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: 04 July 2017

.................................................................
To my wife Van and daughter Tuong Vi

For their love, support, and patience
ACKNOWLEDGEMENTS

It is a great pleasure to thank my supervisor, Professor Erkan Topal, for all the valuable suggestions, discussions, guidance, and encouragements during my study. I have particularly appreciated his effort in connecting me with the industry. Thanks are due to my co-supervisor, Dr. Oktay Erten, for his guidance about ore resource estimation and timeless effort in reviewing my writing.

I would also like to thank Rio Tinto Iron Ore Company (RTIO) for financial support, data provision and for a chance to apply the proposed concepts in this study on one of their iron ore projects. Special thanks are to Bruce Somerville as my industry supervisor, Phil Savory and Cameron Boyle.

I want to acknowledge Assoc. Prof. Dr. Xuan Nam Bui at Hanoi University of Mining and Geology, Dr. Salih Ramazan, Hai Nguyen at University of Alberta, and my colleagues Zhao Fu, Joyce Chung, and Don Sandanayake at WASM for valuable discussions.

Financial support for this research is provided by Curtin Strategic International Research Scholarship (CSIRS), International Association for Mathematical Geosciences (IAMG), and RTIO.

Finally, I am eternally grateful to my wife Van, daughter Tuong Vi, and family for their encouragement, patience, and love.
PUBLICATIONS DURING CANDIDATURE


ABSTRACT

A well-known challenge to the application of mathematical programming techniques to solve open pit mine planning problems is the intensity of data. Specifically, industry-standard resource block models may contain up to hundreds of thousands or millions of blocks, and this makes obtaining an exact, optimal mine plan solution from these large-scale models intractable given the limitation of computing technology. To overcome this challenge, one common approximation approach is to aggregate blocks into larger units and then formulate and solve mathematical models on these aggregates. Currently available block aggregation algorithms, however, either fail to control the number of aggregates generated or do not consider slope constraints for aggregated blocks. These drawbacks significantly lower the performance of downstream mathematical programming-based production scheduling models in finding high-quality solutions.

In this study, a new block aggregation algorithm is developed, called TopCone Algorithm (TCA), where blocks are clustered into TopCones (TCs) having five outstanding features:

1. Slope angle is strictly respected;
2. The number of TCs generated is controlled;
3. Rapid solution time;
4. Can be implemented on limited computing facilities;
5. Near-optimal ultimate pit is determined during block aggregation process.

TCs form a novel platform for the application of mathematical programming techniques to solve both deterministic and stochastic mine planning problems. In the deterministic scenario, a kriged resource model is fed into TCA, so blocks are aggregated into TCs. These TCs then form the basis of an integer programming (IP) model specially formulated for production scheduling to maximise net present value (NPV) subject to a wide range of operational constraints, such as mining and processing capacities, blending grades, and slope constraints. In the stochastic
scenario, a new stochastic integer programming (SIP) is developed to consider geological uncertainty via TCs.

Attempts have also been made to characterise geological uncertainty using multivariate conditional simulation. In the study, a new simulation framework is proposed using minimum/maximum autocorrelation factors (MAF) and sequential Gaussian simulation (SGS) to generate orebody realisations. These stochastic resource models provide valuable information to characterise the grade and tonnage variability of the deposit as well as input data for stochastic mine planning.

To verify the proposed methodologies, case studies on an iron ore deposit were carried out. In the non-risk-based scenario, comparisons between the proposed deterministic mine planning method using TCA and IP versus a commercial planning software package, the Whittle Milawa NPV, showed considerably higher NPV values of 5.5% to 7%. In the risk-based scenario, the TCA-based SIP model outperformed considerably as compared to the deterministic models in terms of minimising the risk of not meeting production targets. Specifically, NPV risk profiles showed that the application of deterministic models would yield 7.01% to 7.81% lower NPV values than anticipated, where the corresponding figure of the proposed stochastic model is only -0.32% to 0.41%. In addition, all mine planning solutions based on TCA are typically obtained less than two hours using an ordinary computing power.
# TABLE OF CONTENTS

DECLARATION ....................................................................................................................................................... i
ACKNOWLEDGEMENTS ........................................................................................................................................ iii
PUBLICATIONS DURING CANDIDATURE ........................................................................................................... iv
ABSTRACT ........................................................................................................................................................... v
TABLE OF CONTENTS ........................................................................................................................................ vii
LIST OF FIGURES .................................................................................................................................................. x
LIST OF TABLES .................................................................................................................................................... xiii
LIST OF ABBREVIATIONS ..................................................................................................................................... xiv

CHAPTER 1. INTRODUCTION ............................................................................................................................... 1
  1.1 BACKGROUND ............................................................................................................................................. 1
  1.2 STATEMENT OF PROBLEM ......................................................................................................................... 2
  1.3 RESEARCH AIMS AND OBJECTIVES ........................................................................................................ 3
  1.4 SCOPE AND LIMITATION OF RESEARCH ............................................................................................... 4
  1.5 SIGNIFICANCE OF THE RESEARCH ........................................................................................................ 5
  1.6 THESIS ORGANISATION ............................................................................................................................ 6

CHAPTER 2. OVERVIEW OF OPEN PIT MINE PLANNING AND THE APPLICATION OF MATHEMATICAL PROGRAMMING ................................................................................................................... 8
  2.1 INTRODUCTION TO OPEN PIT MINE PLANNING ..................................................................................... 8
  2.2 FUNDAMENTAL CONCEPTS OF OPTIMISING MINE PLANNING PROBLEMS USING MATHEMATICAL PROGRAMMING ......................................................................................................... 9
    2.2.1 Ultimate pit limit problem .................................................................................................................... 10
    2.2.2 Pushback design problem .................................................................................................................. 12
    2.2.3 Production scheduling problem ......................................................................................................... 13
  2.3 SOLVING LARGE-SCALE MATHEMATICAL MODELS .............................................................................. 18
    2.3.1 Branch-and-bound technique .......................................................................................................... 19
    2.3.2 Heuristics/Metaheuristics methods .................................................................................................... 21
    2.3.3 Block aggregation methods ............................................................................................................. 22
  2.4 BLOCK PRECEDENCE ............................................................................................................................... 24
  2.5 GEOLOGICAL UNCERTAINTY AND THE IMPACT TO MINE PLANNING .............................................. 26
  2.6 CONCLUSIONS ............................................................................................................................................. 27

CHAPTER 3. METHODOLOGY OF TOPCONE ALGORITHM .................................................................................... 28
  3.1 INTRODUCTION ............................................................................................................................................. 28
  3.2 METHODOLOGY OF TOPCONE ALGORITHM .......................................................................................... 30
LIST OF FIGURES

Figure 1.1. Demonstration of the integration between software packages and computer programs used in the thesis ................................................................. 5

Figure 2.1. Illustration of the traditional three-stage open pit mine planning .................. 8

Figure 2.2. Demonstration of a directed graph with vertices (circles), arcs (arrows), mass (index inside circles) and a maximum closure ...................................................... 11

Figure 2.3. An illustration of branch-and-bound algorithm (Source: www.gurobi.com) .... 20

Figure 2.4. Two traditional sequencing rules of block precedence ................................. 24

Figure 2.5. Exact sequencing rule using block coordinates and slope angle ................. 25

Figure 2.6. NPV risk profiles of kriging-based mine plan (after Mai et al. (2016b)) .......... 26

Figure 3.1. Traditional three-stage mine planning framework (blue arrows) and the proposed two-stage framework with TopCone Algorithm (red arrows) (after Mai et al. (2017)) ................................................................. 29

Figure 3.2. Illustration of a network flow with all components ....................................... 31

Figure 3.3. Steps of TopCone Algorithm (after Mai et al. (2017)) ................................ 32

Figure 3.4. The hypothetical 2D case study ................................................................... 35

Figure 3.5. Representation of network flow of the of bench 2 ....................................... 36

Figure 3.6. Linear programming formulation of bench 2 ............................................... 37

Figure 3.7. Value of non-zero flows .............................................................................. 37

Figure 3.8. Representation of the TopCones for bench 2 ............................................. 38

Figure 3.9. Block model after mining out valid TopCones ............................................ 38

Figure 3.10. Representation of network flow of the of bench 3 ..................................... 39

Figure 3.11. Linear programming formulation of bench 3 ........................................... 39

Figure 3.12. Value of non-zero flows ........................................................................... 39

Figure 3.13. Representation of the TopCones for bench 3 .......................................... 40

Figure 3.14. Final result of TopCones and the ultimate pit limit .................................. 40

Figure 3.15. A demonstration of MCS relaxation ....................................................... 41

Figure 3.16. A hypothetical 2D case study, MCS is set of three nodes ....................... 42

Figure 3.17. Representation of the TopCones for level 2 ............................................. 42

Figure 3.18. Block model when two TopCones at level 2 are removed ....................... 43

Figure 3.19. One valid TopCone is found at level 3 ................................................... 43

Figure 3.20. TopCone result before relaxing ............................................................... 43

Figure 3.21. Schematic presentation of TopCone precedence relaxation .................... 44

Figure 3.22. TopCones after relaxing ........................................................................... 45

Figure 3.23. Positive node 8 become a TopCone after relaxation ............................... 46

Figure 3.24. MCS of 1, the ultimate pit is comprised of four blocks and has value of +4 .... 48

Figure 3.25. MCS of 5, the ultimate pit is comprised of nine blocks and has value of +2 .... 49
Figure 4.1. Location plan of Silvergrass deposit (Source: RTIO) ........................................56
Figure 4.2. Silvergrass East drill hole map and study area (red polygon) (Source: RTIO).....57
Figure 4.3. Comparison of blending grades between Whittle Milawa NPV, Milawa Balanced and TCA-IP models with different MCS options ................................................................. 61
Figure 4.4. Plan view of TCA-IP mine plan with MCS of 500 ........................................... 63
Figure 4.5. Typical cross-sections of TCA-IP mine plan with MCS of 500 ...................... 63
Figure 5.1. Schematic presentation of the proposed simulation framework ...................... 71
Figure 5.2. Histograms of six original variables ............................................................... 73
Figure 5.3. Histogram of primary Gaussian transformed variables ................................. 75
Figure 5.4. Histograms of MAF factors .......................................................................... 77
Figure 5.5. Histograms of normal score MAF factors ...................................................... 79
Figure 5.6. Variogram map (a) and fitting models (directional) of NS MAF1 (b) .......... 80
Figure 5.7. Variogram map (a) and fitting models (directional) of NS MAF2 (b) .......... 81
Figure 5.8. Variogram map (a) and fitting model (omnidirectional) of NS MAF3 (b) .... 82
Figure 5.9. Variogram map (a) and fitting models (directional) of NS MAF4 (b) ........... 83
Figure 5.10. Variogram map (a) and fitting model (omnidirectional) of NS MAF1 (b) ....84
Figure 5.11. Variogram map (a) and fitting models (directional) of NS MAF6 (b) ........... 85
Figure 5.12. Samples of final simulation results .............................................................. 87
Figure 5.13. Reproduced variograms of six NS MAF factors ......................................... 90
Figure 5.14. Reproduced histogram of Fe between 1st SGS realisation (left) and composite data (right) ......................................................................................................................... 90
Figure 5.15. QQ plot of SGS versus composite data ............................................................ 91
Figure 5.16. Grade/tonnage curves kriging vs 20 SGS realisations (after (Mai et al., 2016b)) ................................................................................................................................. 92
Figure 5.17. Comparison of ordinary kriging (a) and simulation (realisation No.1) (b) resource models ................................................................................................................. 93
Figure 6.1. Schematic presentation of the optimisation process developed for SIP model ....97
Figure 6.2. Risk profiles - Ore tonnage of Cou = 0.75 (a) and Cou = 1 (b)............... 107
Figure 6.3. Compare NPV Risk profiles of these two Cou options ............................... 108
Figure 6.4. Plan view of SIP model Cou 0.75 ................................................................. 109
Figure 6.5. Typical cross-sections of SIP mine plan with Cou of 0.75 ......................... 109
Figure 6.6. Plan view of SIP model Cou 1 ................................................................. 110
Figure 6.7. Typical cross-sections of SIP mine plan with Cou of 1 ........................... 110
Figure 7.1. Comparison of risk profiles between TCA/IP and TCA/SIP methods ......... 116
Figure 7.2. Compare NPV of TCA/IP and TCA/SIP without uncertainty costs .......... 117
Figure 7.3. Risk profile of NPV of TCA/IP mine planning method ......................... 118
Figure 7.4. Risk profile of NPV of TCA/SIP mine planning method ............................... 118

Figure 7.5. Correlation of number of TopCones and solution time of IP-based scheduling models .................................................................................................................................. 119
LIST OF TABLES

Table 4.1. Production scheduling parameters ................................................................. 58
Table 4.2. The implementation of TCA and IP with various MCS options .................. 59
Table 4.3. Bonus/Penalty for blending grades ............................................................... 62
Table 4.4. Comparison of NPV between Milawa NPV, Milawa Balanced and TCA-IP with and without grade bonus and penalty ................................................................. 62
Table 5.1. Statistics of attributes inside ore domain ...................................................... 72
Table 5.2. Pearson correlation matrix of borehole samples within the ore domain of Silvergrass East deposit (Mai et al., 2016b) ................................................................. 74
Table 5.3. Correlation matrix of primary Gaussian transformed variables ................. 76
Table 5.4. MAF transformation matrix from variables to factors: .............................. 76
Table 5.5. MAF transformation matrix from factors to variables: .............................. 76
Table 5.6. Correlation matrix of MAF factors ................................................................ 78
Table 5.7. Summary of best-fit semi-variogram models (Sph stands for spherical, Rel. stands for relative) ........................................................................................................ 86
Table 5.8. Comparison of correlations between original data and realisation No.1 ........ 88
Table 6.1. Summary of TCA implementing TCA and scheduling parameters .......... 104
Table 6.2. Information of the formulation and solving SIP models using different Cou options .................................................................................................................. 105
LIST OF ABBREVIATIONS

2D  Two Dimensional
3D  Three Dimensional
BEV Block Economic Value
CDF Cumulative Distribution Function
FTA Fundamental Tree Algorithm
FTs Fundamental Trees
Gaus Gaussian
GOL Ochreous Goethite
IAMG International Association for Mathematical Geosciences
IP Integer Programming
LG Lerchs-Grossman
LOI Loss on Ignition
LP Linear Programming
MAF Minimum/Maximum Autocorrelation Factors
MIP Mixed Integer Programming
MCS Minimum Cone Size
NPV Net Present Value
NS Normal Score
OK Ordinary Kriging
OR Operations Research
PCA Principal Component Analysis
QQ Quantile-Quantile
RTIO Rio Tinto Iron Ore
SCT Stepwise Conditional Transformation
SGS Sequential Gaussian Simulation
SIP Stochastic Integer Programming
TCA TopCone Algorithm
TCs TopCones
CHAPTER 1. INTRODUCTION

1.1 BACKGROUND

Open pit mining, by the simplest definition, is an excavation into the earth to extract useful mineral by digging pits. This entire concept may sound very straightforward but the actual works to conduct it in real life mining projects are very sophisticated. Given that a usable mineral deposit buried beneath the earth’s surface has been located by geologists, a mining engineer needs to figure out numerous questions of what to do next. These typically include where to start mining the first pit, which direction to follow up, where and when to remove, haul, store, and discard different materials, and most importantly, how to do all these tasks safely and profitably. All these features are a quick portrait of strategic mine planning, the most important task amongst mining activities and also the main theme of this thesis.

Generally, an open pit mine planning process is divided into three consecutive stages, namely ultimate pit limit, pushback design, and production scheduling. In the first stage, an ultimate pit limit is determined to delineate the final boundary of the pit, where the total undiscounted value from mining the deposit is maximum. In the next stage, the ultimate pit is commonly divided into several smaller pits called “pushbacks”, which are mainly used to guide the sequence of extraction and to reduce the amount of data fed into the next stage. Finally, life-of-mine production scheduling is constructed to determine the extraction time and destination for each block within the pushbacks or ultimate pit. Of the three, production scheduling attracts the most attention. Firstly, this stage directly determines the financial outcome and thus feasibility of mining projects. Secondly, optimising a production schedule is a very complicated process as it involves processing an enormous amount of data constrained by numerous conditions. Traditionally, it used to be highly tedious and time consuming to obtain an acceptable scheduling solution, let alone optimisation.

Since the birth of operations research (OR) techniques in World War II (Hillier, 2012), as with other research areas, mining engineers and scholars have quickly recognised the potential of these techniques in solving the complicated mine planning problems.
In fact, since the pioneering work of T. B. Johnson (1968), new models and algorithms using operations research techniques, particularly mathematical programming, have been proposed virtually on a yearly basis. Mine planning and production scheduling tasks can now be efficiently solved by highly automated and effective optimisation mathematical models. There is sufficient evidence to say that the application of operations research is a paradigm shift to open pit mine planning.

1.2 STATEMENT OF PROBLEM

A long-standing challenge to the application of mathematical programming to address real-life mine planning problems is the intensity of input data. It is generally accepted that one of the criteria to characterise the usefulness of an optimisation model using operations research techniques is its ability in obtaining a solution within a practical timeframe. As aiming at an optimal solution is normally impossible in most large-scale instances, approximation approaches are thriving in this research area. A common practice of this kind is block aggregation, where blocks are clustered into larger units to significantly reduce the data intensity before formulating and solving mathematical models. The majority of currently available block aggregation algorithms, however, either cannot compress the data efficiently or the shape of aggregates does not facilitate the yielding of high-quality production scheduling results. Thus, a new block aggregation algorithm which can cope with both issues is desperately needed.

In addition, the proposed mine planning model using the new block aggregation algorithm and mathematical programming should be capable of dealing with uncertainty as stochastic mine planning is an inevitable trend of the mining industry. This is a challenging task given that stochastic mine planning is a relatively new research topic.

Finally, the iron ore deposit being considered in this study is multivariate with six cross-correlated variables of interest. While it is widely accepted that simulation is the best technique to assess and generate geological uncertainty, ignoring or mishandling the multivariate nature of the deposit would lead to an impractical simulation result. Therefore, to ensure the high quality of stochastic mine planning,
efforts are needed to investigate and select an appropriate simulation technique for iron ore deposits.

1.3 RESEARCH AIMS AND OBJECTIVES

There are three major aims to be achieved in this research project: The first aim is to examine the application of multivariate conditional simulation in an iron ore deposit to produce a source of geological uncertainty for the mine planning process. The second aim is to establish a new block aggregation algorithm to facilitate the application of mathematical programming in solving large-scale production scheduling problems, so that high-quality solution can be obtained within a practical timeframe. The third aim is to develop stochastic optimisation ability for the proposed mine planning method, particularly focusing on incorporating geological uncertainty.

Objectives were set to achieve the aforementioned aims:

i. Review those most common multivariate conditional simulation techniques and compare their requirements with the typical characteristics of iron ore deposits to propose a practical simulation framework.

ii. Develop a new block aggregation technique which has two key features: honouring slope constraints while clustering blocks and the number of aggregates can be controlled.

iii. Develop a new integer programming (IP) model based on the aggregated blocks to address long-term production scheduling problems in the deterministic scenario. The objective of the IP model maximises the project’s net present value (NPV) while subject to all relevant production targets and operational constraints. The input deterministic geological data is estimated using the ordinary kriging (OK) interpolation technique.

iv. Develop a new stochastic integer programming (SIP) model based on the aggregated blocks to address long-term production scheduling problems in consideration of geological uncertainty, which is characterised and generated by the multivariate conditional simulation process. The objective of the SIP
model not only maximises the NPV but also minimises the risk of not meeting production targets due to the impact of geological uncertainty.

v. Develop computer programs and tools to compute and implement the proposed concepts on large-scale datasets.

1.4 SCOPE AND LIMITATION OF RESEARCH

In terms of operations research in general and mathematical programming in particular, this research is an intensive application of various techniques to deal with a wide range of problems in open pit mine planning: Linear programming resorts to group mining blocks into aggregates; Integer programming is used to solve deterministic production scheduling; Stochastic integer programming is used to optimise the production schedule under uncertainty.

In terms of the methodology, the research covers a systematic series of theories from resource estimation and simulation, to ultimate pit limit, production scheduling, and NPV.

In terms of computation, the concepts proposed in this research are verified intensively using both commercial software packages and computer programs written by the author. A link between these tools is presented in Figure 1.1.

- The simulation and kriging workload is implemented using ISATIS™ (Bleines & de Paris, 2000) and GSLIB™ (Deutsch & Journel, 1998).

- The TCA, IP and SIP models are computed using Visual Basic .Net programming language (Microsoft, 2012). The total length of those programs is approximately 5,000 code lines.

- The visualisation of scheduling results is completed using Surpac™ (Surpac, 2016).

- For a comparison purpose, production schedules are also implemented in Whittle Milawa NPV™ (Whittle, 2016), one of the most common commercial schedulers.
While there are three uncertainty sources of a mine planning project, namely geology, finance and technology, this study limits on geological uncertainty only.

1.5 SIGNIFICANCE OF THE RESEARCH

The work contained in this thesis provides several contributions to the application of operations research techniques to solve the open pit mine planning and production scheduling problems.

• **The main contribution of this research is to provide a novel platform for the application of operations research techniques in solving large-scale open pit mine planning problems.** A new block aggregation algorithm is developed, called TopCone Algorithm (TCA), which is not only capable of controlling the number of aggregates generated, which allows solving large-scale optimisation models but also incorporating slope angles during the aggregation process. Consequently, the mathematical programming-based scheduling models formulated based on the TopCones can yield high-quality solutions in a practical timeframe.

• **A new mine planning method is proposed which consists of two stages.** Firstly, blocks are grouped using the TCA, and ultimate pit limit is a by-product of this process. Secondly, TopCones are fed into a mathematical model to optimise the production scheduling subject to a full set of operational constraints to ensure the quality of the scheduling, including upper and lower bound of mining and processing capacity, blending grades of multiple elements and slope safety of mining pits. From the perspective of the industrial application, by eliminating the pushback design and
ultimate pit limit stages, the new method is simpler to practice than the traditional framework in terms of methodology, software, and solution time.

- **This research contributes to the relatively new area of stochastic mine planning by proposing a new optimisation framework using TCA and SIP.** In the model, the geological uncertainty characteristics of the deposit are transformed from block scale to aggregate scale as TopCones, which are subsequently fed into a new SIP model to simultaneously optimise the expected NPV and minimise the risk of not achieving targets. Attempts are also made to analyse the sensitivity of those cost units associated with stochastic optimisation to the scheduling result, and a method is proposed to calculate such parameters.

- **The contribution is also made to the area of multivariate conditional simulation.** Based on the investigation into available simulation techniques and characteristics of iron ore deposits, a new simulation framework is proposed using minimum/maximum autocorrelation factors (MAF) and sequential Gaussian simulation (SGS). A case study presented in this thesis has proved its robustness in evaluating and generating the in-situ grade and tonnage variability.

All concepts developed in this research, from simulation to open pit mine planning, are scalable so they can be easily adapted to other types of multi-element deposit, such as rare earth, coal, and gold.

1.6 THESIS ORGANISATION

**Chapter 1** provides an overview of the research, including background, state of the problem which leads to this study, significance, and purpose of the study.

**Chapter 2** provides literature reviews related to the research, including open pit mine planning and the application of mathematical programming, integrating uncertainty, multivariate conditional simulation, and block aggregation techniques.

**Chapter 3** presents the methodology of the TopCone Algorithm together with a simple demonstration in a two-dimensional hypothetical instance.
Chapter 4 discusses a new deterministic mine planning method using TopCone Algorithm and integer programming. An application of a large-scale case study of this method on an iron ore deposit is presented in this chapter.

Chapter 5 discusses the application of multivariate conditional simulation. A methodology review, previous works, a simulation framework most suitable for iron ore deposits, and a large-scale case study are the main contents of this chapter.

Chapter 6 discusses a new stochastic mine planning method using TopCone Algorithm and stochastic integer programming. Similar to the structure of Chapter 4, the formulation of the mathematical-based scheduling model and a case study using the same dataset and scheduling parameters are presented.

Chapter 7 compares and discusses the results gained from previous chapters, including NPV, the ability to cope with geological uncertainty, and solution time.

Chapter 8 draws the conclusions of the study and suggest recommendations for future research.
CHAPTER 2. OVERVIEW OF OPEN PIT MINE PLANNING AND THE APPLICATION OF MATHEMATICAL PROGRAMMING

2.1 INTRODUCTION TO OPEN PIT MINE PLANNING

In mine planning, it is standard practice to approximate a mineral deposit by subdividing it into arrays of rectangular units called blocks. This discretisation of a deposit into a block model is essential to adopt mathematical models so that blocks can be adequately described in mathematical terms. Each block contains the necessary information for the design and optimisation of mine plan, such as metal grades, density, lithology codes, block economic value (BEV), etc. Conventionally, block attributes are interpolated from exploration drill holes data by geostatistical estimation methods, of which the most popular is ordinary kriging (OK) (David, 1977; Goovaerts, 1997; Isaaks & Srivastava, 1989).

Mine planning process is then performed on three consecutive stages, namely ultimate pit limit, pushback design, and production scheduling (Dagdelen, 2001; J. Whittle, 1988), as illustrated in Figure 2.1.

Figure 2.1. Illustration of the traditional three-stage open pit mine planning

First of all, ultimate pit limit is determined to define a minable part of the deposit, including ore material and the corresponding overburden, so the total undiscounted
profit of mining the deposit is maximum. At this stage, only commodity prices, such as metal prices and operational costs, and pit slope angle are considered in the calculations. Those typical algorithms for this stage are moving cone (Lemieux, 1979; Pana & Carlson, 1966), Lerchs-Grossman (LG) algorithm (Lerchs & Grossmann, 1965) and Maximum Flow-based algorithms, e.g., the Pseudoflow algorithm (Hochbaum, 2008; Hochbaum & Chen, 2000).

Commonly, the total number of mining blocks within an ultimate pit limit is still too large to start implementing production scheduling, so ultimate pit limit is usually divided into smaller sequencing nested pits called “pushbacks”, “incremental cuts”, or “phases”. Price parameterisation technique, such as (J. Whittle, 1988), is the most popular approach to generating these nested pits, which includes trial-and-error process creating multiple pits based on different price scenarios.

Finally, a long-term production schedule is determined within pushbacks or pit limit, where decision-making process is made for each block to determine whether it is extracted in a particular period or not. The purpose of production scheduling is to maximise discounted cash flow, or net present value (NPV) of the project, subject to various constraints, which typically are mining and processing capacity constraints, blending grade constraints, and slope angle restrictions.

In the following sections, this chapter provides an overview of fundamental concepts of formulating and solving several mathematical models to optimise open pit mine planning problems. Several mathematical models for production scheduling are reviewed. The importance of solving large-scale instances is discussed together with a methodology review of some of those most well-known solutions. Advantages and difficulties of considering geological uncertainty into mine planning optimisation process are also discussed.

2.2 FUNDAMENTAL CONCEPTS OF OPTIMISING MINE PLANNING PROBLEMS USING MATHEMATICAL PROGRAMMING

In this section, different methods to determine block precedence and mathematical programming techniques applicable to various mine planning problems are discussed.
2.2.1 Ultimate pit limit problem

This problem is also known as the maximum-weight closure problem (Ahuja et al., 1993) or UPIT problem (Espinoza et al., 2013). The only objective of the problem is to delineate a pit contour consisting of ore blocks and the relevant overburden from original block model so that the total economic value of the pit is maximum. Hence, precedence relationship between blocks is the only constraint to ensure slope safety, while other operational constraints or time variable are not considered. Hochbaum and Chen (2000) describe this problem as an integer programming model. For a block model consisting of \( N \) blocks, let \( x_i \) be a binary variable that returns value of 1 if block \( i \) is inside the ultimate pit, and 0 otherwise. Let \( V_i \) be the economic value of block \( i \).

The ultimate pit problem can be addressed as:

\[
\text{Max} \quad \sum_{i \in N} V_i x_i \tag{2.1}
\]

Subject to:
\[
x_j - x_i \geq 0 \quad \forall (i, j) \in N \tag{2.2}
\]
\[
x_i \in \{0, 1\} \quad \forall i \in N \tag{2.3}
\]

The objective (2.1) is a maximisation function of the undiscounted value of all extracted blocks. Constraints (2.2) enforce block \( i \) is extracted only if all blocks \( j \) which are overlying a given block \( i \) according to slope angle are extracted. These constraints are also known as slope safety constraints. Constraints (2.3) restrict decision variables \( x_i \) as binary to ensure the integrality of blocks.

It has been proven that ultimate pit problems can be solved optimally and efficiently by graph theory. Specifically, if the ultimate pit problem is represented as a directed graph, optimising the Equation (2.1) is equivalent to finding a maximum closure of the graph. The optimality and convergence of this method are first given by Lerchs and Grossmann in an algorithm named after the authors as the well-known Lerchs-Grossmann’s (LG) algorithm (Lerchs & Grossmann, 1965). In this algorithm, blocks are represented as vertices associated with weights representing the block economic value and an arc is generated between two vertices if one is immediately overlying...
the other according to slope angle. Hence the block model can be described as a directed graph $G = (X, A)$ with a set of vertices $X$ and arcs $A$ as demonstrated in Figure 2.2.

![Figure 2.2. Demonstrations of a directed graph with vertices (circles), arcs (arrows), mass (index inside circles) and a maximum closure](image)

The LG algorithm provides an effective way to find the maximum closure of a given graph. Starting with constructing an initial tree $T^0$ consisting of only dummy arcs connecting a dummy root node to all vertices, the algorithm then repeatedly alters and re-labels arcs to create successive trees $T^1, T^2 \ldots$ until no further transformation is available. Detailed discussions of the steps and set of rules to transform trees can be found at (Lerchs & Grossmann, 1965; Zhao, 1992).

Zhao (1992) develops a new graph theory algorithm for optimum ultimate pit limit design with a focus on improving computational efficiency over the LG algorithm. The number of arc labels is reduced by half compared to LG algorithm (i.e. positive and negative arcs compared to plus-strong, plus-weak, minus-strong, and minus-weak arcs) and steps are defined to re-arrange waste vertices and arcs from negative trees to positive trees, similar to LG algorithm. Two advantages of the proposed algorithm over LG algorithm is that it is (1) computationally efficient to identify the value of vertices and trees, hence reduces memory requirements; and (2) easier to programme. A large-scale case study of the proposed model shows a significant improvement in terms of solution time. Hochbaum (2008) develops a new maximum flow algorithm, termed as Pseudoflow algorithm, to optimally solve the ultimate pit
limit problem. A rigorous comparison between the Pseudoflow algorithm and LG algorithm (Hochbaum & Chen, 2000) shows that the former is faster by several folds in all experiments. More discussions on maximum flow and graph theory algorithms can be found at (Picard, 1976; Underwood & Tolwinski, 1998; Zhao & Kim, 1992).

2.2.2 Pushback design problem

Commonly, pushbacks are generated by parameterisation, i.e. changing the value of input parameters, of an existing ultimate pit limit algorithm rather than developing new algorithms. J. Whittle (1988) proposes to vary the ratio of metal price to mining cost while implementing the LG algorithm to generate pushbacks. This clever idea is quickly turned into a dominant mine planning tool, known as Whittle software, in the mining industry until nowadays. Other parameters can also be considered to generate pushbacks. Recognising the significant gaps between consecutive pushbacks are a major issue in the Whittle implementation (Dagdelen, 1985), Seymour (1995) proposes to use pit volume as the variable in the parameterisation process. Describing the pit optimisation problem using graph theory, the author claims to generate ultimate pit and practical pushbacks in a single run. Ramazan and Dagdelen (1998) argue traditional pushback design algorithms which rely on parameterisation of commodity price may be not suitable to maximise NPV due to the impact of stripping ratio, hence the authors propose to generate pushbacks based on this parameter. Developed as an extension of the algorithm of Seymour (1995) but for stripping ratio, the new algorithm shows an ability to produce up to 6% higher NPV than the conventional Whittle method.

There are also several attempts to use OR techniques to generate pushbacks. Dagdelen and Johnson (1986) formulate an LP model for the pushback design problem as follows:

Max \[ \sum_{i \in N} V_i x_i \] \hspace{1cm} (2.4)

Subject to: \[ x_j - x_i \geq 0 \quad \forall (i, j) \in N \] \hspace{1cm} (2.5)
$$\sum_{i \in N} x_i = b \quad (2.6)$$

$$x_i \in [0,1] \quad \forall i \in N \quad (2.7)$$

While the objective function (2.4) and slope constraints (2.5) are identical to those of the IP model developed for the ultimate problem, constraints (2.7) define the decision variables $x_i$ as continuous variables. The distinct feature is the constraints (2.6), with $b$ is the desired size of pushbacks, so each implementation of the LP model will generate a pushback. The LP model is solved using Lagrangian relaxation method. Arguing that LP models tend to produce fractional solutions for pushbacks, Meagher et al. (2014) propose to solve this problem using IP. The authors expand the basic IP model discussed in Section 2.2.1 by considering mill capacity constraints to control the size of pushbacks. Pipage rounding technique is then deployed to solve the LP relaxation of the IP model. The application of the algorithm on a copper deposit shows a highly consistent of ore production between pushbacks. However, as waste tonnage is mostly scheduled in the early periods, the pushback design solution is not convincing.

2.2.3 Production scheduling problem

Different from ultimate pit problems where all blocks are assumed to be extracted at once, the target of production scheduling is to determine when and where to remove the individual blocks within the ultimate pit limits. Therefore, time value of money is accounted in the form of the discounted value of blocks. In addition, operational constraints, such as mining and processing equipment capacity, grade requirements, and production targets are considered to ensure the schedule is practical.

Four mathematical formulations to address production scheduling problems are discussed in this section: Linear Programming (LP), Integer Programming (IP), Mixed Integer Programming (MIP), and Stochastic Integer Programming (SIP).

a. Linear programming (LP)
T. B. Johnson (1968) proposes an LP model to optimise mine scheduling. Let \( t \in T \) be a set of scheduling period \( t \) in the horizon; Let \( k \in K \) be a set of production targets, e.g. mining, processing capacity, in period \( t \); Let \( m \in M \) be a set of material types \( m \) available in the deposit, such as ore, waste; Let \( \underline{R}_k \) and \( \overline{R}_k \), respectively be minimum and maximum bounds of production target \( k \) at period \( t \); Let \( x_{imt} \) be linear variable that governs the proportion of material \( m \) in period \( t \); Let \( C_{imt} \) be expected value from mining material \( m \) in block \( i \) being extracted at period \( t \); Let \( TB_i \) be the total material tonnage in block \( i \). The LP model can be represented as follows:

\[
\text{Max} \sum_{t \in T} \sum_{m \in M} \sum_{i \in N} TB_i \cdot C_{imt} \cdot x_{imt} \quad (2.8)
\]

Subject to

\[
\underline{R}_k \leq \sum_{m \in M} \sum_{i \in N} TB_i \cdot x_{imt} \leq \overline{R}_k \; \; \; \forall t \in T, \forall k \in K \quad (2.9)
\]

\[
\sum_{t \in T} \sum_{m \in M} x_{imt} = 1 \; \; \; \forall i \in N \quad (2.11)
\]

\[
x_{imt} \in [0,1] \; \; \; \forall i \in N, \forall t \in T, \forall m \in M \quad (2.12)
\]

Objective function (2.8) maximises NPV of extracted blocks over the life of mine. Constraints (2.9) enforce the total extracted amount of material \( m \) of any period \( t \) is within lower and upper bounds of the corresponding production targets. Constraints (2.10) enforce block \( i \) minable only if predecessor blocks of \( i \) are extracted not later than \( i \). Constraints (2.11) enforce if a block is extracted, that extraction must be completed within the life of mine. Constraints (2.12) restrict decision variables \( x_{imt} \) are linearly continuous from 0 to 1. More discussion of LP models for production scheduling problems can be found at (Osanloo et al., 2008).

The key advantage of LP models is solution time as its linear variables can be solved efficiently using commercially available solvers such as CPLEX (CPLEX, 2009) or Gurobi
The block mining sequence of LP-based solutions, however, normally violates slope constraints. This issue is because the fractional extraction of blocks as a function of linear variables provides situations where some portions of blocks are hanging in the air while the below blocks have already been extracted.

Given the pros and cons of LP models in solving production scheduling problems, there are two main recent application areas for this technique:

- LP-based scheduling is implemented to yield an initial scheduling solution quickly, and then its block mining sequence is calibrated, normally by heuristic algorithms, to solve the slope safety issue. The most recent work of this type can be found at Lamghari et al. (2015), where the authors use a variant of variable neighbourhood search, called variable neighbourhood descent, to improve an initial solution generated by linear programming. By applying the proposed method on the large-scale MineLib instances (Espinoza et al., 2013), solutions can be obtained in a relatively quick timeframe.

- For extremely large-scale scheduling problems, says hundreds of millions of blocks, and the mining conditions are fairly simple so a certain level of block mining sequence errors can be tolerated and easily handled in operational activities, LP could provide acceptable solutions. A case study of this type is reported by Aros and Reilly (2014), where the authors formulate LP models to solve production scheduling problems for iron ore deposits consisting of over 150 million blocks and the iron ore formations lie relatively even and shallow across large areas.

b. Integer programming (IP)

To overcome the fractional extraction of LP, integer variables are used as decision variables in IP models. For a given block model of \( N \) blocks to be scheduled in \( T \) periods, each block \( i \) is assigned a set of \( T \) binary variables denoted as \( x_{i}^{t} \), which returns a value of 1 if block \( i \) is scheduled in period \( t \), 0 otherwise. Let \( C_{i}^{t} \) be expected a value of block \( i \) being extracted in period \( t \). IP model can be presented as follows:

\[
\text{Max} \quad \sum_{i \in N} \sum_{t \in T} C_{i}^{t} \cdot x_{i}^{t} 
\]

\[(2.13)\]
Subject to

\[ R_{kt} \leq \sum_{i \in N} TB_i \cdot x_i' \leq \overline{R}_{kt} \quad \forall t \in T, \forall k \in K \]  

(2.14)

\[ \sum_{t' \in T} x_{ij}' - x_j' \geq 0 \quad \forall (i, j) \in N, \forall t \in T \]  

(2.15)

\[ \sum_{i \in T} x_i' \leq 1 \quad \forall i \in N \]  

(2.16)

\[ x_i' \in \{0,1\} \quad \forall i \in N, \forall t \in T \]  

(2.17)

Objective function (2.13) maximises the discounted value of the extracted blocks over the life of mine. Constraints (2.14) limit the extracted resources to be within bounds of production targets. Constraints (2.15) require block \( i \) is minable only if all of its predecessor blocks \( j \) are scheduled not later than \( i \). Constraints (2.16) enforce that blocks are extracted only once. Constraints (2.17) enforce the integrality of decision variables.

This optimisation technique has attracted many attentions from researchers as well as the industry. Intensive studies on IP-based scheduling models are discussed in the works of Caccetta and Giannini (1988); Chicoisne et al. (2012); Williams (1978). In the public-domain MineLib mine planning examples (Espinoza et al., 2013), this IP-based production scheduling problem is equivalent to the constrained pit limit (CPIT) problem.

c. Mixed Integer Programming (MIP)

In essence, MIP is a hybrid of LP and IP techniques, where blocks can be extracted partially while maintaining practical block sequencing. Let \( d \in D \) be a set of possible destinations for material extracted from the pit, such as processing plant, waste dump, and stockpile. Let \( y_{id} \) be the proportion of block \( i \) sent to destination \( d \) in time period \( t \). Let \( c_{id} \) be the expected value of one unit of block \( i \) sent to destination \( d \) in time period \( t \). MIP model can be presented as follows:

Max

\[
\sum_{i \in N} \sum_{t \in T} \sum_{d \in D} c_{id} \cdot y_{id}^{\prime}
\]  

(2.18)
Subject to \[ R_{ik} \leq \sum_{i \in N} y_{id} \cdot TB_i \leq \overline{R}_{ik} \quad \forall t \in T, \forall d \in D, \forall k \in K \] (2.19)

\[ \sum_{i \in CI} x_{ij} - x_{ij}^t \geq 0 \quad \forall (i, j) \in N, \forall t \in T \] (2.20)

\[ x_{ij}^t = \sum_{d \in D} y_{id} \quad \forall i \in N, \forall t \in T \] (2.21)

\[ \sum_{i \in T} x_{ij} \leq 1 \quad \forall i \in N \] (2.22)

\[ x_{ij}^t \in \{0, 1\}, \quad y_{id} = [0, 1] \quad \forall i \in N, \forall t \in T, \forall d \in D \] (2.23)

Constraints (2.21) enforce a consistency between processing variables and sequencing variable. That is, if a block is scheduled in a period, all of its material must be processed. The objective function and other constraints of MIP model can be interpreted similarly as IP model. This problem is discussed as a precedence constrained production scheduling problem (PCPSP) in MineLib (Espinoza et al., 2013). A great number of production scheduling algorithms and commercial software packages has been developed based on MIP in the past few decades (Achterberg & Wunderling, 2013; Askari-Nasab et al., 2011; Micromine, 2016; Minemax, 2016; Ramazan & Dimitrakopoulos, 2004; Smith & Dimitrakopoulos, 1999; Tabesh & Askari-Nasab, 2011).

d. Stochastic Integer Programming

All abovementioned production scheduling models rely on a critical assumption that the input data is provided with certainty but this is hardly true in real-life scenarios. Given that fact, SIP is developed as an extension of IP and MIP to cope with uncertainty, particularly of geological data. In essence, SIP is capable of simultaneously considering all possibilities of input data and produce an optimal solution in a single run. Let \( s \in S \) be a set of orebody realisations of a given mineralised deposit. Let \( \overline{C}_{ij}^t \) be the averaged expected value from mining block \( i \) in time period \( t \) from all realisations. Let \( A_{il}^t \) and \( A_{lu}^t \) respectively are the amount of lower (denoted as \( l \)) and upper (denoted as \( u \)) deviations from production target \( k \) in
period $t$ when orebody realisation $s$ being considered. Let $C^k_i$ and $C^k_u$ be the cost units of $A^k_{sl}$ and $A^k_{su}$, respectively, to penalise the excess production $k$.

SIP model deploys a two-course objective function, as follows:

$$\text{Max} \quad \sum_{t \in T} \left( \sum_{i \in N} \bar{C}^t_i . X^t_i - \sum_{s \in S} \left( C^k_i . A^k_{sl} + C^k_u . A^k_{su} \right) \right)$$

(2.24)

The first course is to maximise average expected value from mining block $i$ in time period $t$ from all $S$ realisations. The second course is to penalise the excess or deficit production if each realisation $s \in S$ is considered. This objective function is normally restricted by stochastic constraints as first proposed by Ramazan and Dimitrakopoulos (2007). The fundamental concept of stochastic constraints is that any solution is feasible regardless the violations of bounds, which are measured as $A^k_{sl}$ and $A^k_{su}$. These deviations from production targets are then minimised by the second course of the objective function as penalties.

In the last decade, a good number of studies has gradually discovered the capability of SIP in solving the stochastic mine planning problems and some researchers even predict that this approach will thrive as the next paradigm shift of future mine planning technology (Goodfellow, 2014). In Chapter 6 of this thesis, the literature review, methodology, formulation, and application of stochastic mine planning will be discussed in greater detail.

2.3 SOLVING LARGE-SCALE MATHEMATICAL MODELS

There is a consensus that one major challenge to the application of mathematical programming techniques to solve mine planning problems is the intensity of data. Thus one important criterion to evaluate the usefulness of a proposed model is its ability in obtaining a feasible solution for large-scale problems in a practical timeframe. In the last several decades, many techniques, algorithms, and
mathematical tools have been developed to tackle this challenge, and some of the most well-known ones are reviewed as follows:

2.3.1 Branch-and-bound technique

Integer-based mathematical models, namely integer programming, mixed integer programming, and stochastic integer programming, are typically solved using the linear programming-based branch-and-bound algorithm. Theory and applications of this algorithm are presented intensively in (Ahuja et al., 1993; Narendra & Fukunaga, 1977), an easy-understanding explanation can be found in (Gurobi, 2016).

In short, the principle of the branch-and-bound algorithm is to solve problems with bounded integer variables. Firstly, all integrality restrictions of the original mathematical models are relaxed, and this results in an LP relaxation model. Secondly, by solving this LP model and applying integrality restriction on one integer variable, the original problem is branched into two separate sub-problems, and the branching point is called the node. Keep iterating relaxing sub-problems and applying integrality restrictions on other integer variables until feasible solutions are found, which satisfy all constraints of the original problem. Once all nodes have been branched, the best feasible solution is also the optimum one, and the problem is optimally solved.

In mine planning problems, the objective function is generally to maximise the profit. The upper bound can be given by LP relaxation or Lagrangian relaxation (Ahuja et al., 1993), while lower bound is the current highest feasible solution. Bounds are used to terminate solving low-quality branches, which in turn speeds up the solving time.

A demonstration of branch-and-bound technique is presented in Figure 2.3.
Figure 2.3. An illustration of branch-and-bound algorithm (Source: www.gurobi.com)

(circles represent nodes, thick arrows indicate the path to the optimal solution)

A side effect of the branch-and-bound technique is that the number of sub-problems increases exponentially during the solving process, which makes the application of branch and bound algorithm to solve large-scale problems very expensive. In recent years, many improvements have been made to accelerate the branch-and-bound algorithm, and the four biggest contributors are presolve, cutting planes, heuristics, and parallelism. Well-written explanations of these four techniques, as well as a list of references, can be found in (Achterberg & Wunderling, 2013; Bixby et al., 2000). Especially, the combination of cutting planes technique and the branch and bound algorithm forms a very efficient approach called the branch-and-cut (Mitchell, 1999). Caccetta and Hill (2003) outline the application of the branch-and-cut technique to solve a MIP-based production scheduling problem in mining. The branch-and-bound and all acceleration techniques are available in commercial solvers such as CPLEX™ CPLEX (2009) and Gurobi™ (Gurobi, 2016).

However, it is common that applying these algorithmic techniques is still inadequate to solve truly large problems. Hence, researchers have resorted approximation approaches where near-optimal solutions can be achieved in a practical timeframe. They include block aggregation methods and heuristics/metaheuristic methods, as discussed in the following sections.
2.3.2 Heuristics/Metaheuristics methods

Heuristic and metaheuristic algorithms are approximation approaches which can release near-optimal solution within reasonable computational time, but the result has no guarantee of optimality. Some early discussions of the use of heuristic approaches to production scheduling are opened by Gershon (1987), where the author advocates the importance of obtaining acceptable solutions in practical timeframe over the optimality when dealing with large-scale instances. Two heuristic approaches are provided in this paper, one focuses on grade blending and the other focuses on block sequencing. In the production scheduling model proposed by Chicoisne et al. (2012), the LP relaxation of the IP-based scheduling problem is decomposed into sub-problems, each contains only a single capacity constraint per time period, and thus can be solved relatively quickly. The solution is then post-processed by two heuristic algorithms sequentially, TopoSort to correct the block sequencing and a local-search algorithm to improve the quality. The main drawback of the model is that only upper bounds of mining and processing capacities are considered. In addition, the practicality of mine plan is not guaranteed as blocks are likely scheduled in a scattered manner throughout the mine. Jélvez et al. (2015) reduce the size of an IP-based scheduling model by repeatedly formulating and solving the model for each period of time and reblocking the resource model. The solution is then post-processed by a heuristic algorithm to return the original resolution of the block model while fixing the slope constraints. Other noticeable heuristic algorithms developed for open pit production scheduling can be found at the works of Elevli (1995), Caccetta et al. (1998), and Erarslan and Celebi (2001)

Metaheuristics are at a more sophisticated level than heuristics, as defined by Voß et al. (2012): “A meta-heuristic is an iterative master process that guides and modifies the operation of subordinate heuristics to efficiently produce high-quality solutions”. Recently, there are some successful applications of metaheuristic method in solving large-scale production scheduling problems. Lamghari and Dimitrakopoulos (2012) develop two variants of Tabu search technique and their computational experiments show a significant improvement in terms of solution time over the commercial solver CPLEX. Tabesh and Askari-Nasab (2011) combine hierarchical clustering technique
and Tabu search in a two-stage clustering procedure to improve the solution time and practicality of the downstream production scheduling. Godoy and Dimitrakopoulos (2004) use simulated annealing to combine various production schedules, which are generated separately based on different geological scenarios, in an attempt to incorporate geological uncertainty into production schedule. Goodfellow (2014) resorts simulated annealing technique to solve large-scale SIP models built for mining complexes. Attempts to apply other metaheuristic methods, such as genetic algorithm (Denby & Schofield, 1994) and ant colony optimisation (Sattarvand & Niemann-Delius, 2008), have also been made and showed promising results.

On the dark side of this approach in general, the workflows and methodology of these heuristic and metaheuristic models are complicated and normally lose their effectiveness and efficiency when the planning and scheduling conditions change (Goodfellow, 2014).

2.3.3 Block aggregation methods

The idea of block aggregating/clustering/grouping (these three words are interchangeable) is that blocks are clustered into larger units to significantly reduce the intensity of data before formulating and solving scheduling problems, normally using solving techniques discussed in the previous sections.

a. Reblocking of block size

This approach is by far the simplest block aggregation method. By reducing the resolution of block model, one can easily achieve the number of aggregated blocks as desired. However, one obvious drawback is that the precision of information is compromised and thus the quality of scheduling solutions decreases accordingly. To regain the accuracy, Jélvez et al. (2015) develop a heuristic de-aggregation algorithm to restore the original resolution of block model while smooth out block sequencing. The proposed production scheduling method is three-fold: Firstly, the original block model is re-blocked to significantly reduce the data scale. Secondly, production scheduling is implemented on the re-blocked model using integer programming for one-period time window separately and consecutively. Finally, a heuristic algorithm
processes the scheduling solution obtained from the previous step to restore the original resolution of block model and achieve practical slopes. Besides the abovementioned negative impact of reduced block model resolution, one major shortfall of this model is that the optimisation is local in time, not global.

Reblocking is also a common option to handle large-scale instances in commercial production scheduling software packages, with Minemax (Minemax, 2016) and MineSched (MineSched, 2016) are typical examples among others.

b. Same-level block aggregation

One common trend is to group blocks on the same level based on their similarity of attributes, such as location, material, grade or economic value. Tabesh and Askari-Nasab (2011) group blocks into mining-cuts using hierarchical clustering. Firstly, similarity indices between blocks are calculated as a function of rock types, ore grades, and distances between blocks. Blocks are then merged sequentially based on their similarity indices until the pre-defined number of clusters is reached. A maximum size of clusters is also pre-defined based on equipment constraints to avoid clusters getting too large. These clusters are then fed into a MIP-based production scheduling model and is solved by Tabu Search procedure. Tolwinski (1998) proposes to use dynamic programming to divide the rock mass within benches of an ultimate pit limit into “atoms”. All possible schedules, which are different sequences of atoms, are constructed as a tree data structure and a heuristic algorithm is proposed to find a feasible solution from the tree. A same-bench clustering algorithm and MIP technique is also deployed by Minemax (Minemax, 2016). However, the optimisation process is local in time as only one period is considered in the production schedule at a time. An instance of same-bench clustering using Fuzzy C-means is conducted by Ren and Topal (2014).

As these abovementioned clustering methods do not respect slope constraints between blocks, which are compulsory components of scheduling models, this conflict may strongly reduce the possibility of finding the optimal solution. In some instances, such as Minemax (Minemax, 2016). and Jévez et al. (2015), the number of aggregates is still too large to globally optimise the whole life-of-mine plan.
c. Slope-based block aggregation

Incorporating block precedences into clustering process only gained attention in recent years. Ramazan (2007) proposes Fundamental Trees Algorithm (FTA), where an LP model is formulated to solve a network flow problem to cluster blocks into Fundamental Trees (FTs), with slope angle is strictly integrated during the clustering process. However, due to “single trees problem” where most FTs only consist of a single ore block, in most large-scale instances, the number of FTs is still very high. Blasor (Froyland & Menabde, 2009), a well-known in-house mine planning tool of BHP Billiton, deploys an upward propagation procedure to group blocks into “clumps” having a cone shape. Although the effectiveness of this tool is well-known, there is no evidence that the aggregation process can control the number of clumps generated.

2.4 BLOCK PRECEDENCE

One fundamental concept in open pit mine planning is the block precedence, i.e. the removal sequence between blocks, typically at different levels, to ensure slope safety. Traditionally, there are two sequencing rules to determine block precedence: To remove a given block, five or nine immediate overlying blocks must be removed, as illustrated in Figure 2.4.

![Figure 2.4. Two traditional sequencing rules of block precedence](image)

*Left: Rule 5:1, Right: Rule 9:1*

There are several disadvantages of these two sequencing rules: These are approximation method and not able to address slope angle precisely; Only robust to
be applied for slope angle of 45 degree and identical, cubic blocks; Incapable of applying different slope angles at different directions.

T. B. Johnson (1968) discussed several combinations of the two rules to reduce the number of blocks misclassified. Aiming at the same purpose, mine planning instances in MineLib (Espinoza et al., 2013) are provided with precedence determined by rule 5:1 with appropriate modifications for eight consecutive levels. These methods can improve the precision of slope approximation but still incapable of neither applying any other slope angles rather than 45 degrees nor altering slope angle at different directions.

Exact methods to determine block precedence need to be based on block coordinates and slope angle (Fytas et al., 1993; Khalokakaie et al., 2000). Let us assume two blocks $i$ and $j$ have coordinates $X_i$, $Y_i$, $Z_i$ and $X_j$, $Y_j$, $Z_j$ for their block centroids, as presented in Figure 2.5. Block $j$ is overlying block $i$ for a given slope angle $\alpha$ only when block $j$ belongs to the cone of angle $\alpha$ starts at the centroid of block $i$. This condition can be addressed by Equation (2.25)

$$\sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2} \leq (Z_j - Z_i) \cdot \tan \alpha$$  \hspace{1cm} (2.25)

![Figure 2.5. Exact sequencing rule using block coordinates and slope angle](image)

This exact sequencing rule overcomes the shortfalls of the traditional 5:1 and 9:1 rules. It can be applied for various block sizes and slope angles. Most importantly, this
sequencing rule is particularly suitable for block aggregation to allow clusters of blocks having a cone shape and thus well correspond to slope constraints in scheduling stage.

2.5 GEOLOGICAL UNCERTAINTY AND THE IMPACT TO MINE PLANNING

Geostatistical estimation methods, typically kriging techniques, have been broadly applied in the mining industry to interpolate the attributes of mining blocks but their inability to characterise the in-situ grade and tonnage variability of deposits is also well-known. Ignoring or mishandling such that geological uncertainty in mine planning models may lead to a severe risk of not meeting production expectations and project failures (Baker & Giacomo, 1998; Dimitrakopoulos et al., 2002; Groeneveld & Topal, 2011; Vallee, 2000). Mai et al. (2016b) report that deterministic mine plan overestimated the expected NPV of an iron ore deposit by 6.70 to 7.34% due to the impact of geological uncertainty, as shown in Figure 2.6.

![Figure 2.6. NPV risk profiles of kriging-based mine plan (after Mai et al. (2016b))](image-url)

Integrating geological uncertainty in mine planning process is, therefore, critical. To overcome that shortfall, conditional simulation techniques have been developed. The product of simulation is a series of equally-probable scenarios of the orebody, called realisations, of which each has equal probability to become real. In recent years,
conditional simulation has been considered as input data to assess and generate geological uncertainty for stochastic mine planning. For multiple deposits, such as iron ore, the cross-correlation of variables need to be reserved using a particular technique of multivariate conditional simulation. This discussion will continue in greater details in Chapter 5.

2.6 CONCLUSIONS

In modern open pit mine planning, operations research techniques, such as linear programming, integer programming, mixed integer programming and stochastic integer programming, have proved their capability in solving planning and scheduling problems. However, in realistic instances, the number of integer variables (for IP, MIP, and SIP) required is usually beyond today’s commercial hardware and software ability to be solved. Given that difficulty, block aggregation is normally referred to as one of the most potential approaches to overcome the scale problem of data.

The robustness of a block aggregation algorithm can be evaluated through two factors:

(1) The ability to incorporate slopes into aggregation process as slope constraints are a compulsory and most important component of scheduling models.

(2) The ability to control the efficiency of the aggregation process, which is the number of aggregates compared to the number of original blocks. This requirement is critical because the number of aggregates in large-scale instances can still be too large for the scheduling model to be solved in a practical timeframe.

The majority of available block aggregation algorithms, however, lack either of the two factors. Hence, this is the motivation for this PhD study to develop a new block aggregation technique to fulfil this need. In the next chapter, the methodology of that new technique, termed as TopCone Algorithm, is presented.
CHAPTER 3. METHODOLOGY OF TOPCONE ALGORITHM

3.1 INTRODUCTION

A novel and very interesting method, called Fundamental Tree Algorithm (FTA), of using linear programming (LP) to aggregate blocks is proposed by Ramazan (2001), Ramazan et al. (2005) and some minor improvements later by the main author Ramazan (2007). The authors describe block aggregation problem using a network flow where the flows of economic values sent between blocks in a network can be characterised by linear variables. The objective of FTA is to cluster blocks into Fundamental Trees (FTs) to reduce the intensity of data and then formulate a mixed integer programming (MIP) model based on FTs to optimise production scheduling. There are two unique features of FTA over the contemporary algorithms: (1) slope angle is respected during the aggregation process of FTs and (2) as the algorithm is described purely using linear programming, solutions can be obtained in a very practical timeframe.

Nevertheless, there are several major drawbacks which limit the potential of this algorithm in applying in the large-scale industrial optimisation.

1. Although FTA can considerably reduce the intensity of input data by grouping blocks into FTs, the conversion ratio from blocks to FTs is inadequate. This is because FTA is not able to group ore blocks together, except when joint support is required. Consequently, the majority of FTs contains only one ore block, the so-called “single tree” problem. The author of this thesis has re-programmed FTA based on published documents (Ramazan, 2001, 2007; Ramazan et al., 2005). Via intensive numerical experiments, it is evident that the number of FTs is only slightly less than the number of ore blocks available inside the ultimate pit where FTA is implemented. Therefore, in most large-scale instances, the number of FTs is still too large to be solved.

2. To ensure the feasibility, FTA must be implemented within optimal ultimate pit limits (Ramazan, 2001) and this not necessarily guarantees an optimal solution for FTA.
3. FTA considers the whole block model in its network flow to formulate a single LP model, and this may lead to out of memory problem when the computing power is limited and the block model is too large.

Inspired by the novel ideas as well as drawbacks of FTA, in this thesis a new concept for block aggregation is developed, termed as TopCone algorithm (TCA). The basic idea is that blocks of resource block model are clustered into TopCones (TCs) using linear programming in the way that the number of TCs is controlled and slope constraints are respected. In addition, the combination of all TCs generate a near-optimal ultimate pit, in other words, TCA is a hybrid of block aggregation algorithm and ultimate pit algorithm. This feature of TCA makes the practice of mine planning process simpler by eliminating pit optimisation stages, namely ultimate pit limit and pushback design. After blocks are aggregated into TCs, mathematical-based scheduling models are formulated based on TCs and be solved by a commercial solver, such as CPLEX (CPLEX, 2009), to optimise the production schedule. The new two-stage mine planning framework in comparison with the traditional three-stage one is illustrated in Figure 3.1.

**Figure 3.1.** Traditional three-stage mine planning framework (blue arrows) and the proposed two-stage framework with TopCone Algorithm (red arrows) (after Mai et al. (2017))
In this chapter, the methodology of TCA is discussed in Section 3.2, followed by a demonstration in 2-dimensional (2D) instance in Section 3.3. A discussion on the relaxation of TCs is presented in Section 3.4, followed by discussions and conclusions.

### 3.2 METHODOLOGY OF TOPCONE ALGORITHM

#### 3.2.1 Properties of TopCones

The TCs have four properties: (I) Can be mined independently without violating slope constraints with other TCs; (II) Economic value of a TopCone (TC), calculated by summing all comprising blocks, is positive; (III) The TCs subject to certain constraints; (IV) A TC cannot be separated into smaller TCs without violating (I), (II) and (III). The constraints applicable to property (III) could be minimum cone size (MCS) in the forms of a number of blocks, ore or waste tonnage, or other constraints which could be applicable in the downstream scheduling model. In this study, the only minimum number of blocks per TC is considered to control the number of TCs generated.

#### 3.2.2 Steps of TopCone Algorithm

The following notations are used to present the methodology of TCA

**Indices and sets**

- \( i \in N \) set of underlying nodes \( i \) found at current searching level
- \( j \in M \) set of overlying nodes \( j \) established at current network flow
- \( j \in O_i \subset M \) subset of overlying nodes \( j \) that belong to cone of underlying node \( i \)
- \( i \in U_j \subset N \) subset of underlying nodes \( i \) which have their cones cover overlying node \( j \)
- \( t \) sink node
- \( s \) source node

**Parameters**

- \( CV_i \) cone value of underlying node \( i \)
$C_i$ coefficient of underlying node $i$

$CP_i$ coefficient of pseudo flow sent to underlying node $i$

$V_i$ economic value of block $i$. If $V_i > 0$, block $i$ is called "positive node", otherwise, "negative node"

$\xi$ a very small decimal value, for example, 0.01

**Variables**

$f_{si}$ continuous variables: Flow from source node to underlying node $i$

$f_{ij}$ continuous variables: Flow from underlying node $i$ to overlying node $j$

$f_{jt}$ continuous variables: Flow from overlying node $j$ to sink node $t$

$pf_i$ continuous variables: Pseudo flow added to underlying node $i$

A demonstration of a network flow, including nodes and flows, is presented in Figure 3.2

![Network Flow Diagram](image)

**Figure 3.2.** Illustration of a network flow with all components

TCA groups blocks into TCs level by level, from top to bottom. Steps are schematically illustrated in Figure 3.3.
Figure 3.3. Steps of TopCone Algorithm (after Mai et al. (2017))

Step 1. The level that TCA is temporarily searching on is termed as current searching level, and all positive nodes of this level are termed as underlying nodes. All nodes of higher levels which are overlying the underlying nodes according to slope angle are termed as overlying nodes. The block precedence between underlying nodes and overlying nodes is determined using the exact method discussed in Section 2.4 and Equation (2.25), which involves block coordinates and slope angles. If no
positive nodes available at the current searching level, the algorithm goes to a lower level and repeats step 1.

**Step 2.** A network flow consisting of the source node, sink node, underlying nodes, overlying nodes, and flows is generated as demonstrated by Figure 3.2.

**Step 3.** Cone value of each underlying node is calculated by summing economic value of its comprising nodes and the underlying node itself, as described by Equation (3.1)

\[ CV_i = \sum_{j \in O_i} V_j + V_i \quad \forall i \in N \]  

(3.1)

**Step 4.** Coefficients of underlying nodes are determined in accordance with their cone value. Specifically, the underlying node with the highest cone value has a coefficient of 1, the second-highest has a coefficient of 2, and so on until underlying node with the smallest cone value has a coefficient of \( N \). Coefficients of corresponding pseudo flows are calculated from \( C_i \):

\[ CP_i = N + 1 - C_i \]  

(3.2)

**Step 5.** Set up and solve an LP model for the given network flow. This is a minimisation problem to cluster blocks into TCs so that the size of clusters are minimal given that they meet the predefined properties.

**Objective Function**

\[ \text{Min} \sum_{i \in N} \sum_{j \in O_i} C_i \cdot f_{ij} + \sum_{i \in N} CP_i \cdot pf_i \]  

(3.3)

**Subject to**

\[ f_{xi} \leq V_i + \sum_{j \in O_i} V_j, \text{ if } V_j > 0 \quad \forall i \in N \]  

(3.4)
\[ f_{ji} = \begin{cases} -V_j + \xi, & \text{if } V_j < 0 \\ \xi, & \text{if } V_j > 0 \end{cases} \quad \forall j \in M \] (3.5)

\[ \sum_{i \in U_j} f_{ij} - f_{ji} = 0 \quad \forall j \in M \] (3.6)

\[ pf_i + f_{si} - \sum_{j \in O_i} f_{ij} = 0 \quad \forall i \in N \] (3.7)

\[ pf_i, f_{si}, f_{ij}, f_{ji} \geq 0 \quad \forall i \in N, \forall j \in M \] (3.8)

Objective function (3.3) is comprised of two parts:

**Part 1** is a minimisation of connections between underlying and overlying nodes. Together with the effect of coefficients, where underlying nodes with high cone value are assigned small coefficients, the objective is to give priority to generate connections between high-value nodes so they are feasible for extraction before the lower value ones. This feature is critical to facilitate the downstream scheduling model to yield better NPV score.

**Part 2** is a substitution of pseudo value for underlying nodes when necessary to prevent LP model from being infeasible. This infeasibility happens when at least one overlying waste node does not receive sufficient support from those underlying nodes that it has connections, so the mass balance at this note is violated. In this case, the pseudo flows will add value to the network flow to prevent it from being infeasible.

Constraints (3.4) prevent underlying node and its positive overlying nodes from sending more support than their value. Constraints (3.5) ensure overlying nodes always receive enough support. Note that positive overlying nodes are extracted together with their corresponding underlying nodes. Hence they only need a representative support \( \xi \). The role of \( \xi \) in constraints (3.5) is to ensure TCs always have a positive value as it will require joint support from other underlying nodes if the current value of a TC is zero. Constraints (3.6) and (3.7) enforce the mass balance...
of flows coming and leaving overlying and underlying nodes. Constraints (3.8) define all flows as non-negative linear variables.

As the mathematical model only contains linear variables, it can be solved quickly by any solver, such as CPLEX (CPLEX, 2009) or Gurobi (Gurobi, 2016).

**Step 6.** TCs are formed by positive flows obtained from solving the LP model. As the block precedence between underlying nodes and overlying nodes strictly subject to slopes in Step 1, all TCs automatically respect and accommodate slope constraints. TCs are validated against properties (II) and (III) to find valid TCs. Note that the condition on the size of TCs can be relaxed if a TC has a positive value but has no possibility to be reconsidered in other iteration, i.e. there are no other positive nodes below. This relaxation is to improve the ultimate pit value and will be discussed in greater details in Section 3.4.

**Step 7.** Qualified TCs are removed from the block model.

**Step 8.** If the current searching level is the bottom, the algorithm goes to step 9, or else it goes to a lower level and back to step 1 with all variables reset.

**Step 9.** Print the results and stop.

3.3 DEMONSTRATION OF TCA IN A 2D INSTANCE

Figure 3.4 illustrates a 2-dimensional hypothetical instance of a mineral deposit consisting of twenty-one blocks. In this instance, property (III) is defined as minimum cone size of 2 blocks.

<table>
<thead>
<tr>
<th>Level 1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Level 3</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
</tbody>
</table>

*Figure 3.4. The hypothetical 2D case study (Inside circles are block indexes, whereas outsiders are block economic value)*
**Step 1.** Start from level 1, no positive node is found. Then, the algorithm moves to level 2 where there are five positive nodes, namely 9, 10, 11, 12 and 13. These nodes are now termed as underlying nodes.

**Step 2.** Generate a flow network for level 2 (Figure 3.5).

![Flow Network Diagram](image)

**Figure 3.5.** Representation of network flow of the of bench 2

**Steps 3, 4:** Cone values of five underlying nodes are -7, -5, -11, +3 and -8; therefore, their coefficients are 3, 2, 5, 1 and 4, respectively and the coefficients of corresponding pseudo flows are 3, 4, 1, 5 and 2, respectively.

**Step 5:** Formulate the LP model (Figure 3.6)
Minimise

\[3f_{9.1} + 3f_{9.2} + 3f_{9.3} + 2f_{10.2} + 2f_{10.3} + 2f_{10.4} + 5f_{11.3} + 5f_{11.4} + 5f_{11.5} + f_{12.4} + f_{12.5} + f_{12.6} + 4f_{13.5} + 4f_{13.6} + 4f_{13.7} + 3pf_{9} + 4pf_{10} + pf_{11} + 5pf_{12} + 2pf_{13}\]

Subject to

\[
\begin{align*}
    f_{9.1} - f_{1t} &= 0 \\
    f_{9.2} + f_{10.2} - f_{2t} &= 0 \\
    f_{9.3} + f_{10.3} + f_{11.3} - f_{3t} &= 0 \\
    f_{10.4} + f_{11.4} + f_{12.4} - f_{4t} &= 0 \\
    f_{11.5} + f_{12.5} + f_{13.5} - f_{5t} &= 0 \\
    f_{12.6} + f_{13.6} - f_{6t} &= 0 \\
    f_{13.7} - f_{7t} &= 0 \\
    pf_{9} - f_{9.1} - f_{9.2} - f_{9.3} &= 0 \\
    pf_{10} + f_{10.2} - f_{10.3} - f_{10.4} &= 0 \\
    pf_{11} + f_{11.3} - f_{11.4} - f_{11.5} &= 0 \\
    pf_{12} + f_{12.4} - f_{12.5} - f_{12.6} &= 0 \\
    pf_{13} + f_{13.5} - f_{13.6} - f_{13.7} &= 0
\end{align*}
\]

**Figure 3.6.** Linear programming formulation of bench 2

Solve the LP problem using CPLEX solver (CPLEX, 2009). Flows which are greater than zero are displayed in Figures 3.7 and 3.8, respectively.

\[
\begin{align*}
    f_{9.1} &= 1.01 \\
    f_{9.2} &= 2.02 \\
    f_{9.3} &= 1.01 \\
    f_{10.2} &= 5.01 \\
    f_{10.3} &= 2.99 \\
    f_{11.3} &= 1.01 \\
    f_{12.4} &= 3.01 \\
    f_{12.5} &= 4.01 \\
    f_{12.6} &= 3.01 \\
    f_{13.5} &= 1.01 \\
    f_{13.6} &= 2.01 \\
    f_{13.7} &= 2.01 \\
    f_{14.5} &= 4.01 \\
    f_{14.6} &= 3.01 \\
    f_{14.7} &= 2.01 \\
    f_{14.8} &= 1.01
\end{align*}
\]

**Figure 3.7.** Value of non-zero flows
Step 6: Three cones are found based on the LP solution (highlighted by dot-dash boundary). According to the conditions of post-processing, there are only two valid TCs, termed as TC 1 and TC 2. The total value of the cone constituted by nodes 7 and 13 is negative, so it is not a TopCone. Considering the mining sequence, TC 2 will be mined after TC 1.

Steps 7: Mine out two TCs found at step 6. The rest of blocks are illustrated in Figure 3.9.

Step 8: Go to bench 3 and repeat steps 1-8 to find more TCs. The flow network, LP formulation, flow result and TCs of bench three are presented in
Figures 3.10, 3.11, 3.12 and 3.13, respectively. Note that the positive value +1 of overlying node 11 is merged with underlying node 18 because it has a smaller coefficient than underlying node 19. More discussion on this value merging will be given in Proposition 3.5.4, Section 3.5.

Minimise

\[ f_{18.11} + 2f_{19.7} + 2f_{19.11} + 2f_{19.13} + 2pf_{18} + pf_{19} \]

Subject to

\[ fs_{18} \leq 3 \]  \[ f_{19.7} - f_{7t} = 0 \]
\[ fs_{19} \leq 3 \]  \[ f_{18.11} + f_{19.11} - f_{11t} = 0 \]
\[ f_{7t} = 2.01 \]  \[ f_{19.13} - f_{13t} = 0 \]
\[ f_{11t} = 0.01 \]  \[ pf_{18} + fs_{18} - f_{18.11} = 0 \]
\[ f_{13t} = 0.01 \]  \[ pf_{19} + fs_{19} - f_{19.7} - f_{19.11} - f_{19.13} = 0 \]

Figure 3.10. Representation of network flow of the of bench 3

Figure 3.11. Linear programming formulation of bench 3

Figure 3.12. Value of non-zero flows
Step 9: Because bench 3 is the last, the algorithm stops.

All TCs found are presented in Figure 3.14.

Figure 3.13. Representation of the TopCones for bench 3

Figure 3.14. Final result of TopCones and the ultimate pit limit
3.4 CONDITION RELAXATION

There are three situations where conditions applied on TCs are relaxed.

3.4.1 Relaxation of Condition (III): Minimum Cone Size (MCS) relaxation

In this study, minimum cone size is used as a condition (III) to control the number of TCs generated. In case that a TC has positive value but its number of nodes is less than MCS, the condition of MCS is relaxed if there is no possibility for its nodes to be reconsidered in other iteration to increase the size. This relaxation is to improve the value of TCA ultimate pit.

A demonstration of this condition relaxation is given in a 2D instance in Figure 3. Apply steps of TCA as discussed previously for level 2, two TCs were found. If MCS is set of 3 nodes, MCS relaxation is applied for TC2 to set this cone value and add +1 value to TCA ultimate pit.

![Figure 3.15. A demonstration of MCS relaxation](image)

3.4.2 Relaxation of Condition (I): TopCone precedence relaxation

The post-processing of TCA, where those TCs which are not satisfying MCS are left to be reconsidered in the lower level iteration, may lead to a situation of violating slope safety. A demonstration shown in Figures 3.16, 3.17 and 3.18 presents an example of this.
Figure 3.16. A hypothetical 2D case study, MCS is set of three nodes

Figure 3.17. Representation of the TopCones for level 2

Three cones are found at level 2. According to MCS condition of minimum three blocks per TC, there are only two valid TCs (indexed as TC 1 and TC 2), whereas the cone in the middle only has two blocks 4 and 10. Remove TC 1 and TC 2 out of the block model, as shown in Figure 3.18, and continue finding TCs in level 3, the results are displayed in Figure 3.17.
Figure 3.18. Block model when two TopCones at level 2 are removed

Figure 3.19. One valid TopCone is found at level 3

Figure 3.20. TopCone result before relaxing
Final representation of all TCs is presented in Figure 3.20. According to the order of formation, TC3 is extracted after TC2. However, this causes a slope safety issue with block 4. To handle this, block 4 should be swapped from TC3 to TC2 to return the correct mining sequence of TCs. Steps of this process are illustrated in Figure 3.21.

![Figure 3.21](image_url)

**Figure 3.21.** Schematic presentation of TopCone precedence relaxation

The process of TC relaxation can be interpreted as follows: Start with the first TC, it will then be compared to all preceding TCs to find if there are blocks that overlie the first TC. Once found, these blocks will be merged to the TC that they are overlying. The process continues until all TCs have been checked.
Apply this process to the case study above, block 4 of ST 3 is merged into TC 2, and the final result is displayed in Figure 3.22.

![Figure 3.22. TopCones after relaxing](image)

Note that the precedence relaxation is performed after TCA finishes scanning and yielding TCs at the bottom level, and only re-arrange blocks between TCs. Hence the total value of TCA ultimate pit is unchanged.

3.4.3 Relaxation of single-ore-block TopCones

In a network flow of TCA, there is a situation that some underlying nodes cannot establish connections with overlying nodes as these overlying nodes receive sufficient support from stronger underlying nodes. If there is no possibility that these weak underlying nodes are reconsidered in other iteration, i.e. they are at the bottom of the deposit, each of these nodes is valid as a TC.

As demonstrated in Figure 3.23, positive node 8 is unable to send support to any overlying node, and as there is no other lower level, it becomes a single TC after relaxation.
3.5 PROPOSITIONS OF TOPCONE ALGORITHM

**Proposition 3.5.1:** There always exists a feasible solution to the LP formulations of TopCone Algorithm

**Support 3.5.1:**

In the linear formulation of LP as discussed in the subsection 3.2.2, network flow constraints from underlying nodes to sink node are hard constraints, as presented below:

\[
f_{jt} = \begin{cases} 
-V_j + \xi, & \text{if } V_j < 0 \\
\xi, & \text{if } V_j > 0 
\end{cases} \quad \forall j \in M
\]

\[
\sum_{i \in U_j} f_{ij} - f_{jt} = 0 \quad \forall j \in M
\]

\[
pf_i + f_{si} - \sum_{j \in O_i} f_{ij} = 0 \quad \forall i \in N
\]

The only case to make this equation system infeasible is that an overlying negative node (i.e. waste block) does not receive enough support from its underlying nodes (i.e. ore blocks). Hence, the role of pseudo flows \( pf_i \), which are non-constrained non-negative linear variables, is to compensate the network whenever necessary to ensure the network is always feasible.

To demonstrate the role of pseudo flows \( pf_i \), we recall the 2D demonstration in Figure 3.8, Section 3.3:
In this network flow, overlying node 7 (negative value of -2) has only support from underlying node 13 (positive value of +1). As node 13 is insufficient to support node 7, this would cause infeasibility to the LP formulation without the pseudo support from the pseudo flow $pf_{13}$. Note that in the post-processing step 6, as the total value of nodes 7 and 13 is negative, they cannot form a valid TopCone and hence will be left in the network and reconsidered in the lower level.

To sum up, the addition of $pf_i$ is only to prevent LP formulation from being infeasible and does not contribute to the formulation of valid TopCones as well as TCA ultimate pit limit.

**Proposition 3.5.2:** The combination of all TopCones found by TopCone Algorithm is an ultimate pit

**Support 3.5.2:**

The mechanism of TCA is scanning through all ore blocks available in the orebody, from the top bench to the bottom, and sequentially extract as many ore blocks, together with the overburden, as possible, given that each time of extraction gains profit. By definition provided in the literature review, the result of this process is an ultimate pit.
Generally speaking, this scan-and-remove mechanism of TCA recalls the conventional moving cone technique. Indeed, both methods scan through each level of orebody from the top down and remove clusters of blocks until the final result is an ultimate pit, given that the total value of any cluster is positive. However, the advanced LP formulation of TCA allows TopCones to have two key advantages over the moving cones: the joint support ability and high-value blocks available for mining before low-value ones to facilitate the optimality of the downstream scheduling model.

**Proposition 3.5.3:** The TopCone Algorithm’s ultimate pits are near optimal but the optimality can be achieved by implementing TCA within an optimal LG ultimate pit limit

**Support 3.5.3:**

The non-optimality of TCA ultimate pits is expected because the flexibility of pit contours is reduced, as blocks are grouped into larger units. As a result, when MCS increases, TCA ultimate pits tend to have less value and more tonnage.

A 2D demonstration is given in Figures 3.24 and 3.25, where different pit limit solutions are generated using two MCS options. Further large-scale experiments of this phenomenon are discussed in the next chapter.

**Figure 3.24.** MCS of 1, the ultimate pit is comprised of four blocks and has value of $+4$
Figure 3.25. MCS of 5, the ultimate pit is comprised of nine blocks and has value of +2

This effect of MCS on TCA ultimate pit optimality, however, can be eliminated by implementing TCA within a predefined optimal ultimate pit. In Figure 3.24, it is obvious that the optimal ultimate pit consists of four blocks, namely 1, 2, 3, and 8. If TCA is implemented within this pit with MCS of 5, the relaxation on condition of a minimum size of TCs will be activated, so the final result is a TopCone consisting of four blocks, the same as the optimal solution given in Figure 3.24.

**Proposition 3.5.4:** TopCones are capable of joint support between positive nodes (i.e. ore blocks) in different levels

**Support 3.5.4:**

The joint support ability of positive nodes between different levels is enabled by constraints (3.4), where positive values of overlying nodes are merged into their corresponding underlying nodes. Consequently, the bonds between these ore blocks are established to support other waste blocks jointly.

\[ f_{ri} \leq V_i + \sum_{j \in O_i} V_j, \ \text{if} \ V_j > 0 \ \forall i \in N \]

Note that in case a positive overlying node has connection to more than one underlying node, its positive value is merged with the underlying node with the
highest cone value, which also means the smallest coefficient. This is to ensure the consistency with the role of coefficients to allow high-value blocks to be available for extraction before the lower ones.

Proposition 3.5.5: *TopCones are capable of joint support between positive nodes (i.e. ore blocks) at the same level*

Support 3.5.5:

In the objective function of LP formulation:

$$\text{Min} \sum_{i \in N} \sum_{j \in O_i} C_{i,j} f_{ij} + \sum_{i \in N} CP_i pf_i$$

Part 1 \hspace{1cm} Part 2

Under the minimisation function, the underlying node with the smallest coefficient is prioritised to send support to overlying nodes and establish connections first. If all of its overlying nodes are not yet fully supported, joint support is required from the next-smallest-coefficient underlying node which shares a mutual overlying node with the smallest-coefficient underlying node. Note that in one iteration, all underlying nodes are at the same level (i.e. termed as current searching level) so this joint support happens at the same level.

As discussed in Proposition 3.5.1, asking for support from pseudo flows resorts only when joint support from other underlying nodes is not available.

Proposition 3.5.6: *TopCones always obey slope constraints*

Support 3.5.6:

In Step 1 of TopCone Algorithm, overlying nodes on any underlying node are determined based on slope constraints. Therefore, when a TopCone is generated consisting of underlying and overlying nodes, slope constraints are automatically satisfied.

Proposition 3.5.7: *The proposed mine planning optimisation method can be applied to solve any large-scale problems*
Support 3.5.7:

By varying MCS at post-processing step, the number of TCs generated is controlled although the desired number is only achieved by trial and error. Theoretically, TCA can reduce the number of TCs as close as or equal to 1 when MCS is set large enough. The smallest value can be one only if the shape of the ultimate pit limit allows all blocks to be connected to a root block according to the slope constraints. That theoretical ultimate pit limit has a cone shape with an appropriate slope angle and an ore block located at the bottom. This ability allows TCA to convert any large-scale block models into a small number of TCs so when the IP model is formulated, the number of binary variables can be appropriate in order to yield a solution within a practical time.

In addition, the nature of TCA is linear programming, the most robust programming technique, so the solution time of generating TCs is not a major concern. Theoretically, given a computer with sufficient memory capacity, TCA can quickly yield a solution for any large-scale block models. Together with the ability to control the number of TCs generated, TCA-based production scheduling framework can be applied to any large-scale block models to obtain a solution within the practical time frame.

Proposition 3.5.8: Selecting an appropriate Minimum Cone Size parameter largely depends on the size of input block model, computing facility, and the amount of time allowed to solve the mine planning problem

Support 3.5.8:

Selecting an appropriate MCS parameter is a case-specific decision under the consideration of the size of input block model, computing facility, and the amount of time allowed to solve the mine planning problem. Generally, a very large MCS parameter will return a very small number of TCs and hence facilitate solving the downstream scheduling model quickly. However, a too small number of TCs may cause the scheduling model infeasible, depending on the narrowness of upper and lower bound constraints.
This chapter discussed a new block aggregation technique, called TopCone Algorithm (TCA), for the application of operations research techniques in solving open pit production scheduling problems. There are two most important features of TopCones (TCs): (1) Cone shape and (2) the number can be controlled. While feature (1) is to ensure consistency with slope constraints of downstream scheduling model, feature (2) is critical to keep the size of scheduling model tractable so feasible solution can be obtained in a practical timeframe.

One interesting feature of TCA is its ability in determining a near-optimal ultimate pit limit. Despite the near-optimality, this opens a simpler and easier mine planning framework consisting of two stages: Ultimate pit can be obtained during block aggregation process, and production scheduling is achieved by formulating and solving mathematical programming models on aggregated blocks.

The development of TCA is a novel platform to the application of mathematical programming techniques to solve large-scale mine planning problems: (1) the long-standing obstacle of the intensity of data is completely eliminated as blocks are clustered into a very small number (the users can actually control this number) so that production scheduling solutions can be obtained in a practical timeframe; (2) slope angles are incorporated during the aggregation process to eliminate the conflict between block aggregation and production scheduling in terms of slope constraints; (3) the elimination of pit optimisation stages, i.e. ultimate pit limit and pushback design, is clearly beneficial in terms of reducing the complexity of mine planning framework, time and financial expenditures. It is well-known that pushback design is not an optimal method, therefore, by discarding this step in the proposed mine planning framework using TCA, the result is beneficial towards global optimality.
CHAPTER 4. DETERMINISTIC OPTIMISATION OF LONG-TERM PRODUCTION SCHEDULING USING TOPCONE ALGORITHM AND INTEGER PROGRAMMING

4.1 INTRODUCTION

In this chapter, an optimisation method of deterministic long-term production scheduling using TopCones (TCs) and integer programming (IP) is discussed. Firstly, formulation of the IP model which takes TCs as input data and find optimal production schedule is provided. Then the performance of TopCone Algorithm (TCA)-based IP mine planning model is tested on the Silvergrass East iron ore deposit, Pilbara region, Western Australia. The code of this optimisation model written in VB.Net programming language is presented in Appendix A.

4.2 MATHEMATICAL FORMULATION OF INTEGER PROGRAMMING

In the scheduling model, each TC is considered as a mining unit and be taken at any period over the life of mine. TC’s ore, waste tonnage and economic value are calculated by summing up corresponding values of its member blocks. TC’s grades are assumed to be homogeneous within its ore material and are calculated by averaging grade from its member ore blocks. Because the whole TC is extracted at once when it is scheduled in a time period, the assumption of homogeneity has no impact on the scheduling result.

The objective of the IP model is to maximise net present value (NPV) subject to 8 production targets of ore tonnage, total tonnage, and blending grades of Fe, SiO₂, Al₂O₃, P, loss on ignition (LOI) and Ochreous Goethite (GOL) as presented below:

**Notation**

*Indices and sets*

\[ i \in N \] set of TCs i  
\[ j \in M_i \subseteq N \] subset of TCs j that are predecessor TCs for TC i  
\[ t, t' \in P \] set of time periods t, t’ in the horizon
\( k \in K \) set of \( K \) blending grade targets (\( k=1: \) Fe, \( k=2: \) SiO\(_2\), \( k=3: \) Al\(_2\)O\(_3\), \( k=4: \) P, \( k=5: \) LOI, \( k=6: \) GOL)

**Parameters**

\( V_i \) economic value of TC \( i \)

\( d \) economic discount rate, %

\( G_i^k \) grade \( k \) of TC \( i \), \( k = 1, ..., K \)

\( O_i \) ore tonnage of TC \( i \)

\( W_i \) waste tonnage of TC \( i \)

\( G_{\min}^k / G_{\max}^k \) minimum/maximum blending grade, \( k = 1, ..., K \)

\( PC_{\min} / PC_{\max} \) minimum/maximum processing plant capacity

\( MC_{\min} / MC_{\max} \) minimum/maximum capacity of the mine’s equipment

**Binary variables**

\( x_i^t \) equal to 1 if TC \( i \) is scheduled in period \( t \); 0 otherwise.

**Objective function**

\[
\text{Max } \sum_{t \in P} \sum_{i \in N} \frac{1}{\left(1 + \frac{d}{100}\right)^t} V_i x_i^t
\]

**Subject to**

**Processing capacity constraints**

\[
\sum_{i \in N} \left(O_i x_i^t\right) \geq PC_{\min} \quad \forall t \in P
\]

Lower Bound (4.2)
\[ \sum_{i \in N} (O_i \cdot x_i^t) \leq PC_{\text{max}} \quad \forall t \in P \]  \hspace{1cm} \text{Upper Bound} \quad (4.3)

**Mining capacity constraints**

\[ \sum_{i \in N} (O_i + W_i)x_i^t \geq MC_{\text{min}} \quad \forall t \in P \]  \hspace{1cm} \text{Lower Bound} \quad (4.4)

\[ \sum_{i \in N} (O_i + W_i)x_i^t \leq MC_{\text{max}} \quad \forall t \in P \]  \hspace{1cm} \text{Upper Bound} \quad (4.5)

**Blending grade constraints**

\[ \sum_{i \in N} (G_i^k - G_{\text{min}}^k)O_i \cdot x_i^t \geq 0 \quad \forall t \in P, \forall k = 1, \ldots, K \]  \hspace{1cm} \text{Lower Bound} \quad (4.6)

\[ \sum_{i \in N} (G_i^k - G_{\text{max}}^k)O_i \cdot x_i^t \leq 0 \quad \forall t \in P, \forall k = 1, \ldots, K \]  \hspace{1cm} \text{Upper Bound} \quad (4.7)

**Sequencing constraints**

\[ M_i \cdot x_i^t - \sum_{j \in M_i} \sum_{t' \in t} X_{j t'}^t \leq 0 \quad \forall i \in N, \forall t \in P \]  \hspace{1cm} (4.8)

**Resource constraints**

\[ \sum_{t \in P} x_i^t \leq 1 \quad \forall i \in N \]  \hspace{1cm} (4.9)

**Integrality constraints**

\[ x_i^t = \{0,1\} \text{ and } x_j^{t'} = \{0,1\} \quad \forall i, j \in N, \forall t, t' \in P \]  \hspace{1cm} (4.10)

The objective function (4.1) is a maximisation of discounted cash flow from mining \( N \) TCs over \( P \) periods of time. Constraints (4.2), (4.3), (4.4), and (4.5) limit production of ore and total tonnage in each period. Constraints (4.6) and (4.7) enforce the average grade of material sent to mill. Constraints (4.8) ensure mining sequence between TCs,
where a TC can be extracted at period $t$ only if its entire predecessor TCs is scheduled not later than that TC. In addition, a TC is considered to be a predecessor of another if it has at least one block overlying any block of that underlying TC according to slopes. Constraints (4.9) ensure that a TC is extracted in one period only. Constraints (4.10) enforce integrality of binary variables.

4.3 COMPUTATIONAL EXPERIMENTS

The case study is provided by Rio Tinto Iron Ore (RTIO), containing a part of the Silvergrass East iron ore deposit, Pilbara, Western Australia.

4.3.1 General information of Silvergrass East deposit

The Silvergrass deposits are located approximately 65 km north-west of the Tom Price Township, and 15 km north of the Brockman 2 Mine. The Silvergrass East orebody lies on the eastern flank of the Silvergrass project area, which also includes the Silvergrass West orebody. The location plan of the Silvergrass deposits is presented in Figure 4.1.
The Silvergrass East deposit has 2,717 drill holes, of which 1,574 are provided by RTIO as a case study for this thesis, as shown in Figure 4.2.

![Silvergrass East drill hole map and study area (red polygon)](image)

**Figure 4.2.** Silvergrass East drill hole map and study area (red polygon) (*Source: RTIO*)

A resource block model constructed for the Silvergrass East deposit consists of 985,088 blocks with a regular block size of 25 x 12.5 x 10 m. The geological attributes of block cells are estimated by the ordinary kriging technique following the standard procedure, including statistical analysis, variography, kriging and validation (Goovaerts, 1997). This is a standard and popular procedure, so these resource estimation works are not going to be mentioned in this thesis.

### 4.3.2 Numerical experiments

In numerical experiments, the proposed deterministic mine planning method using TCA and IP was implemented multiple times using 7 MCS options of 1, 50, 100, 200, 300, 400, and 500 blocks per TC. The ultimate pits generated by TCA were compared with an ultimate pit limit generated by Whittle software, while IP-based scheduling solutions were compared with the both Milawa NPVTM and Milawa BalancedTM scheduling algorithms (Whittle, 2016). The long-term mine plan consists of 8 periods (years) with a discount rate of the dollar value of 10%. A hypothetical set of scheduling parameters is presented in Table 4.1. All experiments were conducted
using a normal office computer having an Intel(R) Core(TM) i7 with 3.4 GHz CPU processor and 8 Gb of RAM.

**Table 4.1.** Production scheduling parameters

<table>
<thead>
<tr>
<th>Production targets</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min/Max mining capacity (Mt)</td>
<td>30</td>
<td>75</td>
</tr>
<tr>
<td>Min/Max processing capacity (Mt)</td>
<td>22</td>
<td>28</td>
</tr>
<tr>
<td>Fe (%)</td>
<td>58.5</td>
<td>60.5</td>
</tr>
<tr>
<td>SiO₂ (%)</td>
<td>0</td>
<td>5.7</td>
</tr>
<tr>
<td>Al₂O₃ (%)</td>
<td>0</td>
<td>2.8</td>
</tr>
<tr>
<td>P (%)</td>
<td>0</td>
<td>0.062</td>
</tr>
<tr>
<td>LOI (%)</td>
<td>0</td>
<td>6.8</td>
</tr>
<tr>
<td>GOL (%)</td>
<td>0</td>
<td>15.2</td>
</tr>
</tbody>
</table>

Table 4.2 presents the implementation of the proposed mine planning method. In each run, the following figures are reported:

- **Number of TCs generated by TCA**

- **Number of binary variables**: Equal to the number of TCs multiplied by the number of time periods.

- **Number of precedences**: Number of precedence relationships between TCs.

- **Number of blocks within TCA’s ultimate pits**

- **Ore tonnage of TCA’s ultimate pits (million tonne)**

- **Waste tonnage of TCA’s ultimate pits (million tonne)**

- **Solution time of TCA in clustering blocks into TCs (minutes)**

- **Gap to Whittle’s ultimate pit value (%)**

- **Solution time of IP-based scheduling model using TCs to obtain a solution within 5% optimality gap (minutes)**
Table 4.2. The implementation of TCA and IP with various MCS options

<table>
<thead>
<tr>
<th>MCS</th>
<th>0</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of TCs</td>
<td>20,404</td>
<td>4,592</td>
<td>1,526</td>
<td>1,074</td>
<td>766</td>
<td>654</td>
<td>588</td>
<td>530</td>
</tr>
<tr>
<td>No. of binary vars</td>
<td>163,232</td>
<td>36,736</td>
<td>12,208</td>
<td>8,592</td>
<td>6,128</td>
<td>3,896</td>
<td>4,704</td>
<td>4,240</td>
</tr>
<tr>
<td>No. of precedences</td>
<td>9,331</td>
<td>7,003</td>
<td>2,244</td>
<td>1,952</td>
<td>1,443</td>
<td>1,224</td>
<td>1,000</td>
<td>810</td>
</tr>
<tr>
<td>No. of blocks of TCA pit</td>
<td>62,988</td>
<td>62,826</td>
<td>63,586</td>
<td>63,049</td>
<td>62,920</td>
<td>63,313</td>
<td>63,378</td>
<td>63,427</td>
</tr>
<tr>
<td>Ore tonnage (Mt)</td>
<td>232.4</td>
<td>232.2</td>
<td>233.0</td>
<td>233.0</td>
<td>232.4</td>
<td>232.4</td>
<td>232.3</td>
<td>232.3</td>
</tr>
<tr>
<td>Waste tonnage (Mt)</td>
<td>203.1</td>
<td>202.2</td>
<td>206.7</td>
<td>206.7</td>
<td>203.6</td>
<td>203.6</td>
<td>202.8</td>
<td>202.8</td>
</tr>
<tr>
<td>Sol. time of TCA (mins)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Gap to Whittle ult. Pit</td>
<td>-0.002</td>
<td>-0.020</td>
<td>-0.025</td>
<td>-0.026</td>
<td>-0.002</td>
<td>-0.042</td>
<td>-0.073</td>
<td>-0.061</td>
</tr>
<tr>
<td>Sol. time of IP (mins)</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>474</td>
<td>242</td>
<td>111</td>
<td>57</td>
<td>40</td>
</tr>
<tr>
<td>IP final gap %</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>4.27</td>
<td>3.83</td>
<td>2.10</td>
<td>2.12</td>
<td>2.96</td>
</tr>
<tr>
<td>TCA-IP's NPV (M$)</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>1,889.7</td>
<td>1,891.8</td>
<td>1,888.3</td>
<td>1,897.5</td>
<td>1,914.7</td>
</tr>
</tbody>
</table>

Regarding ultimate pit limit, it can be seen from Table 4.2 that TCA is able to determine near optimal ultimate pit solutions with approximately 0.1% less value than optimality. The lower value of TCA’s ultimate pits is expected due to the reduced flexibility of pit contours as blocks are aggregated into larger units. The experiments also demonstrate the tendency of losing more ultimate pit value when MCS increases. Given that the proposed method does not require pit optimisation stages, namely ultimate pit limit and pushback design, this simplification means less expenditure spent for software and algorithm. This benefit is worth considering in exchange for a relatively small loss of ultimate pit value. On the other hand, TCA can also be implemented within a predefined optimal pit limit. Regarding production scheduling, multiple runs with different MCS options prove the ability of TCA in controlling the number of aggregates generated and significantly reduce the intensity of production scheduling problems. To be more specific, at MCS of 1 or there is no restriction on the size of TCs, 20,404 TCs were generated, and the corresponding TCA ultimate pit limit consists of 62,988 blocks, where the block-to-TopCone ratio is 3. However, at MCS of 500, only 530 TCs were generated, and the ratio is approximately 120. As the scale of the scheduling problem is relatively small, the whole two-stage mine planning process was completed in 51 minutes. Given the limitation of computing power, that timeframe is very practical.
These comparisons of blending grade between TCA-IP and Whittle Milawa NPV, Milawa Balanced scheduling methods are given in Figure 4.3. Obviously, the ability of the proposed TCA-IP model in meeting blending grade constraints is well proven in all MCS options. Meanwhile, both two Milawa algorithms show violations of blending grade targets on many occasions:

- **Fe grade**: Milawa NPV and Milawa Balanced violate the lower bound at periods 1 and 7.

- **SiO₂ grade**: Milawa NPV and Milawa Balanced violate the upper bound at periods 1 and 2.

- **Al₂O₃ grade**: Milawa NPV violates the upper bound at periods 6 and 7, Milawa Balanced violates the upper bound at periods 5 and 6.

- **P grade**: Milawa NPV violates the upper bound at periods 4, 5, and 6, Milawa Balanced violates the upper bound at periods 7 and 8.

- **LOI grade**: Milawa Balanced violates the upper bound at period 6.

- **GOL grade**: Milawa NPV violates the upper bound at periods 3, 4, and 5, Milawa Balanced violates the upper bound at period 3.
Figure 4.3. Comparison of blending grades between Whittle Milawa NPV, Milawa Balanced and TCA-IP models with different MCS options.
In iron ore mining, it is common to apply a penalty for ore tonnage which does not meet grade requirements, i.e. Fe grade is too low or impurity grade is too high. The bonus/penalty factors used in this study are presented in Table 4.3.

**Table 4.3. Bonus/Penalty for blending grades**

<table>
<thead>
<tr>
<th>Spec</th>
<th>Min/max</th>
<th>Bonus</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>blending grade</td>
<td></td>
</tr>
<tr>
<td>Fe</td>
<td>58.5-60.5</td>
<td>1$/ton for each 1% above 58.5%</td>
<td>1$/ton for each 1% below 58.5%</td>
</tr>
<tr>
<td>SiO2</td>
<td>0-5.7</td>
<td>-</td>
<td>0.05$/ton for each 1% above 5.7%</td>
</tr>
<tr>
<td>Al2O3</td>
<td>0-2.8</td>
<td>-</td>
<td>0.05$/ton for each 1% above 2.8%</td>
</tr>
<tr>
<td>P</td>
<td>0-0.062</td>
<td>-</td>
<td>0.05$/ton for each 1% above 0.062%</td>
</tr>
<tr>
<td>LOI</td>
<td>0-6.8</td>
<td>-</td>
<td>0.05$/ton for each 1% above 6.8%</td>
</tr>
<tr>
<td>GOL</td>
<td>0-15.2</td>
<td>-</td>
<td>0.05$/ton for each 1% above 15.2%</td>
</tr>
</tbody>
</table>

A comparison of NPV values of the TCA-based IP model versus Whittle Milawa NPV and Whittle Milawa Balanced methods is provided in Table 4.4. When blending grade bonus and penalties are not accounted, the proposed method generated compatible results with Milawa NPV while higher than Milawa Balanced in all experiments. When the bonus and penalties are accounted into NPV calculation, the NPV scores of two Milawa methods are reduced considerably due to the violations of blending grades as shown in Figure 4.3. Consequently, TCA-IP yielded 5.5-7.0% NPV higher than Milawa NPV and 6.5-8.0% NPV higher than Milawa Balanced.

**Table 4.4. Comparison of NPV between Milawa NPV, Milawa Balanced and TCA-IP with and without grade bonus and penalty**

<table>
<thead>
<tr>
<th>MCS</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gap to Whittle Milawa NPV schedule</td>
<td>-0.460</td>
<td>-0.533</td>
<td>0.855</td>
<td>0.727</td>
<td>-0.183</td>
</tr>
<tr>
<td>Gap to penalised Whittle Milawa NPV schedule</td>
<td>5.646</td>
<td>5.765</td>
<td>5.569</td>
<td>6.084</td>
<td>7.042</td>
</tr>
<tr>
<td>Gap to Whittle Milawa Balanced schedule</td>
<td>1.816</td>
<td>1.742</td>
<td>3.161</td>
<td>3.030</td>
<td>2.099</td>
</tr>
</tbody>
</table>

Plan view and typical cross-sections of the mine plan suggested by TCA-IP model with MCS of 500 are presented in Figures 4.4 and 4.5.
4.5 CONCLUSIONS

In this chapter, the performance of the proposed deterministic mine planning framework using TopCone Algorithm (TCA) and integer programming (IP) on a large-scale instance was intensively investigated. By varying Minimum Cone Size (MCS),
TCA was able to reduce the intensity of scheduling problem significantly. Therefore, a high-quality solution was obtained in a very practical timeframe.

The relationship between MCS and the quality of ultimate pits and scheduling solutions was also investigated. As a result, a high MCS leads to a small number of TopCones (TCs) generated and hence reduce solution time of solving IP models exponentially. However, this also tends to reduce the value of ultimate pit and project’s NPV. As discussed previously, LG can also be used prior to TCA to eliminate the lost value of ultimate pits.

Comparisons with Whittle Milawa NPV proved that the proposed mine planning method was able to obey the operational constraints strictly. Therefore, if all the violations of constraints are penalised into NPV calculation, the TCA-IP mine planning method generated considerably higher NPV score than the commercial scheduler at a magnitude of 5.5% to 7%.
CHAPTER 5. MULTIVARIATE CONDITIONAL SIMULATION AND THE APPLICATION ON IRON ORE DEPOSIT

5.1 INTRODUCTION

To characterise the risk associated with geology, conditional simulation has been proven to be one of the best geostatistical approaches (Dimitrakopoulos, 1998; Dowd, 1994, 1997; Goovaerts, 1997; Journel, 1974; Journel & Huijbregts, 1978). Conditional simulation is a class of Monte Carlo technique (Halton, 1970) that can be used to generate a series of equally probable possibilities, or realisations, of an orebody, each of which has an equal chance to occur in reality. The term “conditional” indicates that the simulation honours the sampling data where the simulated and sampling points are co-located, and this is a key requirement for the simulation techniques in ore reserve estimation.

It is common for mineralised deposits that attributes of interest are cross-correlated, typically iron ore deposits. For example with iron ore deposits, strong correlations between variables of interest, including Fe, SiO₂, Al₂O₃, P, LOI and others, can be clearly observed (Mai et al., 2016a, 2016b). As the nature of simulation is random, performing simulation on these variables without bearing the correlation will return an unrealistic result (Mai et al., 2016a). Consequently, the presence of correlation leads to a new branch of simulation techniques, which is commonly referred to as joint simulation, co-simulation or multivariate conditional simulation (Almeida & Journel, 1994; M. E. Johnson, 1987; Myers, 1989; Verly, 1993).

In this chapter, firstly methodology of common joint simulation techniques is reviewed. By comparing the requirements of these techniques with typical features of iron ore deposits, a multivariate conditional simulation framework is proposed. Finally, the application of the proposed framework on the Silvergrass East iron ore deposit is presented.

5.2 METHODOLOGY REVIEW

There are several joint simulation techniques are developed to preserve the correlation of variables, and they are generally divided into two groups:
a. **Co-simulation method**

The basis of co-simulation approach is that the simulation algorithms are applied directly on correlated variables using cross-variograms and co-kriging matrices (Carr & Myers, 1985; Matheron, 1979; Verly, 1993). Thus the inter-variable correlations are preserved in the output realisations. Main drawbacks of this method are:

- Determining fitting co-regionalisation models for cross-variograms is a complicated task even with the help of advanced geostatistical tools, such as ISATIS™ (Bleines & de Paris, 2000). Especially, when there are more than three variables of interest, building acceptable co-regionalisation models becomes difficult and time-consuming.

- Dowd (1985) points out that to efficiently apply co-simulation, the correlation of variables must be linear.

- Co-simulation technique becomes computationally inefficient when there are more than three variables (De-Vitry et al., 2007).

b. **Data transformation method**

The basic idea of this group is to transform correlated variables into non-correlated factors before feeding into a simulation model. After simulation, the simulated values are back-transformed to gain original characteristics. There are several well-tested techniques of this kind, including:

- **Turning bands (TB):** This method generates arrays of one-dimensional simulated values which are regularly distributed to produce simulation results in two or three-dimensional space (Journel & Huijbregts, 1978). While computational efficiency is the principal advantage of this method, the results in three-dimensional space are relatively poor, particularly the unwanted anisotropy (Davis, 1987; Tompson et al., 1989)

- **Lower/Upper (LU) decomposition:** The LU decomposition involves decomposing the covariance matrix of correlated variables into a lower and an upper triangular matrix, then conditional realisations are generated as a linear function of these matrices and parametric vectors. Detailed steps of LU decomposition method
is given by Goovaerts (1997). Due to a computational limitation of the decomposition of covariance matrix, this method is only applicable to relatively small instances in both terms of number of variables and grid size.

- **Principal Component Analysis (PCA):** The aim of PCA is the transformation of original spatially correlated variables into uncorrelated or non-negligible spatial correlation variables called principal components (Goovaerts, 1993; Joreskog et al., 1976). However, this method only successfully de-correlates variables at zero or small lag distance (Goovaerts, 1993), which restricts the application of PCA on practical mining case study.

- **Minimum/maximum autocorrelation factors (MAF):** Derived from PCA, MAF uses a two-structure linear model of co-regionalisation to transform correlated variables into factors that have no spatial cross-correlation at any lag distance (Desbarats & Dimitrakopoulos, 2000). MAF has been widely applied in the mining industry in general and iron ore deposits in particular (Bandarian et al., 2008; Boucher, 2003; Boucher & Dimitrakopoulos, 2009, 2012; Dimitrakopoulos & Fonseca, 2003; Rondon, 2012).

- **Stepwise Conditional Transformation (SCT):** Leuangthong and Deutsch (2003) proposed a transformation method that removes the spatial correlation between variables meanwhile converting them into normal score space. There are two main drawbacks of SCT. Firstly, de-correlation can be well obtained at zero lag but not satisfactory at non-zero lags. Secondly, to ensure the reliability of conditional distribution, the number of samples should be $10^N$ to $20^N$, with $N$ is the number of variables (Leuangthong, 2003). When dealing with more than five correlated variables, the required number of samples is rarely satisfied.

**c. Conclusion of relevant method**

The typical features of simulating iron ore datasets are: 1) at least five variables are spatially correlated, and 2) the number of samples is not very abundant. In the specific case of Silvergrass East, there are six cross-correlated variables and 19,652 samples. Considering the features of the abovementioned joint simulation techniques, the conclusions are as follows:
- The simulation results of TB are relatively poor.
- LU decomposition method is not capable of handling such intensity of the given dataset.
- Co-simulation is not suitable to deal with six variables as the computational work will be too intense and there is no guarantee about the quality of the six-variable cross-variograms.
- PCA only successfully de-correlates variables at zero or small lag distance.
- There are not enough samples as required by SCT. Also, this approach is proved to be not effective at a non-zero lag distance.
- After all, MAF approach has the highest potential for iron ore deposits: It is capable of de-correlating at any lag distance, suitable to work with six variables and the given number of sampling data. This technique is also available in ISATIS\textsuperscript{TM}, a commercial geostatistical package.

In this study, MAF will be used as a de-correlation tool to handle the cross-correlation of variables before running simulation. As a requirement of MAF, the input data must be transformed into a normal score space (Gaussian) so it would be the most efficient to select a Gaussian-based simulation technique to do the downstream simulation part. Sequential Gaussian simulation (SGS) is arbitrarily selected in this study as its efficiency has been widely confirmed (Dimitrakopoulos, 1998; Dimitrakopoulos et al., 2002; Dowd, 1994; M. E. Johnson, 1987; Journel, 1994).

5.3 PROPOSED SIMULATION FRAMEWORK USING SGS AND MAF

5.4.1 Normal score transformation

Gaussianity is required for all variables before applying MAF and for all Gaussian-based simulation techniques. A distribution has a Gaussian (normal) feature if it has zero mean and variance is one. Goovaerts (1997) introduces a convenient technique for normal score transformation, called graphical transformation. The transformation methodology can be presented as follows:
Let $F(z)$ be the stationary one-point conditional distribution function (CDF) of original data $Z(u)$ and $G(y)$ be the stationary one-point CDF of a standard normal data $Y(u)$.

\begin{align*}
F(z) &= \text{prob}[Z(u) \leq z] \\
G(y) &= \text{prob}[Y(u) \leq y]
\end{align*}

(5.1)

(5.2)

Based on graphical correspondence, let $\phi$ be the normal score transform function which allows any equal $p$-quantile from $F(z)$ can be located at $G(y)$:

\[ Y(u) = \phi(Z(u)) = G^{-1}[F(Z(u))] \]

(5.3)

where $G^{-1}$ is the inverse Gaussian function of $Y(u)$. In other words, at certain $p$-quantile, two CDFs receive equal value:

\[ G(y_q) = G[G^{-1}(F(z_p))] = F(z_p) = p \]

(5.4)

After normal score transformation of all original variables, it is necessary to check the correlation of normal scores and compare to those of original variables. If the cross-correlations are not reasonably preserved, one should consider other non-Gaussian simulation techniques (Deutsch & Journel, 1998).

Back transform:

\[ z^*_i = F^{-1}(G(y^*_i)) \]

(5.5)

where $y^*_i$ is simulated data and $z^*_i$ is back transformed simulated data.

5.4.2 MAF transformation

The MAF factors are defined as $p$ orthogonal linear combinations $Y_i(x) = a_i^T Z(x)$, $i=1,\ldots, p$ of sampling data $Z(x) = [Z_1(x), \ldots, Z_p(x)]^T$ over a study region $A$. Transform factors $Y_i(x)$ are defined by:

\[ Y_i(x) = A^T Z(x) \]

(5.6)

where $A^T$ is transformation matrix.
A computational approach to calculate $A^T$ is proposed by Switzer and Green (1984), which can be implemented in five steps as follows (Desbarats & Dimitrakopoulos, 2000):

**Step 1.** Decompose the variance-covariance matrix $B$ of sampling data $Z(x)$ into eigenvectors matrix $H$ and eigenvalues matrix $D$

$$B = HDH^T \quad (5.7)$$

**Step 2.** Calculate the transformed variables $V(x) = W^T Z(x)$, where $W = HD^{-1/2}$. Note that $W^T BW = I$

**Step 3.** Calculate covariance matrix of $V(x)$ of the lag $\Delta$ using a two-structure linear co-regionalisation model, given that $\rho_1(\Delta)$ is the spatial correlation between $Y_i(x)$ and $Y_i(x+h)$ for a short lag $h = \Delta$

$$\text{cov}[V(x)-V(x+\Delta),(V(x)-V(x+\Delta))] = 2(1-\rho_1(\Delta))I + 2(\rho_1(\Delta)-\rho_0(\Delta))W^T B_0 W \quad (5.8)$$

**Step 4.** Calculate the spectral decomposition of the covariance matrix of $(V(x)-V(x+\Delta))$ into a matrix of orthogonal eigenvectors $C$ and a diagonal matrix of eigenvalues $\Lambda$

$$(1-\rho_1(\Delta))I + 2(\rho_1(\Delta)-\rho_0(\Delta))W^T B_0 W = C \frac{\Lambda}{2} C^T \quad (5.9)$$

$$\Leftrightarrow (1-\rho_1(\Delta))I + 2(\rho_1(\Delta)-\rho_0(\Delta))C^T W^T B_0 WC = \frac{\Lambda}{2} \quad (5.10)$$

**Step 5.** Finally, calculate transformation matrix $A^T$

$$A^T = C^T D^{-1/2} H^T \quad (5.11)$$

Transform factors $Y_i(x)$ are de-correlated at all lag distance; hence, forward procedures are carried out as univariate tasks. More details about MAF transformation can be found on the work of Desbarats and Dimitrakopoulos (2000).
5.4.3 Joint simulation steps

After MAF transformation, the factors are no longer spatially correlated and can be simulated using any direct Gaussian simulation techniques like independent variables. The whole joint simulation procedure can be summarised in the following six steps and systematically illustrated in Figure 5.1.

Figure 5.1. Schematic presentation of the proposed simulation framework

(NS stands for normal score; Transf. stands for transformation; anam. stands for anamorphosis) (after (Mai et al., 2016b))

Step 1. Primary Gaussian transformation: Transform sample data $Z(x)$ into normal (Gaussian) scores $Y(x)$:

$$Y(x) = \phi(Z(x)) \quad (5.12)$$

Step 2. MAF transformation: Transform $Y(x)$ into uncorrelated MAF factors $M(x)$

$$M(x) = A^T(Y(x)) \quad (5.13)$$

Step 3. Secondary Gaussian transformation: Implement another Gaussian transformation if the Gaussianity of MAF factors are not adequate. After this transformation, the output data is called normal score MAF factors $N(x)$:

$$N(x) = \phi(M(x)) \quad (5.14)$$

Step 4. Implement the continuity analysis and construct variography of normal score
MAF factors $N(x)$

Step 5. Perform SGS on each normal score MAF factors $N(x)$ individually;

$$N^*(x) = y(N(x))$$ (5.15)

Step 6. Back transformation of secondary Gaussian back transformation (if available); MAF back transformation; and primary Gaussian back transformation.

Where: $Z(x)$: original data; $Y(x)$: normal scores; $M(x)$: MAF factors; $N(x)$: normal score MAF factors; $N^*(x)$: simulated data of $N(x)$; $\phi$: primary Gaussian anamorphosis; $\phi'$: secondary Gaussian anamorphosis; $A^T$: MAF transformation matrix; $y$: sequential Gaussian simulation

It is worth noting that although normal score MAF factors are simulated independently, they are parts of a mutual dataset. Therefore, after simulation, each simulated normal score MAF factors need to be grouped in a correct order for each realisation to be properly back transformed.

5.4 A MULTIVARIATE CONDITIONAL SIMULATION CASE STUDY OF SILVERGRASS EAST DEPOSIT

5.4.1 Statistical analysis

The statistics of sampling data are presented in Table 5.1, the Pearson correlations and histograms of six variables of interest, namely Fe, SiO$_2$, Al$_2$O$_3$, P, LOI and GOL, are presented in Table 5.2 and Figure 5.2, respectively.

Table 5.1. Statistics of attributes inside ore domain

<table>
<thead>
<tr>
<th>Variable</th>
<th>Samples Count</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE</td>
<td>19625</td>
<td>0</td>
<td>66.66</td>
<td>59.23</td>
<td>5.17</td>
<td>26.77</td>
</tr>
<tr>
<td>SiO$_2$</td>
<td>19625</td>
<td>0</td>
<td>55.32</td>
<td>4.94</td>
<td>3.92</td>
<td>15.36</td>
</tr>
<tr>
<td>Al$_2$O$_3$</td>
<td>19625</td>
<td>0</td>
<td>20.45</td>
<td>2.59</td>
<td>2.22</td>
<td>4.94</td>
</tr>
<tr>
<td>P</td>
<td>19625</td>
<td>0</td>
<td>0.84</td>
<td>0.06</td>
<td>0.02</td>
<td>0</td>
</tr>
<tr>
<td>LOI</td>
<td>19625</td>
<td>0</td>
<td>13.47</td>
<td>6.49</td>
<td>1.84</td>
<td>3.38</td>
</tr>
<tr>
<td>GOL</td>
<td>19625</td>
<td>0</td>
<td>100</td>
<td>13.86</td>
<td>13.55</td>
<td>183.65</td>
</tr>
</tbody>
</table>
Figure 5.2. Histograms of six original variables
Table 5.2. Pearson correlation matrix of borehole samples within the ore domain of Silvergrass East deposit (Mai et al., 2016b).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Fe</th>
<th>SiO₂</th>
<th>Al₂O₃</th>
<th>P</th>
<th>LOI</th>
<th>GOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fe</td>
<td>1</td>
<td>-0.72</td>
<td>-0.67</td>
<td>0.19</td>
<td>-0.45</td>
<td>0</td>
</tr>
<tr>
<td>SiO₂</td>
<td>-0.72</td>
<td>1</td>
<td>0.5</td>
<td>-0.23</td>
<td>0.21</td>
<td>-0.02</td>
</tr>
<tr>
<td>Al₂O₃</td>
<td>-0.67</td>
<td>0.5</td>
<td>1</td>
<td>-0.14</td>
<td>0.39</td>
<td>-0.08</td>
</tr>
<tr>
<td>P</td>
<td>0.19</td>
<td>-0.23</td>
<td>-0.14</td>
<td>1</td>
<td>0.19</td>
<td>0.17</td>
</tr>
<tr>
<td>LOI</td>
<td>-0.45</td>
<td>0.21</td>
<td>0.39</td>
<td>0.19</td>
<td>1</td>
<td>0.24</td>
</tr>
<tr>
<td>GOL</td>
<td>0</td>
<td>-0.02</td>
<td>-0.08</td>
<td>0.17</td>
<td>0.24</td>
<td>1</td>
</tr>
</tbody>
</table>

From the histograms in Figure 5.2, some outliers of sample set can be observed. Although outliers cause a negative impact on simulation quality, eliminating them also reduces the number of valuable sampling data and may incidentally discard some important information of the deposit. In this study, all outliers were kept in the dataset, and their impact on simulation result was analysed later.

5.4.2 Data transformation

a. Primary Gaussian Transformation

Histograms and correlation matrix of primary Gaussian transformed variables are presented in Figure 5.3 and Table 5.3
Figure 5.3. Histogram of primary Gaussian transformed variables
## Table 5.3. Correlation matrix of primary Gaussian transformed variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Gaus Fe</th>
<th>Gaus SiO₂</th>
<th>Gaus Al₂O₃</th>
<th>Gaus P</th>
<th>Gaus LOI</th>
<th>Gaus GOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaus Fe</td>
<td>1</td>
<td>-0.81</td>
<td>-0.73</td>
<td>0.24</td>
<td>-0.56</td>
<td>0.01</td>
</tr>
<tr>
<td>Gaus SiO₂</td>
<td>-0.81</td>
<td>1</td>
<td>0.62</td>
<td>-0.29</td>
<td>0.28</td>
<td>-0.03</td>
</tr>
<tr>
<td>Gaus Al₂O₃</td>
<td>-0.73</td>
<td>0.62</td>
<td>1</td>
<td>-0.16</td>
<td>0.43</td>
<td>-0.08</td>
</tr>
<tr>
<td>Gaus P</td>
<td>0.24</td>
<td>-0.29</td>
<td>-0.16</td>
<td>1</td>
<td>0.2</td>
<td>0.23</td>
</tr>
<tr>
<td>Gaus LOI</td>
<td>-0.56</td>
<td>0.28</td>
<td>0.43</td>
<td>0.2</td>
<td>1</td>
<td>0.24</td>
</tr>
<tr>
<td>Gaus GOL</td>
<td>0.01</td>
<td>-0.03</td>
<td>-0.08</td>
<td>0.23</td>
<td>0.24</td>
<td>1</td>
</tr>
</tbody>
</table>

The correlations of transformed variables are very similar to the original variables as shown in Table 5.2. This proves the quality of the primary Gaussian transformation step, where the correlations of variables are well reserved.

### b. MAF Transformation

Tables 5.4 and 5.5 present MAF transformation matrices from variables to factors and vice versa, respectively.

## Table 5.4. MAF transformation matrix from variables to factors:

<table>
<thead>
<tr>
<th>Variable</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>-0.197</td>
<td>-0.6899</td>
<td>0.0334</td>
<td>-0.1182</td>
<td>-0.3079</td>
<td>2.2373</td>
</tr>
<tr>
<td>V2</td>
<td>1.0673</td>
<td>-0.3703</td>
<td>-0.2099</td>
<td>0.0669</td>
<td>-0.7219</td>
<td>1.2672</td>
</tr>
<tr>
<td>V3</td>
<td>-0.9431</td>
<td>0.6524</td>
<td>0.3073</td>
<td>-0.5116</td>
<td>-0.4198</td>
<td>0.589</td>
</tr>
<tr>
<td>V4</td>
<td>-0.1191</td>
<td>0.2524</td>
<td>-0.2046</td>
<td>0.7894</td>
<td>-0.7028</td>
<td>-0.248</td>
</tr>
<tr>
<td>V5</td>
<td>-0.1844</td>
<td>-1.3029</td>
<td>-0.1268</td>
<td>-0.3979</td>
<td>-0.1971</td>
<td>0.3891</td>
</tr>
<tr>
<td>V6</td>
<td>0.1429</td>
<td>0.1303</td>
<td>1.0453</td>
<td>0.0665</td>
<td>0.0269</td>
<td>-0.0248</td>
</tr>
</tbody>
</table>

## Table 5.5. MAF transformation matrix from factors to variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>-0.3008</td>
<td>0.6181</td>
<td>-0.2065</td>
<td>-0.3272</td>
<td>-0.1722</td>
<td>0.1095</td>
</tr>
<tr>
<td>F2</td>
<td>-0.0746</td>
<td>0.1573</td>
<td>0.3148</td>
<td>-0.1385</td>
<td>-0.6585</td>
<td>-0.1656</td>
</tr>
<tr>
<td>F3</td>
<td>0.01</td>
<td>-0.0576</td>
<td>0.0475</td>
<td>0.025</td>
<td>0.1349</td>
<td>0.9521</td>
</tr>
<tr>
<td>F4</td>
<td>0.6098</td>
<td>-0.4955</td>
<td>-0.6853</td>
<td>0.7591</td>
<td>-0.3594</td>
<td>0.1875</td>
</tr>
<tr>
<td>F5</td>
<td>0.5289</td>
<td>-0.5869</td>
<td>-0.6201</td>
<td>-0.5346</td>
<td>-0.5396</td>
<td>-0.1226</td>
</tr>
<tr>
<td>F6</td>
<td>0.5023</td>
<td>-0.0032</td>
<td>-0.0434</td>
<td>-0.1054</td>
<td>-0.3135</td>
<td>-0.0626</td>
</tr>
</tbody>
</table>
Histograms and correlations of MAF factors are presented in Figure 5.4 and Table 5.6, respectively.

Figure 5.4. Histograms of MAF factors
Table 5.6. Correlation matrix of MAF factors

<table>
<thead>
<tr>
<th>Variable</th>
<th>MAF1</th>
<th>MAF2</th>
<th>MAF3</th>
<th>MAF4</th>
<th>MAF5</th>
<th>MAF6</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAF1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MAF2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MAF3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MAF4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MAF5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MAF6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

As can be observed from Table 5.6, the correlations between variables are completely eliminated after MAF transformation step. Also from the histograms of MAF factors in Figure 5.4, the inadequate Gaussianity of MAF factors suggests another Gaussian transformation step.

c. Secondary Gaussian Transformation

Another Gaussian transformation is implemented on the MAF factors. The histograms of the transformed variables (denoted as NS MAF) are presented in Figure 5.3. It is clear from these histograms that the Gaussianity of the MAF factors are restored. At this step, the correlations between variables are completely eliminated and the Gaussianity is secured, so the two prerequisites to implement a Gaussian-based simulation technique are met.
5.4.3 Variography of normal score MAF factors

In this stage, firstly variogram maps were built to confirm the existence of spatial anisotropy as presented in Figures 5.6, 5.8, 5.10, 5.12, 5.14, and 5.16. It is clear in these variogram maps that the continuity of NS MAF1, NS MAF2, NS MAF4, and NS MAF6 are directional, while that of NS MAF3 and NS MAF5 are omnidirectional.
Based on this analysis, experimental variograms were constructed accordingly in different directions as suggested by variogram maps. If the spatial continuity is directional, variograms were built in three directions: Major, semi-major, and vertical. If the spatial continuity is omnidirectional, only one variogram model was built.

Finally, linear model of co-regionalisation technique was used to fit semi-variogram models to the experimental variograms. The results for six NS MAF scores are presented in Figures 5.6 to 5.11.

**NS MAF1:**

![Variogram map (a) and fitting models (directional) of NS MAF1 (b)](image)

*Figure 5.6. Variogram map (a) and fitting models (directional) of NS MAF1 (b)*
NS MAF2:

Figure 5.7. Variogram map (a) and fitting models (directional) of NS MAF2 (b)
Figure 5.8. Variogram map (a) and fitting model (omnidirectional) of NS MAF3 (b)
NS MAF4:

Figure 5.9. Variogram map (a) and fitting models (directional) of NS MAF4 (b)
**NS MAF5:**

(a) 

(b)

**Figure 5.10.** Variogram map (a) and fitting model (omnidirectional) of NS MAF1 (b)
NS MAF6:

Figure 5.11. Variogram map (a) and fitting models (directional) of NS MAF6 (b)

A summary of best-fit variograms is presented in Table 5.7. Generally, the variograms were of average quality. Relative nugget effect was relatively low for directional variograms, while omnidirectional variograms of NS MAF3 and NS MAF5 had relatively high nugget as a normal feature of this type of variogram.
Table 5.7. Summary of best-fit semi-variogram models (Sph stands for spherical, Rel. stands for relative)

<table>
<thead>
<tr>
<th>Variables</th>
<th>Rel. Nugget</th>
<th>Structure 1</th>
<th>Structure 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Model type</td>
<td>Rel. Sill</td>
</tr>
<tr>
<td>NS MAF1</td>
<td>25%</td>
<td>Sph.</td>
<td>28%</td>
</tr>
<tr>
<td>NS MAF2</td>
<td>20%</td>
<td>Sph.</td>
<td>40%</td>
</tr>
<tr>
<td>NS MAF3</td>
<td>70%</td>
<td>Sph.</td>
<td>23%</td>
</tr>
<tr>
<td>NS MAF4</td>
<td>20%</td>
<td>Sph.</td>
<td>20%</td>
</tr>
<tr>
<td>NS MAF5</td>
<td>75%</td>
<td>Sph.</td>
<td>25%</td>
</tr>
<tr>
<td>NS MAF6</td>
<td>60%</td>
<td>Sph.</td>
<td>25%</td>
</tr>
</tbody>
</table>

5.4.4 Sequential Gaussian Simulation

Sequential Gaussian simulation technique was used to generate 20 realisations for normal score MAF factors using ISATIS software (Bleines & de Paris, 2000). The simulated values of normal score MAF factors were then back-transformed through three stages sequentially, namely secondary Gaussian transformation, MAF transformation, and primary Gaussian transformation. The final result with realisations No. 1, 5, 10, and 20 with Fe grade legend are shown in Figure 5.12. It is worth noting that while all realisations reflex the same trend of grade distribution (i.e. the global accuracy) of the deposit, the *in-situ* variability between realisations is well generated. This uncertainty feature of simulation will be quantified in the next section.

1st realisation
Figure 5.12. Samples of final simulation results
5.4.5 Validation of simulation results

Several techniques were used to evaluate the quality of the achieved 20 orebody realisations.

a. Correlation matrix validation

To assess the ability of the proposed simulation method in preserving the correlation of variables, the correlation of original data is compared with the first simulation image as presented in Table 5.8.

Table 5.8. Comparison of correlations between original data and realisation No.1

<table>
<thead>
<tr>
<th>Variable</th>
<th>Fe</th>
<th>SiO₂</th>
<th>Al₂O₃</th>
<th>P</th>
<th>LOI</th>
<th>GOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fe</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SiO₂</td>
<td>-0.72</td>
<td>-0.66</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Al₂O₃</td>
<td>-0.67</td>
<td>-0.83</td>
<td>0.50</td>
<td>0.68</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>P</td>
<td>0.19</td>
<td>-0.14</td>
<td>-0.23</td>
<td>-0.24</td>
<td>-0.14</td>
<td>0.02</td>
</tr>
<tr>
<td>LOI</td>
<td>-0.45</td>
<td>-0.69</td>
<td>0.21</td>
<td>0.4</td>
<td>0.39</td>
<td>0.71</td>
</tr>
<tr>
<td>GOL</td>
<td>0.00</td>
<td>0.44</td>
<td>-0.02</td>
<td>-0.32</td>
<td>-0.08</td>
<td>-0.5</td>
</tr>
</tbody>
</table>

In general, the conservation of cross-correlations between five attributes Fe, SiO₂, Al₂O₃, P, and LOI is on average to high quality. On the other hand, there are non-negligible deviations of correlations between GOL and five other variables in the realisation. This phenomenon can be explained due to the weak correlation of GOL with other variables. This result suggests that GOL grade should be simulated separately from the other five more strongly correlated variables.

b. Variogram reproduction

To validate whether spatial features of data are preserved during simulation process or not, the variograms were reproduced on simulated data and compared with fitting variogram models, as presented in Figure 5.13.
NS MAF1:

NS MAF2:

NS MAF3:

NS MAF4:
NS MAF5:

Figure 5.13. Reproduced variograms of six NS MAF factors

c. Histogram Reproduction

Variograms of Fe grade of the first realisation and composite data are shown in Figure 5.14. The clear similarity between two histograms proves the quality of the simulation work.

Figure 5.14. Reproduced histogram of Fe between 1st SGS realisation (left) and composite data (right)
d. Quantile-quantile (QQ) validation

QQ plot between Fe grade of the first realisation and composite data is presented in Figure 5.15. At Fe grade above the cut-off grade 50%, the correlation is very good proving the quality of simulation. At Fe grade below the cut-off grade of 50%, the weak correlation of simulated data and composite data can be explained as the impact of outliers and the lack of sample data in this range. However, as the ore material having above Fe cut-off grade is the most important to mine planning, this result is acceptable.

![Figure 5.15. QQ plot of SGS versus composite data](image)

5.5 GEOLOGICAL UNCERTAINTY ANALYSIS

Grade/tonnage curves of the 20 simulated orebody models were plotted together with the kriged resource model from Chapter 4 to analyse the geological uncertainty of the Silvergrass East iron ore deposit, as presented in Figure 5.16. Given that kriging is a standard interpolation technique for ore resource estimation in mining industry while simulation is one of the best tools to mimic reality of ore resource, noticeable geological uncertainty elements can be withdrawn as follows:

- The estimated Fe grade would be considerably less than that of the reality by approximately 1%;
• The Kriged resource model is likely to overestimate ore tonnage when Fe grade is lower than the mean of composite Fe (i.e. 59.23% from Table 5.1) and underestimate when Fe grade is higher. This is a clear evidence of the smoothing effect of ordinary kriging.

• The \textit{in situ} ore tonnage variability is in a magnitude of 3 million tonnes.

• Specifically, at a cut-off grade of 57.5\% Fe, the geological risk is that the actual ore tonnage would be 10-13 million tonnes less than expectation and the actual average Fe grade of ore tonnage is approximately 1.2\% higher.

• By analysing the similarity between orebody realisations, it can be concluded that the number of 20 realisations are sufficient to characterise geological uncertainty features of the given deposit.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_16.png}
\caption{Grade/tonnage curves kriging vs 20 SGS realisations (after (Mai et al., 2016b))}
\end{figure}

A visual comparison of the kriged and simulated resource models (realisation No. 1) is provided in Figure 5.17. It is clear that while the simulation honours the global accuracy of the deposit, it is free of the smoothness as well displayed in the kriged
model. The same phenomenon also appears on other realisations as shown previously in the Figure 5.12.

Figure 5.17. Comparison of ordinary kriging (a) and simulation (realisation No.1) (b) resource models
5.6 CONCLUSIONS

In this chapter, the most common simulation options for multivariate datasets were analysed and compared with the typical characteristics of iron ore deposits. Consequently, minimum/maximum autocorrelation factors (MAF) and sequential Gaussian simulation (SGS) was selected based on the amount of available sampling data, the number of attributes of interest, and the superiority in de-correlating variables in all lag distances of MAF. A simulation framework was suggested using MAF and SGS consisting of six steps and was tested on the Silvergrass East iron ore deposit in Pilbara iron ore region, Western Australia to generate 20 orebody realisations.

Among the six variables of interest being simulated, namely Fe, SiO₂, Al₂O₃, P, LOI, and GOL, the proposed simulation framework was able to adequately reserve the correlations of the first five variables. Regarding GOL, due to the weak correlation of this variable to the other five, the simulation result suggests that it can be simulated separately. Apart from correlation, the orebody realisations are at high quality with accurate reproduction of statistical and spatial continuity properties of the deposit.

By comparing orebody realisations and ordinary kriging (OK) estimation, the geological uncertainty properties of the Silvergrass East iron ore deposit were fully characterised. Specifically, the Fe grade of the estimation model is considerably less than that of the reality by approximately 1%. The tonnage variability of iron ore was also analysed to point out the smoothing effect of kriging method.

This information of the uncertainty of the deposit is critical to the mine planning process as it may directly impact the feasibility of a mining project. In the next chapter, a new stochastic mine planning method using mathematical programming and TopCone Algorithm will be discussed to incorporate this information into the optimisation process.
6.1 INTRODUCTION

As discussed in Sections 2.4 and 5.5, geological uncertainty can pose a considerably detrimental impact on the feasibility of a mine planning solution and hence the success of mining project. Taking that risk into mine planning optimisation process, therefore, is critical and has become a dynamic and fast-growing research area in recent decades. Generally referred to as stochastic mine planning, one of the pioneering discussions is by Dimitrakopoulos et al. (2002). In this work, the authors argue that if a mine planning process is implemented based on a deterministic set of input data, i.e. geology and finance, the result is vulnerable to the risk caused by the inherent uncertainty of data and thus can be far from the actual answer. In regards to dealing with geological uncertainty, conditional simulation techniques, such as sequential Gaussian simulation (Journel, 1994), is implemented to generate a series of equally probable possibilities of the deposit instead of a single estimated model, which is commonly interpolated by kriging techniques (David, 1977; Goovaerts, 1997). These orebody realisations provide a critical platform for stochastic mine planning optimisation as a source of grade and tonnage variability.

In early attempts, orebody realisations are sequentially fed into traditional scheduling model to evaluate the risk of in-situ grade and tonnage variability on project’s outcome. Dowd (1994) analyses the impact of uncertainty on ultimate pit limits by repeatedly implementing Lerchs-Grossman (LG) algorithm using a combination of twelve financial and six geological scenarios. The author then concludes a considerable variation between ultimate pit results. Specifically, the differences in total tonnage, average grade, and undiscounted profit can reach up to 15, 10, and 19% respectively. Smith and Dimitrakopoulos (1999) analyse the impact of deposit uncertainty on short-term production scheduling by sequentially applying a Mixed Integer Programming (MIP) model on a series of eight orebody realisations of a uranium deposit. The outcome shows a strong variability of both grade and tonnage in each scheduling period and the authors propose to use the probability of
occurring of each block in a given period as a guide in setting a risk-based schedule. Another attempt in sequentially feeding orebody realisations into mine scheduling is also conducted by Ravenscroft (1992). Groeneveld and Topal (2011) repeatedly implement a MIP model to optimise available mine design options using a combination of numerous independent scenarios of price, recovery, cost, and utilisation, which are generated by Monte Carlo Simulation, with different options of mining, stockpiling, processing, and port capacities. Consequently, by analysing the frequency of execution of those pre-defined options, the most beneficial mine design is suggested.

A forward step in this research area is made when the independent scheduling outcomes generated by using different orebody realisations are post-processed to construct a final but not optimal solution. Godoy and Dimitrakopoulos (2004) use simulated annealing technique to derive a single mining sequence from separate production schedules, which are generated using Whittle Milawa algorithm (Whittle, 2016). Firstly, the algorithm freezes those blocks with very high probability of occurring between schedules as an initial mining sequence. This sequence is then perturbed in a simulated annealing process aiming at lowering the chance of violating production targets as well as slope safety. The case study in a gold mine shows that the proposed method is able to achieve a long-term mine plan with less than 3.5% chance of deviating from production targets and an increase of 28% NPV compared to a base-case schedule. Leite and Dimitrakopoulos (2007) apply the similar concept of using conditional simulation and simulated annealing on a low-grade copper deposit and obtain a 26% NPV improvement over the traditional approach.

Recent work in this area has focused on fully integrating geological uncertainty into open pit mine production scheduling optimisation process using stochastic integer programming (SIP). This is an extension of integer programming (IP) with a capability of dealing with stochastic data. Ramazan and Dimitrakopoulos (2007) define “SIP is a type of mathematical programming and modelling that considers multiple equally probably scenarios and generates the optimal result for a set of defined objectives within the feasible solution space bounded by a set of constraints”. In their work, the authors propose a two-stage SIP formulation that seeks to maximise the expected
project’s NPV while minimising the risk of not meeting production targets due to geological uncertainty. The authors also propose the idea of “stochastic constraints” which are feasible for any solution but any violation of lower or upper bounds is measured and minimised by the objective function. The SIP-based production scheduling model is formulated based on all orebody realisations, and hence the solving the SIP model means all orebody realisations are considered simultaneously during the optimisation process. This novel SIP model has been tested and improved over time (Benndorf & Dimitrakopoulos, 2013; Consuegra & Dimitrakopoulos, 2009; Dimitrakopoulos & Jewbali, 2013; Goodfellow, 2014; Lamghari & Dimitrakopoulos, 2012; Ramazan & Dimitrakopoulos, 2013). However, similar to other mathematical programming techniques which resort integer variables, the intensity of mine planning problems still remains a major challenge to the application of SIP. For instance, the iron ore deposit used in the SIP model of Benndorf and Dimitrakopoulos (2013) has only 3,049 blocks. The gold deposit used in SIP model of Ramazan and Dimitrakopoulos (2013) consists of 22,296 blocks, but integer variables are only assigned for ore blocks.

This chapter presents a new SIP mathematical model that takes TopCones (TCs) with their stochastic values as input to generate optimum production schedule for open pit mines. The SIP model has the objective function of maximising project’s NPV while minimising the possibility of violating scheduling targets. Before formulating SIP model, blocks are aggregated to reduce the data scale. The optimisation process can be depicted in Figure 6.1:

![Figure 6.1](image)

**Figure 6.1.** Schematic presentation of the optimisation process developed for SIP model
The process has four stages:

**Stage 1:** A number of 20 orebody realisations are generated to provide geological uncertainty data for the stochastic mine planning model. The simulation procedure was discussed in Chapter 5.

**Stage 2:** The realisations are then averaged to find an E-type model of the orebody.

**Stage 3:** TopCone Algorithm (TCA) is implemented on the E-type model to cluster blocks into TCs.

**Stage 4:** Simulated values of TCs are calculated by sequentially replacing 20 orebody realisations into TCs. These values are then fed into SIP model, whose formulation is discussed in the next section, to optimise a long-term production schedule of the mine plan.

The Visual Basic .Net code of the program calculating simulated values of TCs is presented in Appendix B, while the code of the SIP model is presented in Appendix C.

### 6.2 MATHEMATICAL FORMULATION OF STOCHASTIC INTEGER PROGRAMMING

The SIP model developed here in accounts for uncertainty inputs by considering simulated TCs values of ore tonnage, total tonnage, and grades. The model utilises the idea of stochastic constraints, or soft stochastic constraints, which are initially proposed by Ramazan and Dimitrakopoulos (2007), which allows feasible solutions to violate lower and upper bounds of scheduling constraints but come with certain costs, which are being minimised in the objective function.

**Notation**

**Indices and sets**

- \( i \in N \) set of TCs \( i \)

- \( j \in M_i \subset N \) subset of TCs \( j \) that are predecessor TCs for TC \( i \)

- \( t, t' \in P \) set of time periods \( t, t' \) in the horizon

- \( k \in K \) set of \( K \) production targets (\( k = 1: \) Fe, \( k = 2: \) SiO\(_2\), \( k = 3: \) Al\(_2\)O\(_3\), \( k = 4: \) P, \( k = 5: \) LOI, \( k = 6: \) GOL)
\( s \in S \) set of simulations

**Parameters**

d \hspace{1cm} economic discount rate, \%  
\( \bar{V}_i \) representation of averaged economic value of ST \( i \) derived from S realisation, \$

\( G_{si}^k \) grade \( k \) of ST \( i \) on simulation \( s \), \%

\( \bar{G}_{i}^k \) representation of averaged grade \( k \) of ST \( i \) derived from S realisations, \%,

\( O_{si} \) ore tonnage of ST \( i \) on simulation \( s \), tonne

\( \bar{O}_{i} \) representation of averaged ore tonnage of ST \( i \) derived from S realisations, tonne

\( W_{si} \) waste tonnage of ST \( i \) on simulation \( s \), tonne

\( G_{min}^k / G_{max}^k \) minimum/maximum blending grade, \%

\( PC_{min} / PC_{max} \) minimum/maximum processing plant capacity, tonne

\( MC_{min} / MC_{max} \) minimum/maximum capacity of the mine’s equipment, tonne

\( Cou \) cost unit associated with the excess amount of ore, \$/tonne

\( Col \) cost unit associated with the deficient amount of ore, \$/tonne

\( Cmu \) cost unit associated with the excess amount of total mining tonne, \$/tonne

\( Cml \) cost unit associated with the deficient amount of total mining tonne, \$/tonne

\( Cgul \) cost unit associated with the excess amount of grade \( k \),

\( Cgl^k \) cost unit associated with the deficient amount of grade \( k \), \$/%tonne

**Linear variables**

\( dou_t \) excess amount of ore in simulation \( s \) in period \( t \), tonne
\[ dol_t \] deficient amount of ore in simulation \( s \) in period \( t \), tonne

\[ dml_t \] excess amount of total tonnage in simulation \( s \) in period \( t \), tonne

\[ dml_t^t \] deficient amount of total tonnage in simulation \( s \) in period \( t \), tonne

\[ dgk_t \] excess amount of grade \( k \) in simulation \( s \) in period \( t \), % tonne

\[ dgk_t^t \] deficient amount of grade \( k \) in simulation \( s \) in period \( t \), % tonne,

**Binary variables**

\[ x_t^i \] equal to 1 if \( ST \) \( i \) is scheduled in period \( t \); 0 otherwise.

**Objective function**

\[
\max \sum_{t \in P} \left[ \frac{1}{(1 + \frac{d}{100})^t} \left( \sum_{s \in S} V_s^t x_t^i - \sum_{x \in S} \left( \frac{Cou.dou_x^t + Col.dol_s^t + Cmu.dmu_s^t}{\text{Part 1}} + \frac{Cml.dml_s^t + Cgu^k.dgu_s^t + Cgl^k.dgl_s^t}{\text{Part 2}} \right) \right) \right] \quad (6.1)
\]

**Subject to**

**Processing capacity constraints:**

\[
\sum_{i \in N} O_{si} x_t^i + dol_s^t \geq PC_{\min} \quad \forall s \in S, \forall t \in P \quad \text{Lower bound} \quad (6.2)
\]

\[
\sum_{i \in N} O_{si} x_t^i - dou_s^t \leq PC_{\max} \quad \forall s \in S, \forall t \in P \quad \text{Upper bound} \quad (6.3)
\]

**Mining capacity constraints**

\[
\sum_{i \in N} (O_{si} + W_{si}) x_t^i + dml_s^t \geq MC_{\min} \quad \forall s \in S, \forall t \in P \quad \text{Lower bound} \quad (6.4)
\]

\[
\sum_{i \in N} (O_{si} + W_{si}) x_t^i - dmu_s^t \leq MC_{\max} \quad \forall s \in S, \forall t \in P \quad \text{Upper bound} \quad (6.5)
\]
**Blending grade constraints**

\[ \sum_{i \in N} \left( G_{s_i}^k - G_{\min}^k \right) O_{s_i} x_i^t + d g_{s_i}^t \geq 0 \quad \forall k \in K, \forall s \in S, \forall t \in P \]  
Lower bound (6.6)

\[ \sum_{i \in N} \left( G_{s_i}^k - G_{\max}^k \right) O_{s_i} x_i^t - d u_{s_i}^t \leq 0 \quad \forall k \in K, \forall s \in S, \forall t \in P \]  
Upper bound (6.7)

**Sequencing constraints**

\[ M_i x_i^t - \sum_{j \in M_i} \sum_{t' \in t} X_{j}^{t'} \leq 0 \quad \forall i \in N, \forall t \in P \]  
(6.8)

**Linear variable constraints**

\[ d o u_i^t, d o l_i^t, d m u_i^t, d m l_i^t, d g u_s^k, d g l_s^t \geq 0 \quad \forall k \in K, \forall t \in P, \forall s \in S \]  
(6.9)

**Integer variable constraints**

\[ x_i^t \in \{0,1\} \text{ and } x_j^{t'} \in \{0,1\} \quad \forall i, j \in N, \forall t, t' \in P \]  
(6.10)

The objective function (6.1) of the SIP model is a resource function consisting of two parts. Part 1 maximises the expected values from mining TopCones. Part 2 minimises the uncertainty costs caused by the violation of scheduling targets due to grade and tonnage variability. The uncertainty costs are constructed as linear functions of the amount of violation of each simulation of each type of scheduling targets in a certain period, multiplied by the corresponding cost units. For instance, \( d o u_i^t \) measures the amount of excess ore tonnage (i.e. higher than maximum processing capacity) of simulation \( s \) in period \( t \). \( d o u_i^t \) is associated with \( C o u \) which is the cost for each tonnage of \( d o u_i^t \) violated. This mechanism to calculate uncertainty costs is applied for the amount of excess/deficient amount of ore tonnage, total tonnage and grade in each period of all simulations.

Constraints (6.2) to (6.7) are soft constraints to govern feasible scheduling solutions for each simulation in each time period. These type of constraints allow feasible
solutions to have the outcome mining capacity, processing capacity and blending grade deviate from the predefined lower and upper bounds as recorded by linear variables \(dou_s^l, dol_l, dml_s^l, dgl_s^l, dgl_s^{kl}\), respectively. These deviations are controlled by a minimisation function in Part 2 of the objective function. Constraints (6.8) ensure mining sequence between TCs. Constraints (6.9) and (6.10) define linear and integer variables.

**Sensitivity of cost units to stochastic mine planning results**

The deployment of stochastic constraints makes the SIP models very flexible, so there is always feasible to obtain a solution for SIP models, as any value of decision variables are feasible under this type of constraints. Therefore, a major downside of this formulation is that the quality of SIP solution is highly sensitive to the way the cost units are being determined. This behaviour of SIP models has been pointed out in the work of Ramazan and Dimitrakopoulos (2007) and Benndorf and Dimitrakopoulos (2013).

Although it is accepted that the specific values of cost units depend on the tolerance and strategy towards risk handling of a specific mine, there is a fundamental rule to determine those cost units. That is the cost of violating operational constraints must be greater than the benefit gained by doing so, or else the recourse objective function will return extreme violations of operational constraints and, consequently, impractical mine planning solutions.

In iron ore mining, it can be easily argued that maximum processing capacity is the most important operational constraint due to its dominant influence on economic calculation. Therefore, the cost unit of excess ore tonnage (i.e. \(Cou\), is the most sensitive parameter. If \(Cou\) is too small, the SIP model will arrange most ore tonnage in early periods as the effect of discounted dollar value in NPV calculation. This can be explained as the cost of violating maximum processing capacity being smaller than the economic value gained.

On the one hand, the lower bound of \(Cou\) thus needs to satisfy a threshold to ensure that the cost of excess one ore tonne is greater than the economic value of that one ore tonne, or mathematically expressed as follows:
\[
\sum_{s \in S} \text{Cou} \geq Eo
\]
\[
\iff \text{Cou} \geq \frac{Eo}{S}
\]

(6.11)

where \( Eo \) is the net profit of mining one ore tonne.

On the other hand, the upper bound of \( \text{Cou} \) relies heavily on the strategy of mining companies. Large-scale mining companies may prefer a high-risk-high-return strategy so upper bound \( \text{Cou} \) might be close to the lower bound. In contrast, small companies with only one or a few mining projects may prefer a highly confident mine plan where the risk of not meeting ore production target is minimal, despite less NPV expectation.

Other cost unit parameters can be determined similarly using this concept.

6.3 COMPUTATIONAL EXPERIMENTS

The stochastic mine planning method using TopCone Algorithm and stochastic integer programming proposed in this chapter was applied on the Silvergrass East mining project. From 20 orebody realisations generated in Chapter 5 using minimum/maximum autocorrelation factors (MAF) and sequential Gaussian simulation (SGS), an E-type orebody model was constructed by averaging all simulated attributes, namely tonnage, grades of Fe, SiO₂, Al₂O₃, P, LOI, and GOL, and block economic value.

TCA was then implemented on this E-type orebody model, so a set of 533 TCs were generated with minimum cone size of 500. A summary of this process together with the information of the scheduling problems before and after implementing TCA is provided in Table 6.1.
Table 6.1. Summary of TCA implementing TCA and scheduling parameters

<table>
<thead>
<tr>
<th>Block model</th>
<th>E-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Cone Size</td>
<td>500</td>
</tr>
<tr>
<td>Solution time of TCA (mins)</td>
<td>11</td>
</tr>
<tr>
<td>Number of scheduling periods</td>
<td>8</td>
</tr>
</tbody>
</table>

Before implementing TCA

| Number of blocks             | 985,088 |

After implementing TCA

| Number of TCs                | 533     |
| Number of blocks within TCA pit | 63,069 |
| Number of binary variables   | 4264    |
| Number of linear variables   | 2560    |
| Number of precedences        | 641     |
| Ore tonnage within TCA pit (Mt) | 232.2  |
| Waste tonnage within TCA pit (Mt) | 203.9  |
| TCA ultimate pit value (M$)   | 2,671.8 |

From Table 6.1, the role of TCA in reducing the data scale of the scheduling problem is demonstrated. From 985,088 blocks of the E-type block model, only 533 TCs were generated. If compared with the number of blocks within the TCA ultimate pit, the conversion ratio from blocks to TCs is nearly 120 times. Consequently, the number of variables and precedences of the scheduling model are relatively small.

After calculating stochastic values of TCs, they were fed into the SIP model using the same set of scheduling parameters as the deterministic mine planning experiments as shown in Table 4.2, Chapter 4. To investigate the impact of cost units of excess ore tonnage to the feasibility of scheduling solution, different Cou options were fed into the SIP model, while other cost units had a constant value of 1. Note that, given the total number of orebody realisations of 20 and a net profit of mining one ore tonne is 15$, the minimum Cou parameter as calculated by Equation (6.11) is 0.75 $/tonne. Table 6.2 presents information on the formulation and solving SIP models.
with various $Cou$ options, while Table 6.3 provides details of the schedule of each $Cou$ option.

**Table 6.2.** Information of the formulation and solving SIP models using different $Cou$ options

<table>
<thead>
<tr>
<th>$Cou$</th>
<th>0.0</th>
<th>0.2</th>
<th>0.75</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopCones</td>
<td>533.0</td>
<td>533.0</td>
<td>533.0</td>
<td>533.0</td>
</tr>
<tr>
<td>binary variables</td>
<td>4264</td>
<td>4264</td>
<td>4264</td>
<td>4264</td>
</tr>
<tr>
<td>linear variables</td>
<td>2560</td>
<td>2560</td>
<td>2560</td>
<td>2560</td>
</tr>
<tr>
<td>precedences</td>
<td>641</td>
<td>641</td>
<td>641</td>
<td>641</td>
</tr>
<tr>
<td>Solution time of SIP (mins)</td>
<td>98</td>
<td>83</td>
<td>95</td>
<td>165</td>
</tr>
</tbody>
</table>

As shown in Table 6.2, the size of SIP models are relatively small, so high-quality solutions with optimality gap of less than 5% were achieved fairly quickly using CPLEX.

Table 6.3 clearly demonstrates the importance of setting cost units appropriately, particularly in this case is $Cou$. To be more specific:

- At $Cou$ of 0.0, or there is no penalty applied for the violation of schedule over the maximum processing capacity, a very high amount of ore has been arranged to period 1. The NPV of schedule $Cou$ of 0.0 is the highest among all scenarios, but it cannot be achieved through actual operation due to the high deviation.

- At $Cou$ of 0.2, strong violation of ore production can still be observed, but in a less magnitude than the scenario $Cou$ of 0.0.

- At $Cou$ of 0.75 which is the threshold as calculated by equation (6.11), the violation of maximum ore production still exists but at a mild level.

- At $Cou$ of 1, there is almost no deviation from the upper bound limit of ore production. However, NPV of this scenario is also the smallest.
<table>
<thead>
<tr>
<th></th>
<th>Cou</th>
<th>Period</th>
<th>Fe (%)</th>
<th>Ore tonnage (Mt)</th>
<th>Waste tonnage (Mt)</th>
<th>Total tonnage (Mt)</th>
<th>Undiscounted value (M$)</th>
<th>NPV (M$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>59.624</td>
<td>57.8</td>
<td>10.4</td>
<td>68.1</td>
<td>835.1</td>
<td>759.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>60.076</td>
<td>26.5</td>
<td>19.7</td>
<td>46.2</td>
<td>338.9</td>
<td>280.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>60.199</td>
<td>22.3</td>
<td>16.8</td>
<td>39.1</td>
<td>284.4</td>
<td>213.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>60.419</td>
<td>25.4</td>
<td>25.8</td>
<td>51.2</td>
<td>303.4</td>
<td>207.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>60.313</td>
<td>22.8</td>
<td>20.9</td>
<td>43.7</td>
<td>279.2</td>
<td>173.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>60.38</td>
<td>22.1</td>
<td>33.7</td>
<td>55.8</td>
<td>230.3</td>
<td>130.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>59.662</td>
<td>22.1</td>
<td>42.8</td>
<td>64.9</td>
<td>203.0</td>
<td>104.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>59.675</td>
<td>22.2</td>
<td>44.9</td>
<td>67.0</td>
<td>197.6</td>
<td>92.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sum/Mean</td>
<td>59.983</td>
<td>221.2</td>
<td>215.0</td>
<td>436.2</td>
<td>2,671.8</td>
<td>1,959.8</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>59.749</td>
<td>53.1</td>
<td>5.0</td>
<td>58.1</td>
<td>781.3</td>
<td>710.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>59.78</td>
<td>27.8</td>
<td>18.8</td>
<td>46.6</td>
<td>360.1</td>
<td>297.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>60.06</td>
<td>27.9</td>
<td>29.7</td>
<td>57.6</td>
<td>328.6</td>
<td>246.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>60.002</td>
<td>22.9</td>
<td>15.4</td>
<td>38.2</td>
<td>297.0</td>
<td>202.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>60.298</td>
<td>22.6</td>
<td>25.8</td>
<td>48.4</td>
<td>261.3</td>
<td>162.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>60.076</td>
<td>22.5</td>
<td>28.5</td>
<td>51.0</td>
<td>252.0</td>
<td>142.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60.081</td>
<td>22.3</td>
<td>43.5</td>
<td>65.8</td>
<td>203.3</td>
<td>104.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>60.17</td>
<td>22.2</td>
<td>48.3</td>
<td>70.5</td>
<td>188.2</td>
<td>87.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sum/Mean</td>
<td>59.983</td>
<td>221.2</td>
<td>215.0</td>
<td>436.2</td>
<td>2,671.8</td>
<td>1,954.2</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>1</td>
<td>59.412</td>
<td>28.1</td>
<td>2.0</td>
<td>30.1</td>
<td>415.9</td>
<td>378.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>59.689</td>
<td>27.8</td>
<td>4.5</td>
<td>32.4</td>
<td>403.9</td>
<td>333.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>60.321</td>
<td>28.1</td>
<td>21.4</td>
<td>49.6</td>
<td>357.7</td>
<td>268.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>60.386</td>
<td>28.5</td>
<td>28.5</td>
<td>57.0</td>
<td>342.5</td>
<td>233.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>60.227</td>
<td>28.2</td>
<td>17.5</td>
<td>45.7</td>
<td>370.1</td>
<td>229.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>60.121</td>
<td>30.6</td>
<td>39.0</td>
<td>69.6</td>
<td>341.4</td>
<td>192.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>59.8</td>
<td>27.7</td>
<td>47.3</td>
<td>75.0</td>
<td>273.0</td>
<td>140.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>59.925</td>
<td>21.8</td>
<td>53.5</td>
<td>75.3</td>
<td>166.2</td>
<td>77.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sum/Mean</td>
<td>59.99</td>
<td>220.9</td>
<td>213.8</td>
<td>434.7</td>
<td>2,670.7</td>
<td>1,854.7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>59.578</td>
<td>27.8</td>
<td>2.2</td>
<td>30.0</td>
<td>410.8</td>
<td>373.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>59.795</td>
<td>27.8</td>
<td>3.7</td>
<td>31.5</td>
<td>406.7</td>
<td>336.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>59.629</td>
<td>22.9</td>
<td>10.0</td>
<td>32.9</td>
<td>313.5</td>
<td>235.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>59.948</td>
<td>27.5</td>
<td>26.0</td>
<td>53.5</td>
<td>334.6</td>
<td>228.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>60.104</td>
<td>23.3</td>
<td>19.7</td>
<td>43.0</td>
<td>290.3</td>
<td>180.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>60.42</td>
<td>27.7</td>
<td>36.3</td>
<td>64.0</td>
<td>306.5</td>
<td>173.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60.207</td>
<td>27.2</td>
<td>43.8</td>
<td>71.0</td>
<td>277.0</td>
<td>142.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>60.181</td>
<td>25.5</td>
<td>43.5</td>
<td>69.0</td>
<td>251.1</td>
<td>117.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sum/Mean</td>
<td>59.985</td>
<td>209.8</td>
<td>185.1</td>
<td>394.9</td>
<td>2,590.5</td>
<td>1,786.2</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6.3.** Sensitivity of excess ore cost unit **Cou** to SIP solution
Further investigation on schedules $Cou$ of 0.75 and $Cou$ of 1 are shown in Figures 6.2 and 6.3, which respectively present the risk profiles of ore production and NPV through the life of mine.

**Figure 6.2.** Risk profiles - Ore tonnage of $Cou$ = 0.75 (a) and $Cou$ = 1 (b)

In Figure 6.2, there is virtually no violations of ore production of schedule $Cou$ of 1.0 except one possibility at time period 1. In other words, the ore production suggested by this schedule is highly certain to be within the predefined lower and upper bound constraints. In contrast, the risk profile of the schedule $Cou$ of 0.75 suggests that ore production can deviate from upper bound in time periods 1, 3, 4, 5, and particularly 6. This hints the mine planner who wants to apply this scenario to be well prepared for a situation of over ore production, particularly in time period 6, of which the ore production could be approximately 3 million tonnes higher than expectation.

Regarding economic outcome, Figure 6.3 provides details on comparing NPV between the two schedules $Cou$ of 0.75 and $Cou$ of 1. In Figure 6.3a, the NPV calculation discards the uncertainty costs, the terms to address the potential violations of scheduling targets. NPV of schedule $Cou$ of 0.75 has 68 million dollars higher than the schedule $Cou$ of 1. When uncertainty costs are taken into account when calculating NPV as in Figure 6.3b, the advantage in NPV of schedule $Cou$ of 0.75 is only 4 million dollars higher than the schedule $Cou$ of 1.
Figure 6.3. Compare NPV Risk profiles of these two Cou options

(a) Without uncertainty costs; (b) With uncertainty costs

The plan views and typical cross-sections of schedule Cou of 0.75 and Cou of 1 are displayed in Figures 6.4 to 6.7, respectively.
Figure 6.4. Plan view of SIP model *Cou* 0.75

Figure 6.5. Typical cross-sections of SIP mine plan with *Cou* of 0.75
Figure 6.6. Plan view of SIP model Cou 1

Figure 6.7. Typical cross-sections of SIP mine plan with Cou of 1
6.4 CONCLUSIONS

This chapter presented a new stochastic mine planning method using TopCone Algorithm (TCA) and stochastic integer programming (SIP). One important contribution of the proposed method is the deployment of TCA to aggregate blocks into TopCones (TCs) to control the scale of the SIP model. The objective function consists of two stages, where the first stage maximises the expected project’s NPV and the recourse stage minimises the deviations from expected targets. The stochastic constraints are adapted to cope with both lower and upper bound constraints. In addition, these stochastic constraints are particularly suitable to remediate the oversize problem of TCs. Specifically, if the minimum cone size (MCS) parameter is set unreasonably high, which leads to a very small number of TCs, the SIP model with stochastic constraints is still able to yield solution with high penalty on upper bound mining and/or processing constraints, and this is an indication that MCS should be set lower.

In addition, the sensitivity of SIP model to the magnitude of cost units was investigated. An algorithmic approach to determining thresholds for cost unit parameters was proposed, where the fundamental concept is that the benefit gained from breaking stochastic constraints must be less than the penalty by doing so. This also provides a useful tool to the mine planners to quantify the suitable level of risk that they are willing to incorporate into the mine plan. Generally, large companies with a sound financial background and highly flexible mining operations may prefer a high-risk level to seek higher potential profit, and vice versa for small companies.

It is noted that the stochastic values of TCs for each realisation are calculated indirectly via an E-type model, therefore, this is an approximation. Otherwise, it is not possible to calculate these values directly from realisations as implementing TCA on each realisation will produce unique sets of TCs and hence impossible to formulate a single SIP model.
CHAPTER 7. COMPARATIVE ANALYSIS AND DISCUSSION

7.1 INTRODUCTION

In this chapter, various comparative analyses of those mine planning methods discussed previously in this thesis are presented.

Firstly, the deterministic mine planning method using TopCone Algorithm (TCA) and integer programming (IP) is compared with the stochastic mine planning method using TCA and stochastic integer programming (SIP) regarding their capabilities in handling the risk caused by geological uncertainty. The idea is that as orebody realisations are the most reliable tool available to mimic and reproduce geological uncertainty, the schedules proposed by the two mine planning methods will be applied sequentially on these 20 orebody realisations to be validated against geological uncertainty. Each run is recorded as a possibility of implementing these mine plans in reality. Hence, the whole set of 20 possibilities plays a critical role as risk profiles of these two mine plans. In the risk profiles, the closer the simulated values to the expected values, the lower the risk of not meeting scheduling targets and vice versa. In this study, risk profiles are constructed for ore production, blending grades of Fe, SIO₂, Al₂O₃, P, loss on ignition (LOI), and Ochreous Goethite (GOL), as well as net present value (NPV).

Secondly, as one of the main purposes of aggregating blocks using TCA is to reduce the intensity of data fed into mathematical-based scheduling model to facilitate feasible solution being generated in a practical timeframe, the correlation between the number of TopCones (TCs) and the corresponding solution time will be analysed.

Finally, attempts are also made to compare the difficulties of the application of the two mine planning methods discussed in this thesis in terms of methodology and implementation.

7.2 RISK PROFILES OF SCHEDULING SOLUTIONS

The risk profiles of ore production, blending grades of Fe, SIO₂, Al₂O₃, P, LOI, and GOL, of the two mine planning methods using TCA/IP and TCA/SIP are presented in Figure 7.1.
(c) Risk profile - SiO2 grade SIP

(d) Risk profile - Al2O3 grade SIP
Figure 7.1. Comparison of risk profiles between TCA/IP and TCA/SIP methods
(a - ore production; b - Fe grade; c - SiO₂ grade; d - Al₂O₃ grade; e - P grade; f - LOI grade; and g - GOL grade)

The significant role of the risk profiles presented in Figure 7.1 is to quantify

It is clear from Figure 7.1 that TCA/SIP method outperforms TCA/IP method in the capability of dealing with geological uncertainty. Most noticeably, all risk profiles of TCA/IP pointed out that the real-time implementation of this schedule will return results considerably far from expectations. For example, the ore production in all eight time periods is approximately 1.5 million tonnes less than expected figures. Meanwhile, Fe blending grade is considerably higher than planned values and SiO₂ and Al₂O₃ blending grades are lower, which are reasonable due to the negative correlations between Fe and SiO₂, Al₂O₃.

The detrimental impacts of geological uncertainty on TCA/IP schedule also include the risk of violating production targets. Notably, the risk of breaching upper bound GOL grade can be observed at time periods 3, 4, and 8. A noticeable deviation of Fe grade can also be observed in time period 6.

In contrast, TCA/SIP schedule well proved its capability in incorporating geological uncertainty in its optimisation process. The expected production targets are well contained within the simulated values. In addition, only few possibility of violating
scheduling targets are observed, including one possibility of overproduction of ore tonnage in time period 1 and two possibilities of exceeding Fe maximum blending grade in time period 6.

By comparing the NPV between TCA/IP and TCA/SIP methods without considering uncertainty costs, as shown in Figure 7.2, the TCA/IP mine plan produced 3.54% higher NPV, equal to 108 million dollars.

![NPV Comparison - IP and SIP](image)

**Figure 7.2.** Compare NPV of TCA/IP and TCA/SIP without uncertainty costs

However, when analysing the risk profile of NPV, as shown in Figure 7.3, TCS/IP shows a very slim chance to obtain its expected NPV score. Specifically, the simulated NPV scores are from 7.01% to 7.81% lower than anticipated, equal to 124 M$ to 137 M$.

In contrast, the risk profile of NPV of TCA/SIP mine plan, as presented in Figure 7.4, shows a consistency between expected and simulated data, where it is very likely that the mine plan will achieve the expected NPV target.
Figure 7.3. Risk profile of NPV of TCA/IP mine planning method

Figure 7.4. Risk profile of NPV of TCA/SIP mine planning method
7.3 SOLUTION TIME REDUCTION

This section investigates the correlation of the number of TCs and IP solution time. Apart from eight MCS options being experimented as shown in Table 4.4, other experiments have been implemented in an attempt to generate more data to effectively characterise the correlation of solution time of solving IP-based scheduling models, and the number of TCs fed in. The result is presented in Figure 7.5.

![Correlation between number of TopCones and IP solution time](image)

**Figure 7.5.** Correlation of number of TopCones and solution time of IP-based scheduling models

It is apparent from Figure 7.5 that an exponential trend best describes the correlation between solution time and the number of TCs. This proves the significance of the ability of TCA in controlling the intensity of scheduling models. Although the solution time is affected by three other factors, namely the power of computing facility, the margin and the number of operational constraints, the first one is financially expensive to improve while the second and the third are very difficult or even impossible to adjust in many instances. To this perspective, TCA is an effective and convenient approach in regards to tackle solution time issue.
7.4 THE COMPARISON OF IMPLEMENTING DETERMINISTIC AND STOCHASTIC MINE PLANNING APPROACHES

Deterministic mine planning approach can be applied fairly straightforward. Starting with a kriged resource model, whose methodology and framework have been well established and tested widely in the mining industry, the IP mathematical model can be formulated and solved relatively simply using a commercial solver such as CPLEX (CPLEX, 2009), given the issue of the intensity of data has been overcome, such as using a block aggregation algorithm like TCA.

In contrast, although the advantages of stochastic mine planning have been well addressed in recent studies and this thesis, a proper application of this approach is rather challenging and sophisticated. First of all, high-quality realisations of the orebody need to be available as a source of geological uncertainty. Doing such task is much more complicated than a standard kriging estimation in all factors, namely methodology, data analysis, computation, and the multivariate nature of dataset as discussed previously in this thesis. Second of all, the methodology of stochastic mine planning has only been established in the last few decades, and there is still a scarcity of effectiveness-proven models and case studies. Finally, considering that kriging-based mine planning approach has been considered as a standard for the mining industry for many decades, a switch into the stochastic approach, which relies on multiple simulated orebodies rather than a single estimated one, undoubtedly needs time to ease the scepticism of stakeholders.

7.5 CONCLUSIONS

This chapter provides intuitive comparisons between deterministic and stochastic mine planning methods using TopCone Algorithm (TCA) and mathematical programming techniques proposed in this thesis. On the one hand, the stochastic framework using stochastic integer programming (SIP) overwhelms its deterministic counterpart using integer programming (IP) in the ability to cope with geological uncertainty. Specifically, the risk-based mine plan can be achieved with a minimal risk of not meeting production targets. On the other hand, the stochastic framework is
difficult to implement in all factors, from preparing orebody simulations, methodology, and determining unit costs associated with deviation penalty.

The mining industry is known for its slow reaction and scepticism to new ideas as any changes in mine planning method could pose a stake to its highly valuable assets. However, it is reasonable to believe that risk-based optimisation approaches using mathematical programming will be the future of the industry and the research in this thesis was indeed inspired by such perspective.

In a short-term period, acknowledging that kriging estimation and deterministic mine planning still well remain their popularity in the industry, it is recommended to validate deterministic mine planning model against uncertainty, particularly geological grade and tonnage variability. Ignoring this factor could turn a promising mining project in feasibility study stage into a failure in the future.
CHAPTER 8. CONCLUSIONS AND FUTURE WORK

8.1 CONCLUSIONS

Over the last six decades, the application of operations research techniques, namely linear programming (LP), integer programming (IP), mixed integer programming (MIP), and stochastic integer programming (SIP), has proved their capability in solving the mine planning and production scheduling optimisation problems. The major remaining challenge to address real-life problems is that the intensity of data is too great to be solved. One common approach to mitigate this issue is to aggregate blocks into larger units. However, the majority of available algorithms either fails to control the number of aggregates generated or incorporate slope constraints in the aggregation process.

One main contribution of this study is the development a new block aggregation algorithm, called TopCone Algorithm (TCA), which can effectively and efficiently address both drawbacks of available algorithms. By formulating an LP model for a network flow consisting of underlying nodes, overlying nodes, sink node, source node, and flows, blocks are grouped in clusters called TopCones (TCs) having major features:

- Can be extracted without violating slope safety;
- The number of TCs can be managed as small as close to or equal to 1;
- The union of TCs found by TCA is a near-optimal ultimate pit, so this makes TCA a hybrid of block aggregation and ultimate pit limit methods.

These unique characteristics of TCA lead to the development of a new mine planning method consisting of two stages instead of three like traditional approach. In the first stage, blocks aggregated into TCs, whose combination also forms an ultimate pit limit. In the second stage, TCs form the basis of a mathematical-based scheduling model, which can be IP or SIP, to solve the long-term production scheduling problem of mine planning.
Another significant contribution of this thesis is to investigate the relatively new concept of stochastic mine planning as well as the impact of geological uncertainty into mine planning solutions. These contributions include:

- Firstly, a simulation framework using minimum/maximum autocorrelation factors (MAF) and sequential Gaussian simulation (SGS) was proposed to tackle the multivariate nature of the iron ore deposit given as case study in this thesis. This simulation framework is scalable to be applied to other multivariate deposits, such as coal or rare earth.
- Secondly, TCA was applied on an averaged model constructed from 20 orebody realisations generated by the simulation framework to significantly reduce the data scale.
- Thirdly, a new SIP model was developed to read stochastic values of TCs as input data and simultaneously maximise discounted cash flow of the production schedule and minimise the risk of not meeting scheduling targets.
- Finally, risk profiles of the mine plan were constructed by sequentially applying the suggested mining sequence on 20 orebody realisations, considering that each iteration yields an equally possible outcome of the mine plan when it is implemented in the future. The risk profiles of the stochastic mine plan proved the ability of this method in significantly reducing the risk caused by geological uncertainty. Furthermore, risk profiles were also constructed for a kriging-based mine planning using TCA and IP. The results unveiled that the deterministic mine plan overestimated the expected project’s net present value (NPV) as well as a high risk of not meeting ore production and blending grade targets.

The above concepts and models were programmed using Visual Basic .Net programming language and compiled by Visual Studio 2012 (Microsoft, 2012). Also, several small subroutines were also built to automate the computational experiments, such as building E-type orebody model from realisations, calculating stochastic values of TCs, and calculating risk profiles of a given schedule. Within the three main models, CPLEX functions were invoked internally by using three callable libraries of ILOG, namely cplex124.dll, ILOG.Concert.dll, and ILOG.CPLEX.dll, instead of implementing CPLEX executable program externally to solve mathematical
models. This advanced programming technique allows to automate the solving process and reduce computing time by multiple folds.

The concepts, models and executable programs proposed and developed in this thesis were applied on the Silvergrass East iron ore deposit in Pilbara region, Western Australia. Both lower and upper bound operational constraints of mining and processing capacities, blending grades of Fe, SiO₂, Al₂O₃, P, LOI, and GOL (a particular type of iron ore which increases the rate of lost ore and reduces the capacity of screening station), were taken into the scheduling optimisation process.

The deterministic mine planning model using TCA and IP showed an improvement of 5.5% to 7% higher NPV values as well as more consistent grade blending requirements than a commercial mine planning software package. The stochastic mine planning model using TCA and SIP demonstrated its ability in minimising the risk of not meeting production targets. The risk profiles of blending grades showed virtually no violation to blending requirements. In addition, the risk profile of NPV showed only -0.32% to 0.41% gap to the expected NPV target, while the corresponding figure of the non-risk-based model is 7.01% to 7.81%.

8.2 FUTURE WORK

In this study, minimum cone size (MCS) parameter was introduced during the post-processing of TCs to control the number of TCs generated. In some cases, particularly when MCS is set too high, some TCs have a very high number of blocks, the so-called oversize TCs. These oversize TCs may reduce the flexibility of the downstream scheduling model and even make it infeasible depending on the narrowness of operational constraints. One interesting future work could be to introduce a mechanism to control the maximum cone size or develop a partial mining strategy for TCs, such as by using MIP.

It would also be interesting to investigate the possibility of applying a very high MCS parameter, so only a tiny number of TCs is generated to play a role as pushbacks. TCs now can be divided into smaller units according to elevation benches and develop mathematical-based scheduling models on these units. This approach will overcome the oversize problem of TCs automatically.
Although the solution time of TCA-based scheduling models is practical where large-scale mine planning problems can be solved in less than one hour using a normal office computer, there is still room to improve the code of TCA to reduce the time generating TCs and apply new technique to solve mathematical-based scheduling models. Moreover, additional variables can be considered in scheduling models to improve its quality, such as the distance between TCs to avoid mining sequencing of TCs being too scattering as well as improving mining width.

The proposed two-stage framework could also be beneficial toward the global optimality of open pit mine planning as ultimate pit limit and pushback design stages are discarded. Future study should focus on quantifying this potential benefit of the new mine planning framework.
REFERENCES


Pana, M., & Carlson, T. (1966). Description of a computer technique used in mine planning of the Utah Mine of Kennecott Copper Corp. *6th APCOM*,


Orebody modelling and strategic mine planning. The Australasian Institute of Mining and Metallurgy, Melbourne, 385-391.


APPENDIX A. VB .NET CODE OF THE DETERMINISTIC PRODUCTION SCHEDULING MODEL

This appendix presents the VB .Net code to implement the deterministic production scheduling using TopCones and integer programming derived in Chapter 4.

'Deterministic production scheduling using TopCones and Integer Programming
'Copyright (c) 2017 Ngoc Luan Mai (maingocluan@gmail.com)

'The program reads block data from the block model csv file, TopCones from topcone.csv
'then optimises production schedule using Integer Programming
'The scheduling result is then exported in the csv format, which is readable by a mine design software, 'such as Surpac

Imports ILOG.Concert
Imports ILOG.CPLEX

Module Module1

    Dim nconeFinal As Long 'number of TopCones

    Public Class BlockValue
        Public id As Long 'block ID
        Public tonnage As Long 'block tonnage
        Public fe As Double 'block's iron grade
        Public sio2 As Double 'block's silica grade
        Public al2o3 As Double 'block's aluminum grade
        Public p As Double 'block's phospho grade
        Public loi As Double 'block's loss on ignition grade
        Public gol As Double 'block's ochreous goethite grade
        Public bev As Double 'block economic value
        Public schedule As Integer 'scheduling period
        Public X As Long 'I value of block
        Public Y As Long 'J value of block
        Public Z As Long 'K value of block
        Public coorX As Double 'coordinate X value of block
        Public coorY As Double 'coordinate Y value of block
        Public coorZ As Double 'coordinate Z value of block
    End Class

    Public Class coneDefine
        Public Name As Long
        Public value As Double
        Public oreton As Double
        Public wasteton As Double
        Public fe As Double
        Public sio2 As Double
        Public al2o3 As Double
        Public p As Double
        Public loi As Double
        Public gol As Double
    End Class

End Module
Public schedule As Integer
Public nodeID() As Long  'belonging nodes: root node + branch nodes
Public precedence() As Long
Public Sub New()
  ReDim nodeID(10000)
  ReDim precedence(1000)
End Sub
End Class

Dim Xmax As Long, Ymax As Long, Zmax As Long
Dim sa As Double = 45  'slope angle
Dim maxblock As Long  'total blocks of the block model
Dim block() As BlockValue
Dim STfinal() As coneDefine  'scheduling parameters:
Dim mipgap As Double = 0.05  'relative optimality gap tolerance
Dim femax As Double = 60.5  'max blending fe grade
Dim femin As Double = 58.5  'min blending fe grade
Dim sio2max As Double = 5.7  'max blending sio2 grade
Dim sio2min As Double = 0  'min blending sio2 grade
Dim al2o3max As Double = 2.8  'max blending al2o3 grade
Dim al2o3min As Double = 0  'min blending al2o3 grade
Dim pmax As Double = 0.062  'max blending p grade
Dim pmin As Double = 0  'min blending p grade
Dim loimax As Double = 6.8  'max blending loi grade
Dim loimin As Double = 0  'min blending loi grade
Dim golmax As Double = 15.2  'max blending gol grade
Dim golmin As Double = 0  'min blending gol grade
Dim pcmax As Long = 28000000  'max processing capacity
Dim pcmin As Long = 22000000  'min processing capacity,
Dim mcmax As Long = 75000000  'max mining capacity
Dim mcmin As Long = 30000000  'min mining capacity
Dim dr As Double = 10  'discount rate
Dim p As Integer = 8  'number of scheduling periods

Public Sub Main()
  Dim sw1 = New Stopwatch()
  sw1.Start()
  Console.WriteLine("Reading input data ...")
  readinput()
  ip()
  Console.WriteLine("Writing schedule file ...")
  PrintScheduleIP()
  Console.WriteLine("Total solution time: ")
  sw1.Stop()
  Console.WriteLine(sw1.Elapsed)
  Console.Write("press any key to finish ...")
  Console.ReadKey()
End Sub

Sub readinput()
  'This subroutine reads input data from csv file 'silver.csv'
  'which is the csv file holding information of the block model of the Silvergrass East iron ore deposit
'1. Read input data from silver.csv

Dim InputDataTable As New DataTable("DataSet")
InputDataTable.Columns.Add("CoorX", GetType(Double))
InputDataTable.Columns.Add("CoorY", GetType(Double))
InputDataTable.Columns.Add("CoorZ", GetType(Double))
InputDataTable.Columns.Add("X", GetType(Long))
InputDataTable.Columns.Add("Y", GetType(Long))
InputDataTable.Columns.Add("Z", GetType(Long))
InputDataTable.Columns.Add("fe", GetType(Double))
InputDataTable.Columns.Add("sio2", GetType(Double))
InputDataTable.Columns.Add("al2o3", GetType(Double))
InputDataTable.Columns.Add("p", GetType(Double))
InputDataTable.Columns.Add("loi", GetType(Double))
InputDataTable.Columns.Add("gol", GetType(Double))
InputDataTable.Columns.Add("tonnage", GetType(Double))
InputDataTable.Columns.Add("BEV", GetType(Double))
FileOpen(1, "silver.csv", OpenMode.Input)
Dim text() As String, textline As String
Dim i, j As Long
Do Until EOF(1)
    i = i + 1
    textline = LineInput(1)
    text = Split(textline, ",")
    If i <> 1 Then
        InputDataTable.Rows.Add(text(0), text(1), text(2), text(4), text(5), text(6), text(7), _
            text(8), text(9), text(10), text(11), text(12), text(13), text(14))
    End If
Loop
InputDataTable.AcceptChanges()
maxblock = i - 1

FileClose(1)

'2. Sort the DataTable

Dim view As New DataView(InputDataTable)
' sort by Z and Y and X in decending order
view.Sort = "Z DESC, Y DESC, X DESC"

For Each row As DataRowView In view
    Xmax = CLng(row("X"))
    Ymax = CLng(row("Y"))
    Zmax = CLng(row("Z"))
Next
Console.WriteLine("Model Framework Dimensions:" & vbTab & "Xmax: " & Xmax & vbTab & "Ymax: " & Ymax & vbTab & "Zmax: " & Zmax)

'3. CallClass
ReDim block(0 To maxblock)
For i = 0 To maxblock
    block(i) = New BlockValue
Next

'==================================================================================
' 4. Transfer information for DataTable into blocks
'==================================================================================

i = 0
For Each row As DataRowView In view
    i = i + 1
    block(i).coorX = CDbl(row("CoorX"))
    block(i).coorY = CDbl(row("CoorY"))
    block(i).coorZ = CDbl(row("CoorZ"))
    block(i).X = CLng(row("X"))
    block(i).Y = CLng(row("Y"))
    block(i).Z = CLng(row("Z"))
    block(i).id = i
    block(i).fe = Math.Round(CDbl(row("fe")), 3)
    block(i).sio2 = Math.Round(CDbl(row("sio2")), 3)
    block(i).al2o3 = Math.Round(CDbl(row("al2o3")), 3)
    block(i).p = Math.Round(CDbl(row("p")), 3)
    block(i).loi = Math.Round(CDbl(row("loi")), 3)
    block(i).gol = Math.Round(CDbl(row("gol")), 2)
    block(i).tonnage = Math.Round(CDbl(row("tonnage")), 1)
    block(i).bev = Math.Round(CDbl(row("BEV")), 1)
Next

Console.WriteLine(maxblock & " blocks has been read")
End Sub

Sub ip()
    'The subroutine formulates and solves an integer programming model to
    'optimise production schedule

    Console.WriteLine("Initialising IP model ...")
    Try

'==================================================================================
' 1. Read TopCone.csv to collect information of TopCones
'==================================================================================

Dim coneTot(0 To 50000) As coneDefine
FileOpen(1, "TopCone.csv", OpenMode.Input)
Dim text() As String, textline As String
Dim i, j, k As Long
Do Until EOF(1)
    textline = LineInput(1)
    text = Split(textline, ",")
    i = i + 1
    coneTot(i) = New coneDefine
    coneTot(i).value = Math.Round(CDbl(text(0)), 1)
    coneTot(i).oreton = Math.Round(CDbl(text(1)), 1)
    coneTot(i).wasteton = Math.Round(CDbl(text(2)), 1)
    coneTot(i).fe = Math.Round(CDbl(text(3)), 3)
coneTot(i).sio2 = Math.Round(CDbl(text(4)), 3)
coneTot(i).al2o3 = Math.Round(CDbl(text(5)), 3)
coneTot(i).p = Math.Round(CDbl(text(6)), 4)
coneTot(i).loi = Math.Round(CDbl(text(7)), 3)
coneTot(i).gol = Math.Round(CDbl(text(8)), 3)
Dim j1 As Long = 0
ReDim coneTot(i).nodeID(text.Length - 10)
For j = 9 To text.Length - 1
    coneTot(i).nodeID(j1) = text(j)
    j1 = j1 + 1
Next
Loop
FileClose(1)

'===================================================================
'2. Declare relevant variables and parameters for IP model
'===================================================================
Dim n5 As Long = i 'n5: number of TopCones
Dim n6 As Long = n5 * p 'n6: number of binary variables
Console.WriteLine("number of scheduling cones: " & vbTab & n5)
Console.WriteLine("number of binary variables: " & vbTab & n6)

Dim cplex As New Cplex()
Dim var(1)() As INumVar
Dim rng(1)() As IRange

Dim lb(n6) As Double
Dim ub(n6) As Double
Dim varname(n6) As String
Dim xt(n6) As NumVarType

For j = 0 To n6
    lb(j) = 0.0
    ub(j) = 1.0
    xt(j) = NumVarType.Bool
Next

Dim objvals(n6) As Double
objvals(0) = 0.0
j = 1
For i = 1 To n5
    Dim j1 As Long = 0
    For k = j To j + p - 1
        j1 = j1 + 1
        varname(k) = "X" & j1 & "." & i
        objvals(k) = Math.Round((1 / ((1 + dr / 100) ^ j1)) * coneTot(i).value, 1)
    Next
    j = j + p
Next

'declare binary variables x
Dim x As INumVar() = cplex.NumVarArray(n6 + 1, lb, ub, xt, varname)
var(0) = x
'3. Declare objective function of the IP model
'=================================================================
Console.WriteLine("Writing objective function ...")
cplex.AddMaximize(cplex.ScalProd(x, objvals))
'=================================================================

'4. Declare constraints of the IP model
'=================================================================
Console.WriteLine("Writing constraints ...")
rng(0) = New IRangenn6 * 2 {}
k = 0
FileOpen(1, "precedence.csv", OpenMode.Output) 'store precedence relationship of TopCones

'4.1. Sequencing constraints
For i = 1 To n5
    Dim k9 As Long = 0
    For j = 1 To i - 1
        Dim j1 As Long = 0
        While j1 <= coneTot(i).nodeID.Length - 1
            Dim j2 As Long = coneTot(j).nodeID.Length - 1
            While j2 > 0
                If block(coneTot(i).nodeID(j2)).Z - block(coneTot(i).nodeID(j1)).Z = 1 Then
                    If Math.Round((block(coneTot(i).nodeID(j2)).coorX -
                        block(coneTot(i).nodeID(j1)).coorX) ^ 2 -
                        block(coneTot(i).nodeID(j2)).coorY +
                        block(coneTot(i).nodeID(j1)).coorY) ^ 2, 1) <=
                        Math.Round((Math.Tan(sa * Math.PI / 180) -
                        (block(coneTot(i).nodeID(j2)).coorZ -
                        block(coneTot(i).nodeID(j1)).coorZ)) ^ 2, 1) Then
                        k9 = k9 + 1
                        coneTot(i).precedence(k9) = j
                        Continue For
                    End If
                End If
            End While
            j1 = j1 + 1
        End While
    Next
coneTot(i).precedence = coneTot(i).precedence.Distinct().ToArray()
Array.Sort(coneTot(i).precedence)

For j = 1 To coneTot(i).precedence.Length - 2
    For j1 = j + 1 To coneTot(i).precedence.Length - 1
        For j2 As Long = 1 To coneTot(coneTot(i).precedence(j1)).precedence.Length - 1
            If coneTot(i).precedence(j) = coneTot(coneTot(i).precedence(j1)).precedence(j2) then
            End If
        Next
coneTot(i).precedence(j) = 0
End If
Next
Next
Next

coneTot(i).precedence = coneTot(i).precedence.Distinct().ToArray()
Array.Sort(coneTot(i).precedence)

Dim text1 As String = ""
If coneTot(i).precedence.Length > 1 Then
  For j = 1 To coneTot(i).precedence.Length - 1
    text1 = text1 & coneTot(i).precedence(j) & ","
  Next
End If
PrintLine(1, text1)

'write sequencing constraints for cone i:
If coneTot(i).precedence.Length > 1 Then
  For j1 = 1 To p
    Dim expr As ILinearNumExpr = cplex.LinearNumExpr()
    expr.AddTerm(coneTot(i).precedence.Length - 1, x((i - 1) * p + j1))
    For j2 As Long = 1 To coneTot(i).precedence.Length - 1
      For j3 = 1 To j1
        expr.AddTerm(-1.0, x((coneTot(i).precedence(j2) - 1) * p + j3))
      Next
    Next
    k = k + 1
    rng(0)(k) = cplex.AddLe(expr, 0, "c" & k)
  Next
End If

FileClose(1)

'4.2. Blending constraints
'4.2.1. Upper bound constraints
For k1 = 1 To p
  'max blending constraints of fe
  Array.Clear(objvals, 0, n6 + 1)
  For i = 1 To n5
    objvals((i - 1) * p + k1) = Math.Round((coneTot(i).fe - femax) * coneTot(i).oreton, 0)
  Next
  k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), 0, "c" & k)

  'max blending constraints of sio2
  Array.Clear(objvals, 0, n6 + 1)
  For i = 1 To n5
    objvals((i - 1) * p + k1) = Math.Round((coneTot(i).sio2 - sio2max) * coneTot(i).oreton, 0)
  Next
  k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), 0, "c" & k)

  'max blending constraints of al2o3
  Array.Clear(objvals, 0, n6 + 1)
For $i = 1$ To $n5$
  objvals((i - 1) * $p + k1) = Math.Round((coneTot(i).al2o3 - al2o3max) * coneTot(i).oreton, 0)
Next
k = k + 1
rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), 0, "c" & k)

' max blending constraints of $p$
Array.Clear(objvals, 0, n6 + 1)
For $i = 1$ To $n5$
  objvals((i - 1) * $p + k1) = Math.Round((coneTot(i).p - pmax) * coneTot(i).oreton, 0)
Next
k = k + 1
rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), 0, "c" & k)

' max blending constraints of $loi$
Array.Clear(objvals, 0, n6 + 1)
For $i = 1$ To $n5$
  objvals((i - 1) * $p + k1) = Math.Round((coneTot(i).loi - loimax) * coneTot(i).oreton, 0)
Next
k = k + 1
rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), 0, "c" & k)

' max blending constraints of $gol$
Array.Clear(objvals, 0, n6 + 1)
For $i = 1$ To $n5$
  objvals((i - 1) * $p + k1) = Math.Round((coneTot(i).gol - golmax) * coneTot(i).oreton, 0)
Next
k = k + 1
rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), 0, "c" & k)
Next

' 4.2.2. Lower bound constraints
For $k1 = 1$ To $p$
  ' min blending constraints of $fe$
Array.Clear(objvals, 0, n6 + 1)
For $i = 1$ To $n5$
  objvals((i - 1) * $p + k1) = Math.Round((coneTot(i).fe - femin) * coneTot(i).oreton, 0)
Next
k = k + 1
rng(0)(k) = cplex.AddGe(cplex.ScalProd(x, objvals), 0, "c" & k)
Next

' 4.3. Processing plant capacity constraints
' 4.3.1. Upper bound constraints
For $k1 = 1$ To $p$
  Array.Clear(objvals, 0, n6 + 1)
For $i = 1$ To $n5$
  objvals((i - 1) * $p + k1) = coneTot(i).oreton
Next
k = k + 1
rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), pcmax, "c" & k)
Next

' 4.3.2. Lower bound constraints
For $k1 = 1$ To $p$
  Array.Clear(objvals, 0, n6 + 1)
  For $i = 1$ To $n5$
    '
objvals((i - 1) * p + k1) = coneTot(i).oreton
Next
k = k + 1
rng(0)(k) = cplex.AddGe(cplex.ScalProd(x, objvals), pcmin, "c" & k)
Next

'4.4. Mining capacity constraints
'4.4.1. Upper bound constraints
For k1 = 1 To p
    Array.Clear(objvals, 0, n6 + 1)
    For i = 1 To n5
        objvals((i - 1) * p + k1) = coneTot(i).oreton + coneTot(i).wasteton
        Next
    k = k + 1
    rng(0)(k) = cplex.AddLe(cplex.ScalProd(x, objvals), mcmax, "c" & k)
Next

'4.4.2. Lower bound constraints
For k1 = 1 To p
    Array.Clear(objvals, 0, n6 + 1)
    For i = 1 To n5
        objvals((i - 1) * p + k1) = coneTot(i).oreton + coneTot(i).wasteton
        Next
    k = k + 1
    rng(0)(k) = cplex.AddGe(cplex.ScalProd(x, objvals), mcmin, "c" & k)
Next

'4.5. Reserve constraints
For i = 1 To n5
    Dim expr As ILinearNumExpr = cplex.LinearNumExpr()
    For j = 1 To p
        expr.AddTerm(1, x((i - 1) * p + j))
    Next
    k = k + 1
    rng(0)(k) = cplex.AddLe(expr, 1, "c" & k)
Next

'=================================================================
'5. Solve the IP model
'=================================================================
Console.WriteLine("Solving IP model by CPLEX ...")

'Set optimality gap
If cplex.Solve() Then
    Dim a As Double() = cplex.GetValues(var(0))

'5.1. Print out solution status and value
    cplex.Output().WriteLine("solution status = " + cplex.GetStatus().ToString)
    cplex.Output().WriteLine("solution value = " + cplex.ObjValue.ToString)
    Console.WriteLine("Total TCs: " & n5)
    Console.WriteLine("gap: " & mipgap)
Dim nvars As Integer = a.Length
i = 0
j = 0
For j = 1 To nvars - 1
    If (j - 1) Mod p = 0 Then
        i = i + 1
    End If
    If Math.Round(a(j), 0) = 1 Then
        If j Mod p = 0 Then
            coneTot(i).schedule = p
        Else
            coneTot(i).schedule = j Mod p
        End If
    End If
Next
Else
    Console.WriteLine("Infeasible")
    Console.ReadKey()
End If

'=================================================================
'6. Calculate scheduling results of each period
'=================================================================
Dim oreton(0 To p) As Double  'store total ore ton of period p
Dim wasteton(0 To p) As Double  'store total waste ton of period p
Dim bev(0 To p) As Double  'store total value of period p
Dim gradefe(0 To p) As Double  'store average fe grade of period p
Dim gradesio2(0 To p) As Double  'store average sio2 grade of period p
Dim gradeal2o3(0 To p) As Double  'store average al2o3 grade of period p
Dim gradeal2o3(0 To p) As Double  'store average value of period p
Dim gradep(0 To p) As Double  'store average p grade of period p
Dim gradeloi(0 To p) As Double  'store average loi grade of period p
Dim gradegol(0 To p) As Double  'store average gol grade of period p

For i = 1 To n5
    j = 0
    While j <= coneTot(i).nodeID.Length - 1
        block(coneTot(i).nodeID(j)).schedule = coneTot(i).schedule
        j = j + 1
    End While

For j1 = 1 To p
    If coneTot(i).schedule = j1 Then
        oreton(j1) = oreton(j1) + coneTot(i).oreton
        wasteton(j1) = wasteton(j1) + coneTot(i).wasteton
        bev(j1) = bev(j1) + coneTot(i).value
        gradefe(j1) = gradefe(j1) + coneTot(i).fe * coneTot(i).oreton
        gradesio2(j1) = gradesio2(j1) + coneTot(i).sio2 * coneTot(i).oreton
        gradeal2o3(j1) = gradeal2o3(j1) + coneTot(i).al2o3 * coneTot(i).oreton
        gradeal2o3(j1) = gradeal2o3(j1) + coneTot(i).p * coneTot(i).oreton
        gradeloi(j1) = gradeloi(j1) + coneTot(i).loi * coneTot(i).oreton
        gradegol(j1) = gradegol(j1) + coneTot(i).gol * coneTot(i).oreton
    End If
Next
Next
For i = 1 To p
    gradefe(i) = gradefe(i) / oreton(i)
    gradesio2(i) = gradesio2(i) / oreton(i)
gradeal2o3(i) = gradeal2o3(i) / oreton(i)
gradepl(i) = gradepl(i) / oreton(i)
gradeloi(i) = gradeloi(i) / oreton(i)
gradegol(i) = gradegol(i) / oreton(i)
Next

'=================================================================
FileOpen(1, "production.csv", OpenMode.Output)

FileClose(1)

Dim bgol As Double = 0 'store gol
Dim bloi As Double = 0 'store loi
Dim bsio2 As Double = 0 'store sio2
Dim bfe As Double = 0 'store fe
Dim b5 As Double = 0 'store total NPV

For j = 1 To p
    PrintLine(1, j & "," & Math.Round(gradepl(j), 3) & "," & Math.Round(gradesio2(j), 3))
    gradep(j) = Math.Round(gradepl(j), 3)
    gradeloi(j) = Math.Round(gradesio2(j), 3)
    gradegol(j) = Math.Round(gradepl(j) + gradeloi(j), 3)
    bp = Math.Round(bp / b1, 3)
next

bfe = Math.Round(bfe / b1, 3)
bsio2 = Math.Round(bsio2 / b1, 3)
bal2o3 = Math.Round(bal2o3 / b1, 3)
bp = Math.Round(bp / b1, 3)
bloi = Math.Round(bloi / b1, 3)
bgol = Math.Round(bgol / b1, 3)

PrintLine(1, "NPV")

FileClose(1)
Public Sub PrintScheduleIP()
    'The subroutine prints out the scheduling result into a csv file,
    'which is readable by any mine design software, such as Surpac
    FileOpen(1, "scheduleTCA.csv", OpenMode.Input)
    FileOpen(2, "scheduleIP.csv", OpenMode.Output)
    Dim textline As String
    Dim i As Long = 0
    Do Until EOF(1)
        i = i + 1
        textline = LineInput(1)
        PrintLine(2, textline & "," & block(i).schedule)
    Loop
    FileClose(2)
    FileClose(1)
End Sub
End Module
APPENDIX B. VB .NET CODE OF THE UNCERTAINTY GENERATOR PROGRAM

This appendix presents the VB .Net code to calculate the uncertainty values of TopCones from orebody realisations derived in Chapter 6.

'Uncertainty Generator Program
'Copyright (c) 2017 Ngoc Luan Mai (maingocluan@gmail.com)

'The program calculates stochastic values of TopCones from orebody realisations. The result is printed 'out as TCUncertainty.csv, which is readable by the downstream stochastic production scheduling program

Module Module1

Public Class BlockValue
    Public tonnage() As Long 'block tonnage
    Public fe() As Double 'block's iron grade
    Public sio2() As Double 'block's silica grade
    Public al2o3() As Double 'block's aluminium grade
    Public p() As Double 'block's phospho grade
    Public loi() As Double 'block's loss on ignition grade
    Public gol() As Double 'block's ochreous goethite grade
    Public bev() As Double 'block's economic value
    Public Sub New()
        ReDim tonnage(S)
        ReDim bev(S)
        ReDim fe(S)
        ReDim sio2(S)
        ReDim al2o3(S)
        ReDim p(S)
        ReDim loi(S)
        ReDim gol(S)
    End Sub
    End Class

Public Class coneDefine
    Public value() As Double 'economic value of TopCone
    Public oreton() As Double 'ore tonnage of TopCone
    Public wasteton() As Double 'waste tonnage of TopCone
    Public fe() As Double 'iron grade of TopCone
    Public sio2() As Double 'silica grade of TopCone
    Public al2o3() As Double 'aluminium grade of TopCone
    Public p() As Double 'phospho grade of TopCone
    Public loi() As Double 'loss on ignition grade of TopCone
    Public gol() As Double 'ochreous goethite grade of TopCone
    Public nodeID() As Long 'belonging nodes: root node + branch nodes
    Public Sub New()
        ReDim nodeID(5000)
        ReDim value(S)
        ReDim oreton(S)
        ReDim wasteton(S)
        ReDim fe(S)
ReDim sio2(S)
ReDim al2o3(S)
ReDim p(S)
ReDim loI(S)
ReDim gol(S)
End Sub
End Class

Dim Xmax As Long, Ymax As Long, Zmax As Long
Dim S As Long = 20 'number of orebody realisations
Dim p As Long = 10 'number of scheduling periods
Dim fe_cutoff As Double = 50 'cut-off grade of iron content
Dim maxblock As Long 'number of block of the block model
Dim block() As BlockValue

Public Sub Main()

'==================================================================================
'1. Declare block class
'==================================================================================
Dim text() As String, textline As String
Dim n, i, j As Long

FileOpen(1, "silver_sim1.csv", OpenMode.Input)
Do Until EOF(1)
    LinenInput(1)
    maxblock = maxblock + 1
Loop
maxblock = maxblock - 1
FileClose(1)

'declare blocks
ReDim block(0 To maxblock)
For i = 0 To maxblock
    block(i) = New BlockValue
Next

'==================================================================================
'2. Collect simulated values of blocks from realisations
'==================================================================================
Console.WriteLine("Collecting simulated blocks attributes from realisations...")
For n = 1 To S
    i = 0
    FileOpen(1, "silver_sim" & n & ".csv", OpenMode.Input)
    Do Until EOF(1)
        i = i + 1
        textline = LinenInput(1)
        If i = 1 Then
            Continue Do
        End If
        text = Split(textline, ",")
        block(i - 1).fe(n) = Math.Round(CDbl(text(7)), 2)
        block(i - 1).sio2(n) = Math.Round(CDbl(text(8)), 2)
        block(i - 1).al2o3(n) = Math.Round(CDbl(text(9)), 2)
    Loop
End Do

}
block(i - 1).p(n) = Math.Round(CDbl(text(10)), 2)
block(i - 1).lo(n) = Math.Round(CDbl(text(11)), 2)
block(i - 1).gol(n) = Math.Round(CDbl(text(12)), 2)
block(i - 1).tonnage(n) = Math.Round(CDbl(text(13)), 3)
block(i - 1).bev(n) = Math.Round(CDbl(text(14)), 1)

Loop
FileClose(1)
Next

'==================================================================================
'3. Collect cones' deterministic attributes from TopCone.csv
'==================================================================================
Console.WriteLine("Collect cones' deterministic attributes from TopCone.csv...")
Dim coneTot(0 To 10000) As coneDefine
FileOpen(1, "TopCone.csv", OpenMode.Input)
i = 0
Do Until EOF(1)
textline = LineInput(1)
text = Split(textline, ",")
i = i + 1
coneTot(i) = New coneDefine
coneTot(i).value(0) = Math.Round(CDbl(text(0)), 1)
coneTot(i).oreton(0) = Math.Round(CDbl(text(1)), 1)
coneTot(i).wasteton(0) = Math.Round(CDbl(text(2)), 1)
coneTot(i).fe(0) = Math.Round(CDbl(text(3)), 3)
coneTot(i).sio2(0) = Math.Round(CDbl(text(4)), 3)
coneTot(i).al2o3(0) = Math.Round(CDbl(text(5)), 3)
coneTot(i).p(0) = Math.Round(CDbl(text(6)), 3)
coneTot(i).loi(0) = Math.Round(CDbl(text(7)), 3)
coneTot(i).gol(0) = Math.Round(CDbl(text(8)), 3)
Dim j1 As Long = 0
For j = 9 To text.Length - 1
    coneTot(i).nodeID(j1) = text(j)
j1 = j1 + 1
Next
FileClose(1)
Dim n5 As Long = i   'n5: total cones

'==================================================================================
'4. Calculate cones' stochastic attributes from blocks' stochastic value
'==================================================================================
Console.WriteLine("Calculate TopCone' stochastic attributes and standard deviation...")
For i = 1 To n5
    For n = 1 To S
        j = 0
        While coneTot(i).nodeID(j) <> 0
            If block(coneTot(i).nodeID(j)).fe(n) >= fe_cutoff Then
                coneTot(i).oreton(n) = coneTot(i).oreton(n) + block(coneTot(i).nodeID(j)).tonnage(n)
                coneTot(i).fe(n) = coneTot(i).fe(n) + block(coneTot(i).nodeID(j)).fe(n) _
                * block(coneTot(i).nodeID(j)).tonnage(n)
                coneTot(i).sio2(n) = coneTot(i).sio2(n) + block(coneTot(i).nodeID(j)).sio2(n) _
                * block(coneTot(i).nodeID(j)).tonnage(n)
                coneTot(i).al2o3(n) = coneTot(i).al2o3(n) + block(coneTot(i).nodeID(j)).al2o3(n) _
                * block(coneTot(i).nodeID(j)).tonnage(n)
            End If
            j = j + 1
        Wend
    Next
Next
coneTot(i).p(n) = coneTot(i).p(n) + block(coneTot(i).nodeID(j)).p(n)
* block(coneTot(i).nodeID(j)).tonnage(n)
coneTot(i).loi(n) = coneTot(i).loi(n) + block(coneTot(i).nodeID(j)).loi(n)
* block(coneTot(i).nodeID(j)).tonnage(n)
coneTot(i).gol(n) = coneTot(i).gol(n) + block(coneTot(i).nodeID(j)).gol(n)
* block(coneTot(i).nodeID(j)).tonnage(n)

Else
coneTot(i).wasteton(n) = coneTot(i).wasteton(n) + block(coneTot(i).nodeID(j)).tonnage(n)
End If

coneTot(i).value(n) = coneTot(i).value(n) + block(coneTot(i).nodeID(j)).bev(n)
j = j + 1
End While

If coneTot(i).oreton(n) > 0 Then
coneTot(i).fe(n) = Math.Round(coneTot(i).fe(n) / coneTot(i).oreton(n), 3)
coneTot(i).sio2(n) = Math.Round(coneTot(i).sio2(n) / coneTot(i).oreton(n), 3)
coneTot(i).al2o3(n) = Math.Round(coneTot(i).al2o3(n) / coneTot(i).oreton(n), 3)
coneTot(i).p(n) = Math.Round(coneTot(i).p(n) / coneTot(i).oreton(n), 3)
coneTot(i).loi(n) = Math.Round(coneTot(i).loi(n) / coneTot(i).oreton(n), 3)
coneTot(i).gol(n) = Math.Round(coneTot(i).gol(n) / coneTot(i).oreton(n), 3)
End If
Next

'==================================================================================
'5. Calculate average value of cones’s attributes
'==================================================================================

For i = 1 To n5
coneTot(i).value(0) = 0
coneTot(i).oreton(0) = 0
coneTot(i).wasteton(0) = 0
coneTot(i).fe(0) = 0
coneTot(i).sio2(0) = 0
coneTot(i).al2o3(0) = 0
coneTot(i).p(0) = 0
coneTot(i).loi(0) = 0
coneTot(i).gol(0) = 0

For n = 1 To S
coneTot(i).value(0) = coneTot(i).value(0) + coneTot(i).value(n)
coneTot(i).oreton(0) = coneTot(i).oreton(0) + coneTot(i).oreton(n)
coneTot(i).wasteton(0) = coneTot(i).wasteton(0) + coneTot(i).wasteton(n)
coneTot(i).fe(0) = coneTot(i).fe(0) + coneTot(i).fe(n)
coneTot(i).sio2(0) = coneTot(i).sio2(0) + coneTot(i).sio2(n)
coneTot(i).al2o3(0) = coneTot(i).al2o3(0) + coneTot(i).al2o3(n)
coneTot(i).p(0) = coneTot(i).p(0) + coneTot(i).p(n)
coneTot(i).loi(0) = coneTot(i).loi(0) + coneTot(i).loi(n)
coneTot(i).gol(0) = coneTot(i).gol(0) + coneTot(i).gol(n)
Next

coneTot(i).value(0) = Math.Round(coneTot(i).value(0) / S, 1)
coneTot(i).oreton(0) = Math.Round(coneTot(i).oreton(0) / S, 1)
coneTot(i).wasteton(0) = Math.Round(coneTot(i).wasteton(0) / S, 1)
coneTot(i).fe(0) = Math.Round(coneTot(i).fe(0) / S, 3)
coneTot(i).sio2(0) = Math.Round(coneTot(i).sio2(0) / S, 3)
coneTot(i).al2o3(0) = Math.Round(coneTot(i).al2o3(0) / S, 3)
coneTot(i).p(0) = Math.Round(coneTot(i).p(0) / S, 3)
coneTot(i).loi(0) = Math.Round(coneTot(i).loi(0) / S, 3)
coneTot(i).gol(0) = Math.Round(coneTot(i).gol(0) / S, 3)
Next

'==================================================================================
'6. Print out the stochastic values of TopCones into TCUncertainty.csv file
'==================================================================================

FileOpen(1, "TCUncertainty.csv", OpenMode.Output)
Dim sLine As String
Dim sLine1 As String
For i = 1 To n5
    sLine = ""
sLine1 = ""
j = 0
While coneTot(i).nodeID(j) <> 0
    sLine = sLine & "," & coneTot(i).nodeID(j)
j = j + 1
End While
For n = 1 To S
    sLine1 = sLine1 & "," & coneTot(i).value(n) & "," & coneTot(i).oreton(n) _
    & "," & coneTot(i).wasteton(n) & "," & coneTot(i).fe(n) & "," & coneTot(i).sio2(n) _
    & "," & coneTot(i).al2o3(n) & "," & coneTot(i).p(n) _
    & "," & coneTot(i).loi(n) & "," & coneTot(i).gol(n)
Next
PrintLine(1, Math.Round(coneTot(i).value(0), 1) & "," & Math.Round(coneTot(i).oreton(0), 3) _
    & "," & Math.Round(coneTot(i).wasteton(0), 3) & "," & Math.Round(coneTot(i).fe(0), 3) _
    & "," & Math.Round(coneTot(i).sio2(0), 3) & "," & Math.Round(coneTot(i).al2o3(0), 3) _
    & "," & Math.Round(coneTot(i).p(0), 3) & "," & Math.Round(coneTot(i).loi(0), 3) _
    & "," & Math.Round(coneTot(i).gol(0), 3) & sLine1 & sLine)
Next
FileClose(1)

End Sub

End Module
APPENDIX C. VB .NET CODE OF THE STOCHASTIC PRODUCTION SCHEDULING MODEL

This appendix presents the VB .Net code to implement the stochastic production scheduling using TopCones and stochastic integer programming derived in Chapter 6.

'Uncertainty Generator Program
'Copyright (c) 2017 Ngoc Luan Mai (maingocluan@gmail.com)

'The program reads block data from the block model csv file, uncertainty values of TopCones
't from TCUncertainty.csv then optimises production schedule under uncertainty using
'The Uncertainty Generator Program. The scheduling result is then exported in the csv format,
'which is readable by a mine design software, such as Surpac

Imports ILOG.Concert
Imports ILOG.CPLEX

Module Module1

Dim nconeFinal As Long 'number of cones after relaxing

Public Class BlockValue
    Public id As Long 'block ID
    Public tonnage As Long 'block tonnage
    Public fe As Double 'block's iron grade
    Public sio2 As Double 'block's silica grade
    Public al2o3 As Double 'block's aluminum grade
    Public p As Double 'block's phospho grade
    Public loi As Double 'block's loss on ignition grade
    Public gol As Double 'block's ochreous goethite grade
    Public bev As Double 'block economic value
    Public schedule As Integer 'scheduling period
    Public X As Long 'I value of block
    Public Y As Long 'J value of block
    Public Z As Long 'K value of block
    Public coorX As Double 'coordinate X value of block
    Public coorY As Double 'coordinate Y value of block
    Public coorZ As Double 'coordinate Z value of block
End Class

Public Class coneDefine
    Public Name As Long 'root node name
    Public RootID() As Long 'ore tonnage of TopCone
    Public oreton() As Double 'ore tonnage of TopCone
    Public wasteton() As Double 'waste tonnage of TopCone
    Public fe() As Double 'iron grade of TopCone
    Public sio2() As Double 'silica grade of TopCone
    Public al2o3() As Double 'aluminium grade of TopCone
    Public p() As Double 'phospho grade of TopCone
    Public loi() As Double 'loss on ignition grade of TopCone
    Public gol() As Double 'ochreous goethite grade of TopCone
End Class
Public schedule As Integer  'scheduling period
Public nodeID() As Long  'belonging nodes: root node + branch nodes
Public precedence() As Long
Public Sub New()
    ReDim RootID(1000)
    ReDim nodeID(10000)
    ReDim precedence(1000)
    ReDim value(S)
    ReDim oreton(S)
    ReDim wasteton(S)
    ReDim fe(S)
    ReDim sio2(S)
    ReDim al2o3(S)
    ReDim p(S)
    ReDim loi(S)
    ReDim gol(S)
End Sub
End Class

Public Class FinalSchedule
    Public oreton() As Double
    Public totalton() As Double
    Public wasteton() As Double
    Public fe() As Double
    Public sio2() As Double
    Public al2o3() As Double
    Public ph() As Double
    Public loi() As Double
    Public gol() As Double
    Public Sub New()
        ReDim oreton(p)
        ReDim totalton(p)
        ReDim wasteton(p)
        ReDim fe(p)
        ReDim sio2(p)
        ReDim al2o3(p)
        ReDim ph(p)
        ReDim loi(p)
        ReDim gol(p)
    End Sub
End Class

Dim Xmax As Long, Ymax As Long, Zmax As Long
Dim sa As Double = 45  'slope angle
Dim S As Long = 20  'number of orebody realisations
Dim fe_cutoff As Double = 50  'cut-off grade of iron
Dim maxblock As Long  'total blocks of the block model
Dim block() As BlockValue
Dim STfinal() As coneDefine

'scheduling parameters:
Dim mipgap As Double = 0.05  'relative optimality gap tolerance
Dim femax As Double = 60.5  'max blending fe grade
Dim femin As Double = 58.5  'min blending fe grade
Dim sio2max As Double = 5.7  'max blending sio2 grade
Dim sio2min As Double = 0  'min blending sio2 grade
Dim al2o3max As Double = 2.8  'max blending al2o3 grade
Dim al2o3min As Double = 0 'min blending al2o3 grade
Dim pmax As Double = 0.062 'max blending p grade
Dim pmin As Double = 0 'min blending p grade
Dim loimax As Double = 6.8 'max blending loi grade
Dim loimin As Double = 0 'min blending loi grade
Dim golmax As Double = 15.2 'max blending gol grade
Dim golmin As Double = 0 'min blending gol grade
Dim pcmax As Long = 28000000 'max processing capacity
Dim pcmin As Long = 22000000 'min processing capacity,
Dim mcmax As Long = 75000000 'max mining capacity
Dim mcmin As Long = 30000000 'min mining capacity
Dim dr As Double = 10 'discount rate
Dim p As Integer = 8 'number of scheduling periods

'Uncertainty cost
Dim Cou As Double = 1 'cost of ore tonnage excess, $/ton
Dim Col As Double = 1 'cost of ore tonnage deficient, $/ton
Dim Cmu As Double = 1 'cost of total tonnage excess, $/ton
Dim Cml As Double = 1 'cost of total tonnage deficient, $/ton
Dim Cfeu As Double = 1 'cost of Fe grade excess, $/%ton
Dim Cfel As Double = 1 'cost of Fe grade deficient, $/%ton
Dim Csio2u As Double = 1 'cost of SiO2 grade excess, $/%ton
Dim Csio2l As Double = 1 'cost of SiO2 grade deficient, $/%ton
Dim Cal2o3u As Double = 1 'cost of Al2O3 grade excess, $/%ton
Dim Cal2o3l As Double = 1 'cost of Al2O3 grade deficient, $/%ton
Dim Cpu As Double = 1 'cost of P grade excess, $/%ton
Dim Cpl As Double = 1 'cost of P grade deficient, $/%ton
Dim Cloiu As Double = 1 'cost of LOI grade excess, $/%ton
Dim Cloil As Double = 1 'cost of LOI grade deficient, $/%ton
Dim Cgolu As Double = 1 'cost of GOL grade excess, $/%ton
Dim Cgoll As Double = 1 'cost of GOL grade deficient, $/%ton

Public Sub Main()
    Dim sw1 = New Stopwatch()
    sw1.Start()
    Console.WriteLine("Reading input data ...")
    readinput()
    sip()
    Console.WriteLine("Writing schedule file ...")
    PrintScheduleSIP()
    Console.Write("Total solution time: ")
    sw1.Stop()
    Console.WriteLine(sw1.Elapsed)
    Console.Write("press any key to finish ...")
    Console.ReadKey()
End Sub

Sub readinput()
    'This subroutine reads input data from csv file 'silver.csv'
    'which is the csv file holding information of the block model of the Silvergrass East iron ore deposit

'================================================================================================
'1. Read input data from silver.csv
'================================================================================================
Dim InputDataTable As New DataTable("DataSet") 'create a data table to import all raw input data, then use data view to sort/process
InputDataTable.Columns.Add("CoorX", GetType(Double))
InputDataTable.Columns.Add("CoorY", GetType(Double))
InputDataTable.Columns.Add("CoorZ", GetType(Double))
InputDataTable.Columns.Add("X", GetType(Long))
InputDataTable.Columns.Add("Y", GetType(Long))
InputDataTable.Columns.Add("Z", GetType(Long))
InputDataTable.Columns.Add("fe", GetType(Double))
InputDataTable.Columns.Add("sio2", GetType(Double))
InputDataTable.Columns.Add("ai2o3", GetType(Double))
InputDataTable.Columns.Add("p", GetType(Double))
InputDataTable.Columns.Add("loi", GetType(Double))
InputDataTable.Columns.Add("gol", GetType(Double))
InputDataTable.Columns.Add("tonnage", GetType(Double))
InputDataTable.Columns.Add("BEV", GetType(Double))
FileOpen(1, "silver.csv", OpenMode.Input)
Dim text() As String, textline As String
Dim i As Long = 0
Do Until EOF(1)
    i = i + 1
    textline = LineInput(1)
    text = Split(textline, ",")
    If i <> 1 Then 'skip the header
        InputDataTable.Rows.Add(text(0), text(1), text(2), text(4), text(5), text(6), text(7), _
        text(8), text(9), text(10), text(11), text(12), text(13), text(14))
    End If
Loop
InputDataTable.AcceptChanges()
maxblock = i - 1
FileClose(1)

'==================================================================================
'2. Sort the DataTable
'==================================================================================
Dim view As New DataView(InputDataTable)
'sort by Z and Y and X in decending order
view.Sort = "Z DESC, Y DESC, X DESC"
For Each row As DataRowView In view
    Xmax = CLng(row("X"))
    Ymax = CLng(row("Y"))
    Zmax = CLng(row("Z"))
Exit For
Next
Console.WriteLine("Model Framework Dimensions:" & vbTab & "; Xmax: " & Xmax & vbTab & _
    "; Ymax: " & vbTab & Ymax & vbTab & "; Zmax: " & vbTab & Zmax)

'==================================================================================
'3. CallClass
'==================================================================================
ReDim block(0 To maxblock)
For i = 0 To maxblock
    block(i) = New BlockValue
Next
i = 0
Dim totOreTon As Double = 0
For Each row As DataRowView In view
    i = i + 1
    block(i).coorX = CDbl(row("CoorX"))
    block(i).coorY = CDbl(row("CoorY"))
    block(i).coorZ = CDbl(row("CoorZ"))
    block(i).X = CLng(row("X"))
    block(i).Y = CLng(row("Y"))
    block(i).Z = CLng(row("Z"))
    block(i).id = i
    block(i).fe = Math.Round(CDbl(row("fe")), 3)
    block(i).sio2 = Math.Round(CDbl(row("sio2")), 3)
    block(i).al2o3 = Math.Round(CDbl(row("al2o3")), 3)
    block(i).p = Math.Round(CDbl(row("p")), 3)
    block(i).loi = Math.Round(CDbl(row("loi")), 3)
    block(i).gol = Math.Round(CDbl(row("gol")), 2)
    block(i).tonnage = Math.Round(CDbl(row("tonnage")), 1)
    block(i).bev = Math.Round(CDbl(row("BEV")), 1)
Next
Console.WriteLine(maxblock & " blocks has been read")
End Sub

Sub sip()
    'The subroutine formulates and solves a stochastic integer programming model to
    'optimise production schedule under uncertainty

    Console.WriteLine("Initialising SIP model ...")
    Try
        '===================================================================
        '1. Read TCUncertainty.csv to collect uncertainty values of TopCones
        '===================================================================
        Dim coneTot(0 To 10000) As coneDefine
        FileOpen(1, "TCUncertainty.csv", OpenMode.Input)
        Dim text() As String, textline As String
        Dim i, j, k As Long
        Dim j1 As Long = 0
        Do Until EOF(1)
            textline = LineInput(1)
            text = Split(textline, ",")
            i = i + 1
            coneTot(i) = New coneDefine
            coneTot(i).value(0) = Math.Round(CDbl(text(0)), 1)
            coneTot(i).oreton(0) = Math.Round(CDbl(text(1)), 1)
            coneTot(i).wasteton(0) = Math.Round(CDbl(text(2)), 1)
            coneTot(i).fe(0) = Math.Round(CDbl(text(3)), 3)
            coneTot(i).sio2(0) = Math.Round(CDbl(text(4)), 3)
            coneTot(i).al2o3(0) = Math.Round(CDbl(text(5)), 3)
            coneTot(i).p(0) = Math.Round(CDbl(text(6)), 3)
            coneTot(i).loi(0) = Math.Round(CDbl(text(7)), 3)
            coneTot(i).gol(0) = Math.Round(CDbl(text(8)), 3)
            coneTot(i).tonnage = Math.Round(CDbl(text(9)), 1)
            coneTot(i).bev = Math.Round(CDbl(text(10)), 1)
        Loop
        FileClose(1)
    Catch ex As Exception
        Console.WriteLine("Error: " & ex.Message)
    End Try
End Sub
For k = 1 To 20
  coneTot(i).value(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 1)), 1)
  coneTot(i).oreton(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 2)), 1)
  coneTot(i).wasteton(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 3)), 1)
  coneTot(i).fe(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 4)), 3)
  coneTot(i).sio2(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 5)), 3)
  coneTot(i).al2o3(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 6)), 3)
  coneTot(i).p(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 7)), 3)
  coneTot(i).loi(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 8)), 3)
  coneTot(i).gol(k) = Math.Round(CDbl(text(17 + (k - 1) * 9 + 9)), 3)
Next

j = 9 * (S + 1) '9: number of attributes of TopCones
While text(j) <> Nothing
  coneTot(i).nodeID(j1) = text(j)
  j1 = j1 + 1
  j = j + 1
  If j = text.Length Then
    Exit While
  End If
End While
Loop
FileClose(1)

'===================================================================
'2. Declare relevant variables and parameters for SIP model
'===================================================================
Dim totaldou(p) As Double
Dim totaldol(p) As Double
Dim totaldmu(p) As Double
Dim totaldml(p) As Double
Dim totaldfeu(p) As Double
Dim totaldfel(p) As Double
Dim totaldsio2u(p) As Double
Dim totaldsio2l(p) As Double
Dim totaldal2o3u(p) As Double
Dim totaldal2o3l(p) As Double
Dim totaldpu(p) As Double
Dim totaldpl(p) As Double
Dim totaldloiu(p) As Double
Dim totaldloil(p) As Double
Dim totaldgolu(p) As Double
Dim totaldgoll(p) As Double
Dim n5 As Long = i 'n5: number of TopCones
Dim n6 As Long = n5 * p 'n6: number of binary variables
Dim n7 As Long = 2 * 8 * p * S 'n7: number of linear variables
Console.WriteLine("number of TopCones: " & vbTab & n5)
Console.WriteLine("number of binary variables: " & vbTab & n6)
Dim cplex As New Cplex()
Dim var(1)() As INumVar
Dim rng(1)() As IRange
Dim lb(n6 + n7) As Double
Dim ub(n6 + n7) As Double
Dim varname(n6 + n7) As String
Dim xt(n6 + n7) As NumVarType
Dim objvals(n6 + n7) As Double
objvals(0) = 0.0

j = 1
For i = 1 To n5
    Dim j1 As Long = 0
    For k = j To j + p - 1
        j1 = j1 + 1
        varname(k) = "X" & j1 & "." & i
        objvals(k) = Math.Round((1 / ((1 + dr / 100) ^ j1)) * coneTot(i).value(0), 1)
        lb(k) = 0.0
        ub(k) = 1.0
        xt(k) = NumVarType.Bool
    Next
    j = j + p
Next
xt(0) = NumVarType.Bool

j = n6
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        For i3 As Long = 1 To 8
            For i4 As Long = 1 To 2
                j = j + 1
                lb(j) = 0
                ub(j) = System.Double.MaxValue
                varname(j) = "d" & "." & i1 & "." & i2 & "." & i3 & "." & i4
                xt(j) = NumVarType.Float
            Next
        Next
    Next
Next

Dim x As INumVar() = cplex.NumVarArray(n6 + n7 + 1, lb, ub, xt, varname)
var(0) = x

'=================================================================
'3. Declare objective funtion of the SIP model
'=================================================================
Console.WriteLine("Writing objective function ...")

Dim exprRisk As ILinearNumExpr = cplex.LinearNumExpr()
j = n6
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        For i3 As Long = 1 To 8
            For i4 As Long = 1 To 2
                j = j + 1
                If i4 = 1 Then 'excess
                    If i3 = 1 Then 'ore ton
                        exprRisk.AddTerm(Math.Round(Cou / ((1 + dr / 100) ^ i1), 4), x(j))
                    ElseIf i3 = 2 Then 'total ton
                        exprRisk.AddTerm(Math.Round(Cmu / ((1 + dr / 100) ^ i1), 4), x(j))
                    ElseIf i3 = 3 Then 'fe
exprRisk.AddTerm(Math.Round(Cfeu / ((1 + dr / 100) ^ i1), 4), x(j))
ElseIf i3 = 4 Then 'sio2
  exprRisk.AddTerm(Math.Round(Csio2u / ((1 + dr / 100) ^ i1), 4), x(j))
ElseIf i3 = 5 Then 'al2o3
  exprRisk.AddTerm(Math.Round(Cal2o3u / ((1 + dr / 100) ^ i1), 4), x(j))
ElseIf i3 = 6 Then 'p
  exprRisk.AddTerm(Math.Round(Cpu / ((1 + dr / 100) ^ i1), 4), x(j))
ElseIf i3 = 7 Then 'loi
  exprRisk.AddTerm(Math.Round(Cloiu / ((1 + dr / 100) ^ i1), 4), x(j))
ElseIf i3 = 8 Then 'gol
  exprRisk.AddTerm(Math.Round(Cgolu / ((1 + dr / 100) ^ i1), 4), x(j))
End If
Else 'deficient
  If i3 = 1 Then 'ore ton
    exprRisk.AddTerm(Math.Round(Col / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 2 Then 'total ton
    exprRisk.AddTerm(Math.Round(Cml / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 3 Then 'fe
    exprRisk.AddTerm(Math.Round(Cfel / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 4 Then 'sio2
    exprRisk.AddTerm(Math.Round(Csio2l / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 5 Then 'al2o3
    exprRisk.AddTerm(Math.Round(Cal2o3l / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 6 Then 'p
    exprRisk.AddTerm(Math.Round(Cpl / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 7 Then 'loi
    exprRisk.AddTerm(Math.Round(Cloi / ((1 + dr / 100) ^ i1), 4), x(j))
  ElseIf i3 = 8 Then 'gol
    exprRisk.AddTerm(Math.Round(Cgol / ((1 + dr / 100) ^ i1), 4), x(j))
End If
End If
Next
Next
Next
Next

'The objective function consists of two courses:
cplex.AddMaximize(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, exprRisk)))

=================================================================
4. Declare constraints of the SIP model
=================================================================

rng(0) = New IRange(n6 * 10) {}
k = 0

FileOpen(1, "precedence.csv", FileMode.Output) 'store precedence relationship of TopCones

'4.1. Sequencing constraints
For i = 1 To n5
  Dim k9 As Long = 0
  For j = 1 To i - 1
    Dim j1 As Long = 0
    While j1 <= coneTot(i).nodeID.Length - 1
      Dim j2 As Long = coneTot(j).nodeID.Length - 1
      While j2 > 0
        k9 = k9 + 1
        FileWrite(1, k9, j1, j2)
        j2 = j2 + 1
      End While
      j1 = j1 + 1
    End While
  Next
Next
Next
Next
Next
If block(coneTot(j).nodeID(j2)).Z - block(coneTot(i).nodeID(j1)).Z = 1 Then
    If Math.Round((block(coneTot(j).nodeID(j2)).coorX - block(coneTot(i).nodeID(j1)).coorX)^2 + (block(coneTot(j).nodeID(j2)).coorY - block(coneTot(i).nodeID(j1)).coorY)^2, 1) <= Math.Round((Math.Tan(sa * Math.PI / 180) * (block(coneTot(j).nodeID(j2)).coorZ - block(coneTot(i).nodeID(j1)).coorZ))^2, 1) Then
        k9 = k9 + 1
        coneTot(i).precedence(k9) = j
        Continue For
    End If
End If

If block(coneTot(j).nodeID(j2)).coorZ <= block(coneTot(i).nodeID(j1)).coorZ Then
    Exit While
End If

    Next
Next

coneTot(i).precedence = coneTot(i).precedence.Distinct().ToArray()
Array.Sort(coneTot(i).precedence)

For j = 1 To coneTot(i).precedence.Length - 2
    For j1 = j + 1 To coneTot(i).precedence.Length - 1
        For j2 As Long = 1 To coneTot(coneTot(i).precedence(j1)).precedence.Length - 1
            If coneTot(i).precedence(j) = coneTot(coneTot(i).precedence(j1)).precedence(j2) Then
                coneTot(i).precedence(j) = 0
            End If
        Next
    Next
Next

coneTot(i).precedence = coneTot(i).precedence.Distinct().ToArray()
Array.Sort(coneTot(i).precedence)

Dim text1 As String = ""
If coneTot(i).precedence.Length > 1 Then
    For j = 1 To coneTot(i).precedence.Length - 1
        text1 = text1 & coneTot(i).precedence(j) & ","
    Next
End If
PrintLine(1, text1)

'write sequencing constraints for cone i:
If coneTot(i).precedence.Length > 1 Then
    For j1 = 1 To p
        Dim expr As ILinearNumExpr = cplex.LinearNumExpr()
        expr.AddTerm(coneTot(i).precedence.Length - 1, x((i - 1) * p + j1))
        For j2 As Long = 1 To coneTot(i).precedence.Length - 1
            expr.AddTerm(-1.0, x((coneTot(i).precedence(j2) - 1) * p + j3))
        Next
    Next

Array.Sort(coneTot(i).precedence)

157
k = k + 1
rng(0)(k) = cplex.AddLe(expr, 0, "c" & k)
Next
End If
Next

FileClose(1)

'4.2. Blending constraints
'4.2.1. Upper bound constraints
j = n6
For i1 As Long = 1 To p
For i2 As Long = 1 To S
  'max blending constraints of Fe for each period of each simulation
  Array.Clear(objvals, 0, n6 + n7 + 1)  'reset objvals before each constraint
  For i = 1 To n5
    objvals((i - 1) * p + i1) = Math.Round((coneTot(i).fe(i2) - femax) * coneTot(i).oreton(i2), 0)
    Next
    k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 5))), 0, "c" & k)

  'max blending constraints of sio2 for each period of each simulation
  Array.Clear(objvals, 0, n6 + n7 + 1)
  For i = 1 To n5
    objvals((i - 1) * p + i1) = Math.Round((coneTot(i).sio2(i2) - sio2max) * coneTot(i).oreton(i2), 0)
    Next
    k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 7))), 0, "c" & k)

  'max blending constraints of al2o3 for each period of each simulation
  Array.Clear(objvals, 0, n6 + n7 + 1)
  For i = 1 To n5
    objvals((i - 1) * p + i1) = Math.Round((coneTot(i).al2o3(i2) - al2o3max) * coneTot(i).oreton(i2), 0)
    Next
    k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 9))), 0, "c" & k)

  'max blending constraints of p for each period of each simulation
  Array.Clear(objvals, 0, n6 + n7 + 1)
  For i = 1 To n5
    objvals((i - 1) * p + i1) = Math.Round((coneTot(i).p(i2) - pmax) * coneTot(i).oreton(i2), 0)
    Next
    k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 11))), 0, "c" & k)

  'max blending constraints of loi for each period of each simulation
  Array.Clear(objvals, 0, n6 + n7 + 1)
  For i = 1 To n5
    objvals((i - 1) * p + i1) = Math.Round((coneTot(i).loi(i2) - loimax) * coneTot(i).oreton(i2), 0)
    Next
    k = k + 1
  rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 13))), 0, "c" & k)

158
'max blending constraints of gol for each period of each simulation
Array.Clear(objvals, 0, n6 + n7 + 1)
For i = 1 To n5
    objvals((i - 1) * p + i1) = Math.Round((coneTot(i).gol(i2) - golmax) * coneTot(i).oreton(i2), 0)
Next
k = k + 1
rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 15))), 0, "c" & k)
j = j + 16
Next
Next

'4.2.2. Lower bound constraints
j = n6
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        'min blending constraints of Fe for each period of each simulation
        Array.Clear(objvals, 0, n6 + n7 + 1)
        For i = 1 To n5
            objvals((i - 1) * p + i1) = Math.Round((coneTot(i).fe(i2) - femin) * coneTot(i).oreton(i2), 0)
        Next
        k = k + 1
        rng(0)(k) = cplex.AddGe(cplex.Sum(cplex.ScalProd(x, objvals), x(j + 6)), 0, "c" & k)
    Next
j = j + 16
Next
Next

'4.3. Processing plant capacity constraints
'4.3.1. Upper bound constraints
j = n6  'The position of the first linear variable of d-parameter is n6 + 1
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        Array.Clear(objvals, 0, n6 + n7 + 1)  'reset objvals before each constraint
        For i = 1 To n5
            objvals((i - 1) * p + i1) = coneTot(i).oreton(i2)
        Next
        k = k + 1
        rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 15))), pcmax, "c" & k)
    Next
j = j + 16
Next
Next

'4.3.2. Lower bound constraints
j = n6  'The position of the first linear variable of d-parameter is n6 + 1
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        Array.Clear(objvals, 0, n6 + n7 + 1)  'reset objvals before each constraint
        For i = 1 To n5
            objvals((i - 1) * p + i1) = coneTot(i).oreton(i2)
        Next
        k = k + 1
    Next
k = k + 1
rng(0)(k) = cplex.AddGe(cplex.Sum(cplex.ScalProd(x, objvals), x(j + 2)), pcmin, "c" & k)
    j = j + 16
Next
Next

'4.4. Mining capacity constraints
'4.4.1. Upper bound constraints
j = n6
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        Array.Clear(objvals, 0, n6 + n7 + 1)
        For i = 1 To n5
            objvals((i - 1) * p + i1) = coneTot(i).oreton(i2) + coneTot(i).wasteton(i2)
        Next
        k = k + 1
    rng(0)(k) = cplex.AddLe(cplex.Sum(cplex.ScalProd(x, objvals), cplex.Prod(-1, x(j + 3))), mcmax, "c" & k)
    j = j + 16
Next
Next

'4.4.2. Lower bound constraints
j = n6
For i1 As Long = 1 To p
    For i2 As Long = 1 To S
        Array.Clear(objvals, 0, n6 + n7 + 1)
        For i = 1 To n5
            objvals((i - 1) * p + i1) = coneTot(i).oreton(i2) + coneTot(i).wasteton(i2)
        Next
        k = k + 1
    rng(0)(k) = cplex.AddGe(cplex.Sum(cplex.ScalProd(x, objvals), x(j + 4)), mcmin, "c" & k)
    j = j + 16
Next
Next

'4.5. Reserve constraints
For i = 1 To n5
    Dim expr As ILinearNumExpr = cplex.LinearNumExpr()
    For j = 1 To p
        expr.AddTerm(1, x((i - 1) * p + j))
    Next
    k = k + 1
    rng(0)(k) = cplex.AddLe(expr, 1, "c" & k)
Next

'=================================================================
'5. Solve the SIP model
'=================================================================
Console.WriteLine("Solving SIP model by CPLEX ...")

'Set optimality gap
cplex.SetParam(ILOG.CPLEX.DoubleParam.EpGap, mipgap)

Dim d(n7) As Double
If cplex.Solve() Then
'5.1. Print out solution status and value
    cplex.Output().WriteLine("solution status = " + cplex.GetStatus().ToString())
    cplex.Output().WriteLine("solution value = " + cplex.ObjValue.ToString())
    Console.WriteLine("Total TCs: " & n5)
    Console.WriteLine("gap: " & mipgap)

j = 0

'5.2. Read the integer variable value from the solution
    Dim a As Double() = cplex.GetValues(var(0)) 'var: integer variables
    Dim nvars As Integer = a.Length
    For j = 1 To nvars - n7 - 1
        If (j - 1) Mod p = 0 Then
            i = i + 1
        End If
        If Math.Round(a(j), 0) = 1 Then
            If j Mod p = 0 Then
                coneTot(i).schedule = p
            Else
                coneTot(i).schedule = j Mod p
            End If
        End If
    Next

'5.3. Read the linear variable value from the solution
    Else
        Console.WriteLine("Infeasible")
        Console.ReadKey()
    End If

    For i = 1 To n5
        j = 0
        While coneTot(i).nodeID(j) <> 0
            block(coneTot(i).nodeID(j)).schedule = coneTot(i).schedule
            j = j + 1
        End While
    Next

    Dim finalScheduleSim(S) As FinalSchedule
    For n = 0 To S
        finalScheduleSim(n) = New FinalSchedule
    Next

'=================================================================
'6. Calculate the scheduling outcomes for each scenario
'=================================================================

FileOpen(2, "d_parameter1.csv", OpenMode.Output)
For n = 1 To S
    Dim oreton(0 To p) As Double 'store total ore ton of period p
    Dim wasteton(0 To p) As Double 'store total waste ton of period p
    Dim bev(0 To p) As Double 'store total value of period p
    Dim gradefe(0 To p) As Double 'store average fe grade of period p
    Dim gradesio2(0 To p) As Double 'store average sio2 grade of period p
    Dim gradeal2o3(0 To p) As Double 'store average al2o3 grade of period p
    Dim gradep(0 To p) As Double 'store average p grade of period p
    Dim gradeloi(0 To p) As Double 'store average loi grade of period p
    Dim gradegol(0 To p) As Double 'store average gol grade of period p
    Dim sched_cone(0 To p) As Long 'count number of cones that are scheduled in a period

    For i = 1 To n5
        For j1 = 1 To p
            If coneTot(i).schedule = j1 Then
                sched_cone(j1) = sched_cone(j1) + 1
                oreton(j1) = oreton(j1) + coneTot(i).oreton(n)
                wasteton(j1) = wasteton(j1) + coneTot(i).wasteton(n)
                bev(j1) = bev(j1) + coneTot(i).value(n)
                gradefe(j1) = gradefe(j1) + coneTot(i).fe(n) * coneTot(i).oreton(n)
                gradesio2(j1) = gradesio2(j1) + coneTot(i).sio2(n) * coneTot(i).oreton(n)
                gradeal2o3(j1) = gradeal2o3(j1) + coneTot(i).al2o3(n) * coneTot(i).oreton(n)
                gradep(j1) = gradep(j1) + coneTot(i).p(n) * coneTot(i).oreton(n)
                gradeloi(j1) = gradeloi(j1) + coneTot(i).loi(n) * coneTot(i).oreton(n)
                gradegol(j1) = gradegol(j1) + coneTot(i).gol(n) * coneTot(i).oreton(n)
            End If
        Next
    Next

    For i = 1 To p
        gradefe(i) = gradefe(i) / oreton(i)
        gradesio2(i) = gradesio2(i) / oreton(i)
        gradeal2o3(i) = gradeal2o3(i) / oreton(i)
        gradep(i) = gradep(i) / oreton(i)
        gradeloi(i) = gradeloi(i) / oreton(i)
        gradegol(i) = gradegol(i) / oreton(i)
    Next

    Dim dou As Double = 0
    Dim dol As Double = 0
    Dim dmu As Double = 0
    Dim dml As Double = 0
    Dim dfel As Double = 0
    Dim dsio2u As Double = 0
    Dim dsio2l As Double = 0
    Dim dal2o3u As Double = 0
    Dim dal2o3l As Double = 0
    Dim dpu As Double = 0
    Dim dpl As Double = 0
    Dim dloiu As Double = 0
    Dim dloil As Double = 0
    Dim dgolu As Double = 0
    Dim dgoll As Double = 0

    If oreton(i) > pcmax Then
        totaldou(i) = totaldou(i) + oreton(i) - pcmax
        dou = oreton(i) - pcmax
    End If

End For
If oreton(i) < pcmin Then
  totaldol(i) = totaldol(i) + pcmin - oreton(i)
  dol = pcmin - oreton(i)
End If

If oreton(i) + wasteton(i) > mcmax Then
  totaldmu(i) = totaldmu(i) + oreton(i) + wasteton(i) - mcmax
  dmu = oreton(i) + wasteton(i) - mcmax
End If

If oreton(i) + wasteton(i) < mcmin Then
  totaldml(i) = totaldml(i) + mcmin - oreton(i) - wasteton(i)
  dml = mcmin - oreton(i) - wasteton(i)
End If

If gradefe(i) > femax Then
  totaldfeu(i) = totaldfeu(i) + (gradefe(i) - femax) * oreton(i)
  dfeu = (gradefe(i) - femax) * oreton(i)
End If

If gradefe(i) < femin Then
  totaldfel(i) = totaldfel(i) + (femin - gradefe(i)) * oreton(i)
  dfel = (femin - gradefe(i)) * oreton(i)
End If

If gradesio2(i) > sio2max Then
  totalsio2u(i) = totalsio2u(i) + (gradesio2(i) - sio2max) * oreton(i)
  dsio2u = (gradesio2(i) - sio2max) * oreton(i)
End If

If gradesio2(i) < sio2min Then
  totalsio2l(i) = totalsio2l(i) + (sio2min - gradesio2(i)) * oreton(i)
  dsio2l = (sio2min - gradesio2(i)) * oreton(i)
End If

If gradeal2o3(i) > al2o3max Then
  totaldal2o3u(i) = totaldal2o3u(i) + (gradeal2o3(i) - al2o3max) * oreton(i)
  dal2o3u = (gradeal2o3(i) - al2o3max) * oreton(i)
End If

If gradeal2o3(i) < al2o3min Then
  totaldal2o3l(i) = totaldal2o3l(i) + (al2o3min - gradeal2o3(i)) * oreton(i)
  dal2o3l = (al2o3min - gradeal2o3(i)) * oreton(i)
End If

If gradep(i) > pmax Then
  totaldpu(i) = totaldpu(i) + (gradep(i) - pmax) * oreton(i)
  dpu = (gradep(i) - pmax) * oreton(i)
End If

If gradep(i) < pmin Then
  totaldpl(i) = totaldpl(i) + (pmin - grade(i)) * oreton(i)
  dpl = (pmin - grade(i)) * oreton(i)
End If

If graduoi(i) > loimax Then
  totaldloiu(i) = totaldloiu(i) + (gradueloi(i) - loimax) * oreton(i)
  dloiu = (gradueloi(i) - loimax) * oreton(i)
End If

If graduoi(i) < loimin Then
  totaldloil(i) = totaldloil(i) + (loimin - grade(i)) * oreton(i)
  dloil = (loimin - grade(i)) * oreton(i)
End If

If gradegol(i) > golmax Then
  totaldgolu(i) = totaldgolu(i) + (gradegol(i) - golmax) * oreton(i)
  dgolu = (gradegol(i) - golmax) * oreton(i)
End If

163
If gradegol(i) < golmin
    totaldgoll(i) = totaldgoll(i) + (golmin - gradegol(i)) * oreton(i)
    dgoll = (golmin - gradegol(i)) * oreton(i)
End If

PrintLine(2, "excess ore in sim," & n & ",in period," & i & "," & dou)
PrintLine(2, "deficient ore in sim," & n & ",,in period," & i & ",," & dol)
PrintLine(2, "excess total ton in sim," & n & ",,in period," & i & ",," & dmu)
PrintLine(2, "deficient total ton in sim," & n & ",,in period," & i & ",," & dmu)
PrintLine(2, "excess Fe in sim," & n & ",,in period," & i & ",," & dfeu)
PrintLine(2, "deficient Fe in sim," & n & ",,in period," & i & ",," & dfeu)
PrintLine(2, "excess SiO2 sim," & n & ",,in period," & i & ",," & dsio2u)
PrintLine(2, "deficient SiO2 sim," & n & ",,in period," & i & ",," & dsio2l)
PrintLine(2, "excess ore Al2O3 sim," & n & ",,in period," & i & ",," & dal2o3u)
PrintLine(2, "deficient Al2O3 sim," & n & ",,in period," & i & ",," & dal2o3l)
PrintLine(2, "excess P in sim," & n & ",,in period," & i & ",," & dpu)
PrintLine(2, "deficient P in sim," & n & ",,in period," & i & ",," & dpu)
PrintLine(2, "excess LOI in sim," & n & ",,in period," & i & ",," & dloiu)
PrintLine(2, "deficient LOI in sim," & n & ",,in period," & i & ",," & dloiu)
PrintLine(2, "excess GOL in sim," & n & ",,in period," & i & ",," & dgolu)
PrintLine(2, "deficient GOL in sim," & n & ",,in period," & i & ",," & dgoll)
Next

FileOpen(1, "production" & n & ",.csv", OpenMode.Output)
Dim b1 As Double = 0 'store ore ton
Dim b2 As Double = 0 'store waste ton
Dim b3 As Double = 0 'store total ton
Dim b4 As Double = 0 'store total value
Dim b5 As Double = 0 'store total NPV
Dim bfe As Double = 0 'store average fe grade
Dim bsio2 As Double = 0 'store average sio2 grade
Dim bal2o3 As Double = 0 'store average al2o3 grade
Dim bp As Double = 0 'store average p grade
Dim bloi As Double = 0 'store average loi grade
Dim bgol As Double = 0 'store average gol grade

For j = 1 To p
    PrintLine(1, j & "," & Math.Round(gradefe(j), 3) & "," & Math.Round(gradesio2(j), 3) _
        & "," & Math.Round(gradeal2o3(j), 3) & "," & Math.Round(gradepe(j), 4) _
        & "," & oreton(j) & "," & wasteton(j) & "," & (oreton(j) + wasteton(j)) _
        & "," & bev(j) & "," & bev(j) * (1 / ((1 + dr / 100)^ j))
    b1 = b1 + oreton(j)
    b2 = b2 + wasteton(j)
    b3 = b3 + oreton(j) + wasteton(j)
    b4 = b4 + bev(j)
    b5 = b5 + bev(j) * (1 / ((1 + dr / 100)^ j))
    bfe = bfe + gradefe(j) * oreton(j)
    bsio2 = bsio2 + gradesio2(j) * oreton(j)
    bal2o3 = bal2o3 + gradeal2o3(j) * oreton(j)
    bp = bp + gradepe(j) * oreton(j)
    bloi = bloi + gradeloi(j) * oreton(j)
    bgol = bgol + gradegol(j) * oreton(j)
Next

164
bfe = Math.Round(bfe / b1, 3)
bvio2 = Math.Round(bsio2 / b1, 3)
bal2o3 = Math.Round(bal2o3 / b1, 3)
bp = Math.Round(bp / b1, 4)
bloi = Math.Round(bloi / b1, 3)
b gol = Math.Round(bgol / b1, 3)

PrintLine(1, "Total" & "," & Math.Round(bfe, 3) _
& "," & Math.Round(bsio2, 3) & "," & Math.Round(bp, 4) _
& "," & Math.Round(bloi, 3) & "," & Math.Round(bgol, 3) _
& "," & b1 & "," & b2 & "," & b3 & "," & b4 & "," & b5)

FileClose(1)

For i = 1 To p
    finalScheduleSim(n).oreton(i) = oreton(i)
    finalScheduleSim(n).totalton(i) = oreton(i) + wasteton(i)
    finalScheduleSim(n).fe(i) = gradefe(i)
    finalScheduleSim(n).sio2(i) = gradesio2(i)
    finalScheduleSim(n).al2o3(i) = gradeal2o3(i)
    finalScheduleSim(n).ph(i) = gradep(i)
    finalScheduleSim(n).loi(i) = gradeloi(i)
    finalScheduleSim(n).gol(i) = gradegol(i)
Next

finalScheduleSim(n).oreton(0) = b1
finalScheduleSim(n).totalton(0) = b3
finalScheduleSim(n).fe(0) = gradefe(i)
finalScheduleSim(n).sio2(0) = gradesio2(i)
finalScheduleSim(n).al2o3(0) = gradeal2o3(i)
finalScheduleSim(n).ph(0) = gradep(i)
finalScheduleSim(n).loi(0) = gradeloi(i)
finalScheduleSim(n).gol(0) = gradegol(i)

Next

FileClose(2)

'=================================================================
'7. Calculate scheduling outcome of the average scenario
'=================================================================

'7.1. Calculate scheduling targets of each period

For n = 0 To 0
    Dim oreton(0 To p) As Double  'store total ore ton of period p
    Dim wasteton(0 To p) As Double  'store total waste ton of period p
    Dim bev(0 To p) As Double  'store total value of period p
    Dim gradefe(0 To p) As Double  'store average fe grade of period p
    Dim gradesio2(0 To p) As Double  'store average sio2 grade of period p
    Dim gradeal2o3(0 To p) As Double  'store average al2o3 grade of period p
    Dim gradep(0 To p) As Double  'store average ph grade of period p
    Dim gradeloi(0 To p) As Double  'store average loi grade of period p
    Dim gradegol(0 To p) As Double  'store average gol grade of period p

    For i = 1 To n5
        For j1 = 1 To p
            If coneTot(i).schedule = j1 Then
            '...
Dim oreton(j1) = oreton(j1) + coneTot(i).oreton(n)
wasteton(j1) = wasteton(j1) + coneTot(i).wasteton(n)
bev(j1) = bev(j1) + coneTot(i).bev(n)
gradef(j1) = gradef(j1) + coneTot(i).fe(n) * coneTot(i).oreton(n)
gradesio2(j1) = gradesio2(j1) + coneTot(i).sio2(n) * coneTot(i).oreton(n)
gradeal2o3(j1) = gradeal2o3(j1) + coneTot(i).al2o3(n) * coneTot(i).oreton(n)
gradep(j1) = gradep(j1) + coneTot(i).p(n) * coneTot(i).oreton(n)
gradeloi(j1) = gradeloi(j1) + coneTot(i).loi(n) * coneTot(i).oreton(n)
gradegol(j1) = gradegol(j1) + coneTot(i).gol(n) * coneTot(i).oreton(n)

End If
Next
Next
For i = 1 To p
  gradef(i) = gradef(i) / oreton(i)
gradesio2(i) = gradesio2(i) / oreton(i)
gradeal2o3(i) = gradeal2o3(i) / oreton(i)
grade(i) = grade(i) / oreton(i)
gradeloi(i) = gradeloi(i) / oreton(i)
gradegol(i) = gradegol(i) / oreton(i)
Next

FileOpen(1, "production.csv", OpenMode.Output)

Dim b1 As Double = 0  'store ore ton
Dim b2 As Double = 0  'store waste ton
Dim b3 As Double = 0  'store total ton
Dim b4 As Double = 0  'store total value
Dim b5 As Double = 0  'store total NPV
Dim bfe As Double = 0  'store average fe grade
Dim bsio2 As Double = 0  'store average sio2 grade
Dim bal2o3 As Double = 0  'store average al2o3 grade
Dim bp As Double = 0  'store average p grade
Dim bloi As Double = 0  'store average loi grade
Dim bgol As Double = 0  'store average gol grade
Dim TotalFinalNPV As Double = 0
Dim Totald_costValue As Double = 0  'store total d_cost over all periods
Dim totalo_cost As Double = 0
Dim totalm_cost As Double = 0
Dim totalfe_cost As Double = 0
Dim totalsio2_cost As Double = 0
Dim totalal2o3_cost As Double = 0
Dim totalp_cost As Double = 0
Dim totalloi_cost As Double = 0
Dim totalgol_cost As Double = 0

For j = 1 To p
  Dim finalNPV(p) As Double
  Dim d_costValue As Double = -1 * (totaldou(j) * Cou + totaldol(j) * Col + totaldmu(j) * Cmu _
    + totaldml(j) * Cml + totaldefe(j) * Cfeu + totaldfeu(j) * Cfel _
    + totalsio2u(j) * Csio2u + totalsio2l(j) * Csio2l _
    + totalsio1u(j) * Csio1u + totalsio1l(j) * Csio1l)
End For
Totald_costValue = Totald_costValue + d_costValue

total_cost = totalo_cost - totaldou(j) + Cou - totald(j) + Col

totalm_cost = totalm_cost - totaldmu(j) + Cmu - totaldml(j) + Cml

totalfse_cost = totalfse_cost - totaldfeu(j) + Cfeu - totaldfe(j) + Cfel

totalal2o3_cost = totalal2o3_cost - totaldal2o3u(j) + Cal2o3u - totaldal2o3l(j) + Cal2o3l

totalp_cost = totalp_cost - totaldpd(j) + Cpu - totaldp(j) + Cpl

totalgol_cost = totalgol_cost - totaldgoll(j) + Cgolu - totaldgoll(j) + Cgoll

finalNPV(j) = (bev(j) + d_costValue) * (1 / ((1 + dr / 100) ^ j))

TotalFinalNPV = TotalFinalNPV + finalNPV(j)


b1 = b1 + oreton(j)
b2 = b2 + wasteton(j)
b3 = b3 + oreton(j) + wasteton(j)
b4 = b4 + bev(j)
b5 = b5 + bev(j) * (1 / ((1 + dr / 100) ^ j))

bfe = bfe + gradeal2o3(j) * oreton(j)
bsio2 = bsio2 + gradesio2(j) * oreton(j)
bal2o3 = bal2o3 + gradeal2o3(j) * oreton(j)
bp = bp + gradeal2o3(j) * oreton(j)
blloi = blloi + gradeloil(j) * oreton(j)
bgol = bgol + gradegol(j) * oreton(j)

Next

bfe = Math.Round(bfe / b1, 3)
bsio2 = Math.Round(bsio2 / b1, 3)
bal2o3 = Math.Round(bal2o3 / b1, 3)
bp = Math.Round(bp / b1, 4)
blloi = Math.Round(bloi / b1, 3)
bgol = Math.Round(bgol / b1, 3)

PrintLine(1, "Total" & "," & Math.Round(bfe, 3) & "," & Math.Round(bsio2, 3) & "")

FileClose(1)
Next

Catch ex As Exception

End Try

End Sub

Public Sub PrintScheduleSIP()
' The subroutine prints out the scheduling result into a csv file, ' which is readable by any mine design software, such as Surpac

FileOpen(1, "scheduleTCA.csv", OpenMode.Input)
FileOpen(2, "scheduleSIP.csv", OpenMode.Output)
Dim textline As String
Dim i As Long = 0
Do Until EOF(1)
    i = i + 1
    textline = LineInput(1)
    PrintLine(2, textline & "," & block(i).schedule)
Loop

FileClose(2)
FileClose(1)

End Sub

End Module