

https://dspace.org/

A dynamic long-term and medium-term integrated open-pit mine production scheduling system based on the genetic algorithm

Muke, Pathy Musema

2025

University of the Witwatersrand, Johannesburg

https://hdl.handle.net/10539/50048

Muke, Pathy Musema. (2025). A dynamic long-term and medium-term integrated open-pit mine production scheduling system based on the genetic algorithm. [PhD thesis, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/50048

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

A DYNAMIC LONG-TERM AND MEDIUM-TERM INTEGRATED OPEN-PIT MINE PRODUCTION SCHEDULING SYSTEM BASED ON THE GENETIC ALGORITHM

Pathy Musema Muke

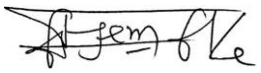
A thesis submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering.

Johannesburg, July 2025

DECLARATION

I declare that this is my own unaided work. It is being submitted for the degree of Doctor of Philosophy at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any university.

I have familiarised myself with the University of the Witwatersrand Student Academic Misconduct Policy (SAMP). I hereby declare that the work I am submitting for assessment is my original work. I affirm that I have not plagiarised, misrepresented, or colluded with any other person or source. I have properly acknowledged, cited, and referenced all the sources that I have used in my work. I also declare that where I have used artificial intelligence (AI) tools, it has been only as a means of guidance. The ideas and arguments presented in my work are my original thoughts and interpretations. I acknowledge that any academic misconduct, such as plagiarism or collusion will result in serious consequences. These consequences may include a reduced grade, a fail mark, or disciplinary action according to the SAMP. The Faculty of Engineering and the University of the Witwatersrand have my consent to investigate any potential academic misconduct in my work. This investigation may include the use of similarly and AI detection software to check the originality of my work. I undertake to abide by the decision of any such investigation.



Pathy Musema Muke

This 16th day of July year 2025

ABSTRACT

Open-pit mine production scheduling (OPMPS) is systematically divided into long-term (LT), medium-term (MT) and short-term (ST) scheduling. There is a clear interdependence between these scheduling horizons. An integrated optimisation approach to these scheduling horizons is essential since the horizons can be linked together to ensure spatial and temporal scheduling alignment. The linkages include feedback loops to enable dynamic updates of the scheduling system to incorporate changes whenever they occur.

Traditionally, optimisation techniques have often focussed on solving LT, MT and ST scheduling horizons independently from each other, despite there being some shortcomings associated with the independent optimisation of LT, MT and ST production schedules. Current studies have not adequately presented an integrated scheduling system in which all horizons are incorporated and correlated to mitigate scheduling misalignment. Generating independent mine production schedules is known to be detrimental to a project's net present value (NPV) due to scheduling misalignment and inconsistency. This raises the question of whether a solution approach can be developed to dynamically integrate scheduling horizons to overcome challenges associated with spatial and temporal misalignment and inconsistency in scheduling. This thesis developed a new approach to dynamically integrate LT and MT production scheduling to improve temporal and spatial alignment while attempting to maximise NPV as opposed to when these scheduling horizons are optimised independently. The reason for integrating LT and MT production schedules is to provide a framework to support ST scheduling which introduces more complexity and variability to account for operational planning.

Firstly, the thesis expressed the scheduling problem as a mixed integer programming (MIP) problem to integrate LT and MT production scheduling horizons. The MIP model was employed in this thesis because it can formulate practical extraction sequences and incorporates continuous variables, compared to linear programming (LP) and integer programming (IP) models. Then, the thesis developed an integrated solution approach based on the genetic algorithm (GA), which is a metaheuristic algorithm, to solve the integrated LT and MT MIP scheduling problem. GA was selected because it can provide solutions to problems that are unattainable with exact or deterministic techniques. GA can also incorporate geological, economic and operational uncertainties, although in this thesis

the focus was on operational uncertainty. Since GA is a stochastic technique, the combined MIP model and GA approach generated stochastic solutions that accounted for constraints and uncertainties associated with mining rate, processing rate and ore grade target. The combined MIP model and GA approach was coded in Python programming language and validated by applying it to a block model downloaded from Geovia Surpac® software. Results show that when the integrated LT and MT production scheduling system was solved using the combined MIP model and GA approach, NPV improved by 2.60% compared to when the scheduling horizons were independently optimised. In addition, spatial and temporal alignment between the LT and MT production schedules was achieved. Therefore, the thesis demonstrated that a combined MIP model and GA approach can dynamically integrate LT and MT production scheduling horizons to improve NPV as opposed to when these scheduling horizons are independently optimised.

The combined MIP model and GA approach was also applied to the Newman and Zuck Small block models downloaded from MineLib, to validate the solution approach. The results were compared to some of the MineLib existing solutions obtained by applying the TopoSort heuristic algorithm to solve a LP relaxation model of the scheduling optimisation problem. The integrated LT and MT scheduling system solved using the combined MIP model and GA approach had a 10.86% higher and a 4.86% lower NPV compared to the TopoSort algorithm when applied to the Zuck Small and Newman block models, respectively. Although the aim was to improve NPV, a lower NPV was generated during the validation using the Newman block model. This is because GA can generate a different outcome each time it is run on the block model, due to its stochastic solution process. Another run of the combined MIP model and GA approach on the same block model with the same inputs may generate a higher NPV. However, in both instances, the combined MIP model and GA approach delivered an integrated production schedule that ensured that spatial and temporal alignment was achieved between LT and MT production schedules. Future research may focus on developing solution approaches that can fully integrate LT, MT and ST production scheduling horizons, and incorporate factors like stockpiling and blending to further improve scheduling effectiveness. Further research could also investigate the fine-tuning of GA parameters to ensure that the application of the combined MIP model and GA approach can consistently generate higher NPVs.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my God for His endless blessings, guidance, and strength throughout my academic journey. Without His grace and support, this PhD research would not have been possible. Additionally, different organisations and individuals contributed towards the completion of the thesis, and I would like to extend the following special acknowledgements to them:

- I would like to extend my sincere gratitude to my supervisors Dr Tinashe Tholana, Prof Cuthbert Musingwini and Prof Montaz Ali, for their invaluable guidance, continuous support, and patience during my research. Their immense knowledge and expertise have been crucial to the completion of this thesis.
- I am so grateful to Prof Cuthbert Musingwini, the former Head of the School of Mining Engineering, for initiating this PhD research study and for arranging the financial support through the School's CORY Scholarship Fund.
- I would like to thank Mr David Osborn, Dassault Systèmes Geovia Africa Country Manager and Mr Ralf Smith, Dassault Systèmes Geovia Global Sales Director, for also providing the financial support and resources necessary to conduct my research.
- I would also like to thank Prof Flory Kiseya, the former Director of Institut Supérieur des Techniques Appliquées (ISTA) in Kolwezi, Democratic Republic of Congo (DRC) for allowing me to pursue this research and for also providing financial support.
- I am thankful to my friend Mr Robert Nkateko for his undeniable assistance during the coding sessions in Python programming language.
- I would like to acknowledge the academic staff and members of the School of Mining Engineering at the University of the Witwatersrand, for providing a supportive and stimulating environment for my research.
- I wish to extend my appreciation to my colleagues at Dassault Systèmes for their insightful comments, encouragement, and constructive criticism, which have greatly improved the quality of this work.
- Special thanks to my family, for their unconditional love, encouragement, and patience throughout this journey. Their support has been my greatest strength.

PUBLICATIONS ARISING OUT OF THE THESIS

The following peer-reviewed journal and conference publications emanated from this research study:

1. Muke, P., Tholana, T., Musingwini, C., and Ali, M. (2025a). A genetic algorithm for temporal and spatial alignment of long- and medium-term mine production scheduling for open-pit mines. *Resources Policy*, Vol. 106, p. 105629. **Published**. https://doi.org/10.1016/j.resourpol.2025.105629
2. Muke, P., Tholana, T., Musingwini, C., Ali, M. and Mutandwa, B. (2025b). Towards a system for integrating open-pit mine production scheduling stages. *Application of Computers and Operations Research in the Minerals Industry (*APCOM) *Conference*, 10-13 August 2025, Perth, Australia. **Accepted**.

TABLE OF CONTENTS

DECLARATION i

ABSTRACT ii

ACKNOWLEDGEMENTS iv

PUBLICATIONS ARISING OUT OF THE THESIS v

LIST OF FIGURES x

LIST OF TABLES xiv

LIST OF ACRONYMS AND ABBREVIATIONS xvi

1. INTRODUCTION 1

1.1 Overview of Chapter 1 1

1.2 Open-pit mine design and planning 1

1.3 Problem statement and research question 5

1.4 Research motivation and significance 8

1.5 Research aim and objectives 10

1.6 Research scope 10

1.7 Structure of the thesis 12

2. OPEN-PIT MINE PRODUCTION SCHEDULING SYSTEM OPTIMISATION TECHNIQUES 13

2.1 Overview of Chapter 2 13

2.2 Open-pit mine production scheduling 13

2.2.1 Long-term mine production scheduling 13

2.2.2 Medium-term mine production scheduling 15

2.2.3 Short-term mine production scheduling 18

2.3 Open-pit mine production scheduling optimisation techniques 23

2.3.1 Exact open-pit mine production scheduling optimisation techniques 26

2.3.2 Metaheuristic open-pit mine production scheduling optimisation techniques 40

2.4 Genetic algorithm 50

2.4.1 Origin of the genetic algorithm 50
2.4.2 Concept of the genetic algorithm 51
2.4.3 Optimisation procedure of the genetic algorithm 52
2.4.4 Exploitation and exploration search 54
2.4.5 Selection operator 56
2.4.6 Crossover operator 56
2.4.7 Mutation operator 60
2.4.8 Genetic algorithm problem representation 64
2.4.9 Genetic algorithm handling of constraints 65
2.5 Summary of Chapter 2 66
3. MIXED INTEGER PROGRAMMING MODELLING OF THE INTEGRATED LONG-TERM AND MEDIUM-TERM MINE PRODUCTION SCHEDULING SYSTEM 68
3.1 Overview of Chapter 3 68
3.2 Modelling of the mixed integer programming mine production scheduling system 68
3.2.1 Mathematical notation of the mixed integer programming mine production scheduling model 69
3.2.2 Mixed integer programming of long-term and medium-term mine production scheduling 72
3.3 Integrating long-term and medium-term mine production scheduling 79
3.3.1 A framework for a dynamic integrated long-term and medium-term mine production scheduling 79
3.3.2 Objective function integrating long-term and medium-term production scheduling of the mixed integer programming model 81
3.3.3 Dynamic integrated long-term and medium-term scheduling constraints 83
3.4 Reconciliation in mine production scheduling 88
3.4.1 Spatial compliance of the mine production scheduling system 89
3.4.2 Temporal compliance of the mine production scheduling system 90
3.5 Summary of Chapter 3 93

4. GA METHODOLOGY FOR SOLVING THE MIP MODEL THAT DYNAMICALLY INTEGRATES THE LT-MT MINE PRODUCTION SCHEDULING SYSTEM 95
4.1 Overview of Chapter 4 95
4.2 Motivation for using a GA methodology 95
4.3 Implementation of GA for dynamically integrating long-term and medium-term production scheduling 97
4.4 Python programming language 101
4.5 Summary of Chapter 4 103
5. OPEN-PIT METAL MINING CASE STUDIES FOR THE APPLICATION OF DYNAMICALLY INTEGRATED LONG-TERM-MEDIUM-TERM MINE PRODUCTION SCHEDULING SYSTEM 105
5.1 Overview of Chapter 5 105
5.2 Data sources 105
5.3 Description of block models and GA input parameters 106
5.4 Optimisation of long-term mine production scheduling on the Surpac block model 109
5.4.1 Discussion of GA parameters for long-term mine production scheduling 110
5.4.2 Discussion of long-term mine production scheduling results 116
5.5 Optimisation of medium-term mine production scheduling on the Surpac block model 119
5.5.1 Discussion of GA parameters results for medium-term mine production scheduling 119
5.5.2 Discussion of medium-term mine production scheduling results 125
5.6 Integrated long-term and medium-term mine production scheduling optimisation 132
5.6.1 Discussion of the GA parameters results for the integrated long-term and medium-term production scheduling 132
5.6.2 Discussion of the integrated long-term and medium-term mine production scheduling results 134

5.6.3 Sensitivity analysis 141
5.7 Comparison of the results between the combined mixed integer programming model and GA approach to existing techniques applied to the Newman and Zuck Small block models 143
5.8 Summary of Chapter 5 148
6. CONCLUSIONS AND RECOMMENDATIONS 150
6.1 Conclusions 150
6.2 Contribution to knowledge 151
6.3 Recommendations 151
6.4 Suggestions for future research work 152
7. REFERENCES 154
8. APPENDIX 166
8.1 Python code representing the combined mixed integer programming model and GA approach integrating LT and MT production schedules 166
8.2 Python code to input data and run the combined mixed integer programming model and GA approach for integrating long-term and medium-term production schedules 188

LIST OF FIGURES

Figure 1.1: A generic cross-sectional view of an open-pit mine design 3
Figure 1.2: Open-pit mine production scheduling system highlighting the link between long-term, medium-term and short-term production schedules 6
Figure 2.1: A typical activity diagram for long-term production scheduling 14
Figure 2.2: A typical planning activity diagram for medium-term six-month production scheduling 17
Figure 2.3: A typical planning activity diagram for monthly short-term production scheduling 19
Figure 2.4: A typical planning activity diagram for weekly short-term production scheduling 21
Figure 2.5: A typical planning activity diagram for daily short-term production scheduling 22
Figure 2.6: Classification of classical optimisation methods.....25
Figure 2.7: General structure of the genetic algorithm 54
Figure 2.8: Mapping from chromosomes to feasible solutions in a solution space 55
Figure 2.9: Illustration of a single-point crossover genetic procedure.....57
Figure 2.10: Illustration of a partial-mapped crossover genetic procedure 59
Figure 2.11: Illustration of an order crossover genetic procedure 60
Figure 2.12: Illustration of an inversion mutation genetic procedure.....61
Figure 2.13: Illustration of an insertion mutation genetic procedure.....62
Figure 2.14: Illustration of the swap mutation genetic procedure 62
Figure 2.15: Illustration of a heuristic mutation genetic procedure.....63
Figure 3.1: Graphical views of block predecessors, 1:5 pattern (a) and 1:9 pattern (b) ...77
Figure 3.2: A framework for a dynamic integrated long-term and medium-term mine production scheduling model 80
Figure 4.1: Framework of the GA methodology for solving the MIP model 100
Figure 5.1: Spatial view of the Newman (left) model with grade variation and Zuck Small (right) model with ore tonnage variation per block 106
Figure 5.2: Spatial view of Surpac block model showing ore (1) and waste (0) blocks. 107
Figure 5.3: Feasible solution space for the optimisation of the LT production schedule in Period 1 110

Figure 5.4: Feasible solution space for the optimisation of the LT production schedule in Period 2 111
Figure 5.5: Feasible solution space for the optimisation of the LT production schedule in Period 3 111
Figure 5.6: Feasible solution space for the optimisation of the LT production schedule in Period 4 111
Figure 5.7: Feasible solution space for the optimisation of the LT production schedule in Period 5 112
Figure 5.8: Feasible solution space for the optimisation of the LT production schedule in Period 6 112
Figure 5.9: Feasible solution space for the optimisation of the LT production schedule in Period 7 112
Figure 5.10: Feasible solution space for the optimisation of the LT production schedule in Period 8 113
Figure 5.11: Feasible solution space for the optimisation of the LT production schedule in Period 9 113
Figure 5.12: Average fitness value of the first and the last generations for each period of the LT production schedule 114
Figure 5.13: Fitness value of the best chromosome per period for the optimisation of the LT production schedule 115
Figure 5.14: A 3D view of the LT production schedule for the Surpac block model 116
Figure 5.15: Total tonnage and the cash flow value in each period of the LT production schedule 118
Figure 5.16: Ore and waste tonnages, stripping ratio and average grade in each period of the LT production schedule 118
Figure 5.17: Feasible solution space for the optimisation of the MT production schedule in Period 1 120
Figure 5.18: Feasible solution space for the optimisation of the MT production schedule in Period 2 120
Figure 5.19: Feasible solution space for the optimisation of the MT production schedule in Period 3 120

Figure 5.20: Feasible solution space for the optimisation of the MT production schedule in Period 4 121
Figure 5.21: Feasible solution space for the optimisation of the MT production schedule in Period 5 121
Figure 5.22: Feasible solution space for the optimisation of the MT production schedule in Period 6 121
Figure 5.23: Feasible solution space for the optimisation of the MT production schedule in Period 7 122
Figure 5.24: Feasible solution space for the optimisation of the MT production schedule in Period 8 122
Figure 5.25: Feasible solution space for the optimisation of the MT production schedule in Period 9 122
Figure 5.26: Feasible solution space for the optimisation of the MT production schedule in Period 10 123
Figure 5.27: Average fitness value of the first and the last generations for each period of the MT production schedule 124
Figure 5.28: Fitness value of the best chromosome per period for the optimisation of the MT production schedule 125
Figure 5.29: A 3D view of the MT production schedule for the Surpac block model 126
Figure 5.30: Total tonnage and the cash flow value in each period of the MT production schedule 128
Figure 5.31: Ore and waste tonnages, stripping ratio and average grade in each period of the MT production schedule 128
Figure 5.32: 4250E cross-sections spatial compliance between LT and MT production scheduling independently optimised 131
Figure 5.33: Average fitness value of the first and the last generations for each period of the integrated LT and MT production schedule 133
Figure 5.34: Fitness value of the best chromosome per period for the optimisation of the integrated LT and MT production schedule 134
Figure 5.35: A 3D view of the integrated LT and MT production schedule for the Surpac block model 135

Figure 5.36: 4250E cross-sections spatial compliance between LT and MT production scheduling integrally optimised 140
Figure 5.37: Total tonnage achieved for the integrated LT and MT production schedule 141
Figure 5.38: Ore tonnage achieved for the integrated LT and MT production schedule 141
Figure 5.39: NPV sensitivity results 142
Figure 5.40: LOM sensitivity results 143

LIST OF TABLES

Table 2.1: Summary of exact techniques applied in solving the production scheduling problem for open-pit mines 34
Table 2.2: Summary of metaheuristic and stochastic methods applied in solving the production scheduling problem of open-pit mines 43
Table 3.1: List of indices used in the MIP mine production scheduling model 70
Table 3.2: Sets of entities used in the MIP mine production scheduling model 70
Table 3.3: Parameters used in the MIP mine production scheduling model 70
Table 3.4: LT and MT mine scheduling decision variables of the mixed integer programming model 73
Table 3.5: MIP models for independently optimising LT and MT mine production scheduling 75
Table 3.6: Description of spatial production compliance categories for an integrated mine production scheduling system 90
Table 4.1: The pseudocode algorithm of the GA methodology for solving the MIP model 102
Table 4.2: Computer specifications 103
Table 5.1: Description of input parameters for the Newman, Zuck Small and Surpac block models 108
Table 5.2: Description of input parameters for the GA methodology 109
Table 5.3: LT production scheduling results of the Surpac block model 117
Table 5.4: MT production scheduling results of the Surpac block model 127
Table 5.5: Scheduling compliance of the isolated LT and MT production scheduling results 130
Table 5.6: Results of the combined MIP model and GA approach when applied on the Surpac block model 136
Table 5.7: Scheduling compliance of the integrated LT and MT production schedule 139
Table 5.8: Results of the combined MIP model and GA approach when applied on the Newman block model 144
Table 5.9: Results of the combined MIP model and GA approach when applied on the Zuck Small block model 146

Table 5.10: Comparison of the combined MIP model and GA approach and TopoSort algorithm results.....147

LIST OF ACRONYMS AND ABBREVIATIONS

2D	Two Dimension
3D	Three Dimension
AA	Approximation Algorithm
ACO	Ant Colony Optimisation
AI	Artificial Intelligence
BEV	Block Economic Value
CP	Constraint Programming
CPIT	Constrained Pit Limit
DP	Dynamic Programming
GA	Genetic Algorithm
IDE	Integrated Development Environment
IP	Integer Programming
LOM	Life of Mine
LP	Linear Programming
LT	Long-Term
MIP	Mixed Integer Programming
MT	Medium-Term
NIA	Nature-inspired algorithm
NPV	Net Present Value
OPMPS	Open-Pit Mine Production Scheduling
OR	Operations Research
OX	Order Crossover
PCPSP	Precedence Constraints Production Scheduling Problem
PMX	Partial-Mapped Crossover
PSH	Problem-Specific Heuristics
PSO	Particle Swarm Optimisation
RL	Reinforcement Learning
SA	Simulated Annealing
SAMREC	South African Code for the Reporting of Exploration Results, Mineral Resources and Mineral Reserves
SMIP	Stochastic Mixed Integer Programming
SPX	Single-Point Crossover
ST	Short-Term
UPL	Ultimate Pit Limit
$	United States Dollar

1. INTRODUCTION

1.1 Overview of Chapter 1

The purpose of this chapter is to introduce the thesis. The chapter provides the background of the thesis, the research problem and research question, and the research aim and objectives. The chapter also provides the motivation for conducting this research study and significance of the thesis. It outlines the scope of the research, and lastly, presents the structure of the thesis.

Section 1.2 provides the background of the research study, describing the open-pit mine planning process. It emphasises the importance of initially optimising and designing the open pit from which the mine production scheduling process is carried out. It then briefly defines the three mine production scheduling horizons, namely LT, MT, and ST production scheduling. Section 1.3 discusses the problem statement, which is the disintegrated or independent optimisation of the mine production scheduling horizons. Consequences of independently optimising the production scheduling horizons are outlined. This section also explains why the genetic algorithm (GA), which is a metaheuristic optimisation technique, was selected to optimise the integrated mine production scheduling system in this thesis. It ends with a research question.

Section 1.4 presents the research motivation for optimising the production scheduling system by integrating LT and MT production scheduling to complement the work done by Levinson and Dimitrakopoulos (2023). The motivation for using metaheuristic techniques over exact techniques is also presented. The section also presents the significance of the research and highlights the knowledge gap in integrating mine production scheduling horizons from LT to MT production scheduling. Section 1.5 presents the aim and objectives of the thesis. Section 1.6 describes the scope of the research and explains why it is important to integrate LT and MT mine production scheduling. Finally, Section 1.7 provides a summary of the structure and layout of the thesis.

1.2 Open-pit mine design and planning

Mining plays a vital role in the economic development of many countries, providing essential resources that power industries and economies worldwide (Hartman and Mutmansky, 2002). At the heart of the mining process lies mine planning, a critical step that determines both the efficiency and profitability of a mine (Hustrulid *et al*., 2013). In

open-pit mining, mine planning aims to deliver a pit design and a mining sequence of an orebody (Tholana *et al*., 2019). Therefore, the mine planning process consists of two main components namely mine designing and mine production scheduling. The starting point of mine designing and mine production scheduling is geological block modelling, a process which models the three-dimensional (3D) shape of an orebody and its surrounding rockmass. Geological block modelling is an essential process that enables the visualisation and understanding of the lithological, structural, geochemical, and geophysical data of the orebody (Fallara *et al*., 2006). The result of the process is a geological block model also called a Mineral Resource model. According to the South African Code for the Reporting of Exploration Results, Mineral Resources and Mineral Reserves (SAMREC Code, 2016:18), a Mineral Resource is "a concentration or occurrence of solid material of economic interest in or on the Earth's crust in such form, grade or quality and quantity that there are reasonable prospects for eventual economic extraction. The location, quantity, grade, continuity and other geological characteristics of a Mineral Resource are known, estimated or interpreted from specific geological evidence and knowledge, including sampling. Mineral Resources are subdivided, and must be so reported, in order of increasing confidence in respect of geoscientific evidence, into Inferred, Indicated or Measured categories."

The geological block model is then converted into an economic block model by applying revenue, cost and recovery factors. From the economic block model, a Mineral Reserve model is obtained by applying modifying factors such as consideration of mining, metallurgical, economic, marketing, legal, environmental, social and governmental factors (SAMREC, 2016). According to the SAMREC Code (2016:25), a Mineral Reserve is "the economically mineable part of a Measured and/or Indicated Mineral Resource. It includes diluting materials and allowances for losses, which may occur when the material is mined or extracted and is defined by studies at PreFeasibility or Feasibility level as appropriate that include application of Modifying Factors. Such studies demonstrate that, at the time of reporting, extraction could reasonably be justified. The reference point at which Mineral Reserves are defined, usually the point where the ore is delivered to the processing plant, must be stated. It is important that, in all situations where the reference point is different, such as for a saleable product, a clarifying statement is included to ensure that the reader is fully informed as to what is being reported."

Economic block modelling calculates the block economic value (BEV) of blocks within the block model. BEV is the difference between the revenue of a block and the costs associated with its extraction and processing. Using the economic block model, open-pit optimisation is done. The open-pit optimisation process determines economic pit limits, also called the ultimate pit limit (UPL). The UPL maximises the undiscounted economic value of an orebody subject to constraints such as geotechnical, economic and geological constraints (Askari-Nasab and Awuah-Offei, 2009). Mining in open-pit mining operations is often conducted on benches within pushbacks. Pushbacks are a strategic way to divide the ultimate pit into manageable phases that can target high grade ore early to improve NPV and allow an optimum schedule (Hustrulid *et al.*, 2013). The design of the pushbacks and final pit is then done within the UPL. Aspects of the pit such as slope angle, benches, and haul road among others are designed. Figure 1.1 illustrates a generic cross-section view of an open-pit mine design.

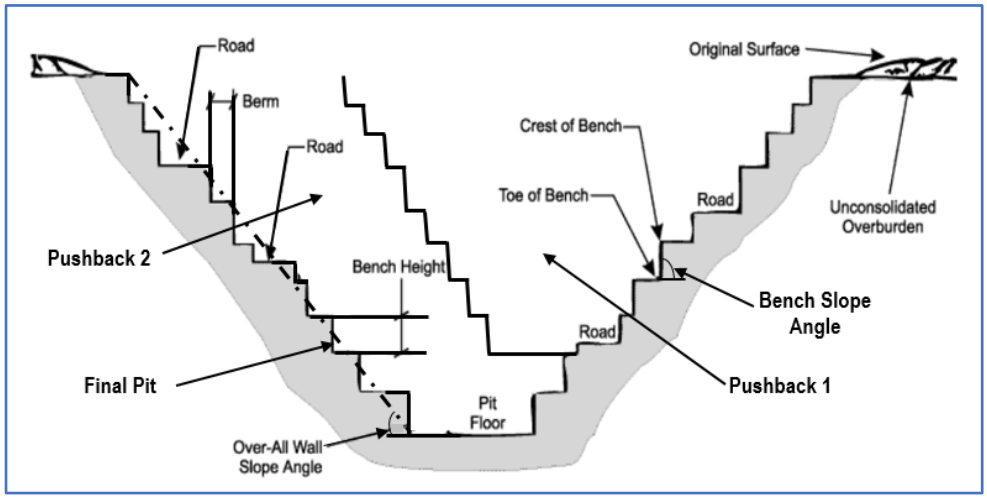


Figure 1.1: A generic cross-sectional view of an open-pit mine design
(Source: adapted from Hustrulid *et al.*, 2013)

Mine production scheduling of blocks inside the designed pushbacks and the final pit is then conducted. One of the outcomes of open-pit mine design is a report on the quantity and quality of potential Mineral Reserves within the UPL. This information provides guidance and necessary input parameters to the mine production scheduling process.

The focus of this thesis is on production scheduling of open-pit mines. Production scheduling defines the sequence with which blocks within the UPL should be extracted so that the maximum NPV of the mine is attained (Behrang *et al.*, 2014). It also establishes

the flow or movement of material from each block along the mineral value chain of a mining operation (Fathollahzadeh *et al*., 2021). Each block within the UPL should be extracted at the optimal time depending on its estimated economic value and physical position in the pit. Finding in advance the optimal time at which the block should be removed is a challenging task in the optimisation of mine production scheduling (Mousavi *et al*., 2016). A sustainable mining operation is guaranteed when mine production scheduling succeeds in defining the optimal time and sequence of mining block extraction.

Open-pit mine production scheduling (OPMPS) plays a crucial role in the mine planning process from LT planning to MT planning and ST planning (Smith, 2012). The OPMPS process aims to maximise the value derived from the recovery of Mineral Reserves by providing the optimal sequence to extract blocks while minimising operational costs and risks associated with the mining activities. The incorporation of production scheduling constraints into the mine planning process enhances overall operational efficiency while ensuring the achievement of project targets (Otto, 2019). Therefore, establishing comprehensive production scheduling strategies in the broader mine planning process becomes indispensable for the sustainable utilisation of Mineral Reserves.

Broadly, production scheduling is systematically divided into three horizons (Smith, 2012), namely LT, MT and ST production scheduling (Osanloo, *et al*., 2008). LT mine production scheduling is embedded in the strategic mine planning level. This horizon outlines a company's LT strategic objectives including maximising NPV and the life of mine (LOM). LT production scheduling generally defines an extraction sequence of blocks that maximises NPV subject to scheduling constraints. It also defines the optimal extraction time for each block within the UPL on an annual basis, spreading from five years up to the LOM horizon (Osanloo *et al*., 2008). LT production scheduling provides a scheduling framework from which a MT schedule is subsequently developed.

MT mine production scheduling is embedded in the tactical mine planning level. This horizon outlines MT tactical objectives to achieve the LT strategic objectives. MT mine production scheduling breaks down the LT production schedule into schedules of a foreseeable future. Therefore, LT production scheduling must inform MT production scheduling with mining blocks to extract in every LT production schedule period. MT production scheduling defines the optimal extraction sequence of blocks from each LT

production schedule period on either a half-yearly, quarterly or monthly basis. Osanloo, *et al*. (2008) suggested that MT schedules should spread over a one- to five-year horizon and must remain consistent with the LT production schedule to maximise NPV.

ST mine production scheduling is embedded in the operational mine planning level. This horizon outlines ST objectives to consecutively achieve MT tactical and LT strategic objectives. ST mine production scheduling breaks down the MT production schedule into shorter scheduling periods. Therefore, MT production scheduling informs ST production scheduling with mining blocks to extract in every MT production schedule period. ST production scheduling defines the optimal extraction sequence of blocks from each MT production schedule period on either a monthly, weekly, daily or shift-by-shift basis. Blom, *et al*. (2017) suggested that a ST production schedule should spread over 12 months and must remain consistent with MT and LT schedules along the production scheduling system. The ST production schedule addresses ST objectives such as minimising schedule deviations and costs, and maximising tonnage production and equipment utilisation. Unlike LT and MT production scheduling which focus on NPV as an economic decision metric, ST production scheduling focusses on profit as the economic decision metric.

1.3 Problem statement and research question

One of the challenges faced during the mine production scheduling process is to seamlessly integrate the production scheduling horizons. Failing to integrate the three production scheduling horizons into a structured production scheduling system is a major concern of the scheduling process. Disintegrated or independent production scheduling leads to sub-optimal extraction of an orebody which can negatively impact NPV and the overall performance of a mine (Levinson and Dimitrakopoulos, 2023). A sub-optimal schedule is characterised by reduced value of the Mineral Reserves, missed production rate targets, increased costs, and missed optimisation opportunities (Lamghari and Dimitrakopoulos, 2018). Also, disintegrated scheduling results in failure to understand important “what if” scenarios.

To address this problem, it is essential to develop an integrated production scheduling system that connects the three scheduling horizons to avoid scheduling misalignment both temporally and spatially. An integrated production scheduling system will ensure that LT production scheduling informs MT production scheduling which in turn informs ST

production scheduling. Figure 1.2 illustrates the mine production scheduling system linking the LT, MT and ST mine production scheduling horizons.

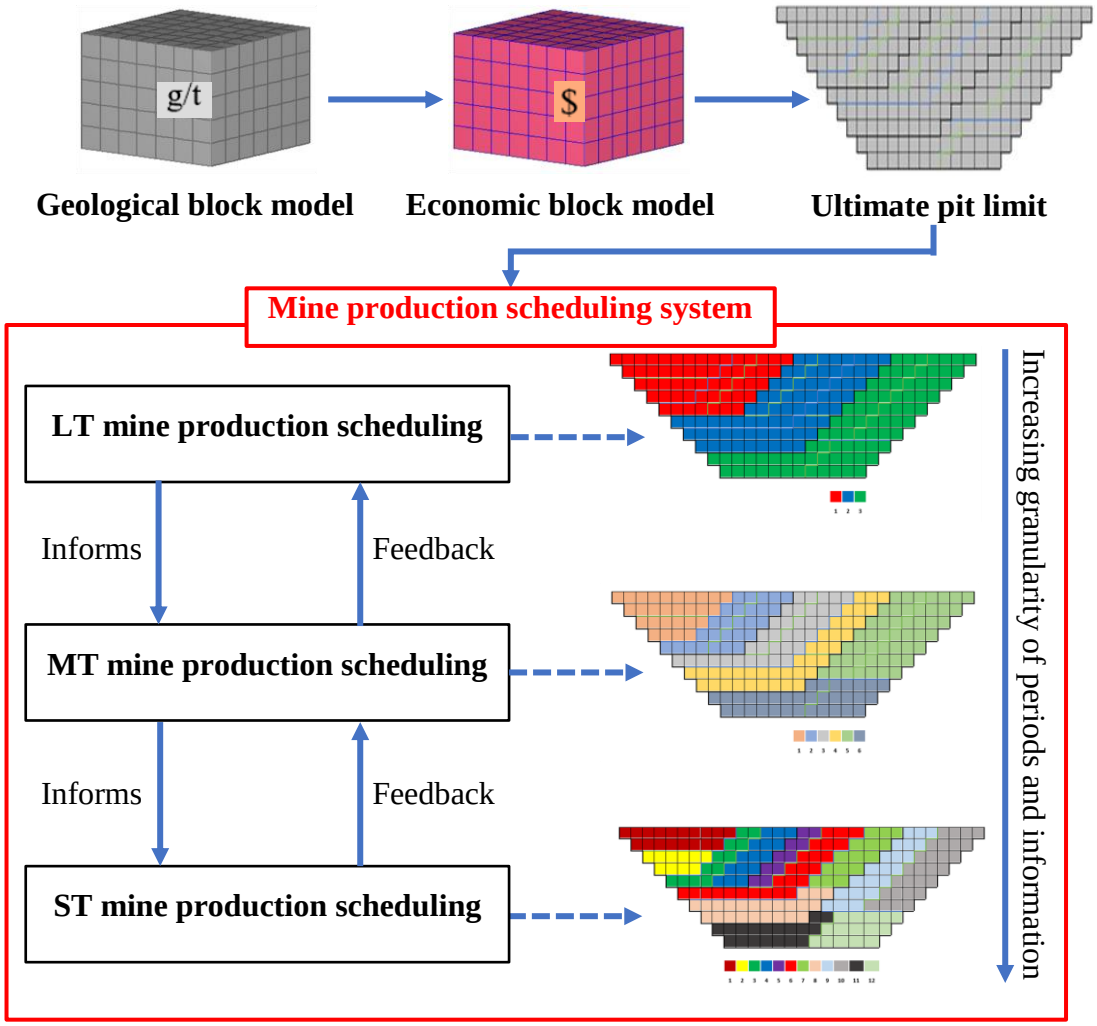


Figure 1.2: Open-pit mine production scheduling system highlighting the link between long-term, medium-term and short-term production schedules
(Muke *et al*., 2025a)

Apart from the need for the mine production scheduling system to be integrated, it must also be dynamic. A dynamic production scheduling system is one that can automatically update the scheduling horizons to incorporate any changes (such as grade and production rates) at any scheduling horizon. A combination of integrating and dynamically updating the production scheduling system will ensure efficient and cost-effective extraction of Mineral Resources (National Research Council, 2002). A mine production scheduling system which does not automatically update itself can suffer from sub-optimality and inaccuracy, which may result in generating an unsustainable NPV (Tolouei *et al*., 2020).

Therefore, ST production scheduling must provide feedback loops of new information to MT production scheduling and subsequently, MT revised information to LT production scheduling (Kloppers *et al*., 2015). Figure 1.2 schematically indicates the feedback process as conditions and information change.

Solving the mine production scheduling problem requires the use of optimisation techniques such as exact and metaheuristic techniques (Boland, *et al*., 2008). Exact or deterministic techniques employ mathematical models to determine the optimum solution that maximises the NPV of a mine (Leite and Dimitrakopoulos, 2014). However, the main limitation of deterministic models is their solution time. The solution time can increase exponentially as the number of decision variables increases, and large-scale models are computationally very expensive to solve (Alipour *et al*., 2017). Another limitation of deterministic models is that they do not account for the uncertainty that is often present in the mine production scheduling process. This has made exact techniques ineffective and impractical in finding the optimal sequence of extracting blocks from an open-pit mine (Bandaru and Deb, 2016). To overcome these shortcomings, recent studies have started to use metaheuristic techniques to find near-optimal solutions to stochastic production scheduling problems that cannot be solved using exact techniques (Yang, 2012).

The application of metaheuristic optimisation techniques for integrating mine production scheduling systems can play a significant role in facilitating seamless linkages and data sharing across different scheduling horizons. However, it is necessary to mathematically express the scheduling system using mathematical models such as mixed integer programming (MIP) models. MIP models can mathematically express complex decision-making problems that involve both discrete (integer) and continuous variables and can incorporate a wide range of constraints. Additionally, they can be used in combination with other metaheuristic methods such as GA to build a complementary hybrid model. MIP is an exact method that is developed for honouring hard constraints (Moxnes, 2015), while GA is flexible, stochastic and good at exploring large and complex search spaces (Gen *et al.*, 2008). Therefore, the combined MIP model and GA approach which was developed in this thesis, is a stochastic approach that can handle uncertainty, which exact and deterministic techniques cannot.

Several metaheuristic optimisation techniques have been applied in combination with exact methods to solve the production scheduling problem for open-pit mines (Lamghari and Dimitrakopoulos, 2016). However, because of the complexity of the problem, there have been no attempts to solve the production scheduling optimisation problem in a way that simultaneously integrates LT, MT and ST production scheduling horizons. Successful development and implementation of the combined MIP model and GA approach can advance the use of metaheuristic optimisation techniques to dynamically integrate the production scheduling process. This raises the following research question which this thesis seeks to answer:

> "*Can a MIP model that is solved using GA be able to optimise and dynamically integrate LT and MT mine production scheduling to improve temporal and spatial alignment while attempting to maximise NPV and accounting for operational uncertainty?*"

1.4 Research motivation and significance

Metaheuristic techniques have proven to be appropriate for exploring large solution spaces efficiently and effectively while taking into consideration uncertainty associated with the mine production scheduling process. These techniques harness the power of natural phenomena in nature-inspired algorithms (NIAs) such as biological processes that include foraging, genetic evolution, and swarm intelligence, to solve complex optimisation problems (Bandaru and Deb, 2016). This thesis employed GA, which is one of the most widely used metaheuristic techniques applied to solve stochastic and difficult mine planning optimisation problems (Gen *et al*., 2008). GA is inspired by the process of natural selection and evolution. It works by creating a population of potential solutions, evaluating their fitness based on predefined criteria, and selecting the best individuals for reproduction (Mitchell, 1999). Through a process of crossover and mutation, new generations are created until a near-optimal solution is found. The benefit of using metaheuristic approaches like GA lies in their ability to handle complex problems with multiple constraints and objectives (Paithankar and Chatterjee, 2019). This is the reason why GA was selected in this thesis.

To date, there are several applications of metaheuristic techniques in optimising production scheduling. However, the current studies optimise LT, MT and ST production schedules

independently, even though the studies have recognised shortcomings associated with disintegrated LT, MT and ST production scheduling (Matamoros and Dimitrakopoulos, 2016; Azzamouri *et al*, 2019; Cutler and Dimitrakopoulos, 2024). Generating independent mine production schedules is known to be detrimental to a project's NPV due to scheduling misalignment and inconsistency. Currently, there is no method that systematically integrates LT, MT and ST production scheduling horizons. With an integrated production scheduling system, all horizons are incorporated and correlated to mitigate scheduling misalignment. Otto (2019) stated that there is a clear interdependence between mine production scheduling horizons, and optimising the scheduling horizons simultaneously is strategically essential. These scheduling horizons should be linked together through feedback loops to enable an efficient correlation between them as in the work by Otto (2019), albeit manually.

There have been attempts by Levinson and Dimitrakopoulos (2023) to integrate LT and ST mine production scheduling using stochastic mathematical programming and reinforcement learning (RL). Nonetheless, the work by Levinson and Dimitrakopoulos (2023) excluded the MT production schedule horizon. When the process skips the MT horizon, it does not allow a smooth transition between LT and ST horizons, because mine production scheduling is a sequential process from LT to MT and ST production scheduling. Therefore, there is a knowledge gap that needs to be investigated to provide an integrated solution approach. To close this knowledge gap, this thesis developed a MIP model which dynamically integrates the LT and MT open-pit mine production scheduling horizons to avoid scheduling misalignment between the two horizons, and the model was solved using GA.

GA solution approaches offer a promising avenue for the integrated production scheduling system. Their flexibility, adaptability, and ability to handle complex problems make them valuable tools for improving efficiency in mining operations. By implementing these advanced techniques into the mine planning processes, the production scheduling process can unlock new levels of productivity and profitability. This further supports why GA was selected in this thesis.

1.5 Research aim and objectives

This thesis aimed to develop a dynamic production scheduling MIP model that integrates LT and MT production scheduling and solved using the GA technique. In addition to dynamically integrating the scheduling horizons, the system also guarantees an inclusive schedule where spatial and temporal scheduling compliance is effectively achieved. To achieve the research aim and answer the research question, the objectives of this thesis were to:

- Review literature on open-pit mine production scheduling.
- Develop a MIP model that integrates LT and MT mine production scheduling horizons. MIP was selected because it can develop a practical mine production scheduling system and uses constraints to link successive scheduling horizons while using continuous, binary and integer variables to cater for partial or whole extraction of blocks.
- Develop a GA approach for solving the MIP model while incorporating uncertainties associated with mining rate, processing rate and ore grade target.
- Code the combined MIP model and GA approach in Python programming language, which is a valuable language due to its versatility and efficiency in data analysis and visualisation. The combined MIP model and GA approach dynamically integrates LT and MT production schedules to improve NPV as opposed to when these scheduling horizons are independently optimised. The MIP model also ensures spatial and temporal alignment between the LT and MT schedules.
- Validate the combined MIP model and GA approach by applying it on three different block models. The validation used the same input parameters and compared its results with those reported on MineLib as best-known feasible linear programming (LP) relaxation solutions obtained using a TopoSort heuristic algorithm.

1.6 Research scope

An integrated MIP model was developed in this thesis to optimise LT and MT production scheduling for open-pit metal mines. The thesis focused on open-pit metal mines because they offer great flexibility in terms of changing production schedules, based on factors such

as geological, operational and economic factors compared to underground mines (Quigley and Dimitrakopoulos, 2020). This flexibility creates more opportunities in open-pit mines than in underground mines, for applying global search optimisation techniques like GA (Alipour *et al*., 2022). Due to that reason, Nhleko *et al.* (2018) stated that fewer optimisation techniques have been developed to optimise underground mines because they are less flexible compared to open-pit mines. Moreover, open-pit mining accounts for 80% of global metal ore production, whereas underground mining contributes about 20% (Ghorbani *et al.*, 2023).

Levinson and Dimitrakopoulos (2023) integrated LT and ST mine production scheduling horizons using reinforcement learning and stochastic programming but skipped the MT production scheduling horizon. The limitation of skipping the MT horizon was discussed in Section 1.4. This thesis focuses on integrating the LT and MT production schedules to provide a framework that can be used to support ST scheduling, while managing the complexity and variability of operational planning. Thus, the integrated LT and MT production scheduling model complements the work done by Levinson and Dimitrakopoulos (2023) and establishes new opportunities for future research to integrate LT, MT and ST production schedules. Azzamouri *et al*. (2019) stated that MT mine production scheduling is an important part of the scheduling process that bridges LT and ST schedules. An integrated LT and MT mine production schedule is critical to avoid spatial and temporal production scheduling misalignment and sterilisation of Mineral Reserves. Sterilisation of parts of a mineral deposit destroys the economic value of the deposit (Tholana and Musingwini, 2022). Integrating production scheduling horizons ensures that the LT scheduling objectives are achievable at the MT horizon and that the MT scheduling objectives are achievable at the ST horizon.

The developed integrated MIP model and GA approach was applied to a block model downloaded from Geovia Surpac® software (Geovia, 2024), to demonstrate the improvement of NPV compared to when LT and MT production schedules are independently optimised. The validation of the integrated MIP model and GA approach was done by using the same input parameters and comparing the results with the best-known feasible solutions obtained using the TopoSort algorithm when applied to hypothetical block models freely downloaded from MineLib (2012).

1.7 Structure of the thesis

This thesis is divided into six chapters. Chapter 1 introduces the thesis and consists of a background on open-pit mine design and scheduling process. This is followed by a discussion about the research problem statement and research question, the motivation and significance of the research highlighting the knowledge gap that was identified. It then presents the aim and objectives of the research and a description of how the knowledge gap was addressed. Furthermore, the chapter presents the scope of the research built around integrating LT and MT mine production scheduling horizons. The chapter ends with a summary of the structure of the thesis.

Chapter 2 reviews literature on open-pit mine production scheduling, and the relationship between LT, MT and ST mine production scheduling. Additionally, the chapter provides a critical review and analysis of the relevant literature on optimising the production scheduling process. Then it describes the GA technique which is the solution approach applied to solve the MIP model developed in this thesis.

Chapter 3 presents an MIP mathematical model that integrates LT and MT production scheduling. The chapter discusses spatial and temporal open-pit mine production scheduling compliance between LT and MT production scheduling. Chapter 4 describes the development of the GA approach used to solve the MIP model which was developed in Chapter 3. Then, the chapter presents how the GA approach that was coded in Python programming language.

Chapter 5 presents and describes the geological block models that were used to implement the combined MIP model and GA approach. Results are presented and analysed in the chapter, as well as the validation of the approach. Chapter 6 presents the conclusions and recommendations of the thesis. It highlights the main outcome of the thesis and its contribution to knowledge. Finally, the chapter provides recommendations for future research work on OPMPS.

2. OPEN-PIT MINE PRODUCTION SCHEDULING SYSTEM OPTIMISATION TECHNIQUES

2.1 Overview of Chapter 2

The purpose of Chapter 2 is to review literature on OPMPS. Firstly, the chapter describes the three scheduling horizons which are LT, MT and ST horizons. Secondly, the chapter discusses optimisation techniques applied to solve these scheduling horizons. Optimisation techniques are classified into two categories namely exact and approximate (or metaheuristic) methods. The advantages of using metaheuristic methods are also presented in this chapter. Finally, the chapter describes GA, its concept and optimisation procedure.

2.2 Open-pit mine production scheduling

2.2.1 Long-term mine production scheduling

Production scheduling plays an important role at every horizon in the LOM. The major element of the mine scheduling process is the optimisation of LT production scheduling which is strategic in nature (Moosavi and Gholamnejad, 2015). LT mine production scheduling is the process of creating a yearly extraction schedule that will be used to guide the extraction of material until the depletion of Mineral Reserves. Many researchers refer to the LT mine production schedule as a high-level LOM plan or strategic mine plan (Akbari *et al*., 2008; Morales *et al*., 2019; Kloppers *et al*., 2015). The overall strategic goal of LT production scheduling is defining how the orebody should be mined over the LOM to reach the economic target of the project (Jélvez *et al*., 2023). The LT production schedule determines when and where to start mining, what ore and waste material to mine in each scheduling period, and how to sequence the mining extraction. An optimal production schedule should aim to maximise the production of ore while minimising the cost of production at an acceptable operational risk.

Kear (2006) stated that LT production scheduling focuses on identifying strategic objectives with a view of more than five years. Due to the ever-changing mining environment, the LT schedule is subject to a continuous review and assessment in light of MT progress. The review is conducted to update the schedule as geological, economic and technical parameters change. Osanloo *et al*. (2008) stated that optimisation of the LT schedule considers scheduling constraints including ore grade target, block precedence,

mining rate, processing rate and Mineral Reserve constraints. Figure 2.1 illustrates different activities involved in developing a LT production schedule.

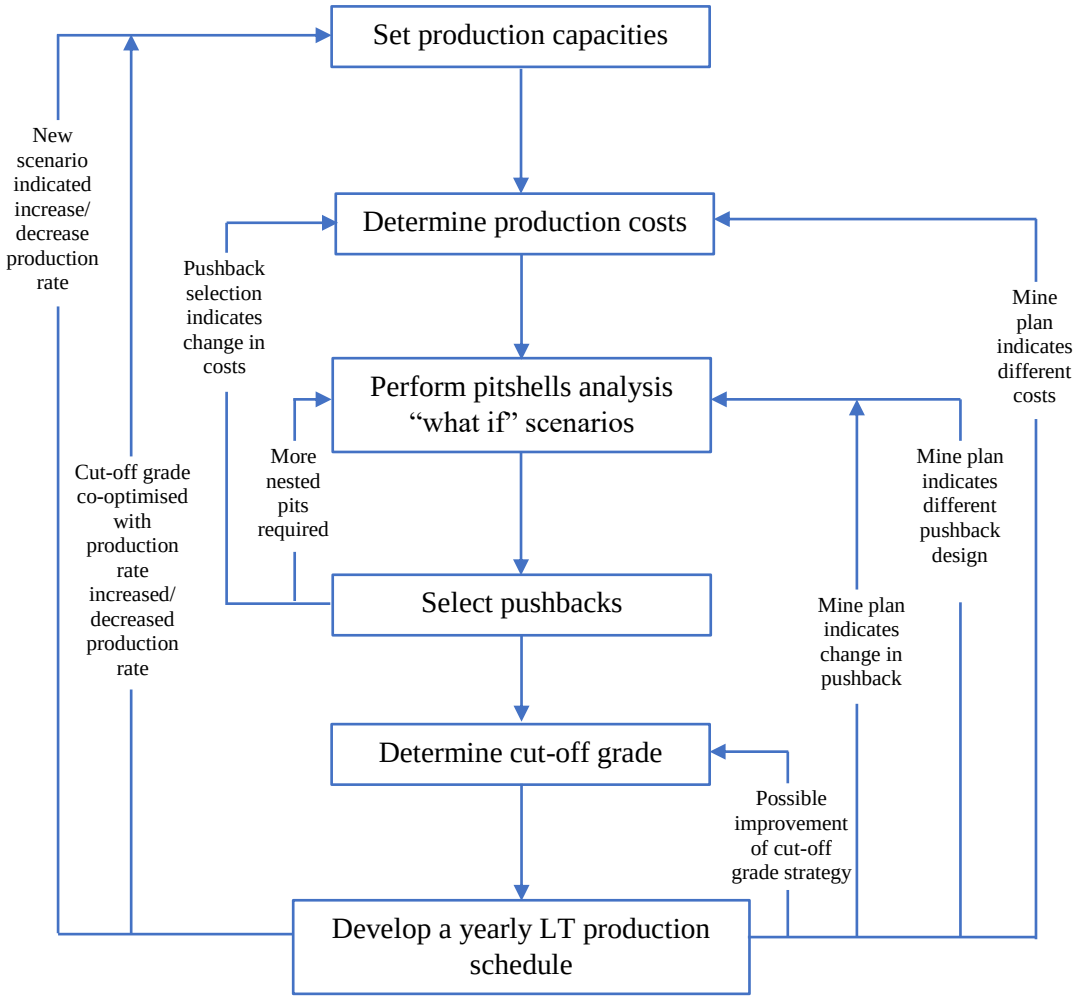


Figure 2.1: A typical activity diagram for long-term production scheduling
(Source: adapted from Thorley, 2012)

The importance of LT production scheduling is not only to determine NPV and LOM of the project, but it is also to set a foundation for MT and ST production scheduling (Blom *et al*., 2017). Additionally, other reasons for developing a LT mine production schedule for an open-pit mine are that the LT production schedule (Osanloo *et al*., 2008):

- Enables a more accurate estimation of Mineral Reserves, which is important for mineral inventories and reporting.
- Provides a blueprint plan of mining operations over the LOM, which ensures effective planning and coordination of these operations.

- Optimises the utilisation of Mineral Reserves and maximises the value of the project.

2.2.2 Medium-term mine production scheduling

MT mine production scheduling is the next horizon of scheduling following the LT production scheduling (Azzamouri *et al*., 2019). MT production scheduling is embedded in the tactical mine planning level and is developed on the established annual LT scheduling periods. The goal of MT production scheduling is to create a production schedule that can be executed over a period that is longer than the ST schedule, but shorter than the LT schedule. MT production scheduling covers the next one to five years of production and it is used to develop half-yearly, quarterly or monthly production schedules (Manríquez *et al*., 2020). The half-yearly production schedule is often generated at the mid-point of a budget year to ensure that mining activities transition smoothly from one year to the next (Thorley, 2012). Therefore, it is important to consider both the LT goals of the company as well as the ST or operational needs of the current situation when creating MT production schedules, thus making MT production scheduling both quasi-strategic and quasi-operational.

A MT production schedule should be flexible enough to accommodate changes that may occur at this horizon of scheduling. For example, if a machine breaks down and needs to be repaired, the schedule should allow for the repair of the machine while minimising the disruption of the whole production process. Once the MT production schedule has been created, it is important to monitor its execution and make changes as necessary. Monitoring the MT schedule should ensure that the scheduling system is being followed and that any changes that need to be made are made on time.

The current literature review on mine production scheduling indicates limited research work in solving the MT production scheduling problem. Azzamouri *et al*. (2019) argued that this might be partly justified by the lack of interest in developing MT schedules. However, MT production scheduling is critical in the development of a mine production scheduling system because it supports the creation of an annual budget. Additionally, MT production scheduling bridges the gap between the LT production schedule and the ST production schedule, while providing a much more detailed extraction sequence to inform the ST production schedules.

Considering the LT horizon and lack of mining location detail and the large number of blocks to schedule over many years in the LT schedule, it is essential to split the yearly LT schedule into shorter scheduling periods (Osanloo *et al*., 2008). That is why the MT schedule breaks down each annual LT schedule period into six-, three- or one-month horizons. This is to increase the mining location detail of blocks that must be scheduled every six months. Half-yearly MT mine production scheduling provides an extraction sequence that optimally splits each annual mining area defined in the LT mine production scheduling, into two mining areas.

The MT six-month production schedule identifies the annual ore advance both in quantity and quality to meet the plant's tonnage and ore grade demand. Since waste blocks are present in the pit, the six-month schedule must also identify the waste advance to maintain the stripping ratio of the mine. Since a MT schedule is much more detailed than a LT schedule, the MT six-month schedule seeks to minimise relocation of equipment, and meet stockpile and dump construction needs (Thorley, 2012). This is to ensure that the MT schedule provides a smooth transition from a LT schedule to a ST schedule. Figure 2.2 illustrates typical activities and constraints involved in MT six-month production scheduling.

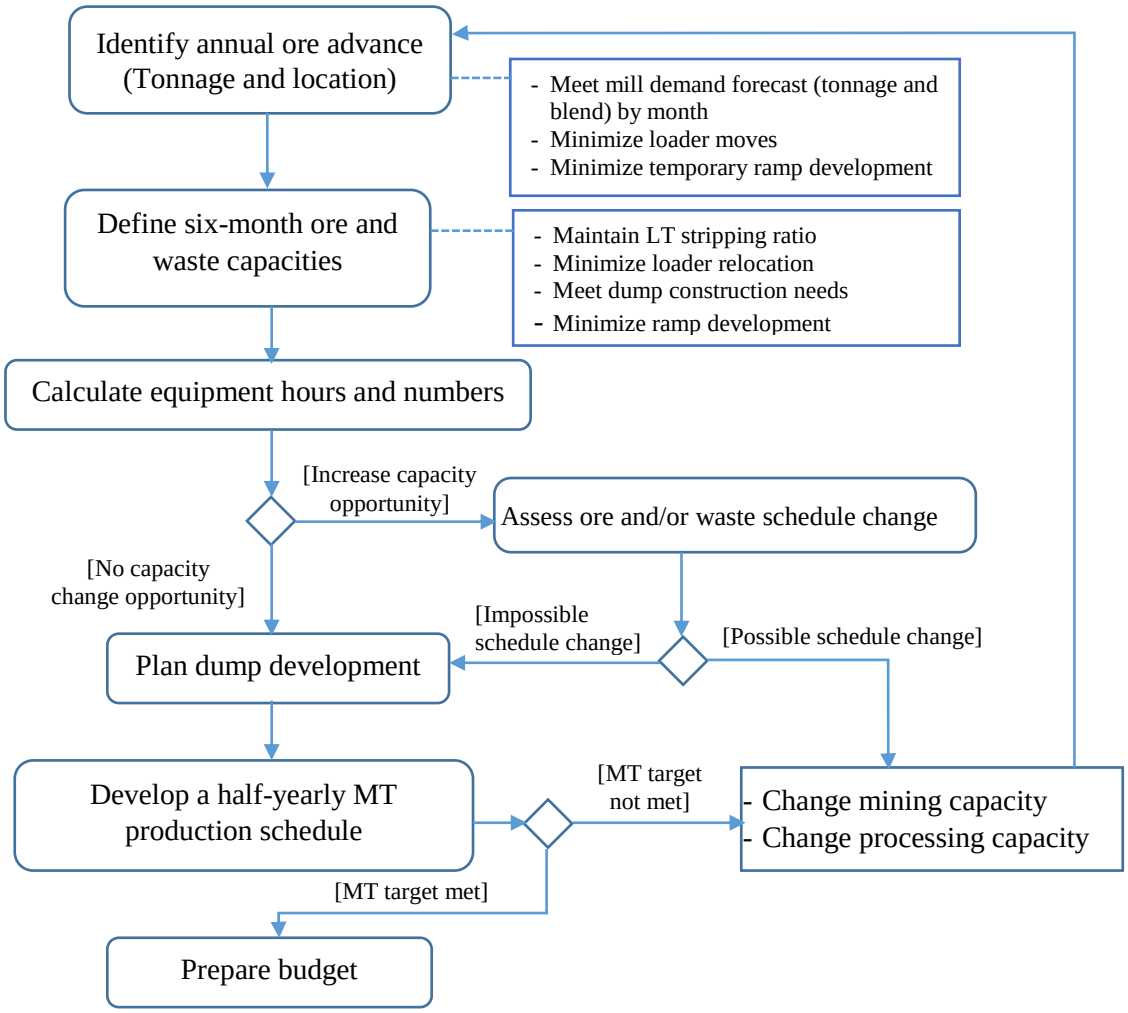


Figure 2.2: A typical planning activity diagram for medium-term six-month production scheduling
(Source: adapted from Thorley, 2012)

The MT three-month production schedule is similar in concept to the six-month MT schedule but it is updated every quarter (Manríquez *et al*., 2020). A quarterly MT mine production schedule is normally developed to deliver a quarterly extraction sequence from the six-month mining areas previously defined in the MT six-month production schedule. Therefore, it covers a period of half-year length and optimally splits each six-month mining area defined in the MT six-month mine production schedule, into two mining areas. Quarterly MT production schedules provide a higher level of certainty concerning planned shutdowns than the six-month schedule and respond to monthly changing conditions. At this scheduling horizon, specific equipment may be assigned to specific active mining faces. This will allow for equipment movements and any related infrastructure changes to be identified and planned for in advance (Thorley, 2012).

The scheduling period in the quarterly MT production scheduling is not yet an absolute timeframe to schedule a shift, daily or weekly schedule for the extraction of selected mining areas for every month. However, it bridges the gap between the LT and ST production schedules and ensures a smooth transition from six-month MT production schedules to one-month ST production schedules. Therefore, it is essential to introduce a ST production schedule to break down the MT three-month schedules into monthly schedules of weeks, days and shift horizons.

2.2.3 Short-term mine production scheduling

To comply with both MT and LT production schedules, as well as achieve daily operational targets, careful consideration is given to the development of ST production schedules. The detailed ST production scheduling is embedded in the operational planning level and focuses on specific mining activities. The operational mining process mainly involves optimising the use of mining equipment fleet and crushers at processing plants, as well as maintaining grade blending requirements (Upadhyay and Askari-Nasab, 2018). Blom *et al*. (2017) also said that operational activities must minimise material re-handling and the cost of production. Therefore, the main goal of ST production scheduling for open-pit mines is to ensure optimum equipment utilisation. This includes efficient shovel placement and truck allocation to meet both quantitative and qualitative production targets while also adhering closely to MT and LT mine plans.

A reliable and sustainable ST production schedule must align with the MT production schedule and aid in the proactive decision-making process to achieve LT production schedule objectives (Manríquez *et al*., 2020). By taking a proactive approach, potential deviations from the overall mine production schedule can be minimised, thus reducing revenue losses. Detailed ST production schedules are desired to provide a sequence of extraction of mining faces over ST horizons such as months, weeks and days (Upadhyay and Askari-Nasab, 2018).

A monthly ST production schedule is a detailed breakdown of the MT three-month production schedule, spread out over a series of monthly timeframes (Osanloo *et al*., 2008). This schedule takes into consideration various factors that may cause delays, such as equipment availability, maintenance, and unpredictable circumstances (Matamoros and Dimitrakopoulos, 2016). The monthly sequence of extraction is determined by the overall

three-month or six-month production schedule, which must be met to reach the target of the annual production schedule. To ensure that optimal financial outcomes are achieved, the monthly production schedule focuses on minimising any deviations from the annual production schedule. Thorley (2012) noted that recalculating equipment hours and truckloads may sometimes be necessary during the monthly production scheduling process to account for changes in ore and waste extraction schedules. Monthly scheduling changes are also used to update the budget plan that guides the production activities by providing detailed production targets (Otto and Musingwini, 2020). In addition, the monthly production schedule also includes details on weekly and daily execution schedules, production rates, and planned tasks. Figure 2.3 shows typical planning activities involved in ST monthly production scheduling.

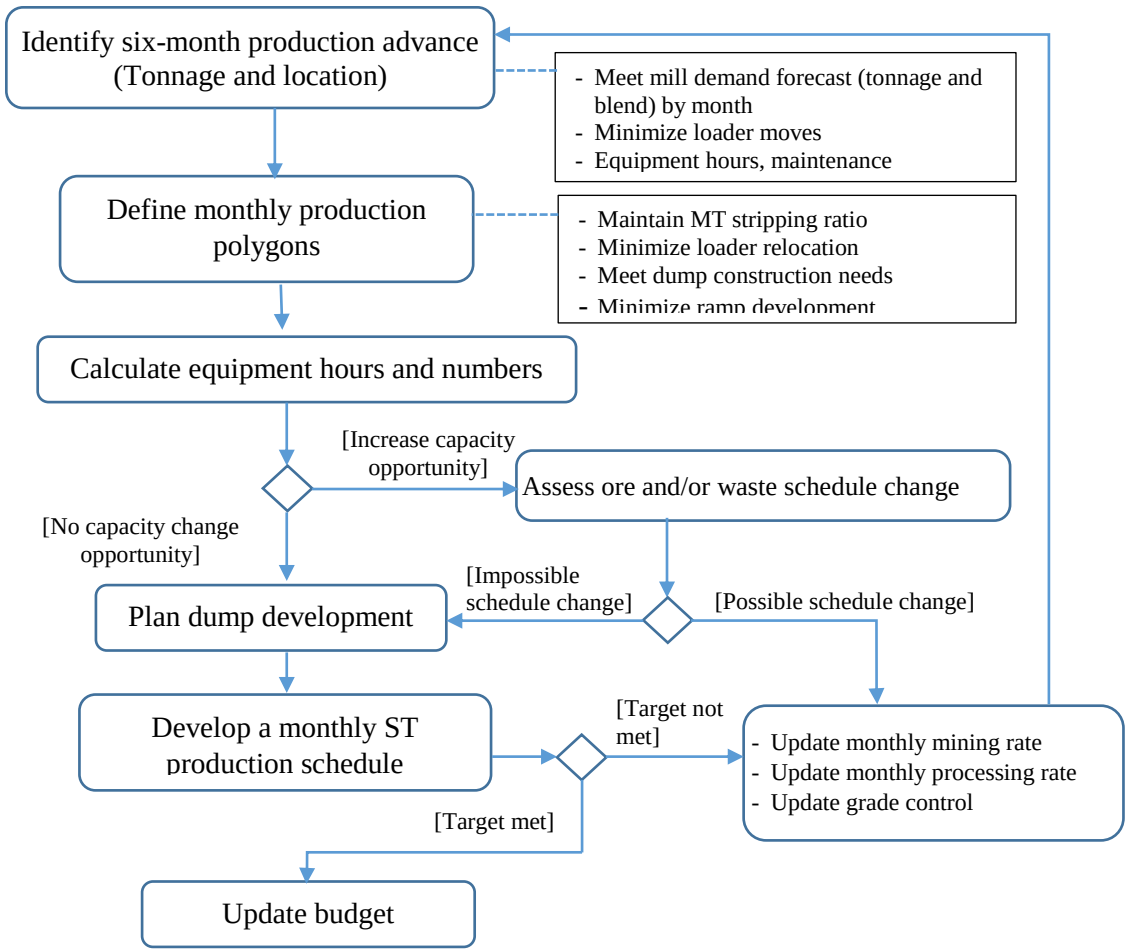


Figure 2.3: A typical planning activity diagram for monthly short-term production scheduling
(Source: adapted from Thorley, 2012)

A weekly ST production schedule is developed to ensure that quality and quantity production targets are met. It determines weekly mining areas which should be mined and where the material should be sent. Blasting schedules are key elements at this level, as they determine when certain mining areas will become available for mining (Nogholi, 2015). Weekly objectives may include maximising equipment utilisation, minimising material re-handling, maintaining the grade of ore produced, and meeting daily production rates (Blom *et al*., 2017). A daily production schedule includes activities such as drilling, blasting, excavation, and hauling. It identifies drill and blast patterns to maintain ore inventory levels and determines the crusher set to receive mined material (Matamoros, 2014). Additionally, waste shovel priority is established along with the number of trucks needed to maintain planned stripping ratios in the mine. Both the weekly and daily schedules work together to develop an effective sequence of extracting a series of mining areas within their respective scheduling horizons (Blom *et al*., 2017). Figure 2.4 shows typical planning activities involved in the creation of a weekly ST production schedule.

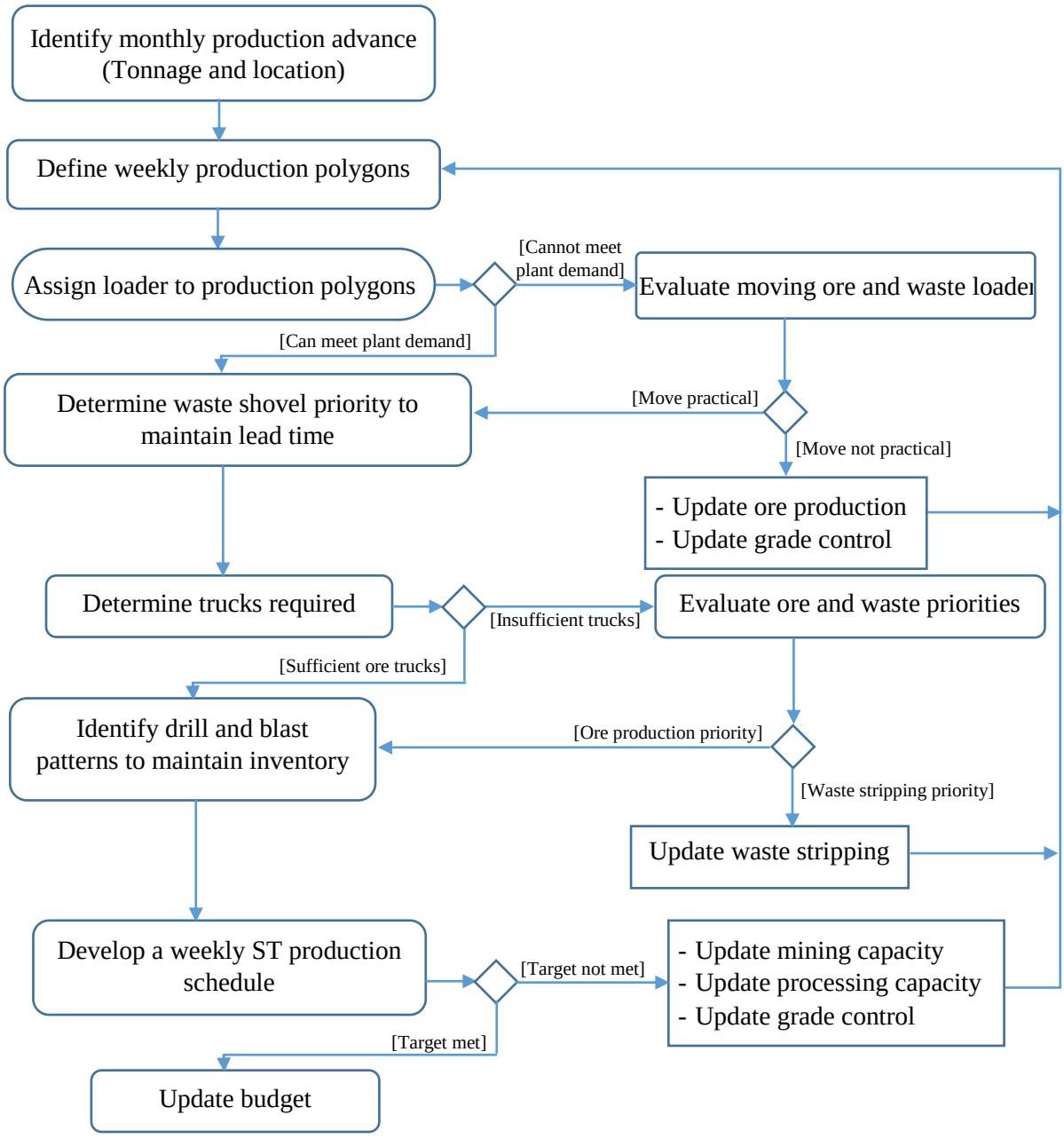


Figure 2.4: A typical planning activity diagram for weekly short-term production scheduling
(Source: adapted from Thorley, 2012)

A daily ST mine production scheduling process involves addressing deviations from the weekly production schedules. This process follows the same flow as the weekly production scheduling and any changes in equipment placement or planned production locations should remain consistent with the original plan (Matamoros and Dimitrakopoulos, 2016). Unforeseen equipment maintenance, unexpected variations in ore or waste qualities, and other issues will require prompt responses from the operational planning team. Usually,

multiple meetings are held with the maintenance and operations teams, along with support function supervisors in drilling, blasting, mine electrical, and dewatering functions, to identify and address any problems accordingly (Thorley, 2012). It is crucial to carefully evaluate reactions to correct operational disruptions. Figure 2.5 shows typical planning activities for daily ST production scheduling.

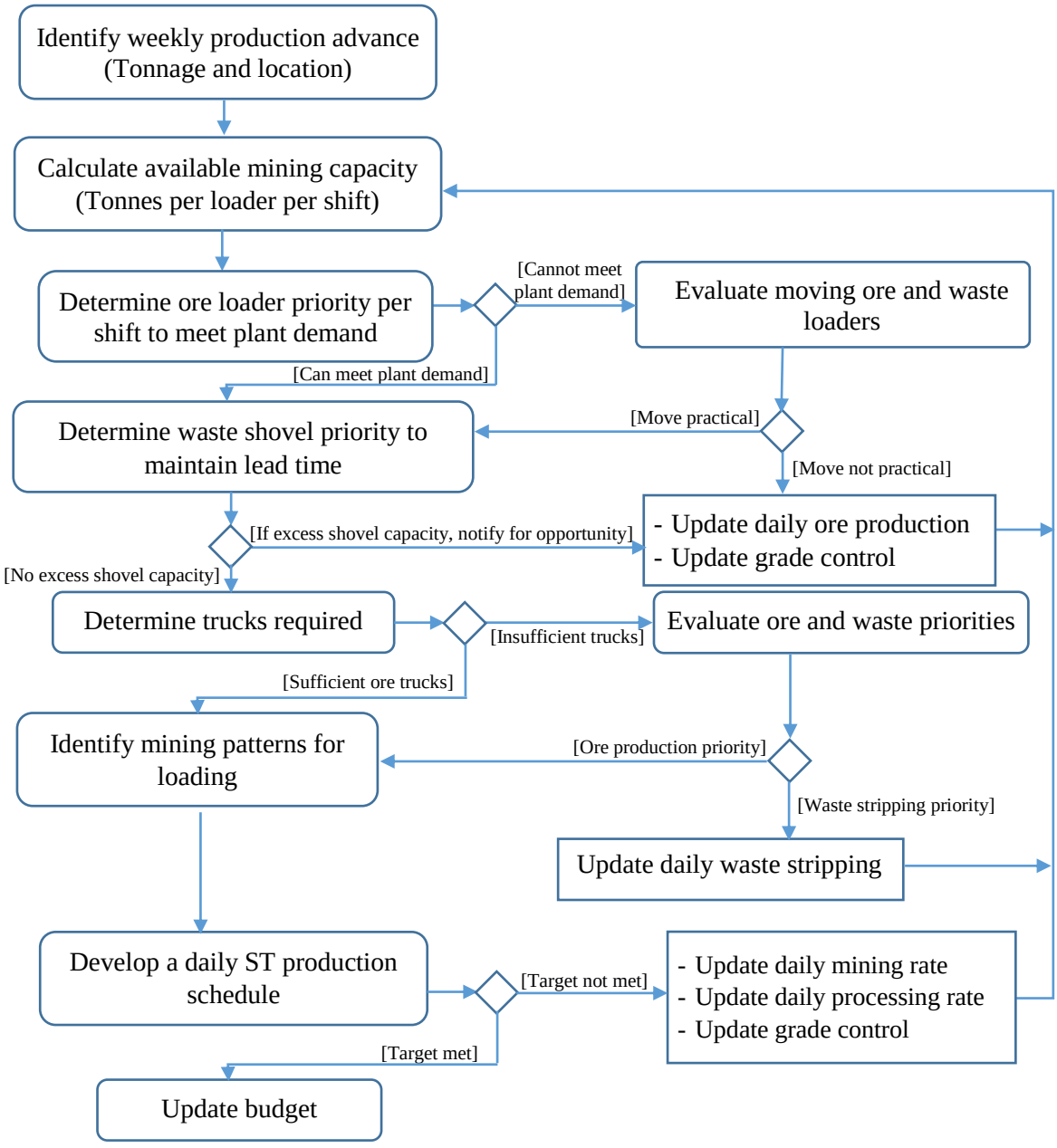


Figure 2.5: A typical planning activity diagram for daily short-term production scheduling
(Source: adapted from Thorley, 2012)

2.3 Open-pit mine production scheduling optimisation techniques

The term “optimum” is derived from Latin to signify “the ultimate ideal”, while “optimus” means “the best” (Andreasson *et al*., 2005). Therefore, mathematically, optimisation implies finding “the optimal value of a function, often subject to constraints” (Rardin, 2017). A mathematical optimisation model comprises an objective function subject to a set of constraints, which are typically expressed as a system of equations or inequalities. These types of optimisation models are commonly found in almost all areas of decision making such as engineering, economics, physical, chemical and biological sciences (Lin *et al*., 2012).

The optimal solution of an objective function in mathematical terms aims to either maximise or minimise values (Rardin, 2017). At its core, optimisation involves defining variables to optimise an objective function while satisfying certain limitations imposed by the problem at hand. These variables represent quantities with numerical values that can be adjusted within specified ranges, allowing decision makers to systematically explore different scenarios and evaluate potential outcomes (Hoeve, 2005). With careful consideration of these variables, optimisation enables the modelling of complex real-world situations mathematically (Bierlaire, 2018). Modelling an optimisation problem is made up of four basic components that include (Andreasson *et al*., 2005; Rardin, 2017; Guenin *et al*., 2014):

1. An objective function: a mathematical expression of the problem to be optimised, whether maximising or minimising.
2. Controllable inputs: are the set of decision variables that affect the value of the objective function. Decision variables are essential in the development of optimisation models. If there are no variables, it is not possible to define the objective function and the problem constraints.
3. Uncontrollable inputs: are called parameters which may be fixed numbers associated with a particular problem. There are also some parameters whose values might be uncertain in the decision-making process.
4. Constraints: are relations between decision variables and the parameters. A set of constraints allows some of the decision variables to take on certain values, and exclude others. Most optimum solutions to real-world problems cannot often be obtained without putting constraints on the decision variables.

Optimisation methods encompass several mathematical models used to enhance the decision-making processes. They are considered indispensable tools employed to improve the efficiency and effectiveness of various mining operations. They leverage mathematical algorithms and computational resources to identify optimal solutions to complex problems involving scarce mineral commodities, uncertain conditions, or conflicting objectives (Murthy, 2007). Employing optimisation methods can systematically enable the evaluation of multiple alternatives while considering constraints and constraint trade-offs to derive efficient strategies that maximise desired outcomes or minimise undesired outcomes.

Optimisation methods are rooted in operations research (OR) and are used to solve various real-world problems (Lyeme and Seleman, 2012; Musingwini, 2016). Monika (2021) defined OR as a combination of two words "operation" and "research". Operation means some action applied in any area of interest, while research implies an organised process of getting and analysing information about the problem environment. Lyeme and Seleman (2012) stated that one of the most important characteristics of OR is its ability to represent a problem as a whole or as a system orientation. This means that an activity by any part of an organisation or system has some effect on the activity of every part. Therefore, to evaluate any decision, OR must be able to identify all possible interactions and determine their impact on the system as a whole.

OR techniques assist in selecting appropriate optimisation methods based on a specific problem at hand. These methods can be (Kozan and Liu, 2011):

- Exact or deterministic methods: These are methods or techniques that can find the exact optimal solutions but can take long computational times to generate solutions. The computational time increases exponentially with respect to (or as a function of) the size or dimension of the problem at hand. Exact methods can solve every instance or any dimension of the optimisation of a problem and they guarantee optimality. They are split into a broad set of methods including branch-and-X, constraint programming (CP) and dynamic programming (DP). Branch-and-X techniques are used to solve LP, integer programming (IP) and MIP problems (Jose *et al*., 2014).
- Approximation methods: These are methods or techniques that can generate solutions within polynomial execution time as a function of problem size (or

dimension) for problems for which exact methods take exponential time. It can be shown that the solution found using approximate methods is some distance away from the optimal solution. Approximate methods employ approximation algorithms (AA) and heuristic algorithms to solve complex problems that involve randomness.

- Heuristics: These are problem-dependent methods that use intuition, mental shortcuts or rules of thumb to solve problems by leveraging on past experiences and information available on the specific problem to guide decision-making. Unlike exact methods, heuristic methods can find solutions faster but ~~it~~ cannot guarantee optimality. Heuristic algorithms include problem-specific heuristics (PSH) and more generic metaheuristics or stochastic methods.
- Metaheuristics: These are methods designed to solve a wide range of optimisation problems, hence not problem dependent, unlike heuristics. For many metaheuristic algorithms, it can be shown that near-optimal or optimal solution is found with a probability, which increases with the algorithm improvement. Figure 2.6 illustrates a classical taxonomy of optimisation methods, and provides common examples of methods in each category.

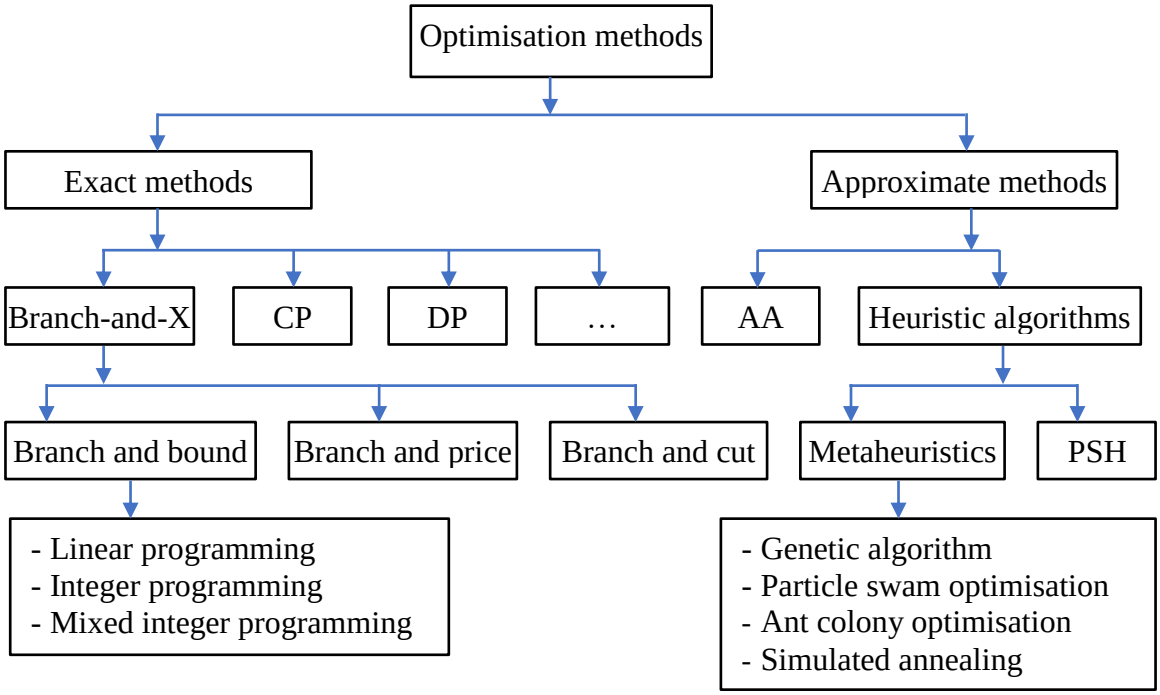


Figure 2.6: Classification of classical optimisation methods
(Source: adapted from Jose *et al*., 2014)

Today, OR has evolved and has been applied to various optimisation problems and opportunities. With the advent of computers and advancements in computational power, OR techniques are now more accessible and computationally possible to solve (Hoeve, 2005). The integration of optimisation algorithms, simulation software, and decision support systems with computer technology further expanded the capabilities of OR techniques. Modern developments in OR include the application of metaheuristics, machine learning, and data analytics to solve complex optimisation problems and make data-driven decisions (Lyeme and Seleman, 2012).

The use of OR techniques and computer models has become entrenched in the mining industry, particularly in the area of mine planning and optimisation (Wilke, 1987). OR provides systematic and analytical approaches to optimise various aspects of mine planning including but not limited to open-pit optimisation, pushback optimisation, production scheduling, equipment selection, maintenance, and mineral processing (Newman *et al*., 2010). The application of OR in mine planning has experienced considerable growth since its first application in 1965, with numerous advances in theory, algorithms, and computational techniques (Lin *et al*., 2012). It is important to note that the selection of OR techniques depends on the specific characteristics of the mine planning problem, data availability, computational resources, and the objectives of the mining operation (Newman *et al*., 2010). A combination of multiple techniques, as well as integration with other technologies like machine learning, can enhance the effectiveness and efficiency of mine planning models (Bertayeva *et al*., 2019).

2.3.1 Exact open-pit mine production scheduling optimisation techniques

An exact or deterministic OR model assumes that all input data is known with certainty and will always generate the same exact outcome for the known set of inputs, regardless of the number of times that the model is re-run (Tholana, 2021). Rardin (2017) said that exact techniques are used because they often produce valid enough results to be useful and because they are easier to model compared to stochastic techniques. In recent decades, exact techniques have found extensive applications in solving mine planning problems requiring consideration of several analytical properties to find optimum solutions. The use of exact optimisation techniques allows the mine planning process to converge to a solution systematically and rigorously (Newman *et al*., 2010).

Exact techniques provide a step-by-step algorithm where given particular inputs, the same output is always produced. No matter how many times the algorithm is run, it will always produce the same result (Tholana, 2021). This means that exact algorithms are more consistent and predictable than stochastic algorithms which can produce different results even when given the same inputs (Moxnes, 2015). Exact methods have been proven to be effective in solving individual or integrated mine planning problems concerning economic return like production rate optimisation or NPV maximisation. Exact techniques offer several advantages in the context of mine production scheduling, providing numerous benefits to a mining operation. Among these advantages include:

- The ability to achieve consistent mine production schedule outcomes and guarantee optimality within the given condition which is not the case for heuristics methods (Mousavi *et al*., 2016).
- The capacity to identify bottlenecks within a mine production scheduling process by analysing each task's dependencies and constraints (Rardin, 2017). By doing so, exact techniques enable companies to streamline their operations efficiently, allocate resources strategically, and ultimately enhance productivity levels throughout the entire mining value chain.
- The ability to provide valuable insights for decision-making processes regarding equipment utilisation along with mine production scheduling (Murthy, 2007). Exact techniques possess the capability to model various scenarios based on different parameters such as machine capacities or maintenance schedules. Thus, they aid in optimising machinery deployment strategies while prioritising tasks effectively according to predefined criteria like profitability or safety considerations.
- The ability to provide a potential tool for effective time management within mine production scheduling metrics (Murthy, 2007). These robust computational methods allow mine planners to precisely determine project durations by considering multiple factors such as task interdependencies and resource availability under given constraints.

Although exact techniques may be a good choice to solve the mine production scheduling problem, they also have limitations such as that they (Lin *et al*., 2012):

- Assume that all input parameters are known with certainty, which is not always the case in real-world problems. Uncertainty makes exact techniques unable to predict future outcomes accurately. This can result in sub-optimal decisions being made based on incorrect assumptions or incomplete information.
- Require a high level of technical expertise and resources to be implemented effectively. The process can be time-consuming and costly, requiring specialised software tools and skilled personnel to operate them.
- May produce results that are too rigid or inflexible to adapt to changing circumstances over time. Real-world problems often involve dynamic conditions such as changes in market demand or resource availability, which may require adjustments beyond what exact models provide.

Monika (2021) stated that despite the limitations associated with exact optimisation techniques, their advantages still outweigh their disadvantages when used appropriately where there is no other alternative. Some studies have been conducted implementing exact methods to solve the production scheduling problem. The following sub-section critically reviews some of the applications of exact techniques in OPMPS.

Production scheduling using linear programming

Johnson (1968) developed an approach to solve the mine production scheduling problem using LP. This method solved the long-term, multi-destination open-pit production scheduling problem. The solution proposed by Johnson (1968) involved breaking down the large multi-period model into smaller sub-models, each focusing on one period at a time to generate optimal results. Moreover, the model took into account discounted values of revenues and costs, various processing methods, and dynamic cut-off grades (Behrang *et al*., 2014). By simplifying the original problem through this decomposition approach, Johnson (1968) was able to overcome the issue of infeasible solutions that arise when using continuous variables to control precedence constraints. However, it should be noted that while this initial model provided satisfactory results for individual periods, it fell short of addressing the entire multi-period problem due to its large size and complexity (Osanloo *et al*., 2008). Gershon (1983) also found that the model did not fully satisfy block extraction precedence and consequently resulted in some suspended blocks above ground level.

Maximising NPV is the common objective when solving the LT mine production scheduling optimisation problem subject to various constraints including mining rate, processing rate, ore grade target, pit slope angle, and Mineral Reserves. The mathematical model of the mine production scheduling problem can be expressed by Equation 2.1 subject to constraints expressed by Equations 2.2 to 2.6 (Osanloo *et al*., 2008):

$$Maximise\ Z = \sum_{n=1}^{N}\sum_{t=1}^{T} v_n^t \times x_n^t, \qquad [2.1]$$

where v_n^t is a discounted value calculated from components of Q_n which are OT_n and WT_n as explained later in Chapter 3.

Subject to:

Processing rate:

$$PC_{min}^t \leq \sum_{n=1}^{N} Q_n \times x_n^t \leq PC_{max}^t;\ \forall\ t = 1, 2, \dots, T. \qquad [2.2]$$

Mining rate:

$$MC_{min}^t \leq \sum_{n=1}^{N} Q_n \times x_n^t \leq MC_{max}^t;\ \forall\ t = 1, 2, \dots, T. \qquad [2.3]$$

Grade target:

$$G_{min}^t \leq \left(\sum_{n=1}^{N} g_n \times Q_n \times x_n^t \Big/ \sum_{n=1}^{N} Q_n \times x_n^t\right) \leq G_{max}^t;\ \forall\ t = 1, 2, \dots, T. \qquad [2.4]$$

Mineral Reserve constraints:

$$\sum_{n=1}^{N} x_n^t = 1;\ \forall\ t = 1, 2, \dots, T. \qquad [2.5]$$

Block precedence constraints:

$$x_b^t - \sum_{s=1}^{t} x_l^s \leq 0;\ \forall l \in \Gamma_b;\ b = 1, 2, \dots, N; \qquad [2.6]$$
$$0 \leq x_n^t \leq 1\ \forall\ t = 1, 2, \dots, T;\ n = 1, 2, \dots, N;$$

Where N is the total number of blocks to be scheduled; T is the maximum number of scheduling periods; n is the block index; v_n^t is the discounted economic value realised by mining and processing block n during period t and is calculated as explained later in Chapter 3; x_n^t is the proportion of material from block n to be extracted in period t; Q_n is the total tonnes of material in block n and is the sum total of OT_n and WT_nwhich are the quantities of ore and waste material comprising the block as explained later in Chapter 3; g_n is the average ore grade of block n; $PC_{min}^t \; and \; PC_{max}^t$ are the minimum and maximum processing rate in period t; $MC_{min}^t \; and \; MC_{max}^t$ are the minimum and maximum mining rate in period t; $G_{min}^t \; and \; G_{max}^t$ are the minimum and maximum average ore grade to be sent to the plant in period t; b is the block index scheduled to be extracted in period t; s is the scheduling period index preceding block n; l is the index of block predecessor of block n; and Γ_b is the set of block predecessor of block b; x_b^t is the proportion of material from block b to be extracted in period t; x_l^s is the proportion of material from block predecessor l to be extracted in period s.

Production scheduling using integer programming

The IP method uses mathematical modelling that restricts all decision variables to be integers. This means that fractional solutions generated in the LP method are discarded, and only integers are considered. Dagdelen and Johnson (1986) were the first to apply IP to optimise block scheduling problems. Their method was based on a directed graph representation of a block model, where nodes represent blocks and arcs indicate their block precedence relationship. Rosas *et al*. (2016) indicated that the graph represents a discrete approach which is ideal for implementing the IP on a block model. The IP model can effectively solve a production scheduling problem providing a complete extraction of a block in one period. However, the IP method becomes more and more impractical when a single block contains heterogeneous material which is usually the case in a real-world block model or due to mining and equipment capacity available in the period. Heterogeneous blocks should be mined according to the material types and quality to reduce dilution and increase ore recovery.

Gleixner (2008) formulated an IP model to solve the mine production scheduling problem. The model assumes a basic block with homogenous ore distribution within each block. A set of a block model containing N blocks is partitioned into one set of ore blocks O to

process and another of waste blocks W to discard. Each block n is assigned a block economic value v_n^t which is the difference between revenue rv_n^t generated and the costs c_n^t associated with mining and processing the block n in period t.

$$v_n^t = \begin{cases} rv_n^t - c_n^t, if\ n = ore; \\ -c_n^t, if\ n = waste. \end{cases}$$

The formulation uses the binary decision variables;

$$x_n^t = \begin{cases} 1, if\ a\ block\ is\ mined\ during\ one\ of\ the\ \ periods\ t; \\ 0, \quad otherwise. \end{cases}$$

In this model, the mining of a block n cannot be spread out over several periods, which means $x_n^t - x_n^{t-1}$ is equal to 1 if and only if block n is completely mined during period t, and 0 otherwise. Equation 2.7 expresses the IP model formulated by Gleixner (2008):

$$Maximise\ Z = \sum_{t=1}^{T}\sum_{n=1}^{N} v_n^t \times (x_n^t - x_n^{t-1}). \qquad [2.7]$$

Subject to constraints expressed by Equations 2.8 to 2.10 (Gleixner, 2008):

Mineral Reserve and block precedence constraints:

$$\begin{aligned} x_n^{t-1} - x_n^t \leq 0, \\ x_n^t - x_j^t \leq 0, \end{aligned} \qquad [2.8]$$

where x_j^t represents the set of block predecessors to block n mined in period t.

Processing rate:

$$PC_{min}^t \leq \sum_{n=1}^{N} Q_n \times x_n^t \leq PC_{max}^t;\ \forall\ t = 1, 2, \dots, T. \qquad [2.9]$$

Mining rate:

$$MC_{min}^t \leq \sum_{n=1}^{N} Q_n \times x_n^t \leq MC_{max}^t;\ \forall\ t = 1, 2, \dots, T. \qquad [2.10]$$

Production scheduling using mixed integer programming

The first mixed integer programming (MIP) model for mine production scheduling was developed by Gershon (1983). The model sought to enhance the model developed by

Johnson (1968) by introducing extra decision variables, to enable partial mining of blocks. The MIP approach offers two benefits when compared to the LP method. Its extraction sequence for mine production scheduling is more practical, as it allows partial mining of blocks under the condition that all preceding blocks have been extracted. This results in only one constraint per block being necessary. MIP also improves IP programming formulations by introducing continuous variables. IP formulations restrict themselves to schedules that force each block to be mined completely during one period. IP models usually cannot be solved on heterogeneous block models. They are only applicable to problems with homogeneous block models where a block can be either ore or waste. Therefore, this restriction may become more unrealistic as the mining of a block or an aggregate of blocks is rather a continuous process than a discrete one. However, one of the drawbacks of the MIP formulations is their inability to handle large problems using commercial software due to having an excessive amount of binary variables (Osanloo *et al*., 2008).

Smith (1999) presented a qualitative description of a MIP formulation for the mine production scheduling problem. The model allows for parcels of blocks to be mined fractionally, but no explicit formulation is given. Within the framework of stochastic mine production scheduling, Menabde *et al*. (2007) also presented a MIP formulation allowing for fractions of blocks to be mined through a continuous variable $y_n^t \in [0,1]$. Fricke (2006) simplified the formulation and gave a generalised MIP version for the mine production scheduling problem, as expressed by Equation 2.11:

$$Maximise\ Z = \sum_{t=1}^{T}\sum_{n=1}^{N} v_n^t \times y_n^t. \qquad [2.11]$$

Equation 2.11 is subject to constraints expressed by Equations 2.12 to 2.17 (Fricke, 2006):

Mineral Reserve and block precedence constraints; a block can be partially mined while satisfying the block precedence relationships:

$$x_n^{t-1} - x_n^t \leq 0, \qquad [2.12]$$

$$x_n^t - \sum_{s=1}^{t} y_l^s \leq 0, \qquad [2.13]$$

$$x_n^t \in \{0,1\},$$

$$0 \leq y_n^t \leq 1. \qquad [2.14]$$

where y_l^s is the proportion of material from block predecessor l to be extracted in period s.

Processing rate:

$$PC_{min}^t \leq \sum_{n=1}^{N} Q_n \times x_n^t \leq PC_{max}^t;\ \forall\ t = 1,2,\dots,T. \qquad [2.15]$$

Mining rate:

$$MC_{min}^t \leq \sum_{n=1}^{N} Q_n \times x_n^t \leq MC_{max}^t;\ \forall\ t = 1,2,\dots,T. \qquad [2.16]$$

Grade target:

$$G_{min}^t \leq \left(\sum_{n=1}^{N} g_n \times Q_n \times x_n^t \Big/ \sum_{n=1}^{N} Q_n \times x_n^t \right) \leq G_{max}^t;\ \forall\ t = 1,2,\dots,T. \qquad [2.17]$$

A summary of exact methods applied to solve the production scheduling problem for open-pit mines is given in Table 2.1 which presents a descriptive, but not exhaustive list of methods.

Table 2.1: Summary of exact techniques applied in solving the production scheduling problem for open-pit mines

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Johnson (1968)	LP using the Dantzig-Wolf decomposition method.	LT	• Grade control. • Mining capacity. • Processing capacity. • Mineral Reserves. • Slope angle.	• Considers dynamic cut-off grades. • Can be applied on a large-size block model. • Generate optimum results for each period individually.	• Fails to solve completely the LT production scheduling problem. • Some blocks will be extracted partially, creating suspension in the air of the un-extracted part of the block. • Has very limited applicability. • Generates too many extended variables.
Roman (1974)	DP using complete enumeration.	LT	• Mining capacity. • Milling capacity. • Slope angle.	• Optimises block sequencing and pit limit simultaneously.	• Does not satisfy mining and milling constraints. • Cannot be implemented on large block models.
Gershon (1983)	MIP	LT	• Precedence block. • Mining capacity. • Plant capacity. • Grade control.	• More practical than Johnson (1968)'s model in block sequencing. • Its mathematical formulation has the potential to reduce computational time. • Only one constraint per block is required.	• Faces difficulties when applied on large block models. • Shows inability to handle dynamic cut-off grades.
Dagdelen and Johnson (1986)	IP using Lagrangian relaxation and sub-gradient multiplier.	LT	• Production capacity. • Blending policy.	• Does not lead to fractional block extraction. • Exploits the structure of the block-sequencing problem.	• Faces difficulties when implemented on large deposits. • Inability to handle dynamic cut-off grade concept.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
				• Reduces the size of the problem through relaxation. • Adjusts the Lagrangian multipliers for blocks on the sides of the pit.	• Prevailing gap problem. • Demonstrates a need for creating pushbacks before the block extraction sequence. • Uses a discrete approach to solve the long-term production scheduling problem, which is a continuous approach. • More than one iteration is required for the creation of the initial fundamental tree.
Tolwinski and Underwood (1992)	DP, AI, and heuristic rules.	ST	• Operational. • Slope angle.	• Delivers practical solutions. • Ability to incorporate many constraints.	• Cannot be implemented on large block models. • Inability to handle dynamic cut-off grades. • Does not guarantee optimum solutions.
Akaike and Dagdelen (1999)	IP using 4D-network relaxation and sub-gradient.	LT	• Grade control. • Production capacity. • Blending policy.	• Takes into account the dynamic cut-off grade concept. • Demonstrates the ability to handle stockpile options.	• Prevailing gap problem. • Faces difficulties when implemented on large block models.
Erarslan and Celebi (2001)	DP algorithm using directed graph, simulative optimisation approach and heuristic rule.	LT	• Mining extraction sequence. • Stockpile and waste dump grade control. • Head grade control. • Mining and milling capacities. • Mining width.	• Simultaneously delivers optimum pit limits and production scheduling. • Provides a realistic estimate of the mining cost. • Considers operational constraints. • Considers the effect of pit volume on the unit cost.	• Encounters difficulties when applied on large block models. • Fails to find an optimal solution.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Caccetta and Hill (2003)	IP using branch-and-cut.	LT	• Volume of material extracted per period. • Mill feed and mill capacity. • Processing different ore types. • Blending. • Logistics. • Stockpiles • Maximum vertical depth. • Minimum pit bottom width and stockpiles.	• Considers several operational constraints. • Obtains a good solution in medium-size deposits.	• It cannot be implemented on large deposits. • Inability to handle dynamic cut-off grades.
Ramazan *et al*. (2005)	IP using fundamental tree generation.	LT	• Production capacity. • Blending policy.	• Elimination of gap problem. • Reduces the number of binary variables. • Delivers higher NPVs compared to any other deterministic algorithms.	• Generating pushbacks is required before scheduling. • Generates fundamental trees after more than one iteration. • The difficulties faced in the implementation reduce its popularity. • The optimality of the solution depends on the optimality of the generated pushbacks.
Gleixner (2008)	MIP using Lagrangian relaxation.	LT	• Mineral Reserves. • Precedence feasibility. • Mining activities.	• Attempts to aggregate processing variables in the LP. • Reduces the size of the production scheduling problem while preventing any loss in the NPV.	• No explicit formulation of the MIP has been proposed. • A large number of variables is needed to model the production plan.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
			• Processing activities.		
Chicoisne *et al*. (2012)	LP using knapsack.	LT	• Block predecessor. • Extraction period. • Processing rate. • Extraction rate.	• Can tackle real-sized instances in practical time. • Can lead to new production scheduling systems, which do not need to schedule bench-phases.	• Cannot accurately capture block precedence. • Blocks scheduled in the same period may be scattered throughout the mine. • May lead to impractical solutions.
Behrang *et al.* (2014)	LP using clustering technique.	LT	• Processing capacity. • Mining capacity. • Block extraction precedence. • Mining processing portion. • Mineral Reserves. • Over and under production rate. • Upper and lower limit for the stockpile.	• Finds the block extraction sequence and respective block destination under grade uncertainty. • Considers the effect of uncertainty in the long-term production scheduling problem. • Reduces the number of variables to make the problem tractable. • Allows a plan to extract extra ore at early stages of the life of the mine.	• The use of a large number of mining cuts does not improve NPV. • Faces weaknesses for real-world case applications.
Moosavi and Gholamnejad (2015)	SMIP	LT	• Mining slope. • Grade blending. • Ore production capacity. • Mining capacity.	• Integrates tonnage uncertainty. • Reduces geological risk in the LT production scheduling problem. • Delivers a better result compared to that of the LP. • Targets higher ore grades in the first periods of mining extraction.	• Fails to consider mining capacity uncertainty. • Tonnage and grade uncertainties can lead to considerable deviation from the production target.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Samavati *et al*. (2018)	LP using the adding cut technique.	LT	• Block precedence. • Operational. • Capacity limitation. • Mineral Resource.	• Converges quickly to a near optimum feasible solution. • Can enhance the production schedule plan obtained with LP; relaxation approach. • Shows a remarkable flexibility compared to other algorithms. • Can potentially be modified to contain minimum resource requirements.	• Difficulties in solving extremely large production scheduling problems.
Gholamnejad *et al*. (2020b)	Integer linear programming.	LT	• Grade blending. • Processing capacity. • Mining capacity. • Local block precedence constraints. • Mineral Reserves. • Equipment movement.	• Generates a more realistic NPV by reducing the costs of equipment movements. • Reduces the number of active benches in any scheduling period. • Increases productivity and efficiency of mining equipment. • Less scattered block sequencing within each bench.	• Needs more computation time. • May reduce the theoretical NPV of a project. • Not applicable to MT and ST scheduling. • Inefficient on a large-size block model.
Joshi and Satpathy (2020)	Exact method based on a sequential branch-and-cut algorithm.	LT	• Slope angle. • Mineral Reserves. • Mining capacity. • Processing capacity. • Metal production capacity.	• Can only generate feasible solutions without lower limit metal production constraints. • Less computational time than iterative longest path algorithm.	• Unable to generate feasible solutions with tight constraints. • Does not consider geological and economic uncertainties. • Generates lower NPVs than iterative longest path algorithm.
	Robust model using iterative	LT	• Slope angle. • Mineral Reserves. • Mining capacity.	• Generates feasible solutions.	• Longer computational time than branch-and-cut.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
	longest path algorithm.		• Processing capacity. • Metal production capacity.	• Can improve branch-and-cut by relaxing resource constraints with a penalty. • Higher NPVs than branch-and-cut.	• Does not consider geological and economic uncertainties.
Maleki *et al*. (2020)	IP	LT	• Mining capacity. • Processing capacity. • Ore grade quality. • Slope angle.	• Delivers multiple simulated block models.	• Unable to deal with uncertainties in geological variables. • The geological information is modelled through a single, smoothed block model. • Real-world size problems are computationally demanding.
Lotfian *et al*. (2021)	Block clustering and GA.	LT	• Slope angle. • Mining capacity. • Processing capacity. • Average grade of the mineralised material. • Predecessor clusters. • Mineral Reserves.	• Improves solution time. • Reduces the number of binary variables. • Delivers feasible solutions.	• Delivers lower NPVs than when using the original block-based scheduling. • Does not consider uncertainty.
Aallaoui *et al*. (2022)	MIP using flexible job-shop theoretical problem with generic minimal time-lags between operations.	ST	• Block precedence. • Operational precedence. • Machine disjunction. • Machine displacement.	• Provides exact solutions for small-scale instances. • Optimise the minimum end date of the last operation. • Short running time with limited constraints.	• Inefficient on large block models. • Does not incorporate all operational constraints • Does not consider uncertainty.

Table 2.1 shows that there is more research done on the LT production scheduling horizon compared to MT and ST production scheduling horizons. This may be due to the importance of the LT horizon of maximising NPV, which is the key criterion for assessing the viability of a mining project (Otto and Musingwini, 2020). Exact techniques applied include LP, IP, MIP and DP using associated methods such as Langrangian relaxation, branch-and-cut, clustering and knapsack. The most common constraints considered when optimising production scheduling are mining capacity, processing or milling capacity, ore grade target, slope angle or block precedence, and the quantity of Mineral Reserves. Each developed exact technique has advantages and drawbacks. Among the advantages include the consideration of stockpiling, blending, cut-off grade, block aggregate, and operational constraints. Some drawbacks include the size of the production scheduling problem, computation time, impractical scheduling solutions, and variable parameters. The next section reviews metaheuristic and stochastic techniques for optimising mine production scheduling.

2.3.2 Metaheuristic open-pit mine production scheduling optimisation techniques

The word metaheuristic is derived from two words "meta" which means "beyond" or "higher level" and "heuristic" which means "to find" or "to discover" by trial and error (Yang, 2011). Therefore, metaheuristic is a higher-level procedure designed to find near-optimal solutions to optimisation problems. Metaheuristic algorithms are generally known to provide solutions to problems that are intractable for exact techniques to solve (Gandomi *et al*., 2013). They employ a combination of local search and global exploration, typically using randomisation to produce a variety of solutions. Some authors use the terms metaheuristics and heuristics interchangeably (Gandomi *et al*., 2013). The incorporation of randomisation allows for a more extensive global search rather than just focusing on local search, making almost all metaheuristic algorithms suitable for nonlinear modelling and global optimisation.

In the last four decades preceding 2011, there has been a steadily growing interest and successful application of NIAs (Yang, 2011). This success can be largely attributed to the fact that these algorithms have been developed by mimicking some of the most successful processes in nature, including biological, chemical and physical processes (Yang, 2011). Nature is a common source of inspiration for metaheuristic algorithms. These algorithms include simulated annealing (SA), ant colony optimisation (ACO), particle swarm

optimisation (PSO) and GA. The introduction of swarm intelligence algorithms like PSO in the 1990s has led to the creation of over a dozen new metaheuristic techniques. These versatile methods have been effectively utilised in various fields, including optimisation, design, scheduling, data mining, and machine intelligence.

Metaheuristics are a valuable tool for generating useful solutions to complex problems within a reasonable timeframe (Kumral and Dowd, 2005). Due to the complexity of real-world problems, it is not feasible to exhaustively search all potential solutions or combinations. Instead, the goal is to find satisfactory solutions within a reasonable timeframe, while considering the variability of inputs. However, there is no guarantee that the optimal solution will be found, and sometimes it is quite difficult to know why a particular algorithm may or may not successfully solve a problem (Yang, 2014). The aim is to have an effective and practical metaheuristic algorithm that produces near-optimal solutions most of the time. It can be expected that among these solutions, some may even be close to optimality, though this cannot be guaranteed.

As compared to exact techniques, metaheuristic techniques provide solutions beyond the reach of exact techniques. As previously mentioned in Sub-section 2.3.1, the application of exact techniques is most suited to problems where input variables are known with certainty and, therefore, not subject to variability. Since exact techniques rely on a fixed set of input parameters, they may not represent a real case (Paithankar and Chatterjee, 2019). However, in reality, input parameters to most real case problems cannot be known with certainty. Metaheuristics involve iteration and employ stochastic operations to modify initial candidate solutions, often obtained through random sampling of the search space (Bandaru and Deb, 2016). In particular, mine production scheduling is a complex and dynamic process, in which uncertainty is inherent and must be considered. Moreover, exact algorithms can be slow to converge to an optimal solution when working on a large dataset.

Blum and Roli (2003) identified two main subprocesses of any metaheuristic techniques, which are diversification and intensification. Diversification involves generating a variety of solutions to fully explore the search space on a global scale. Randomising the process through diversification prevents the solutions from getting stuck at a local optimum and promotes diversity among possible solutions. Conversely, intensification focuses on searching within a local region by utilising information from a current successful solution

in that area. This approach is supplemented by selecting the optimal solution, which leads to convergence towards optimality. By incorporating both diversification and intensification as crucial elements, metaheuristic algorithms can ensure that the optimal solution found converges to optimality (Yang, 2011). A summary of metaheuristic and stochastic methods applied to solve the production scheduling problem is given in Table 2.2 which presents a descriptive, but not exhaustive list of the methods.

Table 2.2: Summary of metaheuristic and stochastic methods applied in solving the production scheduling problem of open-pit mines

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Ravenscroft (1992)	Conditional simulation technique and deterministic algorithm.	LT	• Grade control. • Mining capacity. • Processing capacity. • Block precedence.	• Shows the impact of grade uncertainty on LT production planning.	• It cannot quantify the risk of a project. • It does not generate an optimal solution in the presence of grade uncertainty. • The use of repeated simulations as successive inputs to a mine scheduling process is cumbersome.
Denby and Schofield (1994)	Heuristic method based on the GA.	LT	• Mining capacity. • Processing capacity. • Average grade control. • Extraction sequence.	• Can optimise production scheduling and pit limits simultaneously. • The model converges quite quickly to a near-optimal solution. • The model shows remarkable flexibility to accommodate changes.	• Results produced largely differ from one run to another. • Does not guarantee an optimal solution.
Kumral and Dowd (2005)	SA based on multi-objective SA and stochastic relaxation.	LT	• Access. • Sufficient number of satisfied periods. • Mill and plant capacities. • Ore content.	• Can improve sub-optimal solutions obtained by deterministic methods. • Provides a solution that minimises the extracting cost. • Uses Lagrangian relaxation to reduce the size of the problem.	• Lagrangian parameters reduce the revenue of some ore blocks. • Susceptible to be trapped on a local optimum solution. • Does not reach the maximum profit.
Ferland *et al.* (2007)	PSO-based resource-constrained project scheduling problem.	LT	• Extraction capacity. • Time horizon.	• Generates a better solution if the initial population solution is of good value. • Uses a sub-solution generated from conventional methods as an initial solution.	• Does not seem to be sensitive to the input parameter values of the model. • The quality solution depends on the quality of the initial

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
				• Able to accommodate large-size solution search space to deliver a near-optimal solution. • Generates a large number of different solutions.	values of particles in the swarm. • Indicates no clear impact of modifying the values of some parameters.
Consuegra and Dimitrakopoulos (2009)	SA based on Metropolis algorithm.	LT	• Number of mining sequences. • Slope angle. • Ore production rate. • Waste production rate.	• Able to simultaneously optimise pit limits and production scheduling. • Blocks are assigned according to a probability rank. • Integrates grade uncertainties in the development of the production schedule.	• Loses sensitivity when the number of input mining sequences exceeds 10. • Can be labour-intensive compared to standard conventional techniques.
Khan and Delius (2015)	MIP solution based on PSO.	LT	• Mineral Reserves. • Slope angle. • Mining capacity. • Processing capacity.	• Can produce better solutions in a relatively shorter time compared to deterministic techniques. • The model is well-structured and quite flexible. • Can accommodate multiple variable constraints such as variable slope angle, grade and market uncertainty. • Delivers a very small standard deviation value demonstrating its robustness.	• Susceptible to get trapped in a local optimum solution. • Local variants show a relatively slower rate of convergence. • Computation time depends on the input parameters and the size of the problem. • The standard deviation tends to increase with problem size.
Shishvan and Sattarvand (2015)	ACO based on Max-Min ant system and ant colony system.	LT	• Mining capacity. • Processing capacity. • Average grade. • Variable slope angle.	• Strong capability to improve the quality of the initial sub-optimal solution. • Can simultaneously optimise pit limits and production scheduling problems.	• Does not guarantee to reach the global optimum schedule. • The quality of solutions largely depends on the quality of input parameters.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
				• Capable of considering any type of objective function, non-linear constraints and real technical restrictions. • Able to consider variable slope angles. • Can incorporate some of the constraints into the objective function. • Computation time is less sensitive to the size of the model.	• The procedure is not always proven mathematically to reach the optimal solution. • Small perturbations can occur in the control of the size of pushbacks.
Liu and Zhang (2016)	MIP solution based on PSO.	LT	• Mining horizons. • Mining capacity limits. • Mining sequence requirement. • Sloping requirement.	• Solution time is faster than that of the deterministic techniques. • Delivers good solutions with an average error of 2% compared to the optimum solution.	• Does not guarantee optimal solutions. • Some particles do not update their historical original information over many iterations.
Sari and Kumral (2016)	Heuristic memory and SA.	LT	• Processing capacity. • Transportation cost. • Multiple process destinations.	• Improves sub-optimal solutions generated by exact methods over time. • Almost all parameters can be integrated into the model. • The problem size increases proportionally with the number of parameters. • Average grade and number of blocks send to destinations are kept constant.	• Takes time to reach an optimal solution. • May provide non-optimal solutions when initial parameters are poorly set. • Profile of temperature change is not generic but material-dependent; number of iterations and maximum number of solutions can affect the running time to a great extent.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Alipour *et al*. (2017)	GA based on normalisation method.	LT	• Access. • Mining capacity. • Processing capacity. • Block conservation.	• Demonstrates its efficiency to solve the production scheduling problem. • Able to generate feasible solutions in two-dimensional space. • Delivers an NPV solution with a gap of less than 5% to the optimum solution. • Computing time is relatively shorter.	• Shows limited capability to handle a large number of blocks. • May face challenges in a real-world problem. • Cannot be implemented on a 3D block model space.
Khan (2018)	Global PSO and multi-start PSO.	LT	• Mineral Reserves. • Slope angle. • Mining capacity. • Stochastic processing capacity. • Stochastic metal production.	• Able to handle grade-related uncertainties more efficiently. • Can generate sufficiently good solutions to the problem at hand in a shorter computing time. • Quantifies penalties when violating capacity constraints.	• During the process, the particle positions and velocity are initialised to their initial values. • Likely to get trapped in a local optimum solution. • Due to the one-dimensional nature of the model, solutions may not be feasible in terms of the required slope angles.
Maleki *et al*. (2020)	Stochastic IP model based on simulated block model.	LT	• Mining capacity. • Processing capacity. • Ore grade quality. • Slope angle.	• Delivers multiple simulated block models. • Simultaneously maximises NPV and minimises the risk of losses. • Integrates geological variability and uncertainty. • Improves NPV from exact methods. • Identifies and minimises risks. • High probability of meeting production targets.	• Does not take into consideration the high variability of the mineral of interest and contaminant grades. • Does not capture the behaviour of price and costs over time.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Gholamnejad *et al*. (2020a)	Stochastic chance constraint based on IP.	LT	• Grade blending. • Stockpiling. • Mining capacity. • Processing capacity. • Slope constraint. • Mineral Reserves. • Mining and processing ore Linkage.	• Delivers high NPV while respecting a series of physical and economic constraints. • Incorporates stockpile for storing excess low-grade material. • Better feed grade for the processing plant with a higher NPV compared to exact methods. • Incorporates probability of grade blending.	• Stochastic objective function and constraints cannot be handled directly in the optimisation. • Requires deterministic inputs for stochastic optimisation. • Handles only one stockpile for low-grade material.
Muke *et al*. (2021)	MIP solution based on GA.	LT	• Mining capacity. • Processing capacity. • Grade target. • Block precedence. • Mineral Reserves.	• Delivers NPVs within a comparable range with other solution methods. • Produces practical LT production schedule in 3D space. • Applicable on small and large block models. • Computing time is relatively shorter.	• Need to run multiple times to find the most satisfying solution due to the random selection of the initial population. • The solution is dependent on the quality of the initial population randomly selected.
Alipour *et al*. (2022)	Stochastic differential equations and conventional IP solution based on GA.	LT	• Mining capacity. • Processing capacity. • Block precedence. • Mineral Reserves.	• Integration of forecasted copper prices in the production scheduling problem. • Considers dynamic block economic values. • Dynamic design generates an NPV 43% higher than that of static design. • Incorporates time-dependent discounted values of mining blocks. • Can handle large block models.	• Uses the block economic value as the only factor to affect the production schedule. • Cannot generalise results if not confirmed by future events. • The performance of the model is dependent on the quality of price forecasting. • No visual indicator to show the difference between a static production schedule and a dynamic production schedule.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
Joshi *et al*. (2022a)	SMIP model using branch-and-cut and parametric maximum flow.	LT	• Mineral Reserves. • Slope angle. • Mining capacity. • Processing capacity. • Metal production capacity.	• Uses multiple Mineral Resources model realisations to capture and integrate geological uncertainty. • Generates slightly higher NPVs compared to branch-and-cut. • Evaluates the risk of violating target constraints. • Provides great opportunities to increase project profitability. • Less computational time than exact methods.	• Uses fixed commodity prices. • Does not consider exchange rate fluctuations. • Does not consider mining and processing costs variability. • The performance is dependent on the CPLEX solver.
Joshi *et al*. (2022b)	Stochastic model using integrated parametric graph closure and branch-and-cut algorithm.	LT	• Mineral Reserves. • Slope angle. • Mining capacity. • Processing capacity. • Metal production capacity.	• Improves computational time with a reasonable optimality gap. • Delivers better NPV than the exact method. • Delivers schedules with respect to the Mineral Resources constraint. • Uses multiple simulated orebodies, and accounts for geological uncertainty. • Applicable on large block models where IP takes considerable computational time. • Can provide good initial solutions for other optimisation models.	• Cannot provide a true optimal solution due to the sequential solution strategy. • Does not consider sensitivity on the block size. • Does not incorporate economic and geo-metallurgical uncertainties. • Its performance is dependent on the CPLEX solver.
Levinson and Dimitrakopoulos (2023)	Stochastic mathematical programming and reinforcement learning.	LT and ST	• Accessibility and reserve. • Mineral Reserves. • Mining capacity.	• Jointly optimise LT and ST production scheduling. • Simultaneously considers LT and ST scheduling constraints.	• Does not consider equipment uncertainty and equipment related constraints. • Does not develop weekly, daily and hourly ST schedules.

Developer(s)	Solution method and algorithm	Scheduling horizon	Constraints	Advantages	Main drawbacks
			• Processing capacity. • Destination policy. • Mining recovery constraint.	• Results show significant improvements in the production schedule and financial forecast. • Considers risk of deviating from production capacities. • Ensures LT forecasts are achievable at shorter timescales. • Coarse preconcentration decisions are considered. • The integration production schedule decreases deviations from LT operation by 21% during the first production year.	• Does not consider new grade control information to update the LT schedule. • Skipped MT schedule.
Khan *et al*. (2024)	Heuristic model.	LT	• Mineral Reserves. • Pit slope angle. • Production capacity.	• Better performance in terms of NPV generation. • Delivers an extraction sequence and movement of materials. • Can solve an NP-hard computational problem. • Better computational time as compared to a conventional model.	• Relies on the CPLEX solver. • Limited to use non-variable input parameters.
Cutler and Dimitrakopoulos (2024)	Joint stochastic IP.	LT	• Ramp design. • Ramp predecessor. • Production capacity. • Processing capacity. • Ore grade.	• Ensures feasible equipment access. • Jointly maximises NPV and ramp design.	• Does not consider haulage costs. • Needs to be incorporated into simultaneous optimisation.

Table 2.2 shows that LT production scheduling has been intensively investigated using metaheuristic and stochastic techniques, since the early 1990s. Like the exact techniques shown in Table 2.1, most of the metaheuristic and stochastic techniques (Table 2.2) also do not integrate the production scheduling horizons, except in the work of Levinson and Dimitrakopoulos (2023) who attempted to integrate LT and ST scheduling. Some of the metaheuristic and stochastic techniques applied include SA, conditional simulation, stochastic differential equations, PSO, stochastic chance constraint, ACO, stochastic mathematical programming and reinforcement joint stochastic IP, and GA. The common constraints that these techniques consider include mining capacity, processing or milling capacity, ore grade target, metal production capacity, operational constraint, slope angle or block precedence, and the quantity of Mineral Reserves.

Advantages of these metaheuristic and stochastic techniques include improved NPV, simultaneous optimisation of pit limits and production scheduling, consideration of uncertainty and variable parameters, less computation time, handling of large block models, iterative procedure, and providing feasible solutions. However, the applied metaheuristic and stochastic techniques face some drawbacks that include no guarantee of an optimal solution, susceptibility of being trapped on a local optimal solution, producing different results from one run to another, procedure randomness, and solutions dependent on the initial solution. The next section describes the GA technique, which is the technique employed in this thesis to dynamically optimise the integrated LT and MT production scheduling system.

2.4 Genetic algorithm

2.4.1 Origin of the genetic algorithm

Since the 1950s, researchers have been studying the natural evolutionary process as a means of optimally solving engineering problems (Mitchell, 1999). This involves the improvement of the fitness of a population of candidate solutions using techniques inspired by biological evolution. The applications of this concept have extended across various domains in computer science, including neural networks, machine learning, and evolutionary computation (Gen and Cheng, 2000).

In the 1960s, John Holland, alongside his colleagues and students, pioneered the GA technique as an optimisation technique (Mitchell, 1999). Originally intended to study

natural adaptation, the genetic evolutionary concept was imported into a computer system. It started when Holland (1975) created a specific problem-solving algorithm based on genetics, known as GA. Holland (1975) presented the GA technique as an abstraction of biological evolution and outlined a framework for using it in adaptation. This was a ground-breaking approach to computational evolution using heuristic principles (Mitchell, 1999). During the 1980s, evolutionary computation experienced a resurgence in development and GA became one of its most prominent techniques (Mitchell, 1999). Since then, numerous GA variations have been developed and applied in different fields due to advancements in computer technology.

GA is a technique based on Darwin's concept of natural evolution, designed to aid in solving global optimisation problems (Gen and Cheng, 2000). The GA technique imitates the process found in nature, where genetic traits are passed down through generations. This approach follows the principles of natural selection, known as "survival of the fittest" (Holland, 1975). In essence, it involves a population of species facing environmental challenges and using their fitness to survive. Stronger and fitter individual species will survive while weaker ones perish. As this process continues over time, the population's overall fitness will improve, resulting in a stronger generation. The idea behind GA is inspired by this evolutionary process and its ability to produce fitter and stronger species over generations.

2.4.2 Concept of the genetic algorithm

GA is part of a group of evolutionary algorithms which employ biological evolution and hereditary processes to solve complex optimisation problems. Evolutionary algorithms also known as metaheuristics, imitate the natural behaviour of certain species to find optimal solutions that may be beyond human comprehension (Mitchell, 1999). By utilising probabilistic methods, evolutionary algorithms can quickly reach solutions as compared to exact techniques. Although evolutionary algorithms do not guarantee an optimum solution due to the probabilistic or randomness of the search method approach, they are still valuable in finding near-optimal solutions for problems that are impossible for exact techniques to solve (Verhoeff, 2017).

GA is a technique for searching and optimising potential solutions in a given search space. It operates by evolving the fitness of the population over time. This process involves

expressing the problem as an objective function, which is then maximised or minimised under specific conditions (Gen and Cheng, 2000). Initially, the algorithm randomly generates a set of potential solutions in the form of a chromosome. A chromosome consists of a string of values called genes represented as binary or real numbers. The chromosome's fitness is evaluated based on the fitness value of each gene. To create a new generation of solutions, the algorithm ranks and selects parent chromosomes proportionally to their fitness, meaning higher fitness value chromosomes have a greater chance of being selected (Mitchell, 1999).

Parent chromosomes pass on their fitness characteristics to the offspring using genetic operators such as crossover, mutation and elitism (Mitchell, 1999). The crossover concept is used to pair two parent chromosomes, creating a new generation of chromosomes that is better than the parent's generation. This process involves exchanging certain traits from the parents and also incorporates mutation and elitism to accelerate progress. Mutation identifies weaker genes in the chromosome and replaces them with stronger ones, while elitism selects the best and fitter chromosomes (Gen, *et al*., 2008). These genetic operations are repeated on the next generation to create another generation until an optimal point is reached whereby no more improvement can be made.

2.4.3 Optimisation procedure of the genetic algorithm

The algorithm is structured in such a way as to seek an optimum solution through a step-by-step procedure over a search space. Each potential solution is represented in the form of a chromosome of binary or real numbers. A GA optimisation procedure works as follows (Mitchell, 1999):

- Step 1 begins by randomly creating an initial population of candidate solutions $G(t) = \{x_1^t, x_2^t, \dots, x_N^t\}$ in the form of chromosomes. Each chromosome or candidate solution represents a potential and feasible solution to the objective function.

- Step 2 evaluates each chromosome in the population using a fitness function $f(t)$. The fitness function quantifies how well each chromosome performs with respect to the objective function. All the chromosomes are then ranked according to their fitness, from the fittest to the least fit. Higher fitness values indicate better solutions.

- Step 3 creates a new population of solutions by applying the following genetic operators that alter the genetic composition of offspring:
    - Selection: consists of selecting $m \leq N$ chromosomes from the current population to serve as parents for the next generation and participate in the reproduction of offspring. The selection process is typically guided towards choosing individuals with higher fitness values, as they are more likely to contribute to the next generation. Common selection techniques include roulette wheel selection, tournament selection, and rank-based selection.
    - Crossover: consists of creating offspring chromosomes by combining genetic material from selected parent chromosomes. The crossover process mimics genetic recombination in biological reproduction. Depending on the chosen crossover method, segments of genetic material from two parent chromosomes are exchanged to create one or more offspring chromosomes.
    - Mutation: consists of introducing random changes to the offspring chromosomes to maintain genetic diversity in the population. Mutation helps prevent premature convergence to sub-optimal solutions by exploring new regions of the search space. Mutation typically involves randomly altering individual genes in the offspring chromosomes that have a low probability associated with them in being selected.
- Step 4 replaces the least fit chromosomes in the current population with the newly created offspring. The replacement strategy can vary, but it often involves maintaining a fixed population size by selecting individuals for replacement based on their fitness values. Elitism may also be employed, preserving the best-performing individuals from the current population to ensure that valuable genetic material is not lost.
- Step 5 repeats selection, crossover, mutation, and replacement steps for a predetermined number of generations or until a termination criterion is met. Termination criteria can include reaching a satisfactory solution, exceeding a maximum number of generations, or exhausting computational resources.

A sequential application of these steps allows GA to efficiently explore the search space and converge towards near-optimal solutions for a wide range of optimisation problems. Mitchell (1999) stated that GA is particularly well-suited to solve problems with complex

search spaces, non-linear constraints, and multiple objectives. Figure 2.7 shows a general structure of GA illustrating the general implementation of GA, where $P(t)$ and $C(t)$ are parents and offspring, respectively, in the current generation t.

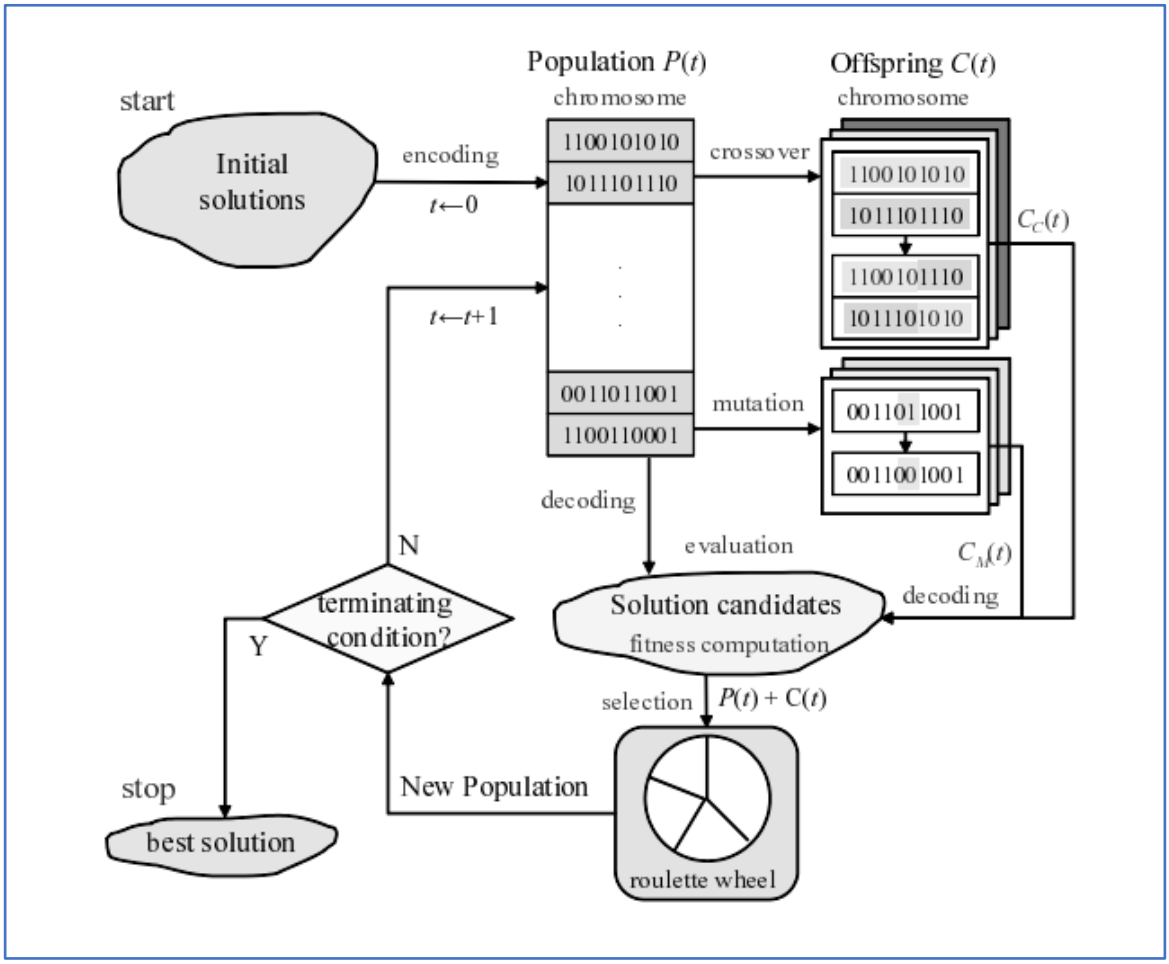


Figure 2.7: General structure of the genetic algorithm
(Source: Gen, *et al*., 2008)

2.4.4 Exploitation and exploration search

Search is a widely used problem-solving method for problems that do not have a predetermined path to a solution. It can be carried out using either blind or heuristic strategies (Gen *et al.*, 2008). Blind strategies do not take into account information about the specific problem, while heuristic strategies use additional knowledge to guide the search towards the best possible directions. There are two main considerations when employing search strategies. Firstly, exploring the search space and secondly, exploiting the optimal solution (Mitchell, 1999). The exploration search strategy consists of randomly exploring the search space regardless of areas of possible promising solutions. The

exploitation search strategy consists of improving the current solution without fully exploring the search space (Gen *et al.*, 2008). Figure 2.8 illustrates the mapping from chromosomes to solutions.

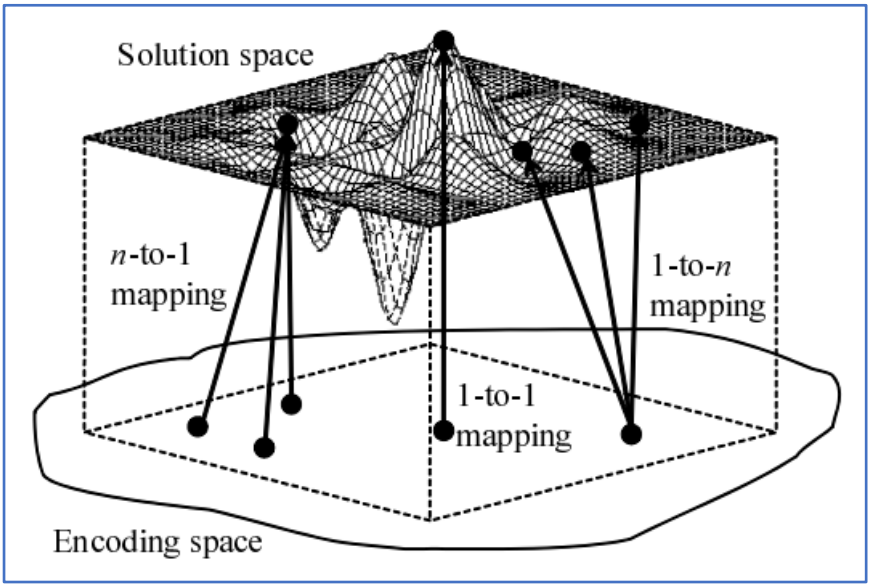


Figure 2.8: Mapping from chromosomes to feasible solutions in a solution space
(Source: Gen, *et al*., 2008)

GA is a flexible optimisation technique that merges directed and random search techniques to achieve a balance between exploration and exploitation approaches within the search space (Goldberg, 1989). That means GA uses a multi-directional or population-based approach by maintaining a population of potential solutions. The multi-directional strategy allows the algorithm to escape local optimum solutions (Goldberg, 1989). As more optimal solutions emerge, the crossover operator adapts to explore their immediate surroundings. This means that the effectiveness of the crossover is influenced by the genetic environment. Exploitation can merely be performed through a mutation operator which adapts to exploit the emerged solutions to improve weak genes.

It is important to understand how the genetic operators work during the implementation of the algorithm to envisage the nature of the outcome (Venkatraman and Yen, 2005). Genetic operators are also necessary in determining the version of GA being developed. Goldberg (1989) stated that GA versions are based not only on the adaptation of the procedure to the problem at hand but also on how the genetic operators function. For instance, one version may consider encoding with binary or Boolean numbers instead of working with real

numbers (Mitchell, 1999). The resulting offspring from the crossover and mutation in Step 3 (Section 2.4.3) may replace their respective parent chromosomes in one version rather than replacing the least fit chromosomes in another version (Mitchell, 1999). There exist different methods of crossover and mutation that also differ from one version to another.

2.4.5 Selection operator

The selection operator consists of selecting chromosomes from a population according to their fitness for reproduction. The fitter the chromosome, the higher the chance that it will be selected for breeding. The selection directs the GA to search towards promising regions in the search space. Goldberg (1989) proposed some selection techniques which include the roulette wheel, tournament selection, sharing, ranking and scaling, and steady-state reproduction.

The roulette wheel selection or fitness proportionate selection, as proposed by Holland (1975) is the most common selection method. The goal is to determine the selection probability for each chromosome proportional to the fitness value. The roulette wheel is divided into a number equal to the number of chromosomes, where the size of the portions corresponds to the probability values. The wheel is spun several times equal to the population size, each time selecting a chromosome to take part in the reproduction phase. The GA probabilistic selection method can be expressed using Equation 2.18 (Gen and Cheng, 2000):

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i}, \qquad [2.18]$$

where p_i is the probability of the chromosome being selected; f_i is the fitness of chromosome i; and N is the number of chromosomes in the population.

2.4.6 Crossover operator

Gen *et al.* (2008) stated that using different genetic operators has a very large influence on the performance of the GA technique. Therefore, it is important to examine different genetic crossover operators. Several crossover operators have been proposed, such as single-point crossover, double or partial-mapped crossover, order crossover, cycle crossover, position-based crossover, and heuristic crossover (Gen *et al.* 2008).

Single-point crossover

Single-point crossover (SPX) is a genetic operator used in evolutionary algorithms to combine information from two parent chromosomes and create offspring with new genetic characteristics. SPX process involves selecting a random point along the length of both parent chromosomes, and then swapping the substrings that follow the point between the two parents (Verhoeff, 2017). The resulting offspring inherit part of their genetic features from each parent, enabling the exploration of the solution space due to different combinations of features present in the parent chromosomes. Figure 2.9 illustrates the genetic procedure of SPX.

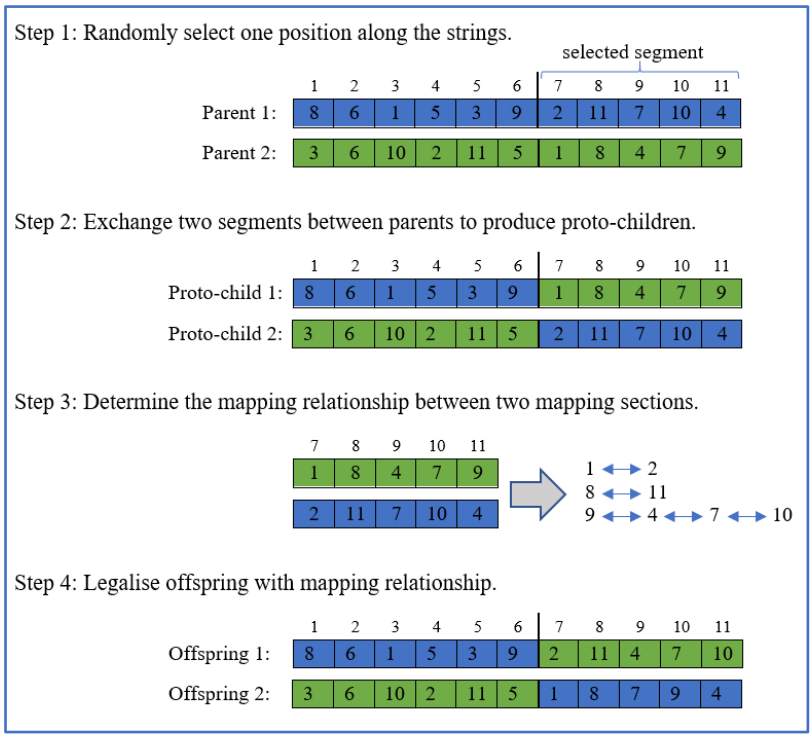


Figure 2.9: Illustration of a single-point crossover genetic procedure
(Source: adapted from Gen *et al*., 2008)

In Figure 2.9, Parent chromosomes 1 and 2 are divided into two segments at Position 6. The segment on the right is randomly selected in Step 1. Step 2 crosses over the selected segment between parents to produce Proto-children 1 and 2. Step 3 determines the mapping relationship between genes of crossed-over segments, where, for example, Gene 1 is mapped to Gene 2 at Position 7; Gene 8 is mapped to Gene 11 at Position 8. Step 4 legalises segments to avoid duplication of genes, and produces Offspring 1 and 2. Offspring 1 inherits from Parent 2, Genes 4, 7 and 10 at Positions 9, 10 and 11; respectively. Offspring

2 inherits from Parent 1, Genes 7, 9 and 4 at Positions 9, 10 and 11; respectively. Mixing genetic features using single-point crossover can promote diversity within the population and increase the chances of generating fitter offspring through recombination. The SPX method effectively mimics natural breeding processes found in biological systems, where new variations emerge through reproduction by combining genes from two distinct sources (Gen *et al.,* 2008).

Partial-mapped crossover

Partial-mapped crossover (PMX) or double-point crossover was proposed by Goldberg and Lingle (1985). In this process, both parent chromosomes are randomly divided into segments which are then exchanged between parents to create one or more offspring. The selected substrings are then mapped onto each other based on their positions within the chromosome. Any overlapping genes are exchanged between the parent chromosomes to create new offspring with unique combinations of genetic features. Gen *et al.* (2008) argued that PMX genetic operator improves the exploration of the search space by promoting genetic diversity at various gene positions. The genetic operator also potentially improves the performance of evolutionary algorithms in finding optimal solutions to complex problems (Gen *et al.,* 2008). Figure 2.10 illustrates the genetic procedure of the PMX.

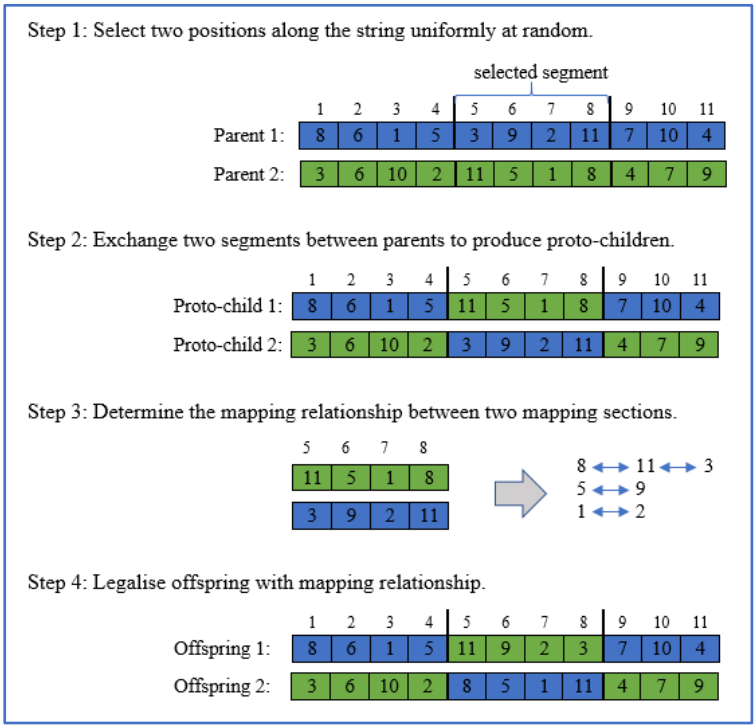


Figure 2.10: Illustration of a partial-mapped crossover genetic procedure
(Source: adapted from Gen *et al*., 2008)

In Figure 2.10, Parent chromosomes 1 and 2 are divided into three segments at Positions 4 and 8, and the segment in the middle is randomly selected in Step 1. Step 2 crosses over selected segments consisting of four genes, to produce Proto-children 1 and 2. Step 3 determines the mapping relationship between genes of crossed-over segments, where, for example, Gene 11 is mapped to Gene 3 at Position 5, and Gene 8 is mapped to Gene 11 at Position 8. Step 4 legalises segments to avoid duplication of genes, and produces offspring 1 and 2. Offspring 1 inherits from Parent 2, Genes 11 and 3 at Positions 5 and 8; respectively. Offspring 2 inherits from Parent 1, Genes 8 and 11 at Positions 5 and 8; respectively.

Order crossover

Order crossover (OX) also known as gene reordering was proposed by Davis (1987). OX can be viewed as a kind of variation of PMX. In this process, two parent chromosomes exchange segments of their genetic features to generate new offspring with a different order of genes. The genetic procedure of OX involves selecting two random points along the parent chromosomes and swapping the genetic features between these points to create new offspring. Figure 2.11 illustrates the genetic procedure of OX.

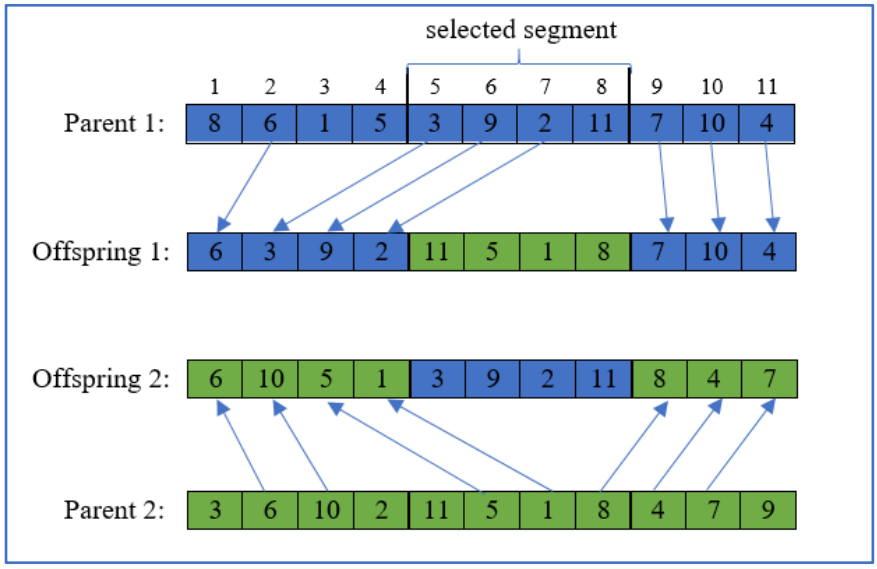


Figure 2.11: Illustration of an order crossover genetic procedure
(Source: adapted from Gen *et al*., 2008)

In Figure 2.11, Parent chromosomes 1 and 2 are divided into three segments at Positions 4 and 8, and the segment in the middle is randomly selected for crossing over to produce Offspring 1 and 2. Offspring 1 inherits the selected segment from Parent 2 and complementary genes from Parent 1. Offspring 2 inherits the selected segment from Parent 1 and complementary genes from Parent 2. OX operation plays a crucial role in exploring new search spaces and generating diverse solutions by introducing new combinations of genes that are not present in either of the parent chromosomes. Through this procedure, OX facilitates the exploration of the search space while preserving characteristics from both parents (Gen *et al.*, 2008).

2.4.7 Mutation operator

The mutation operator is a crucial component in GA-based algorithms as it plays a key role in introducing variation and promoting both the exploitation of chromosomes and the exploration of the search space (Muke *et al*., 2021). In this process, individual genes along the parent chromosomes are randomly selected and altered to generate new offspring with potentially improved characteristics. Gen *et al.* (2008) stated that by randomly changing specific genes, the mutation operator helps prevent premature convergence to sub-optimal solutions by maintaining diversity and allowing for unexpected combinations of traits to emerge. Mutation can occur at any gene position in a chromosome with a probability of 1/L, where L represents the length of the chromosome (Verhoeff, 2017).

The stochastic nature of mutation serves as a mechanism for introducing new genetic characteristics into the population, ensuring that the algorithm continues to explore different regions of the solution space in search of optimal solutions. Through this procedure, the mutation operator contributes to the overall effectiveness and robustness of genetic algorithms in solving complex optimisation problems across various domains. Various genetic mutations have been proposed including inversion mutation, insertion mutation, swap mutation, and heuristic mutation (Gen *et al.*, 2008).

Inversion mutation

Inversion mutation is a type of genetic mutation that involves the rearrangement of genetic features along a chromosome. This phenomenon occurs when a segment of the chromosome breaks and reattaches in reverse orientation, leading to the inversion of the affected region (Gen *et al.*, 2008). Figure 2.12 illustrates the genetic procedure of the inversion mutation.

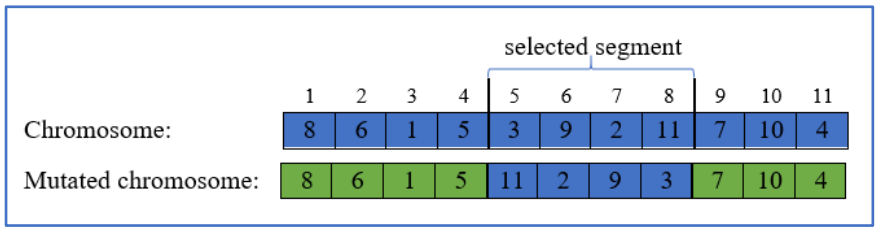


Figure 2.12: Illustration of an inversion mutation genetic procedure
(Source: adapted from Gen *et al.*, 2008)

In Figure 2.12, the chromosome is divided into three segments at Positions 4 and 8, where the segment in the middle is randomly selected for mutation. The order of genes in the selected segment [3,9,2,11] is reversed into [11,2,9,3] to produce the mutated chromosome. In this case, the location of the genes affects the fitness of the chromosome.

Insertion mutation

Insertion mutation is a type of genetic mutation that involves inserting one or more genes along a chromosome, resulting in the alteration of the genetic code to improve its fitness. This process can also occur during gene repositioning, where a gene changes its position and finds its best position along the chromosome from where it contributes the most to the fitness value of the chromosome (Gen *et al.*, 2008). Figure 2.13 illustrates the genetic procedure of the insertion mutation.

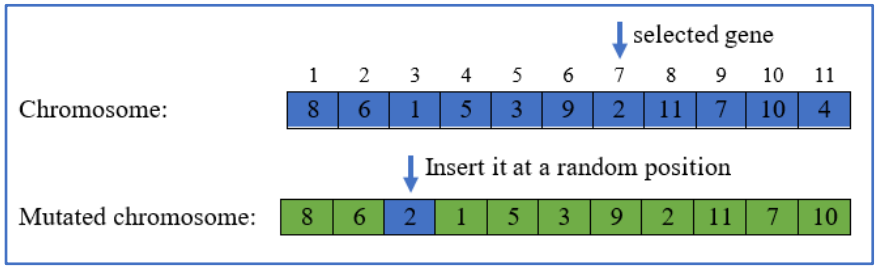


Figure 2.13: Illustration of an insertion mutation genetic procedure
(Source: adapted from Gen *et al*., 2008)

In Figure 2.13, Gene 2 at Position 7 is randomly selected for mutation. The gene is then inserted at Position 3 which is considered as its best position along the chromosome.

Swap mutation

Swap mutation is a type of genetic mutation that consists of identifying two genes along a chromosome, and then swapping their positions. This technique improves the fitness of the chromosome when some genes can contribute to the fitness of a chromosome if they occupy certain positions other than their current positions. However, swap mutation does not bring any effect if both genes have the same fitness value (Gen *et al.*, 2008). Figure 2.14 illustrates the genetic procedure of the swap mutation.

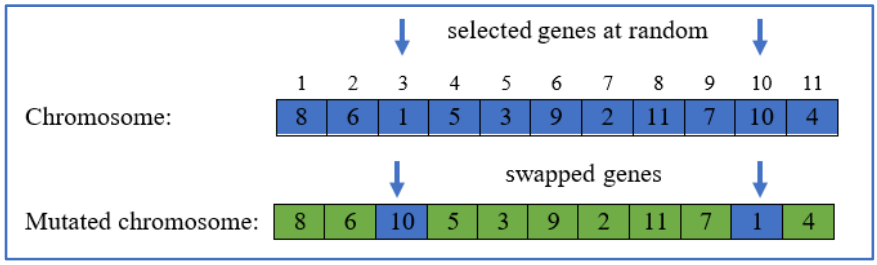


Figure 2.14: Illustration of the swap mutation genetic procedure
(Source: adapted from Gen *et al*., 2008)

In Figure 2.14, Genes 1 and 10 are randomly selected at Position 3 and 10, respectively. Gene 1 moves to Position 10 and Gene 10 moves to Position 3 to produce a mutated chromosome with improved fitness.

Heuristic mutation

Heuristic mutation proposed by Gen and Cheng (2000), is a powerful genetic operator for improving chromosome diversity. During heuristic mutation, new potential solutions are generated by modifying a chromosome to produce multiple mutated chromosomes. This

process involves applying random permutation change of gene positions along a chromosome, such as swapping some gene positions, to explore new regions of the solution space and potentially find better solutions. The purpose of heuristic mutation is to introduce diversity into the population and prevent premature convergence on sub-optimal solutions. By allowing exploration beyond what is dictated by crossover operations, heuristic mutation can help improve the overall performance of genetic algorithms in finding optimal solutions to complex problems (Gen *et al.*, 2008). Figure 2.15 illustrates the genetic procedure of the heuristic mutation.

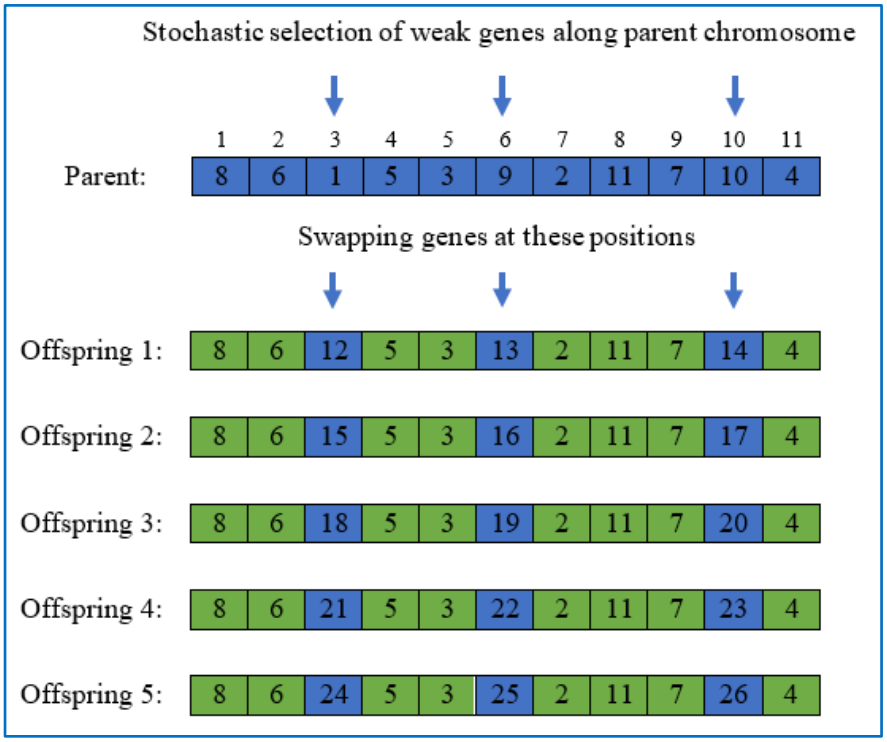


Figure 2.15: Illustration of a heuristic mutation genetic procedure
(Source: adapted from Gen *et al.*, 2008)

In Figure 2.15, Genes 1, 9 and 10 are selected at Positions 3, 6 and 10, respectively. These genes are then swapped with stronger genes to produce five mutated chromosomes which are fitter than the parent chromosome. As stated earlier, crossover and mutation operations are repeated in the next generation to create another generation until no more improvement in fitness values of chromosomes can be made.

2.4.8 Genetic algorithm problem representation

Gen *et al*. (2008) admitted that one of the most challenging aspects of implementing GA is to define how to encode or convert a given problem solution into a chromosome. Encoding is crucial because it directly impacts the efficiency and effectiveness of the algorithm search process. The performance of GA is dependent on how the chromosome is represented with specific genes that encapsulate the decision variables of the problem. Another important key component in this process is the fitness function, which evaluates how well a particular chromosome aligns with the desired objective or goal of the optimisation problem. The encoding of the problem at hand has to be thoroughly examined. To effectively implement GA, different encoding methods have been developed for specific problems. These methods can be categorised as follows based on the type of symbols used as genes:

- Binary encoding: in many cases, binary encoding is commonly used to represent chromosome solutions, where each gene within a chromosome corresponds to a particular decision variable that takes either 0 or 1 values (Gen *et al.*, 2008). However, this type of encoding has been found to have significant flaws when it comes to function optimisation problems. These issues result from the presence of hamming cliffs, where a notable distance in the genotype space corresponds to minimal distances in the phenotype space (Gen *et al.*, 2008).
- Real number encoding: in real-world scenarios, representing solutions using binary encoding can often be challenging (Mitchell, 1999). Real number encoding is the optimal choice for function optimisation problems, as it has consistently shown superior performance compared to binary encoding methods in both function optimisations and constrained optimisations (Mitchell, 1999).
- Integer permutation encoding: this method is a fitting option for combinatorial optimisation problems, as it involves finding the optimal arrangement or combination of items with certain limitations (Mitchell, 1999). Thus, integer permutation encoding may be the most logical approach for tackling combinatorial and complex problems (Mitchell, 1999).
- General data structure encoding: in tackling more intricate real-life problems, it is recommended to use an appropriate data structure, similar to a series of genes, to accurately represent the problem at hand (Mitchell, 1999). This data structure can

take the form of an array gene or a more complex variation, such as one dimensional or multi-dimensional space. Although the one-dimensional encoding method is commonly used in most practices, real-world problems often have multi-dimensional structures. Thus, it is practical to employ a multi-dimensional encoding method to represent these solutions (Gen *et al.*, 2008).

2.4.9 Genetic algorithm handling of constraints

When using GA for constrained objective functions, addressing constraints is essential. Most real-world problems cannot often be tackled without including constraints on the decision variables. Constraints define the relationship between decision variables and parameters and outline the area of feasible solutions. Solutions that do not meet conditions defined in the constraints are considered infeasible solutions. In some cases, the GA model employs genetic operators discussed previously to transform infeasible chromosome solutions into feasible chromosome solutions. In general, to deal with infeasible chromosomes that do not meet the constraint conditions, various methods have been suggested. The methods can be broadly categorised into four categories namely rejecting strategy, repairing strategy, modifying genetic operator strategy and penalising strategy.

The rejecting strategy is a common choice in various GA applications, as it eliminates all infeasible chromosomes produced during the evolutionary process. Gen *et al*. (2008) acknowledged that the rejecting strategy can be effective when the feasible search space is convex and makes up a significant portion of the overall search space. However, it is limited where the initial population consists entirely of infeasible chromosomes which may require modification to comply with constraints. Otherwise, crossing an infeasible region, especially in non-convex feasible search spaces can often lead to infeasible solutions (Goldberg, 1989).

The repairing strategy involves transforming an infeasible chromosome into a feasible one using a specific procedure. This method is often straightforward for various combinatorial optimisation problems. Gen *et al*. (2008) demonstrated the effectiveness of this approach by testing it on a diverse range of constrained combinatorial optimisation problems. Their results revealed that the repair strategy outperformed other strategies in terms of both speed and performance. The effectiveness of the repair strategy relies on having a definitive process for turning a non-viable offspring into a viable one. However, this approach has

its limitations as it is tailored to each problem, requiring a unique algorithm to be created. In some cases, the difficulty of fixing non-feasible chromosomes may be just as challenging as solving the initial problem itself (Gen *et al*., 2008).

The modifying genetic operator strategy is another effective method, for addressing feasibility concerns, used to create tailored representations and design genetic operators specifically for the problem at hand. According to Goldberg (1989), modifying strategies tend to be more consistent than other strategies that rely on penalty methods. This strategy has gained popularity in applying the GA process because of the success of using problem-specific representations and specialised operators in various fields (Gen *et al*., 2008). Nevertheless, modifying the genetic operator strategy limits the search within a feasible region (Gen *et al*., 2008).

The penalising strategy has the advantage of never generating infeasible solutions, but the downside is that it does not consider any points outside of the feasible regions (Gen *et al*., 2008). This can be problematic for highly constrained problems, as the proportion of infeasible solutions in the population may be significant. In such cases, finding feasible solutions becomes challenging if the genetic search is limited to the feasible regions only. To address this limitation, Goldberg (1989) suggested using constraint management techniques that allow for movement through infeasible regions of the search space. Goldberg (1989) found that this strategy leads to faster optimisation and better final solutions compared to strictly searching within feasible regions.

2.5 Summary of Chapter 2

Mine production scheduling is developed in a sequential mode and includes LT scheduling, MT scheduling, and ST scheduling. The interactive nature of the scheduling is reflected in the sense that the scheduling horizons must link to one another. Each scheduling horizon is formed by and derived from the preceding horizon. Preceding scheduling horizons must inform succeeding scheduling horizons, which must provide feedback to the preceding horizons to ensure an integrated mine scheduling process. LT production scheduling delivers a yearly extraction sequence that covers the LOM. MT production scheduling breaks down the LT schedule and delivers a half-yearly, quarterly or monthly extraction sequence that covers one to five years of the LT schedule. ST production scheduling breaks

down the MT schedule and delivers monthly, weekly, daily, shift-by-shift and hourly extraction sequences that cover three months to one year of the MT schedules.

Existing literature and available research focus on independently optimising the LT, MT and ST production schedules, except for the work done by Levinson and Dimitrakopoulos (2023) which jointly optimises a LT and ST production schedule. Most of the optimisation techniques are based on OR techniques. Early application of OR techniques on production scheduling employed exact techniques. These techniques use analytical properties to generate a sequence of points that ultimately converge to an optimal solution. However, exact techniques involve decision variables that are dependent on the initial condition of parameters and do not consider uncertainties in the optimisation process. Their solution time can increase exponentially when solving complex problems such as mine production scheduling problem. This has led to the development and rise of metaheuristic techniques that can find near-optimum solutions faster and account for uncertainty. Metaheuristic approaches generate near-optimum solutions by mimicking natural phenomenon processes. Metaheuristics involve iteration and employ stochastic operations to modify initial candidate solutions, often obtained through random sampling of the search space.

The description of GA suggested that it is a metaheuristic technique used for searching and optimising potential solutions in a given search space. Its optimisation procedure works in five steps to evolve the fitness of a population over time using genetic operators. Initially, GA randomly generates a set of potential solutions in the form of chromosomes. The fitness of chromosomes is evaluated based on the fitness value of each gene. Crossover and mutation are used to produce new chromosomes of which fitness values are improved. It was found that GA is particularly well-suited for problems with complex search spaces like OPMPS. The next chapter (Chapter 3) presents a formulation of OPMPS which was then solved using a GA approach in Chapter 4.

3. MIXED INTEGER PROGRAMMING MODELLING OF THE INTEGRATED LONG-TERM AND MEDIUM-TERM MINE PRODUCTION SCHEDULING SYSTEM

3.1 Overview of Chapter 3

This chapter presents a MIP model that integrates LT and MT mine production scheduling. Section 3.2 describes how the model integrated the two production scheduling horizons using connecting links. Connecting links consist of decision variables and parameters looping between LT and MT production schedules. The section also presents notations, decision variables and parameters for the mine production scheduling.

The MIP model integrating LT and MT production scheduling is presented in Section 3.3. The framework of the model explicitly demonstrates the looping flow of the integrated scheduling system. Firstly, the section presents MIP mathematical models for independently optimising LT and MT scheduling horizons. Then the section presents an integrated LT and MT production scheduling mathematical model. The objective function was formulated to maximise NPV while adhering to various constraints inherent in a mining operation.

Another important aspect of the integrated mine production scheduling process is mine compliance and reconciliation which ensures consistency of scheduling horizons. Compliance of mine to schedule is principally about tracking and assessing the correlation between what is mined operationally from the pit versus what was scheduled. Section 3.4 discusses temporal and spatial reconciliation of the integrated scheduling system.

3.2 Modelling of the mixed integer programming mine production scheduling system

Modelling of the mine production scheduling system should ensure that uncertainty associated with the production scheduling problem is being considered (Vallejo and Dimitrakopoulos, 2018). The source of uncertainty in production scheduling is generally from the variability of the geological characteristics of the orebody, mine production rate, and ore processing rate (Leite and Dimitrakopoulos, 2007). A combination of these uncertainties creates significant mine scheduling challenges that may be computationally expensive to solve using deterministic optimisation techniques. It should be noted that

many other uncertain variables exist. Most notably, the uncertainty in financial variables such as mineral commodity price, exchange rate and input costs could have a significant influence on the NPV of a project.

Senécal and Dimitrakopoulos (2020) presented a mine production scheduling system that accounts for uncertainty associated with production scheduling. The uncertainty was introduced by employing deviations in mining and processing rates that may occur from the implementation of the schedule. These deviations are caused by the shortage or surplus of the quantity of ore and waste mined and the quantity of ore sent to the plant for processing. Integrating uncertainty does not mitigate deviations directly but rather produces a more realistic or practical schedule which, in turn, improves the probability of actual mining operations adhering to the schedule. The mitigation of scheduling deviations can eventually improve the alignment of the LT, MT, and ST production schedules.

This thesis presents a MIP model integrating LT and MT schedules while considering uncertainty associated with meeting targets set for mining rate, processing rate and ore grade. Sub-section 3.2.1 describes the notations, decision variables and parameters that were used for the MIP model formulation. The MIP model was solved using the GA approach, thus, making the results stochastic. The GA approach is presented in Chapter 4. The combined MIP model and GA approach was then coded in Python programming language, of which code is presented in Appendices 8.1 and 8.2.

3.2.1 Mathematical notation of the mixed integer programming mine production scheduling model

Regardless of whether metaheuristic or exact techniques are used to optimise production scheduling, a mathematical expression of the model is essential to describe the construction of the model employed (Moosavi and Gholamnejad, 2015). MIP has significant potential for optimising the OPMPS process where the objective function is to maximise the discounted profit (Ramazan and Dimitrakopoulos, 2004). In this thesis, mathematical models were developed to express an MIP model that can incorporate linking constraints into the mine production scheduling process.

Two basic notions required to set up the MIP model, are decision variables and parameters. Decision variables are controllable inputs whose values are determined by solving the mathematical model. Parameters are known as uncontrollable inputs whose values must be

added to the mathematical model by the user (Rardin, 2017). The descriptions and notations of the indices, set of entities, and parameters used for the MIP formulation are presented in Tables 3.1 to 3.3.

Table 3.1: List of indices used in the MIP mine production scheduling model

Index	Description
i	Mining or source location, $i \in \mathcal{M}$.
e	Destination location, $e \in \mathcal{D}$.
t	Yearly LT scheduling periods, $t \in T^{\ell t}$.
$\mathfrak{t}$	Half-yearly MT scheduling periods, $\mathfrak{t} \in T^{mt}$.
n	Block identification, $n \in \{1, \dots, \mathcal{N}\}$.
c	Material class attribute, $c \in \mathcal{C} = \{ore, waste\}$.
bp	Block predecessors, $bp \in \Gamma$.
lt	LT duration of annual scheduling periods.
mt	MT duration of half-yearly scheduling periods.

Table 3.2: Sets of entities used in the MIP mine production scheduling model

Notation	Description
$\mathcal{M}$	Set of mining or source locations.
$\mathcal{D}$	Set of destination locations.
$\mathcal{C}$	Set of material classes, c.
$\mathcal{N}$	Set of blocks in the block model for each mining location, i.
Γ	Set of direct block predecessors for each block, n.
$T^{\ell t}$	Set of LT yearly scheduling periods.
T^{mt}	Set of MT half-yearly scheduling periods.

Table 3.3: Parameters used in the MIP mine production scheduling model

Parameter	Description
MT_n	Total mass of a block, $n \in \mathcal{N}$.
OT_n	Quantity of ore material contained in a block, $n \in \mathcal{N}$.
OC_n	Total mass of metal ore content in a block, $n \in \mathcal{N}$.
WT_n	Quantity of waste material contained in a block, $n \in \mathcal{N}$.
$g_{n,c}$	Grade quality of ore material class, $c \in \mathcal{C}$ in a block, $n \in \mathcal{N}$.
d	Economic discount rate per annum.
$V_{n,t}^{lt}; V_{n,\mathfrak{t}}^{mt}$	Economic value of a block, n in period, $t \; and \; \mathfrak{t}$, respectively.
$v_n^{lt}; v_n^{mt}$	Discounted economic value generated by extracting block, n in period, $lt \; and \; m\mathfrak{t}$; respectively. Where, $v_n^{lt} = V_{n,t}^{lt}/(1+d)^t$; $v_n^{mt} = V_{n,\mathfrak{t}}^{mt}/(1+d/2)^t$.
$p_{c,i}^{lt};\; p_{c,i}^{mt}$	Discounted selling price $P_{c,i,t}^{lt}, P_{c,i,\mathfrak{t}}^{mt}$, per unit of ore material class produced and recovered at mining location, i in period, $t \in T^{\ell t}$ and

Parameter	Description
	$\mathcal{t} \in T^{mt}$; respectively. Where, $p_{c,i}^{lt} = P_{c,i,t}^{lt}/(1+d)^{t}$; $p_{c,i}^{mt} = P_{c,i,\mathcal{t}}^{mt}/(1+d/2)^{\mathcal{t}}$.
$cm_{c,i}^{lt}$; $cm_{c,i}^{mt}$	Discounted unit mining costs $CM_{c,i,t}^{lt}, CM_{c,i,\mathcal{t}}^{mt}$for mining ore and waste material class, $c \in \mathcal{C}$ from location, i in period, $t \in T^{\ell t}$ and $\mathcal{t} \in T^{mt}$; respectively. Where, $cm_{c,i}^{lt} = CM_{c,i,t}^{lt}/(1+d)^{t}$; $cm_{c,i}^{mt} = CM_{c,i,\mathcal{t}}^{mt}/(1+d/2)^{\mathcal{t}}$.
$cp_{c,i}^{lt}$; $cp_{c,i}^{mt}$	Discounted unit processing costs $CP_{c,i,t}^{lt}, CP_{c,i,\mathcal{t}}^{mt}$for processing ore material class, $c \in \mathcal{C}$ from location, i in period, $t \in T^{\ell t}$ and $\mathcal{t} \in T^{mt}$; respectively. Where, $cp_{c,i}^{lt} = CP_{c,i,t}^{lt}/(1+d)^{t}$; $cp_{c,i}^{mt} = CP_{c,i,\mathcal{t}}^{mt}/(1+d/2)^{\mathcal{t}}$.
$cs_{c,i}^{lt}$; $cs_{c,i}^{mt}$	Discounted unit selling costs $CS_{c,i,t}^{lt}, CS_{c,i,\mathcal{t}}^{mt}$for selling ore material class, $c \in \mathcal{C}$ from location, i in period, $t \in T^{\ell t}$ and $\mathcal{t} \in T^{mt}$; respectively. Where, $cs_{c,i}^{lt} = CS_{c,i,t}^{lt}/(1+d)^{t}$; $cs_{c,i}^{mt} = CS_{c,i,\mathcal{t}}^{mt}/(1+d/2)^{\mathcal{t}}$.
$r_{c,i,t}$; $r_{c,i,\mathcal{t}}$	Recovery of ore material class, $c \in \mathcal{C}$ at location, i in period, $t \in T^{\ell t} and\ \mathcal{t} \in T^{mt}$; respectively.
rd	Penalty discount rate per annum.
$pc_{i}^{lt,RM}; pc_{i}^{mt,RM}$	Discounted mining rate penalty costs $PC_{i,t}^{lt,RM}, PC_{i,\mathcal{t}}^{mt,RM}$for deviating from the defined LT and MT mining rates at location, i in period, $t \in T^{\ell t}$ and $\mathcal{t} \in T^{mt}$; respectively. Where, $pc_{i}^{lt,RM} = PC_{i,t}^{lt,RM}/(1+rd)^{t}$; $pc_{i}^{mt,RM} = PC_{i,\mathcal{t}}^{mt,RM}/(1+rd/2)^{\mathcal{t}}$.
$pc_{i}^{lt,RO}; pc_{i}^{mt,RO}$	Discounted processing penalty rate costs $PC_{i,t}^{lt,RO}, PC_{i,\mathcal{t}}^{mt,RO}$for deviating from the defined LT and MT processing rates at location, i in period, $t \in T^{\ell t}$ and $\mathcal{t} \in T^{mt}$; respectively. Where, $pc_{i}^{lt,RO} = PC_{i,t}^{lt,RO}/(1+rd)^{t}$; $pc_{i}^{mt,RO} = PC_{i,\mathcal{t}}^{mt,RO}/(1+rd/2)^{\mathcal{t}}$.
$pc_{i}^{lt,RG}; pc_{i}^{mt,RG}$	Discounted grade target penalty costs $PC_{i,t}^{lt,RG}, PC_{i,\mathcal{t}}^{mt,RG}$for deviating from the defined LT and MT average grade target at location, i in period, $t \in T^{\ell t}$ and $\mathcal{t} \in T^{mt}$; respectively. Where, $pc_{i}^{lt,RG} = PC_{i,t}^{lt,RG}/(1+rd)^{t}$; $pc_{i}^{mt,RG} = PC_{i,\mathcal{t}}^{mt,RG}/(1+rd/2)^{\mathcal{t}}$.
$pc_{i,lt}^{mt,RM}$	Discounted integrated MT mining rate penalty costs $PC_{i,lt}^{mt,RM}$for deviating from LT mining rate scheduled, at location, i in period, $\mathcal{t} \in T^{mt}$. Where, $mt = lt/2$; $pc_{i,lt}^{mt,RM} = PC_{i,lt,\mathcal{t}}^{mt,RM}/(1+rd/2)^{\mathcal{t}}$.
$pc_{i,lt}^{mt,RO}$	Discounted integrated MT processing rate penalty costs $PC_{i,lt}^{mt,RO}$for deviating from LT processing rate scheduled, at location, i in period, $\mathcal{t} \in T^{mt}$. Where, $mt = lt/2$; $pc_{i,lt}^{mt,RO} = PC_{i,lt,\mathcal{t}}^{mt,RO}/(1+rd/2)^{\mathcal{t}}$.
$pc_{i,lt}^{mt,RG}$	Discounted integrated grade target penalty cost of MT grade target penalty costs $PC_{i,lt}^{mt,RO}$for deviating from LT grade target scheduled, at location, i in period, $\mathcal{t} \in T^{mt}$. Where, $mt = lt/2$; $pc_{i,lt}^{mt,RG} = PC_{i,lt,\mathcal{t}}^{mt,RG}/(1+rd/2)^{\mathcal{t}}$.

Parameter	Description
$L_i^{lt,RM}; L_i^{mt,RM}$	Lower bound for $lt\ and\ mt$ mining rate at location, $i \in \mathcal{M}$ in period, $t \in T^{\ell t}\ and\ \mathfrak{t} \in T^{mt}$; respectively.
$U_i^{lt,RM}; U_i^{mt,RM}$	Upper bound for $lt\ and\ mt$ mining rate at location, $i \in \mathcal{M}$ in period, $t \in T^{\ell t}\ and\ \mathfrak{t} \in T^{mt}$; respectively.
$L_i^{lt,RO}; L_i^{mt,RO}$	Lower bound for $lt\ and\ mt$ processing rate at location, i in period, $t \in T^{\ell t}\ and\ \mathfrak{t} \in T^{mt}$; respectively.
$U_i^{lt,RO}; U_i^{mt,RO}$	Upper bound for $lt\ and\ mt$ processing rate at location, i in period, $t \in T^{\ell t}\ and\ \mathfrak{t} \in T^{mt}$; respectively.
$L_i^{lt,RG}; L_i^{mt,RG}$	Lower bound for $lt\ and\ mt$ average ore grade for processing at location i in period, $t \in T^{\ell t}\ and\ \mathfrak{t} \in T^{mt}$; respectively.
$U_i^{lt,RG}; U_i^{mt,RG}$	Upper bound for $lt\ and\ mt$ average ore grade for processing at location i in period, $t \in T^{\ell t}\ and\ \mathfrak{t} \in T^{mt}$; respectively.
$\emptyset_{i,t}^{lt,RM}(x); \emptyset_{i,\mathfrak{t}}^{mt,RM}(x)$	Penalty costs function for deviating from mining rates at location i during each LT and MT scheduling period, $t \in T^{lt} and\ \mathfrak{t} \in T^{mt}$; respectively.
$\emptyset_{i,t}^{lt,RO}(x); \emptyset_{i,\mathfrak{t}}^{mt,RO}(x)$	Penalty costs function for deviating from ore grade targets at location, i during each LT and MT scheduling period, $t \in T^{lt} and\ \mathfrak{t} \in T^{mt}$; respectively.
$\emptyset_{i,t}^{lt,RG}(x); \emptyset_{i,\mathfrak{t}}^{mt,RG}(x)$	Penalty costs function for deviating from ore grade targets at location i during each LT and MT scheduling period, $t \in T^{lt} and\ \mathfrak{t} \in T^{mt}$; respectively.
$\emptyset_{i,\mathfrak{t},lt}^{mt,RM}(x)$	Integrated penalty costs function of MT mining rate for deviating from actual LT mining rate at location i during period, $\mathfrak{t} \in T^{mt}$.
$\emptyset_{i,\mathfrak{t},lt}^{mt,RO}(x)$	Integrated penalty costs function of MT processing rate for deviating from actual LT processing rate at location i during period, $\mathfrak{t} \in T^{mt}$.
$\emptyset_{i,\mathfrak{t},lt}^{mt,RG}(x)$	Integrated penalty costs function of MT grade target for deviating from actual LT grade target at location i during period, $\mathfrak{t} \in T^{mt}$.

3.2.2 Mixed integer programming of long-term and medium-term mine production scheduling

The outcome of integrating LT and MT mine production scheduling consists of two extraction sequences. The first extraction sequence defines the LT schedule on a yearly basis over the LOM horizon. The second extraction sequence defines the MT schedule on a half-yearly basis over a horizon of one to five years. Each LT and MT scheduling period determines the total mass of ore and waste material mined. The optimal LT extraction sequence is developed in such a way that maximum NPV is achieved over the LOM, while the optimal MT extraction sequence is developed in such a way that the maximum NPV of the LT schedule is maintained or improved.

The LT production scheduling outcome is used as an input to MT scheduling to ensure the alignment between LT and MT schedules. The block model and scheduling parameters of the half-yearly MT scheduling process are provided by the LT schedule to ensure scheduling consistency and alignment. The MT schedule breaks down each LT scheduling period, into two scheduling periods of half-year lengths each. The total mass of material of two consecutive half-year MT schedules must be equal to the total mass of material of the corresponding year of the LT schedule. Therefore, LT scheduling decision variables and parameters must inform the MT scheduling decision variables and parameters. Descriptions of decision variables that are used in this thesis to formulate the MIP model of the LT and MT mine production scheduling system are presented in Table 3.4.

Table 3.4: LT and MT mine scheduling decision variables of the mixed integer programming model

Variable	Description
$X_{i,n,t}^{lt};\ X_{i,n,t}^{mt}$	Binary variable for each block, $n \in \mathcal{N}$ mined from mining location i. It takes value of one if block, n is mined in period, $t \in T^{lt}\ or\ t \in T^{mt}$; zero otherwise.
$RM_{i,t}^{lt};\ RM_{i,t}^{mt}$	Reference continuous variable representing the actual mining rate achieved at location i during period, $t \in T^{lt} and\ t \in T^{mt}$.
$RO_{i,t}^{lt};\ RO_{i,t}^{mt}$	Reference continuous variable representing the actual amount of ore achieved at mining location i to be send to destination, e during period, $t \in T^{lt} and\ t \in T^{mt}$.
$RW_{i,t}^{lt};\ RW_{i,t}^{mt}$	Reference continuous variable representing the actual amount of waste achieved at mining location i to be send to destination, e during period, $t \in T^{lt} and\ t \in T^{mt}$.
$RG_{i,t}^{lt};\ RG_{i,t}^{mt}$	Reference continuous variable representing the actual weighted average ore grade achieved at mining location i during period, $t \in T^{lt} and\ t \in T^{mt}$.
$TNB_{i,t}^{lt};\ TNB_{i,t}^{mt}$	Reference integer variable representing the total number of blocks achieved at location i during each LT and MT scheduling period, $t \in T^{lt} and\ t \in T^{mt}$.
$TOB_{i,t}^{lt};\ TOB_{i,t}^{mt}$	Reference integer variable representing the total number of ore blocks achieved at location i during each LT and MT scheduling period, $t \in T^{lt} and\ t \in T^{mt}$.
$TWB_{i,t}^{lt};\ TWB_{i,t}^{mt}$	Reference integer variable representing the total number of waste blocks achieved at location i during each LT and MT scheduling period, $t \in T^{lt} and\ t \in T^{mt}$.

The objective function developed in this thesis for optimising the MIP integrated LT and MT mine production scheduling aims to maximise NPV or the total discounted profit of

blocks in the block model. The total NPV generated is the sum of the discounted economic value of each block in the model per scheduling period. The economic value of each block represents the difference between the expected revenue from the block and the costs associated with mining and processing the block. Revenue is a function of ore tonnage, ore price, and recovery. The costs of mining are influenced by the location of the block, in terms of both its depth from the surface and its distance to the final destination. The costs of processing are dependent on the grade and geological characteristics of the ore processed. Table 3.5 presents the MIP mathematical models (Equations 3.1 to 3.8) for independently optimising LT and MT mine production schedules.

Table 3.5: MIP models for independently optimising LT and MT mine production scheduling

	MIP model for LT mine production scheduling	**MIP model for MT mine production scheduling**	**Equation**
Discounted profit for a block:	$v_{n,t}^{lt} = \underbrace{[\sum_{n=1}^{N}\sum_{t=1}^{T^{lt}} OT_{i,n}^{lt} \times g_{n,c} \times r_{c,i,t} \times (p_{i,c,t}^{lt} - cs_{i,c,t}^{lt})]}_{Part\ I} - \underbrace{[\sum_{n=1}^{N}\sum_{t=1}^{T^{lt}} (OT_{i,n}^{lt} \times cp_{i,c,t}^{lt}) + (OT_{i,n}^{lt} + WT_{i,n}^{lt}) \times cm_{i,c,t}^{lt}]}_{Part\ II}.$	$v_{n,t}^{mt} = \underbrace{[\sum_{n=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}} OT_{i,n}^{mt} \times g_{n,c} \times r_{c,i,t} \times (p_{c,i,t}^{mt} - cs_{c,i,t}^{mt})]}_{Part\ I} - \underbrace{[\sum_{n=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}} (OT_{i,n}^{mt} \times cp_{c,i,t}^{mt}) + (OT_{i,n}^{mt} + WT_{i,n}^{mt}) \times cm_{c,i,t}^{mt}]}_{Part\ II}.$	[3.1]
Objective function:	$Max\ Z^{lt} = [\underbrace{\sum_{n=1}^{N}\sum_{t=1}^{T^{lt}} (v_{n,t}^{lt} \times X_{n,t}^{lt})}_{Part\ I} - \underbrace{\sum_{t=1}^{T^{lt}} \left(pc_i^{lt,RM} \times \emptyset_{i,t}^{lt,RM}(x)\right)}_{Part\ II} - \underbrace{\sum_{t=1}^{T^{lt}} pc_i^{lt,RO} \times \emptyset_{i,t}^{lt,RO}(x)}_{Part\ III} - \underbrace{\sum_{t=1}^{T^{lt}} \left(pc_i^{lt,RG} \times \emptyset_{i,t}^{lt,RG}(x)\right)}_{Part\ IV}].$	$Max\ Z^{mt} = [\underbrace{\sum_{n=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}} (v_{n,t}^{mt} \times X_{n,t}^{mt})}_{Part\ I} - \underbrace{\sum_{t=1}^{T^{mt}} \left(pc_i^{mt,RM} \times \emptyset_{i,t}^{mt,RM}(x)\right)}_{Part\ II} - \underbrace{\sum_{t=1}^{T^{mt}} pc_i^{mt,RO} \times \emptyset_{i,t}^{mt,RO}(x)}_{Part\ III} - \underbrace{\sum_{t=1}^{T^{mt}} \left(pc_i^{mt,RG} \times \emptyset_{i,t}^{mt,RG}(x)\right)}_{Part\ IV}].$	[3.2]
Mining rate constraint:	$L_{i,t}^{lt,RM} \leq [\sum_{n=1}^{N}\sum_{t=1}^{T^{lt}} (OT_{i,n}^{lt} + WT_{i,n}^{lt}) \times X_{n,t}^{lt}] \leq U_{i,t}^{lt,RM}.$	$L_{i,t}^{mt,RM} \leq [\sum_{n=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}} (OT_{i,n}^{mt} + WT_{i,n}^{mt}) \times X_{n,t}^{mt}] \leq U_{i,t}^{mt,RM}.$	[3.3]
Processing rate constraint:	$L_{i,t}^{lt,RO} \leq [\sum_{i=1}^{N}\sum_{t=1}^{T^{lt}} (OT_{i,n}^{lt}) \times X_{n,t}^{lt}] \leq U_{i,t}^{lt,RO}.$	$L_{i,t}^{mt,RO} \leq [\sum_{i=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}} (OT_{i,n}^{mt}) \times X_{n,t}^{mt}] \leq U_{i,t}^{mt,RO}.$	[3.4]
Target ore grade constraint:	$L_{i,t}^{lt,AG} \leq [\sum_{n=1}^{N}\sum_{t=1}^{T^{lt}} \frac{(OT_{i,n}^{lt} \times g_{n,c}) \times X_{n,t}^{lt}}{TOB_{i,t}^{lt} \times X_{n,t}^{lt}}] \leq U_{i,t}^{lt,AG}.$	$L_{i,t}^{mt,AG} \leq [\sum_{i=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}} \frac{(OT_{i,n}^{mt} \times g_{n,c}) \times X_{n,t}^{mt}}{OT_{i,n}^{mt} \times X_{n,t}^{mt}}] \leq U_{i,t}^{mt,AG}.$	[3.5]

	MIP model for LT mine production scheduling	MIP model for MT mine production scheduling	Equation
Block precedence constraint, 1:5 pattern:	$5X_{ijk}^{lt} - \sum_{bp=1}^{\Gamma^{lt}} (X_{i-1,j,k+1}^{bp} + X_{i,j-1,k+1}^{bp} + X_{i,j,k+1}^{bp} + X_{i,j+1,k+1}^{bp} + X_{i+1,j,k+1}^{bp}) \leq 0;\ bp \in \Gamma^{lt};\ \forall i,j,k.$	$5X_{ijk}^{mt} - \sum_{bp=1}^{\Gamma^{mt}} (X_{i-1,j,k+1}^{bp} + X_{i,j-1,k+1}^{bp} + X_{i,j,k+1}^{bp} + X_{i,j+1,k+1}^{bp} + X_{i+1,j,k+1}^{bp}) \leq 0;\ bp \in \Gamma^{mt};\ \forall i,j,k.$	[3.6]
Block precedence constraint, 1:9 pattern:	$9X_{ijk}^{lt} - \sum_{bp=1}^{\Gamma^{lt}} (X_{i-1,j-1,k+1}^{bp} + X_{i-1,j,k+1}^{bp} + X_{i-1,j+1,k+1}^{bp} + X_{i,j-1,k+1}^{bp} + X_{i,j,k+1}^{bp} + X_{i,j+1,k+1}^{bp} + X_{i+1,j-1,k+1}^{bp} + X_{i+1,j,k+1}^{bp} + X_{i+1,j+1,k+1}^{bp}) \leq 0;\ bp \in \Gamma^{lt};\ \forall i,j,k.$	$9X_{ijk}^{mt} - \sum_{bp=1}^{\Gamma^{mt}} (X_{i-1,j-1,k+1}^{bp} + X_{i-1,j,k+1}^{bp} + X_{i-1,j+1,k+1}^{bp} + X_{i,j-1,k+1}^{bp} + X_{i,j,k+1}^{bp} + X_{i,j+1,k+1}^{bp} + X_{i+1,j-1,k+1}^{bp} + X_{i+1,j,k+1}^{bp} + X_{i+1,j+1,k+1}^{bp}) \leq 0;\ bp \in \Gamma^{mt};\ \forall i,j,k.$	[3.7]
Mineral Reserve constraint:	$\sum_{t=1}^{T^{lt}} X_{n,t}^{lt} \leq 1.$	$\sum_{t=1}^{T^{mt}} X_{n,t}^{mt} \leq 1.$	[3.8]

Two patterns have been defined to express the block precedence constraints regardless of the scheduling horizon (Muke *et al.*, 2025a). Figures 3.1 (a) and 3.2 (b) show a 1:5 and a 1:9 pattern, mathematically expressed by Equations 3.6 and 3.7; respectively.

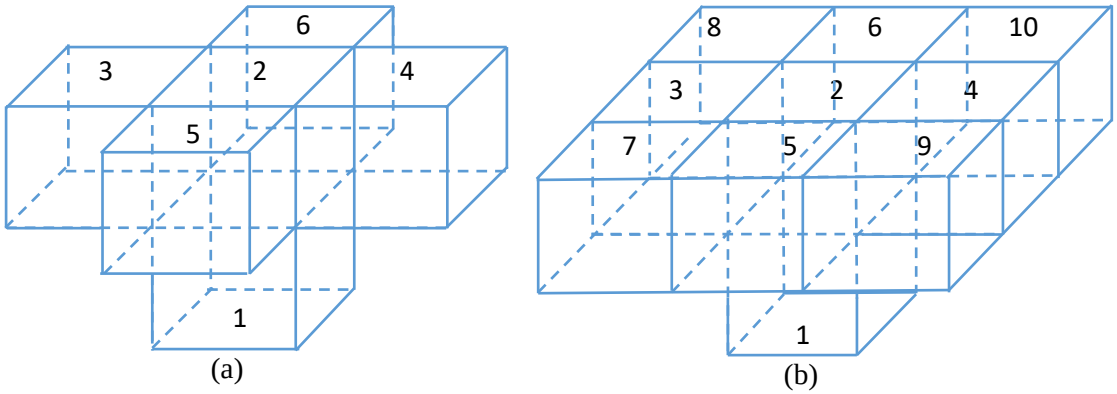


Figure 3.1: Graphical views of block predecessors, 1:5 pattern (a) and 1:9 pattern (b) *(Muke et al., 2025a)*

A 1:5 pattern requires the removal of five blocks directly situated above a block (denoted by 1 in Figure 3.1 (a)) and a 1:9 pattern requires the removal of nine blocks directly situated above a block (denoted by 1 in Figure 3.1 (b)) (Muke *et al.*, 2025a). The discounted profit of each block is expressed by Equation 3.1 consisting of two parts. Part I represents the total estimated discounted revenue that will be generated by mining blocks in the block model on a yearly basis. Part II represents the total costs associated with the mining and processing of blocks. A block with a positive discounted profit is considered an ore block and can be economically mined; while a block with a negative discounted profit is considered a waste block and can only be stripped out to give access to ore blocks.

The objective function as expressed by Equation 3.2 consists of four parts. Part I represents the estimated discounted profit obtained by mining blocks and processing ore blocks on a LT or MT schedule, respectively. Part II represents the LT or MT mining penalty costs associated with deviating from the actual minimum and maximum mining rates. Part III represents the LT or MT processing penalty costs associated with deviating from the actual minimum and maximum processing rates. Part IV represents the LT or MT ore grade penalty costs associated with deviating from the actual minimum and maximum ore grade for processing.

The value of penalty functions $\emptyset_{i,t}^{lt,RM}(x)$ and $\emptyset_{i,t}^{mt,RM}(x)$ for deviations associated with LT and MT mining rates, respectively, are calculated using Equations 3.9 to 3.13:

Let:

$$f_t^{lt,RM}(x) = RM_{i,t}^{lt} \text{ and } f_t^{mt,RM}(x) = RM_{i,t}^{mt}. \quad [3.9]$$

Then:

$$L_{i,t}^{lt,RM} \leq f_t^{lt,RM}(x) \leq U_{i,t}^{lt,RM} \text{ and } L_{i,t}^{mt,RM} \leq f_t^{mt,RM}(x) \leq U_{i,t}^{mt,RM}. \quad [3.10]$$

Lower boundary:

$$\begin{aligned} &f_t^{lt,RM}(x) \geq L_{i,t}^{lt,RM} \text{ and } f_t^{mt,RM}(x) \geq L_{i,t}^{mt,RM}, \\ &-f_t^{lt,RM}(x) \leq -L_{i,t}^{lt,RM} \text{ and } -f_t^{mt,RM}(x) \leq -L_{i,t}^{mt,RM}, \\ &-f_t^{lt,RM}(x) + L_{i,t}^{lt,RM} \leq 0 \text{ and } -f_t^{mt,RM}(x) + L_{i,t}^{mt,RM} \leq 0. \end{aligned} \quad [3.11]$$

Upper boundary:

$$\begin{aligned} &f_t^{lt,RM}(x) \leq U_{i,t}^{lt,RM} \text{ and } f_t^{mt,RM}(x) \leq U_{i,t}^{mt,RM}, \\ &f_t^{lt,RM}(x) - U_{i,t}^{lt,RM} \leq 0 \text{ and } f_t^{mt,RM}(x) - U_{i,t}^{mt,RM} \leq 0. \end{aligned} \quad [3.12]$$

Therefore:

$$\begin{aligned} \emptyset_{i,t}^{lt,RM}(x) &= max\left\{0, -f_t^{lt,RM}(x) + L_{i,t}^{lt,RM}, f_t^{lt,RM}(x) - U_{i,t}^{lt,RM}\right\}, \\ \emptyset_{i,t}^{mt,RM}(x) &= max\left\{0, -f_t^{mt,RM}(x) + L_{i,t}^{mt,RM}, f_t^{mt,RM}(x) - U_{i,t}^{mt,RM}\right\}. \end{aligned} \quad [3.13]$$

Similarly to Equation 3.13, values of penalty functions $\emptyset_{i,t}^{lt,RO}(x)$, $\emptyset_{i,t}^{mt,RO}(x)$, $\emptyset_{i,t}^{lt,RG}(x)$ and $\emptyset_{i,t}^{mt,RO}(x)$ for deviations associated with LT and MT processing rates and ore grade targets, respectively, are calculated using Equations 3.14 and 3.15:

$$\begin{aligned} \emptyset_{i,t}^{lt,RO}(x) &= max\left\{0, -f_t^{lt,RO}(x) + L_{i,t}^{lt,RO}, f_t^{lt,RO}(x) - U_{i,t}^{lt,RO}\right\}, \\ \emptyset_{i,t}^{mt,RO}(x) &= max\left\{0, -f_t^{mt,RO}(x) + L_{i,t}^{mt,RO}, f_t^{mt,RO}(x) - U_{i,t}^{mt,RO}\right\}, \end{aligned} \quad [3.14]$$

and

$$\begin{aligned} \emptyset_{i,t}^{lt,RG}(x) &= max\left\{0, -f_t^{lt,RG}(x) + L_{i,t}^{lt,RG}, f_t^{lt,RG}(x) - U_{i,t}^{lt,RG}\right\}, \\ \emptyset_{i,t}^{mt,RO}(x) &= max\left\{0, -f_t^{mt,RG}(x) + L_{i,t}^{mt,RG}, f_t^{mt,RG}(x) - U_{i,t}^{mt,RG}\right\}. \end{aligned} \quad [3.15]$$

Although penalty costs are imposed on the objective function, Equations 3.3 to 3.8 apply constraints to ensure that production rate, processing rate, grade target and Mineral Reserve constraints are met in every period for LT and MT schedules.

3.3 Integrating long-term and medium-term mine production scheduling

After developing MIP models for isolated LT and MT mine production scheduling, it is necessary to connect these models to create an integrated LT and MT production scheduling system. Firstly, a framework was created to illustrate the dynamic steps of the integrating process, and secondly, to develop the MIP model that integrates LT and MT production schedules.

3.3.1 A framework for a dynamic integrated long-term and medium-term mine production scheduling

The development of a scheduling framework is important to enable the efficient formulation of a dynamic integrated LT and MT production scheduling model. Two aspects must be considered in developing the scheduling framework. The first aspect consists of integrating the mine production scheduling process by linking LT and MT scheduling horizons.

The second aspect consists of the dynamic characteristic of the mine production scheduling process that requires continuous scheduling changes in time and space to optimise the entire process. For instance, updating a block model with new information from the MT schedule can potentially affect the LT schedule and the extraction sequence. A dynamic integrated mine production scheduling system simultaneously aligns scheduling horizons and continuously updates the whole system to consider the changes. Figure 3.2 presents the conceptual framework for the dynamic integrated LT and MT mine production scheduling model.

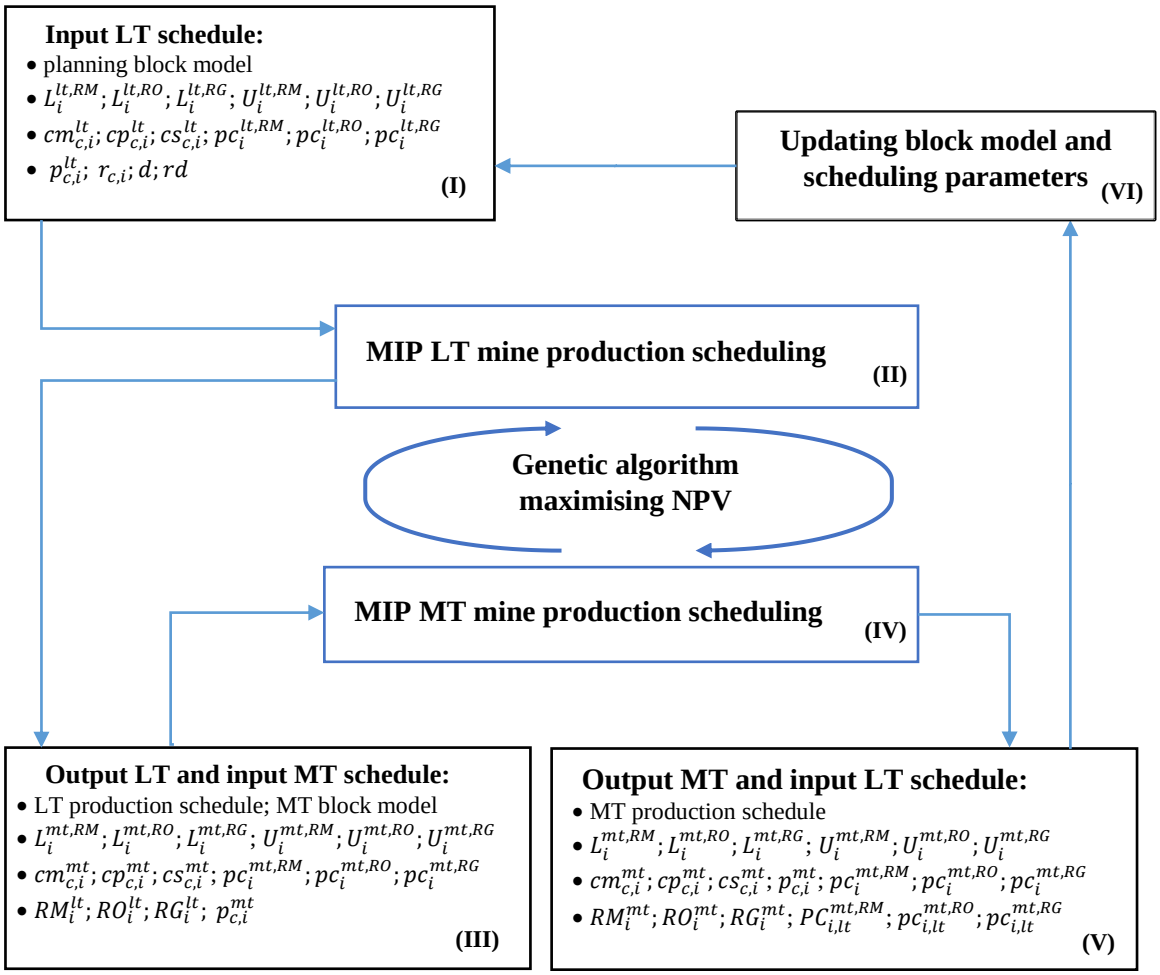


Figure 3.2: A framework for a dynamic integrated long-term and medium-term mine production scheduling model
(Muke *et al*., 2025a)

The framework comprises six steps. Step I involves entering input scheduling parameters into the LT scheduling horizon, which informs MT scheduling parameters in Step III. Step II employs GA to optimise the LT mine production schedule, which is then connected to the MT production schedule in Step IV. Step III consists of collecting LT scheduling results and adjusting MT input parameters to feed the MT scheduling horizon. Step IV employs the GA to optimise the MT production schedule, which is connected back to the LT schedule horizon. Step V consists of collecting MT scheduling results and LT scheduling parameters. Step VI consists of updating the block model and scheduling parameters to provide feedback to the LT schedule horizon. Therefore, this framework provides a synchronized development of the LT and MT mine production scheduling model. It enables decision makers to evaluate different scenarios while ensuring seamless

communication between LT and MT scheduling horizons. Additionally, the framework improves the efficiency of production scheduling by minimising information gaps encountered when using separately optimised production scheduling models.

3.3.2 Objective function integrating long-term and medium-term production scheduling of the mixed integer programming model

As stated earlier, the objective function aims to maximise NPV and align LT and MT scheduling horizons while adhering to various constraints inherent in the mining process. To account for uncertainty in the objective function, scheduling penalties are introduced as a means of penalising deviations from scheduled production targets set for mining rate, processing rate and ore grade. These penalties provide a mechanism for evaluating and addressing potential disruptions or discrepancies that might arise during mining operations. By maximising NPV while considering constraints and assigning appropriate scheduling penalties within the integrated framework, mine production scheduling can effectively ensure sustainable utilisation of Mineral Reserves. The objective function for the MIP integrated LT and MT mine production scheduling model is presented by Equation 3.16.

$$
\begin{aligned}
&Max\ Z_{mt}^{lt} \\
&= [\underbrace{\sum_{n=1}^{N}\sum_{t=1}^{T^{lt}}\left(\sum_{n=1}^{TNB^{lt}}\sum_{t=1}^{T^{mt}}(v_{n,t}^{mt}\times X_{n,t}^{mt})\right)\times X_{n,t}^{lt}}_{Part\ I} \\
&\underbrace{-\sum_{t=1}^{T^{lt}}\left(pc_{i}^{lt,RM}\times \emptyset_{i,t}^{lt,RM}(x)+pc_{i}^{lt,RO}\times \emptyset_{i,t}^{lt,RO}(x)+\ pc_{i}^{lt,RG}\times \emptyset_{i,t}^{lt,RG}(x)\right)}_{Part\ II} \\
&\underbrace{-\sum_{t=1}^{T^{mt}}\left(pc_{i}^{mt,RM}\times \emptyset_{i,t}^{mt,RM}(x)+pc_{i}^{mt,RO}\times \emptyset_{i,t}^{mt,RO}(x)+\ pc_{i}^{mt,RG}\times \emptyset_{i,t}^{mt,RG}(x)\right)}_{Part\ III} \\
&\underbrace{-\sum_{t=1}^{T^{mt}}\left(pc_{i,lt}^{mt,RM}\times \emptyset_{i,t,lt}^{mt,RM}(x)+pc_{i,lt}^{mt,RO}\times \emptyset_{i,t,lt}^{mt,RO}(x)+\ pc_{i,lt}^{mt,RG}\times \emptyset_{i,t,lt}^{mt,RG}(x)\right)}_{Part\ IV}].
\end{aligned}
\qquad [3.16]
$$

The objective function (Equation 3.16) consists of four parts. Part I represents the estimated discounted economic profit for the integrated LT and MT mine scheduling model. To enable the integration of the scheduling process, the model sums up the half-yearly MT economic values into yearly LT economic values. Part II represents the total yearly LT

mining rate, processing rates and ore grade target penalty costs associated with deviations from the actual annual minimum and maximum mining rate, processing rate and ore grade target, respectively. Likewise, Part III represents the total half-yearly MT mining and processing rates, and ore grade target penalty costs associated with deviations from the actual six-month minimum and maximum mining rate, processing rate and ore grade target, respectively. Part IV represents the integrated penalty costs associated with half-yearly MT mining rate and processing rate and ore grade target deviations from the actual annual LT mining rate scheduled.

Referring to Equations 3.13 to 3.15, the integrated LT and MT penalty functions that calculate deviations associated with mining rate, processing rate and ore grade targets are expressed by Equations 3.17 to 3.19, respectively:

$$\emptyset_{i,t,lt}^{mt,RM}(x) = max\left\{0, -f_{t,lt}^{mt,RM}(x) + L_{i,t}^{lt,RM}, f_{t,lt}^{mt,RM}(x) - U_{i,t}^{lt,RM}\right\}, \quad [3.17]$$

$$\emptyset_{i,t,lt}^{mt,RO}(x) = max\left\{0, -f_{t,lt}^{mt,RO}(x) + L_{i,t}^{lt,RO}, f_{t,lt}^{mt,RO}(x) - U_{i,t}^{lt,RO}\right\}, \quad [3.18]$$

$$\emptyset_{i,t,lt}^{mt,RG}(x) = max\left\{0, -f_{t,lt}^{mt,RG}(x) + L_{i,t}^{lt,RG}, f_{t,lt}^{mt,RG}(x) - U_{i,t}^{lt,RG}\right\}. \quad [3.19]$$

Equation 3.20 ensures the alignment of the economic parameter between LT and MT production scheduling horizons:

$$v_{n,t}^{lt} = \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} (v_{n,t}^{mt} \times X_{n,t}^{mt}). \quad [3.20]$$

Equation 3.21 ensures the temporal alignment between LT and MT production scheduling horizons:

$$T^{lt}(t) \subseteq T^{mt}(t). \quad [3.21]$$

Equation 3.22 ensures the spatial alignment between LT and MT production schedule horizons:

$$\begin{cases} N = \sum_{t=1}^{T^{lt}} TNB_{i,t}^{lt}, \\ TNB_{i,t}^{lt} = \sum_{t=1}^{T^{mt}} TNB_{i,t}^{mt}. \end{cases} \quad [3.22]$$

3.3.3 Dynamic integrated long-term and medium-term scheduling constraints

The integrated objective function in Equation 3.16 is subject to several constraints including mining rate, processing rate, ore grade target, block precedence relationship, and Mineral Reserve constraints. These constraints are represented by Equations 3.23 to 3.42.

Integrated LT and MT scheduling mining rate constraint

Equations 3.23 and 3.24 represent the integrated LT and MT mining rate constraints. These constraints are limitations imposed on the amount in tonnes of ore and waste blocks mined during each scheduling period.

$$RM_{i,t,lt}^{mt} = f_{t,lt}^{mt,RM}(x) = \sum_{t=1}^{T^{lt}} \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} \left(OT_{i,n}^{mt} + WT_{i,n}^{mt}\right) \times X_{n,t}^{mt}, \quad [3.23]$$

where:

$$RM_{i,t}^{mt} = f_{t}^{mt,RM}(x) = \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} \left(OT_{i,n}^{mt} + WT_{i,n}^{mt}\right) \times X_{n,t}^{mt}. \quad [3.24]$$

In this model, the scheduling period follows a half-yearly MT scheduling and a yearly LT scheduling demarcation sequence. The mining rate constraint is defined using two key limiting parameters namely the minimum and maximum mining rates. The minimum mining rate indicates the lowest achievable extraction rate that should be maintained, ensuring continuous mining and efficient Mineral Reserve utilisation. The maximum mining rate represents the upper limit beyond which excessive or unsustainable extraction may occur. To ensure alignment between LT and MT schedules, the cumulative half-yearly mining rate must be equal to the corresponding yearly mining rate.

When the minimum and maximum mining rates are not satisfied, the shortage or excess penalties must be applied to adjust the MT and LT mining rate constraint, as expressed by Equations 3.25 and 3.26.

$$\emptyset_{i,t,lt}^{mt,RM}(x) = max\left\{0, -f_{t,lt}^{mt,RM}(x) + L_{i,t}^{lt,RM}, f_{t,lt}^{mt,RM}(x) - U_{i,t}^{lt,RM}\right\}, \quad [3.25]$$

where:

$$\begin{cases} -f_{t,lt}^{mt,RM}(x) + L_{i,t}^{lt,RM} \leq 0, \\ f_{t,lt}^{mt,RM}(x) - U_{i,t}^{lt,RM} \leq 0. \end{cases} \quad [3.26]$$

Alignment of the mining rate constraint in the dynamic integrated LT and MT mine scheduling model consists of linkages expressed by Equations 3.27 and 3.28. Equation 3.27 represents the linking of minimum and maximum mining rates, while Equation 3.28 represents the linking of actual mining rates achieved between LT and MT schedules.

$$L_i^{mt,RM} = \frac{L_i^{lt,RM}}{T^{mt}};\ U_i^{mt,RM} = \frac{U_i^{lt,RM}}{T^{mt}}; \quad [3.27]$$

and

$$RM_{i,t}^{lt} = \sum_{t=1}^{T^{mt}} RM_{i,t}^{mt}. \quad [3.28]$$

Integrated LT and MT mine scheduling processing rate constraint

Equations 3.29 and 3.30 represent the integrated LT and MT processing rate constraints. These constraints are limitations imposed on the ore tonnage during each scheduling period. Ore tonnage realisation in each scheduling period influences scheduling decisions since it directly impacts revenue generation and Mineral Reserves utilisation strategies.

$$RO_{i,t,lt}^{mt} = f_{t,lt}^{mt,RO}(x) = \sum_{t=1}^{T^{lt}} \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} OT_{i,n}^{mt} \times X_{n,t}^{mt}, \quad [3.29]$$

where:

$$RO_{i,t}^{mt} = f_{t}^{mt,RO}(x) = \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} OT_{i,n}^{mt} \times X_{n,t}^{mt}. \quad [3.30]$$

In this model, the scheduling period follows a half-yearly MT scheduling and a yearly LT scheduling demarcation sequence. The processing rate constraint is defined using two key limiting parameters which are minimum and maximum ore processing rates. The minimum processing rate indicates the lowest achievable processing rate that should be maintained, ensuring continuous ore delivery to the plant. The maximum processing rate represents the upper limit beyond which excessive or unsustainable processing may occur. A careful consideration of these rates ensures that processing activities are carried out within practical limits without compromising the quality or quantity of ore processed. To ensure alignment between LT and MT production schedules, the cumulative half-yearly processing ore rate of the MT schedule must be equal to the corresponding ore processing rate of the LT schedule.

When the minimum and maximum processing rates are not satisfied, the shortage or excess penalties must be applied to adjust the MT and LT processing rate constraints, as expressed by Equations 3.31 and 3.32.

$$\emptyset_{i,t,lt}^{mt,RO}(x) = max\left\{0, -f_{t,lt}^{mt,RO}(x) + L_{i,t}^{lt,RO}, f_{t,lt}^{mt,RO}(x) - U_{i,t}^{lt,RO}\right\}, \quad [3.31]$$

where:

$$\begin{cases} -f_{t,lt}^{mt,RO}(x) + L_{i,t}^{lt,RO} \leq 0, \\ f_{t,lt}^{mt,RO}(x) - U_{i,t}^{lt,RO} \leq 0. \end{cases} \quad [3.32]$$

Alignment of the processing rate constraint in the dynamic integrated LT and MT mine scheduling model consists of linkages expressed by Equations 3.33 and 3.34. Equation 3.33 represents the linking of minimum and maximum processing rates, while Equation 3.34 represents the linking of actual processing rates achieved, between LT and MT schedules.

$$L_i^{mt,RO} = \frac{L_i^{lt,RO}}{T^{mt}};\ U_i^{mt,RO} = \frac{U_i^{lt,RO}}{T^{mt}}; \quad [3.33]$$

and

$$RO_{i,t}^{lt} = \sum_{t=1}^{T^{mt}} RO_{i,t}^{mt}. \quad [3.34]$$

Integrated mine scheduling ore grade target constraint

Equations 3.35 and 3.36 represent the integrated LT and MT ore grade target constraint. The constraints are limitations imposed on the quality of ore material based on the geological variability during each scheduling period.

$$RG_{i,t,lt}^{mt} = f_{t,lt}^{mt,RG}(x) = \sum_{t=1}^{T^{lt}} \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} \frac{\left(OT_{i,n}^{mt} \times g_{n,c}\right) \times X_{i,t}^{mt}}{OT_{i,n}^{mt} \times X_{i,t}^{mt}}, \quad [3.35]$$

where:

$$RG_{i,t}^{mt} = f_{t}^{mt,RG}(x) = \sum_{n=1}^{TNB^{lt}} \sum_{t=1}^{T^{mt}} \frac{\left(OT_{i,n}^{mt} \times g_{n,c}\right) \times X_{n,t}^{mt}}{OT_{i,n}^{mt} \times X_{n,t}^{mt}}. \quad [3.36]$$

Different areas within a mine may have different grades, leading to variations in extraction rates and processing requirements. This requires careful scheduling to maximise

production while minimising costs. To maximise the objective function through efficient Mineral Reserves utilisation, the integrated mine scheduling model must ensure a balance between extracting high grade ore efficiently and maintaining sustainable plant feed over the LOM.

Geological data and economic conditions are analysed to determine the optimal extraction sequence within each period to meet plant requirements. This is achieved by setting lower ore grade and upper ore grade constraints. The lower ore grade constraint is a threshold below which mining and processing become uneconomic; thus, only blocks with a grade equal to or above the threshold meet processing requirements. On the other hand, the higher ore grade constraint restrains the extraction of blocks with excessively high ore grades. In mine production scheduling, aiming for higher grades ensures increased revenue. However, higher grades may surpass downstream processing capabilities or result in lower ore recovery. The application of ore grade target at each horizon of production scheduling enables a balanced production rate while maintaining consistent ore quality standards needed for a sustainable production scheduling process.

To ensure alignment between LT and MT schedules, the cumulative half-yearly weighted average ore grade of the MT schedule must be equal to the corresponding weighted average ore grade of the LT schedule. When the minimum and maximum ore grade targets are not satisfied, the shortage or excess penalties must be applied to adjust the MT and LT ore grade target constraints, as expressed by Equations 3.37 and 3.38.

$$\emptyset_{i,t,lt}^{mt,RG}(x) = max\left\{0, -f_{t,lt}^{mt,RG}(x) + L_{i,t}^{lt,RG}, f_{t,lt}^{mt,RG}(x) - U_{i,t}^{lt,RG}\right\}, \quad [3.37]$$

where:

$$\begin{cases} -f_{t,lt}^{mt,RG}(x) + L_{i,t}^{lt,RG} \leq 0, \\ f_{t,lt}^{mt,RG}(x) - U_{i,t}^{lt,RG} \leq 0. \end{cases} \quad [3.38]$$

Alignment of the ore grade target constraint in the dynamic integrated LT and MT mine production scheduling model consists of linkages expressed by Equations 3.39 and 3.40. Equation 3.39 represents the linking of minimum and maximum ore grade targets, while Equation 3.40 represents the linking of actual ore grades achieved, between LT and MT schedules.

$$L_i^{mt,RG} = \frac{L_i^{lt,RG}}{T^{mt}};\ U_i^{mt,RG} = \frac{U_i^{lt,RG}}{T^{mt}}; \quad [3.39]$$

and

$$RG_{i,t}^{lt} = \sum_{t=1}^{T^{mt}} RG_{i,t}^{mt}. \quad [3.40]$$

Integrated block precedence constraints

The block precedence constraint is important in optimising extraction sequences and generating practical schedules. The constraint governs the extraction sequence of mineral blocks within a given period, ensuring that it adheres to certain predefined conditions such as pit slope angle while maximising overall productivity. The process involves analysing various factors such as geotechnical considerations, extraction equipment capacities, ore quality distribution, and waste disposal requirements. By comprehensively assessing these variables, engineers can devise a feasible schedule that optimises production while also minimising operating costs and mitigating potential risks associated with safety and environmental compliance. By defining priorities within the production scheduling model, the constraint ensures that blocks are mined in a sequence that maximises economic returns while minimising waste generation or delaying access to future high-grade areas. The block precedence constraint must be embedded within both the LT and MT schedules, as expressed by Equations 3.6 and 3.7, to ensure efficient utilisation of Mineral Reserves and maximum productivity.

For the schedule to partially address practical short-term operational requirements, it is necessary to include an additional operational constraint to the MT schedule as defined by the number of benches bh along the k direction to be mined in each MT scheduling period (Muke *et al*., 2025a). By letting bt to denote the top bench of MT blocks from the surface, and bb to represent the bottom bench of MT blocks in period t, then the number of benches to be mined in each MT schedule is constrained by Equation (3.41).

$$\sum_{bh=bt=1}^{bb} \left(RM_{bh_k}^{mt}\right) \leq RM_{i,t}^{mt};\ bh_k \in [bt, bb], \forall k, t; \quad [3.41]$$

where $RM_{bh_k}^{mt}$ is the tonnage of material (ore and waste) on bench bh_k for the MT schedule in period t (Muke *et al.*, 2025a). Logically, the maximum number of benches extracted is achieved when the cumulative tonnage satisfies the MT mining rate constraint.

Integrated mine scheduling Mineral Reserve constraints

One key factor that must be carefully considered is the Mineral Reserve constraint in the integrated mine production scheduling model. The Mineral Reserve constraint refers to the estimated number of blocks present within the UPL that can be economically mined. This constraint is used as the foundation upon which all extraction schedules are built. The challenge lies in developing a production scheduling system that maximises block extraction while staying within the UPL. The scheduling model must ensure that in every scheduling horizon, a block is mined at most once. However, the model must allow for partial mining of blocks.

Alignment of the Mineral Reserve constraint in the dynamic integrated LT and MT mine scheduling model consists of linkages expressed by Equations 3.42 and 3.43. Equation 3.42 defines the link between the LT and MT total number of blocks, while Equation 3.43 defines the alignment of the total Mineral Reserve constraint and the LT to MT schedules.

$$TNB_t^{lt} = \sum_{t=1}^{T^{mt}} TNB_t^{mt}, \qquad [3.42]$$

where:

$$N = \sum_{t=1}^{T^{lt}} \sum_{t=1}^{T^{mt}} TNB_t^{mt}. \qquad [3.43]$$

3.4 Reconciliation in mine production scheduling

An effective open-pit mine production scheduling process depends, firstly on the degree of scheduling system integrity as discussed earlier. Secondly, it depends on the level of reconciliation of the scheduling process (Otto and Musingwini, 2019). According to Bester *et al*. (2016), reconciliation is a process that involves identifying, analysing, and reporting any discrepancies between scheduled and actual results to maintain an optimal mining sequence. Reconciliation of the production scheduling process consists not only of measuring what has been scheduled versus what has been actually mined but also of measuring compliance between scheduling horizons. The aim of reconciling the

scheduling process is to help achieve optimal NPV while maximising scheduling productivity, optimising Mineral Reserves utilisation and improving the practicability of the schedule (Otto, 2019). Therefore, reconciling an integrated scheduling system reduces the gap between material that has been scheduled and material that has been mined.

Reconciliation of an integrated mine scheduling process includes measuring spatial and temporal scheduling compliance. Spatial scheduling compliance identifies the location of mining areas or polygons within the production schedule and implements a sequence in which these mining areas should be mined. Temporal scheduling compliance enforces the scheduling time across all the scheduling horizons. Once the reconciliation process is complete, the final production schedule is implemented and monitored to ensure that it is being followed (Morley and Arvidson, 2017). The monitoring process allows adjustments to be made to the schedule, if necessary, to ensure that the objectives of the reconciliation process are met.

3.4.1 Spatial compliance of the mine production scheduling system

Spatial compliance of mine production scheduling combines the fact that scheduling must incorporate both the spatial location of mining areas and the associated production activities. Otto (2019) stated that mine spatial reconciliation is an important measurement to reconcile a scheduling system, from LT to MT and ST schedules. Spatial scheduling compliance indicates not only the locations of the mining sequence but also other mining activities such as drilling and blasting. Although spatial production scheduling compliance is crucial, mining activities tend to overlook the spatial aspects of production scheduling, due to a focus on ST benefits such as productivity improvements and cost reduction. A consequence of overlooking spatial compliance aspects is that ST benefits are achieved at the expense of spatial compliance to mine scheduling (Otto and Musingwini, 2020). As a result, mining activities may occur in unscheduled areas, leading to the sterilisation of Mineral Reserves and potentially hindering the LT economic and operational sustainability of an open-pit mine.

The spatial compliance of an integrated mine scheduling system involves comparing the tonnage mined from mining locations to the scheduled tonnage for mining in LT, MT and ST schedules. It provides a measuring system that can be compared with other intertemporal compliance metrics. Both spatial and temporal scheduling compliance

account for the alignment and discrepancies between actual and scheduled mining areas across the production scheduling horizons. Scheduling alignment and discrepancies can be classified into different spatial compliance categories as shown in Table 3.6.

Table 3.6: Description of spatial production compliance categories for an integrated mine production scheduling system

(Source: adapted from Otto and Musingwini, 2020)

Compliance categories	**Spatial mine compliance integrated categories**	**Description**
Material mined in schedule in sequence	Material mined in LT schedule in LT sequence.	Material from areas that were scheduled to be mined in LT and were mined in LT sequence.
	Material mined in MT schedule in LT-MT integrated sequence.	Material from areas that were scheduled to be mined in MT and were mined in LT-MT integrated sequence.
	Material mined in ST schedule in LT-MT-ST integrated sequence.	Material from areas that were scheduled to be mined in ST and were mined in LT-MT-ST integrated sequence.
Material mined in schedule out of sequence	Material mined in LT schedule out of LT sequence.	Material from areas that were scheduled to be mined in LT but were mined out of LT sequence.
	Material mined in MT schedule out of LT-MT integrated sequence.	Material from areas that were scheduled to be mined in MT but were mined out of LT-MT integrated sequence.
	Material mined in ST schedule out of LT-MT-ST integrated sequence.	Material from areas that were scheduled to be mined in ST but were mined out of LT-MT-ST sequence.
Material mined out of schedule	Material mined out of LT schedule.	Material from areas that were mined completely outside of the LT schedule.
	Material mined out of MT schedule.	Material from areas that were mined completely outside of the MT schedule.
	Material mined out of ST schedule.	Material from areas that were mined completely outside of the ST schedule.
Scheduled not mined	LT scheduled not mined.	Material from areas that were scheduled to be mined in LT but were not mined.
	MT scheduled not mined.	Material from areas that were scheduled to be mined in MT but were not mined.
	ST scheduled not mined.	Material from areas that were scheduled to be mined in ST but were not mined.

3.4.2 Temporal compliance of the mine production scheduling system

Temporal mine scheduling reconciliation is the process of comparing the scheduled with the actual performance (tonnes and grade) over time. This process is important in ensuring that production activities are completed on time and that resources are used efficiently (Morley and Arvidson, 2017). Three factors must be considered when creating a temporal

schedule reconciliation. Firstly, the sequence of activities must be determined. This will ensure that activities are completed in the correct order with appropriate resources. Secondly, the duration of each activity must be estimated to ensure that the schedule is achievable and that activities are not delayed. Finally, the resources required for each activity must be identified. This will ensure that the schedule can be completed on time and within economic targets. It is important to frequently monitor and adjust the temporal schedule reconciliation to ensure that the schedule remains on track and that production goals are met. The following sub-sections discuss a set of metrics used to calculate the schedule compliance benchmark for an integrated mine scheduling system.

Mine production compliance

The mine production compliance represents an intertemporal metric for measuring the total actual material mined, both in and out of the integrated scheduling system, as a percentage of the scheduled material for the scheduling periods under review. Equations 3.44 and 3.45 calculate the LT to MT, and MT to ST integrated mine production compliance, respectively (Otto and Musingwini, 2020):

$$LT\ to\ MT\ production\ compliance = \frac{Total\ actual\ MT\ material\ mined}{Total\ LT\ material\ scheduled}, \quad [3.44]$$

$$MT\ to\ ST\ production\ compliance = \frac{Total\ actual\ ST\ material\ mined}{Total\ MT\ material\ scheduled}. \quad [3.45]$$

Equations 3.44 and 3.45 provide information about how well the actual mining production aligns with the LT and the MT production schedules. If the ratio is greater than 1, it indicates that more material was mined than initially scheduled for in the LT or MT schedule, potentially suggesting accelerated production. If the ratio is less than 1, it implies that less material was mined than scheduled, which may indicate delays or slower progress in meeting LT or MT production goals.

Mine production scheduling adherence index

The mine production adherence index measures actual areas mined, both in sequence and out of sequence of the integrated scheduling system, as a percentage of total actual areas mined. Equations 3.46 and 3.47 calculate the LT to MT, and MT to ST scheduling adherence index, respectively (Otto and Musingwini, 2020):

$$LT\ to\ MT\ mine\ scheduling\ adherence\ index = \frac{MT\ area\ mined\ in\ LT\ sequence + MT\ area\ mined\ out\ of\ LT\ sequence}{Total\ actual\ MT\ area\ scheduled}, \quad [3.46]$$

$$MT\ to\ ST\ mine\ scheduling\ adherence\ index = \frac{ST\ area\ mined\ in\ MT\ sequence + ST\ area\ mined\ out\ of\ MT\ sequence}{Total\ actual\ ST\ area\ scheduled}. \quad [3.47]$$

Equations 3.46 and 3.47 provide information about how well the actual mining production area adheres to the LT or MT schedule. If the index is greater than 1, it indicates that more area was mined (in and out of sequence) than initially scheduled for in the LT or MT schedule, potentially suggesting accelerated production. If the index is less than 1, it implies that less area was mined (in and out of the integrated scheduling sequence) than scheduled, which may indicate deviations from the LT or MT mine schedules.

3.4.2.3 Spatial mine to schedule compliance index

The spatial mine to schedule compliance index measures actual areas mined in schedule, both in sequence and out of sequence of the integrated scheduling system, compared to the scheduled areas. Equations 3.48 and 3.49 calculate the LT to MT, and MT to ST spatial mine to schedule compliance index, respectively (Otto and Musingwini, 2020):

$$LT\ to\ MT\ spatial\ mine\ to\ schedule\ index = \frac{MT\ area\ mined\ in\ LT\ sequence + MT\ area\ mined\ out\ of\ LT\ sequence}{Total\ LT\ area\ scheduled}, \quad [3.48]$$

$$MT\ to\ ST\ Spatial\ mine\ to\ schedule\ index = \frac{ST\ area\ mined\ in\ MT\ sequence + ST\ area\ mined\ out\ of\ MT\ sequence}{Total\ MT\ area\ scheduled}. \quad [3.49]$$

Equations 3.48 and 3.49 provide a view of how well the actual mining production area aligns with the LT production plan, considering the sequence of the mining area schedule. If the index is greater than 1, it indicates that more area was mined (in and out of integrated schedule sequence) than initially scheduled for in the LT or MT, potentially suggesting accelerated production. If the index is less than 1, it implies that less area was mined (in and out of the integrated schedule sequence) than scheduled, which may indicate delays or slower progress in meeting the LT or MT production area goals.

In-sequence spatial mine to schedule index

The in-sequence spatial mine to schedule index measures actual areas mined in the sequence of the integrated scheduling system, compared to the scheduled areas. Equations 3.50 and 3.51 calculate the LT to MT, and MT to ST in-sequence spatial mine to schedule compliance index, respectively (Otto and Musingwini, 2020):

$$LT\ to\ MT\ in\ sequence\ spatial\ mine\ to\ schedule\ index = \frac{MT\ area\ mined\ in\ LT\ sequence}{Total\ LT\ area\ scheduled}, \quad [3.50]$$

$$MT\ to\ ST\ in\ sequence\ spatial\ mine\ to\ schedule\ index = \frac{ST\ area\ mined\ in\ MT\ sequence}{Total\ MT\ area\ scheduled}. \quad [3.51]$$

Equations 3.50 and 3.51 provide information about the efficiency or progress of mining activities compared to the schedule. If the index is greater than 1, it indicates that more material was mined than initially scheduled, suggesting an accelerated mining process. If the index is less than 1, it implies that less material was mined than scheduled, which may indicate delays or slower progress in the mining operations.

3.5 Summary of Chapter 3

An MIP model that integrates LT and MT mine production schedules was presented in this chapter. The model addresses the knowledge gap in the existing literature wherein LT and MT mine production schedules have been independently optimised. The integration of the scheduling horizons is still a novel approach to the optimisation of the mine production scheduling process. The chapter starts by presenting MIP models to independently optimise LT and MT production scheduling horizons. LT production scheduling optimises the yearly extraction sequence, while MT production scheduling optimises the half-yearly extraction sequence. It is important to note that MT production scheduling cannot be limited to half-yearly scheduling but may also break down the LT production scheduling into quarterly and monthly scheduling, which may improve scheduling granularity.

MIP modelling of mine production scheduling aims to capture uncertainty associated with mining rate, processing rate and ore grade target. It was indicated that the source of mine

scheduling uncertainty is generally from geological, operational and financial variabilities that may have a significant impact on the NPV of a project. Integrating uncertainty in the optimisation of the mine production schedule is necessary to minimise deviations from planned targets such as mining rate, processing rate and the grade of ore processed.

A new MIP model was formulated that integrates LT and MT schedules. The objective function of the model aims to maximise NPV by introducing scheduling penalties as a means of penalising deviations from scheduled production targets or ore extraction sequences. These penalties provide a mechanism for evaluating and addressing potential disruptions or discrepancies that might arise during the execution phase of the integrated mine production scheduling system. The dynamic nature of the developed mathematical models was illustrated through linking variables that ensure looping between LT and MT schedules. Linking variables will dynamically update the LT schedule from the MT schedule and vice-versa to capture scheduling changes in the model.

Furthermore, it was highlighted that creating a half-yearly MT schedule enables a smooth transition from LT scheduling to MT scheduling. Another aspect that was considered to improve the smooth transition is the scheduling reconciliation process. The reconciliation of an integrated mine scheduling system involves the development of spatial and temporal scheduling compliance. Spatial scheduling compliance identifies the location of mining areas and aligns them from one horizon to another. Temporal scheduling compliance ensures enforcing the scheduling time across all the scheduling horizons.

The mathematical model developed in this chapter has closed the knowledge gap identified in this research and has provided a novel dynamic integrated mine production scheduling system that maximises NPV. The next chapter (Chapter 4) presents the methodology based on the GA technique for solving the proposed MIP integrated LT and MT production scheduling system.

4. GA METHODOLOGY FOR SOLVING THE MIP MODEL THAT DYNAMICALLY INTEGRATES THE LT-MT MINE PRODUCTION SCHEDULING SYSTEM

4.1 Overview of Chapter 4

The GA was used in this thesis to solve the MIP integrated LT and MT mine production scheduling system developed in Chapter 3. Chapter 4 presents the motivation and advantages of using the GA to solve the MIP OPMPS optimisation problem. Then the chapter outlines the GA solution procedure in the context of an integrated LT and MT production scheduling problem to create the solution framework. The integrated GA methodology outlines the procedure of the solution approach developed in this thesis. The framework was then written as a pseudocode-algorithm for solving the combined MIP model and GA approach, which was then coded and interpreted in Python programming language. The complete code of the combined MIP model and GA approach is presented in Appendices 8.1 and 8.2.

4.2 Motivation for using a GA methodology

The review on production scheduling suggested that the production scheduling problem is a computationally challenging problem to solve, hence resort was made to using a metaheuristic solution methodology and GA was selected. Therefore, a method of searching through a large space of feasible solutions is required to solve the problem amid hard and soft constraints (Gen and Cheng, 2000; Maleki *et al*., 2020). In addition to the advantages previously mentioned in Section 2.4, the motivation for using the GA technique in this thesis is that:

- The GA is argued to be easier to develop, understand, and code in any programming language (Shishvan and Sattarvand, 2015).
- The process of evolution is recognised as a robust and efficient process that involves the application of a biological process, which is applicable to blocks for the optimisation of production scheduling (Sattarvand, 2009).
- The generic block model structure of an open-pit mine can be represented as a framework of chromosomes making the GA a suitable algorithm to represent scheduling solutions.

- The GA can quickly and systematically provide near-optimal solutions to instances that are unattainable with exact techniques (Verhoeff, 2017).
- The GA has been proven to be adaptable, exhibiting a robust mechanism for integrating constraints and uncertainties into the process of solving problems.
- The structure of the GA technique aids in preventing the inherent global search tendency of being trapped at a local optimal solution (Verhoeff, 2017).
- Crossover and mutation operators can be used to explore and exploit a wide range of feasible options (Muke *et al*., 2021).
- The GA has the potential to simultaneously optimise both the UPL and the production schedule problems (Shishvan and Sattarvand, 2015).

The advantages of GA have contributed to its use in solving mine planning problems, such as the OPMPS problem. Ruiseco (2016) used the algorithm to optimise the pit limits on an operating mining bench under constraints such as grade requirements, equipment availability, mining costs and processing costs. GA's flexibility and robustness in solving the pit limit problem were deemed satisfactory by Ruiseco (2016). GA was also applied by Alipour *et al*. (2017) to solve the production scheduling problem of a hypothetical open-pit copper orebody in two-dimensional (2D) space. The objective function that maximises NPV was subject to access, mining rate and processing rate constraints. The experimental results showed the efficiency of GA in solving the OPMPS problem.

Ahmadi and Shahabi (2018) compared GA's performance to that of Lane's method in optimising processing cut-off grade. The algorithm was guided by targeting high ore grade at the beginning of the operation. Then the algorithm was assessed in terms of NPV generated and production rate achieved from mining, processing and refinery processes. Results from the experiment showed that the GA technique and Lane’s models were convergent, but GA produced more accurate and faster results. Muke *et al*. (2021) applied a GA methodology to optimise the LT production scheduling process of open-pit mining. The objective function maximises NPV subject to mining, processing, grade and block precedence constraints. They implemented the GA technique on two hypothetical mining instances and found that the algorithm delivered good and comparable results.

4.3 Implementation of GA for dynamically integrating long-term and medium-term production scheduling

The GA framework developed in this thesis solved the MIP model formulation of the integrated LT and MT production scheduling presented in Chapter 3. The framework was built in such a way that the procedure is adapted to the OPMPS problem representation, hence the problem must be represented as a string of mining blocks suitable for applying genetic operators. Therefore, the following concepts were adopted for the integrated LT and MT production scheduling process:

- Search space: two search spaces were defined, one for LT production scheduling and the other one for MT production scheduling. The LT scheduling search space was defined within the UPL because it could run up to the LOM. The MT scheduling search space is defined by each LT scheduling solution to align the MT schedule with the LT schedule. Since GA ran consecutively period-after-period, the search spaces of these scheduling horizons got smaller after every scheduling period. This was because the extraction solution for a period was excluded when performing the next search process. Therefore, the algorithm succeeded in integrating and complying with the Mineral Reserve constraint while preventing block duplication.
- Chromosome: represented a string of mining blocks which was the feasible solution that complied with the block precedence constraint and was improved by applying genetic operators. Each discrete block represented a gene and was distinguished by its block number or block identification along the chromosome. The block's economic value carries the biological characteristic of a block. The chromosome’s length was determined by the number of blocks it contained and complied with the mining rate constraint per the scheduling period of LT and MT schedules. A penalty cost was applied when a chromosome violated the mining rate constraint.
- Population: comprised of a randomly generated set of chromosomes of the first generation. The LT initial population was generated from the UPL search space while the MT initial population was generated from the LT extraction sequence solution period-by-period. The size of the initial population covered a large area of the search space to increase the chance of reaching the optimal solution. GA operated in a manner that the average economic value of the initial population was

improved generation after generation. The quality and the size of the initial population had a direct impact on the likelihood of the GA process of either escaping or becoming trapped at a local optimum. As the size of the initial population increased, the likelihood of reaching a near-optimal solution also increased.

- Fitness function: a mathematical representation of the objective function that calculates the total economic value or the fitness of chromosomes. Chromosomes in a generation were ranked from the most valuable to the least valuable based on the fitness function.
- Selection: the selection of a chromosome for reproduction was determined based on its fitness value. Higher economic value indicated a fitter chromosome that increased the likelihood of being selected for reproduction.
- Crossover: was performed using PMX that consisted of exchanging some blocks between two Parent Chromosomes P1(t) and P2(t) that have different values. Weak blocks (blocks with negative BEVs) in P1(t) were replaced by strong blocks (blocks with positive BEVs) from P2(t), and vice versa. The algorithm flagged weak blocks from the bottom-up in P1(t) (or P2(t)), to prevent the violation of block precedence requirements. Strong blocks are flagged from top-down in P2(t) (or P1(t)). The crossover operator was expected to deliver two new Offspring Chromosomes (C1(t) and C2(t)) that had a better fitness value compared to the parent chromosomes.
- Mutation: was performed using the heuristic mutation method that sought to enhance the quality of the Offspring Chromosomes C1(t) and C2(t) produced from the crossover operator by modifying some blocks. The process involved either expanding or shortening the size of a chromosome. In the first situation, blocks were appended to the chromosome, whereas in the second situation, blocks were removed from the chromosome to comply with both the mining and processing rate. A penalty cost applies when mutated C1(t) and C2(t) violate mining and processing rate constraints.
- Termination condition: a satisfactory condition was reached when the average fitness of the new generation dropped off or was approximately the same as the previous generation. This condition enabled the algorithm to reach the optimal

solution for a scheduling period and move from one period to another. The algorithm terminated when the search space was empty.

Figure 4.1 illustrates the GA methodology for solving the MIP model. The framework involves links connecting LT and MT production scheduling, and feedback loops from MT to LT scheduling to update the process. Once the termination condition is met to deliver the solution for a scheduling period, the algorithm moves to the next scheduling period until the last period. The algorithm starts by running the LT scheduling process, which triggers the MT scheduling process through connecting links. Feedback loops enable updating LT scheduling from MT scheduling changes. The algorithm stops when the MT scheduling process is complete. In case any changes occur on mining, processing or grade target parameters; at either LT or MT scheduling, the algorithm must rerun from LT scheduling to apply changes in the system.

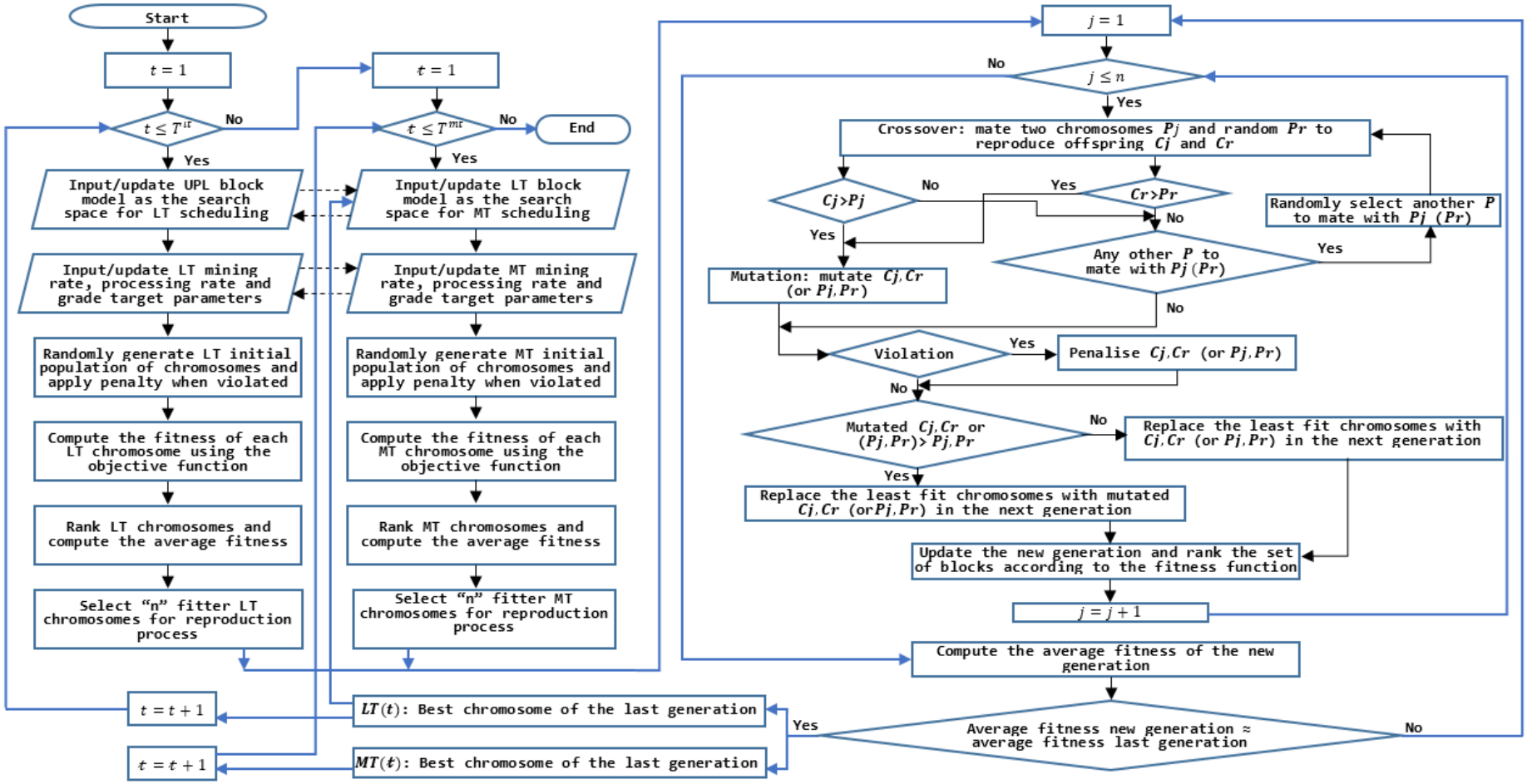


Figure 4.1: Framework of the GA methodology for solving the MIP model
(Muke *et al*., 2025a)

4.4 Python programming language

This thesis used Python programming language to code the GA methodology. The algorithm was written in PyCharm, which is a popular Integrated Development Environment (IDE) for Python and offers a variety of features that enhance productivity and streamline the development process. PyCharm enhances Python development through advanced code editing, debugging, and testing tools, integrated developer tools, and extensive support for databases (Porcu, 2018).

Python is an object-oriented programming language created by Guido Rossum in 1989 (Porcu, 2018). It is a valuable tool for programmers and other practitioners due to its versatility and efficiency in data analysis and visualisation. With the ability to handle large datasets and perform complex calculations, Python enables users to extract valuable insights from their raw data with ease and aid in decision-making processes. Python programming language has advantages that include (Porcu, 2018):

- Clean syntax: Python syntax is straightforward and clear, making it easy to read and write. This enhances code readability and reduces the learning curve for beginners.
- Expressive language: Python allows one to write complex algorithms with fewer lines of code compared to many other programming languages.
- Multi-paradigm: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
- Wide range of applications: Python is used in various domains such as web development, data science, machine learning, artificial intelligence, scientific computing, automation and more.
- Rich library support: Python comes with a comprehensive standard library that provides modules and functions for common tasks, reducing the need to write code from the beginning.

Table 4.1 illustrates the pseudocode algorithm based on the proposed GA framework, while the full Python code is given in Appendices 8.1 and 8.2. Computer specifications on which the running speed depends, are given in Table 4.2.

Table 4.1: The pseudocode algorithm of the GA methodology for solving the MIP model

```
# Initialise parameters
input/update = {s, g, L_i^{lt,RM}, L_i^{mt,RM}, U_i^{lt,RM}, U_i^{mt,RM}, L_i^{lt,RO}, L_i^{mt,RO}, U_i^{lt,RO}, U_i^{mt,RO}, L_i^{lt,RG}, L_i^{mt,RG}, U_i^{lt,RG}, U_i^{mt,RG}}
time period t = 1, t = 1
solution LTt (or MTt) = []

# Main loop
while LT (MT) search_space s size > U_i^{lt,RM}(U_i^{mt,RM}):
      # Initialise population
      while size of population g < 40:
            chromosome ch = []
            while L_i^{lt,RM}; L_i^{mt,RM} > length (ch) or length (ch) < U_i^{lt,RM}; U_i^{mt,RM}:
                  select random ore block [b] from s
                  block_list = b.append(block [b] predecessors)
                  ch.append(block_list)
            return ch
            end while

            g.append(ch)
      return g = set of ch
      end while

      for each ch in G:
          if constraint violation = Yes:
              ch.penalty = penalty_cost(ch)
          ch.fitness = evaluate_fitness(ch)
      evaluate average_fitness of G

      # genetic operators
      while termination_condition is not met:
            select n fitter parents Pj of g
            for each parent Pj:
                select randomly Pr from g
                offspring (Cj, Cr) = crossover(Pj, Pr)
                offspring (Cj, Cr) = mutate(Cj, Cr)

                if constraint violation = Yes:
                    offspring.penalty = penalty_cost(Cj, Cr)
                offspring.fitness = evaluate_fitness(Cj, Cr)

                if offspring.fitness(Cj, Cr) > parents.fitness(Pj, Pr):
                    replace two weak ch in new generation g with (Cj, Cr)
      return best ch in the last generation g = LTt (or MTt)
      end while

      solution.append(LTt (or MTt))
      search space s = s.remove(LTt (or MTt))
      time period t+= 1, (or t+= 1)

return solution
end while
solution = solution.append(s)
end
```

Table 4.2: Computer specifications

Feature	Specification
Operating system	Windows 10 Education
Processor	Intel(R) Xenon(R) CPU X5550 @ 2.67Ghz
RAM	6.00 GB
System type	64-bit OS
Graphics	NVIDIA Quadro FX 580

4.5 Summary of Chapter 4

This chapter described how the MIP model for integrating LT and MT production scheduling was solved using the GA methodology. GA is a random global search and optimisation technique applied in a space of potential solutions, where it improves its solutions through the evolution of population fitness. There are five main steps that the developed GA framework followed which are initialising a population, calculating a fitness function, selection of best chromosomes, crossover and mutation of genes. The framework ensures an integrated scheduling system by connecting the LT and MT production scheduling horizons using scheduling parameters, and that MT scheduling is optimised from LT production scheduling solutions.

The primary advantage of GA comes from the crossover and mutation operations. In the GA methodology, a chromosome is a string or a list of mining blocks. Each block in a chromosome indicates a gene, which is identified by a block number. Blocks carry attributes such as grade, tonnage and economic value or block fitness. The crossover operator on two chromosomes of blocks consists of exchanging some blocks between two Parent Chromosomes P1(t) and P2(t) using the PMX technique. This technique works in such a way that weak genes (blocks with negative BEV) in P1(t) are replaced by blocks with positive BEV from P2(t), and vice-versa. Then a heuristic mutation method is used to improve the quality of the Offspring Chromosome C1(t) and C2(t) generated from the crossover, through the alteration of some blocks to satisfy the processing rate and improve the mining rate.

The GA framework applies penalty costs to chromosomes that do not satisfy the mining and processing constraints. After several iterations, the algorithm terminates when the average fitness of the new population is approximately equal to the average fitness of the last

population. The optimal solution is the best chromosome in the last generation. The next chapter presents the implementation and results of the combined MIP and GA approach when applied on three block models.

5. OPEN-PIT METAL MINING CASE STUDIES FOR THE APPLICATION OF DYNAMICALLY INTEGRATED LONG-TERM-MEDIUM-TERM MINE PRODUCTION SCHEDULING SYSTEM

5.1 Overview of Chapter 5

The implementation and validation of the results of the combined MIP and GA approach are discussed in this chapter. The chapter presents the source of data (geological block models) and the description of the block models that were used in this thesis, namely the Newman, Zuck Small and Surpac block models. The implementation of the combined MIP and GA approach was to prove that the integrated approach can improve NPV as opposed to when the scheduling horizons are independently optimised. The combined MIP and GA approach was implemented on the Surpac block model to independently optimise LT and MT production scheduling. Then the chapter demonstrates the integrated optimisation of LT and MT production scheduling on the Surpac block model. The validation of the combined MIP and GA approach is done by comparing the results obtained against current benchmark solutions already existing from MineLib on the Newman and Zuck Small block models.

5.2 Data sources

This thesis used block models for open-pit mining operations freely downloaded from MineLib (2012). MineLib is a public mining library website that provides block models for testing optimisation problems related to open-pit mine planning. Block models in this library are designed for three classical mining optimisation problems namely the UPL problem, the constrained pit limit (CPIT) problem and the precedence constraints production scheduling problem (PCPSP) (Espinoza *et al*., 2012). The UPL problem involves the selection of a subset of blocks that contains the maximum undiscounted value under slope precedence constraints without taking into consideration the aspect of time (Jélvez *et al*., 2019). The CPIT problem incorporates a temporal dimension as it provides a block scheduling sequence for extraction over a fixed number of periods. The CPIT objective function maximises the discounted value under both slope precedence and mining rate constraints, in which block destinations are fixed in advance (Jélvez *et al*., 2019). The PCPSP problem extends the CPIT problem simply by considering multiple possible destinations for blocks and by complying with general site constraints, such as blending in which the quality of processed material is controlled (Jélvez *et al*., 2019).

The two hypothetical block models that were downloaded from MineLib are the Newman and Zuck Small block models. The reason for using these block models was to validate and prove the practicability of the combined MIP and GA approach and to establish a comparison with existing optimisation techniques applied to these block models. Additionally, the use of the Newman and Zuck Small block models is important to show the performance of the combined MIP and GA approach when the size of the block model increases. The validation of the combined MIP and GA approach was further demonstrated in the study by Muke *et al.*, (2025b) on a larger block model, the KD block model available from MineLib. However, to initially test the efficiency and robustness of the combined MIP and GA approach on actual geological and mining conditions, this thesis used a block model for training and experimental purposes downloaded from Geovia Surpac® software (Geovia, 2024). The next section describes these block models.

5.3 Description of block models and GA input parameters

Newman is an academic open-pit block model constructed by Alexandra Newman, while the Zuck Small model is a fictitious open-pit block model constructed by Mark Zuckerberg (Espinoza *et al.*, 2012). Newman and Zuck Small block models are both gold-bearing orebodies suitable for testing UPL, CPIT and PCPSP optimisation problems. Each block model is made up of four files containing different data. The first file describes the orebody and surrounding rock. This file contains information such as block identifier, block location, and geological and mining characteristics of the orebody. The second file provides the precedence relationship for each block to indicate the immediate block predecessors in the model. The third and fourth files contain benchmark solutions of constrained pit limit and production scheduling, respectively. Figure 5.1 illustrates the spatial view of Newman and Zuck Small block models.

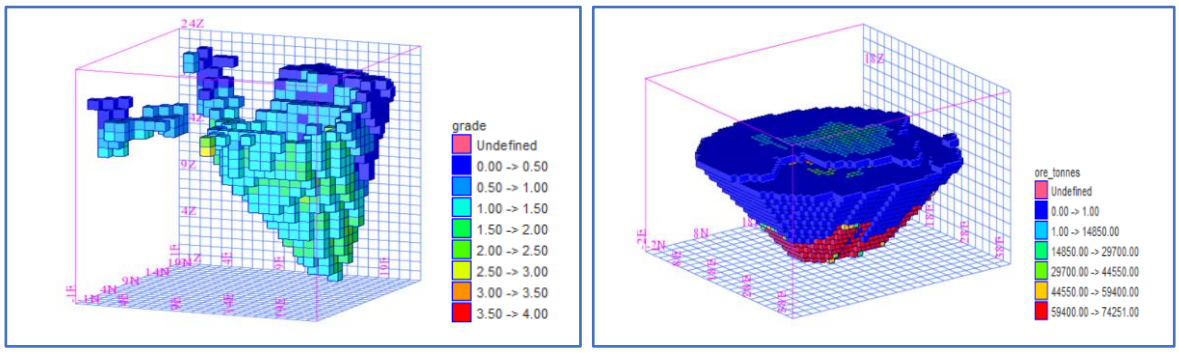


Figure 5.1: Spatial view of the Newman (left) model with grade variation and Zuck Small (right) model with ore tonnage variation per block

The Surpac block model represents a subsurface geology that provides valuable information about the geological and physical distribution of a real-world gold deposit (Geovia, 2024). This model was created for training and experimental purposes. It was constructed by dividing the orebody and surrounding rock into selectable blocks sized 10 m ×10 m ×10 m, suitable for ten metres of bench height. Each block contains descriptive attributes such as grade, lithology, density, mining cost, processing cost, commodity price, and block economic value. Pit design and surface topography files were used to constrain the block model to reveal the Mineral Reserves on which the combined MIP and GA approach was implemented. Figure 5.2 shows the Surpac constrained block model within the pit design.

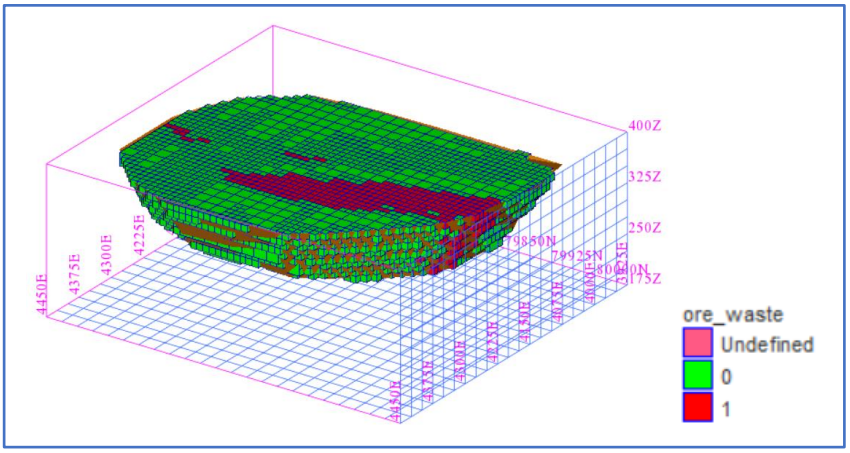


Figure 5.2: Spatial view of Surpac block model showing ore (1) and waste (0) blocks

The implementation of the combined MIP and GA approach on the Newman, Zuck Small and Surpac block models required mine scheduling input data that describes geotechnical, economic and mining/processing details in addition to geological characteristics. Data available for these block models provides mining/processing parameters such as minimum and maximum rates. Economic parameters include discount rate and block economic value, while geotechnical parameters include slope angle defined by the maximum number of direct block predecessors of each block. Scheduling data requirements for the Newman, Zuck Small and Surpac block models are presented in Table 5.1.

Table 5.1: Description of input parameters for the Newman, Zuck Small and Surpac block models

(Source: MineLib, 2012; Geovia, 2024)

Parameter	Newman	Zuck Small	Surpac model
$\mathcal{N}$	1,060	9,400	12,174
Γ_n	5	32	9
d	0.08	0.1	0.1
$g_{n,c}$ [g/tonne]	-	-	0.27
$r_{c,i,t}$; $r_{c,i,t}$ [%]	100	100	100
$P_{c,i,t}^{lt}$; $P_{c,i,t}^{mt}$ [$/tonne]	-	-	1,900
$L_i^{lt,RM}$ [tonne]	≥1,400,000	-	≥3,400,000
$U_i^{lt,RM}$ [tonne]	≤2,000,000	≤60,000,000	≤3,800,000
$L_i^{lt,RO}$ [tonne]	≥900,000	-	≥650,000
$U_i^{lt,RO}$ [tonne]	≤1,100,000	≤20,000,000	≤1,000,000
$L_i^{mt,RM}$ [tonne]	≥700,000	-	≥1,700,000
$U_i^{mt,RM}$ [tonne]	≤1,000,000	≤30,000,000	≤1,900,000
$L_i^{mt,RO}$ [tonne]	≥450,000	-	≥325,000
$U_i^{mt,RO}$ [tonne]	≤550,000	≤10,000,000	≤500,000

Note: the $ denotes the United States dollar

The implementation of the combined MIP and GA approach on these block models requires initialising the algorithm by providing input parameters presented in Table 5.1. In addition to these input parameters, the combined MIP and GA approach requires to be set up with GA parameters to enable the implementation of the algorithm. GA parameters affect both the performance and the outcome results. Table 5.2 describes input GA parameters used in this thesis.

Table 5.2: Description of input parameters for the GA methodology

Parameter	Description
Initial generation of chromosomes	The algorithm uses a random block selection to create chromosomes that comply with the block precedence constraint.
Generation or population size	The size of generations is equal to 40 chromosomes for better coverage of the solution space.
Chromosome length	The length of chromosomes varies between the minimum and maximum mining rate constraints, measured in number of blocks.
Selection probability for crossover	The probability of a chromosome being selected for crossover is proportional to the fitness value of the chromosome.
Partial-mapped crossover rate	This is proportional to the number of weak blocks along two parent chromosomes. Blocks are crossed over together with their block predecessors to comply with the block precedence constraint.
Heuristic mutation rate	This is proportional to the number of weak blocks along a chromosome. A block is mutated together with its block predecessor to comply with the block precedence constraint.
Optimisation criteria for generations	A satisfactory solution is found when there is no increase in average fitness value between successive generations.
Mining penalty cost	This is equal to the total fitness value of violating blocks (ore and waste) along a chromosome.
Processing penalty cost	This is equal to the total fitness value of violating ore blocks along a chromosome.
Grade penalty cost	This is equal to the total fitness value of violating ore blocks considering grade value along a chromosome.
Terminating criteria of the algorithm	No more available blocks in the block model or the search space.

5.4 Optimisation of long-term mine production scheduling on the Surpac block model

As described in Chapter 4, the GA methodology functions by searching and optimising open-pit mine production scheduling in a given search space of feasible solutions. The combined MIP and GA approach was used to independently optimise LT production scheduling of Surpac block model. The independent optimisation of the LT schedule does not inform nor consider feedback information from MT production scheduling. Results of GA parameters such as average fitness value of generations, solution space, number of generations and chromosome fitness value are discussed in the next sub-section.

5.4.1 Discussion of GA parameters for long-term mine production scheduling

Application of the combined MIP and GA approach aimed to find optimum blocks that must be mined in every LT scheduling period. The optimisation process of the algorithm evolved the fitness value of chromosomes or cash flow values of the first population by applying genetic operators such as PMX and heuristic mutation. The average fitness value of the first population was improved generation after generation until the termination condition was met. The size of each generation was 40 representing the number of chromosomes randomly generated in the first generation. Each generated chromosome is a feasible solution that complies with the block precedence constraint. The optimal solution of a scheduling period was the best chromosome in the last generation in that period. After finding the solution for a scheduling period, the algorithm moves to the next scheduling period up to the end of the LT horizon. The algorithm delivered 10 periods for the optimisation of LT production scheduling. Figures 5.3 to 5.11 show the search space of feasible solutions from Periods 1 to 9, respectively. Period 10 was the last period, where the algorithm could simply generate a unique chromosome, which represented the scheduling solution in this period.

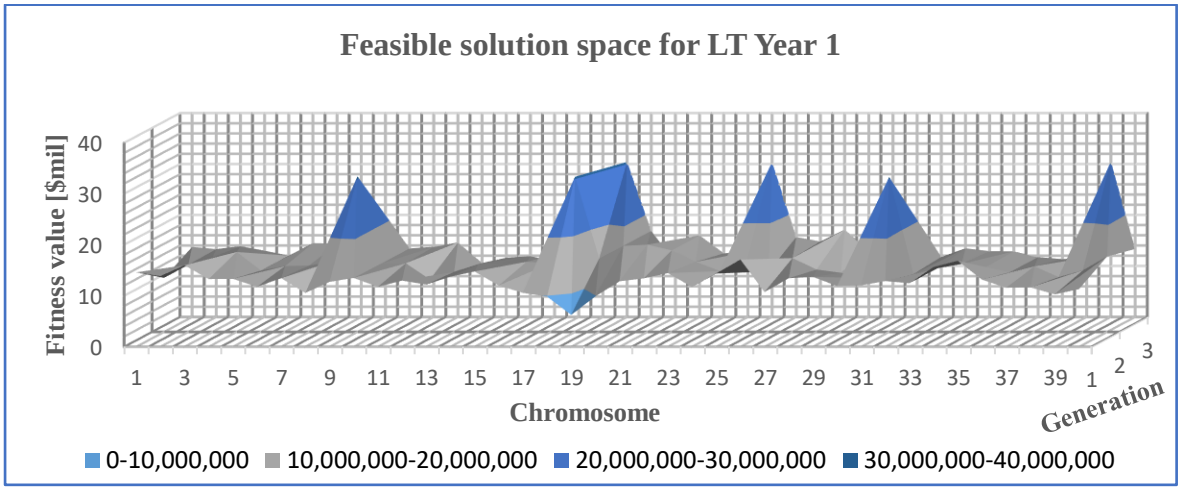


Figure 5.3: Feasible solution space for the optimisation of the LT production schedule in Period 1

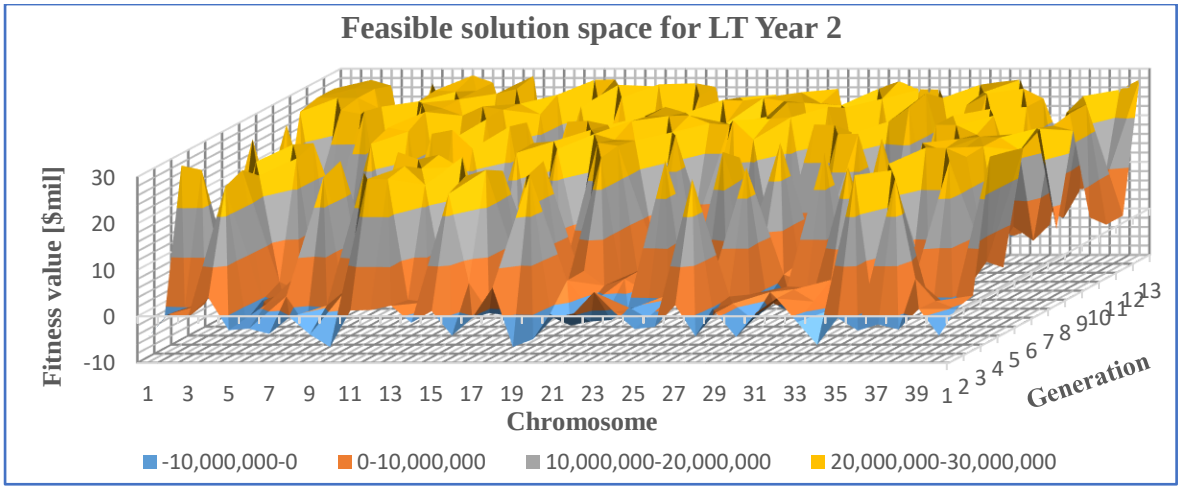


Figure 5.4: Feasible solution space for the optimisation of the LT production schedule in Period 2

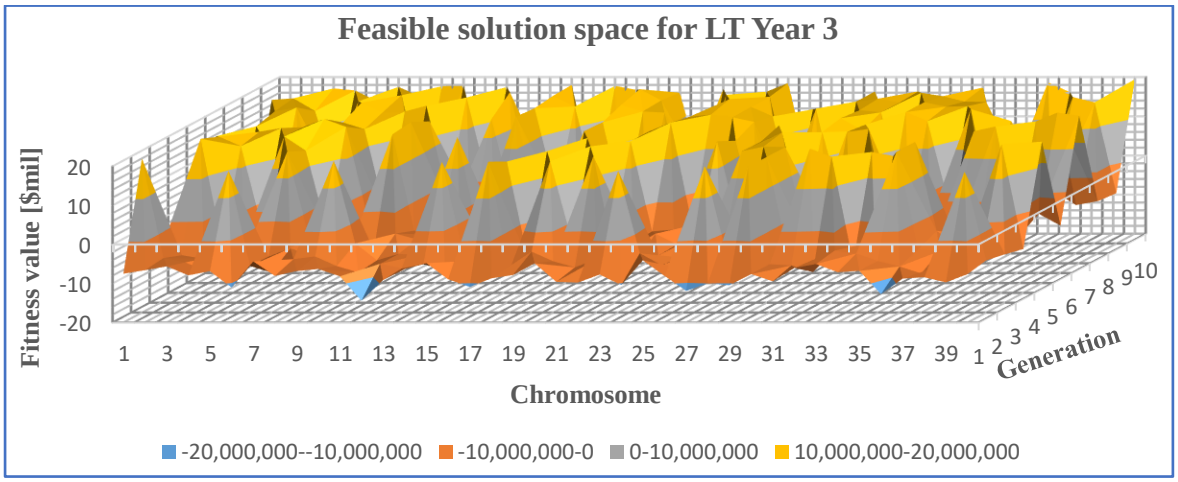


Figure 5.5: Feasible solution space for the optimisation of the LT production schedule in Period 3

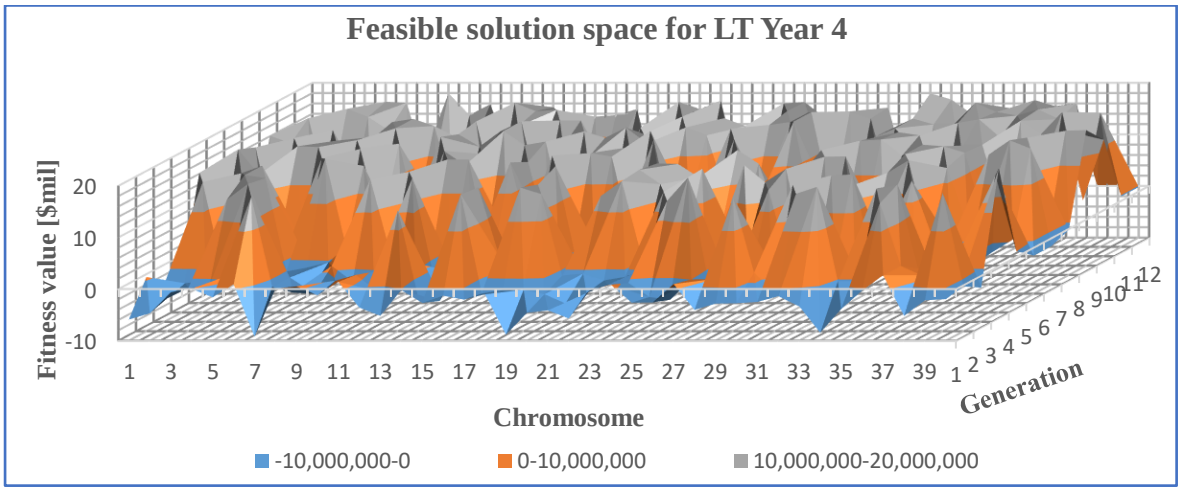


Figure 5.6: Feasible solution space for the optimisation of the LT production schedule in Period 4

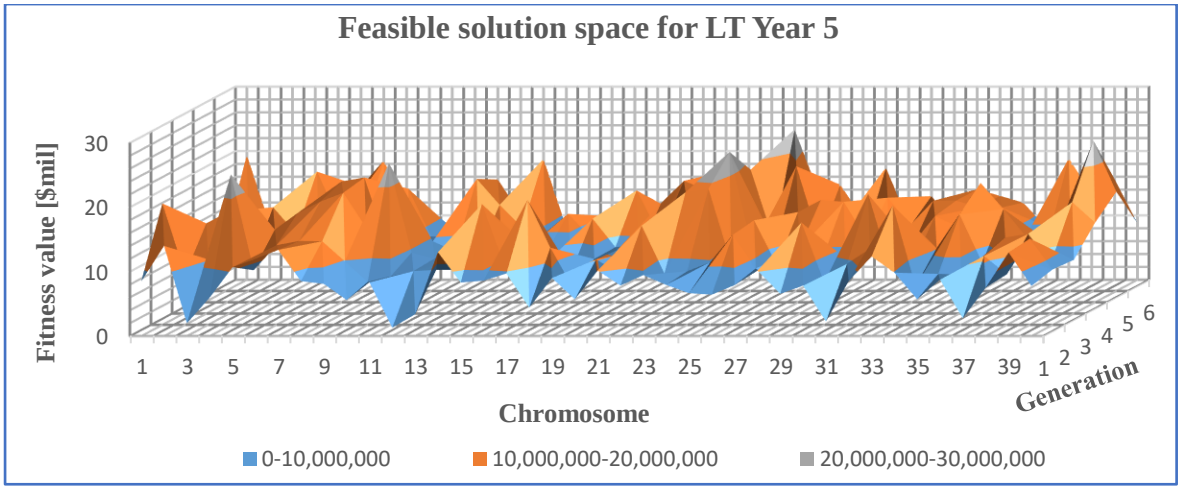


Figure 5.7: Feasible solution space for the optimisation of the LT production schedule in Period 5

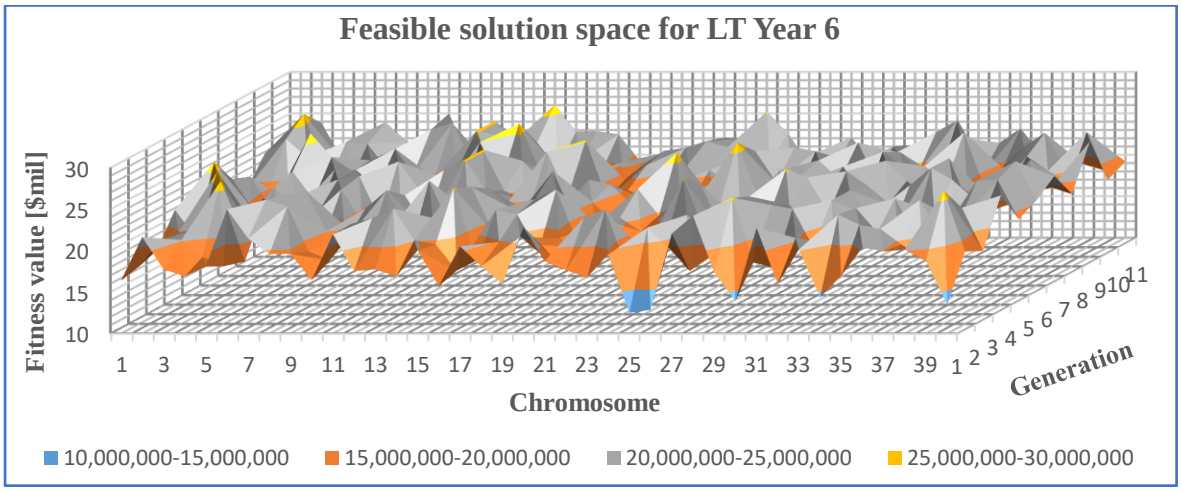


Figure 5.8: Feasible solution space for the optimisation of the LT production schedule in Period 6

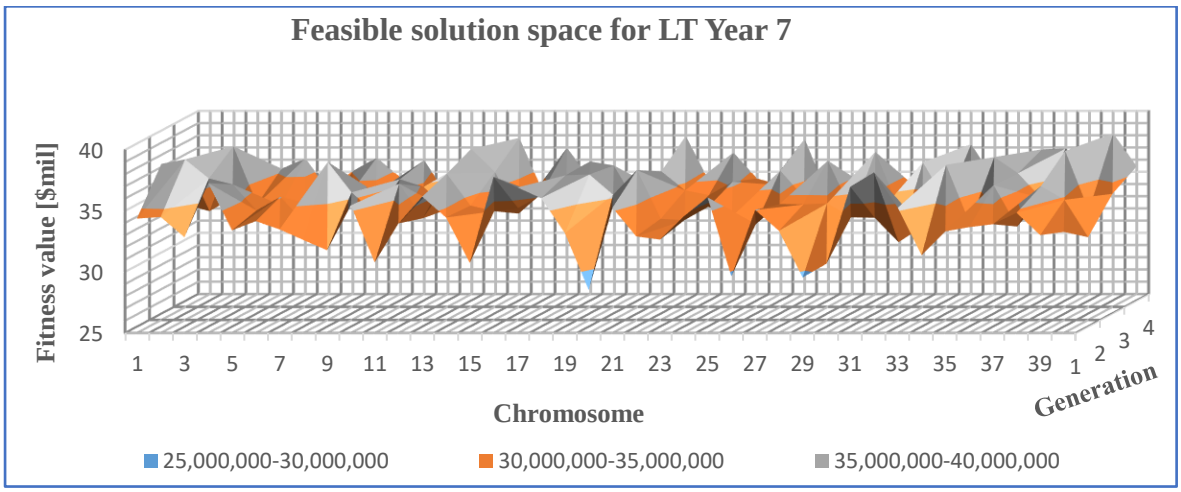


Figure 5.9: Feasible solution space for the optimisation of the LT production schedule in Period 7

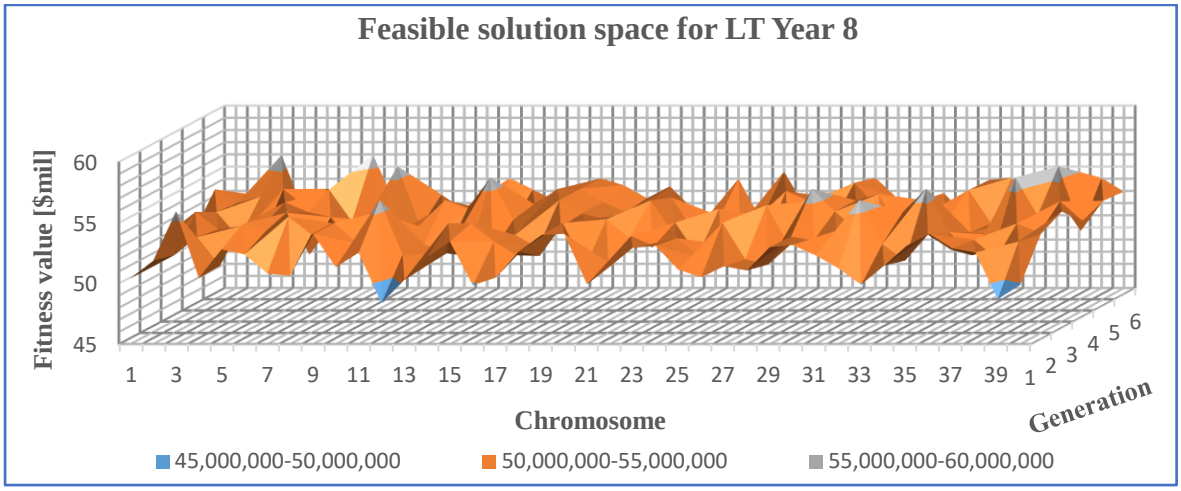


Figure 5.10: Feasible solution space for the optimisation of the LT production schedule in Period 8

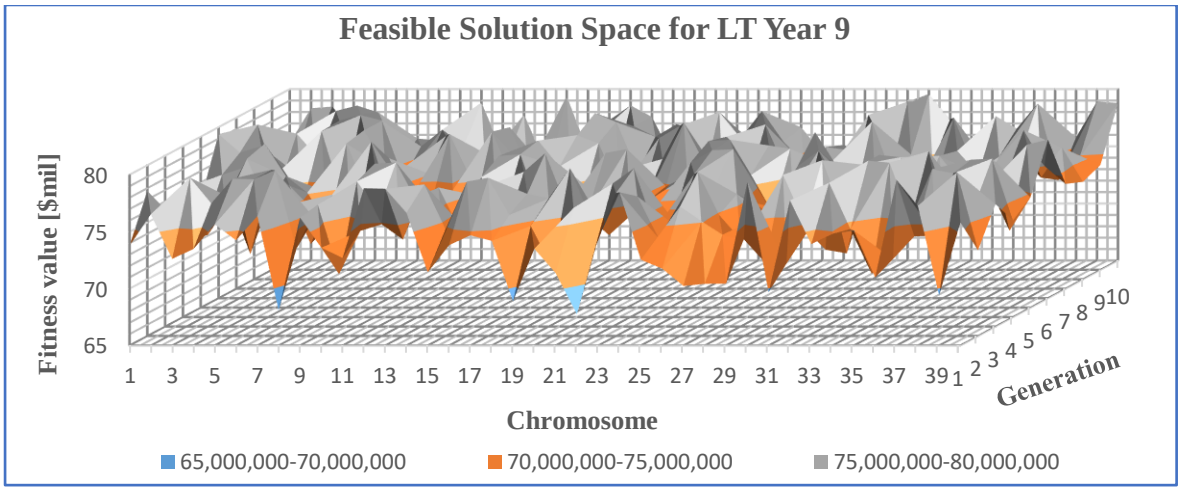


Figure 5.11: Feasible solution space for the optimisation of the LT production schedule in Period 9

Figures 5.3 to 5.11 show that the combined MIP and GA approach can find solutions after reaching the same number of generations in some periods and different number of generations in other periods. The lowest number of generations (3) for the optimisation of LT production scheduling, was reached in Period 1 (Figure 5.3), while the highest number of generations (13) was reached in Period 2 (Figure 5.4). Solutions for Periods 3 (Figure 5.5) and 9 (Figure 5.11) were found after the same number of generations (10); while optimum solutions for Periods 5 (Figure 5.7) and 8 (Figure 5.10) were found after the same number of generations (6). Optimum solutions for Periods 4 (Figure 5.6), 6 (Figure 5.8) and 7 (Figure 5.9) were found after 12, 11 and 4 generations, respectively. Nevertheless, another run of the algorithm may lead to a different number of generations and average fitness values despite using the same inputs due to the randomness of the algorithm. The local optimal

solution from each generation was defined by the highest fitness value of the best chromosome. The use of the PMX and heuristic mutation enabled the algorithm to escape local optimal solutions to reach the global optimal solution in the last generation, from which chromosomes reached their optimal fitness values. When the termination condition is met, the best chromosome of the last generation was considered to be the optimal scheduling solution for the period. Figure 5.12 shows the average fitness values of the first and last generations for each LT scheduling period.

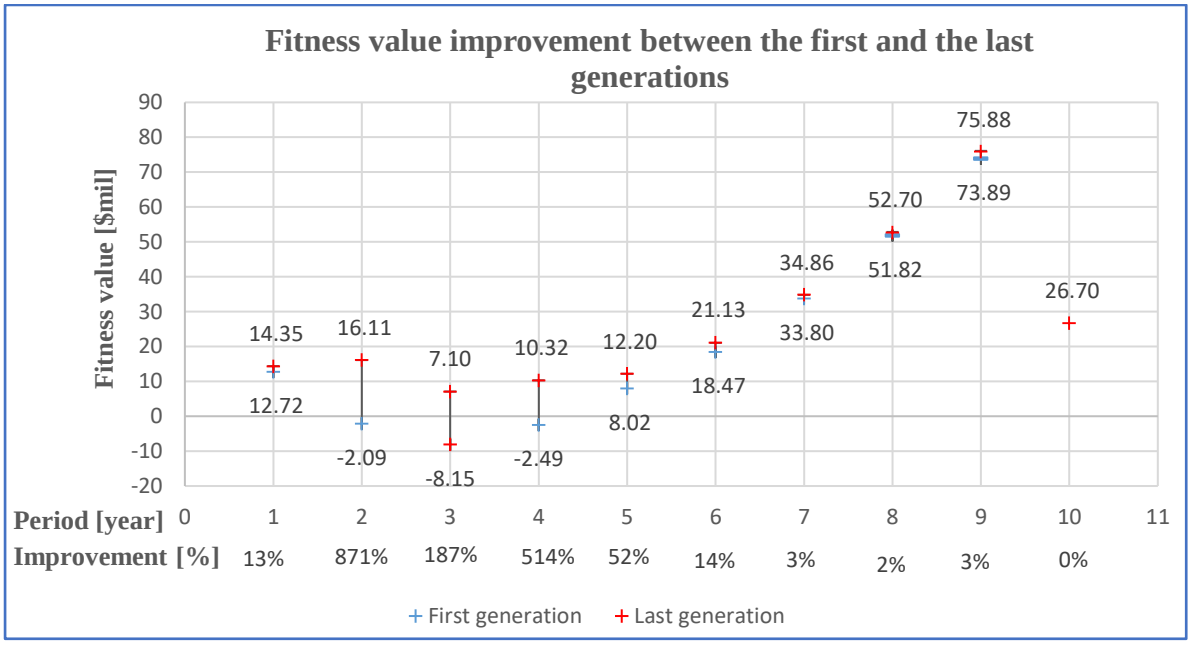


Figure 5.12: Average fitness value of the first and the last generations for each period of the LT production schedule

Results in Figure 5.12 indicate that there are improvements in the average fitness values from the first generation to the last generation for each LT scheduling period. The highest improvement of 871% was achieved in Period 2 where the average fitness value of generations increased from -$2.09 million to $16.11 million, from the 1st to the 13th generations. This is because of the search space which is relatively larger in Period 2 with the highest variability of the average fitness values of generations involved in the optimisation process of Period 2. Figure 5.12 also shows the second and third highest improvements of 514% and 187% in Periods 4 and 3, respectively. The lowest improvement was achieved in Period 8 where the average fitness value increased from $51.82 million to $52.70 million from the 1st to the 6th generations. This is because the size of the search space is smaller with the lowest variability of average fitness values of generations involved in the optimisation of Period 8. Periods 1, 5, 6, 7 and 9 underwent improvements in the average

fitness values of 13%, 52%, 14%, 3% and 3%, respectively. The last Period 10 was not optimised because it contains the remaining blocks to the depletion of the Mineral Reserves. Figure 5.13 shows the fitness values over time periods of the best chromosome in the last generation.

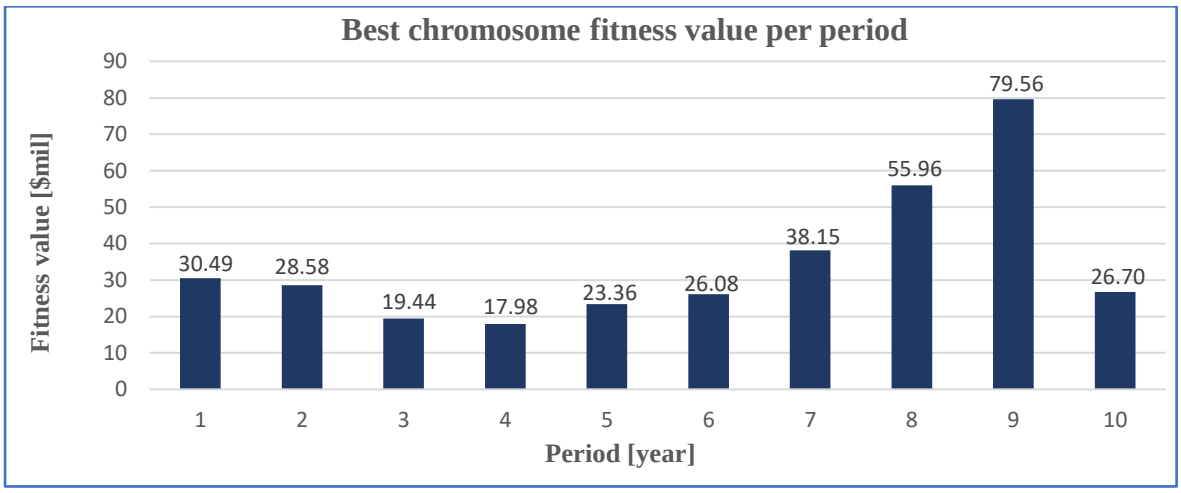


Figure 5.13: Fitness value of the best chromosome per period for the optimisation of the LT production schedule

Although Period 9 had the second lowest average fitness value improvement of 3% (Figure 5.12), Figure 5.13 shows that Period 9 had the highest chromosome fitness value of $79.56 million of the last generation obtained after 10 generations (Figure 5.11). The lowest chromosome fitness value of $17.98 million was achieved after 12 generations (Figure 5.6) with an average fitness value improvement of 514% from the 1st to the 12th generations in Period 4. The highest average fitness value improvement of 871% from the 1st to the 13th generation shown in Figure 5.12 had a chromosome fitness value of $28.58 million in Period 2 after 13 generations. The lowest average fitness value improvement from the 1st to the 6th generations shown in Figure 5.12 had a chromosome fitness value of $55.96 million in Period 8 obtained after 6 generations. Results also show that the chromosome fitness value of $30.49 million was obtained in Period 1 with an average fitness value improvement of 13% from the 1st to the 3rd generation and with the lowest number of generations of 3. Periods 3, 5, 6 and 7 had chromosome fitness values of $19.44 million, $23.36 million, $26.08 million and $38.15 million, after 10, 6, 11 and 4 generations, respectively. The last Period 10 had a fitness value of $26.70 million.

5.4.2 Discussion of long-term mine production scheduling results

The independent optimisation of the LT production scheduling process on the Surpac block model delivered a schedule of 10 years LOM. The schedule determines blocks that must be mined in every year of the LT schedule and the associated cash flow. Figure 5.14 shows a 3D view of the independently optimised LT production schedule.

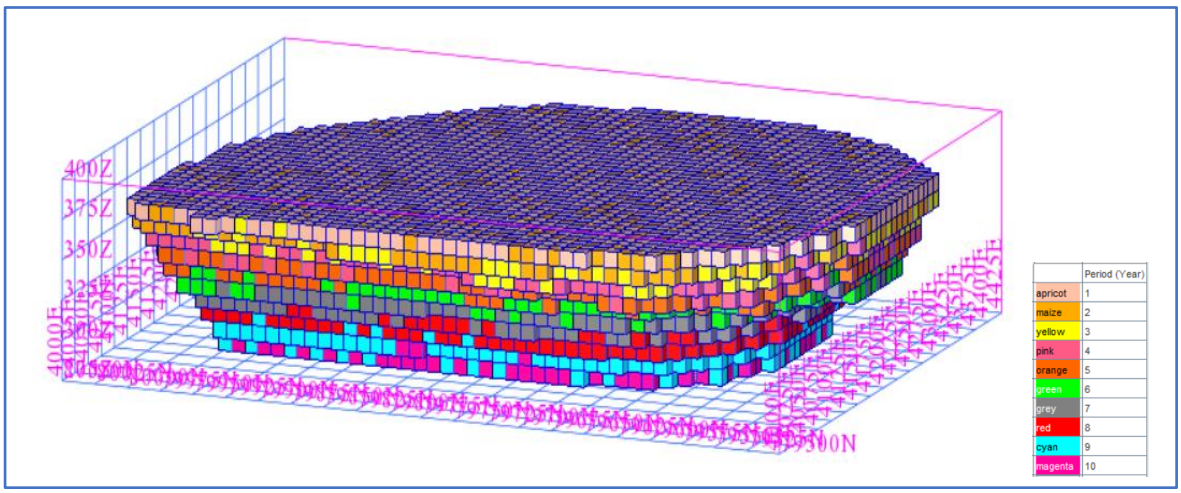


Figure 5.14: A 3D view of the LT production schedule for the Surpac block model

The solution for each LT scheduling period was defined by the best chromosome representing a list of blocks with the highest cash flow. The list of blocks of each LT scheduling period complies with the practicality of the schedule defined by the 1:9 pattern block precedence constraint (Table 5.1). A block cannot be mined if nine blocks directly located above it are not mined. The set of best chromosomes obtained from Periods 1 to 10 defined the extraction sequence shown in Figure 5.14. This sequence does not consider the ramp access point as the starting point of scheduling, which may further constrain the mining direction on a bench, but the approach determines the best starting point based on the ore grade. The combined MIP and GA approach complies not only with the precedence constraint but also with the mining rate constraint. Block precedence and mining rate constraints were considered as hard constraints that must not be violated because the former ensures scheduling practicality, and the latter ensures reaching the expected LOM duration. Another hard constraint considered was the Mineral Reserves constraint to ensure that the total tonnage of material within the UPL is achieved. Processing rate and grade target constraints were considered soft constraints in the algorithm because they are dependent on the type of available blocks to schedule in every period. Table 5.3 shows the results of LT production scheduling when independently optimised.

Table 5.3: LT production scheduling results of the Surpac block model

Period [year]	No. blocks	Total tonnage [tonne]	Ore tonnage [tonne]	Waste tonnage [tonne]	Stripping ratio	Average grade [g/t]	Cash flow [$mil]
t	$TNB_{i,t}^{lt}$	$RM_{i,t}^{lt}$	$RO_{i,t}^{lt}$	$RW_{i,t}^{lt}$	$\frac{RW_{i,t}^{lt}}{RO_{i,t}^{lt}}$	$RG_{i,t}^{lt}$	$V_{n,t}^{lt}$
1	1,350	3,750,731	876,326	2,874,405	3.28	1.69	30.49
2	1,350	3,766,640	833,988	2,932,652	3.52	1.78	28.58
3	1,348	3,765,573	702,890	3,062,683	4.36	1.87	19.44
4	1,314	3,669,349	670,651	2,998,698	4.47	1.90	17.98
5	1,300	3,629,407	712,636	2,916,771	4.09	1.87	23.36
6	1,261	3,516,688	693,568	2,823,120	4.07	1.94	26.08
7	1,250	3,485,369	765,351	2,720,018	3.55	1.95	38.15
8	1,253	3,493,913	825,105	2,668,808	3.23	2.35	55.96
9	1,250	3,485,627	999,270	2,486,357	2.49	2.25	79.56
10	498	1,388,809	375,084	1,013,725	2.70	2.29	26.70
Total	**12,174**	**33,952,106**	**7,454,869**	**26,497,237**	**3.55**	**1.98**	**346.30**
NPV [$mil] at a discount rate of 10%							**197.17**

A total of 12,174 blocks equivalent to 33.9 Mt of material were scheduled. This is the same number of blocks contained in the block model (Table 5.1). Therefore, the combined MIP and GA approach complies with the Mineral Reserve constraint. The total tonnage achieved in each of LT Periods 1 to 9, ranging between 3.4 Mt and 3.8 Mt, suggests that the combined MIP and GA approach complies with the minimum and maximum mining rates presented in Table 5.1. The total ore tonnage achieved ranges between 670.6 kt and 999.3 kt in Periods 4 and 9, respectively. This also shows that the combined MIP and GA approach complies with minimum and maximum processing rates from Periods 1 to 9. The last period (Period 10) of the schedule consists of 498 blocks equivalent to 1.3 Mt of material containing 375.1 kt of ore, for the depletion of Mineral Reserves from the pit, and was not optimised.

The NPV obtained was estimated to be $197.2 million, which is the total discounted cash flow generated through the LT scheduling periods. Figure 5.15 shows the total tonnage (waste and ore) to mine and the cash flow value from each LT scheduling period.

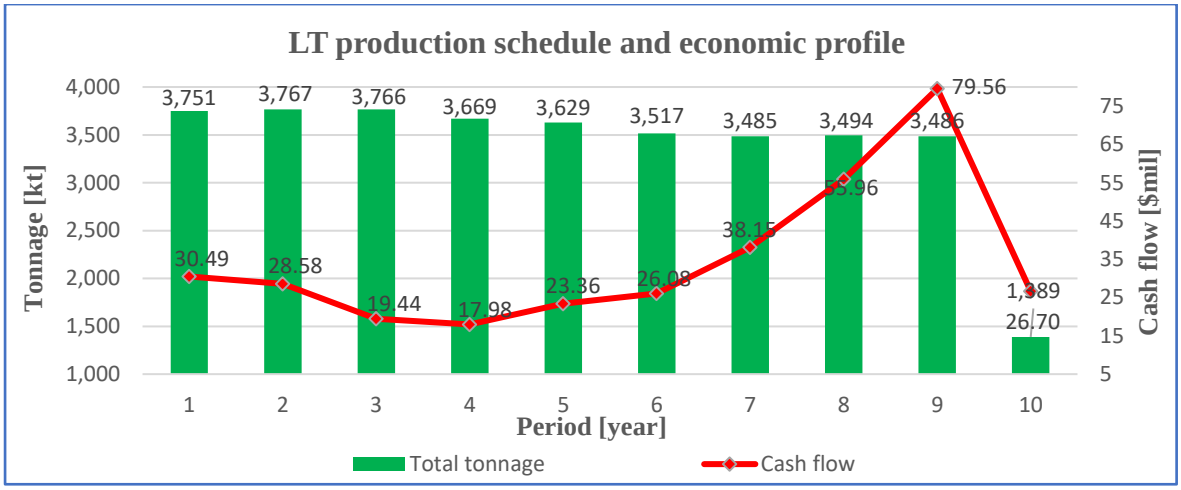


Figure 5.15: Total tonnage and the cash flow value in each period of the LT production schedule

The combined MIP and GA approach delivered positive cash flows in each LT production schedule with an increasing trend from Period 4 to Period 9 and the reason for the peak in Figure 5.15 is explained in more detail in the next paragraph. Positive cash flows throughout the LT schedule are important to increase economic viability and ultimately improve NPV of the project. The graph also shows from Period 1 to Period 9 all total annual tonnages were within the range of 3.4 Mt and 3.8 Mt indicated in Table 5.1. Figure 5.16 shows the ore and waste tonnages, stripping ratio and the average grade for each LT scheduling period.

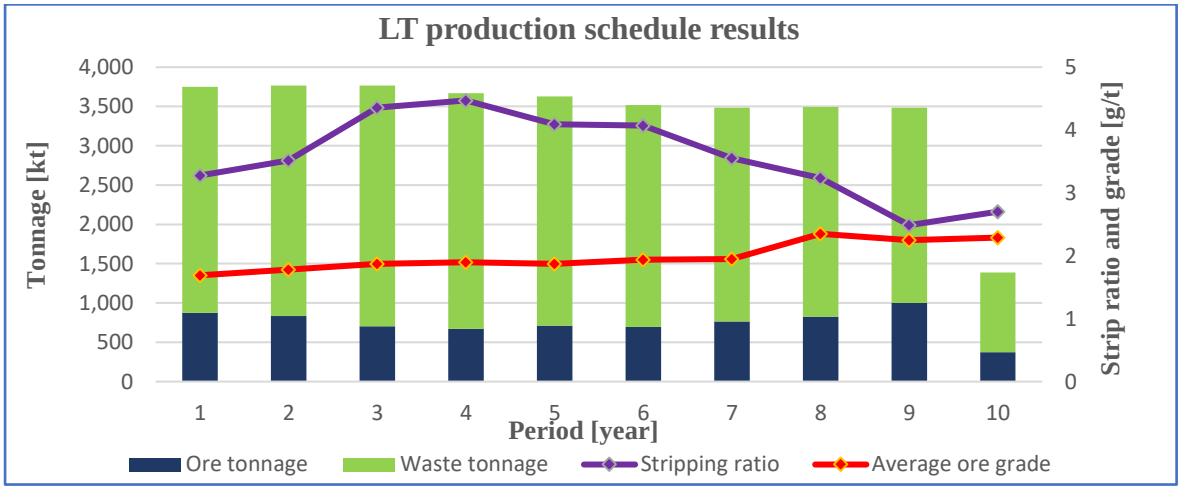


Figure 5.16: Ore and waste tonnages, stripping ratio and average grade in each period of the LT production schedule

Figure 5.16 shows an almost constant average ore grade produced from Periods 1 to 7 to maintain the quality of ore to be fed to the processing plant. From Periods 8 to 10 the average ore grade slightly increases. Stripping ratio ranges between 2.49 and 4.47 showing that an amount of 2.49 Mt of waste must be removed to extract 999.27 kt of ore of 2.25 g/t average

grade in Period 9, and an amount of 3.00 Mt of waste must be removed to extract 670.65 kt of ore of 1.9 g/t average grade in Period 4. The lowest stripping ratio of 2.49 achieved in Period 9 generated the highest cash flow of $79.56 million (Figure 5.15) due to high grade blocks encountered at a deep depth of the orebody. The highest stripping ratio of 4.47 achieved in Period 4 generated the lowest cash flow of $17.98 million (Figure 5.15). This means that a high stripping ratio is not desired because a large amount of waste must be moved to access ore thus lowering cash flow.

5.5 Optimisation of medium-term mine production scheduling on the Surpac block model

This section presents the results from the application of the combined MIP and GA approach to independently optimise MT production scheduling. The independent scheduling optimisation does not connect the MT schedule to the LT schedule, nor allow feedback information from the MT to the LT schedule. The optimisation process of the MT schedule was similar to the LT optimisation discussed in Section 5.4. However, MT production scheduling produces half-yearly production schedules over the first five years of operation. LT and MT scheduling results can be used to determine discrepancies when these scheduling horizons are independently optimised.

5.5.1 Discussion of GA parameters results for medium-term mine production scheduling

Application of the combined MIP and GA approach to independently optimise MT scheduling also used PMX and heuristic mutation for the reproduction process in pairing two chromosomes. Pairing two parent chromosomes generated two offspring chromosomes that are fitter than the parent chromosomes. The size of each generation was 40 representing the number of chromosomes randomly generated in the first generation. Figures 5.17 to 5.26 show the search space of feasible solutions for each generation from Period 1 to 10 of the MT schedule.

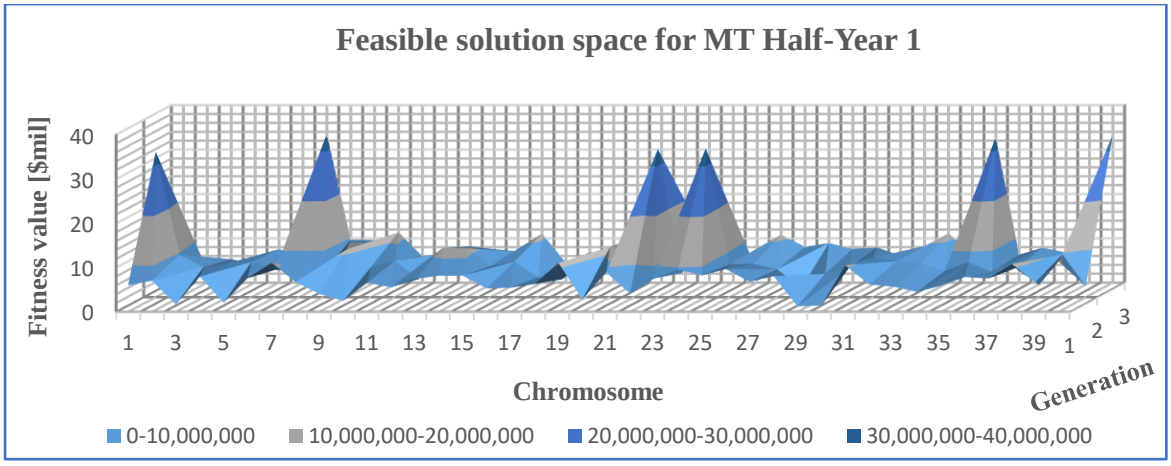


Figure 5.17: Feasible solution space for the optimisation of the MT production schedule in Period 1

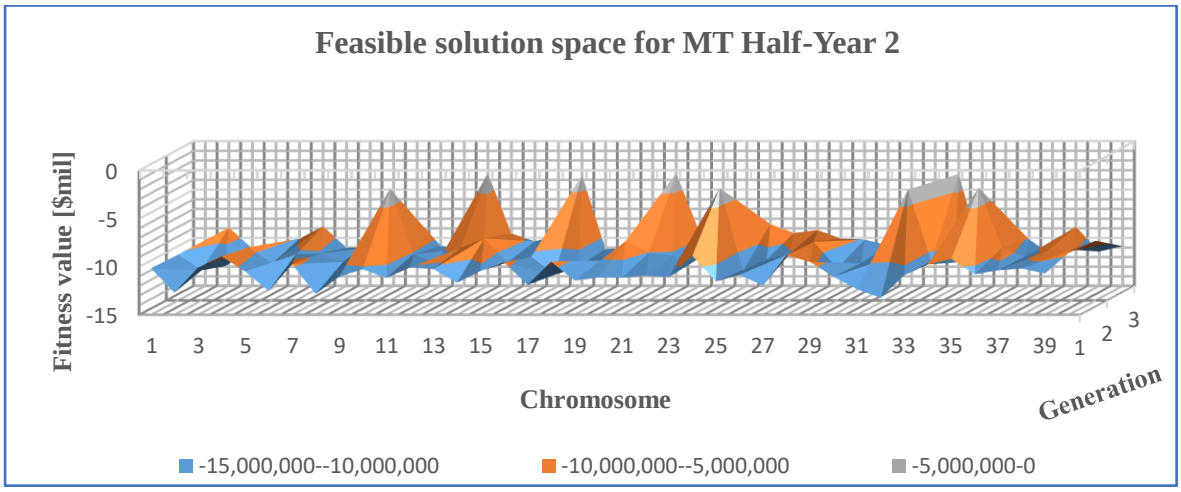


Figure 5.18: Feasible solution space for the optimisation of the MT production schedule in Period 2

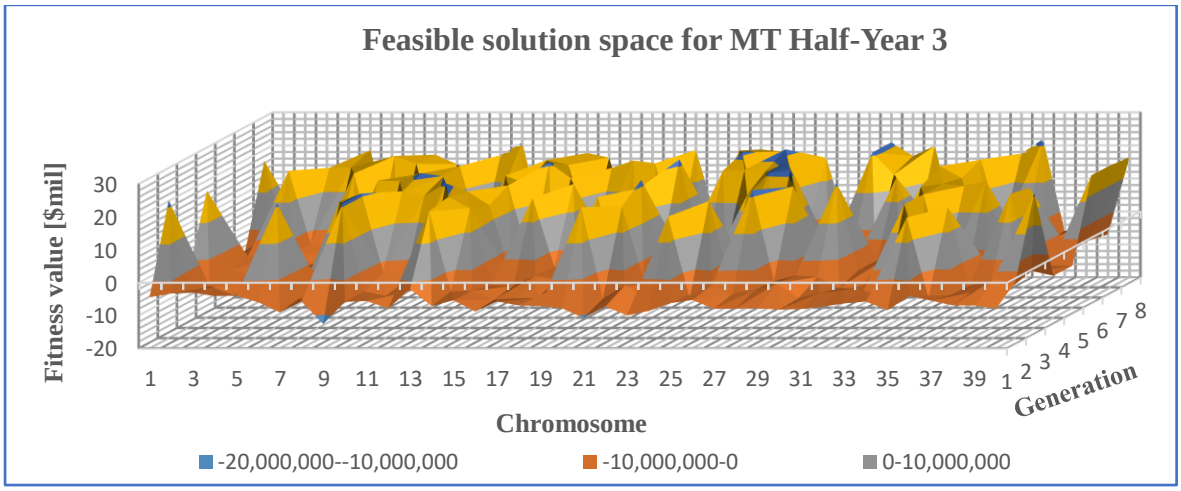


Figure 5.19: Feasible solution space for the optimisation of the MT production schedule in Period 3

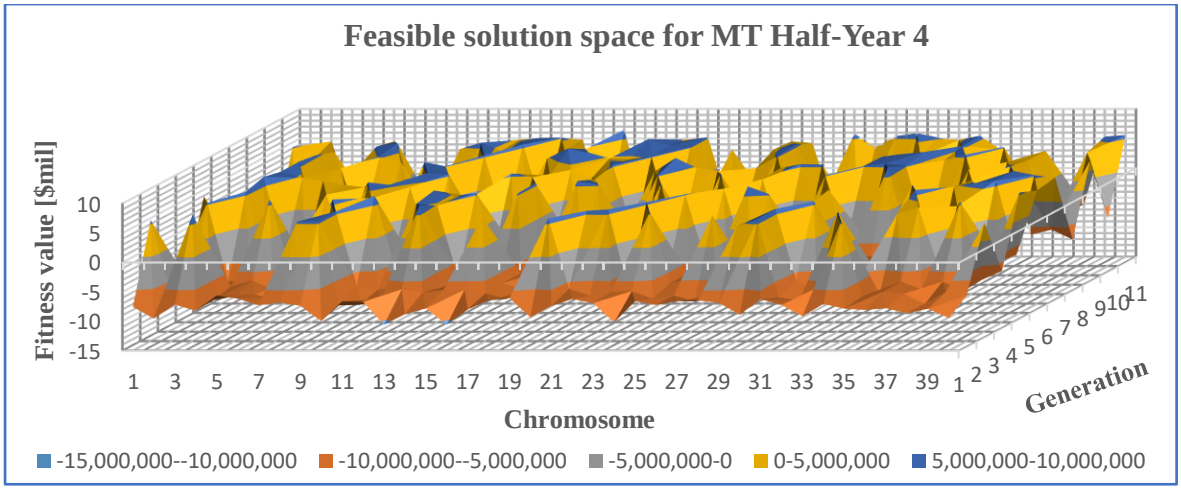


Figure 5.20: Feasible solution space for the optimisation of the MT production schedule in Period 4

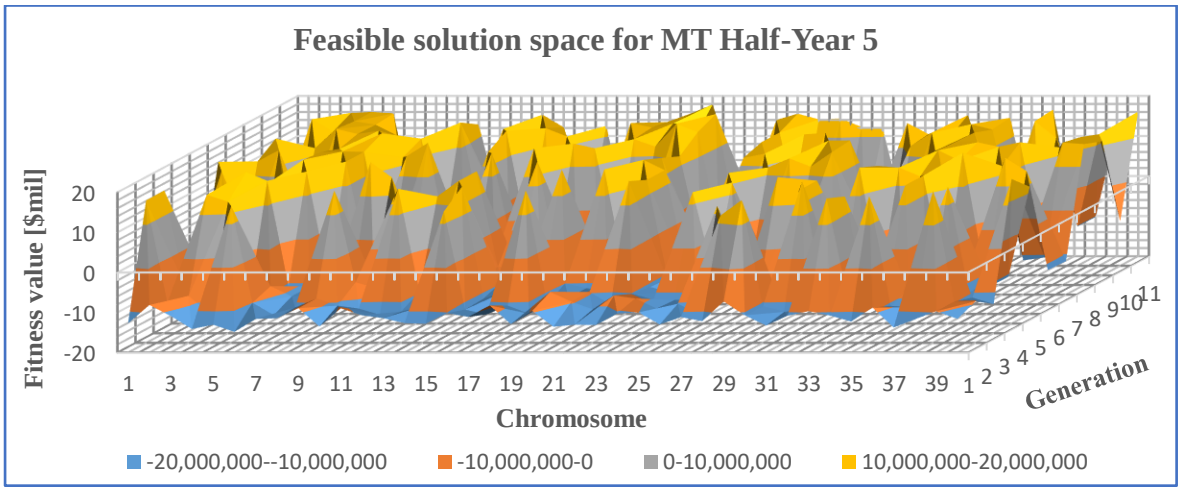


Figure 5.21: Feasible solution space for the optimisation of the MT production schedule in Period 5

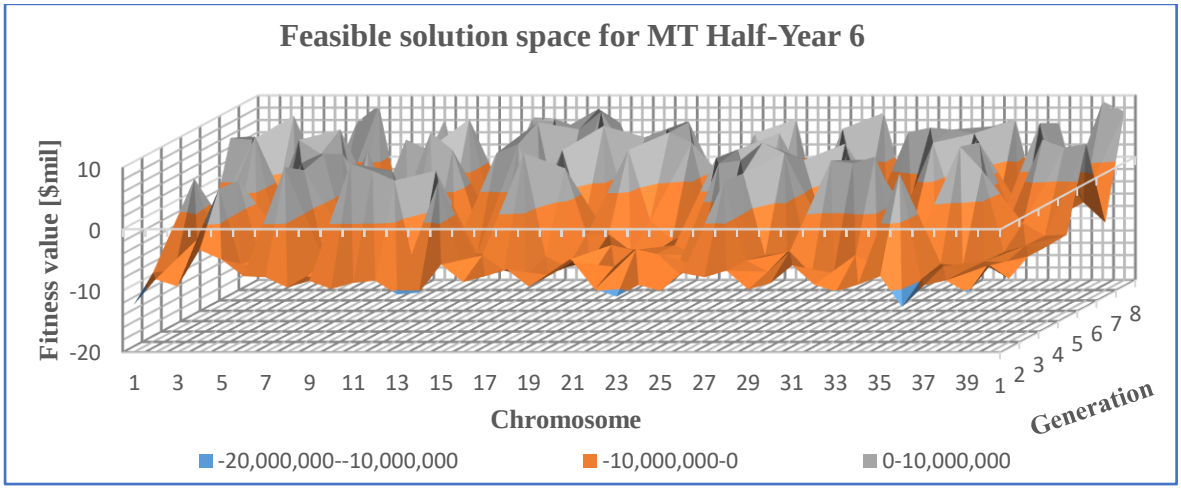


Figure 5.22: Feasible solution space for the optimisation of the MT production schedule in Period 6

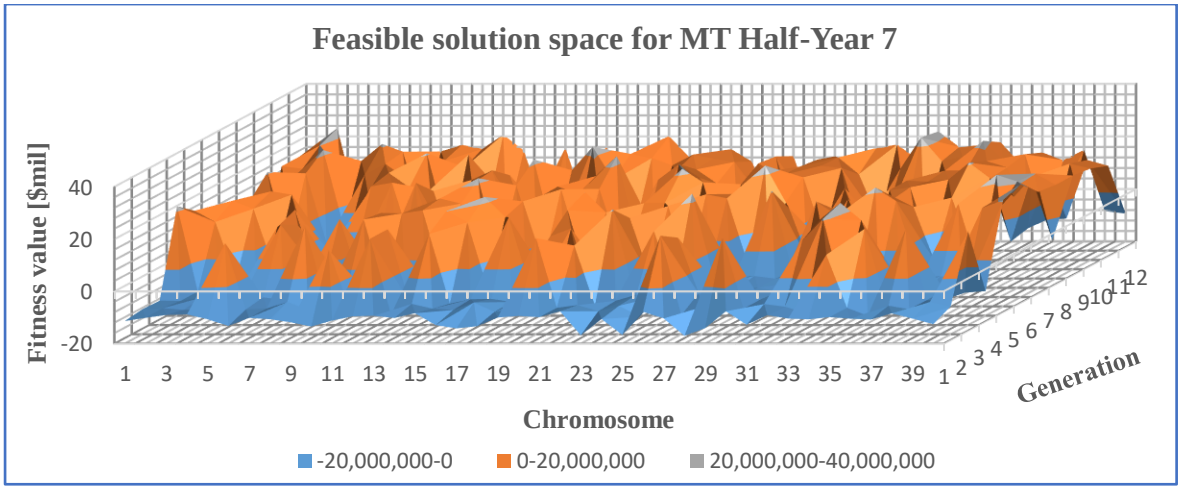


Figure 5.23: Feasible solution space for the optimisation of the MT production schedule in Period 7

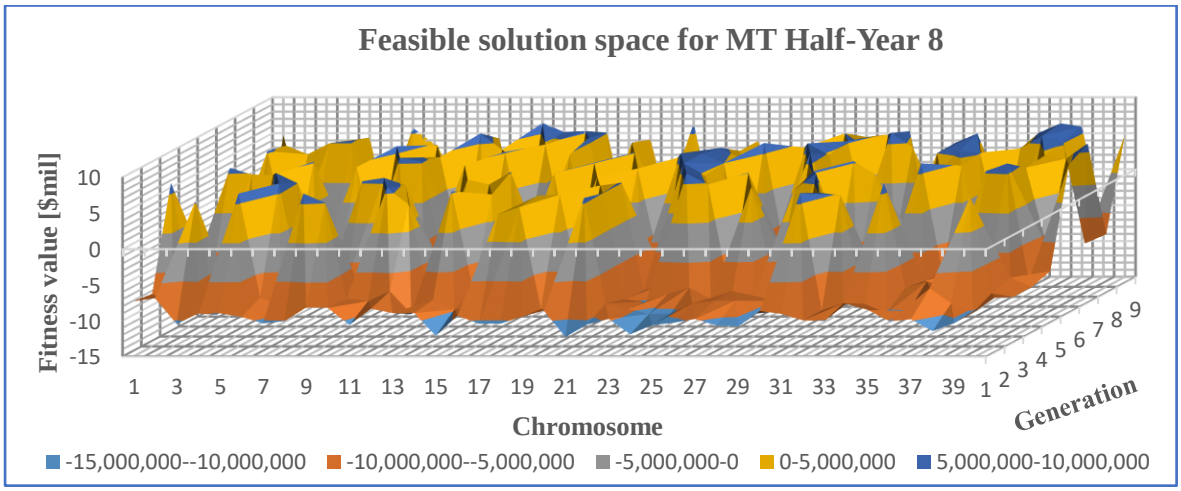


Figure 5.24: Feasible solution space for the optimisation of the MT production schedule in Period 8

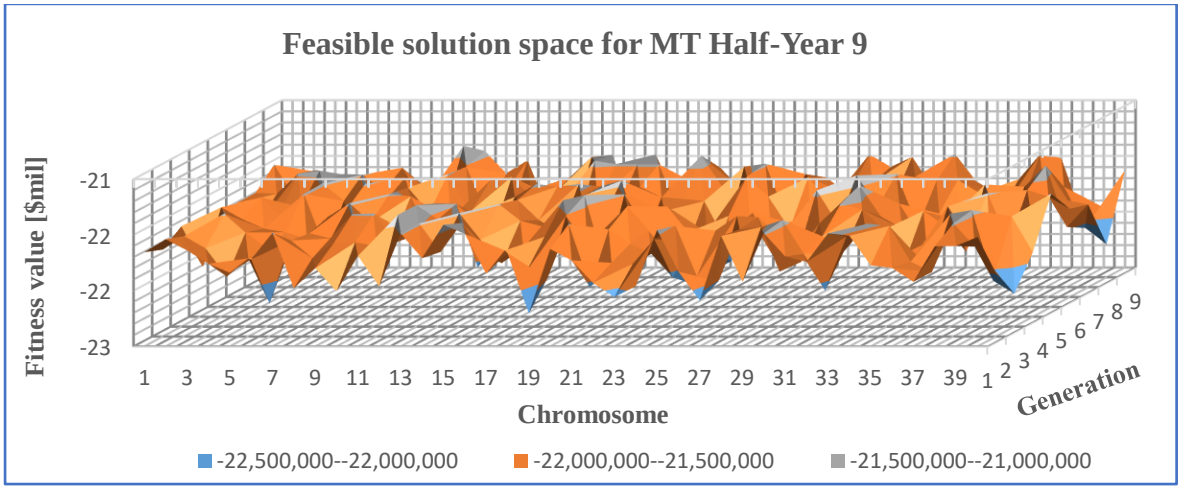


Figure 5.25: Feasible solution space for the optimisation of the MT production schedule in Period 9

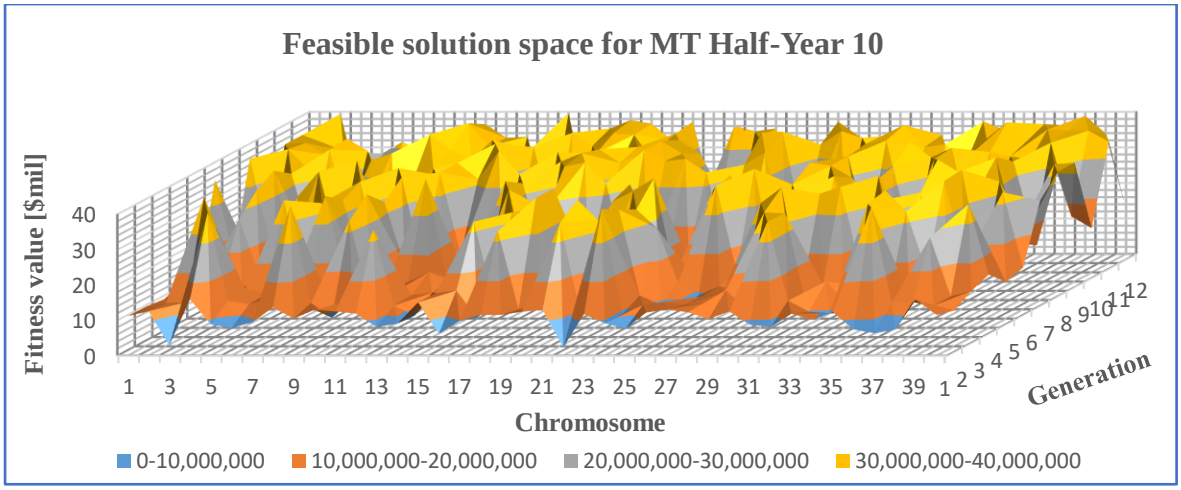


Figure 5.26: Feasible solution space for the optimisation of the MT production schedule in Period 10

As previously mentioned in Sub-section 5.4.1 that for LT production scheduling, different runs of the combined MIP and GA approach may result in different solutions, due to the randomness of the GA algorithm. Results presented in Figures 5.17 to 5.26 show that the lowest number of generations (3) was reached for optimising Periods 1 (Figure 5.17) and 2 (Figure 5.18) of the MT production schedule, while the highest number of generations (12) was reached for optimising Periods 7 (Figure 5.23) and 10 (Figure 5.26). Optimal solutions for Periods 3 (Figure 5.19) and 6 (Figure 5.22) were found after the same number of generations (8), while optimal solutions for Periods 4 (Figure 5.20) and 5 (Figure 5.21) were found after the same number of generations (11). Figures 5.24 and 5.25 show that optimal solutions for Periods 8 and 9, respectively, were found after the same number of generations (9). However, the average fitness value of the first generation was improved generation after generation until no more improvement could be possibly made. Figure 5.27 shows the average fitness values of the first and the last generations for each MT scheduling period.

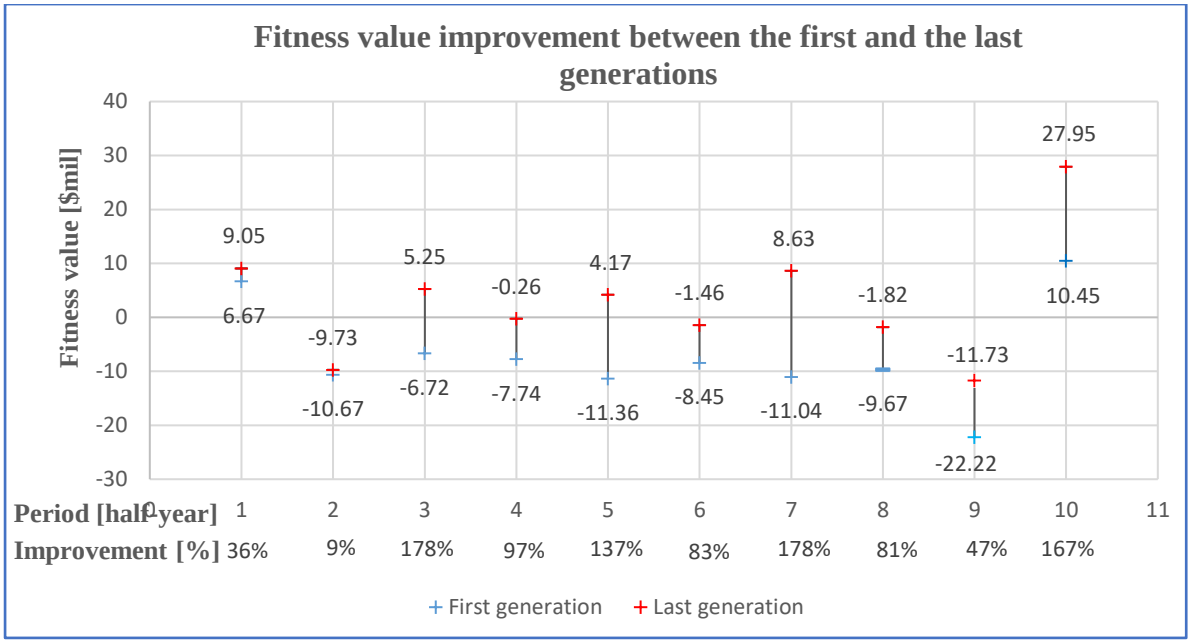


Figure 5.27: Average fitness value of the first and the last generations for each period of the MT production schedule

Results in Figure 5.27 indicate that there were improvements in the average fitness values from the first to the last generations for each MT scheduling period. The combined MIP and GA approach improved negative average fitness values of the first generations to become positive in Periods 3, 5, and 7. These improvements were achieved after some generations of 8 (Figure 5.19), 11 (Figure 5.21) and 12 (Figure 5.23) in Periods 3, 5 and 7, respectively. This shows the efficiency of the algorithm to improve the average fitness value of a generation of chromosomes. The highest improvement of 178% was recorded in Periods 3 and 7 where the average fitness increased from -$6.72 million to $5.25 million and -$11.04 million to $8.63 million, respectively. Period 7 had the highest variability in the average fitness value of generations. Although Periods 2, 8 and 9 have a significant improvement in average fitness values of 9%, 81% and 47% from their first to the last generations, respectively, the periods had negative average fitness values. This is because blocks available in the search space in these periods were mostly waste blocks. The optimal MT schedule solution was selected from the last generation of each MT period. Figure 5.28 shows the fitness values of the best chromosome in the last generation for each MT scheduling period.

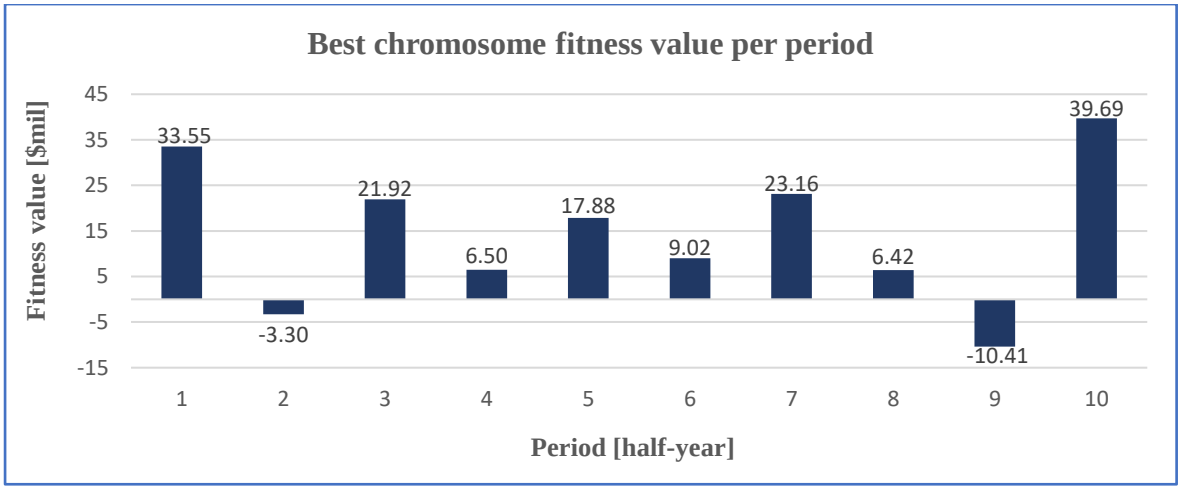


Figure 5.28: Fitness value of the best chromosome per period for the optimisation of the MT production schedule

Results in Figure 5.28 show that the fitness values of the best chromosome in the last generation were positive in eight periods, and negative in two (Periods 2 and 9). The lowest fitness value of the best chromosome was -$10.41 million obtained in Period 9, while the highest fitness value of the best chromosome was $39.69 million obtained in Period 10. It is important to note that Period 6 had a positive fitness value of $9.02 million for the best chromosome, though the average fitness value of the last generation was -$1.46 million as shown in Figure 5.27. This also showed that the fitness value of the best chromosome must be greater than the average fitness value of the last generation, or at least the same if all the chromosomes have the same fitness value.

5.5.2 Discussion of medium-term mine production scheduling results

The optimisation of the MT production scheduling process when done independently delivered a five-year production schedule on a half-yearly basis, resulting in ten MT scheduling periods. This production schedule determines blocks that must be mined every six months of the MT schedule and the associated cash flow. Figure 5.29 shows a 3D view of the independently optimised MT production schedule.

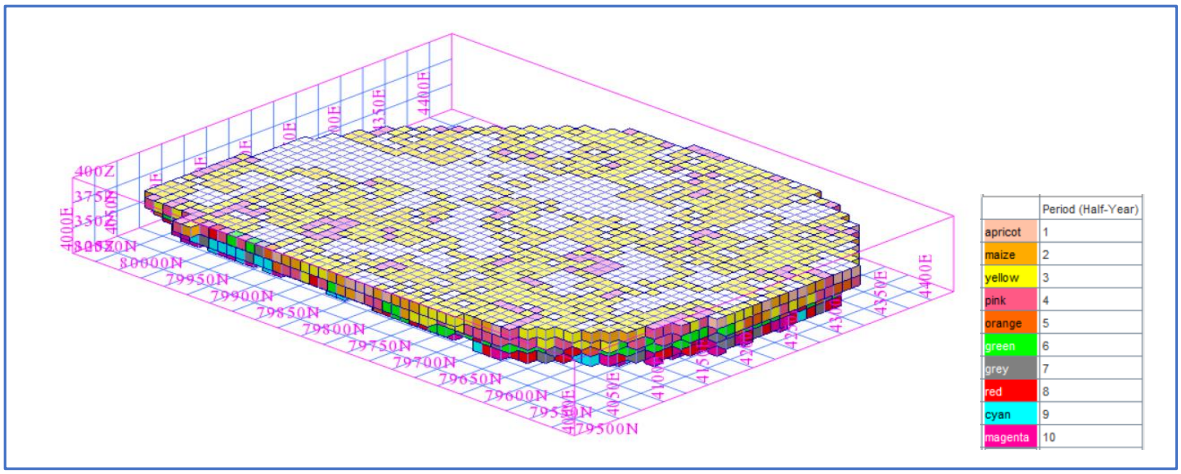


Figure 5.29: A 3D view of the MT production schedule for the Surpac block model

Likewise, in the LT production scheduling results shown in Sub-section 5.4.2, the optimal solution for each MT scheduling period was defined by the best chromosome representing a list of blocks with the highest cash flow. The list of blocks of each MT scheduling period complies with the practicality of the schedule defined by the 1:9 pattern block precedence constraint (Table 5.1) regardless of the scheduling horizon. The set of best chromosomes obtained from Periods 1 to 10 defined the MT extraction sequence shown in Figure 5.29, which does not consider the access point to the ramp as the starting point of scheduling, similar to the LT schedule. Additional constraints presented in Table 5.1 such as MT mining rate limits, processing rate limits, grade target and Mineral Reserves for the first five years, were also considered for the optimisation of MT production scheduling. Table 5.4 shows the results of MT production scheduling independently optimised.

Table 5.4: MT production scheduling results of the Surpac block model

Period [1/2 year]	No. blocks	Total tonnage [tonne]	Ore tonnage [tonne]	Waste tonnage [tonne]	Stripping ratio	Average grade [g/t]	Cash flow [$mil]
t	$TNB_{i,t}^{mt}$	$RM_{i,t}^{mt}$	$RO_{i,t}^{mt}$	$RW_{i,t}^{mt}$	$\frac{RW_{i,t}^{mt}}{RO_{i,t}^{mt}}$	$RG_{i,t}^{mt}$	$V_{n,t}^{mt}$
1	675	1,870,586	511,183	1,359,403	2.66	1.56	33.55
2	675	1,880,021	365,143	1,514,878	4.15	1.20	-3.30
3	675	1,881,727	369,310	1,512,417	4.10	1.90	21.92
4	675	1,885,705	428,942	1,456,763	3.40	2.60	6.50
5	675	1,885,471	355,872	1,529,599	4.30	2.50	17.88
6	675	1,884,301	421,047	1,463,254	3.48	1.78	9.02
7	675	1,882,360	383,443	1,498,917	3.91	2.36	23.16
8	675	1,884,692	382,788	1,501,904	3.92	3.80	6.42
9	675	1,750,923	130,735	1,620,188	12.39	1.20	-21.41
10	675	1,879,928	493,610	1,386,318	2.81	2.40	39.69
Total/Ave	**6750**	**18,685,714**	**3,842,073**	**14,843,641**	**3.86**	**1.82**	**133.43**
NPV [$mil] at a discount of 10%							**105.36**

A total of 6,750 blocks equivalent to 18.69 Mt of material containing 3.84 Mt of ore with an average grade of 1.82 g/t, and a total amount of 14.84 Mt of waste, were scheduled. Results show that 675 blocks were scheduled in each MT scheduling period. The total tonnage achieved ranging between 1.75 Mt and 1.89 Mt in each MT period, suggests that the combined MIP and GA approach complies with the minimum and maximum MT mining rates presented in Table 5.1. The total MT ore tonnage achieved in each year (comprising two half-year periods) is always less than 1 Mt, thus, under-utilising the available processing plant capacity. The lowest ore tonnage achieved in Period 9 does not comply with the MT processing rate constraint which is known as a soft constraint, because this constraint depends on the availability of ore blocks in the block model. Period 9 had 1.62 Mt of waste and 130.74 kt of ore resulting in a higher stripping ratio of 12.39.

The cash flow of $30.25 for the first year is the sum of the two half-year cash flows of $33.55 and $-3.30. The NPV over the five years comprising the MT schedule (MT schedule is only up to a maximum of five years) was estimated to be $105.36 million, which is the total discounted cash flow generated from the 10 half-year MT scheduling periods. Figure 5.30 shows the total tonnage to mine and the cash flow value from each MT scheduling period.

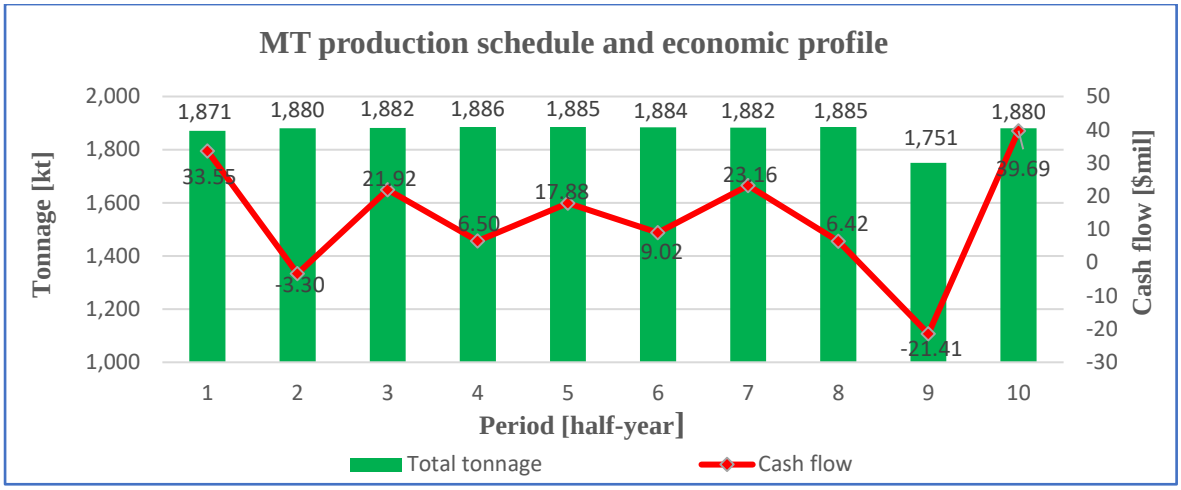


Figure 5.30: Total tonnage and the cash flow value in each period of the MT production schedule

Results show consistent total tonnage achieved throughout the MT schedule periods. The associated half-year cash flow values range between -$21.41 million and $39.69 million. Period 2 had a negative cash flow due to a lower average ore grade and relatively high stripping ratio. Period 9 also had a negative cash flow due to a lower average ore grade, lower ore tonnage achieved and high stripping ratio. Figure 5.31 shows the ore and waste tonnages, stripping ratio and the average grade for each MT scheduling period.

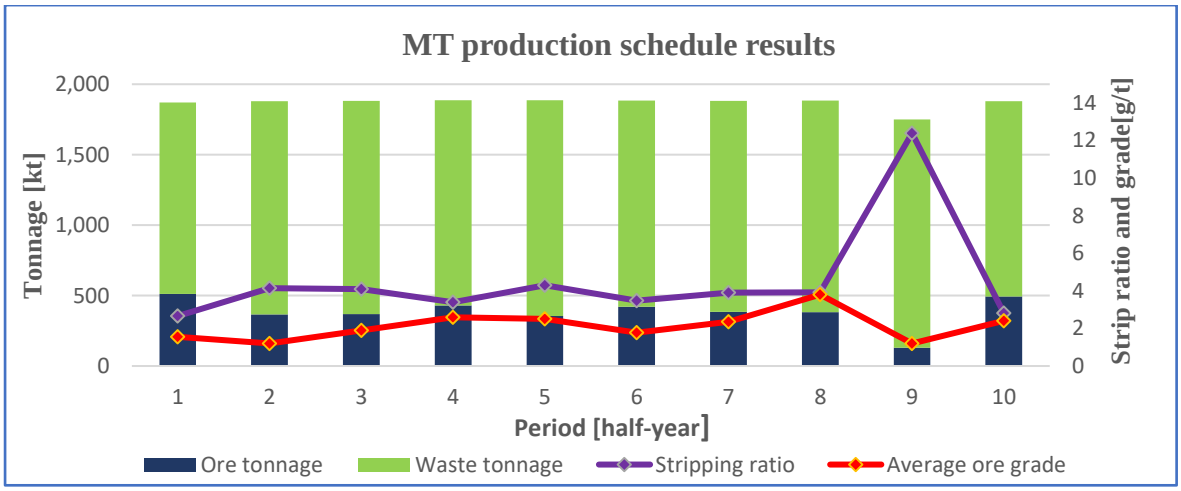


Figure 5.31: Ore and waste tonnages, stripping ratio and average grade in each period of the MT production schedule

Stripping ratio ranges between 2.66 and 12.39. The lowest stripping ratio was achieved in Period 1 suggesting that an amount of 1.36 Mt of waste must be removed to release 511.18 kt of ore of 1.56 g/t average grade. The highest cash flow which was achieved in Period 10 as shown previously in Figure 5.30 had an associated low stripping ratio of 2.81 with 1.39

Mt of waste and 493.61 kt of ore of 2.4 g/t average grade. Period 8 had the highest average grade of 3.8 g/t with 382.79 kt of ore and 1.50 Mt of waste.

MT scheduling results are inconsistent with LT scheduling results when they are independently optimised. From Table 5.3, the first five years have a total number of 6,662 blocks with associated total tonnage, ore tonnage and waste tonnage of 18.58 Mt, 3.79 Mt, and 14.78 Mt; respectively. Table 5.4 shows the total number of 6,750 blocks with associated total tonnage, ore tonnage and waste tonnage of 18.68 Mt, 3.82 Mt and 14.84 Mt; respectively. This shows scheduling misalignment which is detrimental to NPV. Table 5.5 shows the production scheduling compliance between LT and MT scheduling results calculated using Equation 3.44.

Table 5.5: Scheduling compliance of the isolated LT and MT production scheduling results

Period		**LT results achieved**			**MT results achieved**						**LT-MT Scheduling compliance ratio**		
t	$\mathcal{t}$	$RM_{i,t}^{lt}$	$RO_{i,t}^{lt}$	$V_{n,t}^{lt}$	$RM_{i,t}^{mt}$		$RO_{i,t}^{mt}$		$V_{n,t}^{mt}$		RM	RO	V
[year]	$\left[\frac{year}{2}\right]$	[tonne]	[tonne]	[$mil]	[tonne]		[tonne]		[$mil]		$\frac{RM_{i,t}^{mt}}{RM_{i,t}^{lt}}$	$\frac{RO_{i,t}^{mt}}{RO_{i,t}^{lt}}$	$\frac{V_{n,t}^{mt}}{V_{n,t}^{lt}}$
1	1	3,750,731	876,326	30.49	1,870,586	3,750,607	511,183	876,326	33.55	30.25	0.99997	1.00000	0,99213
	2				1,880,021		365,143		-3.30				
2	3	3,766,640	833,988	28.58	1,881,727	3,767,432	369,310	798,252	21.92	28.42	1.00021	0.95715	0.99440
	4				1,885,705		428,942		6.5				
3	5	3,765,573	702,890	19.44	1,885,471	3,769,772	355,872	776,919	17.88	26.9	1.00112	1.10532	1.38374
	6				1,884,301		421,047		9.02				
4	7	3,669,349	670,651	17.98	1,882,360	3,767,052	383,443	766,231	23.16	29.58	1.02663	1.14252	1.64516
	8				1,884,692		382,788		6.42				
5	9	3,629,407	712,636	23.36	1,750,923	3,630,851	130,735	624,345	-21.41	18.28	1.00040	0.87611	0.78253
	10				1,879,928		493,610		39.69				
Total/Average		**18,581,700**	**3,796,491**	**119.85**	**18,685,714**	**18,685,714**	**3,842,073**	**3,842,073**	**133.43**	**133.43**	**1.00560**	**1.01201**	**1.11331**

As stated in Section 3.4, a production scheduling compliance ratio greater than 1 indicates that more material has been scheduled in MT than initially scheduled in LT, and otherwise for a ratio less than 1. Regarding the total tonnage, Year 1 has a scheduling compliance of slightly less than 1, but Years 2, 3, 4 and 5 have compliance slightly greater than 1. This shows that the LT total tonnage is higher than the MT total tonnage in Year 1 but lower in Years 2, 3, 4 and 5. Regarding the total ore tonnage, Year 1 had a scheduling compliance equal to 1; Years 2 and 5 have compliance less than 1; and Years 3 and 4 have compliance greater than 1. This shows that the LT total ore tonnage is the same as the MT total ore tonnage in Year 1; higher in Years 2 and 5; and lower in Years 3 and 4. Although Year 1 has ore tonnage compliance of 1 between LT and MT schedules, the cash flow compliance is less than 1, suggesting lower cash flow achieved in MT than LT. Cash flow compliance is also greater than 1 in Years 3 and 4 reflecting the compliance for both total tonnage and ore tonnage. Years 2 and 5 have cash flow compliance less than 1, which is also reflected in the compliance of ore tonnage during these two years. The overall scheduling compliance results are greater than 1 with respect to total tonnage, ore tonnage and cash flow. Therefore, these results show that independent optimisation of the LT and MT production schedules leads to scheduling misalignment and inconsistency.

Another aspect of scheduling misalignment is the spatial compliance indicating the locations of mining sequence. Figure 5.32 illustrates the spatial compliance on 4250E cross-section, between LT and MT production scheduling independently optimised over the first five years.

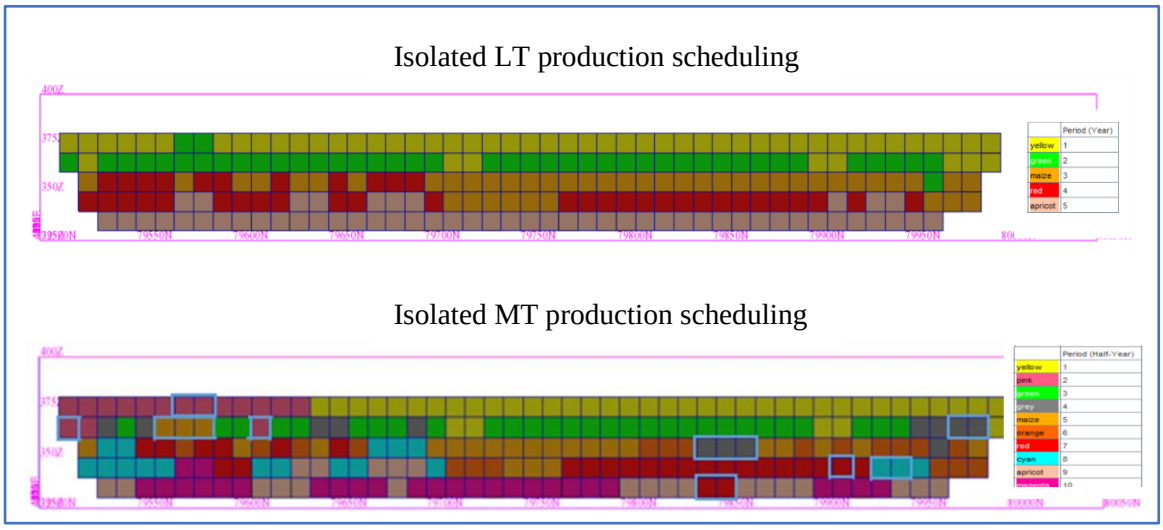


Figure 5.32: 4250E cross-sections spatial compliance between LT and MT production scheduling independently optimised

Figure 5.32 shows that some blocks (with blue outlines) were scheduled for MT but are out of LT sequence. This indicates spatial misalignment between LT and MT schedules across cross-section 4250E, when independently optimised. The next section presents the optimisation results of integrated LT and MT production scheduling that aligns these horizons.

5.6 Integrated long-term and medium-term mine production scheduling optimisation

The combined MIP and GA approach is an integrated LT and MT production scheduling optimisation approach. The integration consisted of connecting mining rate, processing rate and average grade target constraints from LT to MT production scheduling. The algorithm also ensures that the integrated schedule complies with block precedence and Mineral Reserve constraints. The integrated schedule was optimised to improve NPV as compared to the independent optimisation of the LT and MT production schedules. The combined MIP and GA approach was used to independently optimise LT and MT production scheduling and the results are presented in Sections 5.4 and 5.5, respectively. This section presents the application of the combined MIP and GA approach to the Surpac block model. The algorithm delivered a ten-year schedule integrating LT and MT horizons. The MT schedule was integrated with the LT over the first five years of the schedule since the maximum MT period is five years. Results of the integrated optimisation of the LT and MT schedule are presented in the next sub-sections.

5.6.1 Discussion of the GA parameters results for the integrated long-term and medium-term production scheduling

Application of the combined MIP and GA approach to integrate and optimise LT and MT production scheduling used PMX and heuristic mutation for the reproduction process to advance from one generation to another. The size of each generation was 40 and the average fitness value was improved generation after generation until the termination condition was met. Figure 5.33 shows the average fitness values of the first and last generations for each LT and MT integrated scheduling period.

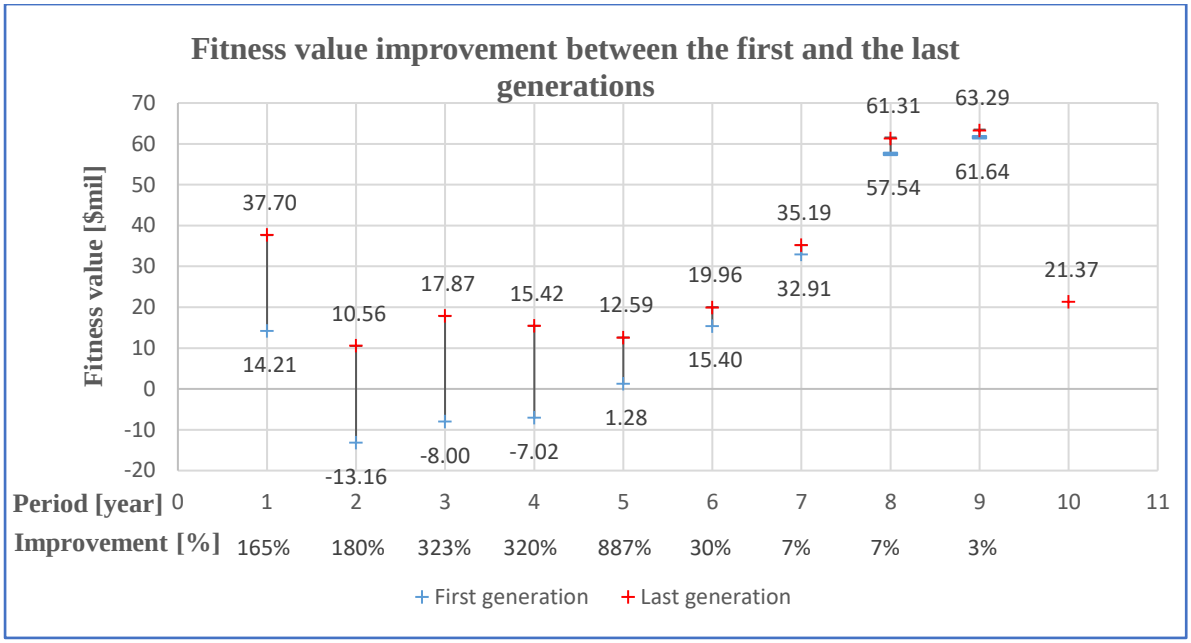


Figure 5.33: Average fitness value of the first and the last generations for each period of the integrated LT and MT production schedule

Results in Figure 5.33 indicate that there are improvements in the average fitness values from the first generation to the last generation of LT and MT integrated scheduling periods. The combined MIP and GA approach improved negative average fitness values to become positive in Periods 2, 3 and 4; from -$13.16 million to $10.56 million, from -$8.00 million to $17.87 million, from -$7.02 million to $15.42 million; respectively. The average fitness values of the first generation were improved by 165%, 180%, 323% and 320% to reach the optimal production scheduling solution in the last generation, respectively in Periods 1, 2, 3 and 4. The highest improvement was achieved in Period 5 with an improvement of about 887% from the first to the last generations. The lowest improvement was 3% from the first to the last generations in Period 9. This is because of the reduced number of blocks available in the search space at the time of optimising Period 9. For each LT and MT integrated scheduling period, the best chromosome of the last generation represents the optimal scheduling solution for that period. Figure 5.34 shows the fitness values of the best chromosome in the last generation for each LT and MT scheduling period.

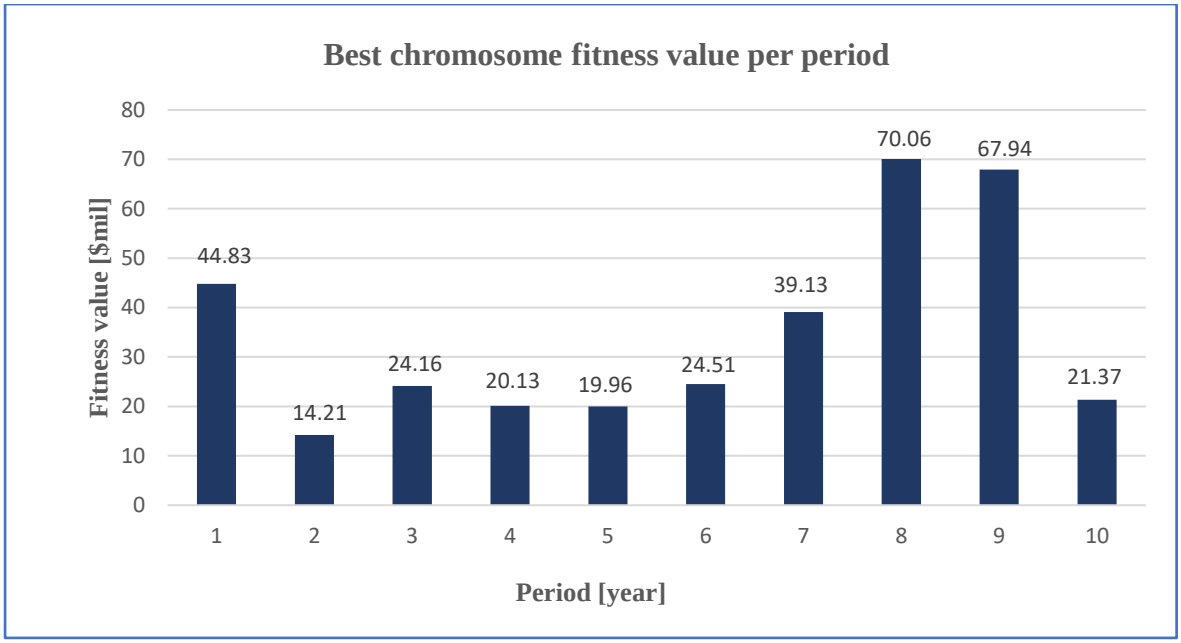


Figure 5.34: Fitness value of the best chromosome per period for the optimisation of the integrated LT and MT production schedule

Figure 5.34 shows that all the fitness values of the best chromosomes of the last generations are positive and above the average fitness values shown in Figure 5.33. The highest fitness value of $70.06 million of the best chromosome in the last generation was achieved in Period 8 due to a 7% improvement of the average fitness value from the first to the last generations. The lowest fitness value of $14.21 million was achieved in Period 2 where the first generation was improved by 180% to reach the last generation.

5.6.2 Discussion of the integrated long-term and medium-term mine production scheduling results

The integrated LT and MT scheduling results show blocks that must be mined in every LT and MT production scheduling period concurrently and the associated cash flow. It also ensures temporal and spatial compliance between LT and MT production schedules. Figure 5.35 provides a 3D visualisation of the integrated LT and MT production schedule.

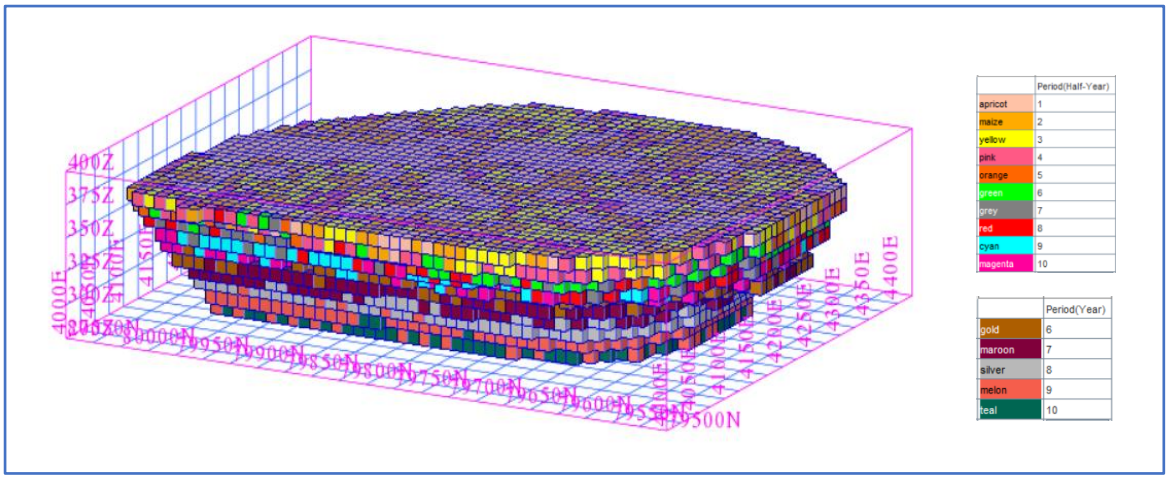


Figure 5.35: A 3D view of the integrated LT and MT production schedule for the Surpac block model

The integrated LT and MT production schedule of block extraction shown in Figure 5.35 complies with the 1:9 pattern block precedence constraint. Regardless of the scheduling horizon, a block is either mined in the same period or in a period after its block predecessors. Figure 5.35 also shows blocks to mine in the first five years of operation on a half-yearly basis of the MT production schedule. From Periods 6 to 10, the figure shows blocks to mine every year of the LT production schedule. The combined MIP and GA approach simultaneously considered LT and MT mining rate and Mineral Reserve constraints, which are hard constraints presented in Table 5.1. The approach also considered the LT and MT processing rate and average grade target constraints, which are soft constraints because they depend on the quality of available blocks for scheduling. The summary of the results obtained when applying the combined MIP and GA approach is presented in Table 5.6.

Table 5.6: Results of the combined MIP model and GA approach when applied on the Surpac block model

Period [year]	$\left[\frac{year}{2}\right]$	**No. Blocks**		**Total tonnage [tonne]**		**Ore tonnage [tonne]**		**Waste tonnage [tonne]**		**Stripping ratio**		**Average grade [g/t]**		**Cash flow [$mil]**	
t	$\mathcal{t}$	$TNB_{i,t}^{lt}$	$TNB_{i,t}^{mt}$	$RM_{i,t}^{lt}$	$RM_{i,t}^{mt}$	$RO_{i,t}^{lt}$	$RO_{i,t}^{mt}$	$RW_{i,t}^{lt}$	$RW_{i,t}^{mt}$	$\frac{RW_{i,t}^{lt}}{RO_{i,t}^{lt}}$	$\frac{RW_{i,t}^{mt}}{RO_{i,t}^{mt}}$	$RG_{i,t}^{lt}$	$RG_{i,t}^{mt}$	$V_{n,t}^{lt}$	$V_{n,t}^{mt}$
LT	MT	LT	MT	LT	MT	LT	MT	LT	MT	LT	MT	LT	MT	LT	MT
1	1	1,350	675	3,745,184	1,872,092	1,010,590	580,145	2,734,594	1,291,947	2.71	2.23	1.97	2.01	44.84	35.51
	2		675		1,873,092		430,445		1,442,647		3.35		1.93		9.33
2	3	1,350	675	3,770,314	1,891,592	699,895	271,759	3,070,419	1,619,833	4.39	5.96	1.82	1.95	14.21	0.25
	4		675		1,878,722		428,136		1,450,586		3.39		1.69		13.96
3	5	1,350	675	3,772,101	1,890,408	727,483	268,740	3,044,618	1,621,668	4.19	6.03	1.86	1.22	24.16	4.06
	6		675		1,881,693		458,743		1,422,950		3.10		2.50		20.10
4	7	1,337	675	3,734,165	1,881,582	703,499	430,069	3,030,666	1,451,513	4.31	3.38	1.92	2.28	20.13	28.26
	8		662		1,852,583		273,430		1,579,153		5.78		1.56		-8.13
5	9	1,303	676	3,637,141	1,890,650	671,114	215,294	2,966,027	1,675,356	4.42	7.78	1.85	1.75	19.96	-10.93
	10		627		1,746,491		455,820		1,290,671		2.83		1.95		30.89
6		1,261		3,518,613		679,888		2,838,725		4.18		1.95		24.51	
7		1,258		3,506,589		809,792		2,696,797		3.33		1.97		39.13	
8		1,273		3,548,085		918,469		2,629,616		2.86		2.35		70.06	
9		1,250		3,485,701		914,243		2,571,458		2.81		2.26		67.94	
10		442		1,234,213		319,896		914,317		2.86		2.28		21.36	
Total		**12,174**	**6,690**	**33,952,106**		**7,454,869**		**26,497,237**		**3.55**		**1.98**		**346.30**	**123.30**
NPV [$mil] at an annual discount rate of 10% (integrated LT and MT schedule)															**202.30**

Comparing the integrated LT and MT scheduling results in Table 5.6 with results in Table 5.3 from independent LT production scheduling, and the input parameters presented in Table 5.1, the following analysis can be made:

- The integrated LT and MT schedule complies with Mineral Reserve and mining rate limit constraints, which are considered hard constraints. The total number of blocks achieved after 10 years is equal to 12,174 blocks, complying with the total number of blocks in Table 5.1. The total tonnage for the integrated LT schedule ranges between 3.49 Mt (Year 8) and 3.77 Mt (Year 3), complying with annual minimum and maximum mining rate limits of 3.40 Mt and 3.80 Mt, respectively, presented in Table 5.1, from Years 1 to 9 (LT). The total tonnage for the integrated MT schedule ranges between 1.75 Mt (Half-Year 10) and 1.89 Mt (Half-Year 3), complying with 6-month minimum and maximum mining rate limits (1.70 Mt and 1.90 Mt, respectively) presented in Table 5.1, from Half-Years 1 to 10 (MT). Year 10 has 442 blocks and 1.23 Mt total tonnage which is the remaining material to deplete the Mineral Reserves.
- The integrated LT and MT schedule complies with the processing rate (soft constraint) from Years 2 to 9. The ore tonnage for the integrated LT schedule ranges between 671.11 kt (Year 5) and 918.47 kt (Year 8), complying with annual minimum and maximum processing rate limits (650 kt and 1 Mt, respectively) presented in Table 5.1. Year 1 exceeds the annual maximum processing rate limit by 10.59 kt, which can be stockpiled for future processing. This is also noted for the MT schedule where Half-Year 1 has an excess of 80.15 kt of ore compared to the 6-month maximum processing rate limit (500 kt).
- A processing penalty cost was incurred in Year 1 of the LT schedule where the processing rate limit was violated by 10.59 kt. The economic value of this amount of ore was estimated at $0.47 million, which was subtracted from cash flow in Year 1, and added to cash flow in Year 2. Similarly, 80.15 kt of ore excess in MT schedule Half-Year 1, was estimated at $4.91 million. This penalty cost was subtracted from cash flow in Half-Year 1, of which $4.40 million was added to cash flow in Half-Year 2 and $0.47 million was added to cash flow in Half-Year 3.
- During the first five years, integrated LT and MT schedule results show that LT objectives were met on MT horizons in all metrics, thus proving the temporal compliance between these two scheduling horizons. For example, in Period 1, the

1,350 blocks of the LT schedule are split into 675 blocks and 675 blocks for the MT schedule. The annual 3.75 Mt of total LT tonnage in Year 1 is split into 1.87 Mt and 1.87 Mt for the MT schedule. The annual 1.01 Mt of LT ore tonnage in Year 1 is split into 580.15 kt and 430.45 kt for the MT horizon. The annual average ore grade 1.97 g/t of the LT in Year 1 is split into 2.01 g/t and 1.93 g/t weighted average grades for the MT schedule. The annual $44.84 million cash flow in Year 1 is split into $35.51 million and $9.33 million for the MT schedule.

- During the first five years, the integrated LT and MT schedule results also show that LT production scheduling remained consistent with MT production scheduling and that MT scheduling fed back LT scheduling with MT updates. This is demonstrated by the fact that a combined MT total tonnage, ore tonnage and waste tonnage achieved in the same year always results in the equivalent LT total tonnage, ore tonnage and waste tonnage, respectively.
- The results show that spatial scheduling compliance is also satisfied by the integrated LT and MT schedule during the first five years, as opposed to independent optimisation of LT and MT scheduling horizons. In each of the first five years, blocks that are scheduled for the LT are the same blocks scheduled for the two MT periods of the corresponding year, based on block location and block identification.
- NPV achieved for the integrated LT and MT schedule was estimated at $202.30 million over ten years, which is 2.60% higher than NPV achieved for the independent LT schedule optimisation of $197.10 million. This is proof that integrated LT and MT scheduling improves NPV as compared to independent optimisation of LT scheduling.
- The results of the integrated schedule prove the alignment between LT and MT scheduling horizons resulting in the production scheduling compliance ratio equal to one in each period, calculated using Equation 3.44. Results of the scheduling compliance of the integrated LT and MT schedule are shown in Table 5.7.

Table 5.7: Scheduling compliance of the integrated LT and MT production schedule

Period		**LT results achieved**			**MT results achieved**						**LT to MT Scheduling compliance ratio**		
t	t	$RM_{i,t}^{lt}$	$RO_{i,t}^{lt}$	$V_{n,t}^{lt}$	$RM_{i,t}^{mt}$		$RO_{i,t}^{mt}$		$V_{n,t}^{mt}$		RM	RO	V
[year]	$\left[\frac{year}{2}\right]$	[tonne]	[tonne]	[$mil]	[tonne]		[tonne]		[$mil]		$\frac{RM_{i,t}^{mt}}{RM_{i,t}^{lt}}$	$\frac{RO_{i,t}^{mt}}{RO_{i,t}^{lt}}$	$\frac{V_{n,t}^{mt}}{V_{n,t}^{lt}}$
1	1	3,745,184	1,010,590	44.84	1,872,092	3,745,184	580,145	1,010,590	35.51	44.84	1	1	1
	2				1,873,092		430,445		9.33				
2	3	3,770,314	699,895	14.21	1,891,592	3,770,314	271,759	699,895	0.25	14.21	1	1	1
	4				1,878,722		428,136		13.96				
3	5	3,772,101	727,483	24.16	1,890,408	3,772,101	268,740	727,483	4.06	24.16	1	1	1
	6				1,881,693		458,743		20.10				
4	7	3,734,165	703,499	20.13	1,881,582	3,734,165	430,069	703,499	28.26	20.13	1	1	1
	8				1,852,583		273,430		-8.13				
5	9	3,637,141	671,114	19.96	1,890,650	3,637,141	215,294	671,114	-10.93	19.96	1	1	1
	10				1,746,491		455,820		30.89				
Total/Average		**18,658,905**	**3,812,581**	**123.30**	**18,658,905**	**18,658,905**	**3,812,581**	**3,812,581**	**123.30**	**123.30**	**1**	**1**	**1**

Results of the integrated LT and MT scheduling compliance presented in Table 5.7 show a ratio of 1 in all metrics including total tonnage, ore tonnage and cash flow. Therefore, the integrated LT and MT schedule prevents the scheduling misalignment over the first five years, that occurred when independently optimised LT and MT schedules. Figure 5.36 illustrates the spatial compliance on 4250E cross-section, between LT and MT production scheduling when they are integrated over the first five years.

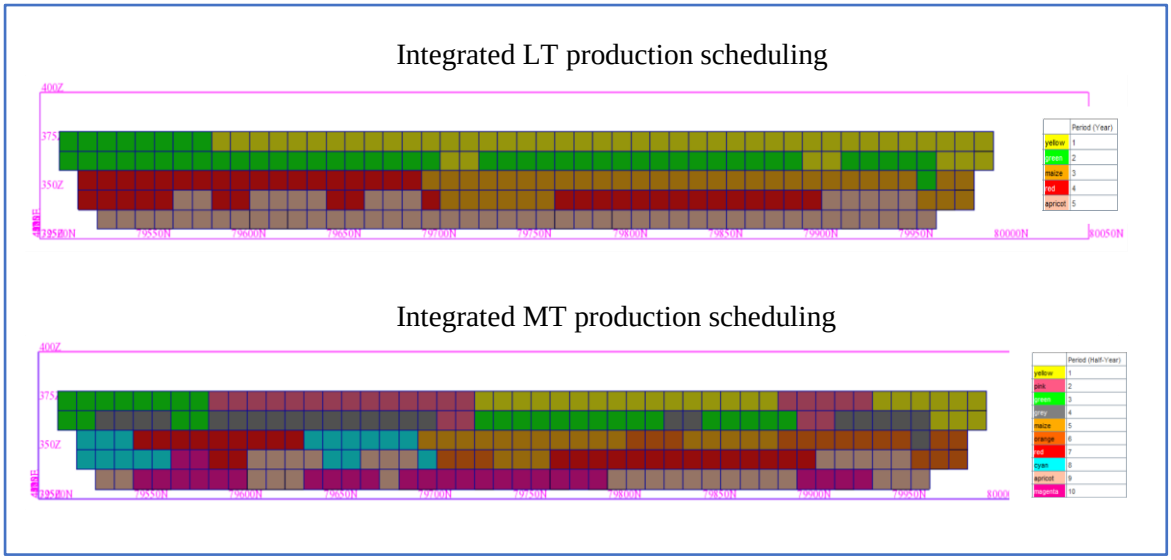


Figure 5.36: 4250E cross-sections spatial compliance between LT and MT production scheduling integrally optimised

Figure 5.36 shows that spatial alignment is ensured between MT and LT schedules across cross-section 4250E, because blocks that were scheduled in the LT sequence were scheduled in their corresponding MT periods. This indicates that the integration of LT and MT production scheduling prevents spatial misalignment as opposed to when these schedules were independently optimised. Figure 5.37 shows the total tonnage achieved for the integrated LT and MT production schedule.

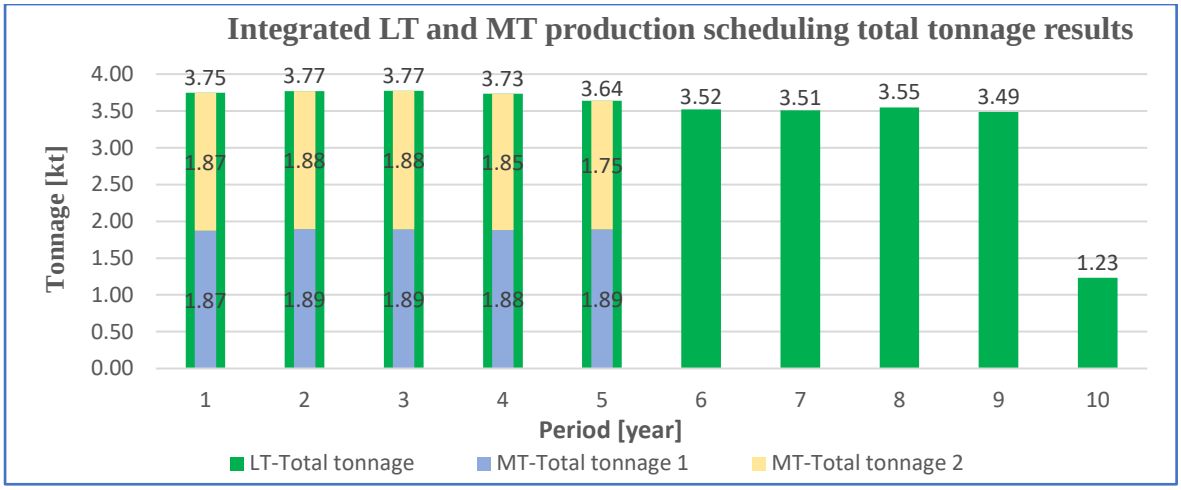


Figure 5.37: Total tonnage achieved for the integrated LT and MT production schedule

Figure 5.38 shows the ore tonnage achieved for the integrated LT and MT production schedule.

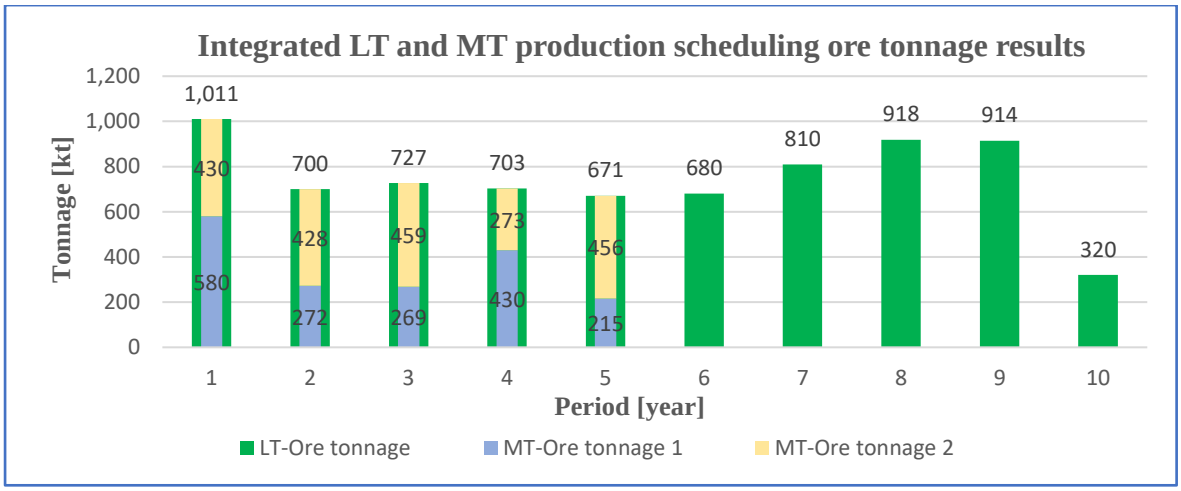


Figure 5.38: Ore tonnage achieved for the integrated LT and MT production schedule

Results in Figure 5.38 show that although the yearly LT ore tonnage is not evenly distributed for the half-yearly MT ore tonnages, total ore tonnages of the MT are equal to the LT ore tonnage in the same year, over the first five years. For instance, the total MT ore tonnage of 580 kt and 430 kt equals 1.01 Mt of LT schedule in Year 1. The annual LT ore tonnage is always achieved at MT of the half-yearly schedule. Therefore, the half-yearly MT ore tonnages are spatially and temporally aligned with yearly LT ore tonnages.

5.6.3 Sensitivity analysis

Sensitivity analysis was performed to show how sensitive the NPV and LOM obtained are to variations in the production scheduling inputs parameters. The combined MIP and GA

approach dynamically updates the LT and MT scheduling horizons when changes occur in either of the horizons. Parameters such as mining rate, processing rate, target grade and market price were investigated. The analysis is critical because it allows an assessment of the behaviour of NPV and LOM of the integrated LT and MT production schedule as those parameters change. Figure 5.39 shows the sensitivity results of NPV when, the mining rate (inclusive of both ore and waste) and processing rate (of ore only), market price and grade target change by a margin of ten per cent.

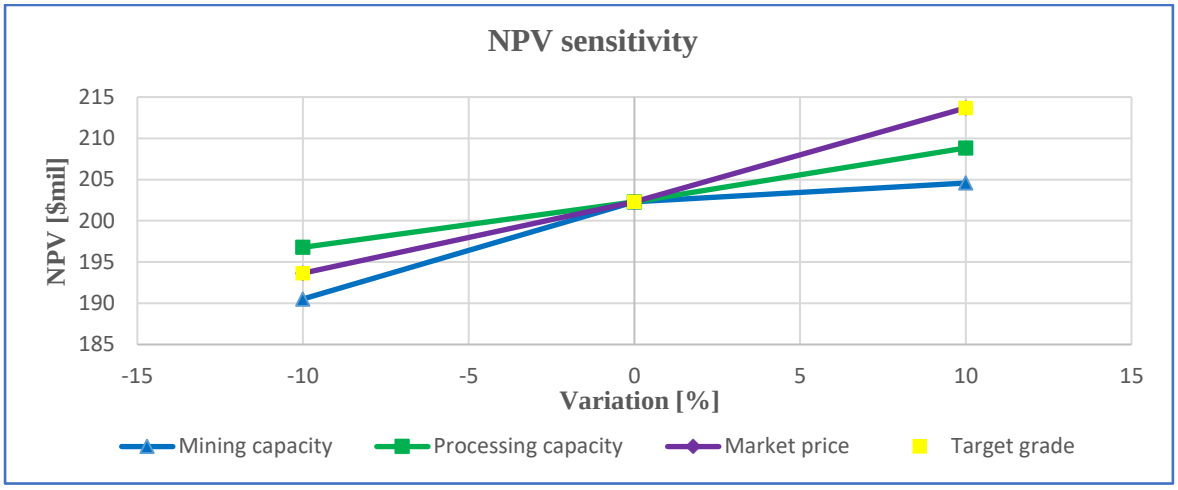


Figure 5.39: NPV sensitivity results

The analysis outlines the strengths and weaknesses of different scheduling scenarios when changes occur to one or more parameters. Figure 5.39 shows that variations in market price affect not only NPV but also the target grade of the integrated LT and MT production schedule. NPV is most sensitive to market price and target grade, and least sensitive to mining rate which includes mining of both ore and waste. The sensitivity of the LOM when mining rate, processing rate, market price and grade target change by a margin of ten per cent, is presented in Figure 5.40.

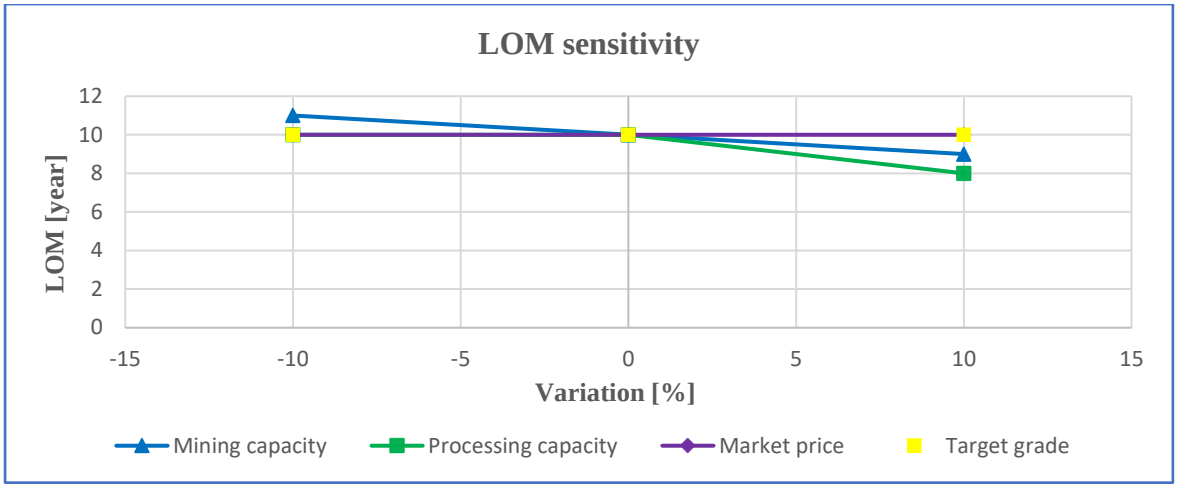


Figure 5.40: LOM sensitivity results

The sensitivity analysis in Figure 5.40 shows that LOM is sensitive to variations in mining rate, and processing rate. LOM is most sensitive to mining rate confirming intuitive reasoning that the more material mined the shorter the LOM. LOM is least sensitive to market price and target grade.

5.7 Comparison of the results between the combined mixed integer programming model and GA approach to existing techniques applied to the Newman and Zuck Small block models

MineLib (2012) reported the best-known feasible LP relaxation solutions obtained when the OPMPS problem was solved using a TopoSort heuristic algorithm applied to the Newman and Zuck Small block models. The combined MIP and GA approach presented in this thesis was also applied to the two block models, and the same input parameters from MineLib were used to enable meaningful comparisons to be made. Table 5.8 presents the results of the integrated LT and MT schedule using the combined MIP and GA approach applied to the Newman block model.

Table 5.8: Results of the combined MIP model and GA approach when applied on the Newman block model

Period		**No. blocks**		**Total tonnage [tonne]**		**Ore tonnage [tonne]**		**Stripping ratio**		**Average grade [g/t]**		**Cash flow [$mil]**	
[year]	$\left[\frac{year}{2}\right]$												
t	t	$TNB_{i,t}^{lt}$	$TNB_{i,t}^{mt}$	$RM_{i,t}^{lt}$	$RM_{i,t}^{mt}$	$RO_{i,t}^{lt}$	$RO_{i,t}^{mt}$	$\frac{RW_{i,t}^{lt}}{RO_{i,t}^{lt}}$	$\frac{RW_{i,t}^{mt}}{RO_{i,t}^{mt}}$	$RG_{i,t}^{lt}$	$RG_{i,t}^{mt}$	$V_{n,t}^{lt}$	$V_{n,t}^{mt}$
LT	MT	LT	MT	LT	MT	LT	MT	LT	MT	LT	MT	LT	MT
1	1	350	175	1,732,580	800,401	757,511	367,940	1.29	1.18	0.93	0.87	7.04	3.23
	2		175		932,179		389,571		1.39		0.99		3.81
2	3	350	175	1,916,919	981,403	828,888	363,381	1.31	1.7	1.09	1.32	8.45	4.64
	4		175		935,516		465,507		1.01		0.91		3.81
3	5	350	175	1,917,660	993,504	1,253,390	462,048	0.53	1.15	1.16	1.12	10.48	4.07
	6		175		924,156		791,342		0.17		1.18		6.41
4	7	10	10	52,217	52,217	48,851	48,851	0.07	0.07	1.51	1.51	0.10	0.10
Total		**1,060**	**1,060**	**5,619,376**	**5,619,376**	**2,888,640**	**2,888,640**	**0.95**	**0.95**	**1.08**	**1.08**	**26.07**	**26.07**
NPV [$mil] at an annual discount rate of 8% (integrated LT and MT schedule)													**22.51**

The combined MIP and GA approach delivered a four-year integrated LT and MT production schedule when applied to the Newman block model. A total of 1,060 blocks equivalent to 5.62 Mt of material were scheduled. This is the same number of blocks contained in the Newman block model presented in Table 5.1. Therefore, the integrated LT and MT production schedule complies with the Mineral Reserve constraint. Excluding Period 4, the total tonnage achieved ranged between 1.73 Mt and 1.92 Mt in each LT scheduling period. It achieved between 800.40 kt and 993.50 kt in each MT scheduling period. This shows that the combined MIP and GA approach complies with the minimum and maximum mining rates in Years 1, 2 and 3. The integrated LT and MT schedule ensures scheduling alignment between the LT and MT schedules of the Newman block model.

After applying the combined MIP and GA approach to the Newman block model, the approach was then applied to the Zuck Small block model. The results of the integrated LT and MT production scheduling using the combined MIP and GA approach applied to the Zuck Small block model, are presented in Table 5.9.

Table 5.9: Results of the combined MIP model and GA approach when applied on the Zuck Small block model

Period		**No. of blocks**		**Total tonnage [tonne]**		**Ore tonnage [tonne]**		**Stripping ratio**		**Cash flow [$mil]**	
[year]	$\left[\frac{year}{2}\right]$										
t	t	$TNB_{i,t}^{lt}$	$TNB_{i,t}^{mt}$	$RM_{i,t}^{lt}$	$RM_{i,t}^{mt}$	$RO_{i,t}^{lt}$	$RO_{i,t}^{mt}$	$\frac{RW_{i,t}^{lt}}{RO_{i,t}^{lt}}$	$\frac{RW_{i,t}^{mt}}{RO_{i,t}^{mt}}$	$V_{n,t}^{lt}$	$V_{n,t}^{mt}$
LT	MT	LT	MT	LT	MT	LT	MT	LT	MT	LT	MT
1	1	900	451	46,122,108	22,300,189	33,692,117	18,953,334	0.37	0.18	131.70	73.44
	2		449		23,821,918		14,738,783		0.62		58.25
2	3	900	541	52,258,203	28,855,443	44,680,654	26,492,393	0.17	0.09	272.68	149.98
	4		449		23,402,760		18,188,260		0.29		122.71
3	5	900	451	58,783,904	29,570,463	37,360,532	21,000,450	0.57	0.41	249.35	147.19
	6		449		29,213,441		16,360,081		0.79		102.17
4	7	900	451	55,974,463	28,504,450	31,682,541	16,862,130	0.77	0.69	179.17	101.04
	8		449		27,470,012		14,820,411		0.85		78.13
5	9	900	451	53,216,946	27,893,920	26,778,241	16,843,921	0.99	0.66	122.26	85.73
	10		449		25,323,025		9,934,320		1.55		36.53
6		900		53,575,187		21,839,900		1.45		113.44	
7		900		56,373,520		23,424,920		1.41		98.16	
8		900		54,960,221		21,571,131		1.55		85.88	
9		900		58,324,021		17,977,081		2.24		46.83	
10		860		56,830,721		23,481,281		1.42		66.84	
11		440		29,710,411		17,742,430		0.67		56.42	
Total		**9,400**	**4,590**	**576,129,705**	**266,355,621**	**300,230,828**	**174,194,083**	**0.92**	**0.53**	**1,422.72**	**955.17**
NPV [$mil] at an annual discount rate of 10% (integrated LT and MT schedule)											**967.09**

The combined MIP and GA approach delivered an eleven-year LT schedule from which the MT schedule was integrated over the first five years when applied to the Zuck Small block model. The MT production schedule was created over the first 5 years, and the LT production schedule covered the LOM duration. A total of 9,400 blocks equivalent to 576.13 Mt of material were scheduled. This is the same number of blocks contained in the Zuck Small block model presented in Table 5.1. Therefore, the application of the combined MIP and GA approach on this block model complies with the Mineral Reserve constraint. Excluding the 11th year, the total tonnage achieved ranged between 46.12 Mt and 58.32 Mt in each LT scheduling period. It achieved between 22.30 Mt and 29.57 Mt in each MT scheduling period. This shows that the combined MIP and GA approach complies with the LT and MT minimum and maximum mining rates. Excluding the 11th year, the total ore tonnage achieved ranged between 17.98 Mt and 44.68 Mt in each LT scheduling period. It achieved between 9.93 Mt and 26.49 Mt in each MT scheduling period. The integrated LT and MT schedule ensures that there is scheduling alignment between the LT and MT schedules of the Zuck Small block model.

In addition to providing mining data sets, MineLib also provides the optimal solutions for the Newman and Zuck Small block models. To validate the combined MIP and GA approach, results of the approach were compared with results of the TopoSort algorithm. Table 5.10 shows the comparison.

Table 5.10: Comparison of the combined MIP model and GA approach and TopoSort algorithm results

Block model	Technique	NPV [$mil]	Gap [%]	Running time [second]
Newman	TopoSort algorithm	23.66	-	-
	Genetic algorithm	22.51	-4.86	649
Zuck Small	TopoSort algorithm	872.37	-	-
	Genetic algorithm	967.09	+10.86	8,058

Firstly, when applied to the Newman block model, the TopoSort algorithm generated an NPV of $23.66 million, while the combined MIP and GA approach an NPV of $22.51 million, which is 4.86% lower. The combined MIP model and GA approach had a lower NPV compared to the TopoSort algorithm because GA can generate a different outcome each time it is run on the block model, due to its stochastic solution process. Another run of

the combined MIP model and GA approach on the same block model with the same inputs may generate a higher NPV (Muke *et al*., 2025a). However, it should be noted that the combined MIP and GA approach provided an integrated and reconciled LT and MT production schedule. Secondly, when applied to the Zuck Small block model, the combined MIP and GA approach generated a higher NPV of $967.09 million which is 10.86% higher than the NPV of $872.37 million reported on MineLib obtained when the OPMPS was solved using the TopoSort algorithm. This demonstrates that the combined MIP model and GA approach can provide comparable results with existing open-pit production scheduling optimisation techniques. Muke *et al.* (2025b) tested the scalability of the combined MIP and GA approach by applying it to a larger dataset, the KD block model from MineLib, and generated comparable results that further validated the approach.

5.8 Summary of Chapter 5

The implementation of the developed model for integrating LT and MT production scheduling has proven to improve NPV as opposed to when the scheduling horizons are optimised independently. Independent optimisation of LT and MT production scheduling was done on the Surpac block model containing 12,174 blocks. The combined MIP and GA approach delivered a ten-year LT production schedule independently optimised and an NPV of $197.17 million. The MT production schedule had NPV of $105.36 million over the first five years horizon. However, the integrated LT and MT production schedule over ten years had NPV of $202.30 million showing an improvement of 2.60% from the independent LT production scheduling optimisation.

Validation of the combined MIP and GA approach was done using Newman and Zuck Small block models which were freely downloaded from MineLib. The Newman block model contains 1,060 blocks while the Zuck Small block model contains 9,400 blocks. Results showed that the combined MIP and GA approach delivered NPVs of $22.51 million on the Newman block model over four years LOM and $967.09 million on the Zuck Small block model over 11 years LOM. Therefore, the combined MIP and GA approach outperformed the TopoSort algorithm when applied to the Zuck Small block model, as the combined MIP and GA approach's NPV is 10.86% higher than the TopoSort NPV of $872.37 million. The combined MIP and GA approach underperformed against the TopoSort algorithm when applied to the Newman block model, as the combined MIP and GA approach's NPV is 4.86% lower than the TopoSort NPV of $23.66 million.

Concerning constraint compliance, the combined MIP and GA approach complied with hard constraints namely block precedence relationship, mining rate and Mineral Reserve constraints at LT and MT production schedules. The combined MIP and GA approach also complied with soft constraints including processing rate and average ore grade as long as they did not compromise hard constraints. The combined MIP and GA approach delivered an integrated schedule that complies with the spatial and temporal alignments between LT and MT schedules. Scheduling alignment is important to ensure that LT objectives are achieved at MT horizons and that scheduling horizons remain consistent with each other. The sensitivity analysis conducted also showed the behaviour of the NPV and LOM to a ±10% variation in mining rate, processing rate, ore grade and market price. NPV was positively sensitive to a 10% increase in mining rate, processing rate and market price, while negatively sensitive, otherwise. LOM was shortened when the mining rate and processing rate increased by +10% and was extended when the mining rate decreased by -10%.

6. CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

This thesis aimed to develop a dynamic OPMPS system that integrates LT and MT mine production scheduling horizons using a combined MIP and GA approach. The thesis provided a new optimisation approach for integrating LT and MT scheduling horizons as opposed to when these horizons are optimised independently. The integrated solution approach was presented as a combined MIP model and GA approach to incorporate uncertainty and scheduling constraints such as block precedence relationships, mining rate, processing rate, grade target and Mineral Reserve constraints. The combined MIP and GA approach was used to dynamically optimise the integrated scheduling system with an objective function that maximises NPV. The approach was coded in a Python programming environment and implemented to a case study block model downloaded from Geovia Surpac® software. The results show that integrating LT and MT production scheduling improves NPV by 2.60% as opposed to independently optimising the two scheduling horizons. The thesis has therefore answered the research question and has demonstrated that the combined MIP model and GA approach can improve temporal and spatial alignment between LT and MT production schedules, while attempting to maximise NPV under operational uncertainty.

The solution obtained was compared with existing optimal solutions to validate the efficiency and robustness of the proposed MIP model and GA approach. The integrated MIP model and GA approach was applied to the Newman and Zuck Small block model instances downloaded from the MineLib database. The combined MIP and GA approach outperformed the existing practical solution obtained by solving the OPMPS problem using the TopoSort algorithm by 10.86% on the Zuck Small block model but underperformed by 4.86% when applied to the Newman block model. This demonstrates that the proposed MIP model and GA approach can be applied to different block models and different geological conditions. Moreover, the combined MIP and GA approach guarantees scheduling alignment between LT and MT production scheduling and ensures that LT objectives are achieved at MT horizons.

6.2 Contribution to knowledge

The thesis provides a new solution approach with a significant value contribution to fundamentally improve open-pit mine production scheduling. The thesis contributes to advancing knowledge and practice in the following areas:

1. A new MIP model for integrating LT and MT production scheduling horizons was developed. The model optimises an objective function that maximises NPV under multiple constraints. The development of an MIP model employs scheduling penalty costs for any deviations from scheduled production targets and ore extraction sequences. Penalties provide a mechanism for evaluating and addressing potential disruptions or discrepancies that might arise during mining operations.
2. Dynamic connecting links were created between LT and MT production scheduling horizons that enable an update of the scheduling system to capture changes whenever they occur. Connecting links consist of expressing LT objectives into MT objectives and defining feedback loops using mining rate, processing rate, grade target and Mineral Reserve constraints to ensure sustainable utilisation of Mineral Reserves.
3. A new combined MIP and GA approach was developed to stochastically optimise the integrated LT and MT scheduling horizons. The combined MIP and GA approach demonstrates the capability of metaheuristic techniques to solve an MIP mine production scheduling model.
4. The thesis has demonstrated that integrating production horizons can result in higher NPV, and improved spatial and temporal compliance.

The integration of LT and MT scheduling horizons has improved the reconciliation process of the mine production scheduling system. Reconciliation enables measurement of spatial and temporal scheduling compliance. Spatial schedule compliance identifies the location of mining areas or polygons within the mine schedule system and implements the sequence in which these mining areas should be mined. Temporal scheduling compliance ensures enforcing the scheduling time across all the scheduling horizons.

6.3 Recommendations

The successful application of the combined MIP and GA approach presented in this thesis enhances the use of metaheuristic techniques for complex and large size mining problems. The MIP model enables a dynamic LT to MT integrated approach while the GA

methodology accounts for operational uncertainty. The recommendations grounded in the findings of this research aim to address the key issues identified, and these include:

- Mine planners should consider dynamically integrating mine production scheduling horizons to enhance their planning processes. Disintegrated planning leads to a reduced economic value, lower Mineral Reserves recovery and lower levels of spatial and temporal alignment between production scheduling horizons.
- The integration of mine production scheduling horizons must follow a sequential and structured approach to allow a smooth transition and connection from one horizon to another.

6.4 Suggestions for future research work

The use of crossover and mutation in the GA methodology, enables the algorithm to exploit and explore a large area of potential solution and escape from being trapped in a local optimal solution. Although the findings of this thesis have demonstrated successful application of the combined MIP model and GA approach, the approach may require multiple runs before a satisfactory solution is reached. This is due to the random generation of the initial population and stochastic crossover, and mutation used in the optimisation procedure from which the optimal solution is subjected to.

Future research should explore the integration solution approach and consider other factors such as stockpiling and blending that were not considered in this study but are critical for a deeper understanding of production scheduling. This should be aligned with work that includes the following:

- Developing MIP models for integrating the MT and ST production scheduling system to incorporate operational scheduling constraints.
- Developing another MIP model to fully integrate LT, MT and ST scheduling horizons.
- Applying other metaheuristic techniques using machine learning tools such as a combination of GA and RL, for optimising integrated mine production scheduling system for open-pit mining operations.
- Shortening of the MT scheduling period from 6 months to 1 month to understand the impact on the integrated scheduling results.
- Investigating the fine-tuning of GA parameters to ensure that the application of the combined MIP model and GA approach can consistently generate higher NPVs. The

fine-tuning could include using different chromosome lengths and different encoding of the chromosomes.

7. REFERENCES

Aallaoui, S., Azzamouri, A. and Tchernev, N., 2022. Mixed integer linear programming model for open pit mine scheduling. *IFAC-PapersOnLine,* Vol. 55, No.10, pp. 2276-2281.

Ahmadi, M. R. and Shahabi, R. S., 2018. Cutoff grade optimization in open pit mines using genetic algorithm. *Resources Policy,* Vol. 55, pp. 184-191.

Akaike, A. and Dagdelen, K., 1999. Strategic production scheduling method for an open pit mine. *Proceedings of the 28th Application of Computers and Operations Research in the Mineral Industry,* pp. 729-738.

Akbari, A., Osanloo, M. and Shirazi, M., 2008. Ultimate pit limit determination through minimizing risk costs associated with price uncertainty. *Economy, Raw Material and Mineral,* Vol. 24, No. 4, pp. 157-170.

Alipour, A., Khodaiari, A. A., Jafari, A. and Moghaddam, R. T., 2017. A genetic algorithm approach for open-pit mine production scheduling. *International Journal of Mining and Geo-Engineering,* Vol. 1, No. 51, pp. 47-52.

Alipour, A., Khodaiari, A. A., Jafari, A. and Moghaddam, R. T., 2022. An integrated approach to open-pit mines production scheduling. *Resources Policy,* Vol. 75, pp. 102459.

Andreasson, N., Evgrafov, A. and Patriksson, M., 2005. *An Introduction to Optimization: Foundations and Fundamental Algorithms.* Sweden: Chalmers University of Technology.

Askari-Nasab, H. and Awuah-Offei, K., 2009. Open pit optimisation using discounted economic block values. *Mining Technology,* Vol. 118, No. 1, pp. 1-12.

Azzamouri, A., Fenies, P., Fontane, F. and Giard, V., 2019. Modelling the tactical decisions for open-pit mines. https://hal.science/hal-02158146v1/file/cahier_380.pdf, [Accessed 2 July 2024].

Bandaru, S. and Deb, K., 2016. Metaheuristic Techniques. In: *Decision Sciences.* Boca Raton: CRC Press, pp. 693-750.

Behrang, K., Hooman, A. N. and Clayton, D., 2014. A linear programming model for long-term mine planning in the presence of grade uncertainty and a stockpile. *International Journal of Mining Science and Technology,* Vol. 24, pp. 451-459.

Bertayeva, K., Panaedova, G., Natocheeva, N., Kulagovskaya, T., Belyanchikova, T., Rudoy, D. and Murgul, V., 2019. Industry 4.0 in the mining industry: global trends and innovative development. *E3S Web of Conferences*, Vol. 135, No. 04026, pp. 1-7.

Bester, M., Russell, T., Van Heerden, J. and Carey, R., 2016. Reconciliation of the mining value chain – mine to design as a critical enabler for optimal and safe extraction of the mineral reserve. *The Southern African Institute of Mining and Metallurgy,* Vol. 116, No. 5, pp. 407-411.

Bierlaire, M., 2018. *Optimization: Principles and Algorithms.* 2nd ed. Lausanne, Switzerland: EPFL Press.

Blom, M., Pearce, A. R. and Stuckey, P. J., 2017. Short-term scheduling of an open-pit mine with multiple objectives. *Engineering Optimization,* Vol. 49, No 5, pp. 777-795.

Blum, C. and Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *Association for Computing Machinery (ACM Computing Surveys),* Vol. 35, No. 3, pp. 268-308.

Boland, N., Dumitrescu, I. and Froyland, G., 2008. A multistage stochastic programming approach to open pit mine production scheduling with uncertain Geology. pp. https://www.researchgate.net/publication/228947110, [Accessed 02 May 2023].

Caccetta, L. and Hill, S. P., 2003. An application of branch and cut to open pit mine scheduling. *Journal of Global Optimization,* Vol. 27, pp. 349-365.

Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E. and Rubio, E., 2012. A new algorithm for the open-pit mine scheduling problem. *Operations Research,* Vol. 60, No. 3, pp. 517-528.

Consuegra, F. A. and Dimitrakopoulos, R., 2009. Stochastic mine design optimisation based on simulated annealing: pit limits, production schedules, multiple orebody scenarios and sensitivity analysis. *Mining Technology,* Vol. 118, No. 2, pp. 79-90.

Cutler, J. and Dimitrakopoulos, R., 2024. Joint stochastic optimisation of open-pit mine production scheduling with ramp design. *International Journal of Mining, Reclamation and Environment,* Vol. 38, No. 6, pp. 480-495.

Dagdelen, K. and Johnson, T., 1986. Optimum open pit mine production scheduling by Lagrangian parameterization. *19th APCOM Symposium of the Society of Mining Engineers (AIME)*, pp. 127-141.

Davis, L., 1987. *Genetic Algorithms and Simulated Annealing.* 1st ed. Michigan: Pitman.

Denby, B. and Schofield, D., 1994. Open pit design and scheduling by use of genetic algorithms. *Transactions of the Institution of Mining and Metallurgy,* Vol. 103, pp. A21-A26.

Erarslan, K. and Celebi, N., 2001. A simulative model for optimum open pit design. *Canadian Institute of Mining, Metallurgy and Petroleum (CIM Bulletin),* Vol. 94, pp. 59-68.

Espinoza, D., Goycoolea, M., Moreno, E. and Newman, A., 2012. MineLib: A library of open pit mining problems. *Annals of Operations Research,* 23 July, Vol. 260, No. 1, pp. 91-114.

Fallara, F., Legault, M. and Rabeau, O., 2006. 3-D integrated geological modelling in the Abitibi Subprovince (Quebec, Canada): techniques and applications. *Exploration and Mining Geology,* Vol. 15, No. 1, pp. 27-43.

Fathollahzadeh, K., Mardaneh, E., Cigla, M. and Asad, M. W. A., 2021. A mathematical model for open pit mine production scheduling with grade engineering and stockpiling. *International Journal of Mining Science and Technology,* Vol. 31, No. 4, pp. 717-728.

Ferland, J. A., Amaya, J. and Djuimo, M. S., 2007. Application of a particle swarm algorithm to the capacitated open pit mining problem. In: S. C. Mukhopadhyay and G. S. Gupta, eds. *Autonomous Robots and Agents*, pp. 127-133.

Fricke, C., 2006. Applications of integer programming in open pit mining. *PhD Thesis,* Melbourne: The University of Melbourne.

Gandomi, A. H., Yang, X. S., Talatahari, S. and Alavi, A. H., 2013. Metaheuristic algorithms in modelling and optimization. In: *Metaheuristic Applications in Structures and Infrastructures*, pp. 1-24.

Gen, M. and Cheng, R., 2000. *Genetic Algorithms and Engineering Optimisation.* 1st ed. New York: John Wiley & Sons, Inc.

Gen, M., Cheng, R. and Lin, L., 2008. *Network Models and Optimization: multiobjective genetic algorithm approach.* London: Springer London.

Geovia, 2024. *Dassault Systèmes, Geovia Surpac 2024 Version. https://software.3ds.com/?ticket=ST-51744690-7VSqMyTYzUZHQb00Fj07-cas,* [Accessed 2 February 2024].

Gershon, M., 1983. Optimal mine production scheduling: evaluation of large scale mathematical programming approaches. *International Journal of Mining Engineering,* Vol. 1, pp. 315-329.

Gholamnejad, J., Azimi, A., Lotfian, R., Kasmaeeyazdi, S. and Tinti, F., 2020a. The application of a stockpile stochastic model into long-term open pit mine production scheduling to improve the feed grade for the processing plant. *Rudarsko-geološko-naftni zbornik,* Vol. 35, No. 4, pp. 115-129.

Gholamnejad, J., Lotfian, R. and Kasmaeeyazdi, S., 2020b. A practical, long-term production scheduling model in open pit mines using integer linear programming. *The Journal of the Southern African Institute of Mining and Metallurgy,* Vol. 120, No. 12, pp. 665-670.

Ghorbani, T., Nwaila, G. T., Zhang, S.E., Bourdeau, J. E., Canovas, M., Arzua, J. and Nikadat, N., 2023. Moving towards deep underground mineral resources: Drivers, challenges and potential solutions. *Resources Policy,* Vol. 80, p. 103222.

Gleixner, A. M., 2008. Solving large-scale open-pit mining production scheduling problems by integer programming, *PhD Thesis,* Germany: Technical University of Berlin.

Goldberg, D., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* 1st ed. United States of America: Addison-Wesley Publishing Company, Inc.

Goldberg, D. and Lingle, R., 1985. Alleles, loci, and the travelling salesman problem. *New York, Psychology Press*, pp. 154-159.

Guenin, B., Könemann, J. and Tunçel, L., 2014. *A Gentle Introduction to Optimization.* 1st ed. Cambridge: Cambridge University Press.

Hartman, H. L. and Mutmansky, J. M., 2002. *Introductory Mining Engineering.* 2nd ed. New Delhi: John Wiley and Sons Inc.

Hoeve, W. J. V., 2005. Operations research techniques in constraint programming, *PhD Thesis,* Amsterdam: PrintPartners Ipskamp.

Holland, J. H., 1975. *Adaptation In Natural And Artificial Systems.* London: The MIT Press.

Hustrulid, W., Kuchta, M. and Martin, R., 2013. *Open Pit Mine Planning and Design.* 3rd ed. London: CRC Press.

Jélvez, E., Morales, N. and Nancel-Penard, P., 2019. Open-pit mine production scheduling: improvements to MineLib Library problems. *Cham, Springer International Publishing*.

Jélvez, E., Ortiz, J., Varela, N. M., Askari-Nasab, H., and Nelis, G., 2023. A multi-stage methodology for long-term open-pit mine production planning under ore grade uncertainty. *Mathematics,* Vol. 11, No. 18, p. 3907.

Johnson, T. B., 1968. *Optimum Open Pit Production Scheduling.* 1st ed. University of California, Berkeley: Defense Technical Information Center.

Jose, C., Pol, A., Daniel, G., Daniel, R., and Angel, J., 2014. Rich vehicle routing problem: survey. *Association for Computing Machinery (ACM Computing Surveys),* Vol. 47, No. 2, pp. 1-28.

Joshi, D., Gholami, H., Mohapatra, H., Ali, A., Streimikiene, D., Satpathy, K., and Yadav, A., 2022a. The Application of stochastic mine production scheduling in the presence of geological uncertainty. *Sustainability,* Vol. 14, No. 16, pp. 9819.

Joshi, D., Paithankar, A., Chatterjee, S. and Equeenuddin, S. M., 2022b. Integrated parametric graph closure and branch-and-cut algorithm for open pit mine scheduling under uncertainty. *Mining,* Vol. 2, No. 1, pp. 32-51.

Joshi, D. and Satpathy, S. K., 2020. Production scheduling of open pit mine using sequential branch-and-cut and longest path algorithm: an application from an African Copper Mine. *Journal Européen des Systèmes Automatisés,* Vol. 53, No. 5, pp. 629-636.

Kear, R., 2006. Strategic and tactical mine planning components. *The Journal of the South African Institute of Mining and Metallurgy,* Vol. 106, pp. 93-96.

Khan, A., 2018. Long-term production scheduling of open pit mines using particle swam and bat algorithms under grade uncertainty. *Journal of the Southern African Institute of Mining and Metallurgy*, Vol. 118, No. 4, pp. 361-368.

Khan, A., Asad, A. M. W. and Topal, E., 2024. A heuristic method for production scheduling of an open pit mining operation. *International Journal of Mining, Reclamation and Environment,* Vol. 38, No. 4, pp. 293-305.

Khan, A. and Delius, C. N., 2015. Application of particle swarm optimisation to the open pit mine scheduling problem. *Proceedings of the 12th International Symposium Continuous Surface Mining*. Springer International Publishing, pp. 195-212.

Kloppers, B., Horn, C. and Visser, J., 2015. Strategic and tactical requirements of a mining long-term plan. *The Journal of the Southern African Institute of Mining and Metallurgy,* Vol. 115, No. 6, pp. 515-521.

Kozan, E. and Liu, S. Q., 2011. Operations research for mining: a classification and literature review. *Mathematical Sciences, Queensland University of Technology,* p. 23.

Kumral, M. and Dowd, P., 2005. A simulated annealing approach to mine production scheduling. *Journal of the Operational Research Society,* Vol. 56, No. 8, pp. 922-930.

Lamghari, A. and Dimitrakopoulos, R., 2016. Network-flow based algorithms for scheduling production in multi-processor open-pit mines accounting for metal uncertainty. *European Journal of Operational Research,* Vol. 250, No. 1, pp. 273-290.

Lamghari, A. and Dimitrakopoulos, R., 2018. Hyper-heuristic approaches for strategic mine planning under uncertainty. *Computers and Operations Research, https://doi.org/10.1016/j.cor.2018.11.010*.

Leite, A. and Dimitrakopoulos, R., 2007. Stochastic optimisation model for open pit mine planning: application and risk analysis at copper deposit. *Mining Technology,* Vol. 116, No. 3, pp. 109-118.

Leite, A. and Dimitrakopoulos, R., 2014. Stochastic optimization of mine production scheduling with uncertain ore/metal/waste supply. *International Journal of Mining Science and Technology,* Vol. 24, No. 6, pp. 755-762.

Levinson, Z. and Dimitrakopoulos, R., 2023. Connecting planning horizons in mining complexes with reinforcement learning and stochastic programming. *Resources Policy,* Vol. 86, No. 1, pp. 104-136.

Lin, M. H., Tsai, J. F., Yu, C. S. and Hu, Y. C., 2012. A Review of deterministic optimization methods in engineering and management. *Mathematical Problems in Engineering,* Vol. 2012, No. 1, p. 756023.

Liu, W. and Zhang, H., 2016. *PSO algorithm for block sequencing problem in open pit mining.* Sanya, China, 2016 Atlantis Press, pp. 434-441.

Lotfian, R., Gholamnejad, J. and Mirzaeian , Y. L., 2021. Effective solution of the long-term open pit production planning problem using block clustering. *Engineering Optimization,* Vol. 53, No. 7, pp. 1119-1134.

Lyeme, H. and Seleman, M., 2012. *Introduction to Operations Research: Theory and Applications.* 1st ed. Deutschland, Germany: LAP LAMBERT Academic Publishing.

Maleki, M., Jélvez, E., Emery, X. and Morales, N., 2020. Stochastic Open-Pit Mine Production Scheduling: A Case Study of an Iron Deposit. *Minerals,* Vol. 10, No. 7, p. 585.

Manríquez, F., Pérez, J. and Morales, N., 2020. A simulation-optimization framework for short-term underground mine production scheduling. *Optimization and Engineering,* Vol. 21, pp. 939-971.

Matamoros, M. E. V., 2014. *Stochastic short-term production scheduling accounting for fleet allocation, operational considerations, blending restrictions and future multi-element ore control data,* Montreal, Quebec: McGill University.

Matamoros, M. E. V. and Dimitrakopoulos, R., 2016. Stochastic short-term mine production schedule accounting for fleet allocation, operational considerations and blending restrictions. *European Journal of Operational Research,* Vol. 255, No. 3, pp. 911-921.

Menabde, M., Froyland, G., Stone, P. and Yeates, G., 2007. Mining schedule optimisation for conditionally simulated orebodies. *The Australasian Institute of Mining and Metallurgy,* Vol. 14, pp. 379-383.

MineLib, 2012. Minelib: A library of open-pit mining problems. http://mansci-web.uai.cl/minelib/Datasets.xhtml, [Accessed 2 July 2021].

Mitchell, M., 1999. *An Introduction to Genetic Algorithms.* 1st ed. London: First MIT Press.

Monika, 2021. *Operations Research Techniques.* 1st ed. Maharshi Dayanand University: Maharshi Dayanand University Press.

Moosavi, E. and Gholamnejad, J., 2015. Long-term production scheduling modelling for the open pit mines considering tonnage uncertainty via indicator kriging. *Journal of Mining Science,* Vol. 51, No. 6, pp. 1226-1234.

Morales, N., Seguel, S., Cáceres, A., Jélvez, E. and Alarcón, M., 2019. Incorporation of Geometallurgical Attributes and Geological Uncertainty into Long-Term Open-Pit Mine Planning. *Minerals,* Vol. 9, No. 2, p. 108.

Morley, C. and Arvidson, H., 2017. *Mine value chain reconciliation – demonstrating value through best practice. Tenth International Mining Geology Conference,* Hobart, Tasmania, pp. 279-292.

Mousavi, A., Kozan, E. and Liu, S. Q., 2016. Open-pit block sequencing optimization: a mathematical model and solution technique. *Engineering Optimization,* Vol. 48, No. 11, pp. 1932-1950.

Moxnes, E., 2015. An introduction to deterministic and stochastic optimization. In: *Analytical Methods for Dynamic Modelers.* Cambridge, Massachusetts: MIT Press, pp. 243-276.

Muke, P., Nhleko, A.S. and Musingwini, C., 2021. A genetic algorithm model for optimising long-term open-pit mine production scheduling, *Application of Computers and Operations Research in the Mining Industry, The Journal of the Southern African Institute of Mining and Metallurgy*, pp. 1-18.

Muke, P., Tholana, T., Musingwini, C., and Ali, M., 2025a. A genetic algorithm for temporal and spatial alignment of long- and medium-term mine production scheduling for open-pit mines. *Resources Policy*, Vol. 106, p. 105629. https://doi.org/10.1016/j.resourpol.2025.105629

Muke, P., Tholana, T., Musingwini, C., Ali, M. and Mutandwa, B., 2025b. Towards a system for integrating open-pit mine production scheduling stages. *Application of Computers and Operations Research in the Minerals Industry (APCOM) Conference*, 10-13 August 2025, Perth, Australia.

Murthy, R. P., 2007. *Operations Research.* 2nd ed. New Delhi: New Age International.

Musingwini, C., 2016. Presidential Address: Optimization in underground mine planning - developments and opportunities, *The Journal of the Southern African Institute of Mining*

and Metallurgy, Vol. 116, No. 9, pp. 809-820. DOI: http://dx.doi.org/10.17159/2411-9717/2016/v116n9a1

National Research Council, 2002. *Evolutionary and Revolutionary Technologies.* Washington, DC: The National Academies Tress.

Newman, A. M., Rubio, E., Caro, R., Weintraub, A. and Eurek, K., 2010. A review of operations research in mine planning. *Interfaces,* Vol. 40, No. 3, pp. 222-245.

Nhleko, A., Tholana, T. and Neingo, P., 2018. A review of underground stope boundary optimization algorithms. *Resources Policy,* Vol. 56, pp. 59-69.

Nogholi, A. M., 2015. Optimisation of open pit mine block sequencing*, PhD Thesis,* Queensland: Queensland University of Technology.

Osanloo, M., Gholamnejad, J. and Karimi, B., 2008. Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment,* Vol. 22, No. 1, pp. 3-35.

Otto, T. J., 2019. A spatial mine-to-plan compliance framework for open-pit iron ore mines, *PhD Thesis,* Johannesburg: University of the Witwatersrand.

Otto, T. and Musingwini, C., 2019. A spatial mine-to-plan compliance approach to improve alignment of short- and long-term mine planning at open pit mines. *The Journal of the Southern African Institute of Mining and Metallurgy,* Vol. 119, pp. 253-259.

Otto, T. and Musingwini, C., 2020. A compliance driver tree (CDT) based approach for improving the alignment of spatial and intertemporal execution with mine planning at open-pit mines. *Resources Policy*, Vol. 69, pp. 1-9.

Paithankar, A. and Chatterjee, S., 2019. Open pit mine production schedule optimization using a hybrid of maximum-flow and genetic algorithms. *Applied Soft Computing Journal*, Vol. 81, pp. 105507.

Porcu, V., 2018. *Python for Data Mining Quick Syntax Reference.* 1st ed. Nuoro, Italy: Apress.

Quigley, M. and Dimitrakopoulos, R., 2020. Incorporating geological and equipment performance uncertainty while optimising short-term mine production schedules.

International Journal of Mining, Reclamation and Environment, Vol. 34, No. 5, pp. 362-383.

Ramazan, S., Dagdelen, K. and Johnson, T., 2005. Fundamental tree algorithm in optimizing production scheduling for open pit mine design. *Transactions of the Institution of Mining and Metallurgy (Sec. A: Mining Technol.),* Vol. 114, pp. A41-A45.

Ramazan, S. and Dimitrakopoulos, R., 2004. Stochastic optimisation of long-term production scheduling for open pit mines with a new integer programming formulation. *Orebody Modelling and Strategic Mine Planning Spectrum Series,* Vol. 14, pp. 139-153.

Rardin, R. L., 2017. *Optimisation in Operation Research.* 2nd ed. Arkansas: Pearson.

Roman, R. J., 1974. The role of time value of money in determining an open pit mining sequence and pit limits. *12th Symposium on the Application of Computer and Operations Research in the Mineral Industry*, pp. 72-85.

Rosas, J. S., Jélvez, E., Amaya, . J. and Morales, N., 2016. Optimizing open pit block scheduling with exposed ore reserve. *Southern African Institute of Mining and Metallurgy,* Vol. 116, No. 7, pp. 655-622.

Rovenscroft, P., 1992. Risk analysis for mine scheduling by conditional simulation. *Transactions of the Institution of Mining and Metallurgy,* Vol. 101, pp. A82-A88.

Ruiseco, J. R., 2016. *Dig-limit optimization in open pit mines through genetic algorithms,* Montreal, *MSc Report*, McGill University Libraries.

Samavati, M., Essam, D., Nehring, M. and Sarker, R., 2018. A new methodology for the open-pit mine production scheduling. *Omega*, Vol. 81, pp. 169-182.

SAMREC, 2016. The South African Code for the Reporting of Exploration Results, Mineral Resources and Mineral Reserves, South Africa.

Sari, Y. and Kumral, M., 2016. An improved meta-heuristic approach to extraction sequencing and block routing. *The Journal of the Southern African Institute of Mining and Metallurgy,* Vol. 116, No. 7, pp. 673-680.

Sattarvand, J., 2009. Long-term open-pit planning by ant colony optimisation, *PhD Thesis,* Germany: RWTH Aachen University.

Senécal, R. and Dimitrakopoulos, R., 2020. Long-term mine production scheduling with multiple processing destinations under mineral supply uncertainty, based on multi-neighbourhood Tabu search. *International Journal of Mining, Reclamation and Environment*, Vol. 34, No. 7, pp. 459-475.

Shishvan, M. S. and Sattarvand, J., 2015. Long-term production planning of open pit mines by ant colony optimization. *European Journal of Operational Research*, Vol. 240, pp. 825-836.

Smith, G., 2012. Strategic long term planning in mining. *The Journal of The Southern African Institute of Mining and Metallurgy,* Vol. 112, No. 1, pp. 761-774.

Smith, M., 1999. Optimizing inventory stockpiles and mine production: An application of separable and goal programming to phosphate mining using AMPL/Cplex. *Canadian Institute of Mining, Metallurgy and Petroleum*, Vol. 92, pp. 61-64.

Tholana, T., Musingwini, C. and Ali, M., 2019. A stochastic block economic value model. Mine Planning Colloquium, *The Southern African Institute of Mining and Metallurgy,* pp. 35-49.

Tholana, T., 2021. Generation of probabilistic stopes using Monte-Carlo simulation of block economic values for use in mine planning under uncertainty. *PhD Thesis*, University of the Witwatersrand.

Tholana, T. and Musingwini, C., 2022. A probabilistic block economic value calculation method for use in stope designs under uncertainty. *Minerals* 2022, 12, 437. https://doi.org/10.3390/min12040437

Thorley, U., 2012. Open pit mine planning: analysis and system modelling of conventional and oil sands applications, *PhD Thesis,* Kingston, Ontario, Canada: Queen's University.

Tolouei, K., Moosavi, E., Bangian A. H., Afzal, P. and Aghajani B. A., 2020. A comprehensive study of several meta-heuristic algorithms for the open-pit mine production scheduling problem considering grade uncertainty. *Journal of Mining and Environment,* Vol. 11, No. 3, pp. 721-736.

Tolwinski, B. and Underwood, R., 1992. An algorithm to estimate the optimal evolution of an open pit mine. *Application of Computers and Operations Research in the Mining Industry*, pp. 399-409.

Upadhyay, S. P. and Askari-Nasab, H., 2018. Simulation and optimization approach for uncertainty-based short-term planning in open pit mines. *International Journal of Mining Science and Technology,* Vol. 28, No. 2, pp. 153-166.

Vallejo, M. N. and Dimitrakopoulos, R., 2018. Stochastic orebody modelling and stochastic long-term production scheduling at the KéMag iron ore deposit, Quebec, Canada. *International Journal of Mining, Reclamation and Environment,* Vol. 33, No. 7, pp. 462-479.

Venkatraman, S. and Yen, G. G., 2005. A Generic Framework for Constrained Optimization Using Genetic Algorithms. *The Institute of Electrical and Electronics Engineers (IEEE) Transactions on Evolutionary Computation,* Vol. 9, No. 4, pp. 424-435.

Verhoeff, R., 2017. Using genetic algorithms for underground stope design optimisation in mining, *PhD Thesis,* Netherland: Delft University of Technology.

Wilke, F., 1987. Recent methodological trends in operations research and computing as applied to mining problems. *The Journal of the Southern African Institute of Mining and Metallurgy*, pp. 109-113.

Yang, X. S., 2011. *Metaheuristic optimization: algorithm analysis and open problems.* Chania, Greece, pp. 21-32.

Yang, X. S., 2012. Nature-inspired metaheuristic algorithms: success and new challenges. *Journal of Computer Engineering and Information Technology,* Vol. 1, No. 1, pp. 1-3.

Yang, X. S., 2014. *Nature-Inspired Optimization Algorithms.* 1st ed. London: Elsevier Inc.

8. APPENDIX

8.1 Python code representing the combined mixed integer programming model and GA approach integrating LT and MT production schedules

```
import csv
import random
import time

# the file that stores the block id, and it's preceds
preceds_file = ''
# the file that stores the block id, and it's value
block_economic_value_file = ''

# the file that stores coordinates of blocks
coordinates_file = ''

max_times = 40
size_of_top_best = 1

index_of_preceds = 1
index_of_tonnage = 6
index_of_z_coordinate = 3
value_lines = []
preceds_lines = []
coordinate_lines = []

# periods_fitness_file_name
periods_fitness_file_name = ''

# we want to measure the time it takes to complete
start = time.time()


def initialize_coordinate_lines():
    print("initialize_coordinate_lines")
    global coordinate_lines

    with open(coordinates_file, newline='') as file:
        coordinate_lines = list(csv.reader(file, delimiter=';'))


def set_coordinates_file_path(path):
    print("set_coordinates_file_path")
    global coordinates_file
    coordinates_file = path
    initialize_coordinate_lines()


def set_periods_fitness_file_name(name):
        print("set_periods_fitness_file_name")
        global periods_fitness_file_name
        periods_fitness_file_name = name


def set block economic value file(path):
    print("set_block_economic_value_file")
    global block_economic_value_file
    block_economic_value_file = path
```

```
    global value_lines

    with open(block_economic_value_file, newline='') as file:
        value_lines = list(csv.reader(file, delimiter=';'))


def initialize_preceds_lines():
    print("initialize_preceds_lines")
    global preceds_lines

    with open(preceds_file, newline='') as cvs_preceds_file:
        preceds_lines = list(csv.reader(cvs_preceds_file,
delimiter=';'))


def set_preceds_file_path(path):
    print("set_preceds_file_path")
    global preceds_file
    preceds_file = path
    initialize_preceds_lines()


def initialize_all_data():
    print("Initialize values")
    # initialize preceds lines()


def get_preced_line(generated_block):
    return preceds_lines[generated_block]


# generate block_number not in the list
def generate_block_not_in_block_list(search_space_1, blocked_list):
    choices = [x for x in search_space_1 if (x not in blocked_list)]

    # if no more choices[choices are empty]
    if len(choices) == 0:

        #  return -1 so we can terminate
        return -1

    else:

        return random.choice(choices)


def get_block_id_direct_preceds(original_search_space, block_id):
    line = get_preced_line(block_id)
    num_preceds_in = int(line[index_of_preceds])

    new_list = []

    # check if num_preceds_in is greater than 0
    if num_preceds_in > 0:

        for index in range(num_preceds_in):
            preced = int(line[index + 2])

            if preced in original_search_space:
                new_list.append(preced)
```

```
    return new_list


def get_block_id_preceds(original_search_space, block_id, found,
excluded_list):
    line = get_preced_line(block_id)
    num_preceds_in = int(line[index_of_preceds])

    new_list = []

    # check if num_preceds_in is greater than 0
    if num_preceds_in > 0:

        # range is exclusive
        for index in range(num_preceds_in):

            preced = int(line[index + 2])

            if preced not in excluded_list and preced in
original_search_space:
                new_list.append(preced)

    index = 0
    len_of_new = len(new_list)
    # print("new list without block id: ", new list)

    while index < len_of_new:

        preced_id = new_list[index]

        if preced_id not in found:
            # print("found: ", found)
            preceds_of_preced =
get_block_id_preceds(original_search_space, preced_id, found,
excluded_list)
            # print("preceds_of_preced: ", preceds_of_preced)
            found = merge_to_list_distinct(found, preceds_of_preced)
            new_list = merge_to_list_distinct(new_list,
preceds_of_preced)

        index += 1

    return new_list


def get_block_id_with_preceds_of_preceds(original_search_space,
block_id, excluded_list):

    line = get_preced_line(block_id)
    num_preceds_in = int(line[index_of_preceds])

    new_list = [block_id]

    # check if num_preceds_in is greater than 0
    if num_preceds_in > 0:

        # range is exclusive
        for index in range(num_preceds_in):
            preced = int(line[index + 2])

            if preced not in excluded_list and preced in
```

```
original_search_space:
                # if preced not in excluded_list :
                new_list.append(preced)

    index = 1
    len_of_new = len(new_list)

    found = [block_id]

    while index < len_of_new:
        preced_id = new_list[index]
        preceds_of_preced = get_block_id_preceds(original_search_space,
preced_id, found, excluded_list)
        found = merge_to_list_distinct(found, preceds_of_preced)
        new_list = merge_to_list_distinct(new_list, preceds_of_preced)

        index += 1

    return new_list


def merge_to_list_distinct(old_list, new_list):

    if len(new_list) == 0:
        return old_list

    list__ = old_list.copy()

    for block_in_new_list in new_list:

        if block_in_new_list not in old_list:
            # add to list
            list__.append(block_in_new_list)

    return list__


# TODO remove not used
def get_repeated_values(list_1, list_2):
    list_1_items_in_list_2 = []
    list_2_items_in_list_1 = []

    for block_l1 in list_1:
        if block_l1 in list_2:
            list_1_items_in_list_2.append(block_l1)

    for block_l2 in list_2:
        if block_l2 in list_1:
            list_2_items_in_list_1.append(block_l2)

# TODO remove not used
def find_items_not_in_list(list_1, items):
    items_not_in_list = []

    for block in items:
        if block not in list_1:
            items_not_in_list.append(block)

def remove_list_items(old_list, items):
    list__ = old_list.copy()
```

```
    for kk in items:

        if kk in old_list:
            # print("Removing: ", kk)
            list__.remove(kk)

    return list__


def generate_list_of_list(search_space_1, excluded_list_1, minimum_size,
maximum_size):
    # print(maximum_size)

    list_of_list = []
    excluded_list = excluded_list_1.copy()
    none_duplicated = [excluded_list]
    local_search_space = search_space_1.copy()

    count = 0
    # generate the first [max times] list
    while count < max_times:
        # print('max_times:= ', max_times)

        # generate a valid set and add to set
        new_block = generate_block_not_in_block_list(local_search_space,
none_duplicated)

        if new_block == -1:

            # no more list that can be merged
            print("No more blocks can be generated, Error with list")
            count += 1

        else:
            # add to the list a valid generated block
            block_list =
get_block_id_with_preceds_of_preceds(search_space_1, new_block,
excluded_list)
            # print('post 000000000000000000000')

            num_gens = 0

            # we want to combine and merge
            while len(block_list) < minimum_size and len(block_list) !=
len(search_space_1):
                # print('post 11111111111111111111')

                new_gen_in =
generate_block_not_in_block_list(local_search_space, block_list)
                # print('post 22222222222222222222')

                if new_gen_in == -1:

                    # no more list that can be merged
                    print("No more blocks can be generated, Error with
list")

                else:
                    num_gens += 1

                    generated_new =
```

```
get_block_id_with_preceds_of_preceds(search_space_1, new_gen_in,
excluded_list)

                    clean = []

                    for block in generated_new:
                        # print('post
333333333333333333333333333333333')
                        if block not in block_list:
                            clean.append(block)

                    if len(block_list) + len(clean) <= maximum_size:
                        block_list = merge_to_list_distinct(block_list,
clean)
                        # print('post 444444444444444444444444444444')

            # only add when the block list size is in within the range
of the min and max
            # print('minimum_size:= ', minimum_size)
            # print('block list:= ', block list)
            # print('len(block list):= ', len(block list))
            # print('maximum_size:= ', maximum_size)
            if minimum_size <= len(block_list) <= maximum_size:
                # add to final set
                list_of_list.append(block_list)

                # increment the counter if list is added to blocks set

                count += 1

                none_duplicated =
merge_to_list_distinct(none_duplicated, [new_block])
                # print('post 555555555555555555555555')
            else:
                count += 1

    return list_of_list


# get the value line
def get_value_line(generated_block):
    return value_lines[generated_block]


# for each block in a set read the block value
def get_sorted_by_value_dic(dictionary):
    # return dictionary sorted by value
    return sorted(dictionary.items(), key=lambda x: x[1], reverse=True)


index_of_value = 1


def get_position_values_dic(list_of_list):
    # print('get_position_values_dic list_of_list: ', list_of_list)
    # create s list to store the position and the value
    position_values = {}

    position_counter = 0

    for block_ids in list_of_list:
```

```
        final_value = float('0')

        for block_id in block_ids:
            # get the value line for the block
            # print('Block id: ', block_id)
            value_line = get_value_line(block_id)

            # in the line get value as a float
            block_value = float((value_line[index_of_value]))

            # add the value to final
            # final_value = final_value + block_value
            final_value += block_value

        # add the position and the final value to the position values
list
        position_values[position_counter] = final_value
        position_counter += 1

    return position_values


def get_block_with_values_dictionary(block_list_1):
    dic = {}

    for block_id in block_list_1:
        # get the value of the block and add to dic

        # in the line get value as a float
        value = float(get_value_line(block_id)[index_of_value])
        # add to dictionary
        dic[block_id] = value

    return dic


def get_weakest_block_id(list_best):
    # return the first element of the sorted[ascending order] dictionary
item[the block number]
    # return sorted(get_block_with_values_dictionary(list_best).items(),
key=lambda x: x[1])[0]

    # dic_item1 = dd_sort[0]
    # position_in_set = dic_item[0]
    # print("get_weakest_block_id", dd_sort)
    return (sorted(get_block_with_values_dictionary(list_best).items(),
key=lambda x: x[1])[0])[0]


def get_strongest_block_id(list_weak):
    # return the first element of the sorted[ascending order] dictionary
item[the block number]
    return (sorted(get_block_with_values_dictionary(list_weak).items(),
key=lambda x: x[1], reverse=True)[0])[0]


def get_coordinate_line(generated_block):
    return coordinate_lines[generated_block]
```

```
def get_lowest_value_block_id_at_bottom(waste):
    lowest_z_coordinate = -1

    bottom_blocks = []

    for row in waste:

        coordinate_line = get_coordinate_line(row)

        # in the line get coordinate as an integer
        coordinate_z = int(coordinate_line[index_of_z_coordinate])

        # first time running
        if lowest_z_coordinate == -1:

            lowest_z_coordinate = coordinate_z
            bottom_blocks.append(row)

        else:

            if coordinate_z < lowest_z_coordinate:
                # create a new list and add the block
                lowest_z_coordinate = coordinate_z
                bottom_blocks = [row]

            elif coordinate_z == lowest_z_coordinate:

                bottom_blocks.append(row)

    print('bottom_index_z: ', lowest_z_coordinate)
    print('bottom_blocks: ', bottom_blocks)

    return get_weakest_block_id(bottom_blocks)


def get_highest_value_block_id_at_top(ore):
    top_z_coordinate = -1

    top_blocks = []

    for row in ore:

        coordinate_line = get_coordinate_line(row)

        # in the line get coordinate as an integer
        coordinate_z = int(coordinate_line[index_of_z_coordinate])

        # first time running
        if top_z_coordinate == -1:

            top_z_coordinate = coordinate_z
            top_blocks.append(row)

        else:

            if coordinate_z > top_z_coordinate:
                # create a new list and add the block
                top_z_coordinate = coordinate_z
                top_blocks = [row]

            elif coordinate_z == top_z_coordinate:
```

```
                top_blocks.append(row)

    print('top_index_z: ', top_z_coordinate)
    print('top_blocks: ', top_blocks)

    return get_strongest_block_id(top_blocks)


def get_bottom_blocks(list_blocks):
    lowest_z_coordinate = -1

    bottom_blocks = []

    for row in list_blocks:

        coordinate_line = get_coordinate_line(row)

        # in the line get coordinate as an integer
        coordinate_z = int(coordinate_line[index_of_z_coordinate])

        # first time running
        if lowest_z_coordinate == -1:

            lowest_z_coordinate = coordinate_z
            bottom_blocks.append(row)

        else:

            if coordinate_z < lowest_z_coordinate:
                # create a new list and add the block
                lowest_z_coordinate = coordinate_z
                bottom_blocks = [row]

            elif coordinate_z == lowest_z_coordinate:

                bottom_blocks.append(row)

    return bottom_blocks


def get_top_blocks_without_preceds(original_search_space, list_blocks,
excluded):
    blocks = []

    for block in list_blocks:

        line = get_preced_line(block)
        # print(line)
        num_preceds = int(line[index_of_preceds])

        if num_preceds == 0:

            blocks.append(block)

        else:

            preceds = get_block_id_direct_preceds(original_search_space,
block)

            if list_contains_all_values(excluded, preceds):
```

```
                blocks.append(block)

    return blocks


def get_blocks_value(list_blocks):
    value = float('0')
    # print('get_blocks_value list_blocks: ', list_blocks)
    for block_id in list_blocks:
        # in the line get value as a float
        # print('block_id: ', block_id))
        # print('type block_id: ', type(block_id))
        value_line = get_value_line(block_id)
        # print('value_line: ', value_line)
        # print('type value_line: ', type(value_line))
        block_value_1 = float(value_line[index_of_value])
        # print('block with id: ', block)
        # print('value for block: ', block_value_1)

        # add the value to final
        # final value = final value + block value
        value += block_value_1

    return value


def get_ore_blocks(list_blocks):
    blocks = []

    for block in list_blocks:

        # in the line get value as a float
        block_value_1 = float(get_value_line(block)[index_of_value])
        # print('block with id: ', block)
        # print('value for block: ', block_value_1)

        # add the value to final
        # final value = final value + block value
        if block_value_1 >= 0:
            blocks.append(block)

    return blocks


def get_waste_blocks(list_blocks):
    blocks = []

    for block in list_blocks:
        block_value_1 = float(get_value_line(block)[index_of_value])
        if block_value_1 < 0:
            blocks.append(block)

    return blocks


def get_top_blocks(list_blocks):
    top_z_coordinate = -1

    top_blocks = []

    for row in list_blocks:
```

```
        coordinate_line = get_coordinate_line(row)

        # in the line get coordinate as an integer
        coordinate_z = int(coordinate_line[index_of_z_coordinate])

        # first time running
        if top_z_coordinate == -1:

            top_z_coordinate = coordinate_z
            top_blocks.append(row)

        else:

            if coordinate_z > top_z_coordinate:
                # create a new list and add the block
                top_z_coordinate = coordinate_z
                top_blocks = [row]

            elif coordinate_z == top_z_coordinate:

                top_blocks.append(row)

    return top_blocks


def get_highest_value_blocks_of_size(list_blocks, size):
    #
    sorted_dict =
sorted(get_block_with_values_dictionary(list_blocks).items(), key=lambda
x: x[1], reverse=True)
    # print('get_highest_value_blocks_of_size sorted_dict: ',
sorted_dict)

    blocks = []

    counter_1 = 0

    while counter_1 < size:
        # get the block id from dictionary sorted by the highest value
        # print('dic item: ', sorted_dict[counter_1])
        block_id = sorted_dict[counter_1][0]
        # block id = sorted dict[counter 1]

        # print('get_highest_value_blocks_of_size block id: ', block_id)

        counter_1 += 1
        # add to list of best number
        blocks.append(block_id)

    return blocks


def list_contains_all_values(list_blocks, check_list):
    for block in check_list:

        if block not in list_blocks:
            return False

    return True
```

```
# Mutation genetic operator
def get_mutated_list(search_space_11, list__, excluded_list,
maximum_size):
    list_temp = list__.copy()
    search_space_1 = search_space_11.copy()

    if len(list_temp) < maximum_size:

        # print('get_mutated_list: ', get_mutated_list)
        # print('list_1_size: before adding top cooked data',
len(list_temp))
        open_spaces_of_list = maximum_size - len(list_temp)
        # print('open_spaces_of_list: ', open_spaces_of_list)

        # remove all blocks in list 1 from original blocks
        original_minus_list = remove_list_items(search_space_1,
list_temp)
        # print('original_minus_list: ', original_minus_list)

        # get the ore blocks at the top of original minus list 1
        available_ore_top_blocks =
get_top_blocks_without_preceds(search_space_11, original_minus_list,
excluded_list)
        # print('available ore top blocks: ', available_ore_top_blocks)
        available_ore_top_blocks_size = len(available_ore_top_blocks)
        # print('available_ore_top_blocks_size: ',
available_ore_top_blocks_size)

        to_add = available_ore_top_blocks
        # print('to_add outside length: ', len(to_add))

        if available_ore_top_blocks_size > open_spaces_of_list:
            to_add =
get_highest_value_blocks_of_size(available_ore_top_blocks,
open_spaces_of_list)
            # print('to_add inside: ', to_add)
            # print('to add inside length: ', len(to add))

        list_temp = merge_to_list_distinct(list_temp, to_add)
        # print('after merge inside: ', list_temp)

    # print('list temp length: ', len(list temp))
    return list_temp


# function that will cross a list to create a new list
# Crossover genetic operator


def get_crossed_mutated_lists(search_space_11, list_11, list_22,
excluded_list, minimum_size, maximum_size):
    # print('weakest_bottom_block: ',
get_lowest_value_block_id_at_bottom(top_top))
    # print('strongest_top_block: ',
get_highest_value_block_id_at_top(top_top))
    list_1 = list_11.copy()
    list_2 = list_22.copy()
    search_space_1 = search_space_11.copy()

    bottom_blocks_1 = get_bottom_blocks(list_1)
```

```
    bottom_blocks_2 = get_bottom_blocks(list_2)

    top_blocks_1 = get_top_blocks_without_preceds(search_space_11,
list_1, excluded_list)
    top_blocks_2 = get_top_blocks_without_preceds(search_space_11,
list_2, excluded_list)

    bottom_blocks_value_1 = get_blocks_value(bottom_blocks_1)
    bottom_blocks_value_2 = get_blocks_value(bottom_blocks_2)

    top_blocks_with_ore_1 =
get_top_blocks_without_preceds(search_space_11, top_blocks_1,
excluded_list)
    top_blocks_with_ore_2 =
get_top_blocks_without_preceds(search_space_11, top_blocks_2,
excluded_list)

    top_blocks_with_ore_value_1 =
get_blocks_value(top_blocks_with_ore_1)
    top_blocks_with_ore_value_2 =
get_blocks_value(top_blocks_with_ore_2)

    # print('bottom blocks 1: ', bottom_blocks_1)
    # print('bottom blocks 2: ', bottom_blocks_2)
    # print('bottom blocks value 1: ', bottom blocks value 1)
    # print('bottom blocks value 2: ', bottom_blocks_value_2)
    #
    # print('top blocks 1: ', top_blocks_1)
    # print('top blocks 2: ', top_blocks_2)
    # print('top blocks value 1: ', top blocks with ore value 1)
    # print('top blocks value 2: ', top_blocks_with_ore_value_2)

    # print('old list 1 size: ', len(list_1))
    # print('old list 1 value: ', get blocks value(list 1))

    # print('old list 2 size: ', len(list_2))
    # print('old list 2 value: ', get_blocks_value(list_2))

    less_than_1 = bottom_blocks_value_1 < top_blocks_with_ore_value_2
    list_does_not_contains_all_values_1 =
list_contains_all_values(list_1, top_blocks_with_ore_2) is False
    enter_if_statement_1 = less_than_1 and
list_does_not_contains_all_values_1
    # print('less_than_1: ', less_than_1)
    # print('list_does_not_contains_all_values_1: ',
list_does_not_contains_all_values_1)
    # print('enter_if_statement_1: ', enter_if_statement_1)

    if enter_if_statement_1:

        list_1_temp = list_1.copy()

        list_1_temp = remove_list_items(list_1_temp, bottom_blocks_1)
        # print('after remove list_1_temp: ', list_1_temp)
        # print('after remove list_1_temp length: ', len(list_1_temp))

        list_1_temp = merge_to_list_distinct(list_1_temp,
top_blocks_with_ore_2)
        # print('after merge list_1_temp: ', list_1_temp)
        # print('after merge list_1_temp length: ', len(list_1_temp))
        # print('after merge list_1: ', list_1)
```

```
        # print('after merge list_1 length: ', len(list_1))

        if minimum_size <= len(list_1_temp) <= maximum_size:
            list_1 = list_1_temp

    less_than_2 = bottom_blocks_value_2 < top_blocks_with_ore_value_1
    list_does_not_contains_all_values_2 =
list_contains_all_values(list_2, top_blocks_with_ore_1) is False
    enter_if_statement_2 = less_than_2 and
list_does_not_contains_all_values_2
    # print('less_than_2: ', less_than_2)
    # print('list_does_not_contains_all_values_2: ',
list_does_not_contains_all_values_2)
    # print('enter_if_statement_2: ', enter_if_statement_2)

    if enter_if_statement_2:

        list_2_temp = remove_list_items(list_2, bottom_blocks_2)
        # print('after remove list_2_temp: ', list_2_temp)
        # print('after remove list 2 temp length: ', len(list 2 temp))
        list_2_temp = merge_to_list_distinct(list_2_temp,
top_blocks_with_ore_1)
        # print('after merge list_2_temp: ', list_2_temp)
        # print('after merge list_2_temp length: ', len(list_2_temp))
        # print('after merge list 2: ', list 2)

        if minimum_size <= len(list_2_temp) <= maximum_size:
            list_2 = list_2_temp

    # get highest value blocks of size(list 1, 10)

    # mutate list 1
    # print('new list 1 before mutation: ', len(list_1))
    # print('new list 2 before mutation: ', len(list 2))

    list_1 = get_mutated_list(search_space_1, list_1, excluded_list,
maximum_size)
    list_2 = get_mutated_list(search_space_1, list_2, excluded_list,
maximum_size)

    # print('new list 1: ', list_1)
    # print('new list 1 size: ', len(list_1))
    # print('new list 1 value: ', get blocks value(list 1))
    #
    # print('new list 2: ', list_2)
    # print('new list 2 size: ', len(list_2))
    # print('new list 2 value: ', get_blocks_value(list_2))

    crossed_0 = [list_1, list_2]

    return crossed_0


def get_top_best_in_value_of_size(list_of_list, size):
    sorted_dic =
get_sorted_by_value_dic(get_position_values_dic(list_of_list))

    counter_0 = 0
    best = []

    size = len(list_of_list)
```

```
    while counter_0 < size:
        # get the dictionary item at a position of counter[top_counter]
        # dic item = ()
        # try:
        #     # code that may cause exception
        #     dic_item = sorted_dic[counter_0]
        #     # print('Works')
        # except:
        #     # code to run when exception occurs
        #     print('list index out of range')

        dic_item = sorted_dic[counter_0]
        position_in_set = dic_item[0]

        counter_0 += 1
        # add to list of best number
        best.append(list_of_list[position_in_set])

    return best


def merge_to_list_of_lists_distinct(list_of_list_11, list_of_list_22):
    # print('merge_to_list_of_lists_distinct list_of_list_1: ',
list_of_list_1)
    # print('merge to list of lists distinct list of list 2: ',
list_of_list_2)
    list_of_list_1 = list_of_list_11.copy()
    list_of_list_2 = list_of_list_22.copy()

    counter_q = 0

    while counter_q < len(list_of_list_1):
        list_item = list_of_list_1[counter_q]
        sorted_list_item = sorted(list_item)
        list_of_list_1[counter_q] = sorted_list_item

        counter_q += 1

    counter_q = 0

    while counter_q < len(list_of_list_2):
        list_item = list_of_list_2[counter_q]
        sorted_list_item = sorted(list_item)
        list_of_list_2[counter_q] = sorted_list_item

        counter_q += 1

    combined = list_of_list_1

    for list_2 in list_of_list_2:

        if list_2 not in list_of_list_1:
            combined.append(list_2)

    # print('combined : ', combined)
    return combined


def get_left_overs(list_of_list):
    sorted_dic =
get_sorted_by_value_dic(get_position_values_dic(list_of_list))
```

```
    left_overs_list = []

    # start from top going down
    counter_0 = size_of_top_best

    # getting the leftovers after we got the top_best
    while counter_0 < len(list_of_list):
        # get the dictionary item at a position of counter[top_counter]
        dic_item = sorted_dic[counter_0]
        position_in_set = dic_item[0]

        counter_0 += 1
        # add to list of best number
        left_overs_list.append(list_of_list[position_in_set])

    return left_overs_list


def get_total_tonnage_value(list_):
    tonnage = float('0')

    for block_id in list_:
        line = get_coordinate_line(block_id)

        value_1 = float(line[index_of_tonnage])

        tonnage += value_1

    return tonnage


def get_ore_tonnage_value(list_):
    tonnage = float('0')

    for block_id in list_:

        value_line = get_value_line(block_id)

        # in the line get value as a float
        block_value = float(value_line[index_of_value])

        if block_value >= float(0):
            line = get_coordinate_line(block_id)

            value_1 = float(line[index_of_tonnage])

            tonnage += value_1

    return tonnage


def get_value_of_generation(generation):
    total_value = float('0')

    for list_0 in generation:
        # get the value of the list
        value_of_list = get_blocks_value(list_0)

        total_value += value_of_list
```

```
    return total_value


# function to export data to a file("data_plot.csv)
def write_results_to_file(results, csv_file_name):
    with open(csv_file_name, "w+", newline="") as f:
        writer = csv.writer(f)
        writer.writerows(results)


def write_results_to_file_append(results, csv_file_name):
    with open(csv_file_name, "a") as myfile:
        myfile.write(results)


def get_topography_elevation(search_space):
    top_most_z_value = -1
    for block_id in search_space:
        coordinate_line = get_coordinate_line(block_id)
        z_coordinate_of_block =
int(coordinate_line[index_of_z_coordinate])
        if z_coordinate_of_block > top_most_z_value:
            top_most_z_value = z_coordinate_of_block
    return top_most_z_value


def get_search_space_from_elevation(original_search_space,
lowest_elevation):
    new_search_from_elevation = []
    for block_id in original_search_space:
        coordinate_line = get_coordinate_line(block_id)
        # in the line get coordinate as an integer
        z_coordinate_of_block =
int(coordinate_line[index_of_z_coordinate])
        if lowest_elevation <= z_coordinate_of_block:
            new_search_from_elevation.append(block_id)
    return new_search_from_elevation


def generate_time_periods_data(start_elevation, search_space_main,
minimum_size, maximum_size, generation_lists_file):
    print('Generating best block combination for each period:')
    # setting the search space as a list that includes the ID of blocks
    search_space = search_space_main.copy()
    used_best_list_of_last_gen = []
    best_list_of_last_gen = []
    time_period_counter = 0
    time_period_data = []
    topography_elevation = get_topography_elevation(search_space_main)
    # go through all the blocks, filter out blocks with z value outside
elevation
    lowest_elevation_allowed = topography_elevation - start_elevation
    new_search_from_elevation =
get_search_space_from_elevation(search_space_main,
lowest_elevation_allowed)

    search_space = new_search_from_elevation

    updated_global_search_space = search_space_main

    while len(search_space) < maximum_size and len(search_space) <
```

```
len(updated_global_search_space):
        start_elevation = start_elevation + 1
        lowest_elevation_allowed = topography_elevation -
start_elevation
        new_search_from_elevation =
get_search_space_from_elevation(updated_global_search_space,

lowest_elevation_allowed)
        search_space = new_search_from_elevation

    # list_outside_elevation =
remove_list_items(updated_global_search_space,
new_search_from_elevation)

    generation_fitness_value = []
    while len(search_space) >= maximum_size:
        print('**********search_space:')
        print(search_space)
        print('**********maximum_size:')
        print(maximum_size)
        print('**********Processing result for time period: ',
time_period_counter + 1)
        generated = generate_list_of_list(search_space,
used_best_list_of_last_gen, minimum_size, maximum_size)
        print('post generate_list_of_list')
        # sort the initial list
        generation = get_top_best_in_value_of_size(generated,
len(generated))
        generations = [generation]

        counter = 0
        value_of_prev_gen = float(0)
        value_of_last_gen = get_value_of_generation(generation)
        entered_while = False

        while value_of_last_gen != value_of_prev_gen or entered_while is
False:

            entered_while = True

            top_best = get_top_best_in_value_of_size(generation,
size_of_top_best)
            left_overs = get_left_overs(generation)

            # # start of crossing and mutations
            # counter_in = 0
            # cross_mutated_list = []
            #
            # while counter_in < size_of_top_best:
            #     # get the top list
            #     top_list = top_best[counter_in]
            #
            #     # get the bottom list
            #     bottom_list = top_best[counter_in + 1]
            #
            #     crossed_mutated =
get crossed mutated lists(search space, top list, bottom list,
            #
best_list_of_last_gen, minimum_size,
            #
maximum_size)
```

```
            #     cross_mutated_list.append(crossed_mutated[0])
            #     cross_mutated_list.append(crossed_mutated[1])
            #
            #     counter_in += 2
            # # end of crossing and mutations

            # ******** Random start of crossing and mutations
            counter_in = 0
            cross_mutated_list = []

            while counter_in < size_of_top_best:
                # get the top list
                top_list = top_best[counter_in]
                max_size_generation = len(generation)
                # get the bottom list
                random_num_excluding_counter_in =
random.choice(list(set([x for x in range(0, max_size_generation)])
                                                                                -
{counter_in}))
                bottom_list =
generation[random_num_excluding_counter_in]

                crossed_mutated =
get_crossed_mutated_lists(search_space, top_list, bottom_list,

best_list_of_last_gen, minimum_size,

maximum_size)
                cross_mutated_list.append(crossed_mutated[0])
                cross_mutated_list.append(crossed_mutated[1])

                counter_in += 1

            # ************ Random end of crossing and mutations

            # create a temp_generation by merging init_top_best with the
crossed
            temp_generation = merge_to_list_of_lists_distinct(top_best,
cross_mutated_list)

            # merge a temp_generation with the leftovers
            temp_generation =
merge_to_list_of_lists_distinct(temp_generation, left_overs)

            generation = get_top_best_in_value_of_size(temp_generation,
max_times)

            # add the generation to list of generations
            generations.append(generation)

            value_of_prev_gen = value_of_last_gen
            value_of_last_gen = get_value_of_generation(generation)

            counter += 1

        average_values = []
        total_values = []
        for generation in generations:

            # write_results_to_file(generation, generation_lists_file)
            total_value = float('0')
```

```
            for list_0 in generation:
                # get the value of the list
                value_of_list = get_blocks_value(list_0)

                total_value += value_of_list

            total_values.append(total_value)
            write_results_to_file_append(str(total_values),
generation_lists_file)

            average_value = total_value / float(len(generation))
            # print('-------------------------------------------')

            average_values.append(average_value)

        write_results_to_file_append('\n-----------------\n',
generation_lists_file)

        counter = 0
        while counter < len(average_values):
            print("Average Value for generation: %s is: %s USD" %
(counter, average_values[counter]))

            counter += 1

        counter = 0
        while counter < len(generations):
            generation = generations[counter]
            # get the best list [. i.e ]the first in the list]
            best_list = generation[0]
            best_list_value = get_blocks_value(best_list)

            print("Best list for generation %s is: \n%s \n has length of
%s blocks and value of : %s USD " %
                  (counter, best_list, len(best_list), best_list_value))
            print("has total tonnage of: %s tonnes and ore tonnage of :
%s tonnes "
                  % (get_total_tonnage_value(best_list),
get_ore_tonnage_value(best_list)))
            print('------------- Generation separator -----------------
')
            counter += 1

        # best list is at last generation at the top
        last_gen = generations[len(generations) - 1]

        best_list_of_last_gen = last_gen[0]

        # add the best list from a time period
        time_period_data.append(best_list_of_last_gen)

        for b in best_list_of_last_gen:
            # TODO to be removed, was put for checking purposes.
            if b not in search_space:
                print("-------------b not in search_space--------------
")

            used_best_list_of_last_gen.append(b)

        # remove the items in best list of last gen from the search
```

```
space
        left_overs_from_elevated_search_space =
remove_list_items(search_space, best_list_of_last_gen)
        # list_outside_elevation = remove_list_items(search_space_main,
new_search_from_elevation)
        # left_search_space =
merge_to_list_distinct(list_outside_elevation,
left_overs_from_elevated_search_space)

        # lowest_elevation_allowed = topography_elevation -
(start_elevation + 4)
        # new_search_from_elevation =
get_search_space_from_elevation(left_search_space,
lowest_elevation_allowed)
        # list_outside_elevation = remove_list_items(left_search_space,
new_search_from_elevation)
        #
        # while len(new_search_from_elevation) < maximum_size and
len(left_search_space) >= maximum_size:
        #     new search from elevation =
get search space from elevation(left search space,
(lowest_elevation_allowed - 4))
        #     list_outside_elevation =
remove_list_items(left_search_space, new_search_from_elevation)

        updated_global_search_space =
remove_list_items(updated_global_search_space, best_list_of_last_gen)
        search_space = left_overs_from_elevated_search_space

        while len(search_space) < maximum_size and len(search_space) <
len(updated_global_search_space):
            start_elevation = start_elevation + 1
            lowest_elevation_allowed = topography_elevation -
start_elevation
            new_search_from_elevation =
get_search_space_from_elevation(updated_global_search_space,

lowest_elevation_allowed)
            search_space = new_search_from_elevation

        # list_outside_elevation =
remove_list_items(updated_global_search_space,
new search from elevation)

        # search_space = left_search_space

        time_period_counter += 1
        print("Best list for time period %s is: \n%s \n has length of %s
blocks and value of : %s USD " %
              (time_period_counter, best_list_of_last_gen,
               len(best_list_of_last_gen),
               get_blocks_value(best_list_of_last_gen)))

        print("has total tonnage of: %s tonnes and ore tonnage of : %s
tonnes " %
              (get_total_tonnage_value(best_list_of_last_gen),
               get_ore_tonnage_value(best_list_of_last_gen)))
        print('---------------------------------- Period separator ----
----------------------------------')
        # start Generation lists with fitness value
```

```
        for generation in generations:
            fitness_with_blocks = []
            fitnesses = []

            for gen_int in generation:

                fitness = get_blocks_value(gen_int)
                fitnesses.append(fitness)

            # fitness_with_blocks.append(fitness)
            # fitness_with_blocks.append(generation)
            generation_fitness_value.append(fitnesses)

            # generation_fitness_value.append(fitness_with_blocks)
        # end  Generation lists with fitness value
        generation_fitness_value.append([])

    generation_fitness_value.append([get_blocks_value(search_space)])
    write_results_to_file(generation_fitness_value,
periods_fitness_file_name)
    # write results to file append("----", "generations result 0.csv")

    # add the best list from the last search space
    time_period_data.append(search_space)

    print("Best list for the last time period: %s is: \n%s \n has length
of: %s blocks, total tonnage of: %s "
          "tonnes, ore tonnage of : %s "
          "tonnes and value of: %s USD" %
          (time_period_counter + 1, search_space, len(search_space),
get_total_tonnage_value(search_space),
           get_ore_tonnage_value(search_space),
           get_blocks_value(search_space)))

    return time_period_data


def convert_line_to_blocks(f_line):
    list_0 = []
    f_line_blocks = f_line.split(",")
    for block in f_line_blocks:
        list_0.append(int(block))

    return list_0


def generate_search_space_for_line(f_line, minimum_size_int,
maximum_size_int, start_elevation, generation_lists_file):
    list_0 = convert_line_to_blocks(f_line)

    return generate_time_periods_data(start_elevation, list_0,
minimum_size_int, maximum_size_int,
                                      generation_lists_file)
```

8.2 Python code to input data and run the combined mixed integer programming model and GA approach for integrating long-term and medium-term production schedules

```
import csv
import math
from pathlib import Path
from tkinter import *
import newman_common as nw_commons
from tkinter import ttk, filedialog
import tkinter as tk
import time
import ps_newman_common as plotting

window = Tk()

window.title("GA-BASED ALGORITHM FOR A DYNAMIC LONG- AND MEDIUM-TERMS
INTEGRATED OPEN-PIT MINE PRODUCTION SCHEDULING SYSTEM")
window.geometry("600x400")

tab_control = ttk.Notebook(window)

# Creating the tabs
model_input_tab = ttk.Frame(tab_control)
scheduling_parameters_tab = ttk.Frame(tab_control)
decision_variables_tab = ttk.Frame(tab_control)
scheduling_result_tab = ttk.Frame(tab_control)
# output tab = ttk.Frame(tab control)

# Naming the tabs
tab_control.add(model_input_tab, text='Model Data Input')
tab_control.add(scheduling_parameters_tab, text='Scheduling Constraint
Parameters')
tab_control.add(decision_variables_tab, text='Scheduling Decision
Variables')
tab_control.add(scheduling_result_tab, text='Scheduling Results')
# tab_control.add(output_tab, text='Output')
tab_control.pack(expand=1, fill='both')

csv_file_extension = ".csv"
block_model_file_extension = csv_file_extension
block_precedence_file_extension = csv_file_extension

# SP data variables
sp_terms_data = [None] * 2
mining_bench_lead = [0] * 3
sp_current_selected_mine_tab = 0

# Long term data
lt_output_file_name = "blocks_period_lt.csv"
lt_generation_lists_file = "long_term_generation_lists.csv"
lt_periods_fitness_file_name = "lt_periods_fitness.csv"

# Medium term 6 months data
mt_6_month_output_file_name = "blocks_period_mt_6_month.csv"
mt_6_month_generation_lists_file = "mt_6_month_generation_lists.csv"
mt_6_periods_fitness_file_name = "mt_6_periods_fitness.csv"
```

```
nw_commons.initialize_all_data()


def generate_lt_initial_search_space():
    list_0 = []

    for line_0 in nw_commons.preceds_lines:
        list_0.append(int(line_0[0]))

    return list_0


# MC start
def get_lt_minimum_size_mc():
    min_max_mc = sp_terms_data[0]
    min_mc = min_max_mc[0]
    return min_mc[0]


def get_lt_maximum_size_mc():
    min_max_mc = sp_terms_data[0]
    max_mc = min_max_mc[0]
    return max_mc[1]


def get_mt_6_months_minimum_size_mc():
    min_max_mc = sp_terms_data[1]
    min_mc = min_max_mc[0]
    return min_mc[0]


def get_mt_6_months_maximum_size_mc():
    min_max_mc = sp_terms_data[1]
    max_mc = min_max_mc[0]
    return max_mc[1]
# MC end


# PC start
def get_lt_minimum_size_pc():
    min_max_pc = sp_terms_data[0]
    min_pc = min_max_pc[1]
    return min_pc[0]


def get_lt_maximum_size_pc():
    min_max_pc = sp_terms_data[0]
    max_pc = min_max_pc[1]
    return max_pc[1]


def get_mt_6_months_minimum_size_pc():
    min_max_pc = sp_terms_data[1]
    min_pc = min_max_pc[1]
    return min_pc[0]


def get_mt_6_months_maximum_size_pc():
    min_max_pc = sp_terms_data[1]
```

```
    max_pc = min_max_pc[1]
    return max_pc[1]
# PC end


# AG start
def get_minimum_ag():
    min_max_ag = sp_terms_data[sp_current_selected_mine_tab]
    min_ag = min_max_ag[2]
    return min_ag[0]


def get_maximum_av():
    min_max_ag = sp_terms_data[sp_current_selected_mine_tab]
    max_ag = min_max_ag[2]
    return max_ag[1]
# AG end


def get_lt_mining_bench_lead():
    return mining_bench_lead[0]


def get_mt_6_months_mining_bench_lead():
    return mining_bench_lead[1]


def run_long_term():

nw_commons.set_periods_fitness_file_name(lt_periods_fitness_file_name)
    start = time.time()
    lt_minimum_size = get_lt_minimum_size_mc()
    lt_maximum_size = get_lt_maximum_size_mc()
    lt_mining_bench_lead = get_lt_mining_bench_lead()

    search_space = generate_lt_initial_search_space()
    data = nw_commons.generate_time_periods_data(
        lt_mining_bench_lead,
        search_space,
        lt_minimum_size,
        lt_maximum_size,
        lt_generation_lists_file
    )
    nw_commons.write_results_to_file(data, lt_output_file_name)

    end = time.time()
    print(
        "Total time it took to finish term list of size: %s is: %s
seconds " % (nw_commons.max_times, end - start))

    # Update plotting button after a successful term run
    plotting_lt_button["state"] = NORMAL

    # Enable mt 6 months
    run_mt6_button["state"] = NORMAL


def generate_mt_6_months_search_space_for_time_period(f_line):
    blocks_list = nw_commons.convert_line_to_blocks(f_line)
    # counter = 0
    # number_of_resets = 0
```

```
    lt_minimum_size = get_lt_minimum_size_mc()
    lt_maximum_size = get_lt_maximum_size_mc()
    minimum_size_int = (lt_minimum_size / 2)
    maximum_size_int = (lt_maximum_size / 2)
    lt_mining_bench_lead = get_mt_6_months_mining_bench_lead()

    data = nw_commons.generate_search_space_for_line(f_line,
minimum_size_int, maximum_size_int, lt_mining_bench_lead,

mt_6_month_generation_lists_file)

    while len(data) != 2 and len(blocks_list) >= minimum_size_int:
        # counter += 1
        maximum_size_int = (lt_maximum_size / 2) + ((lt_maximum_size -
lt_minimum_size) / 4)
        minimum_size_int = (lt_minimum_size / 2) + ((lt_maximum_size -
lt_minimum_size) / 4)
        data = nw_commons.generate_search_space_for_line(f_line,
minimum_size_int, maximum_size_int,

lt_mining_bench_lead,

mt_6_month_generation_lists_file)
        # if maximum_size_int > len(blocks_list):
        #     number of resets = number of resets + 1
        #     counter = -1

    output = data

    return output


def run_mt_6_months_term():

nw_commons.set_periods_fitness_file_name(mt_6_periods_fitness_file_name)
    start = time.time()

    with open(lt_output_file_name) as f:
        lines = f.readlines()

    lines_data = []

    for line in lines:
        line_data =
generate_mt_6_months_search_space_for_time_period(line)
        lines_data.extend(line_data)

    nw_commons.write_results_to_file(lines_data,
mt_6_month_output_file_name)

    end = time.time()
    print(
        "Total time it took to finish term list of size: %s is: %s
seconds " % (nw_commons.max_times, end - start))

    # Update plotting button after a successful term run
    plotting_mt6_button["state"] = NORMAL


def browse(file_extension: str, allow_other_files_option=False):
    one_file_type = (file_extension + " files", "*" + file_extension +
```

```
"*")
    ext_prep = "*" + file_extension + "*"
    file_extensions = (file_extension + " files", ext_prep) \
        if allow_other_files_option \
        else (one_file_type, ("All Files", "*.*"))

    f_path = filedialog.askopenfilename(
        initialdir="./",
        title="Select File",
        filetypes=file_extensions
    )
    return f_path


def launch_scheduling_parameters_dialog():
    print("launch_scheduling_parameters_dialog")


# Upload Block Model CSV File button start
block_model_frame = ttk.LabelFrame(model_input_tab, text="Block Model")
block_model_frame.grid(column=0, row=0, sticky='w', padx=5, pady=5)


def temp_auto_populate_files():
    block_model_file_name_var.set('C:/Users/pme22/Documents/My PhD
Research 2021/My_Research/newmann1_model/newmann_main.csv')
    nw_commons.set_coordinates_file_path('C:/Users/pme22/Documents/My
PhD Research 2021/My_Research/newmann1_model/newmann_main.csv')

    block_precedence_var.set('C:/Users/pme22/Documents/My PhD Research
2021/My_Research/newmann1_model/newmann_preceds.csv')
    nw_commons.set_preceds_file_path('C:/Users/pme22/Documents/My PhD
Research 2021/My_Research/newmann1_model/newmann_preceds.csv')

    block_economic_value_var.set('C:/Users/pme22/Documents/My PhD
Research 2021/My_Research/newmann1_model/newmann_pit.csv')

nw_commons.set_block_economic_value_file('C:/Users/pme22/Documents/My
PhD Research 2021/My_Research/newmann1_model/newmann_pit.csv')

    sp_update_values()
    sp_select_mine_term(0)
    run_long_term()


def upload_block_model_file():
    file_path = browse(block_model_file_extension)
    block_model_file_name_var.set(file_path)
    nw_commons.set_coordinates_file_path(file_path)

    # Enable/Disable buttons for run
    check_data_ready_to_be_processed()


ttk.Button(block_model_frame, text="Upload Block Model CSV File",
command=upload_block_model_file) \
    .grid(column=0, row=0, sticky='w')

block_model_file_name_var = StringVar()
block_model_file_name_var.set("Empty")
```

```
block_model_file_name_label = ttk.Label(block_model_frame,
textvariable=block_model_file_name_var)
block_model_file_name_label.grid(column=0, row=1, sticky='w')
# Upload Block Model CSV File button end


# Upload Block Precedence CSV File button start
def upload_block_precedence_file():
    file_path = browse(block_precedence_file_extension)
    block_precedence_var.set(file_path)
    nw_commons.set_preceds_file_path(file_path)

    # Enable/Disable buttons for run
    check_data_ready_to_be_processed()


block_precedence_frame = ttk.LabelFrame(model_input_tab, text="Block
Precedences")
block_precedence_frame.grid(column=0, row=1, sticky='w', padx=5, pady=5)

ttk.Button(block_precedence_frame, text="Upload Block Precedences CSV
File", command=upload_block_precedence_file) \
    .grid(column=0, row=0, sticky='w')

block_precedence_var = StringVar()
block_precedence_var.set("Empty")

block_precedence_file_name_label = ttk.Label(block_precedence_frame,
textvariable=block_precedence_var)
block_precedence_file_name_label.grid(column=0, row=1, sticky='w')
# Upload Block Precedence CSV File button end


# Upload block economic value CSV File button start
def upload_block_economic_value_file():
    file_path = browse(block_precedence_file_extension)
    block_economic_value_var.set(file_path)
    nw_commons.set_block_economic_value_file(file_path)

    # Enable/Disable buttons for run
    check_data_ready_to_be_processed()


block_economic_value_frame = ttk.LabelFrame(model_input_tab, text="Block
Economic Value")
block_economic_value_frame.grid(column=0, row=2, sticky='w', padx=5,
pady=5)

ttk.Button(block_economic_value_frame, text="Upload Block Economic Value
CSV File",
           command=upload_block_economic_value_file) \
    .grid(column=0, row=0, sticky='w')

block_economic_value_var = StringVar()
block_economic_value_var.set("Empty")

block_economic_value_file_name_label =
ttk.Label(block_economic_value_frame,
textvariable=block_economic_value_var)
block_economic_value_file_name_label.grid(column=0, row=1, sticky='w')
# Upload block_economic_value CSV File button end
```

```
# Run Scenario start
run_terms_frame = ttk.LabelFrame(model_input_tab, text="Run Scheduling
Scenario")
run_terms_frame.grid(column=0, row=3, sticky='w', padx=5, pady=5)

run_lt_button = ttk.Button(run_terms_frame, text="Run Yearly Long-Term
Scheduling", command=run_long_term)
run_lt_button.grid(column=0, row=0, sticky='w')

run_mt6_button = ttk.Button(run_terms_frame, text="Run Half-Yearly
Medium-Term Scheduling", command=run_mt_6_months_term)
run_mt6_button.grid(column=1, row=0, sticky='w')

# initially disable the button, they will be enabled when data is ready
run_lt_button["state"] = DISABLED
run_mt6_button["state"] = DISABLED


def is_mining_capacity_values_set():
    mc_minimum = get_lt_minimum_size_mc()
    mc_maximum = get_lt_maximum_size_mc()

    if mc_minimum == 0 or mc_maximum == 0 or mc_minimum >= mc_maximum:
        return False

    return True


def are_file_paths_set():
    if (not nw_commons.preceds_file) or (not
nw_commons.block_economic_value_file) or (not
nw_commons.coordinates_file):
        return False
    return True


def check_data_ready_to_be_processed():

    files_uploaded = are_file_paths_set()
    values_set = is_mining_capacity_values_set()

    if files_uploaded and values_set:
        run_lt_button["state"] = NORMAL
    else:
        run_lt_button["state"] = DISABLED

# Run Scenario end


# Plot terms start
def plot_long_term():
    print("plot_long_term")

    schedule_stage = "Yearly Long-Term Schedule"

    with open(lt_output_file_name, newline='') as cvs_blocks_file:
        blocks_line = list(csv.reader(cvs_blocks_file, delimiter=','))
        split_size = len(blocks_line)
```

```
    plotting.plot_phases(lt_output_file_name, schedule_stage,
split_size)


def plot_mt_6_months_term():
    print("plot_mt_6_months_term")
    schedule_stage = "Half-Yearly Medium-Term Schedule"

    with open(mt_6_month_output_file_name, newline='') as
cvs_blocks_file:
        blocks_line = list(csv.reader(cvs_blocks_file, delimiter=','))
        split_size = len(blocks_line)

    plotting.plot_phases(mt_6_month_output_file_name, schedule_stage,
split_size)


plotting_terms_frame = ttk.LabelFrame(model_input_tab, text="Plotting")
plotting_terms_frame.grid(column=0, row=4, sticky='w', padx=5, pady=5)

plotting_lt_button = ttk.Button(plotting_terms_frame, text="Plot Yearly
Long-Term Schedule", command=plot_long_term)
plotting_lt_button.grid(column=0, row=0, sticky='w')

plotting_mt6_button = ttk.Button(plotting_terms_frame, text="Plot Half-
Yearly Medium-Term Schedule", command=plot_mt_6_months_term)
plotting_mt6_button.grid(column=1, row=0, sticky='w')


# initially disable the buttons, they will be enabled when data is ready
plotting_lt_button["state"] = DISABLED
plotting_mt6_button["state"] = DISABLED
# Plot terms end


# ************ Start of sp (scheduling parameters) **********
def sp_initialize_terms_data():
    for i, x in enumerate(sp_terms_data):
        sp_terms_data[i] = [(0, 0), (0, 0), (0, 0)]


# Call function to initialize data
sp_initialize_terms_data()

# Input scheduling parameters
sp_lt_input_bt = ttk.Button(scheduling_parameters_tab, text="Long-term",
command=lambda: sp_select_mine_term(0))
sp_lt_input_bt.grid(column=0, row=0, sticky='ew')
sp_mt6_input_bt = ttk.Button(scheduling_parameters_tab, text="Medium-
term 6 month",
                             command=lambda: sp_select_mine_term(1))
sp_mt6_input_bt.grid(column=1, row=0, sticky='ew')

def sp_add_frame(label, index=0):
    print("launch_scheduling_parameters_dialog")

    frame_one = ttk.LabelFrame(scheduling_parameters_tab, text=label)
    frame_one.grid(column=0, row=4 + index, sticky='w', padx=5, pady=5)

    ttk.Label(frame_one, text="Minimum").grid(column=0, row=0,
sticky='w')
```

```
    ttk.Label(frame_one, text="Maximum").grid(column=0, row=1,
sticky='w')

    return frame_one


def sp_update_values():
    term_index = sp_current_selected_mine_tab

    min_mc = sp_mining_capacity_min_entry.get()
    max_mc = sp_mining_capacity_max_entry.get()
    min_mc_int = int(min_mc)
    max_mc_int = int(max_mc)

    # TODO Remove these
    # min_mc_int = int(250)
    # max_mc_int = int(350)

    min_pc = sp_processing_capacity_min_entry.get()
    max_pc = sp_processing_capacity_max_entry.get()
    min_pc_int = int(min_pc)
    max_pc_int = int(max_pc)

    min_av = sp_average_grade_min_entry.get()
    max_av = sp_average_grade_max_entry.get()
    min_av_int = float(min_av)
    max_av_int = float(max_av)

    sp_terms_data[term_index] = [(min_mc_int, max_mc_int), (min_pc_int,
max_pc_int), (min_av_int, max_av_int)]

    mining_bench_lead_str = mining_bench_lead_entry.get()
    mining_bench_lead_int = int(mining_bench_lead_str)

    if term_index == 0:
        # MC start
        mc_minimum_size_int_mt_6 = (get_lt_minimum_size_mc() / 2)
        mc_maximum_size_int_mt_6 = (get_lt_maximum_size_mc() / 2)
        sp_terms_data[1][0] = (round_down(mc_minimum_size_int_mt_6),
round_up(mc_maximum_size_int_mt_6))
        # MC end

        # PC start
        pc_minimum_size_int_mt_6 = (get_lt_minimum_size_pc() / 2)
        pc_maximum_size_int_mt_6 = (get_lt_maximum_size_pc() / 2)
        sp_terms_data[1][1] = (round_down(pc_minimum_size_int_mt_6),
round_up(pc_maximum_size_int_mt_6))
        # PC end

    if term_index == 1:

        # MC start
        mc_minimum_size_int_lt = (get_mt_6_months_minimum_size_mc() * 2)
        mc_maximum_size_int_lt = (get_mt_6_months_maximum_size_mc() * 2)
        sp_terms_data[0][0] = (round_down(mc_minimum_size_int_lt),
round_up(mc_maximum_size_int_lt))
        # MC End

        # PC start
        pc_minimum_size_int_lt = (get_mt_6_months_minimum_size_pc() * 2)
        pc_maximum_size_int_lt = (get_mt_6_months_maximum_size_pc() * 2)
```

```
        sp_terms_data[0][1] = (round_down(pc_minimum_size_int_lt),
round_up(pc_maximum_size_int_lt))
        # PC End

    # AG start
    ag_minimum_size_int = get_minimum_ag()
    ag_maximum_size_int = get_maximum_av()

    sp_terms_data[0][2] = (ag_minimum_size_int, ag_maximum_size_int)
    sp_terms_data[1][2] = (ag_minimum_size_int, ag_maximum_size_int)
    # AG end

    mining_bench_lead[term_index] = mining_bench_lead_int

    # Enable/Disable buttons for run
    check_data_ready_to_be_processed()

    print("sp_terms_data:", sp_terms_data)


def round_up(num):
    return math.ceil(num)


def round_down(num):
    return math.floor(num)


def sp_add_entry_to_frame(frame, entry_column_row):
    entry = ttk.Entry(frame)
    entry.grid(column=1, row=entry_column_row, sticky='w')
    return entry


sp_mining_capacity_frame = sp_add_frame("Mining Capacity", 0)
sp_mining_capacity_min_entry =
sp_add_entry_to_frame(sp_mining_capacity_frame, 0)
sp_mining_capacity_max_entry =
sp_add_entry_to_frame(sp_mining_capacity_frame, 1)

sp_processing_capacity_frame = sp_add_frame("Processing Capacity", 1)
sp_processing_capacity_min_entry =
sp_add_entry_to_frame(sp_processing_capacity_frame, 0)
sp_processing_capacity_max_entry =
sp_add_entry_to_frame(sp_processing_capacity_frame, 1)

sp_average_grade_frame = sp_add_frame("Average grade", 2)
sp_average_grade_min_entry =
sp_add_entry_to_frame(sp_average_grade_frame, 0)
sp_average_grade_max_entry =
sp_add_entry_to_frame(sp_average_grade_frame, 1)

mining_bench_lead_frame = ttk.LabelFrame(scheduling_parameters_tab,
text="Mining Bench Lead")
mining_bench_lead_frame.grid(column=0, row=8, sticky='w', padx=5,
pady=5)

mining_bench_lead_entry = sp_add_entry_to_frame(mining_bench_lead_frame,
0)
```

```
def sp_select_mine_term(index):
    global sp_current_selected_mine_tab
    sp_current_selected_mine_tab = index

    sp_populate_values()


def sp_populate_values():
    term = sp_terms_data[sp_current_selected_mine_tab]
    print("sp_current_selected_mine_tab[0]:",
sp_current_selected_mine_tab)

    min_max_mc = term[0]
    min_mc = min_max_mc[0]
    max_mc = min_max_mc[1]

    sp_mining_capacity_min_entry.delete(0, "end")
    sp_mining_capacity_min_entry.insert(-1, str(min_mc))
    sp_mining_capacity_max_entry.delete(0, "end")
    sp_mining_capacity_max_entry.insert(-1, str(max_mc))

    min_max_pc = term[1]
    min_pc = min_max_pc[0]
    max_pc = min_max_pc[1]

    sp_processing_capacity_min_entry.delete(0, "end")
    sp_processing_capacity_min_entry.insert(-1, str(min_pc))
    sp_processing_capacity_max_entry.delete(0, "end")
    sp_processing_capacity_max_entry.insert(-1, str(max_pc))

    min_max_av = term[2]
    min_av = min_max_av[0]
    max_av = min_max_av[1]

    sp_average_grade_min_entry.delete(0, "end")
    sp_average_grade_min_entry.insert(-1, str(min_av))
    sp_average_grade_max_entry.delete(0, "end")
    sp_average_grade_max_entry.insert(-1, str(max_av))

    mining_bench_lead_entry.delete(0, "end")
    mining_bench_lead_entry.insert(-1,
str(mining_bench_lead[sp_current_selected_mine_tab]))


sp_populate_values()

# TODO, might not be needed: Populate this data to the table, depending
on which term is currently selected.
sp_update_values_for_term_bt = ttk.Button(scheduling_parameters_tab,
text="Update values",
                                          command=lambda:
sp_update_values())
sp_update_values_for_term_bt.grid(column=0, row=9, sticky='w')

# ************ End of sp (scheduling parameters) **********

# ************ Start of dv (Decision Variables) **********

# 1
dv_scheduling_horizon_fr = ttk.LabelFrame(decision_variables_tab,
text="Scheduling Horizon")
```

```
dv_scheduling_horizon_fr.grid(column=0, row=0, sticky='w', padx=5,
pady=5)

ttk.Label(dv_scheduling_horizon_fr, text="Long-term
horizon").grid(column=0, row=0, sticky='w')
ttk.Label(dv_scheduling_horizon_fr, text="Medium-term
horizon").grid(column=0, row=1, sticky='w')

dv_lt_horizon_entry = ttk.Entry(dv_scheduling_horizon_fr)
dv_lt_horizon_entry.grid(column=1, row=0, sticky='w')

dv_lt_horizon_entry = ttk.Entry(dv_scheduling_horizon_fr)
dv_lt_horizon_entry.grid(column=1, row=1, sticky='w')

# 2
dv_scheduling_period_length_fr = ttk.LabelFrame(decision_variables_tab,
text="Scheduling Period length")
dv_scheduling_period_length_fr.grid(column=0, row=1, sticky='w', padx=5,
pady=5)

ttk.Label(dv_scheduling_period_length_fr, text="Yearly Long-
term").grid(column=0, row=0, sticky='w')
dv_scheduling_period_length_yearly_lt_entry =
ttk.Entry(dv_scheduling_period_length_fr)
dv_scheduling_period_length_yearly_lt_entry.grid(column=1, row=0,
sticky='w')

ttk.Label(dv_scheduling_period_length_fr, text="Half-Yearly Medium-
term").grid(column=0, row=1, sticky='w')
dv_scheduling_period_length_half_yearly_mt_entry =
ttk.Entry(dv_scheduling_period_length_fr)
dv_scheduling_period_length_half_yearly_mt_entry.grid(column=1, row=1,
sticky='w')


# ************ Start of output **********
output_schedule_data = []
output_time_period_data = []
output_headers = ["", "Value"]


def output_get_table_data():
    table_data = []
    for row in range(1, len(output_schedule_data) + 1):
        row_data = []
        for col in range(len(output_headers)):
            entry = window.grid_slaves(row=row, column=col)[0]
            row_data.append(entry.get())
        table_data.append(row_data)
    return table_data


output_selected_mine_tab = 0
output_selected_period_result_index = 0


def output_populate_mine_term_data():
    print("output_populate_mine_term_data")
    # TODO, populate the scheduling stage results with term data
    #  and populate period results with first period number
    output_populate_schedule_data()
```

```
    clear_frame(output_scheduling_stage_results_frame)
    output_draw_scheduling_stage_results_header()
    output_draw_scheduling_stage_results()

    global output_selected_period_result_index
    output_selected_period_result_index = 0
    output_populate_period_results_for_period_index()


def output_populate_period_results_for_period_index():
    clear_frame(output_period_results_frame)

    output_populate_period_number_data()
    output_create_selection_of_periods()

    output_draw_scheduling_period_results()


def output_select_mine_term(index):
    global output_selected_mine_tab
    output_selected_mine_tab = index

    output_populate_mine_term_data()


# Input scheduling parameters
output_lt_bt = ttk.Button(scheduling_result_tab, text="Long Term Yearly
Schedule",
                          command=lambda: output_select_mine_term(0))
output_lt_bt.grid(column=0, row=0, sticky='ew')
output_mt6_input_bt = ttk.Button(scheduling_result_tab, text="Medium-
term 6 month Schedule",
                                 command=lambda:
output_select_mine_term(1))
output_mt6_input_bt.grid(column=1, row=0, sticky='ew')

output_scheduling_stage_results_frame =
ttk.LabelFrame(scheduling_result_tab, text="Scheduling Stage Results")
output_scheduling_stage_results_frame.grid(column=0, row=1, sticky='w',
padx=5, pady=5)


def get_current_output_file():
    output_file = lt_output_file_name

    if output_selected_mine_tab == 1:
        output_file = mt_6_month_output_file_name

    return output_file


discount_rate = 0.08


def calculate_npv_lt(list_of_block_list):
    sum_pv = 0.0

    for index, blocks in enumerate(list_of_block_list):
        cashflow_blocks_list = nw_commons.get_blocks_value(blocks)
        i = index + 1
        sum_pv += cashflow_blocks_list / ((1 + discount_rate) ** i)
```

```
    return sum_pv


def calculate_npv_mt_6_months(list_of_block_list):
    sum_pv = 0.0

    for index, blocks in enumerate(list_of_block_list):
        cashflow_blocks_list = nw_commons.get_blocks_value(blocks)
        i = index + 1
        sum_pv += cashflow_blocks_list / ((1 + discount_rate/2) ** i)

    return sum_pv


def convert_output_file_list_of_lists(output_file):
    with open(output_file) as f:
        lines = f.readlines()

    lines_data = []
    for line in lines:
        list_0 = []
        blocks = line.split(",")
        for block in blocks:
            list_0.append(int(block))
        lines_data.append(list_0)

    return lines_data


def calculate_npv():
    output_file = get_current_output_file()
    list_of_block_list = convert_output_file_list_of_lists(output_file)
    npv = 0.0

    if output_selected_mine_tab == 0:
        npv = calculate_npv_lt(list_of_block_list)
    if output_selected_mine_tab == 1:
        npv = calculate_npv_mt_6_months(list_of_block_list)

    return format(round(npv, 2), ",")


index_of_block_cost = 8
index_of_block_tonnage = 6


def calculate_total_cost_of_blocks(blocks):
    total_cost = 0.0

    for block in blocks:
        coordinate_line = nw_commons.get_coordinate_line(block)
        block_cost = float(coordinate_line[index_of_block_cost])
        # print("calculate_total_cost_of_blocks")
        total_cost += block_cost

    return total_cost


def output_calculate_total_cost():
    output_file = get_current_output_file()
```

```
    list_of_block_list = convert_output_file_list_of_lists(output_file)
    total_cost = 0.0
    for line in list_of_block_list:
        total_cost += calculate_total_cost_of_blocks(line)

    return format(round(abs(total_cost), 2), ",")


def output_populate_schedule_data():
    output_file = get_current_output_file()
    num_periods = sum(1 for _ in open(output_file))
    output_schedule_data.clear()
    output_schedule_data.append(["Total number of periods",
str(num_periods)])
    # Add total number of period
    output_schedule_data.append(["NPV", str(calculate_npv())])
    # Add total number of period
    output_schedule_data.append(["Total cost",
str(output_calculate_total_cost())])


# output_populate_schedule_data()


def clear_frame(frame):
    # destroy all widgets from frame
    for widget in frame.winfo_children():
        widget.destroy()


def output_draw_scheduling_stage_results_header():
    for col, header in enumerate(["", "Value"]):
        label = ttk.Label(output_scheduling_stage_results_frame,
text=header, font=("Arial", 12, "bold"))
        label.grid(row=0, column=col, padx=10, pady=5, sticky='w')


def output_draw_scheduling_stage_results():
    for row, row_data in enumerate(output_schedule_data, start=1):
        for col, value in enumerate(row_data):
            label = tk.Label(output_scheduling_stage_results_frame,
text=value, font = ('Times', 20), relief = GROOVE)
            label.grid(row=row, column=col, padx=10, pady=5, sticky='w')


# output_populate_mine_term_data()
def output_period_get_num_blocks():
    output_file = get_current_output_file()
    list_of_lists = convert_output_file_list_of_lists(output_file)
    blocks = list_of_lists[output_selected_period_result_index]

    return len(blocks)


def output_period_get_volume():
    # TODO

    return 0.000000


def calculate_tonnage_of_blocks(blocks):
```

```
    total = 0.0

    for block in blocks:
        coordinate_line = nw_commons.get_coordinate_line(block)
        block_tonnage = float(coordinate_line[index_of_block_tonnage])
        total += block_tonnage

    return total


def output_period_get_tonnage():
    output_file = get_current_output_file()
    list_of_lists = convert_output_file_list_of_lists(output_file)
    blocks = list_of_lists[output_selected_period_result_index]
    blocks_tonnage = calculate_tonnage_of_blocks(blocks)

    return format(round(blocks_tonnage, 2), ",")


def output_period_get_ore_tonnage():
    output_file = get_current_output_file()
    list_of_lists = convert_output_file_list_of_lists(output_file)
    blocks = list_of_lists[output_selected_period_result_index]

    ore_blocks = nw_commons.get_ore_blocks(blocks)
    tonnage = 0.0

    for block in ore_blocks:
        coordinate_line = nw_commons.get_coordinate_line(block)
        block_tonnage = float(coordinate_line[index_of_block_tonnage])
        tonnage += block_tonnage

    return format(round(tonnage, 2), ",")


def output_period_get_waste_tonnage():
    output_file = get_current_output_file()
    list_of_lists = convert_output_file_list_of_lists(output_file)
    blocks = list_of_lists[output_selected_period_result_index]

    waste_blocks = nw_commons.get_waste_blocks(blocks)
    tonnage = 0.0

    for block in waste_blocks:
        coordinate_line = nw_commons.get_coordinate_line(block)
        block_tonnage = float(coordinate_line[index_of_block_tonnage])
        tonnage += block_tonnage

    return format(round(tonnage, 2), ",")


def calculate_economic_value_of_blocks(blocks):
    ore_blocks = nw_commons.get_ore_blocks(blocks)
    blocks_value = nw_commons.get_blocks_value(blocks)

    return blocks_value


def output_period_get_economic_value():
    output_file = get_current_output_file()
    list_of_lists = convert_output_file_list_of_lists(output_file)
```

```
    blocks = list_of_lists[output_selected_period_result_index]
    economic_value = calculate_economic_value_of_blocks(blocks)

    return format(round(economic_value, 2), ",")


def output_period_get_mining_cost():
    output_file = get_current_output_file()
    list_of_lists = convert_output_file_list_of_lists(output_file)
    blocks = list_of_lists[output_selected_period_result_index]

    total_cost = calculate_total_cost_of_blocks(blocks)

    return format(round(abs(total_cost), 2), ",")


def output_period_get_strip_ratio():
    if output_period_get_ore_tonnage() == '0.0':
        return '-.-'
    else:
        strip_ratio =
float(output_period_get_waste_tonnage().replace(',', '')) /
float(output_period_get_ore_tonnage().replace(',', ''))
        return round(strip_ratio, 2)


def output_populate_period_number_data():

    # extract this to be the header
    # output time period data.append(["Period Number", "4"])
    output_time_period_data.clear()
    output_time_period_data.append(["Number of blocks",
str(output_period_get_num_blocks())])
    output_time_period_data.append(["Volume",
str(output_period_get_volume())])
    output_time_period_data.append(["Tonnage",
str(output_period_get_tonnage())])
    output_time_period_data.append(["Ore Tonnage",
str(output_period_get_ore_tonnage())])
    output_time_period_data.append(["Waste Tonnage",
str(output_period_get_waste_tonnage())])
    output_time_period_data.append(["Strip Ratio",
str(output_period_get_strip_ratio())])
    output_time_period_data.append(["Economic Value",
str(output_period_get_economic_value())])
    output_time_period_data.append(["Mining Cost",
str(output_period_get_mining_cost())])


output_period_results_frame = ttk.LabelFrame(scheduling_result_tab,
text="Period Results")
output_period_results_frame.grid(column=0, row=2, sticky='w', padx=5,
pady=5)


def output_period_result_select(event):
    period = event.widget.get()
    period_int = int(period)
    print(period_int)

    global output_selected_period_result_index
```

```
    output_selected_period_result_index = period_int - 1
    output_populate_period_results_for_period_index()


def output_create_selection_of_periods():

    ttk.Label(output_period_results_frame, text="Period Number",
font=("Arial", 12, "bold")).grid(row=0, column=0, padx=10, pady=5,
sticky='w')

    total_number_of_periods = int(output_schedule_data[0][1])

    output_time_period_options = ()
    for i in range(total_number_of_periods):
        output_time_period_options = output_time_period_options + (str(i
+ 1),)

    output_combobox_str_var = StringVar()
    output_period_number_combo_box =
ttk.Combobox(output_period_results_frame,
textvariable=output_combobox_str_var, state="readonly")
    output_period_number_combo_box['values'] =
output_time_period_options
    output_period_number_combo_box['textvariable'] = str(1)

    output_period_number_combo_box.grid(row=0, column=1, padx=10,
pady=5, sticky='w')
    output_period_number_combo_box.current()

    output_period_number_combo_box.bind("<<ComboboxSelected>>",
output_period_result_select)


def output_draw_scheduling_period_results():
    for row, row_data in enumerate(output_time_period_data, start=1):
        for col, value in enumerate(row_data):
            label = ttk.Label(output_period_results_frame, text=value,
font = ('Times', 20), relief = GROOVE)
            label.grid(row=row, column=col, padx=10, pady=5, sticky='w')


output_draw_scheduling_period_results()

# TODO remove - Set temporal data
# temp_auto_populate_files()

# ************ End of output **********
window.mainloop()
```