simulation,

(c) Truncated Gaussian simulation; and

(d) Simulated Annealing simulation.

Of these methods, the Turning Bands and Sequential simulations methods are well known in the hard-rock (as opposed to petroleum) mining industry.

The Turning Bands method is one of the earliest simulation methods used and is based on first kriging the data and then generating unconditional simulations using a set of randomly distributed bands, or lines.

The Sequential simulation method uses the same basic algorithm to deal with different data types:

(a) Sequential Gaussian Simulation (SGS) simulates continuous variables, such as grade and metal content;

(b) Sequential Indicator Simulation (SIS) simulates discrete variables, using SGS methodology to create a grid of zeros and ones; and

(c) Bayesian Indicator Simulation (a newer form of SIS) allows direct integration of various attributes, and uses a combination of classification and indicator methods.

SGS is the method that we generally refer to when we talk about Conditional Simulation in this document.

The general procedure for creating sequential simulations includes the normal-score transformation of the raw data followed by a random selection of a data point not yet simulated. Through the use of a local conditional probability distribution function a new value is then simulated and added to the set of conditioning data, within a specified radius of the new target location. This process is then repeated until all grid nodes have a simulated value. The procedure leads to a 'random walk' sequence. For each simulation or realisation, a different random walk leads to a new and different result - see Figure 5.4.

A large number of models (simulations, interpretations or realisations) of the same deposit can be generated using Conditional Simulation. Each one of the generated models will be based on and conditioned to the same underlying actual data and statistical properties (histogram and variogram of the input data) in order to deal with the uncertainty related to the deposit and its attributes of interest. Conditionally simulated models each represent the same deposit and have the two properties that firstly they are constrained to exactly reproduce all available information (known data points), and secondly being equally probable (equi-probable) representations of the actual deposit. Realisations are all different from one another since the interpretation of values in between known data points are uncertain. At known data points however the values between simulations are the same. Individual realisations are therefore 'simulations' rather than 'estimates'. More elaborate explanations of Conditional Simulation is provided by Deutsch (2002).

The two broad approaches that have been proposed for using geo-statistical simulations to take account of uncertainty of the resources involves either generating several conditional simulations of the orebody and optimising the pit for each one (e.g. Dowd (1994), Van Brunt and Rossi (1997)) or generating the single pit contour using the kriged orebody model, and then running the economic analysis for each simulation through it (e.g. Dimitrakopoulos et al. (2002), Kent et al. (2007)).

Dimitrakopoulos (1998) outlines a general framework for modelling uncertainty and assessing geological risk, presenting currently used geo-statistical simulation algorithms.

Dimitrakopoulos and Ramazan (2004) presents a risk-based production scheduling formulation for complex, multi-element deposits. The formulation is based on expected block grades and probabilities of grades being above required cutoffs, both sets of values being derived from jointly simulated deposit models.

Froyland et al. (2004) uses a conditional simulation approach for valuing the trade-off between the cost of extra drilling and schedules of greater value that may be constructed from the resultant block models of greater accuracy.

Godoy and Dimitrakopoulos (2004) examines an optimisation approach based on the effective management of waste mining to maximise net present value (NPV) and in relation to the presence of grade uncertainty considering an economic model as the development of a combinatorial optimisation formulation that integrates multiple grade realisations of the deposit.

Menabde et al. (2004) describes a new mathematical algorithm for mine optimisation under orebody uncertainty which incorporates conditionally simulated orebody models. The software include a number of proprietary algorithms along with the commercially available mixed integer-programming software.

Li et al. (2004) presents an application of stochastic simulation in quantifying uncertainty in coal mining. It deals with resources classification and fault risk.

Gholamnejad and Osanloo (2007a) introduces a new algorithm that incorporates ore grade uncertainty during the push back design process. The suggested strategy tries to seek to schedule risky blocks later in the extraction sequence.

Journel (2007) talks about the illusion that as long as one uses the 'best' estimation algorithm based on quality data and sound geological interpretation, one would provide the best possible evaluation. He then expands on the dangers of local accuracy and global representation as the source of many arguments and severe prediction errors. He suggests simulating the essential components of a mining operation from such numerical models.

Albor Consuegra and Dimitrakopoulos (2009) describes a scheduling method using a simulated annealing (SA) algorithm and conditional simulation realisations used to generate production schedules that minimise the possibility of deviating from production targets.

3.3.4 Stochastic Programming

Stochastic programming (also called Stochastic Integer Programming) is a framework for modelling optimization problems that involve uncertainty. In stochastic programming models, it is assumed that probability distributions are available or can be estimated which define the variations and frequency of inputs. The objective is then to derive a solution that is feasible for all or most of the possible data instances and maximizes some expectation function. In some versions of this approach a decision is broken up into multiple stages during which the decision maker has recourse to make further decision to compensate for initial choices made.

Morales (2003) proposes stochastic programming models considering uncertainty for simultaneous open pit and undergrounds mines.

Askari-Nasab et al. (2007) describe a discrete stochastic simulation aimed at capturing the random field processes associated with open pit design and materials scheduling.

Gholamnejad and Osanloo (2007b) attempts to model long term production scheduling problems by chance constrained binary integer programming in a stochastic environment. This stochastic model is set up to account for ore block grade uncertainty. The model is solved using a genetic algorithm.

Leite and Dimitrakopoulos (2007) explores the practical intricacies and performance of a stochastic scheduling approach based on simulated annealing, in an application at a copper deposit with relatively low grade variability. Despite the relatively low grade variability of the deposit, the results of the study show that there are significant differences between the stochastic and the conventional schedules.

Ramazan and Dimitrakopoulos (2007) presents a new SIP mathematical model that generates optimum long-term production schedules for open pit mines for a defined objective function, considering the operational requirements at the mine. The SIP model takes multiple simulated orebody models, without averaging the grades, and maximises the total NPV when considering geological uncertainty caused by grade variability.

Boland et al. (2008) proposes introducing multiple geological estimates in a mixed integer multistage stochastic programming approach, in which decisions made in later time periods can depend on observations of the geological properties of the material mined in earlier periods.

Consuegra and Dimitrakopoulos (2010) introduces an approach that makes use of a SIP model to integrate geological uncertainty and economic discounting into the process of pushback design. The approach consists of open pit parameterisation in order to generate a set of nested pits, grouping the resulting nested pits into pushbacks based on a required number of pushbacks, and the use of a SIP to generate life-of-mine production schedules that maximise NPV, while meeting production targets and NPV forecasts, based on the pushback designs obtained in the previous stage.

3.3.5 Scenario Optimisation

Scenario Optimisation as proposed by Dembo (1992) introduces an elegant approach to take account of uncertainty in parameters of mathematical problems based on a particular technique of combining scenario outputs into one feasible solution. Most stochastic programming attempts in the literature tend to lead to intractable results or difficulty in interpretation, whereas it is fairly common practice for decision makers to solve deterministic models with different sets of assumptions or scenarios for the uncertain inputs.

As far as the authors are aware this approach has been applied to projects in hydro-thermal power generation as well as financial portfolio optimisation. No application has previously been made in the mining context.

Dembo (1992) presents a simple approach to solving a stochastic model, based on combining scenario solutions into a single, feasible policy. The approach is computationally simple and easy to understand. Because of its generality, it can handle multiple competing objectives, complex stochastic constraints and may be applied in contexts other than optimization.

In this thesis we will draw components from Dembo's method to develop our own scenario optimisation application (see Chapter 6 for further explanation).

Chapter 4

MILP formulation

4.1 Introduction

In order to evaluate various ways of incorporating grade uncertainty into the mine planning process, we first construct a deterministic MILP model to use as the basis for subsequent analysis.

The objective of this chapter is to summarise the formulation and assumptions used in a basic MILP model. This model is based on existing MILP formulation approaches and size reduction strategies.

Section 4.2 briefly describes key concepts in mathematical modelling relevant to our research.

Section 4.3 considers two easily implemented methods known from previous research to improve the MILP solve times in terms of either reducing the size of the data set or reducing solution accuracy requirements by modifying CPLEX solver options. Both these approaches are tested against small case studies and the results are noted for illustration.

Section 4.4 starts by describing the generalised data and input assumptions and then proceeds to define a MILP model formulation framework which will be used in subsequent chapters.

Section 4.5 describes a case study for an actual mine block model on which the MILP formulation is applied to find an optimum mine scheduling solution.

The chapter concludes with Section 4.6 giving a brief snapshot of the next chapter.

4.2 Preliminaries

4.2.1 Mathematical Modelling

Mathematical optimisation methods such as Linear Programming, Integer Linear Programming or Mixed Integer Linear Programming are used by many modern mine planning software packages and commercial optimisers such as CPLEX and Gurobi. The idea is to maximise or minimise the value of an objective function by finding input variables from within an allowed solution space (confined by the imposed constraints) and computing the value of the function.

Linear programming (LP, or linear optimisation) is a mathematical method for determining a way to achieve the best outcome of a linear objective function (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements represented as linear relationships. More formally, linear programming is a technique for the optimisation of a linear objective function, subject to linear equality and linear inequality constraints. The feasible region is a convex polyhedron and a linear programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

In LP problems, the decision variables and objective function terms are allowed to take on real values, whereas in Mixed Integer Linear Programs problems, at least some of the decision variables or terms in the objective function have to be integer values.

Linear programming relaxation methods entail replacing or 'relaxing' the requirement of certain variables to be integer values to more achievable real values. This relaxation technique transforms an integer programming problem, which is potentially computationally challenging, into a related linear programming problem that is solvable in a reasonable time frame.

The solution to the relaxed linear program can be used to gain information about the solution to the original integer program, and in fact the original linear relaxation solution serves as an initial upper bound in a maximization problem.

4.2.2 Branch-and-bound Algorithm

Branch-and-bound is a general algorithm for finding optimal solutions of various optimisation problems, especially in discrete and combinatorial optimisation. A branch-and-bound algorithm consists of a systematic evaluation of prospective candidate solutions, where fruitless candidates are discarded, by using upper and lower estimated bounds of the quantity being optimised (Land and Doig, 1960).

Using this divide-and-conquer method, the optimisation works its way through the solution space through the implementation of trimming and pruning, gradually improving the optimum solution. In a maximisation problem, the best integer solution found up to any point in time (the incumbent solution) is continually compared with the upper bound defined by the linear relaxation of the original problem. The difference between the current best solution and the upper bound is called the MIP (mixed integer program) gap (ILOG, 2012). The solution to the relaxed linear problem serves as a bound (upper if maximising, lower if minimising) in subsequent optimisations in conjunction with the branch-and-bound algorithm. Using the branch-and-bound method the best integer solution is progressively found and is continually compared with the predetermined upper bound.

The difference between the current best feasible integer solution and the upper bound (the MIP gap) can be expressed as a relative or absolute percentage or ratio. This percentage is by default set to a very low value (10^{-4} or 0.01%) in commercial optimisation software such as CPLEX (ILOG, 2012).

The MIP gap expresses the ratio between the solution quality of the integer program and of its relaxation. This is usually stated in relative or absolute terms. The MIP gap tolerance is the maximum permissible value for the MIP gap (ILOG, 2012).

The relative MIP gap tolerance is the most commonly used, whereas the absolute MIP gap tolerance is appropriate in cases where the expected optimal objective function is rather small in magnitude (ILOG, 2012).

The relative MIP gap is calculated by the following expression:

$$Relative\ MIP\ gap = \frac{|best\ node\ -\ best\ integer|}{|best\ integer|} \tag{4.1}$$

- *best node* = objective value of best non-integer node solution.
- *best integer* = objective value of best integer solution.

By monitoring the MIP gap, one can assess the value of the current best solution when compared with the upper bound and terminate the optimisation when the incumbent solution falls within the precision imposed by the MIP gap setting. This concept will be further explored in the next section.

4.3 Data Pre-processing and Optimisation Settings

Surprisingly, not much has been written about data reduction strategies for mine planning problems.

In preparing the data used in this thesis, we have adopted a data trimming approach similar to that described by Caccetta and Giannini (1985). They refer to the so-called 'trivial bound' which separates blocks unavailable to be mined due to pit slope requirements from those available to be considered for mining. We prefer to call this the 'geometric bound' and the process 'geometric trimming'. We illustrate this pre-processing step in this section by initially showing it in two dimensions and thereafter by extension into three dimensions.

4.3.1 Geometric (rectangular) Trimming

Consider a block model consisting of N squares in the horizontal (NH) and M squares in the vertical (MV) dimensions respectively (see Figure 4.1).

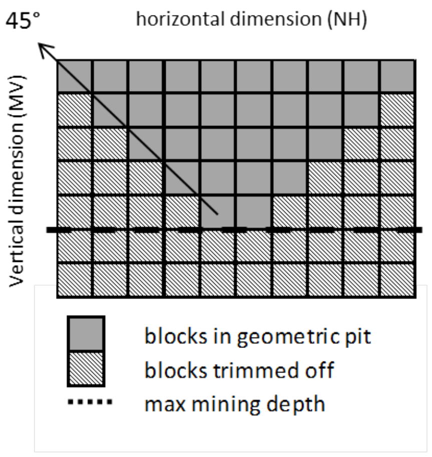


Figure 4.1: Block precedences in 2-D section

We will assume that the blocks from top to bottom in this two dimensional section represent the various layers in a three dimensional block model and that there exists certain conditions for removal of blocks. In this 2-D case we will assume that a block can only be removed if the three blocks in the level directly above it has already been removed.

Following this logic and if we are bounded by the sides of the model, it can be seen that certain blocks will be impossible to be removed due to the geometric configuration and block removal constraints of the model. It can also be seen from the figure that it is impossible to remove blocks from levels (depth) below five (max mining depth in Figure 4.1). It is clear that these obsolete blocks can be trimmed from the model upfront which will have benefits in terms of model size and solution times.

An algorithm that trims off obsolete blocks during a pre-processing stage will systematically remove an increasing number of blocks off the sides of each layer as it progresses from the topmost layers downwards.

This configuration can easily be extended into a three dimensional context by changing the horizontal dimension, referred to in the two dimensional case, to x and y axes and replacing the vertical dimension with the z axis.

The above can easily be extended to a three dimensional case with a systematic trimming of blocks off the sides as the levels go deeper until a level is reached where no further blocks are retained (i.e. all blocks are trimmed away on that level).

For all subsequent (deeper) layers all blocks will be removed from the block model. This pre-processing data removal step can lead to a significant reduction in the number of blocks considered during the optimisation.

Figure 4.2 illustrates the rectangular trimming of a small model originally containing 10 x 10 x 10 blocks in the x, y and z dimensions respectively (i.e. 1000 blocks in total).

It is clear from the example that only five levels will remain in the modified block model. All levels deeper than level five will be trimmed away.

Algorithm 4.1 systematically considers blocks in layers starting at the top layer (highest z value) moving downwards and trimming blocks off the edges. Z_n is the maximum number of blocks in the Z dimension and L ranges from 0 to Z_n.

Algorithm 4.1 ***Rectangular trimming***

```
Required: $X_{curr}$, $Y_{curr}$, $X_n$, $Y_n$, $Z_n$, Current level ($L$)
Objective: Derive rectangularly trimmed subset of original model
Start with original block model
for all blocks in model and level ($L$) do
    if $(Z_n\text{-}L)> X_{curr} < (X_n \text{ - } (Z_n\text{-}L))$ and $Z_n\text{-}L > Y_{curr} < (Y_n \text{ - } (Z_n\text{-}L))$ then
        include data for level ($L$) in trimmed model
    end if
end for
```

By applying the algorithm, blocks are trimmed based on a simplistic inter-

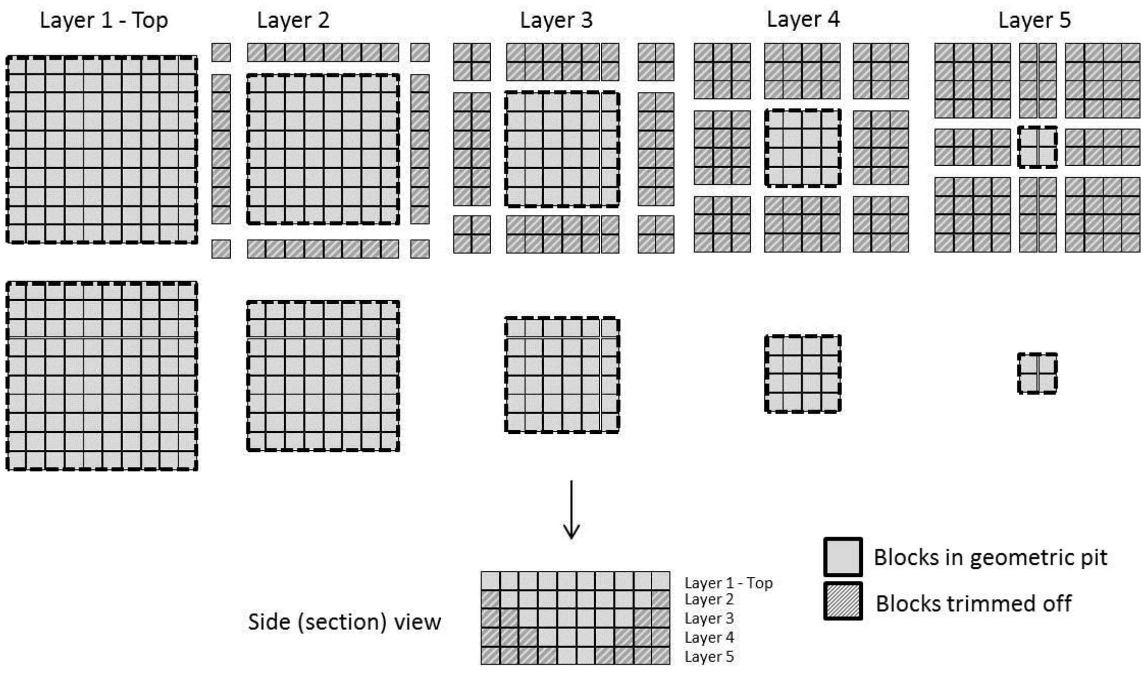


Figure 4.2: Edge trimming by level

pretation of the constraints imposed by precedence relationships between blocks (45 degree wall slopes). All blocks in the top-most layer remains. The next layer down, a row and column along each edge is removed, followed by two rows and two columns in the layer one further down, and so on continuing downwards.

Case Study

In order to demonstrate the impact of such a simplistic top-down rectangular trimming exercise we use a case study which has a total of 11,200 blocks in the original block model. Rectangular trimming reduces the number of blocks to only 3,740, a reduction of 67% (see Table 4.1).

Ultimate Pit Trimming

Even though rectangular trimming removes a large number of blocks from the model that, due to their geometric constraints, cannot possibly be in the final schedule (and can therefore be removed without affecting the solution), many blocks still remain that are not economically viable at assumed prices. The Ultimate Pit Optimisation method is a linear programming method (i.e not a MILP) that considers the value contained in each block under the simplified

Table 4.1: Rectangular trimming per level

Model	Blocks (X)	Blocks (Y)	Levels	Blocks in level
ORIGINAL	20	40	14	11,200
TRIM L1 (top)	20	40	1	800
TRIM L2	18	38	1	684
TRIM L3	16	36	1	576
TRIM L4	14	34	1	476
TRIM L5	12	32	1	384
TRIM L6	10	30	1	300
TRIM L7	8	28	1	224
TRIM L8	6	26	1	156
TRIM L9	4	24	1	96
TRIM L10 (bottom)	2	22	1	44
TRIM TOTAL				3,740

(naive) assumption that all blocks are mined in the first period (i.e. without period discounts applied) and without annual constraints on mining capacity. This additional trimming exercise can further reduce the size of the model in preparation for the schedule optimisation.

Caccetta and Hill (2003) proved that only blocks within the Ultimate Pit Limits can be in the optimal solution to the scheduling problem (under the same assumptions applied) and therefore by utilising this approach as a trimming exercise there is no risk of accidentally removing blocks that might have been feasible for mining.

It could be argued, why not just use the Ultimate Pit Limit approach directly without the rectangular trimming exercise. One argument in favour of first using rectangular trimming is the fact that rectangular trimming does not require mathematical modelling or solver software and can be readily applied with very basic software as a pre-process. This is especially useful in large models where the geometric dimensions and pit slope requirements could lead to a large reduction in blocks through trimming.

Figure 4.3 depicts various views of the ultimate pit trimmed model as developed in Section 4.5.

The top view shows that the ultimate pit solution has removed a large amount of low grade or waste blocks in the eastern side (right-hand side of the figure) of the deposit. The section view similarly shows how waste blocks have been

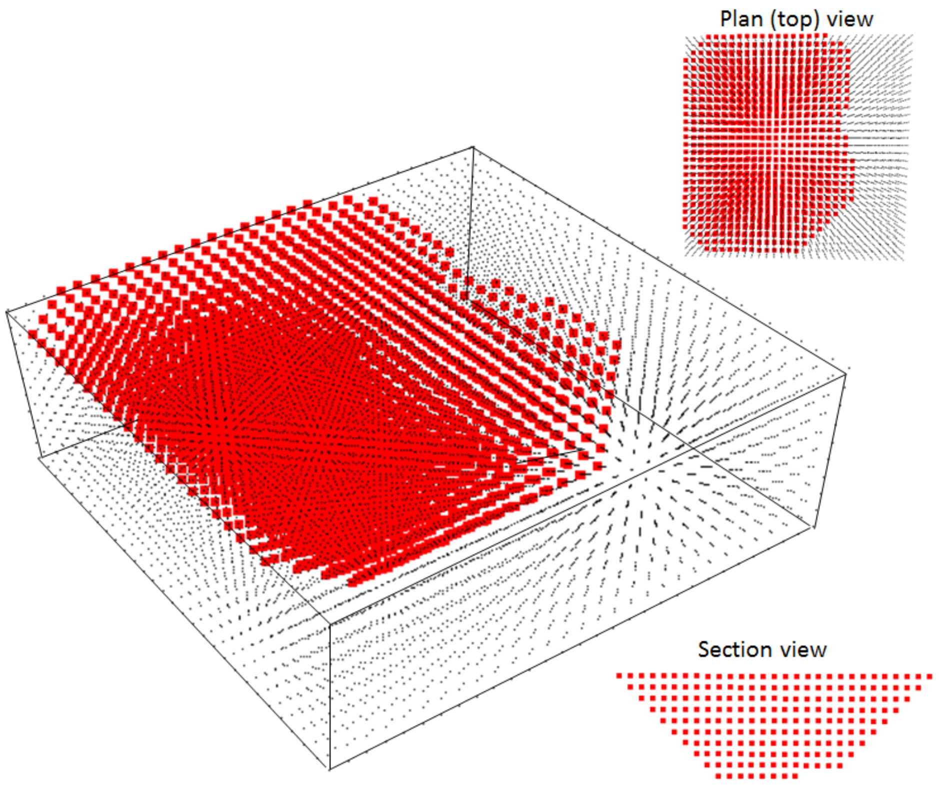


Figure 4.3: Case study - optimised Ultimate Pit

removed from the bottom of the pit.

Based on the findings from the pre-processing of data in the form of rectangular trimming and the resultant data reductions of up to 60% and more we decided to implement this trimming algorithm as part of our standard data preprocessing stage in this thesis.

In addition, due to the further data reduction benefits in using he Ultimate Pit optimisation as an additional data trimming phase, we similarly introduce the ultimate pit optimisation on the trimmed data model as a second stage of data pre-processing within our methodology.

Since this was not a key objective of this thesis we did not test the actual solution time benefits to be gained by using these data trimming steps.

4.3.2 Realisitic MIP gap setting

In the application of LP relaxation as described in Section 4.2, the MIP gap and its tolerance are important concepts allowing a predetermined termination criterion that the user can set explicitly in the CPLEX software ILOG (2012).

Klotz and Newman (2013) discuss the setting of a realistic MIP gap as one of the easy and quick win strategies that a user can employ and that large improvements can be gained in optimiser performance on MILP problems in software such as CLPEX.

When the value of the MIP gap falls below the pre-set tolerance value of this parameter, the optimisation procedure is terminated and the current best solution (the incumbent) provides the final solution. For example, to instruct CPLEX to stop as soon as it has found a feasible integer solution proved to be within five percent of optimal, the relative MIP gap tolerance is set to 0.05 (5%). In CPLEX, the relative MIP gap is set to 10^{-4} or 0.01% by default ILOG (2012).

The absolute MIP gap sets an absolute tolerance on the gap between the best integer objective and the objective of the best node remaining but is usually more relevant for small objective function values. In CPLEX, the absolute MIP gap can take on any non-negative number but is set to 10^{-6} by default ILOG (2012).

The MIP gap affords the modeler an opportunity to accept a solution which is not certain to be optimal but sufficient to be closely representative. For example, a relative MIP gap of 1% (or 0.01) implies that the solution is at most one percent from the optimum.

CPLEX states that on a difficult model with input data obtained with only approximate accuracy and where a proved optimum is thought to be unlikely within a reasonable amount of computation time, a user might choose a larger relative MIP gap to allow early termination; for example, a relative MIP gap of 5%. Conversely, in a model where the objective function amounts to billions of dollars and the data are accurate to a degree that further processing is worthwhile, a tighter relative MIP Gap (even up to 0%) may be advantageous to avoid any chance of missing the best possible solution.

The practical implication of the MIP gap tolerance is that the optimisation algorithm stops when an integer solution is guaranteed to be within this percentage (relative MIP gap) or amount (absolute MIP gap) from the true optimal solution.

By monitoring the MIP gap, one can assess the quality of the current best solution when compared with solutions derived from relaxing the integrality constraints and using the branch-and-bound algorithm.

Case Study

In order to illustrate the effect of the MIP gap on solution times, two small case studies were tested. The models used in this section are proxies for real cases and aim to illustrate the impact of the MIP gap setting as a concept rather than attempting to be of a practical size.

The individual blocks in both models are 30 m cubes and the pit slopes from adhering to precedence restrictions are all 45 degrees. Both models are reduced in size using the rectangular trimming pre-process described earlier in this section.

Both of the case studies are allowed to solve to a MIP gap setting of 10^{-3} (0.001 or 0.1%) or are terminated after 5,000 seconds (20 minutes). Note that at the default MIP gap setting of 10^{-3} a run will only terminate successfully (i.e. 'solve') when the MIP gap has been sufficiently reduced so that the difference between the current best integer solution and the linear programming solution is less than 0.1%.

The technical assumptions for the case studies are summarised in Table 4.2.

Table 4.2: Technical assumptions for MIP gap case studies

Assumption	**Value**
CASE STUDY 1	
Blocks (original)	1,350 (15x15x6)
Blocks (after rectangular trimming)	680
Periods (years)	6
Maximum Processing Capacity (tonnes per year)	1,000,000
CASE STUDY 2	
Blocks (original)	13,500 (30x30x15)
Blocks (after rectangular trimming)	4,960
Periods (years)	8
Maximum Processing Capacity (tonnes per year)	5,000,000

Refer to Equation 4.1 (Page 45) for the formula of the MIP gap calculation.

Figure 4.4 summarises the progression of improved integer solutions (NPV in US$) for Case Study 1 as the optimisation is left to run its course.

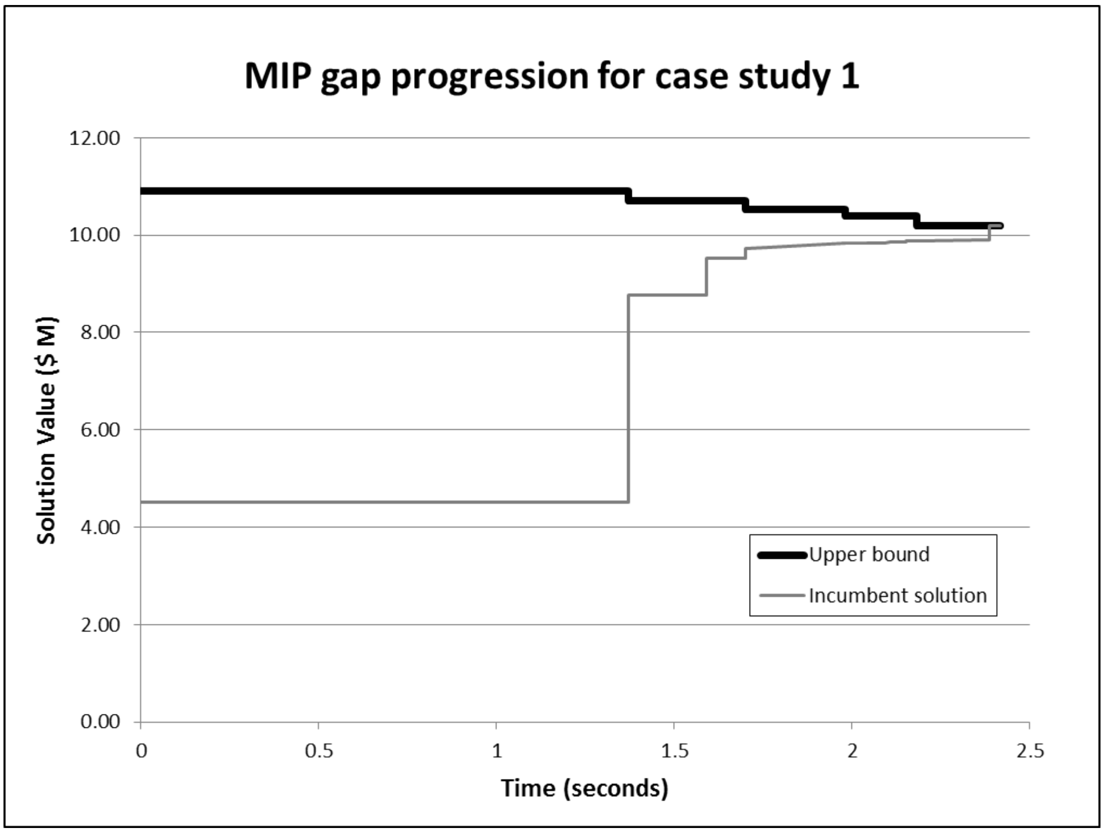


Figure 4.4: Solution progression for case study 1

(a) The original relaxed LP solution sets off as the upper bound.

(b) The optimisation rapidly (after 1.4 seconds) finds an initial integer solution within 18% of the upper bound.

(c) Improved solutions are found as indicated on the graph.

(d) Case study 1 solves in less than 2.5 seconds after finding an improving solution which reduces the MIP gap to less than the target tolerance (0.1%).

Figure 4.5 summarises the progression of improved integer solutions (NPV in US$) for Case Study 2 as the optimisation is left to run its course.

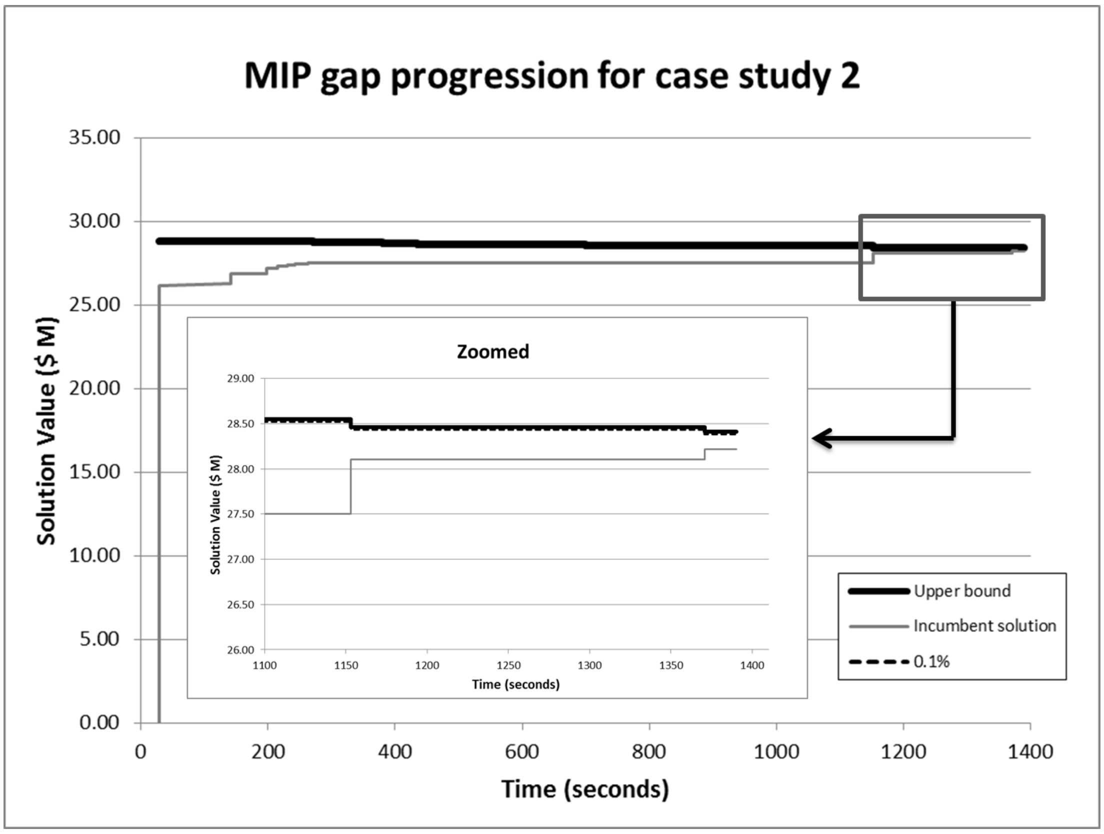


Figure 4.5: Solution progression for case study 2

(a) The original relaxed LP solution sets off as the upper bound.

(b) The optimisation finds an initial integer solution within 28% of the upper bound after 55 seconds.

(c) Improved solutions are found as indicated on the graph:

- During the computational time interval of 100 to 200 seconds the MIP gap is reduced to less than 5%
- The next 90 seconds are spent trying to find an improved solution which is found at around 1150 seconds reducing the MIP gap further to 4.3%
- At around 1380 seconds an improved solution is found reducing the MIP gap to 2% after which the optimisation remains at this gap until it times out after 5,000 seconds (20 minutes)

Conclusions from MIP gap analysis

In order to assess the impact of the MIP gap setting on optimisation performance and hence potential inclusion into our methodology, we tested the run times for two case studies of different size compared to their respective reductions in MIP gap (target was 0.1% precision). The objective was to determine the *trade-off* between solution quality (value) and additional solution time.

The results illustrated the taxing impact of the very tight default MIP gap setting on the solution time of optimisations. In all cases the optimisation achieved solutions which were well within the levels of tolerance typically acceptable by the mining industry (5%-10% of value for feasibility level estimates) in a reasonable time frame. However, due to the excessive MIP gap requirements (0.01% of value) imposed on mining models by the optimisation software, unrealistic and unnecessary improvements to the solution are sought at the price of significant additional solution time.

Although the MIP gap setting is readily accessible to seasoned users of CPLEX (mathematicians and researchers), it remains a fairly obscure concept to mine planers in general, often embedded within or external to the mine planning software. The MIP gap is most often too tightly set by default and that the resultant additional solve time required to improve the incumbent solution is overly burdensome on the optimisation process for the type of solutions needed in a mining context. Some optimisation runs which appear to fail due to non-compliance with the unreasonably tight MIP gap setting (such as our second case study) often contain incumbent solutions which are in fact quite acceptable and implementable in reality.

A very tight MIP gap in the order of 10^{-6} to 10^{-2} (0.00001% to 1%) adds unnecessary precision to problems which often contain a low inherent confidence on the accuracy of the input parameters (grade and price uncertainty). This is most often due to:

(a) limited representative information (drilling defining grade and geology) and forecasting (prices and costs) ability, and

(b) due to much lower acceptable levels of accuracy that decision makers assume for scoping (20-50%), pre-feasibility (10-20%) and feasibility (5-10%) studies.

Based on our findings we set the MIP gap to 5% in all subsequent models in this thesis.

4.4 MILP Model Formulation

4.4.1 Hardware and Software Considerations

The optimisations and processing of results for our research were conducted on an Intel Core i5-2500 (3.30GHz) processor with 8.00GB RAM and a 64-bit operating system.

Optimisations runs were performed using IBM ILOG's CPLEX Optimization Studio (CPLEX) software. CPLEX solves integer programming problems and very large linear programming problems using either primal or dual variants of the simplex method or the barrier interior point method (ILOG, 2012).

All pre-processing was done using Python 2.7.5 and interfacing between Python and CPLEX was established using the Python Pyomo library. Schedule assessment and post-processing was done in Python and Excel. Pyomo draws from the capabilities of the Coopr software library, which integrates Python packages for defining optimizers, modelling optimization applications, and managing computational experiments. Coopr is a collection of open-source optimization-related Python packages that supports a diverse set of optimization capabilities for formulating and analysing optimization models (SandiaLabs, 2014).

CPLEX default settings were used for all optimisations except for the MIP gap. The MIP gap for the ultimate pit optimisations was set to 1% whereas the MIP gap for schedule optimisations was set to 5%. Beyond that, CPLEX was allowed to follow its own strategies for solving optimisation problems.

4.4.2 Data

Block Model

Data used in optimisation is obtained from a block model which is a partitioning of rock material contained in a geological or ore model into same-sized rectangular cubes. Such a model is the result of geologists having preformed various interpretations and extrapolation on drilling and sampling data and assay values.

Each cube has location variables which provide it with a definitive position in x, y and z dimensional space. In addition to the location, each cube has a number of attributes defining it. These often include a density or tonnage value dependant on the particular rock type to which it belongs, and a quality or value indicator typically expressed as a grade or percentage of a potentially valuable metal or mineral (e.g. 2% Cu or parts per million Au).

The revenue value of the block is determined by considering the contained metal grade and applying an assumed market value (commodity price) to the ore in the block. Often, an additional recovery percentage is also applied as a function of the rock type or density. The profit value considers the block revenue value minus the costs of removing and processing the material in the block.

Coordinate Conversion

A key attribute of a block is its position in space. This is typically expressed as a set of three coordinates corresponding to the block's position (or more accurately the block's centroid) in three-dimensional Cartesian space. Visualisation software requires the expression of information in terms of three dimensions in order to generate visualisations.

In this study, the block model is assumed to be rectangular without surface topography or irregular extensions in the x, y or z dimensions. The case study used in the following sections is based on a rectangular block model with exactly 20, 40 and 14 blocks (cubes of 30 x 30 x 30 m) in the x, y and z dimensions respectively, resulting in a total of 11,200 blocks. Block coordinates in the x dimension increase from left to right, whereas block coordinates in the y dimension increase from front to back. Block coordinates in the z dimension increase from the bottom to the top of the model (see Figure 4.6).

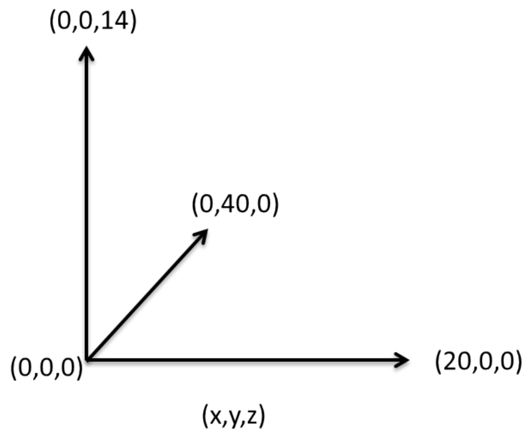


Figure 4.6: Increase in X, Y and Z from origin (0,0,0)

As part of the pre-processing stage, the block model coordinates are first converted into unit values. That means the following: the block model originally contains unique coordinates representing the physical location of the deposit with its blocks in 3-dimensional space. However, for ease of calculations during modelling, coordinates are converted into unit values starting with a minimum of zero for each of the dimensions, and a maximum of one less than the number of blocks in each of the three dimensions. Such conversion is a function of the maximum coordinate values in the x, y and z dimensions (X_{max}, Y_{max}, Z_{max}), the blocks sizes in the various dimensions (X_{dim}, Y_{dim}, Z_{dim}) and the current coordinate values (X_{curr},Y_{curr},Z_{curr}) - see Algorithm 4.2.

Algorithm 4.2 ***Convert to unit coordinates***

```
**Required**: $X_{max}$, $Y_{max}$, $Z_{max}$, $X_{dim}$, $Y_{dim}$, $Z_{dim}$, $X_{curr}$, $Y_{curr}$, $Z_{curr}$
**Objective**: Convert coordinates to unit values
Start with original block model
**for all** x,y and z coords in original model **do**
    $X_{new}$ = int(($X_{max}$ - $X_{curr}$) / $X_{dim}$)
    $Y_{new}$ = int(($Y_{max}$ - $Y_{curr}$) / $Y_{dim}$)
    $Z_{new}$ = int(($Z_{max}$ - $Z_{curr}$) / $Z_{dim}$)
**end for**
```

An example of original coordinates converted to unit coordinates can be seen in Table 4.3. The example contains 10 blocks extracted from a 10 x 10 x 10 block model (i.e. 1000 blocks). All the blocks are cubes of dimensions 20m x 20m x 20m.

Table 4.3: Example conversion of original to unit coordinates

	ORIGINAL			**UNIT**		
	X	Y	Z	X	Y	Z
1	20,200	35,210	220	9	9	9
2	20,220	35,230	240	8	8	8
3	20,240	35,250	260	7	7	7
4	20,260	35,270	280	6	6	6
5	20,280	35,280	300	5	5	5
6	20,300	35,310	320	4	4	4
7	20,320	35,330	340	3	3	3
8	20,340	35,350	360	2	2	2
9	20,360	35,370	380	1	1	1
10	20,380	35,390	400	0	0	0
Max	**20,380**	**35,390**	**400**			

In addition to the conversion to unit values, it is often more convenient in mathematical model formulations to deal with a single location identifier (BlockID) instead of three variables associated with a block. It is thus common to convert block coordinates to a single location identifier while doing modelling and optimisation and then back transforming the optimisation results into the original three-dimensional coordinates for relating results to their true physical locations.

Within this thesis, three dimensional coordinates related to individual blocks in a block model are converted to a single location identifier per block. Such conversion is a function of the current coordinates (X_{curr},Y_{curr},Z_{curr}), the number of blocks in the x, y and z dimensions (X_n,Y_n,Z_n) as well as the block dimensions (X_{dim}, Y_{dim}, Z_{dim}).

The algorithms used to respectively convert block locations from x-y-z to single block location format and back to x-y-z formate are shown in Algorithm 4.3 and Algorithm 4.4.

Algorithm 4.3 ***Converting x,y,z coordinates to single block location***

```
**Required**: $X_{curr}$, $Y_{curr}$, $Z_{curr}$, $X_n$, $Y_n$, $Z_n$
**Objective**: Convert block coordinates to single block location variable
Start with original block model
**for all** blocks in model **do**
    BlockID = ($Z_{curr}$ $X_n$ $Y_n$)+($Y_{curr}$ $X_n$) + $X_{curr}$
**end for**
```

Algorithm 4.4 ***Converting single block location back to x,y,z coordinates***

```
**Required**: BlockID, blocksInX ($X_n$), blocksInY ($Y_n$), blocksInZ ($Z_n$)
**Objective**: Convert single block location variable back to block coordinates
Start with original block model
**for all** blocks in model **do**
    $Z_{curr}$ = int(BlockID/($X_n$ $Y_n$))
    $Y_{curr}$ = int((BlockID - ($Z_{curr}$ $X_n$ $Y_n$))/$X_n$)
    $X_{curr}$ = BlockID - ($Z_{curr}$ $X_n$ $Y_n$) - ($Y_{curr}$ $X_n$)
**end for**
```

It is worth noting that after the optimisation results have been generated, it is common to convert the block coordinates back to the original true coordinates in order to enable visualisation or location in space.

4.4.3 Assumptions

Block Values

Block values are pre-calculated before the data is submitted for optimisation. A block is subjected to a profitability test during which it is determined whether the block contains sufficient valuable material (grade) to pay for both its mining and treatment, in which case it is considered ore and allocated to the treatment plant. If a block is assessed and found to contain no or insufficient ore to pay for its mining and treatment, then it is categorised as waste and destined to be discarded if mined in the process of exposing more valuable ore.

Algorithm 4.5 summarises the procedure used to determine whether a block is to be categorised as ore or waste.

Algorithm 4.5 ***Profitability test***

```
Required: SalesPrice ($P$), BlockGrade ($B_{gr}$), BlockTonnage $B_t$ (Bt), BlockRecovery ($B_{rec}$), UnitMiningCost ($C_{mu}$), UnitProcessingCost ($C_{pu}$), BlockValue ($B_{val}$)
Objective: Test whether a block should be categorised as ore or waste
Start with original block model
for all blocks in model do
    if $(P\ B_{gr}\ B_t\ B_{rec})$-$((C_{mu} + C_{pu})\ B_t)) > 0$ then
        $B_{val} = (P\ B_{gr}\ B_t\ B_{rec})$-$((C_{mu} + C_{pu})\ B_t))$
    else
        $B_{val} = -(C_{mu})\ B_t$
    end if
end for
```

No partial blocks are used i.e. blocks are either 100% ore or 100% waste.

Pit Slope and Precedence Constraints

In the models investigated a number of simplifying assumptions have been made regarding the sizes and shapes of individual blocks and the block model in general:

(a) all blocks are equally sized cubes and therefore the three dimensions are the same (in all cases the block dimensions used are $30m$ x $30m$ x $30m$).

(b) pit slopes of 45 degrees are assumed in all directions.

(c) block models are rectangular i.e. the same number of blocks in each of the 3 dimensions respectively.

(d) flat on top, as surface topography has been removed to simplify the modelling.

Simplistic methods commonly used by researchers in considering the slope requirements imposed on target blocks is to ensure that either the nine or five blocks that form a rectangle or plus (+) sign respectively above the block in question (see Figure 4.7) have already been previously removed before a block is considered for extraction.

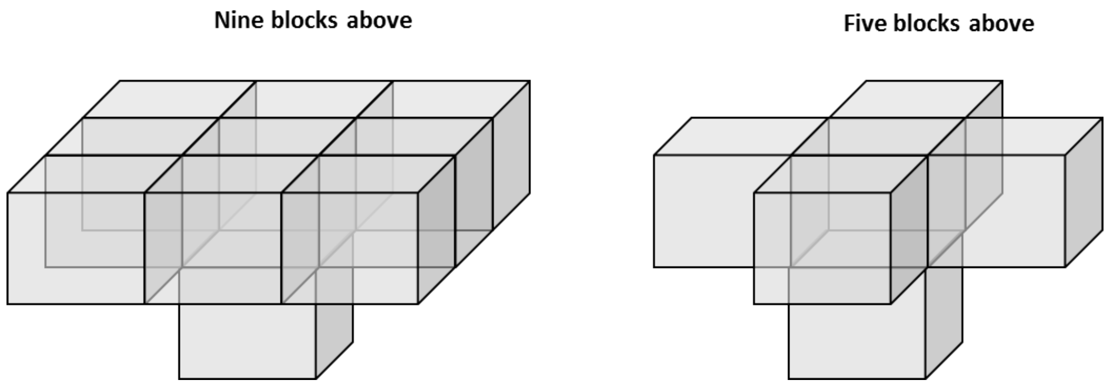


Figure 4.7: Block precedences

However, the assumptions made in this thesis in relation to the extraction of blocks are an expansion of the *knight's move* minimum search pattern idea utilised by Caccetta and Giannini (1988) :

(a) the five blocks located (in a plus sign) in the layer directly above the focus block need to have been extracted before the focus block can be accessed (see Figure 4.8)

(b) in addition, a further eight blocks located two layers directly above the focus block require to have been extracted before the focus block can be accessed (see Figure 4.8).

The objective and benefit of using this configuration is that it better enables maintaining the required pit slope angles in all directions. Schedules and extraction sequences based on the nine or five block methods described above often leads to pits with an unnatural rectangular or diamond shape, whereas the knights move approach leads to more contoured outlines better honouring pit slopes in all directions.

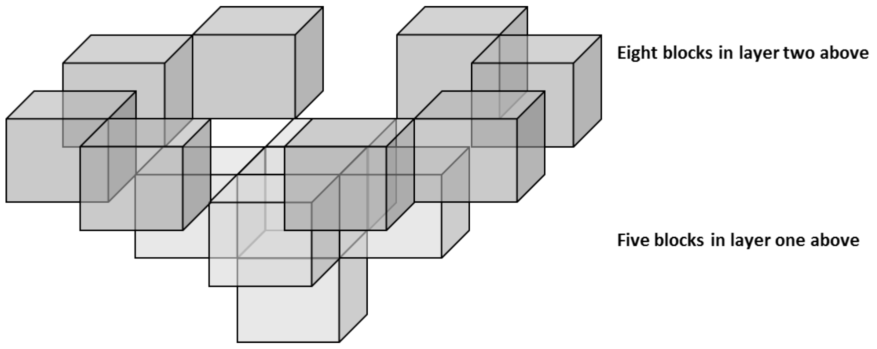

Eight blocks in layer two above
Five blocks in layer one above

Figure 4.8: Block precedences using 'knight's move' pattern

4.4.4 Variables, Constraints and Objective Function

Decision Variables

All decision variables in our simplistic formulation are binary integer and related to the decision of whether or not to remove a specific block in (at) a specific time period (indicated as x in the formulation to follow).

Constraints

Three types of constraints are used in this thesis. The first type of constraint has to do with the physical requirements imposed in order to maintain safe pit slopes. The overall pit slope angle is the angle measured from the bottom bench to the top bench of an open pit. It is the angle at which the wall of the open pit can be safely maintained and is determined by rock strength, geological structure and water conditions. Within this thesis this angle is assumed to be 45 degrees as explained in the previous section. From a practical point of view this requires the definition of a set of blocks related to each block, which requires to be removed before a block can be mined.

The second type of constraint relates to the fact that there is a limitation on the available mining and processing capacity in each time period (indicated as MMC and MPC in the formulation below). These constraints are dependent on the number of trucks and excavators, as well as the physical limitations of the processing plant. In our formulation we put an upper (maximum) limit on this capacity but no lower limit. This means in effect that the solver can choose to mine zero tonnage if that leads to a more feasible solution.

The final type of constraint forces the decision variable to be binary. This means the variable can only take on the values of 0 if a blocks is not to be mined or 1 if a block is to be mined in a particular time period. It is worth noting that this constraint is relaxed during the optimisation process in order for CPLEX to use a branch-and-bound strategy to solve the problem.

Objective Function

In mining scheduling problems a number of approaches can be followed when it comes to optimising the model.

In some cases the objective might be to minimise the sum of discounted future costs. A typical mining example of this might be if the problem under review is a waste removal (or pre-stripping) project. In such a case there might be no revenues anticipated from the exercise and therefore the sole objective would be to execute the project at the lowest possible cost subject to other potential constraints and limitations.

In other cases, the objective might be to maximise the volume produced given the limitations imposed by capacity constraints. The most common objective, however, is to maximise the net present value derived from future expected cash flows ensuing from the profitable mining of valuable material and associated (unavoidable) waste. In our model we will use this approach i.e. maximising the net present value.

4.4.5 Formulation

The model used in this thesis is a directed graph D = (V, A) where V represents the blocks (vertices) and A contains precedence relationships (arcs). The approach is equivalent in form to that proposed by Caccetta and Hill (2003) and Gaupp (2008) and takes on the following general form.

Notation

- The decision variable (x) is of the form:

$$x^{b,t} = \begin{cases} 1 & \text{if block } b \text{ is extracted at period } t \\ 0 & \text{otherwise} \end{cases}$$

- The number of time periods $t \in \{1, \ldots, T\}$
- Mining blocks $b \in \{1, \ldots, B\}$
- Value of individual blocks (v^b) $\forall\ b \in \{1, \ldots, B\}$

- Discount factor at time t (d^t) $\forall\ t \in \{1, \ldots, T\}$
- Tonnes of material in block b (MT^b) $\forall\ b \in \{1, \ldots, B\}$
- Maximum annual mining capacity (MMC), assumed to be the same for every period
- Maximum annual processing capacity (MPC), assumed to be the same for every period
- Tonnes of material in block p destined for processing (PT^p)

Optimisation Model

For the problem under consideration we will use the following objective function and constraints similar to that proposed by Caccetta and Hill (2003) :

$$(\mathcal{B}) \ \max \ \ Z = \sum_{t=1}^{T} \sum_{b=1}^{B} x^{b,t} \cdot v^b \cdot d^t \tag{4.2}$$

s.t.

$$\sum_{t=1}^{T} x^{b,t} \leq 1 \qquad \forall b \in \{1, \ldots, B\} \tag{4.3}$$

$$\sum_{b=1}^{B} x^{b,t} \cdot MT^b \leq MMC \qquad \forall t \in \{1, \ldots, T\} \tag{4.4}$$

$$\sum_{t=1}^{T} x^{\tilde{b},t} \cdot PT^p \leq MPC \qquad \forall \tilde{b} \in \{1, \ldots, B\} \tag{4.5}$$

$$\sum_{t=1}^{k} x^{p,t} \leq \sum_{t=1}^{k} x^{j,t} \qquad \forall (p,j) \in E, \forall k \in \{1, \ldots, T\} \tag{4.6}$$

$$x^{b,t} \in \{0,1\} \qquad \forall b \in \{1, \ldots, B\}, \forall t \in \{1, \ldots, T\} \tag{4.7}$$

With respect to the formulation above constraint (4.3) ensures that the each block is only extracted once or not at all. Constraint (4.4) ensures that total blocks mined per period does not exceed mining capacity for that period. Constraint (4.5) ensures that total blocks designated for processing per period does not exceed processing capacity for that period. Constraint (4.6) is the wall slope restriction which ensures that a block cannot be extracted before a predetermined set of blocks is entirely removed in a previous period, or previously in the same period. Finaly, constraint (4.7) ensures that the decision variable only takes on binary integer values.

4.5 Case Study

4.5.1 Background

We have tested our deterministic formulation on a publicly accessible realistic model called 'Marvin' which a number of researchers have used for case studies and benchmarking over the years. The data was obtained from a library of publicly accessible test problem instances made available by a research group associated with universities in Chile and USA (Espinoza et al., 2012).

Since the original model had blocks in the upper layers missing due to previous surface excavation and mining we decided to remove all the blocks in the top five layers. Our resultant rectangular deposit model consists of 30 blocks by 30 blocks in the x and y dimensions (i.e. 900 blocks per layer), and 10 layers in the z dimension leading to 9,000 blocks in total. All blocks are 30 m cubes. Block densities (and therefore tonnages) depend on the particular rock types contained in a block. Block densities in the model range from a low of 2.27 g/cm^3 to a high of 2.75 g/cm^3, corresponding to tonnages between 61,200 and 74,250 respectively.

Figure 4.9 shows a diagonal top-down view of the deposit. Blocks have been coloured based on their Cu content (%). Zero and very low grade values are greyed out. The legend shows the colours related to different grade ranges from lower grades (green) containing around 0.6 to 0.7% Cu. The grades increase along the color spectrum up to the warmer colors with the highest grades being around 1.5% Cu (pink).

Figure 4.10 is a cumulative frequency curve showing statistics related to the grade distribution in the deposit. Approximately half of the deposit contains zero grade values with around 45% of the deposit consisting of grades in the range between 0 and 0.9% Cu, and around 5% of the values lying in the high grade range between 0.9 and 1.5% Cu.

4.5.2 Assumptions

The assumptions and parameters used in the optimisation are summarised in Table 4.4.

It is worth noting here that due to the upper limit on mining tonnage which can be produced per period (assumed to be annual periods), the maximum number of blocks that can be mined in a single period is between a theoretical 67 and 81 with an average of 74 depending on the mix of rock types and their respective densities in the annual production schedule.

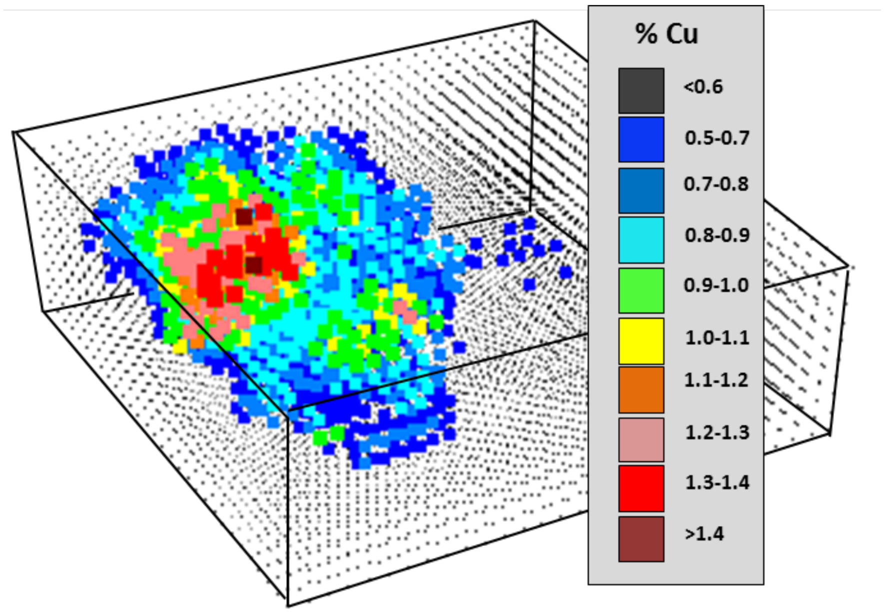


Figure 4.9: Perspective view of Marvin deposit

Table 4.4: Optimisation parameters used for Marvin case study

Parameter	**Value**
Upper limit to production capacity (tons)	5,000,000
Sales price ($ per t Cu)	6,500
Mining operating cost ($ per ton mined)	8
Processing operating cost ($ per ton processed)	14
Maximum number of periods (years)	8
Discount rate (%)	10

Also, because we have put an upper limit on the number periods allowed (8 years), the maximum tonnage that can be mined overall is around 40 million. Similarly, this means that only around 600 blocks can be mined in this particular formulation of the problem.

The implementation code for the MILP formulation can be found in the Appendix under Code Snippet A.

4.5.3 Results

The optimiser finds an optimum solution for the Marvin case study of $946,423,526 over eight periods. The schedule related to the optimum solution can be seen in

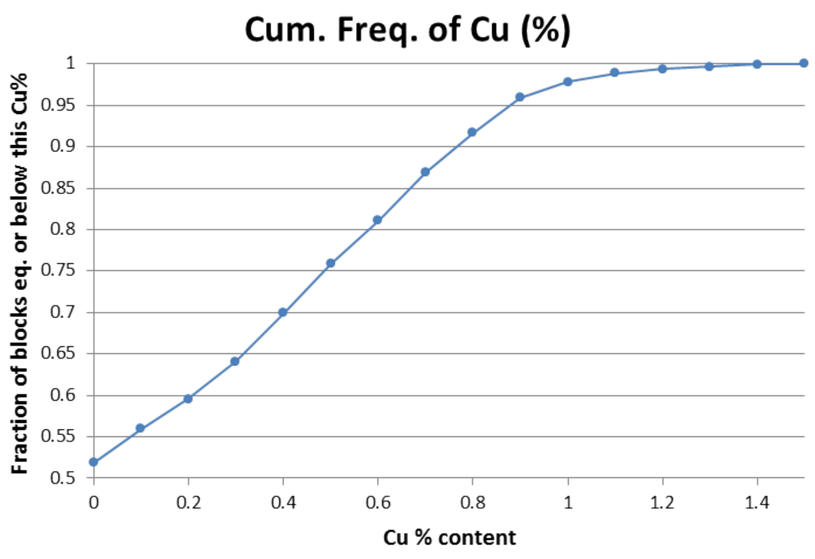


Figure 4.10: Marvin distribution of grade ranges

Figure 4.11.

The small inset on the bottom of Figure 4.11 shows a section through the middle of the schedule (the red rectangle in the figure) indicating the blocks removed during different periods.

Figures 4.12, 4.13 and 4.14 are screen grabs from the CPLEX optimisation logs captured after completion of optimisation run.

Figure 4.12 shows CPLEX employing various default strategies to reduce the solution space. Typical strategies include pre-solving, branch and bound, and introduction of cuts. Note that we did not intervene with CPLEX default solution strategies appert from the setting of the MIP Gap.

In Figure 4.13 the progression of valid integer solutions can be seen and the accompanying MIP gap (%) between the best integer solution (the incumbent) and the upper bound is shown as the "Gap" on the right. Since the solution tolerance has been pre-set to 5% as part of CPLEX optimisation criteria, the optimisation terminates after a value of 5% or less is found. The best integer solution at this gap (4.80%) is $946,423,526, which is the final solution value of the optimisation. From our research we have learned that similar MIP Gap setting of between 2% to 5% is commonly used by researchers solving scheduling problems via MILP models.

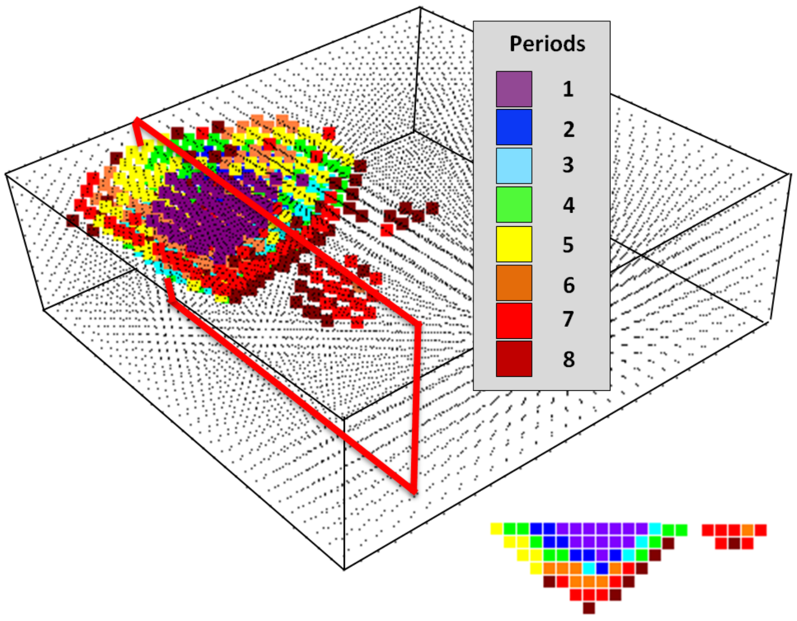
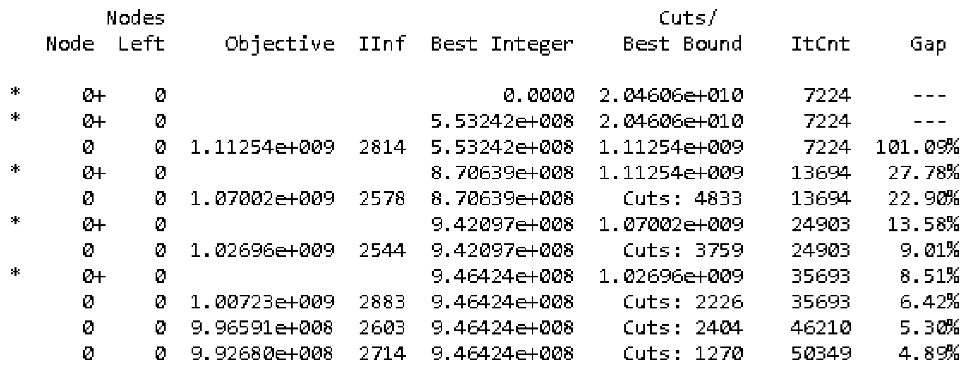


Figure 4.11: Marvin optimised schedule

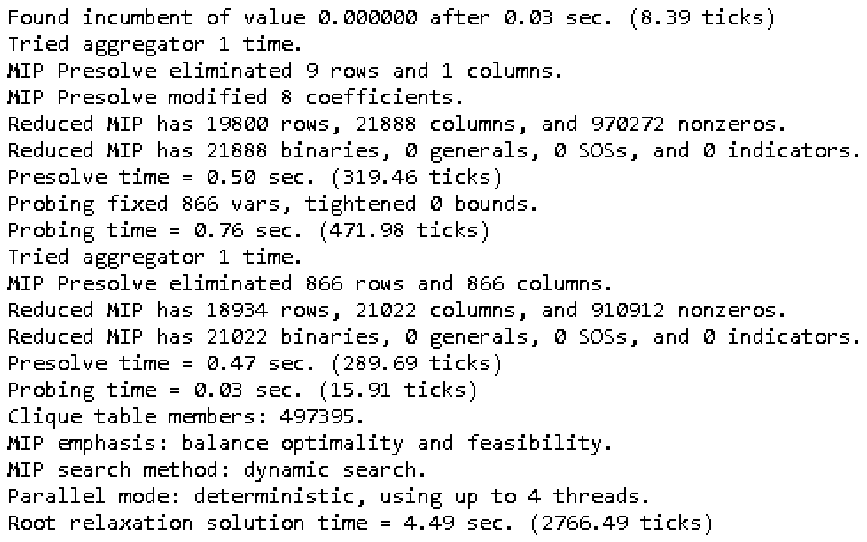


```
          Nodes                                        Cuts/
     Node  Left      Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                                0.0000  2.04606e+010    7224     ---
*     0+    0                          5.53242e+008  2.04606e+010    7224     ---
      0     0  1.11254e+009  2814  5.53242e+008  1.11254e+009    7224  101.09%
*     0+    0                          8.70639e+008  1.11254e+009   13694   27.78%
      0     0  1.07002e+009  2578  8.70639e+008    Cuts: 4833   13694   22.90%
*     0+    0                          9.42097e+008  1.07002e+009   24903   13.58%
      0     0  1.02696e+009  2544  9.42097e+008    Cuts: 3759   24903    9.01%
*     0+    0                          9.46424e+008  1.02696e+009   35693    8.51%
      0     0  1.00723e+009  2883  9.46424e+008    Cuts: 2226   35693    6.42%
      0     0  9.96591e+008  2603  9.46424e+008    Cuts: 2404   46210    5.30%
      0     0  9.92680e+008  2714  9.46424e+008    Cuts: 1270   50349    4.89%
```

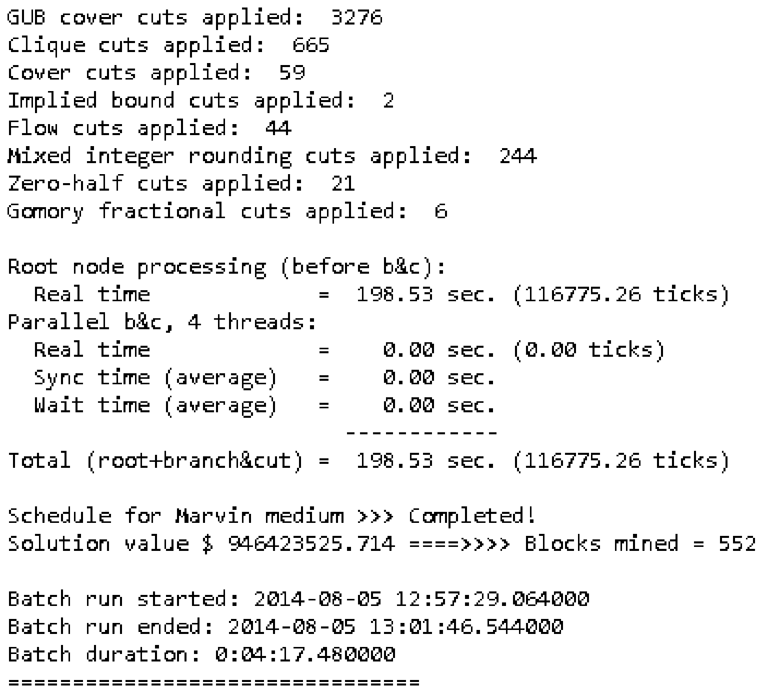
Figure 4.13: Marvin CPLEX - MIP progression

Figure 4.14 shows the final section of the optimisation log which includes the duration of the run. The total of 198.53 sec (approx. 3.3 minutes) is the duration only for the optimisation, whereas the batch duration (approx. 4.2 minutes) includes various trimming preprocessing and post processing procedures employed by the CPLEX solver.



```
Found incumbent of value 0.000000 after 0.03 sec. (8.39 ticks)
Tried aggregator 1 time.
MIP Presolve eliminated 9 rows and 1 columns.
MIP Presolve modified 8 coefficients.
Reduced MIP has 19800 rows, 21888 columns, and 970272 nonzeros.
Reduced MIP has 21888 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.50 sec. (319.46 ticks)
Probing fixed 866 vars, tightened 0 bounds.
Probing time = 0.76 sec. (471.98 ticks)
Tried aggregator 1 time.
MIP Presolve eliminated 866 rows and 866 columns.
Reduced MIP has 18934 rows, 21022 columns, and 910912 nonzeros.
Reduced MIP has 21022 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.47 sec. (289.69 ticks)
Probing time = 0.03 sec. (15.91 ticks)
Clique table members: 497395.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 4.49 sec. (2766.49 ticks)
```

Figure 4.12: Marvin CPLEX - optimisation strategy



```
GUB cover cuts applied:  3276
Clique cuts applied:  665
Cover cuts applied:  59
Implied bound cuts applied:  2
Flow cuts applied:  44
Mixed integer rounding cuts applied:  244
Zero-half cuts applied:  21
Gomory fractional cuts applied:  6

Root node processing (before b&c):
  Real time             =  198.53 sec. (116775.26 ticks)
Parallel b&c, 4 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =  198.53 sec. (116775.26 ticks)

Schedule for Marvin medium >>> Completed!
Solution value $ 946423525.714 ====>>>> Blocks mined = 552

Batch run started: 2014-08-05 12:57:29.064000
Batch run ended: 2014-08-05 13:01:46.544000
Batch duration: 0:04:17.480000
================================
```

Figure 4.14: Marvin CPLEX - optimisation results

4.6 Conclusion

In this chapter we developed a basic deterministic formulation of a MILP model with some easy and quick enhancements which we tested on a case study.

In the next chapter we will used the MILP model and introduce grade uncertainty, exploring ways in which it can be built into the variables and parameters, as well as ways in which the results can be interpreted.

Chapter 5

Introducing Risk and Uncertainty

5.1 Introduction

In this chapter we use the deterministic MILP formulation developed in Chapter 4 and introduce grade uncertainty by means of Conditional Simulation. Through the use of multiple equally probable instances (simulations or realisations) of grade input data we generate a range of output schedules and net present values. Each derived schedule is then tested against the full set of conditional simulations and the resulting range of net present values are recorded. Basic statistics reflecting the value range or standard deviation and minimum and maximum values are then considered in relation to choosing between various solutions. The approach up to this point has been well researched and reported on by a number of authors most notably those in collaboration with Dimitrakopoulos (Dimitrakopoulos (1990), Dimitrakopoulos (1994), Dimitrakopoulos (1998), Dimitrakopoulos et al. (2002), Dimitrakopoulos et al. (2007)). However, this approach up to this point still leave the decision maker with the problem of having to select the most attractive solution from the set of results generated, which is not a trivial task since both value as well as risk need to be considered.

In this chapter we use the standard process described above, generating basic statistical outputs (average values and standard deviations) but then progress beyond this point by introducing a novel Interpretive Framework which employs the Coefficient of Variation (CV) as a proxy for the risk/reward trade-off. We then test this framework on the schedules developed which enables us to identify the most attractive solutions from the 40 options.

This chapter is organised as follows:

Section 5.2 defines key concepts related to general risk and uncertainty. We look at mining risk and uncertainty in particular, and specifically focus on geological risk introducing Conditional Simulation.

Section 5.3 describes a methodology used to introduce multiple conditional simulations into the optimisation and capture the results - this is based on work by Dimitrakopoulos et al. (2007).

Section 5.4 utilises the methodology developed in Section 5.3 to generate schedules for a case study using 40 conditional simulations.

Section 5.4.1 we introduce the Interpretative Framework based on the Coefficient of Variation (CV) as a proxy for the risk/reward trade-off. We then test this framework on the schedules developed in the previous section. Using the CV and interpretive framework

The chapter concludes in Section 5.5 with conclusions as well a glimpse at an alternative approach (Scenario Optimisation - developed further in the following chapter) for modelling uncertainty which derives a single most attractive solution.

5.2 Preliminaries

Most traditional geo-statistical methods take a deterministic view on the resource model and grade data, leading to a single 'version of the truth'. However in reality, the data contained in such deterministic models have a large amount of uncertainty associated with their accuracy and variability. Deterministically derived plans based on such data (i.e. ignoring uncertainty) could potentially lead to vastly incorrect decisions in terms of what is valuable and what is waste, and thus whether and when to mine blocks.

5.2.1 Risk, Uncertainty and Variability defined

Renowned mathematician and statistician Sir David Cox famously said: "Variability is a phenomenon in the physical world to be measured, analysed and where appropriate explained. By contrast, uncertainty is an aspect of knowledge" (Vose, 2008).

Generally, there are two components influencing our ability to predict what the future holds: these are uncertainty and variability.

Uncertainty

Uncertainty is a state of limited information or knowledge (level of ignorance) about the parameters characterising the system or phenomenon being modelled. It is sometimes reducible (usually at an additional cost) through further measurement or study, or by consulting more experts. Uncertainty is by definition subjective, as it is a function of the assessor, but there are techniques available to allow one to be 'objectively subjective'. This essentially amounts to a logical assessment of the information contained in available data about model parameters without including any prior, non-quantitative information. The result is an uncertainty analysis that any logical person should agree with, given the available information (Vose, 2008).

Geologists analysing drilling and sampling data often have such discussions and compare notes when they interpret results. In Figure 5.1, deposit A has more uncertainty than deposit B due to the fact that there is a lot less information (drill hole data) available. However, because it contains less variability (i.e. more consistent geological and stratigraphic layering) than B it is nonetheless easier to come up with a reasonable interpretation in terms of the potential lateral extension of the geology. In deposit B however, even though a lot of information is available on the geology, due to its variability it will still be comparatively more difficult to derive a model of the geology.

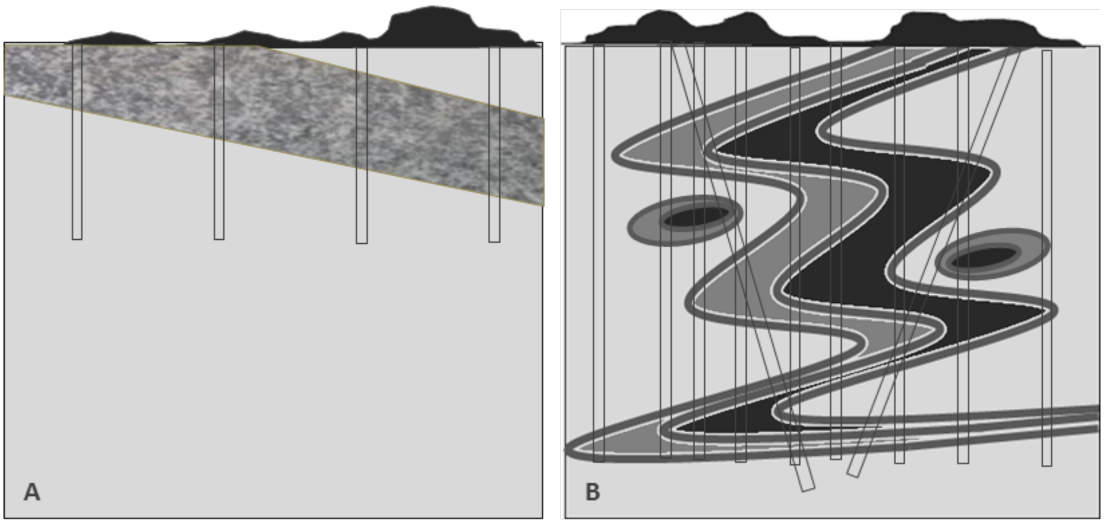


Figure 5.1: Uncertainty (A) and Variability (B) in geology

Variability

Variability is the effect of chance and is a function of the system - it is the 'nature of the beast'. In the geological context it's the change in content and quality spatially across the ore body. It is not reducible through either study or further measurement. A simple illustration of variability can be found in tossing a coin. If a coin is tossed, the expectation will be a head (H) or tail (T), each with a probability of 50% if the coin is presumed to be fair. If the coin is tossed twice, four possible outcomes are possible HH, HT, TH, TT, each with a probability of 25% because of the coin's symmetry. Exactly what the tosses of a coin will produce cannot be predicted upfront with certainty because of the inherent randomness of the coin toss.

In Figure 5.1, deposit A has less variability than B.

Total Uncertainty

Total uncertainty is often cited as the combination of uncertainty and variability. These two components act together to erode our ability to be able to predict what the future holds. Uncertainty and variability are philosophically very different, and they are usually kept separate in risk analysis modelling.

Probability

Probability is a numerical measurement of the likelihood of an outcome of some stochastic process. It is thus one of the two components, along with the values of the possible outcomes, that describe the variability of a system. Probability is used to define a probability distribution, which describes the range of values the variable may take, together with the probability (likelihood) that the variable will take any specific value.

Measuring the Center of the Distribution (The First Moment)

The first moment of a distribution measures the expected rate of return on a particular project. It measures the location of the project's scenarios and possible outcomes on average. The common statistics for the first moment include the mean (average), median (center of a distribution), and mode (most commonly occurring value) - see Figure 5.2.

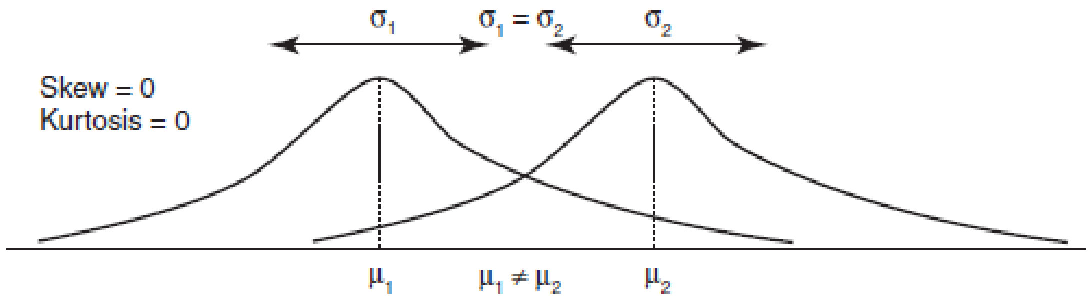


Figure 5.2: Statistic to demonstrate central tendency - source: Mun (2006)

Measuring the Spread of the Distribution (The Second Moment)

The second moment measures the spread of a distribution, which is a measure of risk or uncertainty. The spread or width of a distribution measures the variability of a variable, that is, the potential that the variable can fall into different regions of the distribution. The width or risk of a variable can be measured through several different statistics, including the range, standard deviation (s), variance, coefficient of variation, volatility, and percentiles - see Figure 5.3.

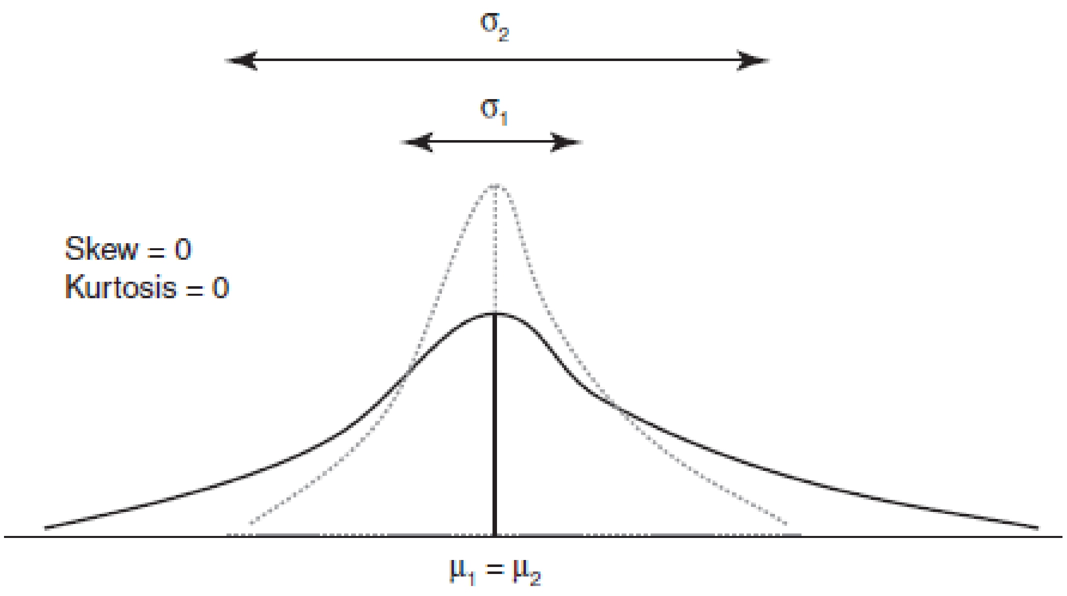


Figure 5.3: Statistic to demonstrate spread - source: Mun (2006)

If we assume we have two projects for which the underlying input uncertainty is different. If likely outcomes for the two projects are generated (e.g. using a Monte Carlo simulation method) then the results for each can be summarised in a statistical distribution. Figure 5.2 and Figure 5.3 illustrate the comparison of two projects by considering their average (μ_1 v.s. μ_2) and standard deviation (σ_1 v.s. σ_2) values.

In Figure 5.2 project 2 has a higher average value than project 1, however both project 1 and 2 have the same spread or standard deviation.

In Figure 5.3 project 1 and 2 have the same average value but project 2 has a greater (wider) spread or standard deviation than project 1.

Coefficient of Variation (CV)

The CV is defined as the ratio of standard deviation to the mean, which means that various risks can be common sized or 'normalised' for the sake of comparison. This measure of risk or dispersion is applicable when a variable's estimates, measures, magnitudes, or units differ. The CV is useful as a measure of risk per unit of return, or when inverted, can be used as a measure of 'bang for your buck' or returns per unit of risk. Thus, in portfolio optimization, one would be interested in minimizing the CV or maximizing the inverse of the CV. In the following sections we will be building on the view that the CV is a suitably representative metric to reflect the risk-return trade off for various options.

A number of statistical concepts required definition here as they will be used extensively in the chapters to come to make sense of outputs and results. The source used in describing these terms is Mun (2006).

5.2.2 Simulation

'Simulation' typically refers to any analytical method meant to imitate a real-life system, especially when other analyses are too mathematically complex or too difficult to reproduce.

Monte Carlo Simulation

'Monte Carlo Simulation' in particular, is a form of simulation that randomly generates values for uncertain variables over and over to simulate a model. Without the aid of simulation, a spreadsheet model for instance will only reveal a single outcome, generally the most likely or average scenario.

Monte Carlo Simulation in its simplest form is a random number generator that is useful for forecasting, estimation, and risk analysis. A simulation calculates numerous scenarios of a model by repeatedly picking values from a user-predefined probability distribution for the uncertain variables and using those values for the model. As all those scenarios produce associated results in a model, each scenario can have a forecast. Forecasts are events (usually with formulas or functions) that you define as important outputs of the model. In our case we will be forecasting the Net Present Value (NPV) of a mining schedule.

Monte Carlo Simulation is a fundamental concept that will be used in the generation of Conditional Simulations to be discussed in the following section.

Conditional Simulation

Conditional Simulation is a class of Monte Carlo techniques (Halton, 1970) that deals with ore body uncertainty in the generation of open pit schedule optimisation.

Uncertainty in ore grades due to imperfect geological knowledge of the deposit has traditionally been modelled using deterministic geo-statistical tools. But one of the possible alternative approaches to model such uncertainty is based on the generation of simulations (iterations), each representing a realistic interpretation of the underlying data.

The general procedure for creating sequential simulations includes the normal-score transformation of the raw data followed by a random selection of a node not yet simulated. Through the use of a local conditional probability distribution function a new value is then simulated and added to the set of conditioning data, within a specified radius of the new target location. This process is then repeated until all grid nodes have a simulated value. The procedure leads to a self-avoiding 'random walk' sequence. For each simulation or realisation, a different random walk leads to a different set of estimated block values.

For the sake of illustration, Figure 5.4 shows a set of two-dimensional 20 by 20 grids containing 27 (out of a total 400 possible) known data points (indicated in black). These might represent 27 points which have been drilled and sampled and for which grades are known. The rest of the unknown points are generated by conditional simulation using a random walk pattern. During each consecutive step (only the first 20 are shown) when the next random point is visited (indicated in red) the value for the new point is interpolated using the set of original sampled as well as the increasing set of newly estimated data points (both indicated in black).

Random walk here refers to the method in which successive data locations are selected for the generation of estimated values. A random walk entails a selection strategy whereby the next point selected could be any unvisited point. The next

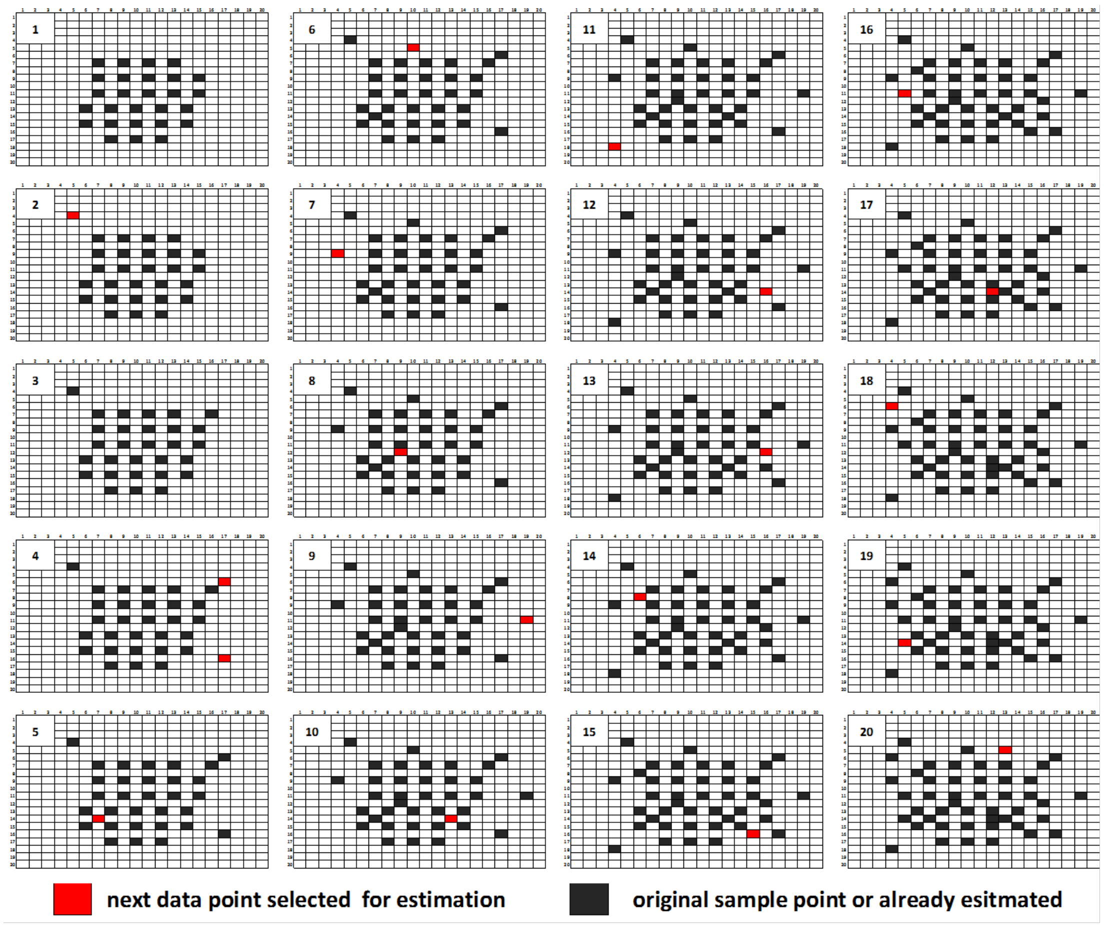

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
next data point selected for estimation
original sample point or already esitmated

Figure 5.4: Random walk generation of conditional simulation data points

value is selected randomly. This can be compared with other potential selection approaches which might be more systematic or sequential, for instance using the next point in a row or a column - see Figure 5.5.

Note that in the illustration (Figure 5.4) only 20 newly simulated points are shown progressively however in order to complete a particular simulation all points (in this case up to 400) need to either contain a known value (drilled and sampled) or have an estimated data point. Also, the illustration only represents a single simulation. Additional simulations can be generated by following a different random walking path.

A large number of models (simulations, interpretations or realisations) of the same deposit can be generated using Conditional Simulation. Each one of the generated models will be based on and conditioned to the same underlying actual data and statistical properties (histogram and variogram of the input data) in order to deal with the uncertainty related to the deposit and its attributes of

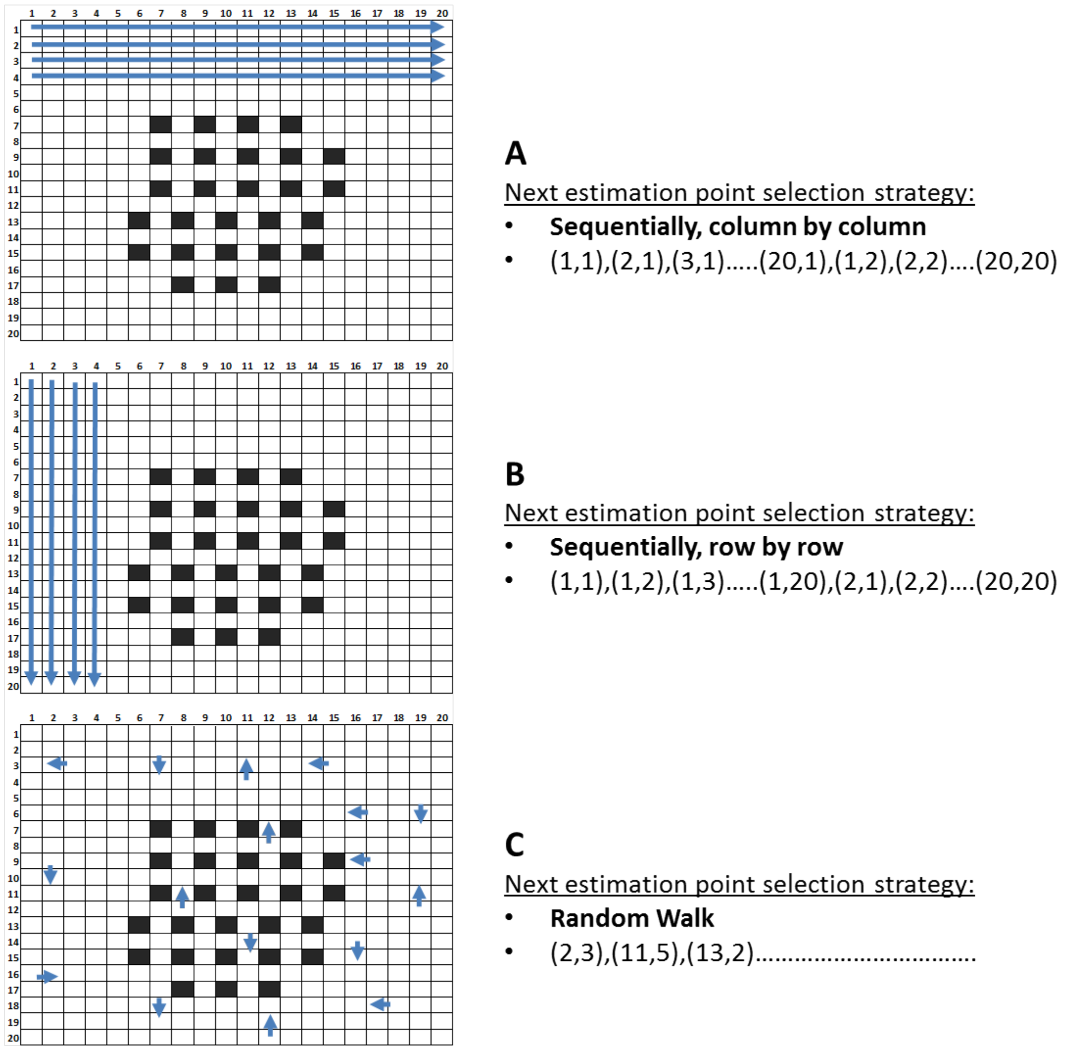

A
Next estimation point selection strategy:
• Sequentially, column by column
• (1,1),(2,1),(3,1).....(20,1),(1,2),(2,2)....(20,20)
B
Next estimation point selection strategy:
• Sequentially, row by row
• (1,1),(1,2),(1,3).....(1,20),(2,1),(2,2)....(20,20)
C
Next estimation point selection strategy:
• Random Walk
• (2,3),(11,5),(13,2).................................

Figure 5.5: Various estimation point selection approaches

interest.

Conditionally simulated models each represent the same deposit and have the two properties that firstly they are constrained to exactly reproduce all available information (known data points), and secondly being equally probable (equi-probable) representations of the actual deposit. Realisations are all different from one another since the interpretation of values in between known data points are uncertain. At known data points however the values between simulations are the same. Individual realisations are therefore 'simulations' rather than 'estimates' (such as those produced from kriging). More elaborate explanations of Conditional Simulation is provided by Deutsch (2002).

5.3 Optimisation using Conditional Simulations

The inherent uncertainty in estimated block values are introduced into our modelling procedure by iteratively pre-processing and optimising multiple equally probable block models previously generated through conditional simulation.

The resultant objective values and schedules for each unique optimisation is recorded for further treatment during the schedule assessment stage.

During the subsequent schedule assessment, each of the unique schedules (derived from individually optimising each conditional simulated block model) is tested against each conditional simulation (see Figure 5.6).

This is followed by statistical analysis and interpretation when the basic statistics (average and standard deviation) are calculated for each schedule when measured across each conditional simulation and the results are interpreted.

The following section describes a case study consisting of 40 optimised schedules which were generated and tested against the 40 conditionally simulated block models thus generating 1600 (40 x 40) data points.

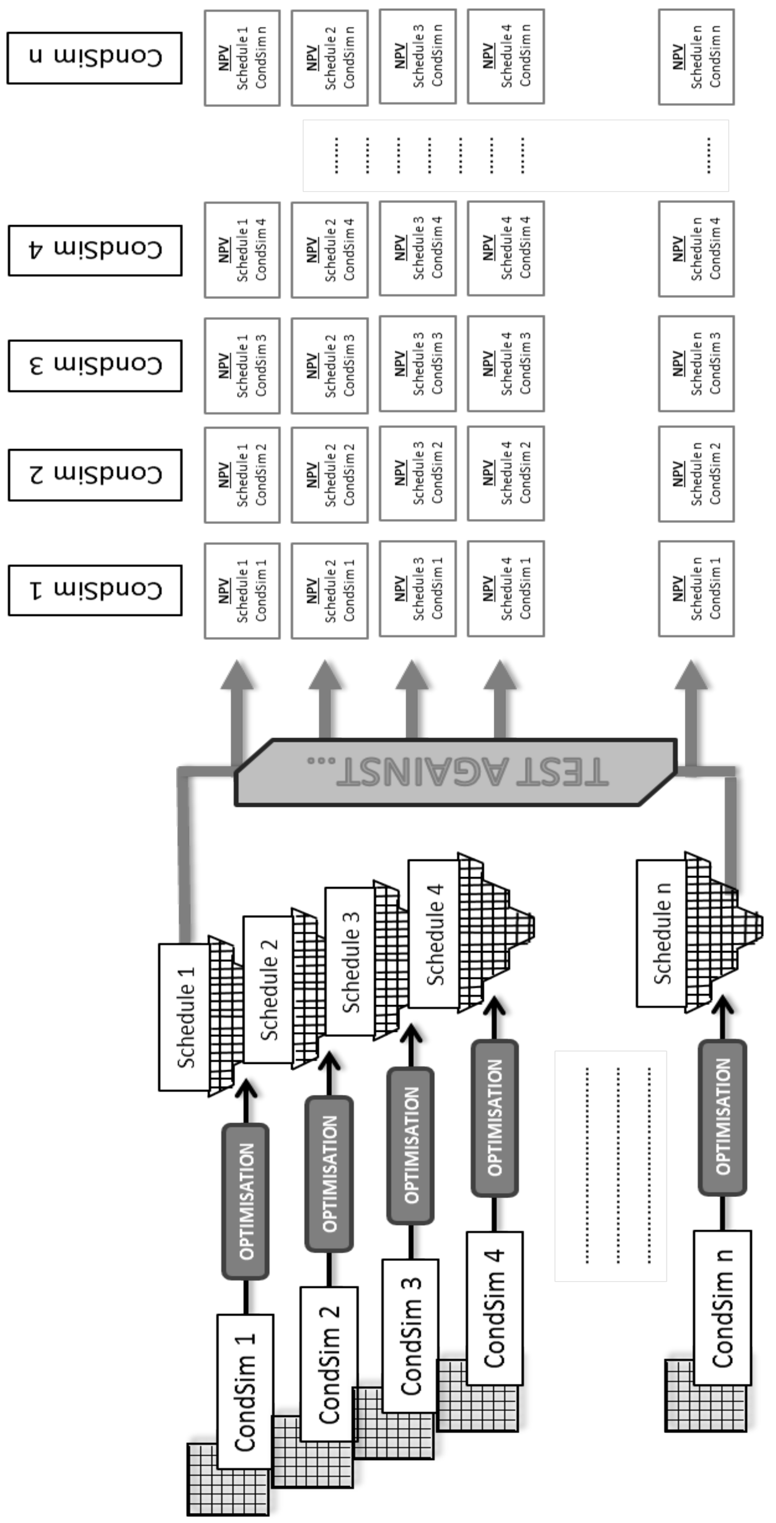


Figure 5.6: Comparing each Schedule against each Conditional Simulation

5.4 Conditional Simulation Case Study

The case study used for introducing conditional simulation is a grade model of and existing copper-gold deposit obtained from a commercial software provider (Datamine Australia). The data set includes 40 conditional simulations previously generated using Datamine's geo-statistical modelling software. The block model consisting of 11,200 blocks which are all equally sized cubes of 20 m in all dimensions. The model consists of 20, 40 and 14 blocks in each of the x, y and z dimensions respectively. Conditional Simulation was used on the original block model data in order to generate 40 equally probable simulations (realisations).

Figure 5.7 shows plan views of a random selection of 10 out of the 40 conditional simulations to illustrate comparative variability in a subset of the 40 resource models. Note that the green blocks represent high grades, whereas yellow and orange are lower grades and red is waste (i.e. containing no mineralisation).

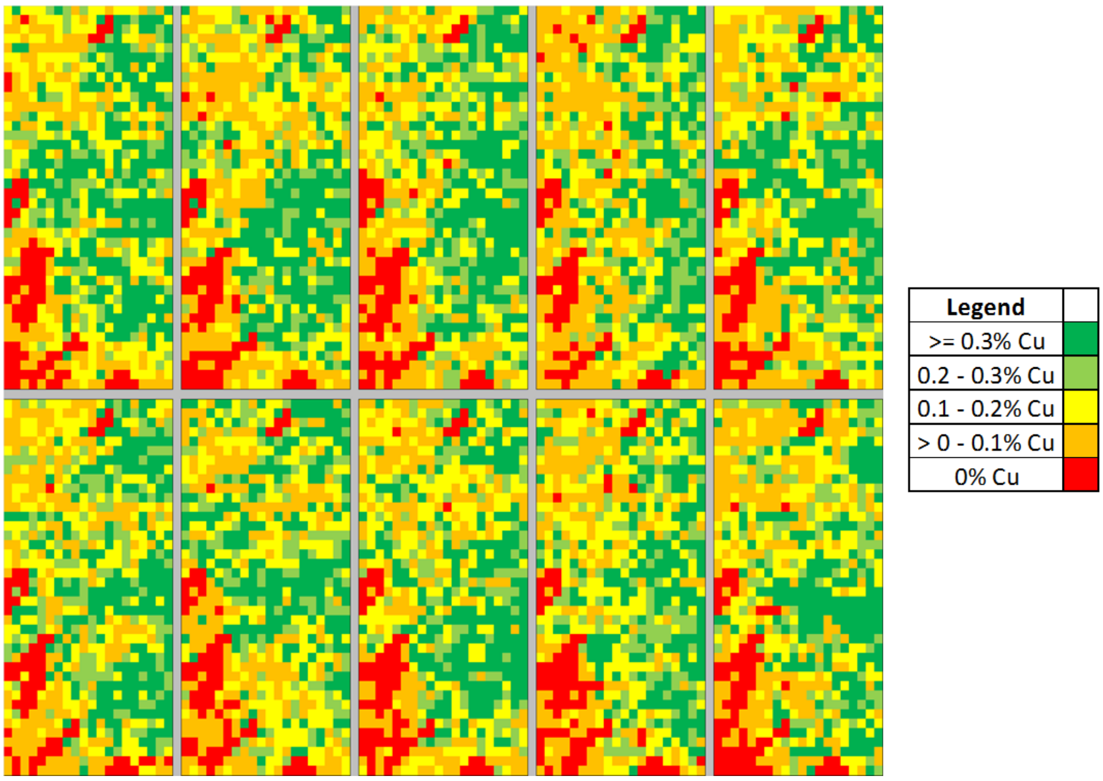


Figure 5.7: Ten example Conditional Simulations from 40

We use the same pre-processing strategies (data trimming and reduction), MILP formulation and optimised CPLEX settings (MIP gap) developed in previous chapters on each of the 40 conditional simulations.

The following technical and economic assumptions are used in defining the modelling parameters used in all optimisations:

- all blocks are equally sized cubes and therefore the three dimensions are the same (in all cases the block dimensions used are $20m$ x $20m$ x $20m$).
- blocks have variable densities depending on ore/waste characteristics.
- pit slopes of 45 degrees are assumed in all directions.
- The Cu price assumed is USD7200/t Cu.
- Process recovery is 85%.
- Mining cost is USD8/t mined.
- Ore processing cost is USD14/t treated.
- Discount rate = 10%.
- Number of annual mining periods = 8 years.
- Maximum Annual Processing Capacity = 1,000,000 tpa

5.4.1 Conventional application of Conditional Simulation

The procedures followed here in the generation of optimised results for multiple conditional simulation data sets have been used by others (e.g. Dimitrakopoulos et al. (2007)) before to generate ranges of outcomes, but they fail to provide a mechanism for choosing the most attractive option.

Optimisation

Each of the 40 conditional simulation datasets are used to derive an optimised mining schedule using the MILP formulation developed in previous chapters.

Figure 5.8 shows the 40 optimised schedules generated. The figure shows the plan view of blocks in the top layer of the schedules with different mining periods coloured in various shades of gray.

The purpose of this figure is to illustrate how vastly different the schedules are depending on which simulation is used for the optimisation. It is interesting to note that the variability in schedule outcomes are more noticeable than the apparent variability in the conditional simulations shown in Figure 5.7.

Schedule Assessment

During the schedule assessment stage, the blocks in each optimised schedule is evaluated using the block values of each of 40 conditional simulated data sets in order to assess the 'attractiveness' of each schedule (in terms of NPV).

Figure 5.9 captures the 1,600 NPVs derived from individually evaluating the 40 optimised schedules against each of the 40 conditional simulation data sets. As expected, the values (indicated diagonally) related to the evaluation of each optimised schedule against its own conditional simulated data set shows the original values obtained in the previous stage.

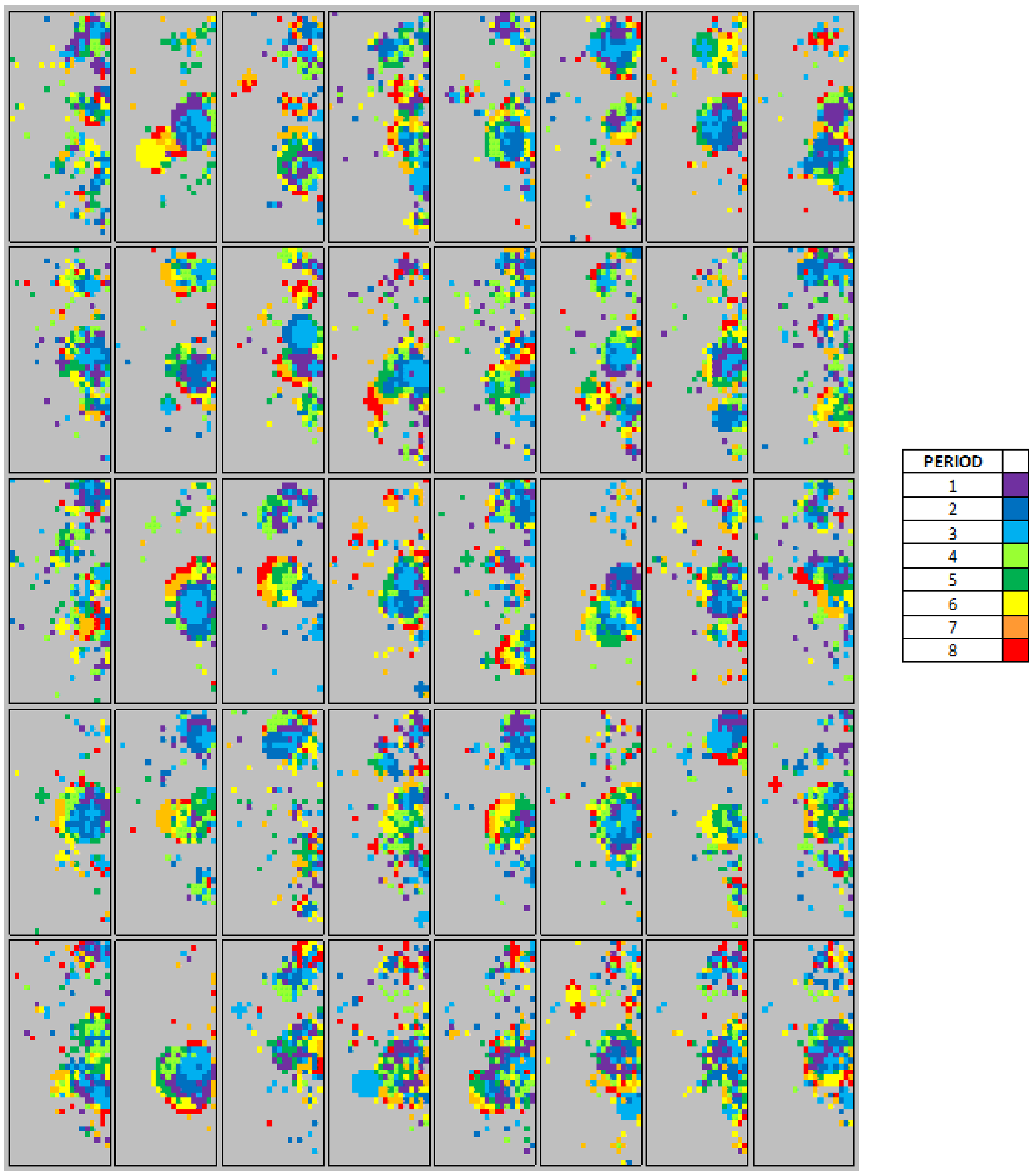
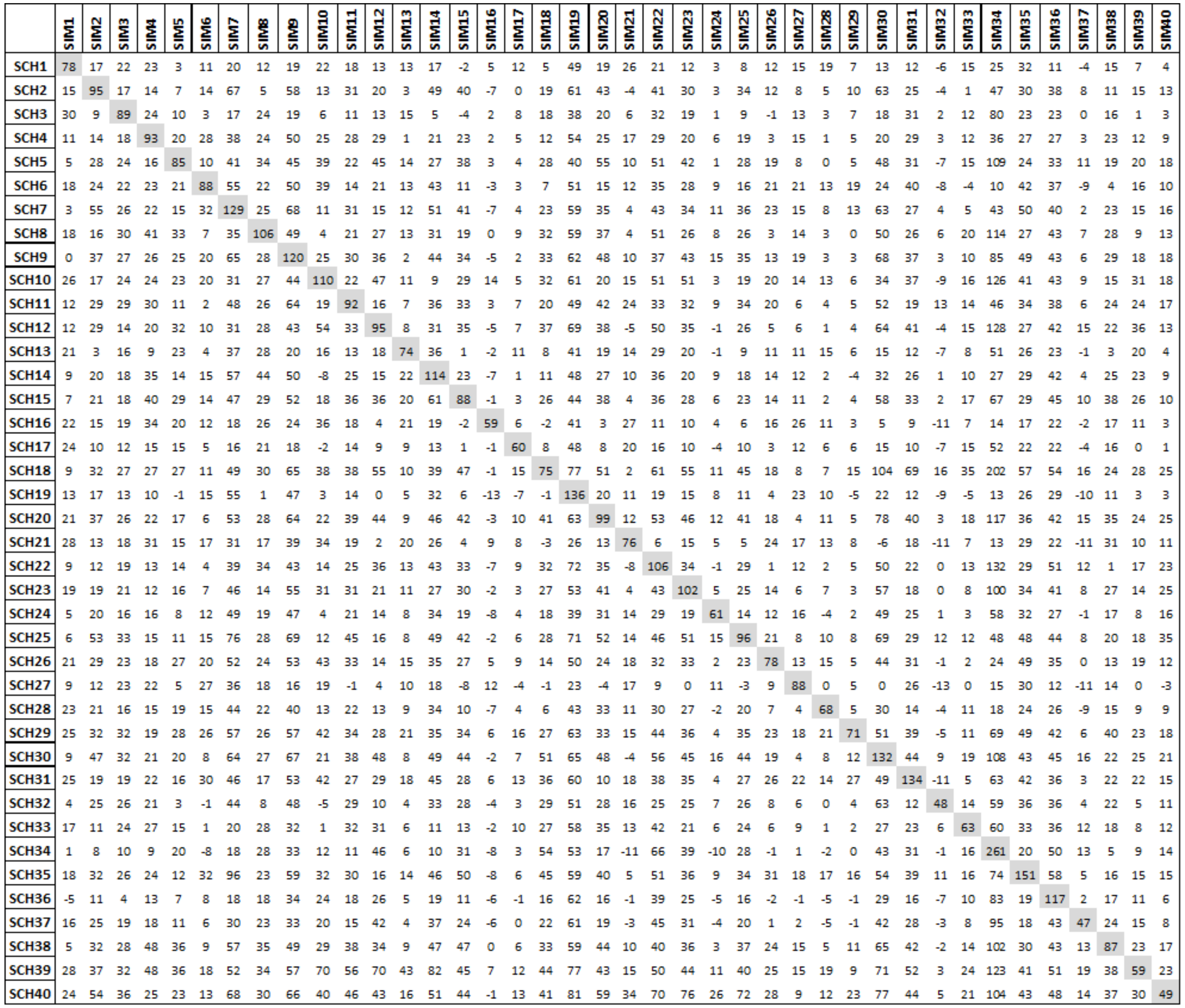


Figure 5.8: Schedule optimisations



	SIM1	SIM2	SIM3	SIM4	SIM5	SIM6	SIM7	SIM8	SIM9	SIM10	SIM11	SIM12	SIM13	SIM14	SIM15	SIM16	SIM17	SIM18	SIM19	SIM20	SIM21	SIM22	SIM23	SIM24	SIM25	SIM26	SIM27	SIM28	SIM29	SIM30	SIM31	SIM32	SIM33	SIM34	SIM35	SIM36	SIM37	SIM38	SIM39	SIM40
SCH1	78	17	22	23	3	11	20	12	19	22	18	13	13	17	-2	5	12	5	49	19	26	21	12	3	8	12	15	19	7	13	12	-6	15	25	32	11	-4	15	7	4
SCH2	15	95	17	14	7	14	67	5	58	13	31	20	3	49	40	-7	0	19	61	43	-4	41	30	3	34	12	8	5	10	63	25	-4	1	47	30	38	8	11	15	13
SCH3	30	9	89	24	10	3	17	24	19	6	11	13	15	5	-4	2	8	18	38	20	6	32	19	1	9	-1	13	3	7	18	31	2	12	80	23	23	0	16	1	3
SCH4	11	14	18	93	20	28	38	24	50	25	28	29	1	21	23	2	5	12	54	25	17	29	20	6	19	3	15	1	5	20	29	3	12	36	27	27	3	23	12	9
SCH5	5	28	24	16	85	10	41	34	45	39	22	45	14	27	38	3	4	28	40	55	10	51	42	1	28	19	8	0	5	48	31	-7	15	109	24	33	11	19	20	18
SCH6	18	24	22	23	21	88	55	22	50	39	14	21	13	43	11	-3	3	7	51	15	12	35	28	9	16	21	21	13	19	24	40	-8	-4	10	42	37	-9	4	16	10
SCH7	3	55	26	22	15	32	129	25	68	11	31	15	12	51	41	-7	4	23	59	35	4	43	34	11	36	23	15	8	13	63	27	4	5	43	50	40	2	23	15	16
SCH8	18	16	30	41	33	7	35	106	49	4	21	27	13	31	19	0	9	32	59	37	4	51	26	8	26	3	14	3	0	50	26	6	20	114	27	43	7	28	9	13
SCH9	0	37	27	26	25	20	65	28	120	25	30	36	2	44	34	-5	2	33	62	48	10	37	43	15	35	13	19	3	3	68	37	3	10	85	49	43	6	29	18	18
SCH10	26	17	24	24	23	20	31	27	44	110	22	47	11	9	29	14	5	32	61	20	15	51	51	3	19	20	14	13	6	34	37	-9	16	126	41	43	9	15	31	18
SCH11	12	29	29	30	11	2	48	26	64	19	92	16	7	36	33	3	7	20	49	42	24	33	32	9	34	20	6	4	5	52	19	13	14	46	34	38	6	24	24	17
SCH12	12	29	14	20	32	10	31	28	43	54	33	95	8	31	35	-5	7	37	69	38	-5	50	35	-1	26	5	6	1	4	64	41	-4	15	128	27	42	15	22	36	13
SCH13	21	3	16	9	23	4	37	28	20	16	13	18	74	36	1	-2	11	8	41	19	14	29	20	-1	9	11	11	15	6	15	12	-7	8	51	26	23	-1	3	20	4
SCH14	9	20	18	35	14	15	57	44	50	-8	25	15	22	114	23	-7	1	11	48	27	10	36	20	9	18	14	12	2	-4	32	26	1	10	27	29	42	4	25	23	9
SCH15	7	21	18	40	29	14	47	29	52	18	36	36	20	61	88	-1	3	26	44	38	4	36	28	6	23	14	11	2	4	58	33	2	17	67	29	45	10	38	26	10
SCH16	22	15	19	34	20	12	18	26	24	36	18	4	21	19	-2	59	6	-2	41	3	27	11	10	4	6	16	26	11	3	5	9	-11	7	14	17	22	-2	17	11	3
SCH17	24	10	12	15	15	5	16	21	18	-2	14	9	9	13	1	-1	60	8	48	8	20	16	10	-4	10	3	12	6	6	15	10	-7	15	52	22	22	-4	16	0	1
SCH18	9	32	27	27	27	11	49	30	65	38	38	55	10	39	47	-1	15	75	77	51	2	61	55	11	45	18	8	7	15	104	69	16	35	202	57	54	16	24	28	25
SCH19	13	17	13	10	-1	15	55	1	47	3	14	0	5	32	6	-13	-7	-1	136	20	11	19	15	8	11	4	23	10	-5	22	12	-9	-5	13	26	29	-10	11	3	3
SCH20	21	37	26	22	17	6	53	28	64	22	39	44	9	46	42	-3	10	41	63	99	12	53	46	12	41	18	4	11	5	78	40	3	18	117	36	42	15	35	24	25
SCH21	28	13	18	31	15	17	31	17	39	34	19	2	20	26	4	9	8	-3	26	13	76	6	15	5	5	24	17	13	8	-6	18	-11	7	13	29	22	-11	31	10	11
SCH22	9	12	19	13	14	4	39	34	43	14	25	36	13	43	33	-7	9	32	72	35	-8	106	34	-1	29	1	12	2	5	50	22	0	13	132	29	51	12	1	17	23
SCH23	19	19	21	12	16	7	46	14	55	31	31	21	11	27	30	-2	3	27	53	41	4	43	102	5	25	14	6	7	3	57	18	0	8	100	34	41	8	27	14	25
SCH24	5	20	16	16	8	12	49	19	47	4	21	14	8	34	19	-8	4	18	39	31	14	29	19	61	14	12	16	-4	2	49	25	1	3	58	32	27	-1	17	8	16
SCH25	6	53	33	15	11	15	76	28	69	12	45	16	8	49	42	-2	6	28	71	52	14	46	51	15	96	21	8	10	8	69	29	12	12	48	48	44	8	20	18	35
SCH26	21	29	23	18	27	20	52	24	53	43	33	14	15	35	27	5	9	14	50	24	18	32	33	2	23	78	13	15	5	44	31	-1	2	24	49	35	0	13	19	12
SCH27	9	12	23	22	5	27	36	18	16	19	-1	4	10	18	-8	12	-4	-1	23	-4	17	9	0	11	-3	9	88	0	5	0	26	-13	0	15	30	12	-11	14	0	-3
SCH28	23	21	16	15	19	15	44	22	40	13	22	13	9	34	10	-7	4	6	43	33	11	30	27	-2	20	7	4	68	5	30	14	-4	11	18	24	26	-9	15	9	9
SCH29	25	32	32	19	28	26	57	26	57	42	34	28	21	35	34	6	16	27	63	33	15	44	36	4	35	23	18	21	71	51	39	-5	11	69	49	42	6	40	23	18
SCH30	9	47	32	21	20	8	64	27	67	21	38	48	8	49	44	-2	7	51	65	48	-4	56	45	16	44	19	4	8	12	132	44	9	19	108	43	45	16	22	25	21
SCH31	25	19	19	22	16	30	46	17	53	42	27	29	18	45	28	6	13	36	60	10	18	38	35	4	27	26	22	14	27	49	134	-11	5	63	42	36	3	22	22	15
SCH32	4	25	26	21	3	-1	44	8	48	-5	29	10	4	33	28	-4	3	29	51	28	16	25	25	7	26	8	6	0	4	63	12	48	14	59	36	36	4	22	5	11
SCH33	17	11	24	27	15	1	20	28	32	1	32	31	6	11	13	-2	10	27	58	35	13	42	21	6	24	6	9	1	2	27	23	6	63	60	33	36	12	18	8	12
SCH34	1	8	10	9	20	-8	18	28	33	12	11	46	6	10	31	-8	3	54	53	17	-11	66	39	-10	28	-1	1	-2	0	43	31	-1	16	261	20	50	13	5	9	14
SCH35	18	32	26	24	12	32	96	23	59	32	30	16	14	46	50	-8	6	45	59	40	5	51	36	9	34	31	18	17	16	54	39	11	16	74	151	58	5	16	15	15
SCH36	-5	11	4	13	7	8	18	18	34	24	18	26	5	19	11	-6	-1	16	62	16	-1	39	25	-5	16	-2	-1	-5	-1	29	16	-7	10	83	19	117	2	17	11	6
SCH37	16	25	19	18	11	6	30	23	33	20	15	42	4	37	24	-6	0	22	61	19	-3	45	31	-4	20	1	2	-5	-1	42	28	-3	8	95	18	43	47	24	15	8
SCH38	5	32	28	48	36	9	57	35	49	29	38	34	9	47	47	0	6	33	59	44	10	40	36	3	37	24	15	5	11	65	42	-2	14	102	30	43	13	87	23	17
SCH39	28	37	32	48	36	18	52	34	57	70	56	70	43	82	45	7	12	44	77	43	15	50	44	11	40	25	15	19	9	71	52	3	24	123	41	51	19	38	59	23
SCH40	24	54	36	25	23	13	68	30	66	40	46	43	16	51	44	-1	13	41	81	59	34	70	76	26	72	28	9	12	23	77	44	5	21	104	43	48	14	37	30	49

Figure 5.9: Schedule optimisations compared against simulations

It is clear from the data that the evaluation of a particular schedule against a range of conditional simulations could lead to a potentially wide range of values.

See for instance the range of values when Schedule 1 (taken as example) is compared against the full set of 40 conditional simulation data sets (Figure 5.1 - tabulated in five columns by eight rows for ease of visualisation).

Table 5.1: Schedule 1 evaluated against all 40 conditional simulation data sets (NPVs in \$ m in brackets)

Sim1 (78)	Sim9 (19)	Sim17 (12)	Sim25 (8)	Sim33 (15)
Sim2 (17)	Sim10 (22)	Sim18 (5)	Sim26 (12)	Sim34 (25)
Sim3 (22)	Sim11 (18)	Sim19 (49)	Sim27 (15)	Sim35 (32)
Sim4 (23)	Sim12 (13)	Sim20 (19)	Sim28 (19)	Sim36 (11)
Sim5 (3)	Sim13 (13)	Sim21 (26)	Sim29 (7)	Sim37 (-4)
Sim6 (11)	Sim14 (17)	Sim22 (21)	Sim30 (13)	Sim38 (15)
Sim7 (20)	Sim15 (-2)	Sim23 (12)	Sim31 (12)	Sim39 (7)
Sim8 (12)	Sim16 (5)	Sim24 (3)	Sim32 (-6)	Sim40 (4)

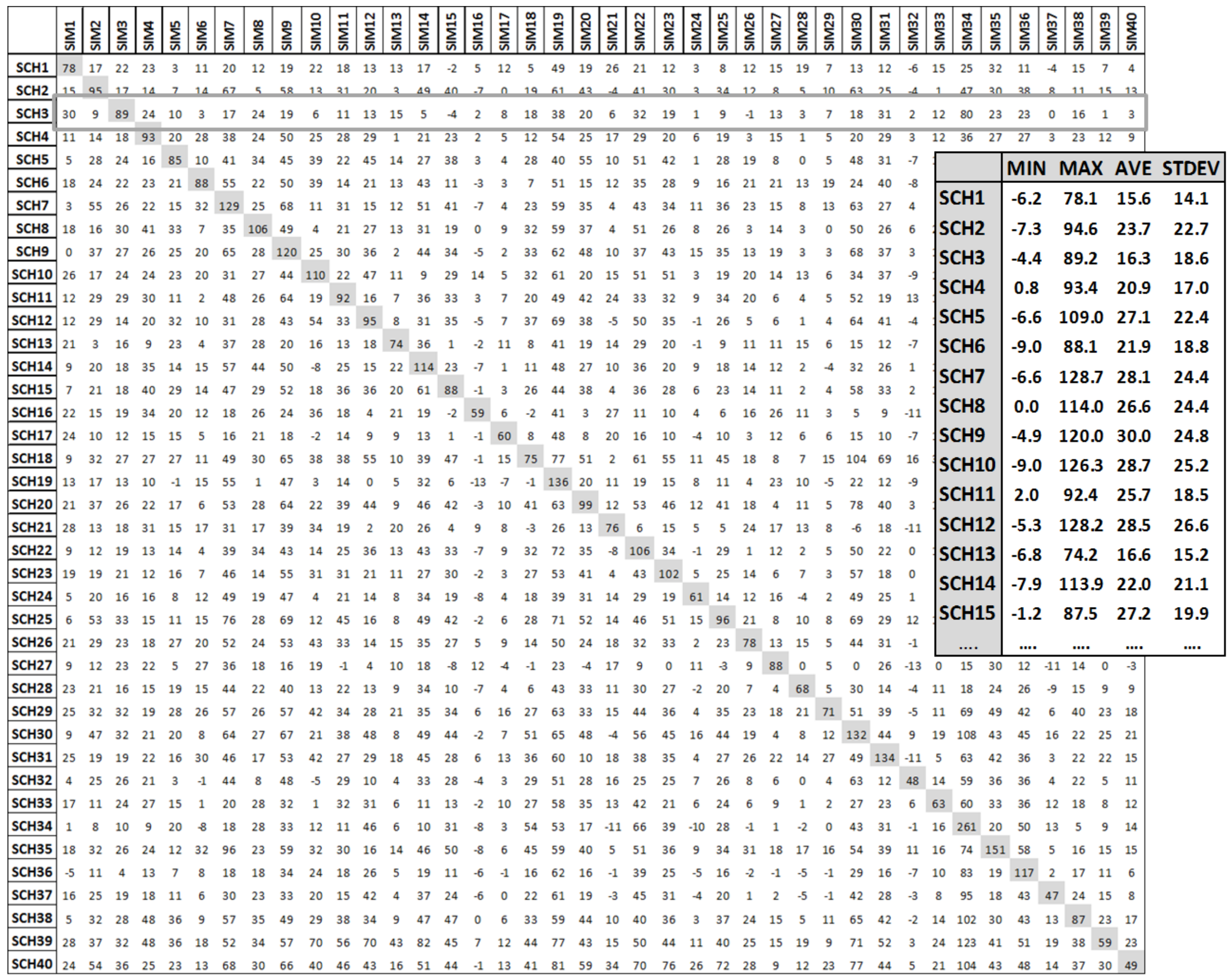
In the following section we conduct basic statistical analysis on the data (NPVs in \$M) and develop a framework to interpret the results.

Statistical Analysis and Interpretation

Notation:

- Let $L \in \{1, ..., 40\}$ be the index set of optimised mining schedules
- Let $T \in \{1, ..., 40\}$ be the set of values for each mining schedule
- The NPV associated with the t^{th} observation of the l^{th} optimisd plan is defined by NPV_t^l $\forall t \in T$, $\forall l \in L$
- The average NPV when comparing a specific optimised schedule across all 40 conditional simulation data sets i.e. $\mu NPV^l = \frac{\sum_{t \in T} NPV_t^l}{T}$
- The maximum NPV when comparing a specific optimised schedule across all 40 conditional simulation data sets i.e. $\max NPV^l = \max_{l \in L} \mu NPV^l$

Basic statistical analysis was completed for the data generated during the schedule assessment (Stage B).



	SIM1	SIM2	SIM3	SIM4	SIM5	SIM6	SIM7	SIM8	SIM9	SIM10	SIM11	SIM12	SIM13	SIM14	SIM15	SIM16	SIM17	SIM18	SIM19	SIM20
SCH1	78	17	22	23	3	11	20	12	19	22	18	13	13	17	-2	5	12	5	49	19
SCH2	15	95	17	14	7	14	67	5	58	13	31	20	3	49	40	-7	0	19	61	43
SCH3	30	9	89	24	10	3	17	24	19	6	11	13	15	5	-4	2	8	18	38	20
SCH4	11	14	18	93	20	28	38	24	50	25	28	29	1	21	23	2	5	12	54	25
SCH5	5	28	24	16	85	10	41	34	45	39	22	45	14	27	38	3	4	28	40	55
SCH6	18	24	22	23	21	88	55	22	50	39	14	21	13	43	11	-3	3	7	51	15
SCH7	3	55	26	22	15	32	129	25	68	11	31	15	12	51	41	-7	4	23	59	35
SCH8	18	16	30	41	33	7	35	106	49	4	21	27	13	31	19	0	9	32	59	37
SCH9	0	37	27	26	25	20	65	28	120	25	30	36	2	44	34	-5	2	33	62	48
SCH10	26	17	24	24	23	20	31	27	44	110	22	47	11	9	29	14	5	32	61	20
SCH11	12	29	29	30	11	2	48	26	64	19	92	16	7	36	33	3	7	20	49	42
SCH12	12	29	14	20	32	10	31	28	43	54	33	95	8	31	35	-5	7	37	69	38
SCH13	21	3	16	9	23	4	37	28	20	16	13	18	74	36	1	-2	11	8	41	19
SCH14	9	20	18	35	14	15	57	44	50	-8	25	15	22	114	23	-7	1	11	48	27
SCH15	7	21	18	40	29	14	47	29	52	18	36	36	20	61	88	-1	3	26	44	38
SCH16	22	15	19	34	20	12	18	26	24	36	18	4	21	19	-2	59	6	-2	41	3
SCH17	24	10	12	15	15	5	16	21	18	-2	14	9	9	13	1	-1	60	8	48	8
SCH18	9	32	27	27	27	11	49	30	65	38	38	55	10	39	47	-1	15	75	77	51
SCH19	13	17	13	10	-1	15	55	1	47	3	14	0	5	32	6	-13	-7	-1	136	20
SCH20	21	37	26	22	17	6	53	28	64	22	39	44	9	46	42	-3	10	41	63	99
SCH21	28	13	18	31	15	17	31	17	39	34	19	2	20	26	4	9	8	-3	26	13
SCH22	9	12	19	13	14	4	39	34	43	14	25	36	13	43	33	-7	9	32	72	35
SCH23	19	19	21	12	16	7	46	14	55	31	31	21	11	27	30	-2	3	27	53	41
SCH24	5	20	16	16	8	12	49	19	47	4	21	14	8	34	19	-8	4	18	39	31
SCH25	6	53	33	15	11	15	76	28	69	12	45	16	8	49	42	-2	6	28	71	52
SCH26	21	29	23	18	27	20	52	24	53	43	33	14	15	35	27	5	9	14	50	24
SCH27	9	12	23	22	5	27	36	18	16	19	-1	4	10	18	-8	12	-4	-1	23	-4
SCH28	23	21	16	15	19	15	44	22	40	13	22	13	9	34	10	-7	4	6	43	33
SCH29	25	32	32	19	28	26	57	26	57	42	34	28	21	35	34	6	16	27	63	33
SCH30	9	47	32	21	20	8	64	27	67	21	38	48	8	49	44	-2	7	51	65	48
SCH31	25	19	19	22	16	30	46	17	53	42	27	29	18	45	28	6	13	36	60	10
SCH32	4	25	26	21	3	-1	44	8	48	-5	29	10	4	33	28	-4	3	29	51	28
SCH33	17	11	24	27	15	1	20	28	32	1	32	31	6	11	13	-2	10	27	58	35
SCH34	1	8	10	9	20	-8	18	28	33	12	11	46	6	10	31	-8	3	54	53	17
SCH35	18	32	26	24	12	32	96	23	59	32	30	16	14	46	50	-8	6	45	59	40
SCH36	-5	11	4	13	7	8	18	18	34	24	18	26	5	19	11	-6	-1	16	62	16
SCH37	16	25	19	18	11	6	30	23	33	20	15	42	4	37	24	-6	0	22	61	19
SCH38	5	32	28	48	36	9	57	35	49	29	38	34	9	47	47	0	6	33	59	44
SCH39	28	37	32	48	36	18	52	34	57	70	56	70	43	82	45	7	12	44	77	43
SCH40	24	54	36	25	23	13	68	30	66	40	46	43	16	51	44	-1	13	41	81	59

	SIM21	SIM22	SIM23	SIM24	SIM25	SIM26	SIM27	SIM28	SIM29	SIM30	SIM31	SIM32	SIM33	SIM34	SIM35	SIM36	SIM37	SIM38	SIM39	SIM40
SCH1	26	21	12	3	8	12	15	19	7	13	12	-6	15	25	32	11	-4	15	7	4
SCH2	-4	41	30	3	34	12	8	5	10	63	25	-4	1	47	30	38	8	11	15	13
SCH3	6	32	19	1	9	-1	13	3	7	18	31	2	12	80	23	23	0	16	1	3
SCH4	17	29	20	6	19	3	15	1	5	20	29	3	12	36	27	27	3	23	12	9
SCH5	10	51	42	1	28	19	8	0	5	48	31	-7								
SCH6	12	35	28	9	16	21	21	13	19	24	40	-8								
SCH7	4	43	34	11	36	23	15	8	13	63	27	4								
SCH8	4	51	26	8	26	3	14	3	0	50	26	6								
SCH9	10	37	43	15	35	13	19	3	3	68	37	3								
SCH10	15	51	51	3	19	20	14	13	6	34	37	-9								
SCH11	24	33	32	9	34	20	6	4	5	52	19	13								
SCH12	-5	50	35	-1	26	5	6	1	4	64	41	-4								
SCH13	14	29	20	-1	9	11	11	15	6	15	12	-7								
SCH14	10	36	20	9	18	14	12	2	-4	32	26	1								
SCH15	4	36	28	6	23	14	11	2	4	58	33	2								
SCH16	27	11	10	4	6	16	26	11	3	5	9	-11								
SCH17	20	16	10	-4	10	3	12	6	6	15	10	-7								
SCH18	2	61	55	11	45	18	8	7	15	104	69	16								
SCH19	11	19	15	8	11	4	23	10	-5	22	12	-9								
SCH20	12	53	46	12	41	18	4	11	5	78	40	3								
SCH21	76	6	15	5	5	24	17	13	8	-6	18	-11								
SCH22	-8	106	34	-1	29	1	12	2	5	50	22	0								
SCH23	4	43	102	5	25	14	6	7	3	57	18	0								
SCH24	14	29	19	61	14	12	16	-4	2	49	25	1								
SCH25	14	46	51	15	96	21	8	10	8	69	29	12								
SCH26	18	32	33	2	23	78	13	15	5	44	31	-1								
SCH27	17	9	0	11	-3	9	88	0	5	0	26	-13	0	15	30	12	-11	14	0	-3
SCH28	11	30	27	-2	20	7	4	68	5	30	14	-4	11	18	24	26	-9	15	9	9
SCH29	15	44	36	4	35	23	18	21	71	51	39	-5	11	69	49	42	6	40	23	18
SCH30	-4	56	45	16	44	19	4	8	12	132	44	9	19	108	43	45	16	22	25	21
SCH31	18	38	35	4	27	26	22	14	27	49	134	-11	5	63	42	36	3	22	22	15
SCH32	16	25	25	7	26	8	6	0	4	63	12	48	14	59	36	36	4	22	5	11
SCH33	13	42	21	6	24	6	9	1	2	27	23	6	63	60	33	36	12	18	8	12
SCH34	-11	66	39	-10	28	-1	1	-2	0	43	31	-1	16	261	20	50	13	5	9	14
SCH35	5	51	36	9	34	31	18	17	16	54	39	11	16	74	151	58	5	16	15	15
SCH36	-1	39	25	-5	16	-2	-1	-5	-1	29	16	-7	10	83	19	117	2	17	11	6
SCH37	-3	45	31	-4	20	1	2	-5	-1	42	28	-3	8	95	18	43	47	24	15	8
SCH38	10	40	36	3	37	24	15	5	11	65	42	-2	14	102	30	43	13	87	23	17
SCH39	15	50	44	11	40	25	15	19	9	71	52	3	24	123	41	51	19	38	59	23
SCH40	34	70	76	26	72	28	9	12	23	77	44	5	21	104	43	48	14	37	30	49

	MIN	MAX	AVE	STDEV
SCH1	-6.2	78.1	15.6	14.1
SCH2	-7.3	94.6	23.7	22.7
SCH3	-4.4	89.2	16.3	18.6
SCH4	0.8	93.4	20.9	17.0
SCH5	-6.6	109.0	27.1	22.4
SCH6	-9.0	88.1	21.9	18.8
SCH7	-6.6	128.7	28.1	24.4
SCH8	0.0	114.0	26.6	24.4
SCH9	-4.9	120.0	30.0	24.8
SCH10	-9.0	126.3	28.7	25.2
SCH11	2.0	92.4	25.7	18.5
SCH12	-5.3	128.2	28.5	26.6
SCH13	-6.8	74.2	16.6	15.2
SCH14	-7.9	113.9	22.0	21.1
SCH15	-1.2	87.5	27.2	19.9
....	....	....	....	....

Figure 5.10: Statistical analysis for NPVs ($M) from schedule optimisations

As schedules from 1 to 40 are considered row by row, the NPV (in \$M) statistics resulting from evaluating schedules across the 40 conditional simulation data sets (columns) are calculated (see inset for first 15 schedules showing MIN, MAX, AVE and STDEV of NPVs).

The key statistics considered for each of the schedules are minimum value in the range (NPV_{MIN}), maximum value in the range (NPV_{MAX}), average value over the range (NPV_{AVE}) and standard deviation over the range (NPV_{STD}).

The objective of the interpretive stage of the study is to obtain an appreciation of the relative 'attractiveness' of individual schedules when compared over the range of simulations. Attractiveness is a function of both the average value (also called *return* here) and the *riskiness* of a schedule.

Consider the comparative results shown in Figure 5.11. The x-axis shows the 40 schedules used to test results against simulations. The y-axis shows the spread in NPV as a result of comparing a schedule against the range of simulations. The vertical bars of variable length represent the spread of values between the maximum (NPV_{MAX}) and minimum(NPV_{MIN}), and the circular marker indicates the average value (NPV_{AVE}). The difference between the NPV_{MAX} and NPV_{MIN} (the value range or spread) can be seen as an early indication of risk or uncertainty.

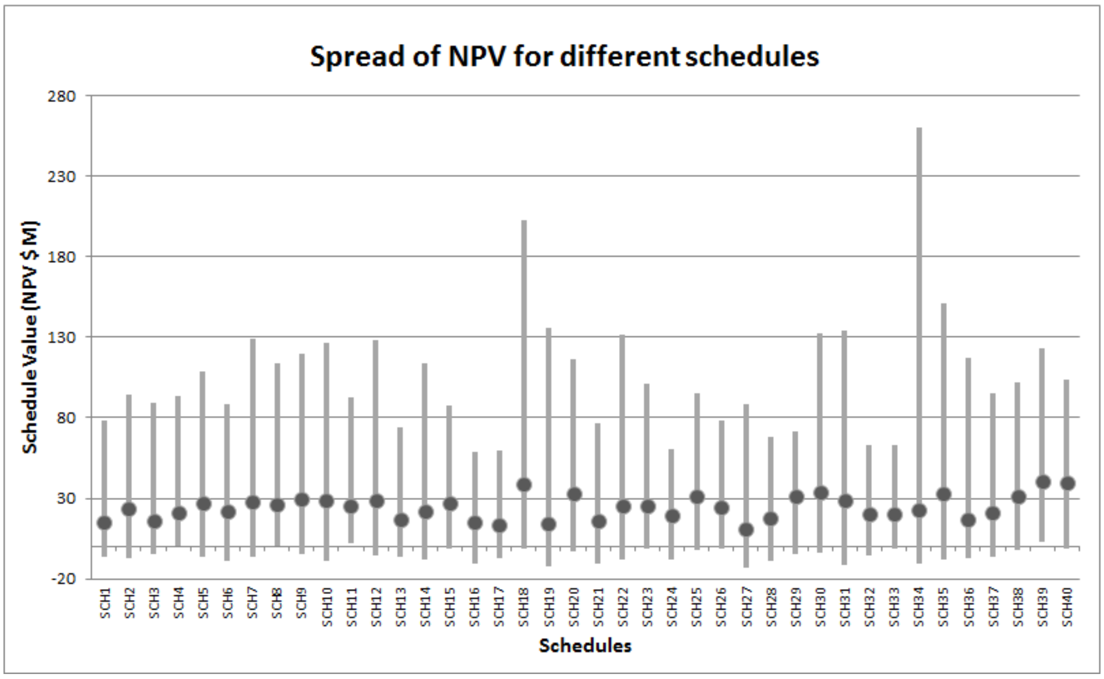


Figure 5.11: Key statistics for schedules when tested against simulations

Following this approach it can be deduced that for instance Schedule 17 has a much lower risk (spread) than Schedule 18. Schedule 17 might therefore be seen as a safer, less risky solution, i.e. not subject to large fluctuations in value when the underlying data is changed over the spectrum of simulations. However as mentioned before, the other important consideration is value, represented by average value (NPV_{AVE}). So considering the same two examples, although Schedule 17 has a lower risk than Schedule 18, it also has a lower average value ($13M) than schedule 18 ($38M). A lower risk solution often comes at the expense of sacrificing potential additional value. Up to this point we have not introduced any new concepts and in fact this point and conclusions have also reached by other researchers notably e.g. Dimitrakopoulos et al. (2007).

The initial excitement that Conditional Simulation generated in the mining industry over the last few decades have since made way for frustration on the side of industry practitioners since the approaches proposed by advocates of conditional simulation still leave decision-makers with numerous outcomes from which a selection needs to be made as to which is the most attractive. In addition, the current approaches do not provide a mechanism to compare risk against return...

5.4.2 Interpretive Framework for considering Risk-Reward trade-off

We now introduce the concept of risk-return 'trade-off' which essentially means that, depending on the tolerance for risk, a decision maker will be interested as much as possible in both maximising value and minimising risk at the same time, leading to potentially conflicting objectives. This often leads to a decision maker having to sacrifice value in order to reduce risk, or take on more risk in order to increase value. It all depends on how many 'units of value' need to the sacrificed in order to reduce a certain amount of 'units of risk'.

When the results seen above are sorted based on decreasing average value (see Figure 5.12) this leads to a ranking of most to least attractive options (schedules).

However if the same information is ranked according to increasing standard deviation (see Figure 5.13) this leads to a different assortment of most to least appealing options.

Note that increasing standard deviation does not necessarily coincide smoothly with increasing range.

When comparing the top ten best options from a value and risk perspective respectively, there is no overlap (i.e. none of the options appear in both - see

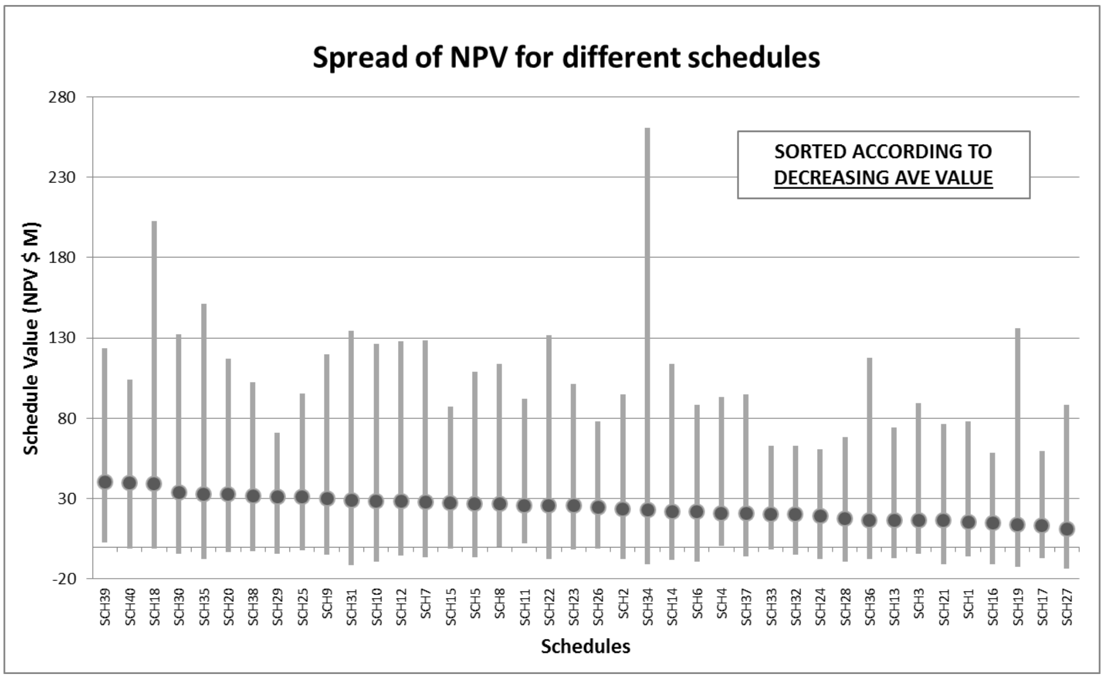


Figure 5.12: Sorting of schedules according to decreasing value

Table 5.2).

One of the potential problems with interpreting the results is that the outcomes (NPV) are on different scales. A useful ratio that can be applied to remove the scale issue is the Coefficient Of Variation (CV). The CV represents the ratio of the standard deviation (STDEV) to the mean or average, and it is a useful statistic for comparing the degree of variation from one data series to another, even if the means are different from each other. The CV can also be interpreted as a statistical measure of the dispersion of data points in a data series around the average.

It is our argument that by using the CV it is possible to determine how much risk (STDEV) is taken on in comparison to the amount of value (AVE NPV) expected. The lower the CV, the better the risk-return trade off.

A plot of the CV values for the data generated is shown in Figure 5.14.

By sorting the CV in order of increasing value it is possible to more effectively rank schedules from most to least attractive (see Figure 5.15).

When the new assortment is compared with the value and risk assortments considered before, it can be seen that the new ranking has very limited correlation with the previous rankings - see Table 5.3.

The CV is a useful tool for identifying prospective options for further consid-

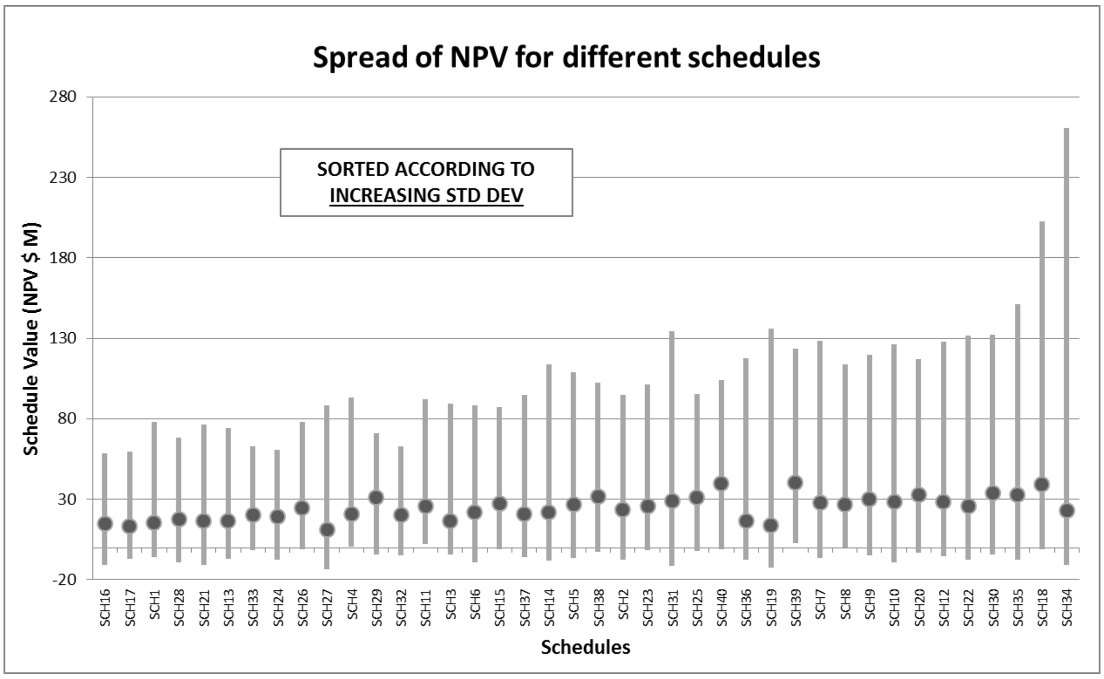


Figure 5.13: Sorting of schedules according to increasing risk

eration in the following section.

A further useful scaling modification in the data is achieved by converting the data ranges for NPV_{AVE} and NPV_{STD} to unit values. This is done by dividing values in the two ranges by the NPV_{MAX} and NPV_{MIN} respectively which produces values between 0 and 1 for $UNPV_{AVE}$ and $UNPV_{STD}$. The rescaling does not change the prioritisation of options as can be seen in figures 5.16 and 5.17.

By plotting the modified unit (or 'normalised') ranges against one another it is possible to obtain a unique interpretation of risk ($UNPV_{STD}$) against return ($UNPV_{AVE}$) - see figure 5.18.

Table 5.2: Comparison of most attractive option from a value and risk perspective

Rank	Sorted on Value	Sorted on Risk
1 (best)	39	16
2	40	17
3	18	1
4	30	28
5	35	21
6	20	13
7	38	33
8	29	24
9	25	26
10	9	27

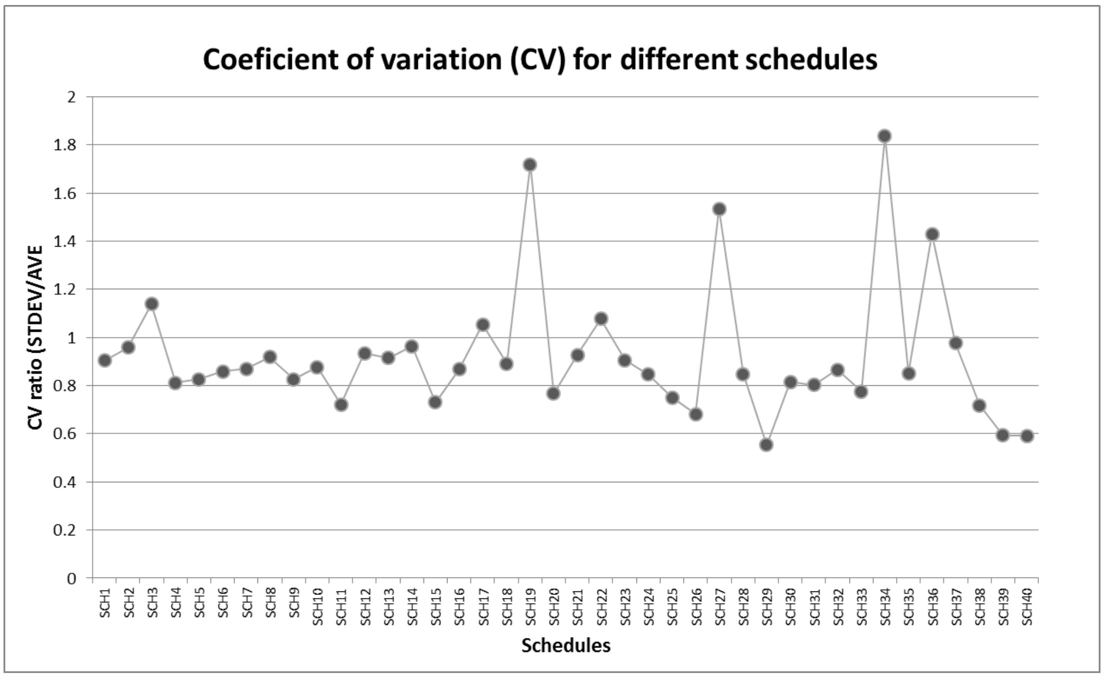


Figure 5.14: CV plotted for each schedule comparison

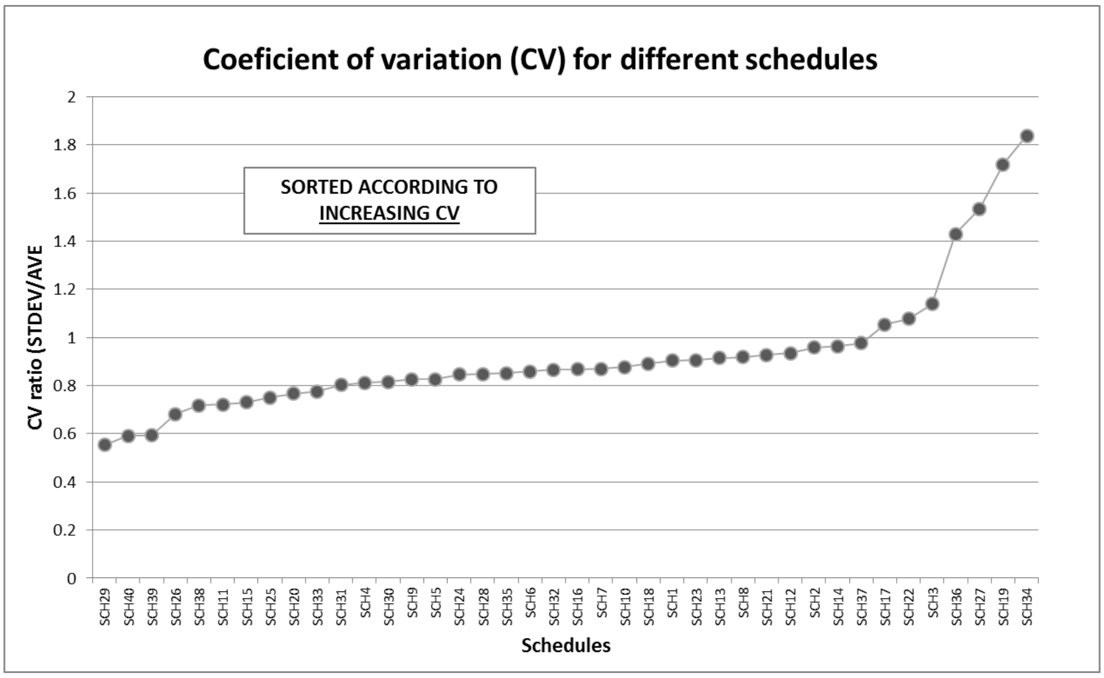


Figure 5.15: CV plotted for each schedule comparison - sorted on increasing CV

Table 5.3: Comparison of most attractive option from a value and risk perspective

Rank	Sorted on Value	Sorted on Risk	Sorted on CV
1 (best)	39	16	29
2	40	17	40
3	18	1	39
4	30	28	26
5	35	21	38
6	20	13	11
7	38	33	15
8	29	24	25
9	25	26	20
10	9	27	33

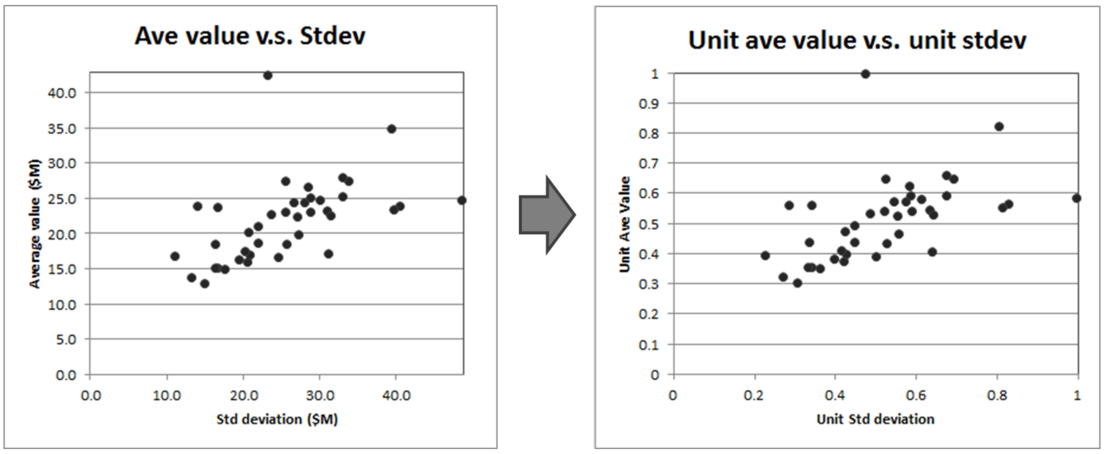


Figure 5.16: Rescaling effect on Ave value and Std deviation

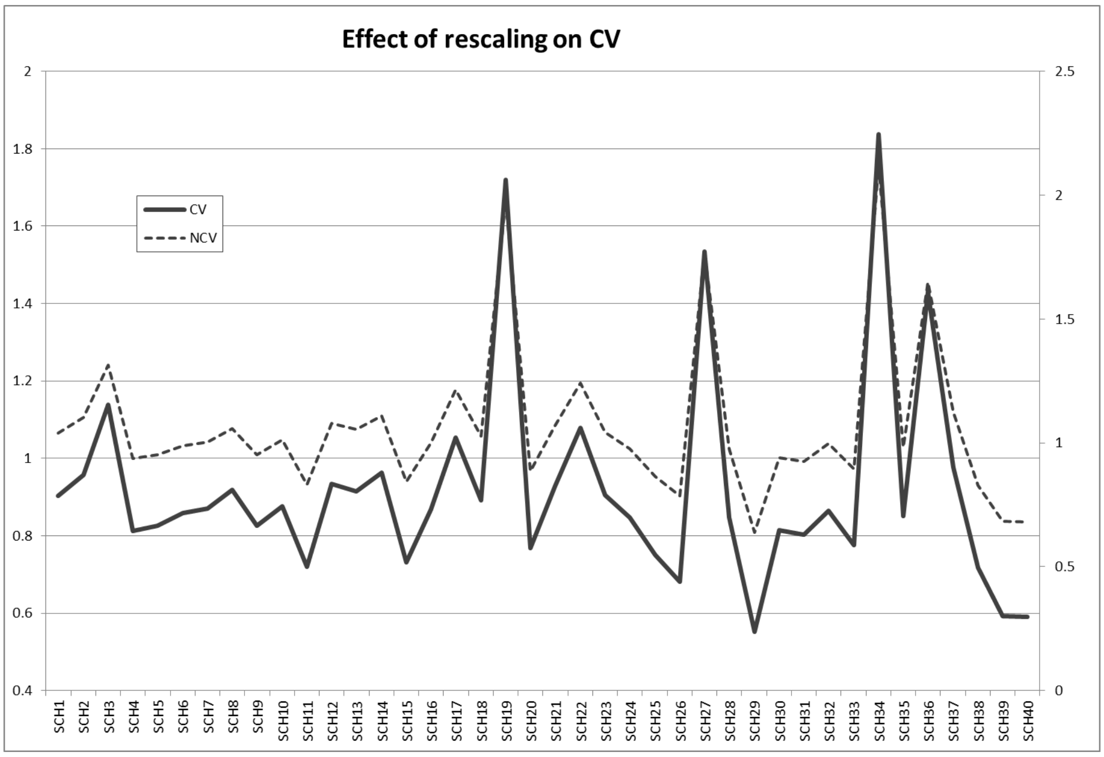


Figure 5.17: Rescaling effect on CV (showing Unit or Normalised CV as NCV)

	AVE	STD	CV
SCH29	0.77	0.41	0.53
SCH40	0.98	0.55	0.56
SCH39	1.00	0.57	0.57
SCH26	0.61	0.39	0.65
SCH38	0.78	0.53	0.68
SCH11	0.63	0.44	0.69
SCH15	0.67	0.47	0.70
SCH25	0.76	0.55	0.72
SCH20	0.81	0.59	0.73
SCH33	0.51	0.37	0.74
SCH31	0.71	0.54	0.77
SCH4	0.52	0.40	0.77
SCH30	0.83	0.65	0.78
SCH9	0.74	0.58	0.79
SCH5	0.67	0.53	0.79
SCH24	0.48	0.39	0.81
SCH28	0.43	0.35	0.81
SCH35	0.81	0.66	0.81
SCH6	0.54	0.44	0.82
SCH32	0.50	0.41	0.83
SCH16	0.37	0.31	0.83
SCH7	0.69	0.57	0.83
SCH10	0.71	0.59	0.84
SCH18	0.97	0.82	0.85
SCH1	0.38	0.33	0.86
SCH23	0.63	0.54	0.86
SCH13	0.41	0.36	0.87
SCH8	0.66	0.57	0.88
SCH21	0.40	0.36	0.89
SCH12	0.70	0.63	0.89
SCH2	0.58	0.53	0.91
SCH14	0.54	0.50	0.92
SCH37	0.51	0.48	0.93
SCH17	0.32	0.33	1.00
SCH22	0.63	0.65	1.03
SCH3	0.40	0.44	1.09
SCH36	0.41	0.56	1.36
SCH27	0.27	0.40	1.46
SCH19	0.34	0.56	1.64
SCH34	0.57	1.00	1.75

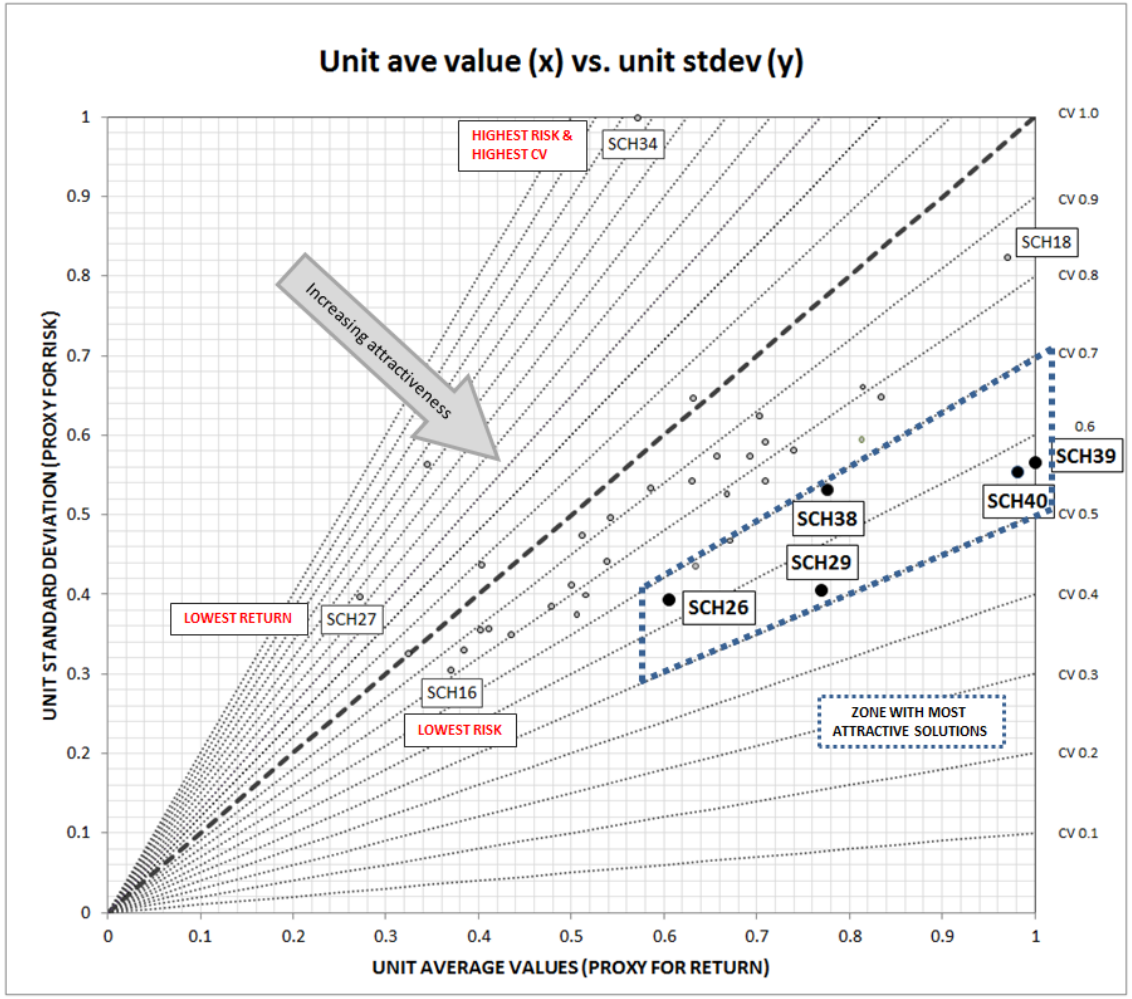


Figure 5.18: Risk against return

Figure 5.18 is a scatter plot of $UNPV_{STD}$ (vertical axis) against $UNPV_{AVE}$ (horizontal axis) for the statistics related to the evaluation of the 40 optimised schedules against the 40 simulation inputs. The plotted data generally forms an elongated cloud trending roughly from the bottom left to top right of the figure.

The points on the scatter plot are labelled for further discussion and statistics for all options in the plot are summarised in the left hand margin of the figure.

The dotted lines fanning outwards from the origin represent different 'contours' connecting similar ratios between $UNPV_{AVE}$ and $UNPV_{STD}$, and the gradient of these lines corresponds to the normalised CV values ($UNPV_{STD}$ / $UNPV_{AVE}$). One of the additional benefits of converting the two statistics to unit values is that the right-hand side of the unit plot area becomes equal to $UNPV_{CV}$ since at that side $UNPV_{AVE}$ is equal to 1 and therefore $UNPV_{CV} = UNPV_{STD}/1$.

These individual contour lines can be interpreted as representing different risk-return options. Points (schedules) on the same dotted line represent similar risk-return trade-off ratios (between unit risk and unit value as previously defined for $UNPV_{CV}$). When comparing schedules, those falling on or between the lowest $UNPV_{CV}$ contours (comparatively) will be the most attractive options. Such schedules will have the most favourable trade off between value and risk amongst the available (defined) options. When comparing schedules with the same (or similar) favourable $UNPV_{CV}$ values, the next consideration will be risk appetite.

The five most attractive options (in order of decreasing 'attractiveness') under the criteria defined above (schedules 29, 40, 39, 26 and 38) have been labelled in Figure 5.18 (note that the statistics in the table on the left of the figure has been sorted in order of decreasing CV).

Schedule 29 can clearly be seen to be the best option in terms of risk-return trade-off with the lowest $UNPV_{CV}$ of 0.53. Schedule 29 has a NPV_{AVE} of \$31M and NPV_{STD} of \$17M. The next best schedule (40) represents a higher $UNPV_{AVE}$ (\$40M) but also a higher $UNPV_{STD}$ (\$24M).

It would however introduce unnecessary (and potentially restrictive) precision into inherently uncertain data if one were to judge these outcomes based on their absolute value. It is therefore recommended to rather consider CV values falling within a bin range to be similar and then to add further differentiation by considering improvements in NPV_{AVE} to indicate more favourable solutions. So for instance it can be seen from Figure 5.18 that the lowest CV values lie in a range between 0.5 and 0.6. Since all three solutions (29, 39 and 40) lie within a similar CV range but solutions 39 and 40 have as much higher higher $UNPV_{AVE}$

(close to 1.0) compared to solution 29 with an $UNPV_{AVE}$ around 0.8, solution 39 and 40 can be interpreted as more favourable than 29.

A further important consideration which influences the selection from available options is the 'risk appetite' of an investor. For the sake of simplicity investors can be grouped into three categories: risk-neutral, risk-averse and risk takers. Risk-neutral investors will prefer to choose options which have a balanced trade-off between risk and return (i.e. lowest $UNPV_{CV}$ values). Risk-averse (or conservative) investor will tend to choose 'safer' lower risk options even at the cost of sacrificing potential additional value (i.e. lower $UNPV_{STD}$ values). Risk takers or speculative investors will tend to be prepared to take additional risk on board if it is accompanied with potential additional value (i.e. higher $UNPV_{AVE}$).

In terms of the defined options, the most preferred option for a risk-neutral investor would be schedule 29 with the lowest CV value.

Note that as discussed above, in the figure there are a number of schedules (e.g. 38, 39 and 40) which have higher $UNPV_{AVE}$ than the current most favourable schedule (29) but these also represent higher risk options at $UNPV_{CV}$ values above 0.53. An investor with a high tolerance for risk might nevertheless opt to choose such higher-value/higher-risk values although their risk-reward trade off is not as favourable as the discussed most attractive solutions.

Schedule 16, which ranked only 21st most favourable option with a $UNPV_{CV}$ value 0f 0.83, has the lowest overall $UNPV_{STD}$ value and therefore represents the lowest risk option of all 40 options. A conservative or risk-averse investor might prefer to select this lower risk option and be prepared to sacrifice the potential additional value when compared to other higher value/higher risk solutions. A risk averse investor will however not be prudent in choosing solutions with values lower than schedule 16 since since the accompanying reduction in value (compared to schedule 16) does not add any additional reduction in risk (in fact the risk increases).

Note also that Schedule 34 has both the highest overall risk ($UNPV_{STD}$) and the worst risk-return ratio ($UNPV_{CV}$). This would thus be the worst option to consider.

One of the unfortunate pitfalls of using this framework is that it is dependant on the application of the average and standard deviation which in turn is based on the assumption that the information is normally distributed (i.e. following a Gaussian probability function). Although it can be argued that as the number of simulations considered increases, the distributions will tend to approach normality as the dynamics of the Central Limit Theorem has its effect. This is clearly not the case when we only use the 40 simulations in the case study. From Figure 5.11 and subsequent graphs, it is clear that the distributions of values are non-symmetrical, in most cases displaying a higher upside than downside 'tail'.

It would be tempting to consider Figure 5.11 and identify schedule 34 (or even 18) as highly attractive due to its large perceived upside potential from an absolute value point of view. A different vantage point - as proposed by Dimitrakopoulos et al. (2007) - would be to consider the upside potential (NPV_{MAX} - NPV_{AVE}) as a proportion of the total range (NPV_{MAX} - NPV_{MIN}). Selecting schedules which produce the highest upside ratios would then be the most attractive. Note that this approach will also automatically lend priority to schedules that simultaneously minimise the downside (NPV_{AVE} - NPV_{MIN}).

In Figure 5.19 the upside potential ratio is showed from most to least attractive indicating that Schedule 34 had the most favourable upside potential as mentioned earlier.

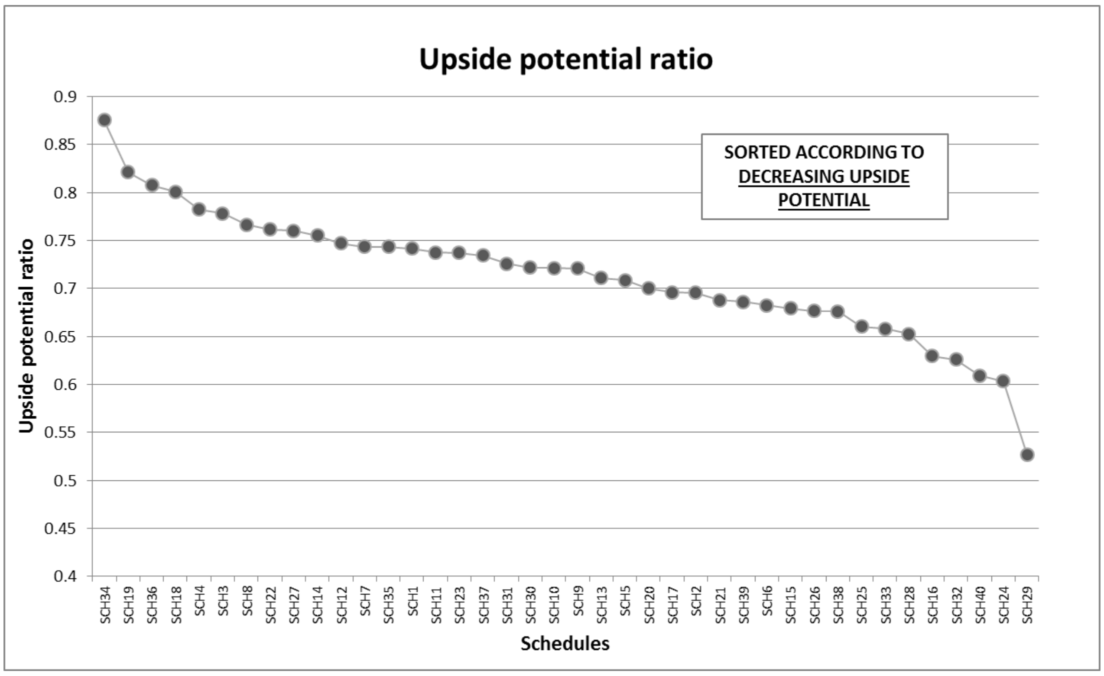


Figure 5.19: Upside potential ratio

Our main reservation in using the upside potential ratio as an indicator of attractiveness of a schedule is the fact that the ratio is highly dependent on the 'Maximum' data statistic per schedule which impose a disproportionally high reliance on potential outliers and extreme values.

5.5 Conclusion

This chapter has built on previous research ((Dimitrakopoulos et al., 2007)) into the use of conditional simulation as a means of introducing grade uncertainty into open pit production scheduling. Some key findings are as follows:

(a) considering the attractiveness of schedule options from a value and spread (NPV_{MAX} - NPV_{MIN}) perspective independently produces inconsistent or conflicting outcomes.

(b) we add a significant contribution by introducing the coefficient of variation (CV) in a novel application as a proposed method to combine value and risk into a metric which can be used to compare the relative units of risk against value.

(c) conversion of value and risk ranges to unit values between 0 and 1 affords the opportunity to generate a risk v.s. return plot which illustrates the interaction between value and uncertainty.

(d) another new contribution is the generations of a risk-return plot which, with its $UNPV_{CV}$ contours, is a very useful, novel interpretive framework that can be used to identify the most attractive schedule(s) with additional consideration of an investor's risk appetite.

(e) a potential caveat with this method is its use of the standard deviation which assumes normally distributed values. This can be minimised by including as many simulations as available in order to benefit from the Central Limit Theorem which can be expected to lead to increasingly Gaussian behaviour of the distribution as the number of instances (simulations) increase.

The next chapter will introduce two alternative methods for considering grade uncertainty by using Scenario-based approaches. This approach will use information (objective values) derived from optimising multiple simulations, but will provide a unique solution (schedule) that either maximises value or minimises risk.

Chapter 6

Scenario Optimisation

6.1 Introduction

In this chapter we employ alternative approaches to introduce grade uncertainty in the open pit schedule optimisation by means of Scenario Optimisation. As in the previous chapters, we utilise Conditional Simulation data, but instead of deriving an optimisation tied to each of the Conditional Simulations, we use information related to the optimised solutions associated with individual simulations to define two scenario optimisation formulations. This allows us to generate two new 'unique' schedules (candidate solutions). We then test these solutions against the data related to the 40 simulations and use the interpretive framework (using the CV) developed in the previous chapter to assess whether these unique solutions lead to improvements in CV, in other words, more attractive alternatives. As before, we use the MILP optimisation approach developed in Chapter 4. This chapter is arranged as follows: Section 6.2 defines key concepts related to Scenario Optimisation, building on the research of Dembo (1992).

Section 6.3 looks at two alternative ways of formulating and solving a scenario optimisation formulation: in the first case we derive a schedule which maximises the sum of the average objective values over all the Conditional Simulations. In the second approach we minimise the sum of the absolute differences of objective values from the new schedule against those of the underlying Conditional Simulations, along similar lines to Dembo (1992).

Section 6.4 generates two different optimisations for the case study developed in the previous chapter using the strategies defined in the previous section. We tests the two scenario optimisation results (candidate schedules)against other solutions generated in the previous chapter and uses the Interpretive Framework to assess their relative attractiveness.

The chapter concludes in Section 6.5 with findings and recommendations as to how well the Scenario Optimisation approach improves the selection of improved candidate schedules when compared to Conditional Simulation results.

6.2 Preliminaries

In the previous chapter we developed an understanding of inherent data uncertainty by evaluating 40 optimised mining schedules (candidate solutions) against the 40 underlying sets of simulation data. This generated a large output of potential outcomes with their respective NPVs. We made sense of these results by conducting basic statistics on the outcomes and considering the average values and standard deviation associated with the data.

In this chapter we consider an alternative two-stage approach to incorporate uncertainty. Firstly we generate optimised schedules (scenarios) for the 40 conditional simulations as before (stage 1), but instead of assessing schedules against the underlying data, we use the set of 40 NPVs generated during the MILP optimisations of the simulations, and then solve a separate scenario optimisation (stage 2) which satisfies a secondary objective. Using this process we derive two additional unique schedules (candidate solutions) which satisfy one of two objectives respectively. We then evaluate the two new solutions against the underlying set of conditional simulation data in order to assess and compare the two solutions with the 40 results generated directly from the conditional simulations.

The difference in the two approaches is illustrated in the figure below.

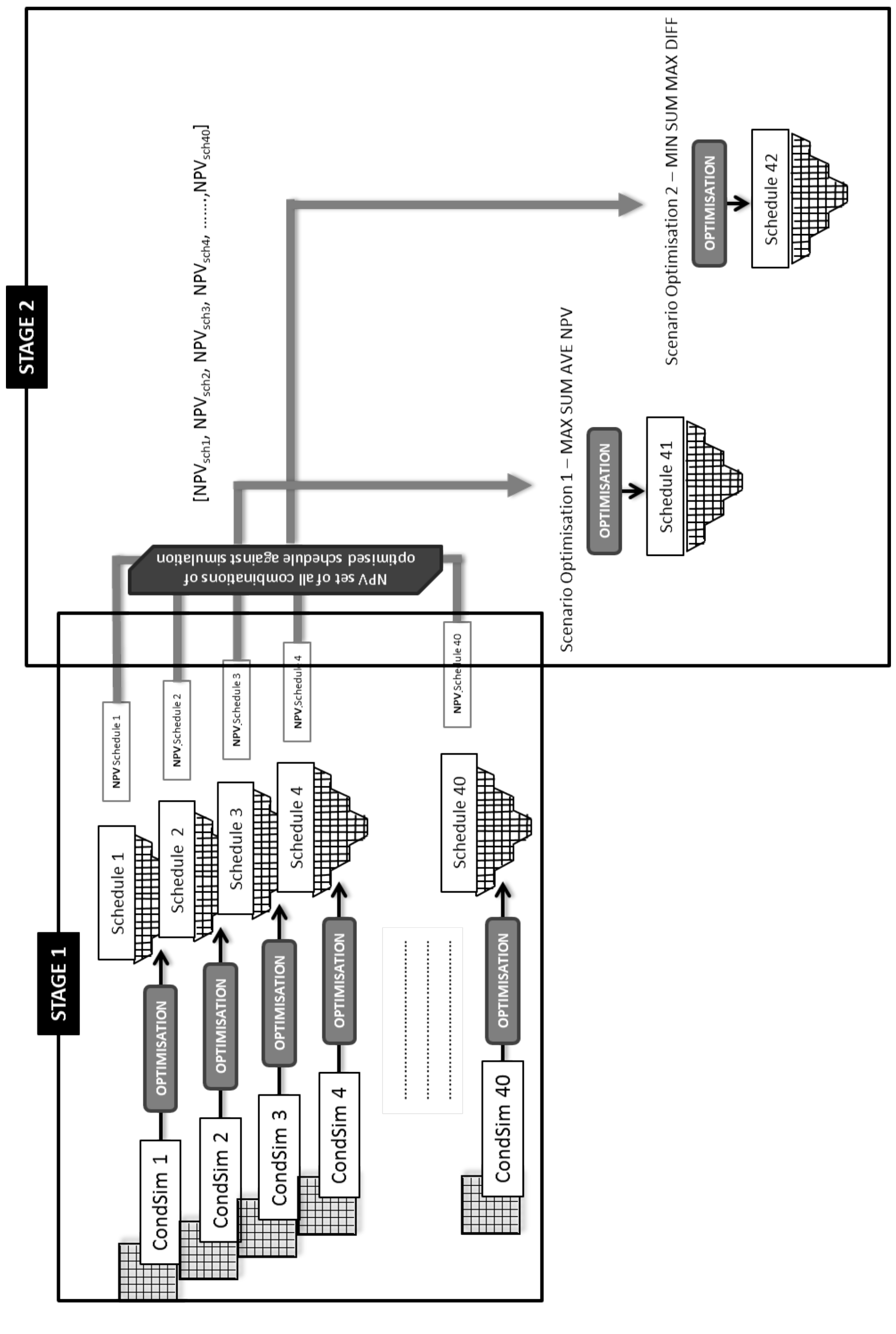


Figure 6.1: Schematic of Scenario Optimisation showing Stages 1 and 2

The scenario optimisation concepts developed in this chapter are based on ideas researched by Dembo (1992) and applied to the fields of portfolio optimisation and hydroelectric power generating system optimisation (reservoir planning and scheduling).

Dembo argues that the limited appeal of many of the approaches in the stochastic programming literature are due to their heavy data and computational burden, their frequent intractability, as well as the difficulty in understanding the models.

Saavedra Rosas (2009) also utilises a scenario optimisation approach for a mining scheduling problem but solves it using a heuristic procedure based on a genetic algorithm. The resultant solution although fast, cannot guarantee optimality.

Dembo presents a simple linear programming approach to solving a stochastic model, based on a particular method for combining such scenario (simulation) solutions into a single, feasible strategy. The approach which is computationally simple and comprehensible, can be applied within a model subject to multiple objectives.

6.2.1 Formulation using Dembo's approach

Dembo notes that if we consider linear optimisation problems of the form shown in equation (6.1) with constraints as in equation (6.2) and equation (6.3), then a portion of the data underlying the formulation may be uncertain in many cases due to limited information.

$$(\mathcal{A}) \ \min \ c^T \cdot x \tag{6.1}$$

s.t.

$$A \cdot x = b \tag{6.2}$$

$$x \geq 0 \tag{6.3}$$

Such a case could then be reformulated as a stochastic linear program as shown in equation (6.4) with certain (deterministic) constraints (equation 6.9) and uncertain constraints (equation 6.10).

$$(\mathcal{B}) \ \min \ c_u^T \cdot x \tag{6.4}$$

s.t.

$$A_d \cdot x = b_d \tag{6.5}$$

$$A_u \cdot x = b_u \tag{6.6}$$

$$x \geq 0 \tag{6.7}$$

Dembo argues that a convenient and intuitive means of incorporating uncertainty into the optimisation is by using scenarios which in this case are defined as a particular set of realizations (or simulations) of uncertain data c_u, A_u and b_u represented as c_s, A_s and b_s. The solution to the problem for a particular scenario is denoted by x_s. For each scenario (s) in the set of all scenarios (S) i.e. $s \in S$, the problem reduces to a deterministic sub-problem such as:

$$(\mathcal{D}) \ \min \ c_s^T \cdot x \tag{6.8}$$

s.t.

$$A_d \cdot x = b_d \tag{6.9}$$

$$A_s \cdot x = b_s \tag{6.10}$$

$$x \geq 0 \tag{6.11}$$

In Dembo's formulation there is a specific probability (p) associated with each scenario (p_s) which might be different from one scenario to another, or changing over time and which is assumed to be given.

In their approach the objective is to find a way to combine the solutions from a number of different scenarios (simulations) into a form that provides a reasonable solution to the underlying stochastic problem. 'Reasonable' or 'feasible' is defined through the following formulation:

$$A_s x = b_s, \ s \in S; \tag{6.12}$$

$$A_d x = b_d; \tag{6.13}$$

$$x \geq 0 \tag{6.14}$$

Then the solution $\hat{x}$ to the stochastic linear system will be 'feasible' if it satisfies the deterministic constraints:

$$A_d \hat{x} = b_d, \tag{6.15}$$

$$\hat{x} \geq 0 \tag{6.16}$$

and minimizes:

$$\sum_{s}^{S} p_s \| A_s x - b_s \| \tag{6.17}$$

Dembo's approach includes two stages. During the first stage deterministic solutions are computed for all scenarios (simulations). This is followed by a subsequent stage during which a coordinating model is solved to find a single unique feasible policy (see Figure 6.1).

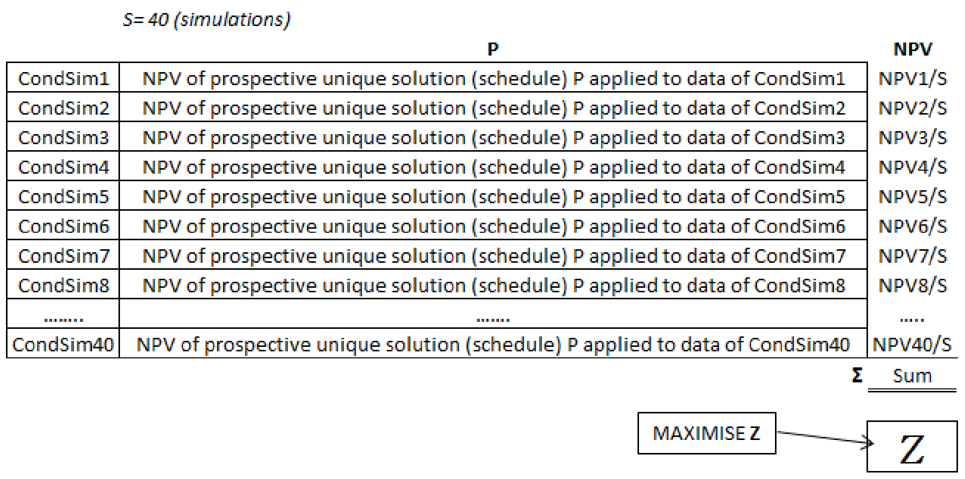
In the following section we will be formulating a stochastic problem based on the foundations of Dembo's approach.

6.3 Our Scenario Optimisation Formulations

We have developed two scenario-based formulations. The first is done in a single stage and is based on considering the combined data from all the Conditional Simulations and finding a unique solution which maximises the sum of the average NPV's. The second formulation is developed in two stages along the themes of Dembo's paper as discussed in the previous section.

6.3.1 Scenario maximisation formulation

In this formulation we consider the data (block values) of all the conditional simulations together and find the solution that maximises the sum of the average values of the solution over all the simulations. The formulation is similar to the original MILP that we have defined in Chapter 3. The key difference is that whereas the formulation was previously applied on individual Conditional Simulation data sets, this formulation has an additional summation over all the Conditional Simulations and divides each solution by the probability of its occurrence (which is assumed as proportional to the number of Conditional Simulations, or $\frac{1}{S}$)- see Figure 6.2.



S= 40 (simulations)

	P	NPV
CondSim1	NPV of prospective unique solution (schedule) P applied to data of CondSim1	NPV1/S
CondSim2	NPV of prospective unique solution (schedule) P applied to data of CondSim2	NPV2/S
CondSim3	NPV of prospective unique solution (schedule) P applied to data of CondSim3	NPV3/S
CondSim4	NPV of prospective unique solution (schedule) P applied to data of CondSim4	NPV4/S
CondSim5	NPV of prospective unique solution (schedule) P applied to data of CondSim5	NPV5/S
CondSim6	NPV of prospective unique solution (schedule) P applied to data of CondSim6	NPV6/S
CondSim7	NPV of prospective unique solution (schedule) P applied to data of CondSim7	NPV7/S
CondSim8	NPV of prospective unique solution (schedule) P applied to data of CondSim8	NPV8/S
....	....	......
CondSim40	NPV of prospective unique solution (schedule) P applied to data of CondSim40	NPV40/S
	Σ	Sum

Figure 6.2: Scenario maximisation

In the illustration (Figure 6.2) a unique solution (P) is found by considering the maximisation of the solution against 40 simulated datasets. The solution values are divided by a constant S (40 in our case) representing the probability of each data set occurring. By maximising the summation (Z) over the 40 data instances a unique solution is obtained that attempts to maximise average value.

The formulation is as follows:

$$(\mathcal{P}) \quad \max \quad Z = \sum_{s=1}^{S} \sum_{t=1}^{T} \sum_{b=1}^{B} \frac{1}{S} \cdot x^{b,t,s} \cdot v^{b,s} \cdot d^{t,s} \tag{6.18}$$

s.t.

$$\sum_{t=1}^{T} x^{b,t,s} \leq 1 \qquad \forall b \in B, \forall s \in S \tag{6.19}$$

$$\sum_{b=1}^{B} x^{b,t,s} \cdot MT^{b,s} \leq MMC \qquad \forall t \in T, \forall s \in S \tag{6.20}$$

$$\sum_{t=1}^{T} x^{\tilde{b},t,s} \cdot PT^{p,s} \leq MPC \qquad \forall \tilde{b} \in B, \forall s \in S \tag{6.21}$$

$$\sum_{t=1}^{k} x^{p,t} \leq \sum_{t=1}^{k} x^{j,t} \qquad \forall (p,j) \in E, \forall k \in T, \forall s \in S \tag{6.22}$$

$$x^{b,t,s} \in \{0,1\} \qquad \forall b \in B, \forall t \in T, \forall s \in S \tag{6.23}$$

$$\tag{6.24}$$

$T = \{1, \ldots, T_{\max}\}, S = \{1, \ldots, S_{\max}\}, B = \{1, \ldots, B_{\max}\}$

Note that by removing the constant $\frac{1}{S}$ we do not impact the solution of the model. The code for this approach can be found in the Appendix under Code Snippet B.

In the following sections, this formulation was applied on the set of 40 Conditional Simulations that was used in the previous sections.

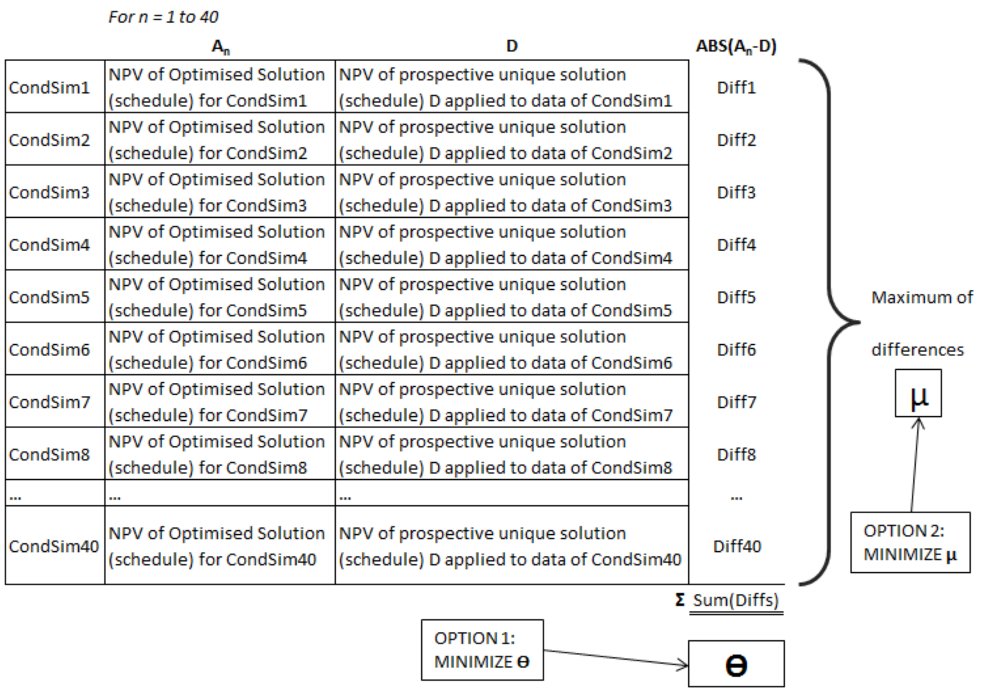
6.3.2 Scenario minimisation formulation

In this formulation we follow the two-stage procedure as defined in the previous section (see also Figure 6.1).

STAGE 1: We firstly use the deterministic MILP formulation defined in previous chapters to generate optimised solutions for each of the 40 scenarios (simulations)in our case study. This enables us to generate a series (list) of 40 optimised NPV's related to each scenario. Note that from the information generated during stage 1 we only require the actual optimised values and not the schedules per se.

STAGE 2: During stage two we use the information from the previous stage related to each scenario (b_s) in order to find a unique solution ($\hat{x}$) that minimises the maximum deviations between the current solution and the tracking values (b_s). The probability (p_s) associated with each scenario is the same i.e. all scenarios are equi-probable and has a value equal to $\frac{1}{n}$ where n is the number of scenarios (simulations) considered.

This approach is illustrated in Figure 6.3.



For n = 1 to 40

	A_n	D	ABS(A_n-D)
CondSim1	NPV of Optimised Solution (schedule) for CondSim1	NPV of prospective unique solution (schedule) D applied to data of CondSim1	Diff1
CondSim2	NPV of Optimised Solution (schedule) for CondSim2	NPV of prospective unique solution (schedule) D applied to data of CondSim2	Diff2
CondSim3	NPV of Optimised Solution (schedule) for CondSim3	NPV of prospective unique solution (schedule) D applied to data of CondSim3	Diff3
CondSim4	NPV of Optimised Solution (schedule) for CondSim4	NPV of prospective unique solution (schedule) D applied to data of CondSim4	Diff4
CondSim5	NPV of Optimised Solution (schedule) for CondSim5	NPV of prospective unique solution (schedule) D applied to data of CondSim5	Diff5
CondSim6	NPV of Optimised Solution (schedule) for CondSim6	NPV of prospective unique solution (schedule) D applied to data of CondSim6	Diff6
CondSim7	NPV of Optimised Solution (schedule) for CondSim7	NPV of prospective unique solution (schedule) D applied to data of CondSim7	Diff7
CondSim8	NPV of Optimised Solution (schedule) for CondSim8	NPV of prospective unique solution (schedule) D applied to data of CondSim8	Diff8
...	...	...	...
CondSim40	NPV of Optimised Solution (schedule) for CondSim40	NPV of prospective unique solution (schedule) D applied to data of CondSim40	Diff40

Figure 6.3: Two-stage scenario minimisation

Figure 6.3 shows the column (under A_n) of optimum solutions generated during stage 1 during which an MILP optimisation is run on each of the 40 conditional simulation datasets. As described previously the MILP formulation maximises the objective function (NPV's) for each dataset.

During stage 2 a unique solution is derived which minimises the sum of the absolute differences between the optimised values derived in stage 1 and the unique solution (D) when applied to the equivalent data sets. Stage 2 can be solved by either minimising the sum of the absolute differences (Option 1 - θ in Figure 6.3) or by minimising the maximum absolute difference (Option 2 - μ in Figure 6.3) based on the euclidean norm.

In our formulation we have modified the function in order to utilise the euclidean norm (μ) i.e.:

$$(\mathcal{D}_\infty)\ min\ \ \mu \tag{6.25}$$

s.t.

$$\sum_{t=1}^{T} x^{b,t} \leq 1 \qquad \forall b \in \{1,\ldots,B\} \tag{6.26}$$

$$\sum_{b=1}^{B} x^{b,t} \cdot MT^{b,s} \leq MMC \qquad \forall t \in \{1,\ldots,T\}, \forall s \in \{1,\ldots,S\} \tag{6.27}$$

$$\sum_{t=1}^{T} x^{\tilde{b},t} \cdot PT^{p,s} \leq MPC \qquad \forall \tilde{b} \in \{1,\ldots,B\}, \forall s \in \{1,\ldots,S\} \tag{6.28}$$

$$\sum_{t=1}^{k} x^{p,t} \leq \sum_{t=1}^{k} x^{j,t} \qquad \forall (p,j) \in E, \forall k \in \{1,\ldots,T\} \tag{6.29}$$

$$x^{b,t} \in \{0,1\} \qquad \forall b \in \{1,\ldots,B\}, \forall t \in \{1,\ldots,T\} \tag{6.30}$$

$$b_s - \sum_{t=1}^{T}\sum_{b=1}^{B} x^{b,t} \cdot v^b \cdot d^t \leq \mu \qquad \forall b \in \{1,\ldots,B\}, \forall s \in \{1,\ldots,S\} \tag{6.31}$$

$$\sum_{t=1}^{T}\sum_{b=1}^{B} x^{b,t} \cdot v^b \cdot d^t - b_s \geq -\mu \qquad \forall b \in \{1,\ldots,B\}, \forall s \in \{1,\ldots,S\} \tag{6.32}$$

The addition of constraints (6.31) and (6.32) ensure that a solution is obtained which attempts to minimise the deviations from the original solutions derived directly from the conditional simulations.

The implementation code for this approach can be found in the Appendix under Code Snippet C.

6.4 Case study

Figure 6.4 is a repeat of the composite figure showed in Chapter 5 plus the two scenario optimisation results added for comparison. The two scenario optimisation schedules are a lot more ‘average’ and less scattered than many of the results which originate from optimising the individual conditional simulations.

The two scenario optimisation results (41 and 42) are introduced as additional candidate solutions and tested against the 40 conditional simulation data sets in a similar fashion as followed in Chapter 5. The resultant mean value and standard deviation are derived for each solution and converted to unit values following the same procedures as was used in Chapter 5. The results are plotted on the interpreted framework for comparison with the other 40 solutions (see Figure 6.5).

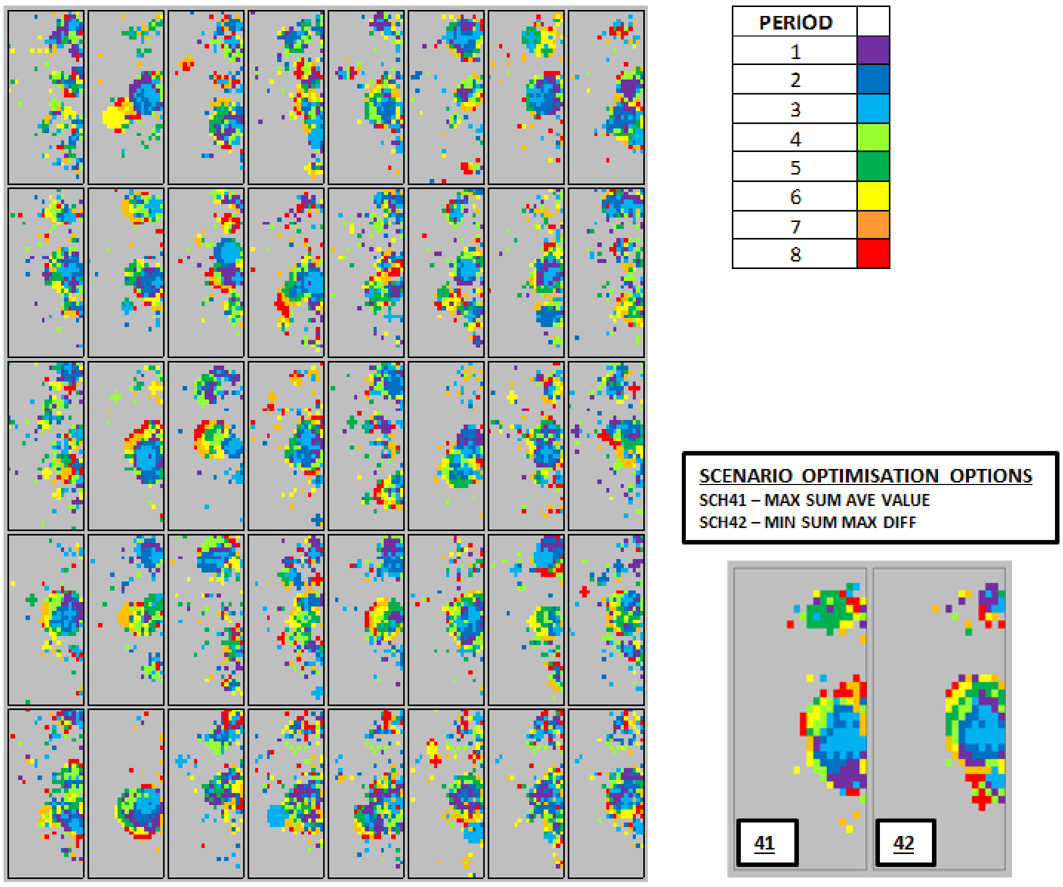


Figure 6.4: Schedule optimisations

	AVE	STD	CV
SCH29	0.64	0.41	0.64
SCH41	1.00	0.67	0.67
SCH40	0.81	0.55	0.68
SCH39	0.83	0.57	0.68
SCH26	0.50	0.39	0.79
SCH38	0.64	0.53	0.83
SCH11	0.52	0.44	0.83
SCH15	0.55	0.47	0.84
SCH25	0.63	0.55	0.87
SCH20	0.67	0.59	0.88
SCH33	0.42	0.37	0.89
SCH42	0.85	0.76	0.90
SCH31	0.59	0.54	0.93
SCH4	0.43	0.40	0.94
SCH30	0.69	0.65	0.94
SCH9	0.61	0.58	0.95
SCH5	0.55	0.53	0.95
SCH24	0.39	0.39	0.98
SCH28	0.36	0.35	0.98
SCH35	0.67	0.66	0.98
SCH6	0.45	0.44	0.99
SCH32	0.41	0.41	1.00
SCH16	0.31	0.31	1.00
SCH7	0.57	0.57	1.00
SCH10	0.59	0.59	1.01
SCH18	0.80	0.82	1.03
SCH1	0.32	0.33	1.04
SCH23	0.52	0.54	1.04
SCH13	0.34	0.36	1.05
SCH8	0.54	0.57	1.06
SCH21	0.33	0.36	1.07
SCH12	0.58	0.63	1.08
SCH2	0.48	0.53	1.10
SCH14	0.45	0.50	1.11
SCH37	0.42	0.48	1.13
SCH17	0.27	0.33	1.21
SCH22	0.52	0.65	1.24
SCH3	0.33	0.44	1.31
SCH36	0.34	0.56	1.65
SCH27	0.22	0.40	1.77
SCH19	0.28	0.56	1.98
SCH34	0.47	1.00	2.12

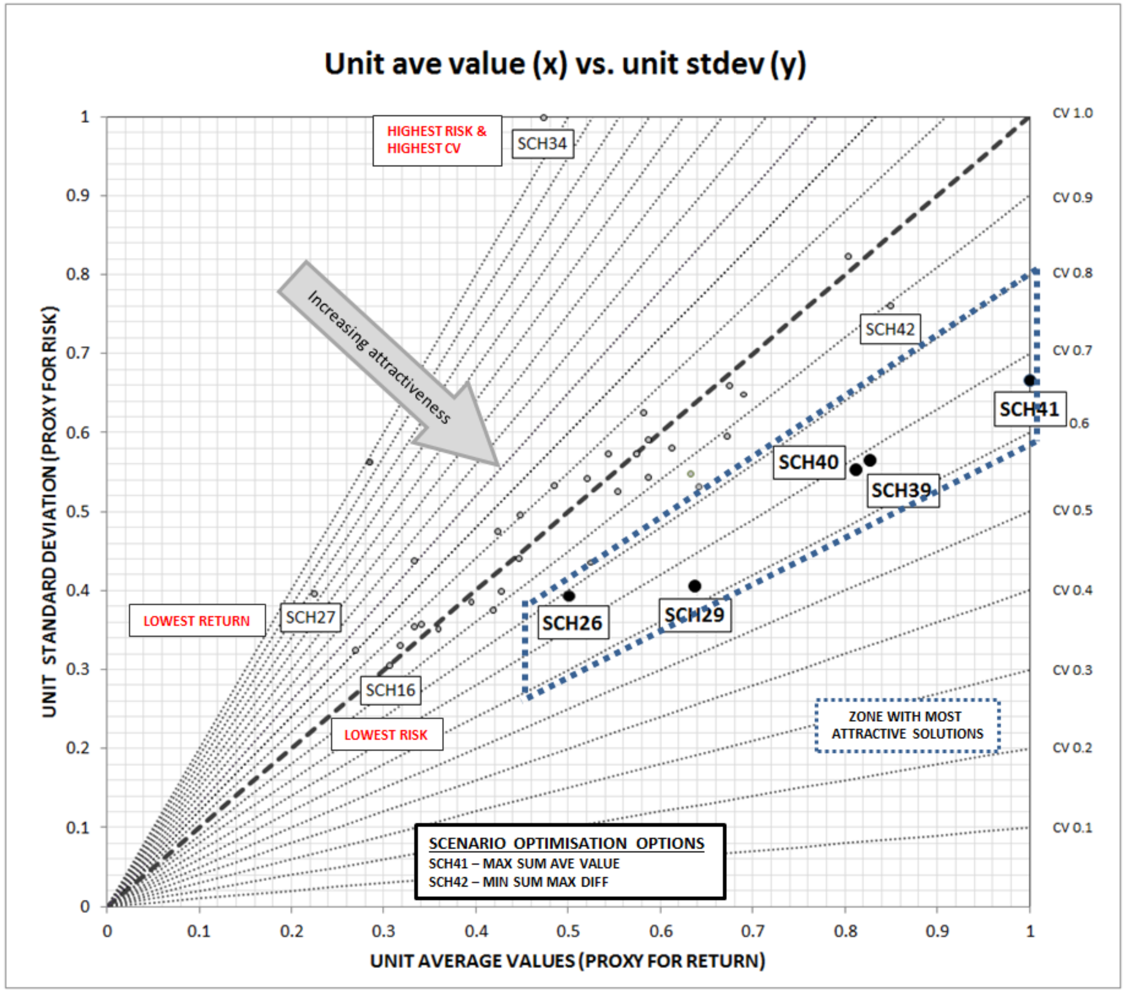


Figure 6.5: Results

This figure should be seen as supplementary to Figure 5.18 in Chapter 5 with the only difference being that the two additional candidate solutions have been added. Note however that the CV ranges have changed with the most favourable solutions being between 0.6 and 0.8 whereas in Chapter 5 the best solutions fell between 0.5 and 0.7. The reason for this difference in values is that solution 41 now introduces a new maximum average value which leads to all the values being divided by a new higher constant value (in order to generate unit average values) with a resultant shift in all values in the horizontal axis.

We argue that it would introduce unnecessary precision into inherently uncertain data if one was to judge these outcomes based on their absolute values and we therefore recommend considering CV values falling within a range to be similar and then to add further differentiation by considering improvements in NPV_{AVE} to indicate more favourable solutions.

It is evident that solution 41 leads to an improved result with a CV in the range 0.6 to 0.7 but significantly higher NPV_{AVE} value.

Solution 42 however does not lead to a noticeably improved result since although its NPV_{AVE} fall within the higher ranges, it is also accompanied by a high NPV_{STDEV} value and therefore unattractive CV (close to 0.9).

6.5 Conclusions

In this chapter we utilise two scenario optimisation approaches to add two unique candidate solutions to the 40 solutions previously produced. The scenario optimisation formulations are based on the data and solutions generated during the previous conditional simulation approach (Chapter 5).

Some key findings in this chapter are as follows:

(a) the first variety of scenario optimisation developed using one stage average value maximisation objective generated a solution which, upon testing against the simulation data, showed a favourable and improved solution (high average value, low standard deviation, low CV) which equals or surpasses previous most favourable solutions.

(b) the second variety of scenario optimisation developed using a two stage average value variability minimisation objective generated a solution which, upon testing against the simulation data, provided a solution which although relatively high in average value also incorporated a high standard deviation and therefore a high CV value. For this specific case, this approach therefore did not provide a favourable or improved solution compared to previous most favourable solutions. This approach nevertheless merits further research and application in generating unique solutions from multiple scenario data or multiple conditional simulations.

(c) through this research we managed to construct two formulations which produce unique exact solutions using conditional simulations results as inputs.

The next chapter will summarise the conclusions from the thesis and make recommendations for further research building on these ideas.

Chapter 7

Conclusions and Recommendations

In this chapter we provide a summary of the findings derived from individual chapters of the thesis and then conclude with a number of recommendations for further study and research.

7.1 Conclusions

The framework for this thesis has been as follows:

(a) In Chapter 1 we briefly described the problem and key issues that we addressed in the thesis followed by some of the main classical and recent research literature. We also provided the key objectives, motivation and methodologies followed to answer the questions raised.

(b) In Chapter 2 we provided an introduction to the various topics related the geological and mining context.

(c) In Chapter 3 we explored the existing literature related to the topics of open pit schedule optimisation and geological uncertainty modelling.

(d) In Chapter 4 we constructed a basic deterministic MILP model. We introduced a number of simple strategies to:

- remove obsolete data thereby reducing the size of the dataset
- improve the solution speed by reducing the required accuracy (MIP gap) of the solution to better suit the level of confidence in data

(e) In Chapter 5 we introduced grade uncertainty in the form of geo-statistical conditional simulations. We used the MILP formulation developed in Chapter 3 to derive optimised solutions to 40 conditional simulations. In this chapter we also developed an interpretive framework which makes sense of multiple equi-probable results (candidate solutions) and which allows comparison within the context of value (average NPV) and risk (standard deviation in NPV). This methodology can easily be scaled up to include more conditional simulations (we only used 40) and larger block model sizes. This framework therefore represents a truly practical tool, useful for various applications.

(f) In Chapter 6 we proposed two alternative methods to define the problem and produce unique alternative candidate solutions.

7.1.1 MILP formulation

In Chapter 4 we developed a basic formulation of a MILP for an open pit scheduling problem based on previous research which we tested on a case study. We then introduced two areas where simplistic algorithms or small enhancements can have a large impact on the solution time and quality of an optimisation.

The two areas focussed on were data preprocessing using a two-phased approach including rectangular trimming and Ultimate Pit trimming, and 'fine-tuning' of the MIP gap setting in CLPEX. These two considerations we incorporated in the standard formulation and solving of all subsequent models in this thesis.

The size of case studies were kept small since the objective of the thesis is not to include all the latest mathematical modelling improvements into the formulation but rather to use easily accessible enhancements to establish a basic model for subsequent inclusion of parameter uncertainty, which is the main area of focus in the thesis.

In Chapter 4 we demonstrated that a basic MILP model can be readily developed using simplistic tools to reduce the underlying superfluous data and relax the solution accuracy expectations to reflect the inherent underlying geological uncertainty. This MILP formulation was used in subsequent chapters

7.1.2 Conditional Simulation and Interpretive Framework

Chapter 5 expanded on previous research into the use of conditional simulation as a means of introducing grade uncertainty into open pit production scheduling. Key findings from this chapter are as follows:

(a) considering the attractiveness of schedule options from a value and spread (NPV_{MAX} - NPV_{MIN}) perspective independently produces inconsistent or conflicting outcomes leaving a decision maker with a potential dilemma in choosing a preferred solution

(b) the coefficient of variation (CV) provides a useful ratio to combine value and risk into a metric which can be used to compare the relative trade off between risk and value.

(c) conversion of value and risk ranges to unit values between 0 and 1 affords the opportunity to generate a standardised risk v.s. return plot which illustrates the interaction between value and uncertainty.

(d) the risk-return plot with its $UNPV_{CV}$ contours is a very useful interpretive framework that can be used to identify the most attractive schedule(s) with additional consideration of an investor's risk appetite.

(e) a potential caveat with this method is its use of the standard deviation which assumes normally distributed values. The effect of this assumption can be minimised by including as many simulations as available in order to benefit from the Central Limit Theorem which can be expected to lead to increasingly Gaussian behaviour of the distribution as the number of instances (simulations) increase.

7.1.3 Scenario Optimisation

In order to try to identify a single unique solution which incorporates both value and risk considerations, a number of scenario optimisation approaches were developed and tested in Chapter 6.

Some key findings in this chapter are as follows:

(a) the first variety of scenario optimisation developed using a single step average value maximisation objective generated a solution which, upon testing against the simulation data, showed a favourable and improved solution (high average value, low standard deviation, low CV) which equals or surpasses previous most favourable solutions.

(b) the second variety of scenario optimisation developed using a two stage average value variability minimisation objective generated a solution which, upon testing against the simulation data, provided a solution which although relatively high in average value also incorporated a high standard deviation and therefore a high CV value. This approach therefore did not provide a favourable or improved solution compared to previous most favourable solutions.

(c) Both scenario optimisation approaches merit further investigation and application on more varied and different sized case studies.

7.2 Recommendations for further research

The following recommendations are offered as potential areas of focus or future research:

(a) Our research and the accompanying case study was based on a relatively small example but much larger models can be solved by expanding the formulation to exploit the structure of the problem and introduce strategies to reduce the size of the solution space.

(b) The the interpretative framework developed in this thesis can be used successfully as a comparative approach to assess the attractiveness of candidate solutions derived through conditional simulation, scenario optimisation and potentially numerous other industrial applications.

(c) The investigation into the various simulations and their derived schedules are limited to the set of given conditional simulations used to generate the schedules. It would be of value to find other methods to derive unique solutions that both minimises risk and maximises value from the individual simulations.

(d) It could be argued that the statistics derived when comparing solutions (schedules) against the range of simulation values might be sensitive or dependent on the specific number of and also the sequencing (order) of introducing individual simulations. A study which systematically introduces an increasingly larger number of simulations up to the point where the resultant outcomes (statistics) are not materially influenced, will assist in identifying the 'optimum' number of simulations for the problem. This approach will also assist in generating increasingly Gaussian behaviour in the distribution statistics which will assist in supporting the use of the standard deviation as a viable metric. This 'optimum' number of simulations might however possibly be problem specific.

(e) We saw that scenario optimisation approaches can be utilised to formulate alternative configurations of the basic schedule optimisation problem for arriving at candidate solutions. However, the findings related to the scenario optimisation approach are based on a single case study, which although supported by 40 simulations, is still dependant on the dynamics of a single grade model. In order to further assess the outcomes of scenario optimisation as a general mechanism for generating candidate solutions further case studies on different geological model types need to be completed using the same approach and using the same interpretive framework for assessing viability.

Within this thesis a number of mechanisms have been used to generate candidate solutions. A candidate solution can be defined as a valid solution (i.e. the result of an optimisation complying with precedence and maximum capacity constraints) but which might not be the most attractive solution and has to be 'tested' using the interpretive framework which we developed in order to assess its attractiveness in terms of value and risk. In Chapter 5 we used the mechanism of deriving the optimum solutions for each of the 40 conditional simulation data sets. In Chapter 6 we developed two additional formulations which generated two additional candidate solutions. During this exercise we tested a number of ideas which merits mentioning here as areas of potential future research since they did lead to interesting results. The two key areas of further research are firstly around the E-type and secondly around various types of data 'perturbations'.

The E-Type data set is an 'average' set which is derived by calculating the geometric means of the values on a block level across all the conditional simulations. It is often accepted in geo-statistical circles that the E-type model approximates the deterministically derived Kriged model. As part of exploratory work done on testing various approaches to developing candidate solutions, we calculated the E-type model from the 40 conditional simulations and used it as input into the

MILP optimisation and subsequent evaluation using the interpretive framework. Exploratory experimentation on a single case study has indicated that the E-type for a set of conditional simulations (the block average over all of the simulations) leads to potentially attractive candidate solutions, which outperforms (in terms of CV value) most if not all of the candidate solutions derived directly from optimum solutions of the individual conditional simulations. We recommend that further research and analysis be done on the E-type, also in comparison with the Kriged value.

Perturbations (on a data set) are created by applying a small factor which changes the true block values with a factor within a percentage range (say plus-minus 5%). We 'perturbed' the E-type model as well as to 40 original conditional simulations and used these as inputs into the MILP optimisation and subsequent evaluation using the interpretive framework. See for instance Figure 7.1 for graphic representation of 30 perturbations done on a data set and the resultant effect on the solution and CV values.

	Val	Min
Original	0.400	0.400
Pert1	0.405	0.400
Pert2	0.412	0.400
Pert3	0.411	0.400
Pert4	0.404	0.400
Pert5	0.399	0.399
Pert6	0.405	0.399
Pert7	0.405	0.399
Pert8	0.404	0.399
Pert9	0.398	0.398
Pert10	0.392	0.392
Pert11	0.397	0.392
Pert12	0.396	0.392
Pert13	0.403	0.392
Pert14	0.411	0.392
Pert15	0.411	0.392
Pert16	0.406	0.392
Pert17	0.407	0.392
Pert18	0.405	0.392
Pert19	0.404	0.392
Pert20	0.397	0.392
Pert21	0.401	0.392
Pert22	0.398	0.392
Pert23	0.399	0.392
Pert24	0.397	0.392
Pert25	0.399	0.392
Pert26	0.391	0.391
Pert27	0.388	0.388
Pert28	0.392	0.388
Pert29	0.399	0.388
Pert30	0.396	0.388

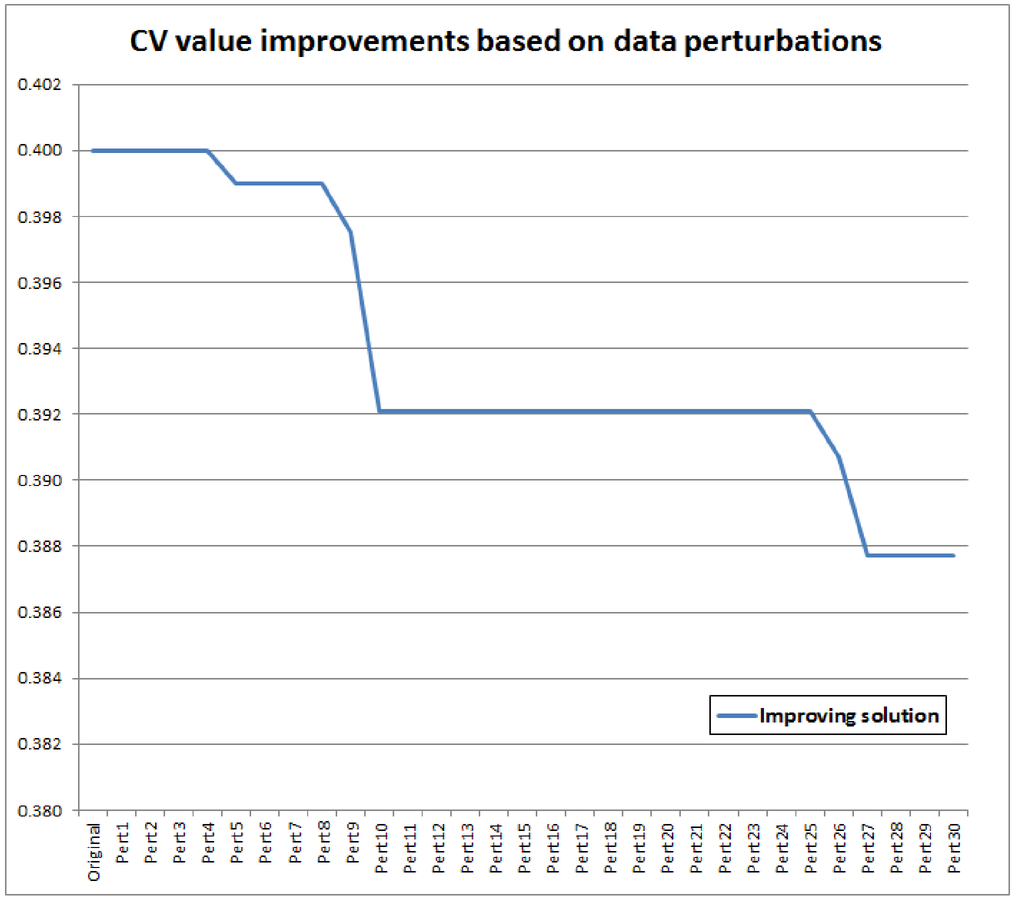


Figure 7.1: Generation of candidate solutions and testing for improved solutions using interpretive framework

Note that the procedure starts with the original CV value of one of the more attractive solutions. The algorithm can then solve the new pseudo-data sets, do comparison of the new CV values with the previous best values and only retain those solutions that lead to improvements in CV. In Figure 7.1 it can be seen that although 30 iterations with different perturbations were performed, only five instances (indicated in green) lead to improvements (reductions) in CV value.

Figure 7.2 shows the impact of improving solutions graphically in the interpretive framework.

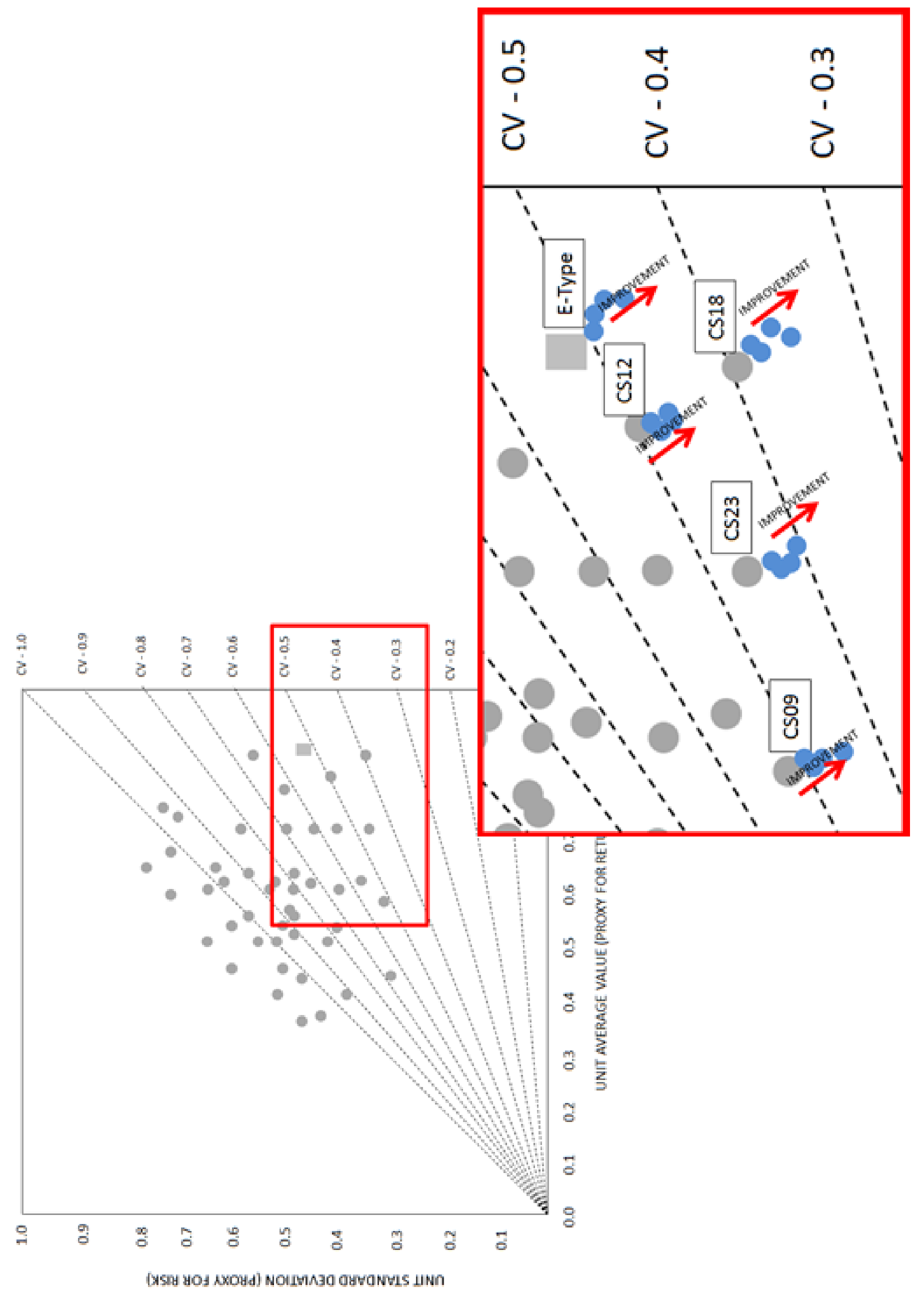


Figure 7.2: Candidate solutions generated through perturbations and testing using interpretive framework

Preliminary results indicate that this approach could lead to candidate solutions that potential outperform (in terms of CV when compared to the underlying data) any of the solutions derived from conditional simulations or the E-type itself. Such perturbations can be employed to generate an almost infinite number of potential candidate solutions. There is often a natural tendency to discredit such 'tampering' with data since it potentially dishonours the geo-statistical integrity of the model metrics (semi-variogram and population mean). However, keep in mind that this method is not intended to add to the suite of underlying data but rather to be used as a mechanism to generate candidate solutions which is only ever tested against the original 'true' conditional simulation data. The only requirement for such a mechanism is that it generates a valid solution i.e. a solution which honours the optimisation constraints (e.g. precedence and maximum capacity constraints). We recommend that this methodology be further investigated and that such perturbations be generated at various levels of variability (e.g. 2%, 5%, 10%) and then tested using the interpretive framework. One suggestion would be to develop a heuristic procedure based on an improving algorithm which generate new perturbations, find optimised solution, test the CV in the interpretive framework, and only retain the solution if it leads to an improved solution. In this fashion, new 'pseudo-data' sets (based on either selected favourable conditional simulations or the E-type) can be generated repeatedly for a predetermined number of iterations, time limit or if the exercise has lead to a predetermined percentage improvement.

Bibliography

Abdel Sabour, S., Dimitrakopoulos, R., and Kumral, M. (2008). Mine design selection under uncertainty. *Mining Technology*, 117(2):53–64.

Akbari, A., Osanloo, M., and Shirazi, M. (2008). Determination of ultimate pit limits in open mines using real option approach. *IUST International Journal of Engineering Science*, 19(5-1):23–38.

Akbari, A., Osanloo, M., and Shirazi, M. (2009). Reserve estimation of an open pit mine under price uncertainty by real option approach. *Mining Science and Technology (China)*, 19(6):709–717.

Albor Consuegra, F. and Dimitrakopoulos, R. (2009). Stochastic mine design optimisation based on simulated annealing: pit limits, production schedules, multiple orebody scenarios and sensitivity analysis. *Mining Technology*, 118(2):79–90.

Alford, C. and Hall, B. (2009). Stope optimisation tools for selection of optimum cut-off grade in underground mine design. In *Proceedings of project evaluation conference, Melbourne*, pages 137–144.

Alvarez, F., Amaya, J., Griewank, A., and Strogies, N. (2011). A continuous framework for open pit mine planning. *Mathematical Methods of Operations Research*, 73(1):29–54.

Amankwah, H. (2011). *Mathematical Optimization Models and Methods for Open-Pit Mining.* PhD thesis, Linköping.

Amaya, J., Espinoza, D., Goycoolea, M., Moreno, E., Prevost, T., and Rubio, E. (2009). A scalable approach to optimal block scheduling. In *Proceedings of APCOM*, pages 567–575.

Armstrong, M. (1990). Definition of mining parameters. *Surface Mining, 2nd Edition (ed. BA Kennedy)*, pages 459–464.

Armstrong, M. and Dowd, P. (1994). *Geostatistical simulations*, volume 7. Springer.

Armstrong, M., Galli, A., and Ndiaye, A. (2009). A case study on the impact of hedging against foreign exchange risk and energy price risk. In *Proceedings of the project evaluation conference organised by the AusIMM in Melbourne*, pages 21–22.

Askari-Nasab, H. and Awuah-Offei, K. (2009). Open pit optimisation using discounted economic block values. *Mining Technology*, 118(1):1–12.

Askari-Nasab, H., Awuah-Offei, K., and Eivazy, H. (2010). Large-scale open pit production scheduling using mixed integer linear programming. *International Journal of Mining and Mineral Engineering*, 2(3):185–214.

Askari-Nasab, H., Frimpong, S., and Szymanski, J. (2007). Modelling open pit dynamics using discrete simulation. *International Journal of Mining, Reclamation and Environment*, 21(1):35–49.

Bienstock, D. and Zuckerberg, D. (2009). A new lp algorithm for precedence constrained production scheduling. *Optimization Online.*

Bley, A., Boland, N., Fricke, C., and Froyland, G. (2010). A strengthened formulation and cutting planes for the open pit mine production scheduling problem. *Computers & Operations Research*, 37(9):1641–1647.

Boland, N., Dumitrescu, I., and Froyland, G. (2008). A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology. *Optimization Online http://www. optimization-online. org/db html/2008/10/2123. html.*

Boland, N., Dumitrescu, I., Froyland, G., and Gleixner, A. (2009). Lp-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers & Operations Research*, 36(4):1064–1089.

Bullock, R. (2011). Accuracy of feasibility study evaluations would improve accountability. *Mining Engineering*, 63(4).

Caccetta, L. and Giannini, L. (1985). On bounding techniques for the optimum pit limit problem. *Proc. AusIMM.*

Caccetta, L. and Giannini, L. (1988). An application of discrete mathematics in the design of an open pit mine. *Discrete applied mathematics*, 21(1):1–19.

Caccetta, L. and Hill, S. (2003). An application of branch and cut to open pit mine scheduling. *Journal of global optimization*, 27(2-3):349–365.

Caccetta, L., Kelsey, P., and Giannini, L. (1998). Open pit mine production scheduling. In *APCOM 98: Computer Applications in the Mineral Industries International Symposium*, pages 65–72.

Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., and Rubio, E. (2012). A new algorithm for the open-pit mine production scheduling problem. *Operations Research*, 60(3):517–528.

Chiles, J.-P. and Delfiner, P. (2009). *Geostatistics: modeling spatial uncertainty*, volume 497. John Wiley & Sons.

Clark, I. (1979). *Practical geostatistics*, volume 3. Applied Science Publishers London.

Clement, S. and Vagenas, N. (1994). Use of genetic algorithms in a mining problem. *International Journal of Surface Mining and Reclamation*, 8(4):131–136.

Cobb, B. and Charnes, J. (2007). Real options valuation. In *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, pages 173–182. IEEE Press.

Consuegra, F. and Dimitrakopoulos, R. (2010). Algorithmic approach to pushback design based on stochastic programming: method, application and comparisons. *Mining Technology*, 119(2):88–101.

Cortazar, G., Schwartz, E., and Casassus, J. (2001). Optimal exploration investments under price and geological-technical uncertainty: a real options model. *R&D Management*, 31(2):181–189.

Cullenbine, C., Wood, R., and Newman, A. (2011). A sliding time window heuristic for open pit mine block sequencing. *Optimization Letters*, 5(3):365–377.

Dagdelen, K. (2001). Open pit optimization-strategies for improving economics of mining projects through mine planning. In *International Mining Congress of Turkey*, volume 17.

Dagdelen, K. and Francois-Bongarcon, D. (1982). Towards the complete double parameterization of recovered reserves in open pit mining. In *Proceedings of 17th international APCOM symposium*, pages 288–296.

Dagdelen, K. and Johnson, T. (1986). Optimum open pit mine production scheduling by lagrangian parameterization. *Proc. 19th Sympos. APCOM, Jostens Publications, PA*, pages 127–141.

Darby-Dowman, K. and Wilson, J. (2002). Developments in linear and integer programming. *Journal of the Operational Research Society*, pages 1065–1071.

David, M., Dowd, P., and Korobov, S. (1974). Forecasting departure from planning in open pit design and grade control. In *12th Symposium on the application of computers and operations research in the mineral industries (APCOM)*, volume 2, pages F131–F142.

Davis, G. and Newman, A. (2008). Modern strategic mine planning. *Australian Mining*, 129.

Dembo, R. (1992). Scenario optimization. US Patent 5,148,365.

Deutsch, C. (2002). Encyclopedia of physical science and technology. *Academic Press*, 3:697–707.

Dimitrakopoulos, R. (1990). Conditional simulation of intrinsic random functions of orderk. *Mathematical geology*, 22(3):361–380.

Dimitrakopoulos, R. (1994). *Geostatistics for the next century: an international forum in honour of Michel David's contribution to geostatistics, Montreal, 1993*. Springer Netherlands.

Dimitrakopoulos, R. (1998). Conditional simulation algorithms for modelling orebody uncertainty in open pit optimisation. *International Journal of Surface Mining, Reclamation and Environment*, 12(4):173–179.

Dimitrakopoulos, R., Farrelly, C., and Godoy, M. (2002). Moving forward from traditional optimization: grade uncertainty and risk effects in open-pit design. *Mining Technology*, 111(1):82–88.

Dimitrakopoulos, R., Martinez, L., and Ramazan, S. (2007). A maximum upside/minimum downside approach to the traditional optimization of open pit mine design. *Journal of Mining Science*, 43(1):73–82.

Dimitrakopoulos, R. and Ramazan, S. (2004). Uncertainty based production scheduling in open pit mining. *SME Transactions*, 316.

Dincer, A. and Golosinski, T. (1993). Pit limit optimization algorithms - present status and developments. In *Symposium on the Application of Computers and Operations Research in the mineral industries (APCOM), Wollongong, NSW.*

Dowd, P. (1994). Risk assessment in reserve estimation and open-pit planning. *Transactions of the Institution of Mining and Metallurgy(Section A: Mining Industry)*, 103.

Dowd, P. and Pardo-Iguzquiza, E. (2002). The incorporation of model uncertainty in geostatistical simulation. *Geographical and Environmental Modelling*, 6(2):147–169.

Dunham, S. and Vann, J. (2007). Geometallurgy, geostatistics and project value-does your block model tell you what you need to know. In *Project evaluation conference, Melbourne, Victoria*, pages 19–20.

Elevli, B., Dagdelen, K., and Salamon, M. (1990). Single time period production scheduling of open pit. *Mining Trans*, 103:1888–1893.

Esfandiari, B., Aryanezhad, M., and Abrishamifar, S. (2004). Open pit optimisation including mineral dressing criteria using 0–1 non-linear goal programming. *Mining Technology*, 113(1):3–16.

Espinoza, D., Goycoolea, M., Moreno, E., and Newman, A. (2012). Minelib: a library of open pit mining problems. *Annals of Operations Research*, pages 1–22.

Fricke, C. (2006). *Applications of integer programming in open pit mining.* PhD thesis, University of Melbourne, Department of Mathematics and Statistics.

Froyland, G., Menabde, M., Stone, P., and Hodson, D. (2004). The value of additional drilling to open pit mining projects. *Proceedings of Orebody Modelling and Strategic Mine Planning-Uncertainty and Risk Management, Perth*, pages 169–176.

Gaupp, M. (2008). Methods for improving the tractability of the block sequencing problem for open pit mining. Technical report, DTIC Document.

Gershon, M. (1982). A linear programming approach to mine scheduling optimization. *Applications of computers and operations research in the mineral industry.*

Gershon, M. (1983). Optimal mine production scheduling: evaluation of large scale mathematical programming approaches. *International journal of mining engineering*, 1(4):315–329.

Gholamnejad, J. and Osanloo, M. (2007a). Incorporation of ore grade uncertainty into the push back design process. *Journal of the South African Institute of Mining & Metallurgy*, 107(3):177.

Gholamnejad, J. and Osanloo, M. (2007b). Using chance constrained binary integer programming in optimising long term production scheduling for open pit mine design. *Mining Technology*, 116(2):58–66.

Giannini, L., Caccetta, L., Kelsey, P., and Carras, S. (1991). Pitoptim: a new high speed network flow technique for optimum pit design facilitating rapid sensitivity analysis. In *Proceedings o Australasian Institute of Mining and Metallurgy*, volume 296, pages 57–62.

Gignac, L. (1975). Computerized ore evaluation and open pit design. In *Proc. 36th Annual Mining Symposium of the Society of Mining Engineers (AIME)*, pages 45–53.

Gleixner, A. (2008). Solving large-scale open pit mining production scheduling problems by integer programming. *Unpublished masters thesis, Technische Universität, Berlin.*

Godoy, M. and Dimitrakopoulos, R. (2004). A multi-stage approach to profitable risk management for strategic planning in open pit mines. In *Orebody Modelling and Strategic Mine Planning–Uncertainty and Risk Management International Symposium 2004*, pages 339–345. The Australasian Institute of Mining and Metallurgy.

Goovaerts, P. (1999). Geostatistics in soil science: state-of-the-art and perspectives. *Geoderma*, 89(1):1–45.

Gu, X., Wang, Q., and Ge, S. (2010). Dynamic phase-mining optimization in open-pit metal mines. *Transactions of Nonferrous Metals Society of China*, 20(10):1974–1980.

Guo, H., Zhu, K., Ding, C., and Li, L. (2010). Intelligent optimization for project scheduling of the first mining face in coal mining. *Expert Systems with Applications*, 37(2):1294–1301.

Hall, B. (2007). Evaluating all the alternatives to select the best project strategy. In *Project Evaluation Conference Melbourne*, pages 53–64.

Halton, J. (1970). A retrospective and prospective survey of the monte carlo method. *Siam review*, 12(1):1–63.

Hekmat, A., Osanloo, M., and Moarefvand, P. (2013). Block size selection with the objective of minimizing the discrepancy in real and estimated block grade. *Arabian Journal of Geosciences*, 6(1):141–155.

Hochbaum, D. and Chen, A. (2000). Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. *Operations Research*, 48(6):894–914.

Hustrulid, W. and Kuchta, M. (2006). *Open pit mine planning and design*, volume 1. Taylor & Francis.

Huttagosol, P. and Cameron, R. (1992). A computer design of ultimate pit limit by using transportation algorithm. In *Proceedings of the 23rd International. APCOM Symposium*, pages 443–460.

ILOG, I. (2012). Cplex optimization studio. *http://www-01. ibm. com/software/integration/optimization/cplex-optimization-studio/, last accessed Oct 2013.*

Jara, R., Couble, A., Emery, X., Magri, E., and Ortiz, J. (2006). Block size selection and its impact on open-pit design and mine planning. *The Journal of The South African Institute of Mining and Metallurgy*, 106:205–211.

Johnson, T. and Barnes, R. (1988). Application of maximal flow algorithm to ultimate pit design.

Johnson, T. and Sharp, W. (1971). *A Three-dimensional dynamic programming method for optimal ultimate open pit design*, volume 7553. Bureau of Mines, US Dep. of the Interior.

Johnson, T. B. (1968). Optimum open pit mine production scheduling. Technical report, DTIC Document.

Johnson, T. B., Dagdelen, K., and Ramazan, S. (2002). Open pit mine scheduling based on fundamental tree algorithm. In *APCOM 2002: 30 th International Symposium on the Application of Computers and Operations Research in the Mineral Industry*, pages 147–159.

Journel, A. (2007). Roadblocks to the evaluation of ore reservesthe simulation overpass and putting more geology into numerical models of deposits. *Orebody modeling and strategic mine planning, AusIIMM, Melbourn, 2nd Edition, Spectrum Series*, 14:29–32.

Journel, A. G. and Huijbregts, C. J. (1978). *Mining geostatistics*. Academic press.

Kent, M., Peattie, R., and Chamberlain, V. (2007). Incorporating grade uncertainty in the decision to expand the main pit at the navachab gold mine, namibia, through the use of stochastic simulation. *The Australasian Institute of Mining and Metallurgy, Spectrum Series*, 14.

Kim, Y. (1979). Open-pit limits analysis: Technical overview. *Computer Methods for the*, 80:297–303.

Kim, Y., Cai, W., and Meyer, W. (1988). A comparison of microcomputer based optimum pit limity design algorithms. *SME Annual Meeting*.

Kim, Y. and Zhao, Y. (1994). Optimum open pit production sequencing-the current state of the art. *PREPRINTS-SOCIETY OF MINING ENGINEERS OF AIME*.

Klotz, E. and Newman, A. (2013). Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1):18–32.

Koenigsberg, E. (1982). The optimum contours of an open pit mine: An application of dynamic programming. *17th Application of Computers and Operations Research in the Mineral Industry*, pages 274–287.

Kumral, M. (2012). Production planning of mines: Optimisation of block sequencing and destination. *International Journal of Mining, Reclamation and Environment*, 26(2):93–103.

Kumral, M. and Dowd, P. (2005). A simulated annealing approach to mine production scheduling. *Journal of the Operational Research Society*, 56(8):922–930.

Lamghari, A. and Dimitrakopoulos, R. (2012). A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. *European Journal of Operational Research*, 222(3):642–652.

Land, A. and Doig, A. (1960). An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520.

Laurich, R. and Kennedy, B. (1990). Planning and design of surface mines. *Surface Mining, Port City Press, Baltimore*, pages 465–469.

Leite, A. and Dimitrakopoulos, R. (2007). Stochastic optimisation model for open pit mine planning: application and risk analysis at copper deposit. *Mining Technology*, 116(3):109–118.

Lemieux, M. (1979). Moving cone optimizing algorithm. *Computer Methods for the*, 80:329–345.

Lerchs, H. and Grossmann, F. (1964). Optimum design of open-pit mines. In *Operations Research*, volume 12, page B59. Inst Operations Research Management Sciences.

Li, S., Dimitrakopoulos, R., Scott, J., and Dunn, D. (2004). Quantification of geological uncertainty and risk using stochastic simulation and applications in the coal mining industry. In *Orebody Modelling and Strategic Mine Planning–Uncertainty and Risk Management International Symposium 2004*, pages 185–192. The Australasian Institute of Mining and Metallurgy.

Mathieson, G. (1982). Open pit sequencing and scheduling. In *SME Fall Meeting, Honolulu, September*, pages 4–9.

Menabde, M., Froyland, G., Stone, P., and Yeates, G. (2004). Mining schedule optimisation for conditionally simulated orebodies. In *Proceedings of the international symposium on orebody modelling and strategic mine planning: uncertainty and risk management*, pages 347–52.

Meyer, M. (1969). Applying linear programming to the design of ultimate pit limits. *Management Science*, 16(2):B–121.

Monkhouse, P. and Yeates, G. (2005). Beyond naive optimization. *AUSIMM Spectrum Series*, 14:3–8.

Morales, N. (2003). Robust models for simultaneous open pit and underground mines.

Moreno, E., Espinoza, D., and Goycoolea, M. (2010). Large-scale multi-period precedence constrained knapsack problem: a mining application. *Electronic notes in discrete mathematics*, 36:407–414.

Morley, C., Snowden, V., and Day, D. (1999). Financial impact of resource/reserve uncertainty. *Journal-South African Institute of Mining and Metallurg*, 99(6):293–302.

Mun, J. (2006). *Modeling risk: Applying Monte Carlo simulation, real options analysis, forecasting, and optimization techniques*, volume 347. John Wiley & Sons.

Newman, A., Rubio, E., Caro, R., Weintraub, A., and Eurek, K. (2010). A review of operations research in mine planning. *Interfaces*, 40(3):222–245.

Nicholas, G., Coward, S., Armstrong, M., and Galli, A. (2006). Integrated mine evaluation–implications for mine management. In *Proceedings International Mine Management Conference*, pages 69–79.

Pendharkar, P. and Rodger, J. (2000). Nonlinear programming and genetic search application for production scheduling in coal mines. *Annals of Operations Research*, 95(1-4):251–267.

Picard, J.-C. (1976). Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11):1268–1272.

Ramazan, S., Dagdelen, K., and Johnson, T. (2005). Fundamental tree algorithm in optimising production scheduling for open pit mine design. *Mining Technology*, 114(1):45–54.

Ramazan, S. and Dimitrakopoulos, R. (2004). Traditional and new mip models for production scheduling with in-situ grade variability. *International Journal of Surface Mining*, 18(2):85–98.

Ramazan, S. and Dimitrakopoulos, R. (2007). Stochastic optimisation of long-term production scheduling for open pit mines with a new integer programming formulation. *Orebody Modelling and Strategic Mine Planning, Spectrum Series*, 14:359–365.

Ravenscroft, P. (1992). Risk analysis for mine scheduling by conditional simulation. *Transactions of the Institution of Mining and Metallurgy. Section A. Mining Industry*, 101.

Rendu, J.-M. (2002). Geostatistical simulations for risk assessment and decision making: the mining industry perspective. *International Journal of Surface Mining, Reclamation and Environment*, 16(2):122–133.

Rendu, J.-M. (2008). *Introduction to Cut Off Grade Estimation.* SME.

Saavedra Rosas, J.-F. (2009). *PhD Thesis - A genetic optimizer for stochastic problems with applications to orebody uncertainty in mine planning.* Laurentian University (Canada).

Sabour, S. and Wood, G. (2009). Modeling financial risk in open pit mine projects: implications for strategic decision-making. *Journal of the Southern African Institute of Mining and Metallurgy*, 109(3):169–175.

Samis, M., Laughton, D., and Poulin, R. (2003). Risk discounting: The fundamental difference between the real option and discounted cash flow project valuation methods.

SandiaLabs (2014). Coopr - python optimization modeling objects (pyomo). *https://software.sandia.gov/trac/coopr/wiki/Pyomo, last accessed Aug 2014.*

Sattarvand, J. (2009). *Long-term open-pit planning by ant colony optimization.* Universitätsbibliothek.

Sattarvand, J. and Niemann-Delius, C. (2008). Perspective of metaheuristic optimization methods in open pit production planning. *Gospodarka Surowcami Mineralnymi*, 24(4):143–156.

Sevim, H. and Lei, D. (1994). Dynamic-programming-based algorithm for optimal production planning in open pit mines. *Transactions-Society for Mining Metallurgy and Exploration Incorporated*, 296:1851–1855.

Seymour, F. (1995). Pit limit parameterization from modified 3d lerchs-grossmann algorithm. *Preprints-Society of Mining Engineers of AIME.*

Smith, M. L. (1998). Optimizing short-term production schedules in surface mining: Integrating mine modeling software with ampl/cplex. *International Journal of Surface Mining, Reclamation and Environment*, 12(4):149–155.

Stone, P., Froyland, G., Menabde, M., Law, B., Pasyar, R., and Monkhouse, P. (2007). Blasorblended iron ore mine planning optimization at yandi, western australia. In *Proc. Orebody Modelling and Strategic Mine Planning Symposium*.

Thomas, G. (1996). Optimization and scheduling of open pits via genetic algorithms and simulated annealing. In *Proceedings of the 1st International Symposium on Balkema Publisher*, pages 44–59.

Topal, E. (2008). Evaluation of a mining project using discounted cash flow analysis, decision tree analysis, monte carlo simulation and real options using an example. *International Journal of Mining and Mineral Engineering*, 1(1):62–76.

Underwood, R. and Tolwinski, B. (1998). A mathematical programming viewpoint for solving the ultimate pit problem. *European Journal of Operational Research*, 107(1):96–107.

Van Brunt, B. and Rossi, M. (1997). Optimising conditionally simulated orebodies with whittle 4d. In *Proceedings Optimizing with Whittle Conference*, pages 119–128.

Vann, J., Bertoli, O., and Jackson, S. (2002). An overview of geostatistical simulation for quantifying risk. In *Proceedings of Geostatistical Association of Australasia Symposium" Quantifying Risk and Error*, volume 1, page 1.

Vose, D. (2008). *Risk analysis: a quantitative guide.* John Wiley & Sons.

Weintraub, A., Pereira, M., and Schultz, X. (2008). A priori and a posteriori aggregation procedures to reduce model size in mip mine planning models. *Electronic Notes in Discrete Mathematics*, 30:297–302.

Wharton, C. (1996). What they dont teach you in mining school–tips and tricks with pit optimizers. *Surface Mining*, pages 17–22.

Wilke, F. and Wright, E. (1984). Determining the optimal ultimate pit design for hard rock open pit mines using dynamic programming. *Erzmetall*, 37(3):139–144.

Wright, E. A. (1987). The use of dynamic programming for open pit mine design: some practical implications. *Mining Science and Technology*, 4(2):97–104.

Yamatomi, J., Mogi, G., Akaike, A., and Yamaguchi, U. (1996). Selective extraction dynamic cone algorithm for three-dimensional open pit designs. In *International Journal of Rock Mechanics and Mining Sciences and Geomechanics Abstracts*, volume 33, pages 220A–220A. Elsevier.

Yegulalp, T. M. and Arias, J. (1992). A fast algorithm to solve the ultimate pit limit problem. In *23rd International Symposium on the Application of Computers and Operations Research in The Mineral Industries*, pages 391–398.

Yun, Q., Lu, C., Chen, Y., and Lian, M. (2005). Applications of computation intelligence in mining engineering. In *APCOM 2005*, pages 163–172. Taylor and Francis Group plc.

Zhao, Y. and Kim, Y. (1992). A new optimum pit limit design algorithm. In *Proceedings 23rd International. APCOM Symposium*, pages 423–434.

```
======================================================================
======================================================================
CODE SNIPPET A - Code for generating Main input files to MILP
======================================================================
======================================================================

import csv
import numpy

class Block:
    def __init__(self):
        self.ijkvalue = 0
        self.attributes = []
        self.precedences = []

    def Decode_ijk(self,NumX,NumY,NumZ):
        z = int(self.ijkvalue/(NumY*NumX))
        y = int((self.ijkvalue-z*(NumY*NumX))/NumX)
        x = self.ijkvalue - (z*NumY*NumX) - (y*NumX)
        return x,y,z

    def Encode_ijk(self,NumX,NumY,NumZ,CurrX,CurrY,CurrZ):
        self.ijkvalue = (CurrZ*NumY*NumX)+(CurrY*NumX) + CurrX
        return self.ijkvalue

def getSchedData(nextSchedMod):

    temp = []
======================================================================
    for line in open(nextSchedMod, 'r'):
        rowSol = line.split(" ")  # SPLIT OF VALUES BASED ON SPACES
        rowSol = [int(e) for e in rowSol] # CREATE A LIST WITH THE_
        SPLIT VALUES
        b,p = int(rowSol[0]),int(rowSol[1]) # DEFINE INDIVIDUAL
        VARIABLES FROM VALUES IN LIST
        schedblocks.append(b) # ADD BLOCK ID TO LIST
        schedperiods.append(p) # ADD PERIOD VALUE TO LIST
    temp.append(schedblocks)
    temp.append(schedperiods)

    return temp # temp holds th two lists - passes back to main

def schedAgainstSims(nextSimMod):

    #LOMSchedSim = []
    counter = 0
    tonnes = 0
    grVal = 0
    blockval = 0
======================================================================
    with open(nextSimMod, 'r') as f:
            LOMSched = []
            reader = csv.reader(f, delimiter=',', _
            quoting=csv.QUOTE_NONE)
            for row in reader:
                counter +=1
                if counter == 1:
                    NumX,NumY,NumZ = int(row[0]),int(row[1]),int(row[2])
                elif counter >= 3:
                    CurrX,CurrY,CurrZ,tonnes,grVal =_
                    int(row[0]),int(row[1]),int(row[2]),float(row[3]),_
                    float(row[4])
                    NewBlock = Block()
                    b = NewBlock.Encode_ijk(NumX,NumY,NumZ,CurrX,CurrY,_
                    CurrZ)
                    if b in schedblocks:
                        for i, j in enumerate(schedblocks): # i = index_
                        and_
                         j = Block ID
                            if j == b:
                                p = schedperiods[i] # p = to the same _
                                index in schedperiods list
                                dr = ((1+DiscountRate)**(p-1)) #
                                discount rate
                                if ((SalesPrice*grVal*tonnes*Recov)_
                                -((MiningCosts+ProcCosts)*tonnes))/dr >
```

```
                                  0:
                                      blockval =
                                      ((SalesPrice*grVal*tonnes*Recov)_
                                      -((MiningCosts+ProcCosts)*tonnes))
                                      /dr
                                 else:
                                      blockval = (-1*(MiningCosts)*tonnes)
                                      /dr
                                 LOMSched.append(blockval) # add block
                                 value to_
                                  list
            LOMSchedSum = sum(LOMSched) # add total schedule value to
            list
            #print schedno, simno, LOMSchedSum # print NPV for each
            sim
            f.close()

            return LOMSchedSum

def doStats(LOMSchedSumAll,schedno,schedblocks):

    noblocks = len(schedblocks)
    minval = min(LOMSchedSumAll)
    maxval = max(LOMSchedSumAll)
    aveval = sum(LOMSchedSumAll)/len(LOMSchedSumAll)
    stddevval = numpy.std(LOMSchedSumAll)
    vrange = maxval - minval
    CV = round(stddevval/aveval,2)
======================================================================
    print 'Schedule: ' +  str(schedno) + ' --- blocks mined = ' +
    str(noblocks)_
     + ' >>>  --min-- ' + str(minval)+ ' --ave-- ' + str(aveval)+
     ' --max-- '_
     + str(maxval)+ ' --range-- ' + str(vrange)+ ' --stdev-- ' +
     str(stddevval)_
     + ' --CV-- ' + str(CV)

    l = [schedno,noblocks,minval,maxval,aveval,stddevval,CV]
    results.append(l)
    return results

def doReports(AllLOMSchedSumAll,results):

    with open('SummaryLOMET.csv', "w") as o:
        writer = csv.writer(o, lineterminator='\n')
        writer.writerows(AllLOMSchedSumAll)

    with open('SummaryET.csv', "w") as output:
        writer = csv.writer(output, lineterminator='\n')
        writer.writerows(results)

    print "Done!"
======================================================================
#====================================MAIN=============================

NumX=20
NumY=40
NumZ=14

SalesPrice = 7000
Recov = 1
MiningCosts = 8
ProcCosts = 14
DiscountRate = 0.1

results = []
AllLOMSchedSumAll = []

for schedno in range(1,41):

    schedblocks = []
    schedperiods = []
    LOMSched = []
    LOMSchedSumAll = []

    nextSchedMod = 'FinalSolutions/ResultIDSimET' + str(schedno) +
    '.dat'_
    # PICK UP THE NEXT SCHEDULE TO TEST
    a = getSchedData(nextSchedMod)
    schedblocks = a[0]
    schedperiods = a[1]

    for simno in range(1,41):

        nextSimMod = 'Spreadsheets/Rec/RecTrim_11200_' + str(simno) +
        '.csv'_
        # PICK UP NEXT SIM TO TEST

        LOMSchedSum = schedAgainstSims(nextSimMod)

        LOMSchedSumAll.append(LOMSchedSum) # All simulation values
        for a_
        specific schedule
    AllLOMSchedSumAll.append(LOMSchedSumAll) # All scheduels for_
    all simulations

    results = doStats(LOMSchedSumAll,schedno,schedblocks)

doReports(AllLOMSchedSumAll,results)



======================================================================
======================================================================
CODE SNIPPET B - Code for generating Scenario Optimisation Version 1
======================================================================
======================================================================

from coopr.opt import SolverFactory
from coopr.pyomo import *
import cplex
from cplex.exceptions import CplexError
import csv
import os

class Block:
    def __init__(self):
        self.ijkvalue = 0
        self.attributes = []
        self.precedences = []

    def Decode_ijk(self,NumX,NumY,NumZ):
        z = int(self.ijkvalue/(NumY*NumX))
        y = int((self.ijkvalue-(z*NumY*NumX))/NumX)
        x = self.ijkvalue - (z*NumY*NumX) - (y*NumX)
        return x,y,z

    def Encode_ijk(self,NumX,NumY,NumZ,CurrX,CurrY,CurrZ):
        self.ijkvalue = (CurrZ*NumY*NumX)+(CurrY*NumX) + CurrX
        return self.ijkvalue

def int_wrapper(reader):
    for v in reader:
        yield map(int, v)

def creatAbsModel_Dembo():

    dembo = AbstractModel() #Declares a variable to hold the model
    dembo.Blocks = Set() #Variables declaration
    dembo.Periods = Set()
    dembo.Destinations = Set()
    dembo.NRealizations = Set()
    dembo.value = Param(dembo.Blocks,dembo.NRealizations)
    dembo.OptimalValues = Param(dembo.NRealizations)
    dembo.discount_factor = Param(dembo.Periods)
    dembo.Tonnage = Param(dembo.Blocks)
    dembo.SalesPrice = Param()
    dembo.LowLimExtract = Param()
    dembo.UppLimExtract = Param()
    dembo.MiningCosts = Param()
    dembo.ProcCosts = Param()

    F_init = {}
    f = open('DemboPrecedence.txt', 'r')
    reader = csv.reader(f, delimiter = ' ',quoting=csv.QUOTE_NONE)
    reader = int_wrapper(reader)
    for row in reader:
        lista = []
        ijkvalue = row[0]
        for j in range(len(row)):
            if j>0:
                data = row[j]
                lista.append(data)
        F_init[ijkvalue] = lista

    dembo.F = Set(dembo.Blocks,initialize=F_init)

    dembo.Preced = Param(dembo.Blocks)

    dembo.x = Var(dembo.Blocks, dembo.Periods, dembo.Destinations,
    within=Boolean)

    dembo.mu = Var() #auxiliar variable

    def obj_rule(dembo):
        return dembo.mu

    dembo.obj = Objective(rule=obj_rule, sense=minimize)

    def extraction_once(dembo,i):
        return sum(dembo.x[i,t,d] for t in dembo.Periods for d in_
        dembo.Destinations) <= 1

    def maximum_capacity(dembo,t):
        return sum(dembo.x[i,t,d]*dembo.Tonnage[i] for i in
        dembo.Blocks_
        for d in dembo.Destinations) <= dembo.UppLimExtract

    def minimum_capacity(dembo,t):
        return sum(dembo.x[i,t,d]*dembo.Tonnage[i] for i in
        dembo.Blocks_
        for d in dembo.Destinations) >= dembo.LowLimExtract

    def min_constraint(dembo,r): #auxiliar constraint
```

```
        return dembo.OptimalValues[r] - sum(dembo.value[i,r]*
        dembo.x[i,t,d]_
        *dembo.discount_factor[t] for i in dembo.Blocks for t in_
        dembo.Periods for d in dembo.Destinations)<= dembo.mu

    def max_constraint(dembo,r): #auxiliar constraint
        return (sum(dembo.value[i,r]*dembo.x[i,t,d]*
        dembo.discount_factor[t]_
        for i in dembo.Blocks for t in dembo.Periods for d in_
        dembo.Destinations) - dembo.OptimalValues[r]) >= -1 *
        dembo.mu
        #return (1*dembo.OptimalValues[r]) - sum(dembo.value[i,r]_
        *dembo.x[i,t,d]*dembo.discount_factor[t] for i in
        dembo.Blocks_
        for t in dembo.Periods for d in dembo.Destinations)>=
        -1 * dembo.mu

    def Precedence(dembo,i,t):
        if dembo.Preced[i] > 0:
            return sum(dembo.x[i,p,d] for d in dembo.Destinations
             for p in_
            dembo.Periods if p <= t)*dembo.Preced[i]  <=
            sum(dembo.x[k,p,d]_
            for k in dembo.F[i] for d in dembo.Destinations
            for p in_
            dembo.Periods if p <= t)
        else:
            return Constraint.Skip

    dembo.extract_con = Constraint(dembo.Blocks,
    rule=extraction_once)
    dembo.maximumcap_con = Constraint(dembo.Periods,
    rule=maximum_capacity)
    dembo.minimumcap_con = Constraint(dembo.Periods,
    rule=minimum_capacity)
    dembo.precedence_con = Constraint(dembo.Blocks, dembo.Periods,_
    rule=Precedence)
    dembo.absupper_con = Constraint(dembo.NRealizations,
    rule=min_constraint)
    dembo.abslower_con = Constraint(dembo.Periods,
    rule=max_constraint)

    print "Dembo Abstract model created - dembo returned"
    print " "

    return dembo

def getDimensions(): # Get X,Y and Z dimensions (number of blocks
in each_
direction)
    sp='Spreadsheets/Orig/Master_11200.csv'
    count=0
    dimensions = []

    with open(sp, 'rb') as f:
        reader = csv.reader(f, delimiter=',',
        quoting=csv.QUOTE_NONE)
        for row in reader:
            count +=1
            if count == 1:
                NumX,NumY,NumZ = int(row[0]),int(row[1]),int(row[2])
                dimensions.append(NumX)
                dimensions.append(NumY)
                dimensions.append(NumZ)
    return dimensions

def createDatFiles(NRealizations): # For creating Dembo.dat and
Precedences.txt

        #First pass will create the ijk values for each block and
        remove infeasible blocks
    NumX = 0
    NumY = 0
    NumZ = 0
    CurrX = 0
    CurrY = 0
    CurrZ = 0
    #OffSetX = 0
```

```
#OffSetY = 0
#OffSetZ = 0
NumAtt = 0
count = 0
newcounter = 0

#----------------
SalesPrice = 7000
Recov = 1
LowLimExtract = 0
UppLimExtract = 800000
NDestinations = 1
DiscountRate = 0.1
MiningCosts = 8
ProcCosts = 14
NumPeriods = 8

#----------------

BlockModel = []
#spmodel = 'Spreadsheets/Orig/Master_11200.csv'
spmodel = 'Spreadsheets/Rec/RecTrim_11200.csv'

with open(spmodel, 'rb') as f:
    reader = csv.reader(f, delimiter=',',
    quoting=csv.QUOTE_NONE)
    for row in reader:
        count +=1
        if count == 1:
            NumX,NumY,NumZ = int(row[0]),int(row[1]),
            int(row[2])
        elif count == 2:
            NumAtt = int(row[0])
        else:
            CurrX = int(row[0]) #+ OffSetX
            CurrY = int(row[1]) #+ OffSetY
            CurrZ = int(row[2]) #+ OffSetZ
            if (CurrX>=1+NumZ-CurrZ) and
            (CurrX<=NumX-(NumZ-CurrZ)):
                if (CurrY>=1+NumZ-CurrZ) and
                (CurrY<=NumY-(NumZ-CurrZ)):
                    newcounter +=1
                    NewBlock = Block()
                    NewBlock.Encode_ijk(NumX,NumY,NumZ,CurrX,_
                    CurrY,CurrZ)
                    for i in range(3,NumAtt):
                        NewBlock.attributes.append(float(row[i]))
                    if (CurrZ == NumZ-2): # CREATE PRECEDENCES
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY-1)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY+1)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY-NumX)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY+NumX)
                    elif (CurrZ < NumZ-2):
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY-1)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY+1)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY-NumX)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+NumX*NumY+NumX)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+2*NumX*NumY+NumX-2)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+2*NumX*NumY+NumX+2)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+2*NumX*NumY-NumX+2)
                        NewBlock.precedences.append(NewBlock._
                        ijkvalue+2*NumX*NumY-NumX-2)
                        NewBlock.precedences.append(NewBlock._
                       ijkvalue+2*NumX*NumY+2*NumX-1)
                       NewBlock.precedences.append(NewBlock._
                       ijkvalue+2*NumX*NumY+2*NumX+1)
                       NewBlock.precedences.append(NewBlock._
                       ijkvalue+2*NumX*NumY-2*NumX+1)
                       NewBlock.precedences.append(NewBlock._
                       ijkvalue+2*NumX*NumY-2*NumX-1)

                   BlockModel.append(NewBlock)

strname = "Dembo.dat"
f = open(strname, 'w')

# Writing Set of Blocks
f.write('set Blocks := \n')
for block in BlockModel:
    f.write(str(block.ijkvalue) + '\n')
f.write(';\n')

f.write('set Periods := \n')
for k in range(NumPeriods):
    f.write(str(k+1) + '\n')
f.write(';\n')

#Write Destinations
f.write('set Destinations := \n')
for k in range(NDestinations):
    f.write(str(k+1) + '\n')
f.write(';\n')

f.write('set NRealizations := \n')
for k in range(NRealizations):
    f.write(str(k) + '\n')
f.write(';\n')

f.write('param : Tonnage Preced:= \n')
for block in BlockModel:
    Tonnage = float(block.attributes[0])
    f.write(str(block.ijkvalue) + ' ')
    f.write(str(Tonnage)+' ')
    f.write(str(len(block.precedences)) + '\n')
f.write(';\n')
======================================================================
# Writes the bi-dimensional data for the values in
all simulations
f.write('param  value:= \n')
for block in BlockModel:
    for k in range(NRealizations):
        f.write(str(block.ijkvalue) + ' ' + str(k) + ' ')

        Grade = float(block.attributes[k+1])
        if ((SalesPrice*Grade*Tonnage*Recov)-(MiningCosts+
        ProcCosts)*Tonnage) > 0:
            f.write(str((SalesPrice*Grade*Tonnage*Recov)-
            (MiningCosts+ProcCosts)*Tonnage)+'\n')
        else:
            f.write(str(-1*(MiningCosts)*Tonnage)+'\n')
f.write(';\n')

#Write Discount Factors
f.write('param discount_factor := \n')
for k in range(NumPeriods):
    f.write(str(k+1) + ' ' + str(float(1)/(1+DiscountRate)**k)
     + '\n')
f.write(';\n')

#Other parameters
#f.write('param Tonnage := ' + str(Tonnage) + ';\n')
f.write('param SalesPrice := ' + str(SalesPrice) + ';\n')
f.write('param LowLimExtract := ' + str(LowLimExtract) + ';\n')
f.write('param UppLimExtract := ' + str(UppLimExtract) + ';\n')
f.write('param MiningCosts := ' + str(MiningCosts) + ';\n')
f.write('param ProcCosts := ' + str(ProcCosts) + ';\n')
f.close()
g = open('DemboPrecedence.txt','w')
for block in BlockModel:
    if len(block.precedences) == 0:
        g.write(str(block.ijkvalue))
    else:
        g.write(str(block.ijkvalue) + ' ')
    for k in range(len(block.precedences)):
        if k == len(block.precedences)-1:
            g.write(str(block.precedences[k]))
        else:
            g.write(str(block.precedences[k]) + ' ' )
    g.write('\n')
g.close()

print "Dembo.dat and DemboPrecedence.txt created..."
print " "

return BlockModel

def getAllSolutions(NRealizations):

Solutions = []
count=0

with open('AllSolutions.csv', 'rb') as f:
    reader = csv.reader(f, quoting=csv.QUOTE_NONE)
    for row in reader:
        count +=1
        if count == 1:
            for i in range(NRealizations):
                newval = str(row[i])
                newval = newval.lstrip('"')
                newval = newval.rstrip('"')
                nval = float(newval)
                Solutions.append(nval)

print "AllSolutions retrieved - passed as Solution..."

return Solutions


def updateDembo(NRealizations, Solutions):
print 'Writing Coordination Model... '

#First we write the data we miss
strname = "Dembo.dat"
f = open(strname, 'a')

f.write('param  OptimalValues := \n')
for k in range(NRealizations):
    f.write(str(k) + ' ' + str(Solutions[k]) + '\n')
f.write(';\n')
f.close()

print "Dembo.dat updated with additional info (Solutions)..."
print " "

def createDemboModelandOptimisation (NRealizations, dembo):
DemboData = 'Dembo.dat'
instance2 = dembo.create(DemboData)
filename2, symbolmap2 = instance2.write("ModelDembo.lp")
DemboModel = 'ModelDembo.lp'

print "got here Aa"

try:

    print 'Starting Dembo Optimization...'
    demcpx = cplex.Cplex(DemboModel)
    demcpx.parameters.workmem.set(1600)
    demcpx.parameters.workdir.set(os.getcwd())
    demcpx.parameters.mip.strategy.file.set(3)
    demcpx.parameters.mip.tolerances.mipgap.set(0.051)
    demcpx.solve()
    my_dembo_solution = demcpx.solution
    symbolvar = symbolmap2.__getstate__()
    varresult = symbolvar['bySymbol']

    print "-------Dembo Model Solved-------"
    print " "
    print ">>>>> Results for Dembo Optimisation:......."
    print " "
    print "Objective Function Value Dembo: ", my_dembo_solution._
    get_objective_value()
    for i in range(len(varresult)):
        if (str(varresult[i][1])[0] == 'x'):
            if my_dembo_solution.get_values(str(varresult[i][0]))_
             > 0.0000001:

               # print  str(varresult[i][1]) + ", " +
               str(my_dembo_solution.get_values_
               (str(varresult[i][0])))

                myvalue = round(my_dembo_solution._
                get_objective_value()/1000000,2)
======================================================================
    g = open('Plots/ScheduleResultSim' + str(NRealizations+1) +
    '.dat', 'w')
    s = open('Plots/ScheduleResultSimSection' + str(NRealizations+1)
     + '.dat', 'w')
    h = open('FinalSolutions/ResultIDSim' + str(NRealizations+1)
    + '.dat', 'w')

    blockcounter=0
    blockcountersect=0
    blockIDs = []
    periods = []

    for j in range(len(varresult)):
        if str(varresult[j][1])[0] == 'x' and
         my_dembo_solution.get_values(str(varresult[j][0]))
          > 0.0000001:
            blockcounter+=1
            st = str(varresult[j][1])
            c11 = int(st.find(','))
            c12 = int(st.find(','))+1
            c20 = int(int(st.find(']'))-2)
            blockID = int(st[2:c11])
            blockIDs.append(blockID)
            periodID = int(st[c12:c20])
            periods.append(periodID)

            a = getDimensions()

            NumX = a[0]
            NumY = a[1]
            NumZ = a[2]

            zval = int(blockID/(NumY*NumX))
            yval = int((blockID-(zval*NumY*NumX))/NumX)
            xval = blockID - (zval*NumY*NumX) - (yval*NumX)
            ##zgraph = zval - (NumZ/1.2)
            #ygraph = yval - (NumY/2)
            #xgraph = xval - (NumX/2)
            zgraph = zval
            ygraph = yval
            xgraph = xval

            g.write(str(blockcounter) + " " + str(xgraph) + " "
             + str(ygraph) + " " + str(zgraph) + " " + str(periodID) + '\n')
            h.write(str(blockID) + " " + str(periodID) + '\n')

            if xval == int(NumX/2)+2:
                blockcountersect+=1
                s.write(str(blockcountersect) + " " + str(xgraph)
                + " " + str(ygraph) + " " + str(zgraph) + " " +
                str(periodID) + '\n')

    g.close()
    s.close()
    h.close()

    print('  ')
    print "Schedule for Dembo Completed!"
    print "Solution value $ " + str(myvalue) + " M" + "
     ====>>>> Blocks mined = " + str(len(blockIDs))
    print('  ')
```

```
    except CplexError, exc:
            print exc


#++++++++++++++++++++++++++++++++++++++++++ MAIN LOOP +++++++++++++++

NRealizations = 40

createDatFiles(NRealizations)
print "got here A"
a = getAllSolutions(NRealizations)
print "got here B"
updateDembo(NRealizations, a)
print "got here C"
dembo = creatAbsModel_Dembo()
print "got here D"
createDemboModelandOptimisation (NRealizations, dembo)
print "got here E"




=====================================================================
=====================================================================
CODE SNIPPET C - Code for generating Scenario Optimisation Version 2
=====================================================================
=====================================================================



from coopr.opt import SolverFactory
from coopr.pyomo import *
import cplex
from cplex.exceptions import CplexError
import csv
import os

class Block:
    def __init__(self):
        self.ijkvalue = 0
        self.attributes = []
        self.precedences = []

    def Decode_ijk(self,NumX,NumY,NumZ):
        z = int(self.ijkvalue/(NumY*NumX))
        y = int((self.ijkvalue-(z*NumY*NumX))/NumX)
        x = self.ijkvalue - (z*NumY*NumX) - (y*NumX)
        return x,y,z

    def Encode_ijk(self,NumX,NumY,NumZ,CurrX,CurrY,CurrZ):
        self.ijkvalue = (CurrZ*NumY*NumX)+(CurrY*NumX) + CurrX
        return self.ijkvalue

def int_wrapper(reader):
    for v in reader:
        yield map(int, v)

def creatAbsModel_Dembo():

    dembo = AbstractModel() #Declares a variable to hold the model
    dembo.Blocks = Set() #Variables declaration
    dembo.Periods = Set()
    dembo.Destinations = Set()
    dembo.NRealizations = Set()
    dembo.value = Param(dembo.Blocks,dembo.NRealizations)
    dembo.OptimalValues = Param(dembo.NRealizations)
    dembo.discount_factor = Param(dembo.Periods)
    dembo.Tonnage = Param(dembo.Blocks)
    dembo.SalesPrice = Param()
    dembo.LowLimExtract = Param()
    dembo.UppLimExtract = Param()
    dembo.MiningCosts = Param()
    dembo.ProcCosts = Param()

    F_init = {}
    f = open('DemboPrecedence.txt', 'r')
    reader = csv.reader(f, delimiter = ' ',quoting=csv.QUOTE_NONE)
    reader = int_wrapper(reader)
    for row in reader:
        lista = []
        ijkvalue = row[0]
        for j in range(len(row)):
            if j>0:
                data = row[j]
                lista.append(data)
        F_init[ijkvalue] = lista

    dembo.F = Set(dembo.Blocks,initialize=F_init)

    dembo.Preced = Param(dembo.Blocks)

    dembo.x = Var(dembo.Blocks, dembo.Periods, dembo.Destinations,
    within=Boolean)

    dembo.mu = Var() #auxiliar variable

    def obj_rule(dembo):
        return dembo.mu

    dembo.obj = Objective(rule=obj_rule, sense=minimize)

    def extraction_once(dembo,i):
        return sum(dembo.x[i,t,d] for t in dembo.Periods for d in
        dembo.Destinations) <= 1

    def maximum_capacity(dembo,t):
        return sum(dembo.x[i,t,d]*dembo.Tonnage[i] for i in
        dembo.Blocks for d in dembo.Destinations) <= dembo.UppLimExtract

    def minimum_capacity(dembo,t):
        return sum(dembo.x[i,t,d]*dembo.Tonnage[i] for i in
        dembo.Blocks for d in dembo.Destinations) >= dembo.LowLimExtract
======================================================================
    def min_constraint(dembo,r): #auxiliar constraint
        return dembo.OptimalValues[r] -
         sum(dembo.value[i,r]*dembo.x[i,t,d]*dembo.discount_factor[t]
         for i in
        dembo.Blocks for t in dembo.Periods for d in dembo.Destinations)
```

```
         <= dembo.mu

    def max_constraint(dembo,r): #auxiliar constraint
        return (sum(dembo.value[i,r]*dembo.x[i,t,d]*dembo.discount_factor[t]
         for i in dembo.Blocks for t in dembo.Periods for d in
         dembo.Destinations) - dembo.OptimalValues[r]) >= -1 *dembo.mu
        #return (1*dembo.OptimalValues[r]) -
         sum(dembo.value[i,r]*dembo.x[i,t,d]*dembo.discount_factor[t]
         for i in
        dembo.Blocks for t in dembo.Periods for d in dembo.Destinations)
         >= -1 * dembo.mu

    def Precedence(dembo,i,t):
        if dembo.Preced[i] > 0:
            return sum(dembo.x[i,p,d] for d in dembo.Destinations for p
            in dembo.Periods if p <= t)*dembo.Preced[i]  <=
            sum(dembo.x[k,p,d] for k in dembo.F[i] for d in
            dembo.Destinations for p in dembo.Periods if p <= t)
        else:
            return Constraint.Skip
======================================================================
    dembo.extract_con = Constraint(dembo.Blocks, rule=extraction_once)
    dembo.maximumcap_con = Constraint(dembo.Periods,
    rule=maximum_capacity)
    dembo.minimumcap_con = Constraint(dembo.Periods,
    rule=minimum_capacity)
    dembo.precedence_con = Constraint(dembo.Blocks,
    dembo.Periods, rule=Precedence)
    dembo.absupper_con = Constraint(dembo.NRealizations,
    rule=min_constraint)
    dembo.abslower_con = Constraint(dembo.Periods,
    rule=max_constraint)

    print "Dembo Abstract model created - dembo returned"
    print " "

    return dembo

def getDimensions(): # Get X,Y and Z dimensions (number of
blocks in each direction)
    sp='Spreadsheets/Orig/Master_11200.csv'
    count=0
    dimensions = []

    with open(sp, 'rb') as f:
        reader = csv.reader(f, delimiter=',',
        quoting=csv.QUOTE_NONE)
        for row in reader:
            count +=1
            if count == 1:
                NumX,NumY,NumZ = int(row[0]),int(row[1]),
                int(row[2])
                dimensions.append(NumX)
                dimensions.append(NumY)
                dimensions.append(NumZ)
    return dimensions

def createDatFiles(NRealizations): # For creating Dembo.dat
and Precedences.txt

        #First pass will create the ijk values for each block
        and remove infeasible blocks
    NumX = 0
    NumY = 0
    NumZ = 0
    CurrX = 0
    CurrY = 0
    CurrZ = 0
    #OffSetX = 0
    #OffSetY = 0
    #OffSetZ = 0
    NumAtt = 0
    count = 0
    newcounter = 0

    #---------------- HARD CODED NEEDS TO BE MODIFIED
    SalesPrice = 7000
    Recov = 1
    LowLimExtract = 0
    UppLimExtract = 800000
    NDestinations = 1
    DiscountRate = 0.1
    MiningCosts = 8
    ProcCosts = 14
    NumPeriods = 8

    #----------------

    BlockModel = []
    #spmodel = 'Spreadsheets/Orig/Master_11200.csv'
    spmodel = 'Spreadsheets/Rec/RecTrim_11200.csv'
======================================================================
    with open(spmodel, 'rb') as f:
        reader = csv.reader(f, delimiter=',',
        quoting=csv.QUOTE_NONE)
        for row in reader:
            count +=1
            if count == 1:
                NumX,NumY,NumZ = int(row[0]),int(row[1]),int(row[2])
            elif count == 2:
                NumAtt = int(row[0])
            else:
                CurrX = int(row[0]) #+ OffSetX
                CurrY = int(row[1]) #+ OffSetY
                CurrZ = int(row[2]) #+ OffSetZ
                if (CurrX>=1+NumZ-CurrZ) and (CurrX<=NumX-(NumZ-CurrZ)):
                    if (CurrY>=1+NumZ-CurrZ) and (CurrY<=NumY-
                    (NumZ-CurrZ)):
                        newcounter +=1
                        NewBlock = Block()
                        NewBlock.Encode_ijk(NumX,NumY,NumZ,CurrX,
                        CurrY,CurrZ)
                        for i in range(3,NumAtt):
                            NewBlock.attributes.append(float(row[i]))
                        if (CurrZ == NumZ-2): # CREATE PRECEDENCES
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY-1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY+1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY-NumX)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY+NumX)
                        elif (CurrZ < NumZ-2):
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY-1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY+1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY-NumX)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+NumX*NumY+NumX)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY+NumX-2)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY+NumX+2)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY-NumX+2)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY-NumX-2)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY+2*NumX-1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY+2*NumX+1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY-2*NumX+1)
                            NewBlock.precedences.append_
                            (NewBlock.ijkvalue+2*NumX*NumY-2*NumX-1)

                        BlockModel.append(NewBlock)

    strname = "Dembo.dat"
```

```
f = open(strname, 'w')

# Writing Set of Blocks
f.write('set Blocks := \n')
for block in BlockModel:
    f.write(str(block.ijkvalue) + '\n')
f.write(';\n')

# Writing Set of Periods, based in capacity and number of blocks
#NumPeriods = int(len(BlockModel)*Tonnage/float(LowLimExtract))

f.write('set Periods := \n')
for k in range(NumPeriods):
    f.write(str(k+1) + '\n')
f.write(';\n')

#Write Destinations
f.write('set Destinations := \n')
for k in range(NDestinations):
    f.write(str(k+1) + '\n')
f.write(';\n')

f.write('set NRealizations := \n')
for k in range(NRealizations):
    f.write(str(k) + '\n')
f.write(';\n')

f.write('param : Tonnage Preced:= \n')
for block in BlockModel:
    Tonnage = float(block.attributes[0])
    f.write(str(block.ijkvalue) + ' ')
    f.write(str(Tonnage)+' ')
    f.write(str(len(block.precedences)) + '\n')
f.write(';\n')

# This next one writes the bidimensional data for the
values in all simulations
f.write('param  value:= \n')
for block in BlockModel:
    for k in range(NRealizations):
        f.write(str(block.ijkvalue) + ' ' + str(k) + ' ')

        Grade = float(block.attributes[k+1])
        if ((SalesPrice*Grade*Tonnage*Recov)-
        (MiningCosts+ProcCosts)*Tonnage) > 0:
            f.write(str((SalesPrice*Grade*Tonnage*Recov)
            -(MiningCosts+ProcCosts)*Tonnage)+'\n')
        else:
            f.write(str(-1*(MiningCosts)*Tonnage)+'\n')
f.write(';\n')

#Write Discount Factors
f.write('param discount_factor := \n')
for k in range(NumPeriods):
    f.write(str(k+1) + ' ' + str(float(1)/
    (1+DiscountRate)**k) + '\n')
f.write(';\n')

#Other parameters
#f.write('param Tonnage := ' + str(Tonnage)
+ ';\n')
f.write('param SalesPrice := ' + str(SalesPrice)
+ ';\n')
f.write('param LowLimExtract := ' + str(LowLimExtract)
+ ';\n')
f.write('param UppLimExtract := ' + str(UppLimExtract)
+ ';\n')
f.write('param MiningCosts := ' + str(MiningCosts)
+ ';\n')
f.write('param ProcCosts := ' + str(ProcCosts)
+ ';\n')
f.close()
g = open('DemboPrecedence.txt','w')
for block in BlockModel:
    if len(block.precedences) == 0:
        g.write(str(block.ijkvalue))
    else:
        g.write(str(block.ijkvalue) + ' ')
    for k in range(len(block.precedences)):
        if k == len(block.precedences)-1:
            g.write(str(block.precedences[k]))
        else:
            g.write(str(block.precedences[k]) + ' ' )
    g.write('\n')
g.close()

print "Dembo.dat and DemboPrecedence.txt created..."
print " "

return BlockModel

def getAllSolutions(NRealizations):

Solutions = []
count=0

with open('AllSolutions.csv', 'rb') as f:
    reader = csv.reader(f, quoting=csv.QUOTE_NONE)
    for row in reader:
        count +=1
        if count == 1:
            for i in range(NRealizations):
                newval = str(row[i])
                newval = newval.lstrip('"')
                newval = newval.rstrip('"')
                nval = float(newval)
                Solutions.append(nval)

print "AllSolutions retrieved - passed as Solution..."

return Solutions


def updateDembo(NRealizations, Solutions):

print 'Writing Coordination Model... '
#First we write the data we miss
strname = "Dembo.dat"
f = open(strname, 'a')

f.write('param  OptimalValues := \n')
for k in range(NRealizations):
    f.write(str(k) + ' ' + str(Solutions[k]) + '\n')
f.write(';\n')
f.close()

print "Dembo.dat updated with additional info (Solutions)..."
print " "

def createDemboModelandOptimisation (NRealizations, dembo):
DemboData = 'Dembo.dat'
instance2 = dembo.create(DemboData)
filename2, symbolmap2 = instance2.write("ModelDembo.lp")
DemboModel = 'ModelDembo.lp'

print "got here Aa"

try:

    print 'Starting Dembo Optimization...'
    demcpx = cplex.Cplex(DemboModel)
    demcpx.parameters.workmem.set(1600)
    demcpx.parameters.workdir.set(os.getcwd())
    demcpx.parameters.mip.strategy.file.set(3)
    demcpx.parameters.mip.tolerances.mipgap.set(0.051)
    demcpx.solve()
    my_dembo_solution = demcpx.solution
    symbolvar = symbolmap2.__getstate__()
    varresult = symbolvar['bySymbol']

    print "-------Dembo Model Solved-------"
    print " "
    print ">>>>> Results for Dembo Optimisation:......."
    print " "
    print "Objective Function Value Dembo: ",
    my_dembo_solution.get_objective_value()
    for i in range(len(varresult)):
        if (str(varresult[i][1])[0] == 'x'):
            if my_dembo_solution.get_values(str(varresult[i][0]))
             > 0.0000001:

               # print  str(varresult[i][1]) + ", " +
                str(my_dembo_solution.get_values(str
                (varresult[i][0])))

                myvalue = round(my_dembo_solution.get_objective_value()
                /1000000,2)

    g = open('Plots/ScheduleResultSim' + str(NRealizations+1)
     + '.dat', 'w')
    s = open('Plots/ScheduleResultSimSection' + str(NRealizations+1)
     + '.dat', 'w')
    h = open('FinalSolutions/ResultIDSim' + str(NRealizations+1)
     + '.dat', 'w')

    blockcounter=0
    blockcountersect=0
    blockIDs = []
    periods = []

    for j in range(len(varresult)):
        if str(varresult[j][1])[0] == 'x' and
         my_dembo_solution.get_values(str(varresult[j][0]))
         > 0.0000001:
            blockcounter+=1
            st = str(varresult[j][1])
            c11 = int(st.find(','))
            c12 = int(st.find(','))+1
            c20 = int(int(st.find(']'))-2)
            blockID = int(st[2:c11])
            blockIDs.append(blockID)
            periodID = int(st[c12:c20])
            periods.append(periodID)
```

```
                a = getDimensions()

                NumX = a[0]
                NumY = a[1]
                NumZ = a[2]

                zval = int(blockID/(NumY*NumX))
                yval = int((blockID-(zval*NumY*NumX))/NumX)
                xval = blockID - (zval*NumY*NumX) - (yval*NumX)
                ##zgraph = zval - (NumZ/1.2)
                #ygraph = yval - (NumY/2)
                #xgraph = xval - (NumX/2)
                zgraph = zval
                ygraph = yval
                xgraph = xval

                g.write(str(blockcounter) + " " + str(xgraph)
                 + " " + str(ygraph) + " " + str(zgraph) + " "
                 + str(periodID) + '\n')
                h.write(str(blockID) + " " + str(periodID) + '\n')

                if xval == int(NumX/2)+2:
                    blockcountersect+=1
                    s.write(str(blockcountersect) + " "
                    + str(xgraph) + " " + str(ygraph) + " "
                    + str(zgraph) + " " + str(periodID) + '\n')

        g.close()
        s.close()
        h.close()

        print('  ')
        print "Schedule for Dembo Completed!"
        print "Solution value $ " + str(myvalue) + " M" +
        " ====>>>> Blocks mined = " + str(len(blockIDs))
        print('  ')


    except CplexError, exc:
            print exc


#++++++++++++++++++++++++++++++++++++++++++ MAIN LOOP +++++++++

NRealizations = 40

createDatFiles(NRealizations)
print "got here A"
a = getAllSolutions(NRealizations)
print "got here B"
updateDembo(NRealizations, a)
print "got here C"
dembo = creatAbsModel_Dembo()
print "got here D"
createDemboModelandOptimisation (NRealizations, dembo)
print "got here E"
```