A DUAL INTERCHANGE ALGORITHM FOR THREE-DIMENSIONAL STOPE BOUNDARY OPTIMISATION FOR UNDERGROUND MINES

Adeodatus Sihesenkosi Nhleko

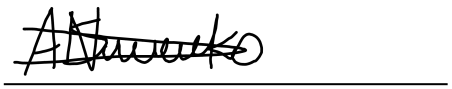
A thesis submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy

Johannesburg, 2022

DECLARATION

I declare that this thesis is my own, unaided work. Where use has been made of the work of others, it has been duly acknowledged. It is being submitted to the degree of Doctor of Philosophy to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University. I have read the University Policy on Plagiarism and hereby confirm that no plagiarism exists in this thesis. I also confirm that there is no copying, nor is there any copyright infringement in this thesis. I willingly submit to any investigation in this regard by the University of the Witwatersrand, and I undertake to abide by the decision of any such investigation.

Signed:



Adeodatus Sihesenkosi Nhleko

This 01 day of February year 2022

ABSTRACT

Mineral Resources are extracted using surface or underground mining methods to generate maximum economic value for the mining company extracting the resources. The objective of value maximisation necessitates the development and application of optimisation algorithms that will ensure maximum value is realised. Several algorithms have been developed to solve the value optimisation problem for near surface deposits. Examples of these algorithms include the Lerch-Grossman algorithm and dynamic programming. It has been observed by some researchers that the study on open pit mine geometry optimisation has reached saturation levels as there are many algorithms that can produce a 'guaranteed optimal solution'. However, the underground geometry optimisation problem remains largely unsolved due to its complexity, thus, there are limited algorithms developed for it. This thesis was therefore, undertaken to contribute to the few existing algorithms for underground geometry optimisation by developing a more versatile algorithm in handling variable stope boundaries and it is a dual interchange algorithm (DIA) that works by combining the strengths of two existing generic algorithms.

Once an appropriate underground mining method has been selected to extract a mineral deposit, mine planners need to generate optimal layouts for development and infrastructure, stope and production schedules that incorporate equipment selection, which are solved as optimisation problems. These problems introduce a circular logic which introduces complexity in deciding on which part of the optimisation problem should be the starting point. In this study, the stope layout optimisation problem was selected as the starting point because when optimising for development layout, production schedule and equipment selection, the spatial position of the stopes to be extracted is one of the constraints.

The DIA was developed by incorporating the principles of the particle swarm optimisation (PSO) algorithm and genetic algorithm (GA). The PSO algorithm was applied for the stope layout optimisation problem to exploit its strength for solving the problem in three-dimensional (3D) space to generate feasible solutions. The GA was used to optimise the stope layout in each level since its evolution capabilities are well-suited for stope layout optimisation. Since metaheuristic-based algorithms also do not

guarantee true optimality, the DIA exploited the strengths of both PSO and GA to generate superior solutions in 3D space.

The DIA was then coded in Python programming language because it is simple to code in Python language and the execution of the code is much faster compared to other programming languages. The DIA was tested using a synthetic Platreef reef mineral deposit where a resource model was used as an input and then converted to an economic block model using economic parameters. The Platreef deposit is a platinum group elements (PGEs) deposit which is amenable to extraction using bulk (or massive) mining methods such as longhole stoping, making it an ideal candidate for stope boundary optimisation. Thereafter, the DIA generated several solutions and selected the one with a maximum value as the optimum solution. The results of the algorithm were validated using the Mineable Shape Optimizer (MSO) available in the commercial Datamine software.

Different scenarios were used to demonstrate the performance of the DIA in different mining scenarios. Two Scenarios A and C considered a fixed stope width, while Scenarios B and D considered a variable stope width to better reflect variable orebody boundaries or contours as encountered in actual mining practice. The DIA generated superior results compared to the MSO where the stope layout economic value solutions for Scenarios A, B and D were 0.3%, 3.4% and 8.3% more profitable than those generated by the MSO, while the MSO generated a solution that was 9.7% more profitable than DIA for Scenario C. The undiscounted economic value was used as a proxy since the study is on stope boundary optimisation, it does not include stope production scheduling. The DIA produced superior results because its architecture is such that it is best suited for variable stope width of the orebody. The solutions of the MSO were generated in much shorter run times than those of the DIA. This is alluded to the fact the MSO creates a single solution, while the DIA generates several solutions during the optimisation process. It is recommended that the DIA could be adapted and applied to other mineral resource models to maximise the economic value of the respective mining projects.

PUBLICATION EMANATING FROM THE THESIS

The following publication has so far emanated from this thesis:

Nhleko, A.S. and Musingwini, C. (2019). Analysis of the particle swarm optimization (PSO) algorithm for application in stope layout optimisation for underground mines, In *Proceedings of the Mine Planner's Colloquium 2019*, The Southern African Institute of Mining and Metallurgy, Glenhove Conference Centre, Melrose Estate, Johannesburg, pp. 57-68.

ACKNOWLEDGEMENTS

I thank God for granting me the opportunity to do this research work. I would like to thank the following people for their support in the completion of this thesis:

- My supervisor, Professor Cuthbert Musingwini (Head of School - Mining Engineering) for his invaluable insight and support throughout this research work.
- Philsiwe Setshogwe for the assistance with the original resource model, its interpretation and ideas on converting it to a synthetic block model.
- My colleagues, Tinashe Tholana, Paskalia Neingo, Dr. Joseph Githiria and Tshepo Mmola for their suggestions and support in improving the thesis.
- Purpose Katakwa and Gift Khangamwa for their valuable guidance and assistance in correcting the coding done in Python programming.
- Datamine team: Reshoketswe Tati, Thembani Chirinda, Brenda Campos, Giovanna Cazali and Bruno Abilleira for assisting with Datamine software and training particularly on the MSO module.
- My wife, Thobile Nhleko and my daughters, Simelokuhle and Mpiloyethu for adding meaning in my life and their understanding during the times I had to be away from them to enable me to complete this thesis.
- My parents, Eunice Makhunga and Thomas Nhleko, and my siblings, Malusi and Sambulo for their support and encouragement in my studies.

DEDICATION

In loving memory of my grandparents, Andreas, and Regina Nhleko.

Ngiyabonga ngiyanconcoza boMgilija ngemfundiso nothando lwenu.

CONTENTS **Page**

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS OR ACRONYMS AND UNITS OF MEASURE

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
4E	Sum of Pt, Pd, Rh and Au grades in grammes per tonne
6E	Sum of Pt, Pd, Rh, Ru, Ir and Au grades in grammes per tonne
ANN	Artificial neural network
Au	Gold
BEV	Block economic value (or B_{ij} in 2D or B_{ijk} in 3D)
csv	Comma-separated values
Cu	Copper
DIA	Dual interchange algorithm
DP	Dynamic programming
g	grammes
GA	Genetic algorithm
Gbest	Global best
GUI	Graphical user interface
IDE	Integrated development environment
IP	Integer programming
Ir	Iridium
LG	Lerchs-Grossman
LHD	Load haul and dump

LP	Linear programming
MCDM	Multi-criteria decision making
MILP	Mixed integer linear programming
MIP	Mixed integer programming
MSO	Mineable Shape Optimizer
MVN	Maximum Value Neighbourhood
Ni	Nickel
NPV	Net present value
OLL	Optimised level stope layout
OR	Operations research
Os	Osmium
OSL	Optimised stope layout
oz	Troy ounce
Pbest	Personal best
Pd	Palladium
PGEs	Platinum group elements
PSO	Particle swarm optimisation
Pt	Platinum
Rh	Rhodium
Ru	Ruthenium
SA	Simulated Annealing
SEV	Stope economic value
SLEV	Stope layout economic value

SLO	Stope Limit Optimizer
t	Metric tonne
TS	Tabu search
UG2	Upper Group 2
UPL	Ultimate pit limit
USD/$	United States dollar

1 INTRODUCTION

1.1 Overview of Chapter 1

Chapter 1 provides background information about the importance of applying optimisation techniques to maximise the economic value of a mineral deposit. This chapter also discusses the different complexity levels when optimising open pit limits in surface mining and stope boundaries in underground mining. The problem statement and objectives of the research study are also presented.

1.2 Optimisation of surface and underground mine designs

Natural resources which include Mineral Resources, play an important role in the economic development of many countries globally. Mineral Resources are extracted using surface or underground mining methods. There are various phases involved in the mining process forming a value chain, i.e., exploration; orebody modelling; mining method selection; mine development/construction; ore extraction and transportation; beneficiation; marketing and distribution; and rehabilitation. Each activity in the mining value chain must add value to the chain of activities to generate value for the mining entity. The activities sometimes involve trade-offs to be made between business processes to optimise the value chain.

The objective of generating maximum value necessitates the development and application of optimisation algorithms or techniques. Optimisation is the act of reaching the best possible result under some defined circumstances (Astolfi, 2006). The appropriate optimisation technique depends on the type of mining method amenable to the mineral deposit. The techniques applied when exploiting mineral deposits using surface mining methods differ from those applied in underground mining methods. The main driver of these differences is the number of options available for the extraction of a certain block of ground under defined constraints. The following sections discuss the different challenges experienced when extracting a block of ground using either of the mining methods.

1.2.1. Surface mine design optimisation

There are several algorithms developed to optimise open pit limits in surface mining and these are, *inter alia*, the Lerchs-Grossman (LG) algorithm, Korobov's algorithm and dynamic programming (DP) (Ataee-Pour, 2000). In general, there are numerous options for

the removal of a certain mining block in open pit mining. The LG algorithm assumes that the open pit is mined in both sideways and downward directions. The sideway direction is limited by the slope angle. The depth of the pit is limited by economics of mining the block of ground on the i^{th} row and j^{th} column, measured by using a block economic value (BEV), shown as B_{ij} in Figure 1.1. The economic value is derived by subtracting from the revenue generated all costs associated with extraction, processing and selling of the mineral recovered from the block of ground. If $B_{ij}<0$, then a negative value is generated as the cost to recover the mineral is greater than the revenue, thus, it should be classified as waste. When $B_{ij}=0$, the cost to recover mineral is equal to the revenue generated, thus the block is a marginal block. Where $B_{ij}>0$, the block is treated as ore since its extraction results in net profit (Musingwini, 2016).

Figure 1.1 depicts a two-dimensional (2D) schematic view of an open pit configuration, which is in an *xy* plane, that demonstrates the simplicity of mining a block of ground in an open pit operation. The block model consists of 54 blocks, distributed as contiguous blocks in a pattern of nine blocks along the *x*-axis and six blocks along the *y*-axis. In open pit mining, a block cannot be mined without mining all the blocks above it. Nonetheless, for a given slope angle constraint dictated by geotechnical requirements, there is only one option for mining a certain block. If the slope angle constraint is 45°, there is only one optimum unique inverted cone for the removal of block B_{ij}, which is the shaded area in Figure 1.1. A different optimum solution can be generated only if the slope angle is changed.

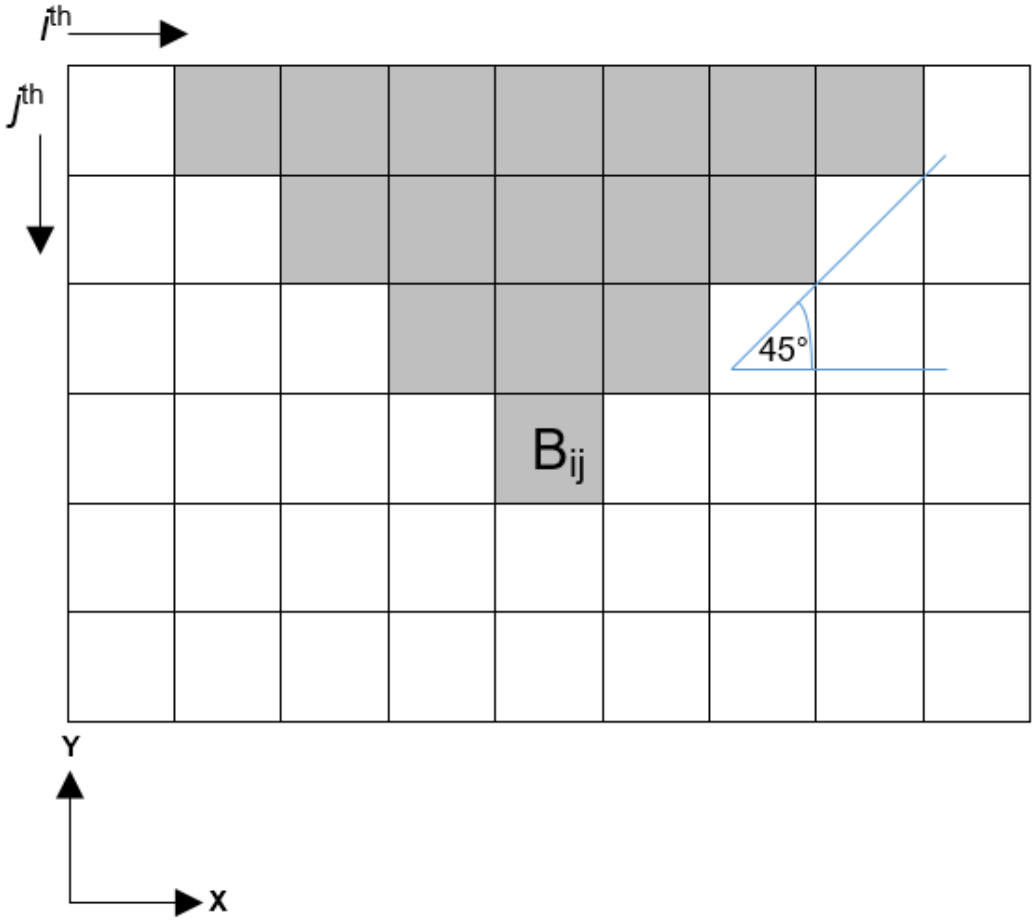


Figure 1.1 Two-dimensional view of the geometry for an open pit

1.2.2. Underground mine design optimisation

When applying underground mining methods there are numerous possible options for the extraction of a certain block (Sandanayake, 2014). For an illustration, assume that sub-level stoping is a suitable method for ore extraction. Sub-level stoping is an underground mining method that is classified as self-supporting and requires a lot of pre-development. The required development is located inside the orebody; thus, it can generate revenue. For ore extraction, large cavities are created which are called stopes. Workers do not access these stopes; however, sub-levels are developed at regular intervals for drilling and blasting purposes.

According to Musingwini (2016), to generate an optimal stope layout, various mining blocks with block economic value, B_{ij}, must be combined. The BEV is based on the principle that the economic value of a block is equivalent to the revenue realised from selling the metal content recovered from the block minus all the costs associated with extracting the block, processing, and refining the metal from the ore to be prepared for sale. The combination of these economic blocks must not violate the physical mining and geotechnical constraints

to generate a stope layout that maximises the economic value of a mineral deposit. The decision whether a block is included in the final stope layout requires that all possible combinations be evaluated for variable stope sizes and pillars. These possible combinations are evaluated from thousands of mining blocks to obtain maximum value. This requirement results in computational complexity in the stope boundary optimisation problem.

For example, assume that a stope size is limited to three-by-three blocks both along the *x*-axis and *y*-axis. Figure 1.2 illustrates that there are nine possible solutions to extract the mining block with a BEV of B_{ij}, indicated as stopes 1 to 9. Nonetheless, a mining block can only belong to one stope, thus, the mine planner must manually select one stope that will be part of the final stope layout. The requirement for a manual manipulation by the mine planner cannot guarantee optimality. When the stope size is increased to 4 by 4 blocks, the candidate mining block with a BEV of B_{ij} becomes part of 16 possible stope combinations. Therefore, as the stope size increases, the number of stope combinations increases exponentially resulting in a more complex stope layout optimisation problem (Musingwini, 2016).

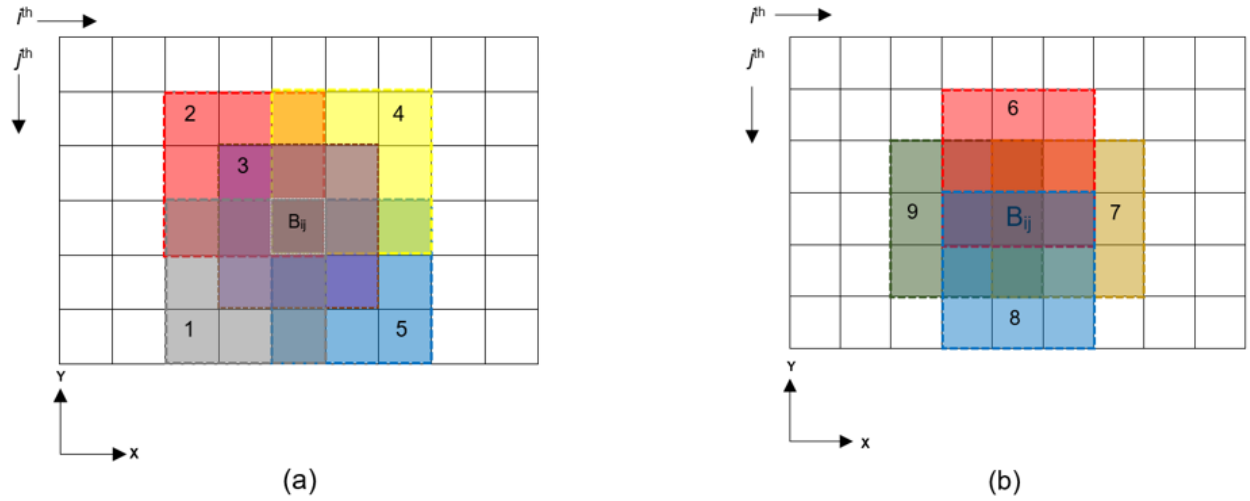


Figure 1.2 Two-dimensional view of the geometry for an underground stope

There are limited optimisation algorithms and techniques that have been developed for underground mining. Consequently, there is a need for the development of algorithms that will yield an 'optimum' solution in underground mine planning. The techniques include those that have been used to generate stope boundaries. However, all these models fail to generate satisfactory optimal solutions in three-dimensional (3D) space, which has the xyz

dimensions. Therefore, there is a need for improvement in the optimisation of stope layouts if truly optimal underground mine designs are to be realised in the mining industry.

1.3 Problem statement

To sustain or expand production in mining operations, there should be continuous investment. Mining projects require intensive capital injection with long payback periods, thus, investing in the mining business can be very risky. Therefore, a small improvement in the mining value chain can lessen the risk on invested capital (Nhleko and Musingwini, 2016; Sens, 2011). To enhance the competitiveness of the industry, mine designs produced must be optimum. However, it is more complex to optimise underground mining operations, as the limitations are greater due to physical geotechnical and geological requirements as discussed earlier in Sections 1.2.1 and 1.2.2. Few attempts have been made to generate optimum stope layouts for underground mines. None of these approaches have been able to guarantee optimum solutions in 3D space due to the complexity of the stope boundary optimisation problem, as discussed earlier.

After the exploration phase, data collected is analysed and interpreted using geostatistical modelling techniques to produce an orebody model. The orebody model is delineated into thousands of mining blocks in 3D space each with assigned grade values. The grade values depend on the type of the orebody being modelled, for instance gold and copper will be in grams per tonne (g/t) and percentage (%), respectively. The geological information will inform the type of mining to be adopted whether surface or underground mining. Consequently, the appropriate mining method is selected. It is now that mine planners can commence with the generation of optimal layouts for development and infrastructure, stope and production schedules.

The generation of an optimal stope layout revolves around selecting a combination of numerous mining blocks. The economic viability of each block is determined using certain parameters as input into the mine planning process. These parameters include, *inter alia*, the commodity price, mining cost, transportation cost, processing cost and marketing cost. Determination of an optimum stope layout is important as it is used as an input into the production scheduling formulation.

Using a sub-optimum stope layout as an input into the formulation of production schedules could result in unrealistic expectations being incorporated into the schedules, subsequently, having a negative impact on both the life of mine (LOM) and maximisation of the stope layout economic value (SLEV). Development of underground stopes requires the consideration of physical, geological, and geotechnical constraints. These include the hanging wall and footwall angles, size of mining equipment, extent of primary development for accessing the stopes, and size and dip of the orebody. These parameters affect the design of the dimensions of the stopes. Optimisation at each of the stages in the mining process is critical to ensure adequate utilisation of the Mineral Resources and reduction of operating cost. Mining operations are concerned with the optimisation of the mine geometry as it plays an important role in improving production efficiency of the overall mine (Atee-Pour, 2000; Bai, 2013; Sandanayake, 2014).

It is improbable to extract a mining block of high-grade ore without some level of dilution due to extraction of waste blocks. Extraction of waste blocks will result in additional costs being incurred and if adequate care is not exercised, might lead to a negative SLEV. Consequently, a viable mining project might be abandoned or shelved. On the other hand, a combination of ore and waste blocks may yield a positive SLEV given attributable or applicable geotechnical constraints.

Most optimisation techniques fall under two broad categories namely, exact methods and heuristic approaches. Exact methods are rigorous approaches that generate optimum solutions within reasonable time for application in solving mining optimisation problems only when applied on a panel or a section of the orebody. Exact methods can also produce optimum solutions in 3D space. However, solution time increases exponentially for large-scale block models and may not generate feasible solutions. Heuristic algorithms can produce stope layouts in 3D but can fail to produce details of the ultimate mining layout such as the location of panels. Hence, these techniques have not been able to yield ‘optimum’ stope boundary solutions. Therefore, this study aimed to develop an algorithm that is a dual interchange algorithm (DIA) to overcome the weaknesses while incorporating strengths of each method adopted. This leads to the question: “*Can a dual interchange algorithm be developed to generate an improved optimal underground mining stope boundary solution in 3D space given applicable physical and geotechnical mining constraints?”*

1.4 Research objectives

This study is aimed at developing an algorithm for stope boundary optimisation for underground mines to maximise the SLEV. This was achieved through incorporating the strengths of two metaheuristic approaches. However, it should be noted that optimisation of development and production scheduling play equally significant roles in the success of a mining project as does stope layout. The newly developed algorithm comprises of the characteristics of the particle swarm optimisation (PSO) and genetic algorithm (GA) which are metaheuristic techniques. The objectives of this study were to:

- Develop an algorithm that yields a guaranteed optimum stope layout in 3D to ensure that orebody expectations included in production scheduling are realistic.
- Implement the proposed algorithm on a synthetic orebody to test its robustness.
- Maximise the SLEV for the selected orebody model.
- Validate the algorithm by testing its robustness against an existing software-based algorithm to ascertain if there is value added to the existing knowledge.

1.5 Project background

The study used block model data which is amenable to the longhole stoping mining method. A synthetic block model was compiled using data from the Platreef deposit in the Northern Limb of the Bushveld Complex in South Africa. This data source was selected because of the availability of data necessary for orebody modelling. The longhole stoping mining method was selected because the Northern Limb platinum group element (PGE) deposit is amenable to this mining method for extraction. The Platreef mineralisation comprises PGEs, gold, copper, and nickel. The thickness of the reef varies from a few metres to a few hundred metres (McCutcheon, 2012). The main reason for choosing this project area is that there are several platinum mines being planned for the Northern Limb, thus, making it a ‘future’ area of platinum mining in South Africa.

1.6 Significance and relevance

Numerous algorithms have been developed for optimisation of open pit limits because the direction of mining is downward and outward to the pit limits resulting in fewer permutations. On the contrary, there has been little focus directed to the optimisation of the underground mine design, especially the stope layout, globally. Alford (1995) developed the floating stope algorithm, and it was incorporated into the Mineable Shape Optimizer (MSO) module

in Datamine commercial software. The Stope Limit Optimiser (SLO), which is a non-commercial software tool was developed for the implementation of the 3D Maximum Value Neighbourhood (MVN) algorithm (Ataee-Pour, 2006). Sens and Topal (2009) developed a heuristic approach that eliminates overlapping stopes in the solution as encountered with the floating stope algorithm. However, stopes are selected in order of the mine planner's preference, such as, from maximum to minimum economic values. Therefore, consideration of all possible solutions is violated, thus, it cannot guarantee generation of optimal solutions (Erdogan *et al.,* 2016). The existing algorithms can produce approximate solutions. However, none of these algorithms guarantees generation of a true optimum solution for the 3D stope boundary optimisation problem.

1.7 Thesis overview

This thesis comprises of six chapters which are outlined as follows:

- Chapter 1 presented the overview of underground stope layout optimisation; outlined the purpose of the research and objectives of the study.
- Chapter 2 provides a review of previous research work on stope boundary optimisation and other related research work to date and provides justification for a DIA.
- Chapter 3 discusses the methodology followed to develop the DIA and to validate its robustness.
- Chapter 4 presents a detailed background on the construction of the DIA, and the constraints considered for the optimisation problem.
- Chapter 5 presents the implementation results of the proposed algorithm using the synthetic Platreef deposit. Furthermore, it includes the validation process for the algorithm that was developed by comparing with existing algorithms.
- Chapter 6 presents the outcomes of the study, scrutinises the performance of the algorithm and provides recommendations for future research work.

2 LITERATURE REVIEW ON STOPE BOUNDARY OPTIMISATION

2.1 Overview of Chapter 2

Chapter 2 discusses different operations research (OR) techniques commonly used in the mining industry for optimisation. These OR techniques can be categorised under exact and heuristic algorithms. Additionally, heuristic algorithms that have been applied to maximise the economic value of mineral deposits are evaluated and contrasted.

2.2 Optimisation in the mining industry

A ‘good’ optimisation model must empower mine planners to produce mine designs and production schedules that yield a high net present value (NPV); and longer life of mine (Dimitrakopoulos, 2011). Some of the main considerations during mine design are flexibility of the design and riskiness of the project. The mine planner must perform risk analysis of the project on different parameters such as commodity price and grade to inform mitigating measures that must be put in place. A good mine plan must be flexible to accommodate the cyclical nature of commodity prices as encountered in the mining industry. According to Musingwini (2016), the circular logic requires fragmenting the underground optimisation problem into broad sub-problems, which are development and infrastructure placement; stope boundary and layout definition; equipment selection and deployment; and production scheduling (Figure 2.1).

The objective of the optimisation problem is to maximise the NPV of the mine plan through optimising development and stope layouts. However, development and stope layouts cannot be produced without knowing the production costs and cut-off grade. On the other hand, production costs and cut-off grade are unknown until mining layout and production scheduling have been determined. However, the latter two parameters cannot be determined unless the optimum mining method and production capacity are known. The mining method to be used can be selected by using several methods such as the multi-criteria decision making (MCDM) method. The MCDM method is usually applied when decisions require several criteria to be considered simultaneously. Mining method selection is also dependent on pre-determined production capacity (Musingwini, 2016). Circular logic presents complexity in knowing which part of the four optimisation sub-problems should be regarded as a starting point. Erdogan *et al.* (2016) argued that stope boundary optimisation

might be regarded as a starting point in a full optimisation process when optimisation of development and production schedule are taken into consideration.

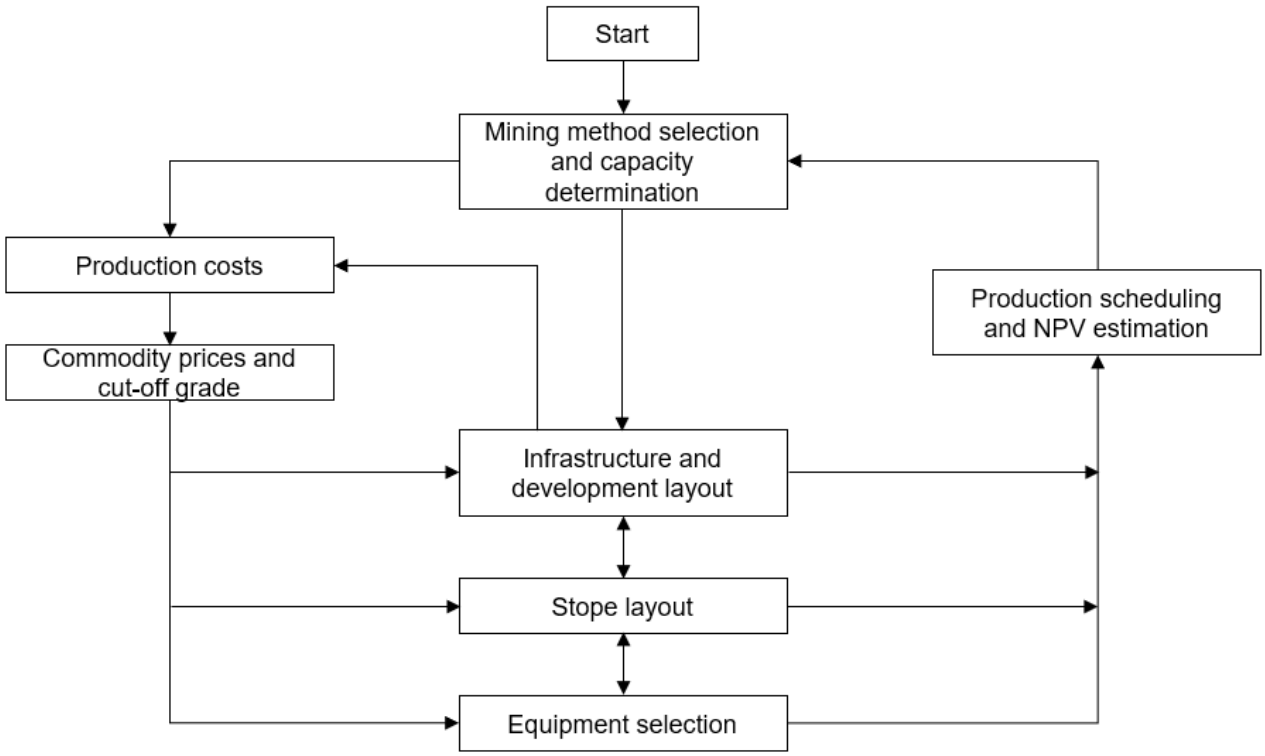


Figure 2.1 Circular logic of underground mine optimisation

Source: Adapted from Musingwini (2016)

Several studies have been conducted focusing on underground mine development and production scheduling optimisation (Sens and Topal, 2009; Topal and Sens, 2010; Sandanayake, 2014), but only a few on stope boundary optimisation. Thus, this study is based on stope boundary optimisation to augment knowledge in this area. After exploration, the data collected are analysed and used to represent the orebody in 3D space through orebody modelling and subsequently generating geological block models. The geological block models are utilised to generate stope layouts. The stope layout is then used to optimise production schedules (Dimitrakopolous, 2011). However, most of the estimation techniques and approaches for mine planning have a common drawback in failing to guarantee true optimality in 3D space (Ataee-Pour, 2000; Sandanayake, 2014). Hence, this study gave more attention to optimisation in 3D space.

Most of the estimation techniques and traditional approaches to mine planning have a common disadvantage, which is failure to account for *in-situ* spatial variability. This failure cannot be ignored, if ignored; it may lead to unrealistic production expectations (Dimitrakopouls, 2011). There are geostatistical estimation methods that are used to

produce an orebody model with a high level of confidence. Geological uncertainty is outside the scope of this research study as the research focus is premised on a geological orebody model with a high level of confidence.

Mining optimisation techniques have been applied to solve mining problems since the 1960s. These techniques revolutionised surface mine design and production scheduling. For example, for pit limit optimisation, there are commercial packages including those that implement the widely used moving cone and Lerchs-Grossmann algorithms (Topal and Sens, 2010; Sandanayake, 2014). Ataee-Pour (2000) stated that the study of pit geometry optimisation has probably reached saturation levels, since there are many algorithms using a range of techniques to produce a 'guaranteed optimal solution'. On the contrary, underground stope optimisation remains a major challenge due to a lack of suitable algorithms to address underground mining optimisation problems (Sandanayake, 2014).

Operations Research (OR) techniques have been used in the mining industry to address optimisation problems. OR techniques are scientific decision-making tools for solving management problems by selecting the best solution from possible alternative solutions. The goal of OR is optimisation in the utilisation of scarce resources. Optimisation can take the form of maximisation (e.g., profit); minimisation (e.g., cost) or a combination of both maximisation and minimisation such as in the case of multi-objective optimisation problems. This study aimed at developing an algorithm that can maximise the economic value of a mining project because economic value is the best proxy for value measurement for a mining project extracting polymetallic commodities. Additionally, most optimisation algorithms use economic value as the proxy for value measurement.

Optimisation models include, amongst others, Linear Programming (LP); Integer Programming (IP); Mixed Integer Linear Programming (MILP) or Mixed Integer Programming (MIP); GA and PSO. There have been several algorithms that have been developed and implemented to solve the stope layout optimisation problem. Sandanayake *et al.* (2015) and Ataee-Pour (2000; 2005a) classified these algorithms as either exact or heuristic. Exact (rigorous) algorithms are supported by mathematical proof and include dynamic programming; branch and bound technique; downstream geostatistical approach; and network flow method. Exact methods are usually partial based, in other words, they are used to optimise a panel or a section of the orebody. When the problem to be optimised is considered in 3D space (i.e., the entire orebody) or when panels and levels are combined,

optimality is violated, thus, failing to guarantee true optimality. Heuristic algorithms are used to solve the problem based on defined rules confining the search region for the optimal stope layout (Ataee-Pour, 2005a; Little, 2012).

Heuristic algorithms can be applied to optimise the entire orebody, and some provide a 3D analysis of the problem. However, Ataee-Pour (2005a) emphasised that these techniques do not give adequate details of the ultimate mining layout such as the location of the levels and panels. Nonetheless, they are suitable for the long-term or conceptual planning phase where little details are known. Examples of heuristic algorithms include the octree division algorithm; floating stope algorithm; Sens and Topal heuristic approach; and Maximum Value Neighbourhood (MVN) algorithm.

Sandanayake *et al.* (2015) stated that heuristic approaches offer better solutions to the stope layout optimisation relative to exact methods. This statement is true in respect of solution time since heuristic algorithms do not guarantee optimal solutions while exact methods can generate optimum stope layout solutions. Artificial intelligence (AI) approaches have not been widely used in the optimisation of stope layouts. The examples of artificial intelligence approaches are GAs and Artificial Neural Network (ANN) approaches. The main drawback of these algorithms is their inability to yield optimal results. The GA is not a true optimisation method but can be applied to obtain a solution even though this may not be optimal.

Producing an optimal stope layout to maximise the economic value of a mine subject to physical and geotechnical constraints is a complex challenge with currently no known solution (Bai, 2013; Sandanayake, 2014). The complexity is because there are numerous variables to be considered in defining a stope such as stope dimensions, geotechnical constraints, and footwall and hanging wall slope angles. In some models, as the number of variables increase, the solution times increase exponentially (Table 2.1). This limits their applications to only simple orebodies with fewer variables. As can be seen from Table 2.1 more comprehensive optimisation techniques such as the Sandanayake and Topal approach have solution times that are in the order of several hours, since they incorporate more variables and complexity of orebodies as encountered in actual mining practice. Such order of magnitude of solution times will be seen later in Section 5.6 in Chapter 5 of this thesis because the DIA attempts to address conditions encountered in actual mining practice.

Table 2.1 Solution times for various optimisation techniques

Source: Erdogan *et al.* (2016)

Technique	Constraint(s)	Number of stopes generated	Solution times (h:mm:ss)
Floating stope algorithm	Stope dimension	N/A (envelopes)	00:02:13
Mine shape optimiser	Stope dimension, pillar dimension	41	00:00:50
MVN algorithm	Stope dimension	18	00:01:56
Sens and Topal approach	Stope dimension	66	00:00:12
Sandanayake and Topal approach	Stope dimension, mining level optimisation	64	05:17:00

Some of the stope optimisation techniques can produce optimum stopes in one-dimensional (1D) and 2D spaces. Simplification of the problem to 1D and/or 2D configurations reduces the complexity of the optimisation problem but results in unrealistic solutions as not all geotechnical constraints are incorporated. The desired solution should be in 3D, however, the challenge with real 3D stope definition is that the techniques:

- Are basically heuristic.
- Cannot directly incorporate geotechnical constraints.

As a result, the mining engineer must manually adjust the stope layout solution to obtain a feasible stope layout. This is subject to the experience of the engineer and violates optimality of the stope layout. The key optimisation techniques that are commonly used in the mining industry for stope layout optimisation are briefly discussed in the next sections.

2.3 Exact methods

This section presents a review of exact methods that are commonly used in stope boundary optimisation and describes their advantages and disadvantages. The exact methods considered are the mixed integer programming-based models, branch and bound approach, downstream geostatistical approach, and dynamic programing.

2.3.1 Mixed integer programming-based models

MIP is the combination of linear programming and integer programming techniques to define feasible solutions. The application of MIP has been in areas that include transportation scheduling, distribution of goods to production planning in manufacturing (Salama *et al.*, 2014) and production scheduling in mining (open pit and underground). Grieco and Dimitrakopoulos (2007) developed a mathematical probabilistic mixed integer programming (MIP) model to determine stope and pillar designs based on grade uncertainty and predefined levels of acceptable risk. The orebody was regularised into mineable rings, assumed to have incorporated the major geotechnical and production constraints. These rings were assigned binary variables in the MIP model. The probabilistic algorithm optimises the stope design based on the following process (Figure 2.2):

- Divide the orebody into a series of layers.
- Subdivide each layer into panels (stopes).
- Further, subdivide each panel into a series of mining rings (blocks).
- Assign each ring to a binary variable in the MIP model.

The objective function is to maximise the metal content extracted. The model considers geological uncertainty during stope design that limits the minimum, maximum mining rings, and the size of pillars to be left unmined. Based on the minimum and maximum allowable stope size, it determines the number of rings a stope will encompass. Therefore, the optimum stope layout is generated based on the rings that are defined relative to the position and size of the most profitable stopes. However, this algorithm does not ensure precise examination of stopes in different locations of the orebody. The performance is directly proportional to the number of binary variables; thus, the solution time increases as the number of rings increases (Little, 2012; Sandanayake, 2014).

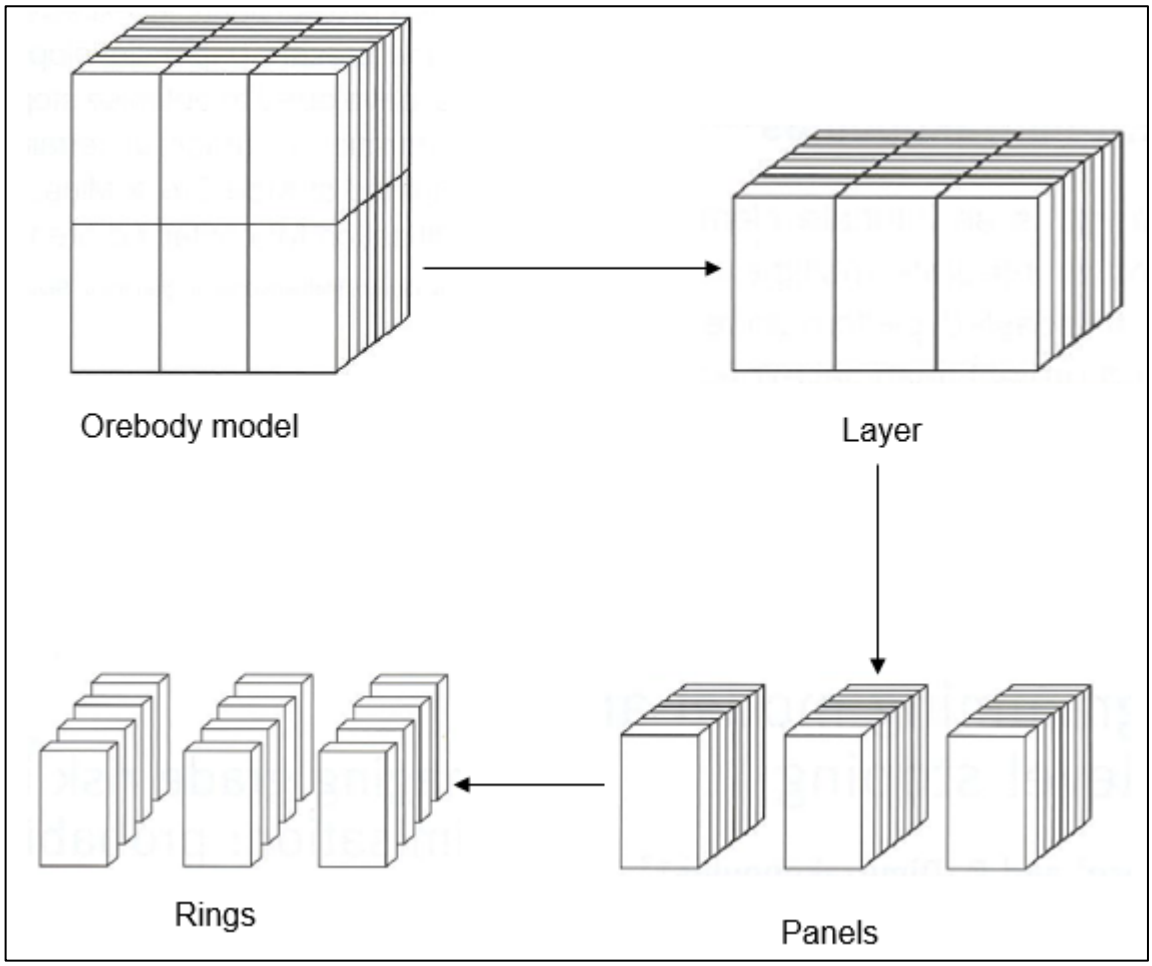


Figure 2.2 Components of required orebody model
Source: Adapted from Grieco and Dimitrakopoulos (2007)

2.3.2 Branch and bound approach for stope boundary optimisation

The branch and bound approach creates an optimum stope boundary by optimising the starting and ending points of each row of blocks. To facilitate the integration of constraints such as stope length, a mixed integer approach known as Type-Two Special Ordered Set is introduced. The objective of this algorithm is to identify the start and end locations for mining given stope dimensions. This objective is realised by the application of the two piecewise linear, cumulative functions for each row. The first function accounts for all the blocks with positive BEVs to be included in the stope boundary along the row, whereas the second function deals with blocks with negative BEVs to be excluded from the stope boundary. Therefore, the difference between these two functions is the sum of all the block values lying between start and finish locations (Ataee-Pour, 2000; Little, 2012; Sandanayake, 2014). Figure 2.3 shows a typical cumulative economic value function of the blocks with seven mining blocks. The algorithm applies the set stope dimension values to locate the optimal start and end positions. The blocks are spaced 15m apart and the

economic value for each block is indicated inside the cell. Figure 2.4 depicts a typical cumulative block value function for each sequential block position.

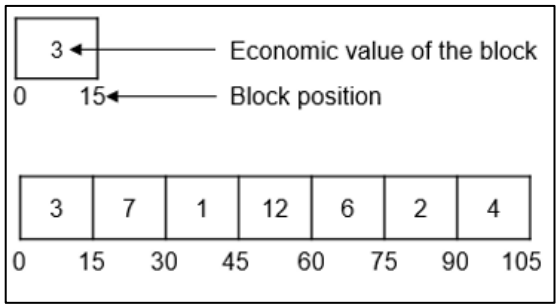


Figure 2.3 Row of economic value blocks

Source: Adapted from Ataee-Pour (2000)

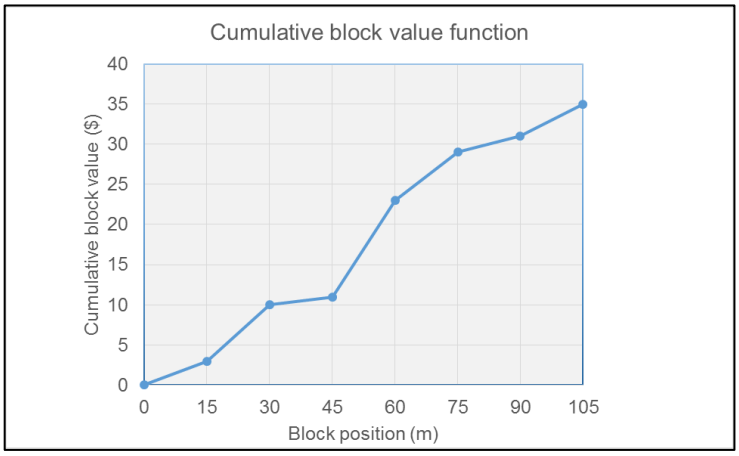


Figure 2.4 Cumulative value function of blocks in the row

Source: Adapted from Ataee-Pour (2000)

Application of this technique is limited to simple orebodies that can be modelled in 1D space along the mining direction. It ignores the slope of the stope walls as one of the constraints. Solving mixed integer programming problems is time consuming when working with large number of blocks. It should be noted that this approach only provides true optimality in 1D space (Ataee-Pour, 2000; Bai, 2013; Sandanayake, 2014).

2.3.3 Dynamic programming algorithm

The DP algorithm is utilised to solve complex problems by breaking them down into simpler sub-problems. DP has been used to define stope layouts for operations using the block caving mining method. DP assumes that there is no footwall region, and that maximum profit is obtained. Consequently, the region is separated by a footwall and all possible combinations of the mining and non-mining areas are examined. If the value generated is greater than or equal to the assumed maximum profit when there was no footwall, the

process is applied again, but to the second areas divided by the footwall. This procedure is repeated until there are no more profitable footwall cases or there is no feasible footwall to introduce (Ataee-Pour, 2000; Little, 2012; Sandanayake, 2014). DP is a widely used approach because it allows the creation of solutions sequentially (Newman *et al.*, 2010).

DP has been used to solve complex problems through a recursive approach. DP problems are solved through forward pass and backward pass algorithms. For the forward pass approach a recursion formula is applied to each stage and the transitory optimum is saved in memory. At the end, a backward pass traces back the path leading to the optimal solution. The general recursion formula is depicted in Equation 2.1. The dynamic process is optimally laid out such that the optimality of the solution is not dependent on the position of the starting point. For instance, if the process is set to begin at a known stage, n, the layout of the process will be optimally laid out regardless of the choice to enter the beginning stage, n. Figure 2.5 shows an optimal solution to a maximisation problem where $S_3 > S_2 > S_1$.

$$\boldsymbol{f_n = opt\,\{f_{n-1} + g_n\}} \qquad 2.1$$

Where:

opt is either maximisation or minimisation.

f_n is the function value at point n which is the given/known stage.

f_{n-1} is the function value on a previous consecutive point n-1.

g_n is the return from moving from n-1 to n.

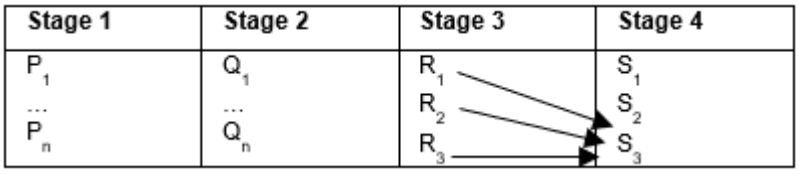


Figure 2.5 Optimal selection for a maximisation problem using DP

Source: Adapted from Musingwini (2016)

DP provides optimum stope layouts in 2D space. Sections of the orebody can be optimised when combining the optimum 2D sections to form a 3D stope boundary. However, it should be noted that with such a simplification the stope constraints can be violated, and optimality is not guaranteed in 3D.

2.3.4 Downstream geostatistical approach

The downstream geostatistical approach uses mathematical morphology operations such as opening and closing, to manipulate a 2D block model image. Figure 2.6 shows where mathematical morphology was used to impose the stope geometry, to the ore-waste images for the transformation of the images. Thereafter, conditional simulation was combined with underground mining simulation to compare selectivity, productivity and profitability of three mining methods. The mining methods that were used are sub-level stoping; cut and fill; and mixed sub-level stoping. This approach is based on constructing a numerical model of the orebody and defining the outlines of the mineable mineral resource for these mining methods. The geometrical constraints for the three mining methods are depicted in Figure 2.6. This approach can also generate a 3D solution; however, it does not consider the economic profit of the block as its morphological operations control the geometry (Ataee-Pour, 2000; Bai, 2013; Little, 2012). Therefore, it cannot ensure optimality as it only satisfies the stope geometry not in conjunction with profitability of the stope.

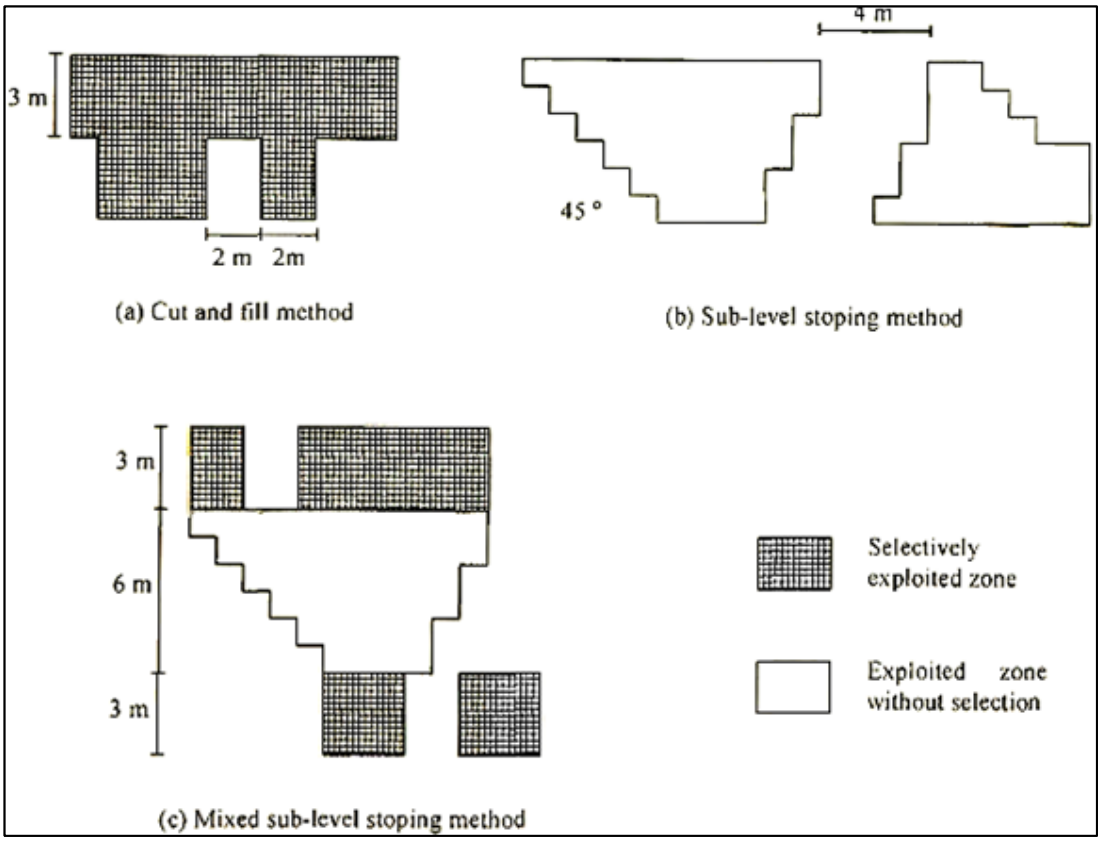


Figure 2.6 Geometrical constraints for three mining methods

Source: Ataee-Pour (2000)

2.4 Heuristic algorithms

Various heuristic algorithms have been developed to optimise stope boundaries. These algorithms include the floating stope, multiple-pass floating stope, maximum value neighbourhood, octree division, Sens and Topal heuristic approach, simulated annealing, and particle swarm optimisation. The next sections explain how some of these heuristic algorithms have been applied to optimise stope boundaries.

2.4.1 Floating stope algorithm

Alford (1995) developed the floating stope algorithm, and it is incorporated into the MSO module in the Datamine commercial software. The floating stope algorithm suggests the stope boundaries for economically mineable ore within the geological block model. The algorithm specifies the minimum stope dimensions. Then it floats the stope in the orebody relative to the origin in defined float increments in three orthogonal directions. The process starts by specifying a cut-off grade to separate ore and waste blocks and the targeted head grade is specified for a stope. The algorithm then determines if any block above cut-off grade can be included in the stope that meets the set head grade. The challenge is that there are several stopes under which a specific block can be assigned, however, the block will be assigned in the stope with highest head grade (Ataee-Pour, 2000).

The process of floating the stope shape in the block model may lead to the definition of two envelopes, (i.e., inner and outer envelopes). An inner envelope is composed of all blocks above the cut-off grade, and it is the union of the best grade stope shapes. An outer envelope may be defined by all blocks above cut-off grade; it is the union of all possible stope positions for each block (Figure 2.7). The final stope design depends on the input by the mine planner. However, it should be close to the inner envelope as far as practicably possible and lie within the outer envelope (Little, 2012).

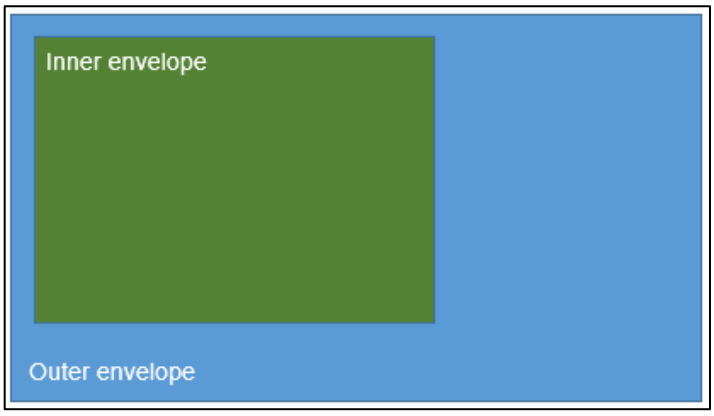


Figure 2.7 Inner and outer envelopes for a single stope

The main disadvantage of using this algorithm is that it suggests a domain of good stope layouts within a range as opposed to exact layouts. It may generate uneconomical stopes by combining two overlapping stopes that share mining blocks that are uneconomical, therefore, requiring manual adjustments to exclude the shared blocks from the final stope layout. It does not consider the slope of stope walls as one of the constraints. Therefore, this algorithm does not guarantee an optimal solution to the stope layout challenge (Sandanayake, 2014).

2.4.2 Multiple-pass floating stope process

The multiple-pass floating stope is an extension of the floating stope and allows more envelopes to be generated by employing a multiple optimisation process. This process is executed through the following steps (Sandanayake, 2014):

- Defining the input parameters such as head grade, cut-off grade and maximum waste.
- File generation where economic stope envelopes are created for each set of parameters.
- File management where data files or statistical files are converted into a Microsoft Excel compatible (csv) format.

This process provides the mine planning engineer with extra information when deciding on the best stope layouts. The problem with this algorithm is that it does not eliminate the shortcomings of the floating stope algorithm (Little, 2012; Sandanayake, 2014). Consequently, it cannot guarantee an optimal stope layout solution.

2.4.3 Maximum value neighbourhood algorithm

Ataee-Pour (2000) developed the MVN algorithm assuming the floating stope premise. The MVN is based on the principle that only the neighbourhood block set with maximum value of interest in all possible neighbour sets will be selected. The size of neighbour sets is constrained by the minimum stope dimension. Consequently, the selective combination of best neighbour sets of all blocks yields an 'optimal' stope layout. The SLO, which is a non-commercial software tool has been developed for the implementation of the 3D MVN algorithm (Ataee-Pour, 2006).

A hypothetical example of how the MVN algorithm works when applied to a one-dimensional block model is illustrated in Figure 2.8. This example consists of a row of seven blocks assigned with block economic values labelled *a-g* for reference.

Assume a minimum stope size of three blocks

a	*b*	*c*	*d*	*e*	*f*	*g*
-2	1	4	3	2	5	-1

Step 1: Start constructing the neighbourhood from block *a* but it is negative and the algorithm moves to the next block;

Step 2: Feasible neighbourhood for block *b* is evaluated and maximum value selected;

i. | -2 | 1 | 4 | Neighbourhood value is: -2+1+4=3

ii. | 1 | 4 | 3 | Neighbourhood value is: 1+4+3=8

Neighbourhood *ii* is selected as the MVN for block *b* and blocks *c* and *d* are flagged along with block *b*.

Step 3: Move to the fifth block as blocks *c* and *d* are flagged, thus, feasible neighbourhood of block *e* is evaluated;

i. | 3 | 2 | 5 | Neighbourhood value is: 3+2+5=10

ii. | 2 | 5 | -1 | Neighbourhood value is: 2+5-1=6

Block e is flagged and neighbourhood *i* is selected as it has the highest value. Blocks d and f are flagged along with block e. thus, feasible neighbourhood of block g is evaluated in step 4.

Step 4: Block *g* violates the minimum stope size constraint, therefore, the algorithm terminates; and

Step 5: The final stope layout consist of the flagged blocks as indicated below with hatches.

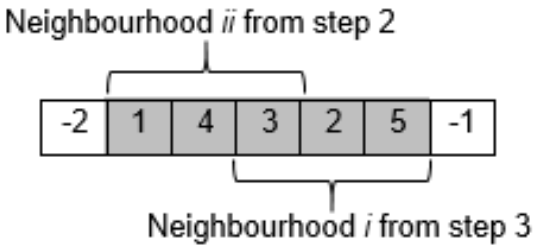


Figure 2.8 Hypothetical example of the MVN algorithm

Source: Adapted from Sandanayake (2014)

In this algorithm, negative blocks, flagged blocks or blocks where the maximum neighbourhood value is negative are excluded during calculation. The example yielded two MVNs comprising of blocks *b*, *c*, *d*, *e* and *f*. This stope layout solution is not practical for mining purposes. For instance, if solution *i* in Step 2 was selected, mining of block *b* would not be possible as it violates the minimum stope size constraint. Similarly, if solution *ii* in Step 3 was selected, mining of block *f* would have been impossible considering the minimum stope size constraint. According to Sandanayake (2014), this algorithm generates different solutions depending on the starting point, thus it fails to guarantee the best solution for the stope optimisation problem.

The shortcomings of the MVN algorithm are that (Little, 2012; Bai, 2013; Sandanayake, 2014):

- The starting location of evaluation alters the set of stope layouts generated.
- Blocks examined earlier in the process are given preferential treatment in the stope layout.
- It does not factor in the cost of mining size of the stope selected, but only individual blocks. However, bigger stopes can provide less cost per tonne compared to small stopes, thus, changing the selected neighbourhood.
- It does not consider the stope wall slope constraints.

The MVN algorithm is based on a simple concept and provides ease of computational implementation. However, the shortcomings indicate that it cannot ensure that the best solution is obtained.

2.4.4 Octree division algorithm

The octree division approach produces a geometrical object of the orebody. Subsequently, it transforms it into Mineral Reserves by applying geometric constraints such as working space dimensions and economic constraints. The algorithm collects the geological data and builds a geological block model. It identifies Mineral Reserves based on economic evaluations and then determines the mining sequence. The Mineral Reserves are divided into sub-volumes for further evaluations. If the mineral content and/or the dimensions of the sub-volume violates the set constraints, the sub-volumes are removed from the model. Figure 2.9(a) depicts the block division into eight sub-volumes, while Figure 2.9(b) shows the removal of a block that violates the constraints of the algorithm (Sandanayake, 2014).

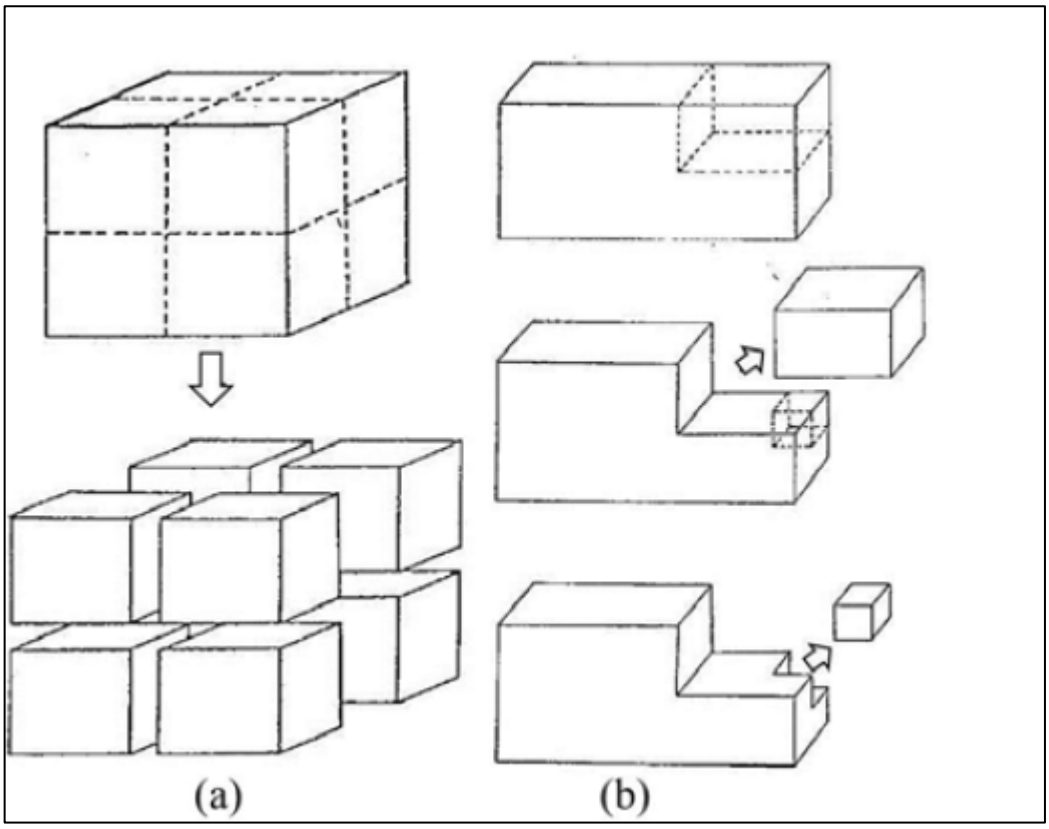


Figure 2.9 (a) Octree division algorithm (b) Removal of sub-volumes

Source: Sandanayake (2014)

These sub-volumes are assessed based on the following instructions:

- Remove any sub-volume that does not contain a mineralised vein but satisfies the constraints.
- Store in the mineable block any sub-volume that has the minimum allowable stope size or has all the mineralised vein.
- Divide into eight equal sub-volumes any sub-volume containing a proportion of the mineralised vein, i.e., the sub-volume is located partly inside the ore and partly outside the ore and does not have the minimum allowable stope dimension.
- Terminate the algorithm when all sub-volumes have been evaluated.

This approach can generate 3D stopes but includes unnecessary waste in the layout as it cannot analyse sub-volumes jointly. Blocks with minimum dimensions irrespective of the quantity of ore contained are also included in the final stope layout. This approach does not guarantee an optimum stope layout (Ataee-Pour, 2000; Bai, 2013; Little, 2012; Sandanayake, 2014).

2.4.5 Sens and Topal heuristic approach

The Sens and Topal heuristic approach converts the mining blocks to a regularised block model such as the block model constituting of mining blocks with consistent sizes. Consequently, stopes constrained by the block heights, lengths and widths are generated from the regularised economic block model. After stopes generation, the algorithm is implemented in Matlab software based on the economic value of the stopes. Matlab is used because of its ability to evaluate multidimensional arrays. The results obtained from Matlab are visualised with respect to different user-defined parameters. These results can be visualised using Minesight software (Erdogan *et al.*, 2016; Sandanayake, 2014).

The advantage of this approach is that it can locate stope boundaries using different stope sizes and selection strategies in 3D. However, it selects stopes in descending order of the economic values of the stope set, while, eliminating overlapping stopes. Therefore, it eliminates the possibility of multiple stope combinations that can be derived from a given stope set. In some of the combinations, there may be combinations with higher total economic value (Sandanayake, 2014). Therefore, these shortcomings result in optimality being violated.

Figure 2.10 explains the problem with Sens and Topal approach by using a hypothetical situation. This approach selected a combination of Stopes 1 and 2 with a total economic value of 15. However, the most profitable combination of stopes is given by Stopes 3 and 4 with a total economic value of 17. This example shows that this approach to select stopes combination in descending order of economic value while discarding overlapping stopes does not guarantee generation of an optimum solution.

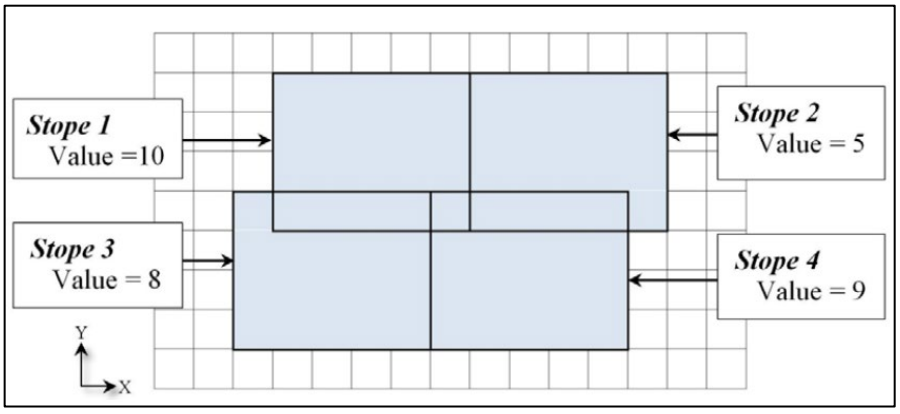


Figure 2.10 Hypothetical 2D application of Sens and Topal heuristic approach

Source: Sandanayake (2014)

2.4.6 Network flow algorithm

Bai (2013) developed an algorithm to optimize stope design for the sub-level stoping method based on a cylindrical coordinate system defined around an initial vertical raise. The algorithm considers hanging wall and footwall slope angle and stope width during the optimisation process. The selection of a block for inclusion into the stope is subject to the following additional constraints:

- Maximum distance of the block from the raise.
- Horizontal width required to bring the outermost block to the raise.

The algorithm optimises the stope boundary through the following steps (Bai, 2013):

- Specify the initial location of the vertical raise.
- Define a cylindrical coordinate system (r, θ, z) centred around the specified raise location, see Figure 2.11(a).
- Establish a linkage between economic blocks subject to the constraints (i.e., hanging wall and footwall slope, and stope width constraints).
- Construct a graph using vertical arcs for the set constraints, see Figure 2.11(b).
- Select a block that maximises the stope value subject to the additional constraints.

The algorithm was validated using the floating stope algorithm and it obtained better results. However, this algorithm is limited to sub-level stoping and small-mineralised orebodies.

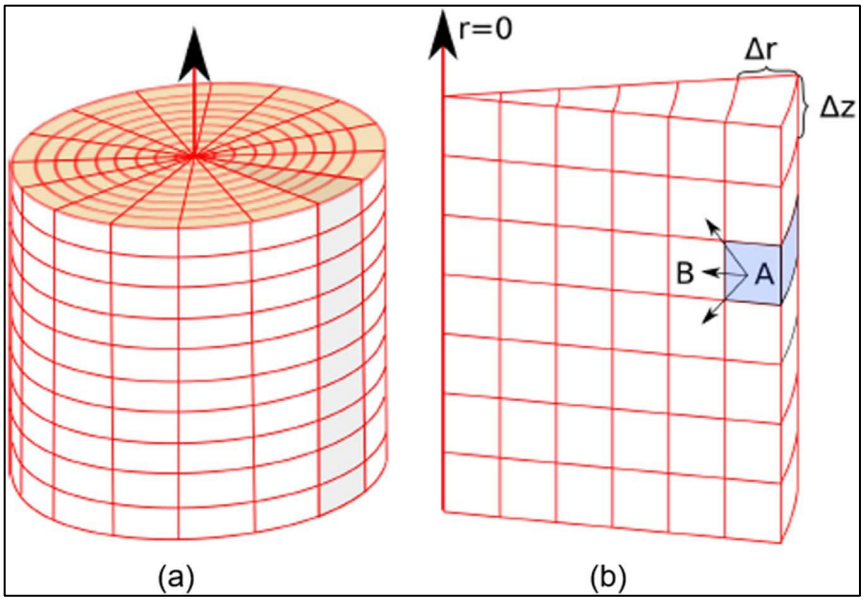


Figure 2.11 Network flow algorithm principle

Source: Bai (2013)

2.4.7 Sandanayake's heuristic algorithm

Sandanayake (2014) developed an algorithm to optimise stope designs. The algorithm converts an irregular geological block model into a regularised geological block model; consequently, converting it to an economic block model. The algorithm defines the stope size in terms of number of economic blocks that constitute a stope in the x, y, and z-axes. Figure 2.12 illustrates a hypothetical stope size, i.e., 2 x 3 x 2 blocks along the x, y and z-axes, respectively. The defined stope size is floated within the economic block model along the axes to identify all stopes that maximise the stope value (Figure 2.13). The algorithm generates numerous solutions containing several stopes that are non-overlapping. The solution with the highest economic value is selected as the optimum stope layout.

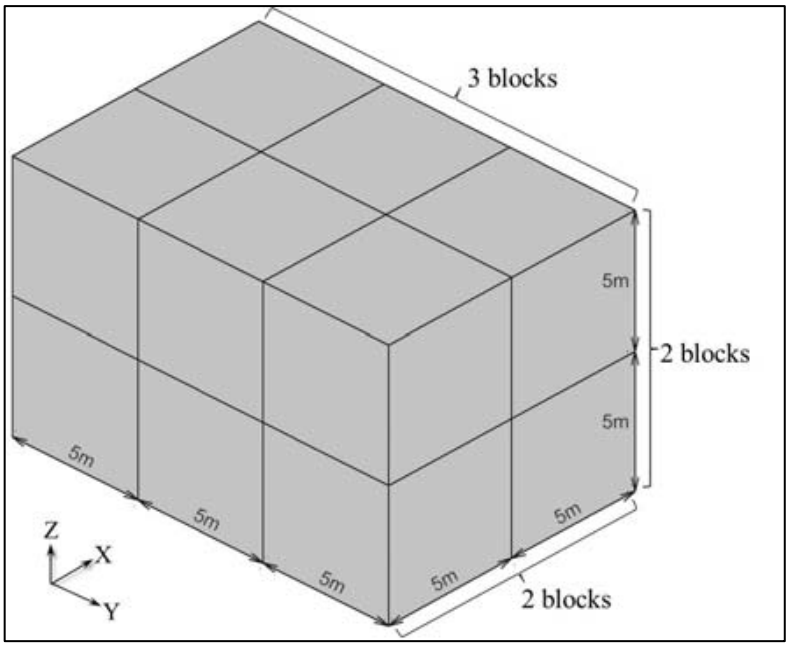


Figure 2.12 Stope size with blocks along the x, y, and z-axes
Source: Sandanayake (2014)

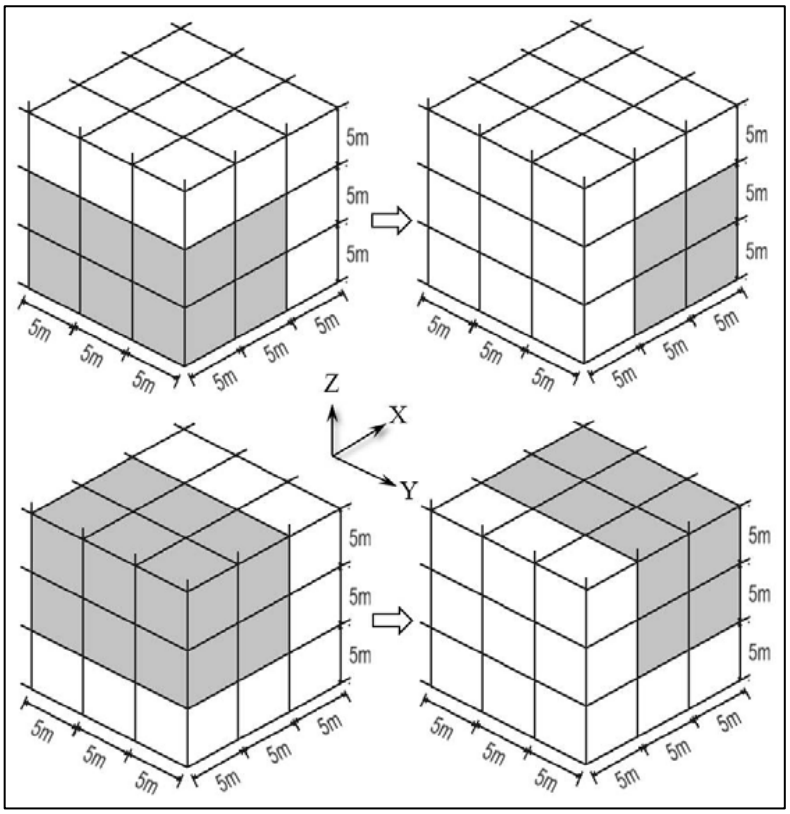


Figure 2.13 Stope generation process

Source: Sandanayake (2014)

2.4.8 Particle swarm optimisation algorithm

The PSO algorithm is a heuristic global optimisation algorithm, which is inspired by the behavioural nature of bird flocks and fish schools (Alam, 2016; Bai, 2010). When birds are searching for food, they either are scattered or flock together. One bird can smell food better than others can, that is, the bird with better resource information. The birds are constantly transmitting information, while searching for food from one place to another. Therefore, the birds will ultimately flock to the place where food can be found. In relation to the particle swarm optimisation technique, solution swarm is the bird swarm, and the movement of birds from one place to another is the development of the solution swarm. Good information transmitted is equal to the most optimistic solution, and the food resource is like the most optimistic solution during the whole course. The most optimistic solution can be realised by the cooperation of each individual particle (Nhleko and Musingwini, 2019).

In the PSO algorithm, each particle modifies its position according to its current position, its current velocity, the distance between its current position and personal best (*Pbest*), and the distance between its current position and global best (*Gbest*) position. Therefore, the PSO can be used to solve the stope boundary optimisation problem in 3D. The PSO can

be used to optimise SLEV in the long term (*Gbest*) whilst optimising the profit in the short term (*Pbest*) for a mining project. Mathematically, particles in the swarm are manipulated based on Equations 2.2 and 2.3:

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_1^k \left(Pbest_{id}^{k} - x_{id}^{k}\right) + c_2 r_2^k \left(Gbest_{d}^{k} - x_{id}^{k}\right) \quad 2.2$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \quad 2.3$$

The parameters $v_{id}^{k}\ and\ x_{id}^{k}$ separately represent the speed of the particle i at its k times and the d-dimension of its position. The speed of the particle is set to control the particle from being far away from the search space, thus, the speed and direction are confined between these parameters. If the speed values are too big the solution will be far from the global best while too small speed values result in the solution being trapped in local optima. The parameter $Pbest_{id}^{k}$ is the d-dimension quantity of the individual i at its most optimistic position at its k times. The parameter $Gbest_{d}^{k}$ is the d-dimension of the swarm at its most optimistic position at k times (Nhleko and Musingwini, 2019).

The coefficients c_1 and c_2 are the acceleration coefficients, which represent the rate at which speed changes, regulating the length when flying to the most optimist individual particle and to the most optimistic particle of the whole swarm. If the coefficient is too small, a particle may be too far away from the target field, whereas a big coefficient value that is too big may cause a particle to fly to the target field suddenly or fly beyond the target field. The figures of $c_1$and c_2 can control the speed of the particles flying in the target field; hence, the solution will not be a partial optimisation solution. Usually, c_1 is equal to c_2 and they are both equal to two (2). The coefficients r_1 and r_2 are random factors between 0 and 1 (Nhleko and Musingwini, 2019).

Different combinations of the PSO input parameters may lead to a slower convergence rate or non-convergence. Thus, it is important that an appropriate combination of these parameters is selected to achieve best convergence. The velocity update, position update and objective function evaluation calculations are iterated until a desired stopping criterion is achieved. In each iteration, the last step is the position update. The position of each particle is updated using its velocity vector as shown in Equation 2.2 and illustrated in Figure 2.14. The parameter p^i represents the best position of the particle i in current and all previous moves. The parameter p_k^g is the position of the particle with the best global fitness at current move k.

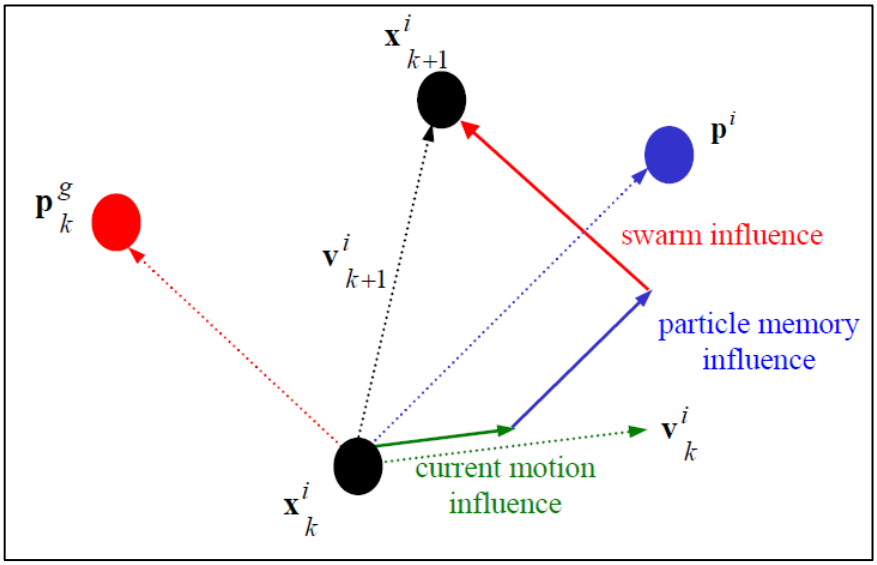


Figure 2.14 Particle velocity and position update in PSO

Source: Hassan *et al.* (2005)

The PSO algorithm can be used to solve complex optimisation problems such as the stope layout optimisation problem (Bai, 2010). The PSO algorithm is increasing in its use because it is easy to implement and requires a few particles to be tuned. The PSO algorithm does not guarantee an optimal solution (Kennedy and Eberhart, 1995; Kok and Lane, 2012), but is more applicable in a 3D space such as in the stope boundary optimisation problem.

2.4.9 The 3D approximate hybrid algorithm

Nikbin *et al.* (2018) developed a 3D approximate hybrid algorithm which is a combination of the DP and greedy algorithms for underground stope boundary optimisation. The 3D optimisation problem is converted to a 1D sub-problem by converting the 2D subsets of the economic block model into single blocks. Thereafter, the DP algorithm is applied to generate a set of solutions for the optimal boundaries of the 1D sub-problem. The best solution is selected based on the greedy approach criterion. Thereafter, the corresponding blocks to the selected solution are added to the initial 3D problem solution (Nikbin *et al.*, 2018).

The algorithm optimisation solution was validated against the floating stope algorithm, MVN algorithm and greedy algorithm solutions. It produced superior results compared to the other three algorithms. However, the algorithm does not guarantee a true optimum solution but generates a better solution (Nikbin *et al.*, 2018).

2.5 Comparison of commonly used stope optimisation algorithms

Table 2.2 depicts a comparison among the commonly used stope optimisation algorithms. Most of the algorithms considered the stope dimension as a geometric constraint, whereas factors like hanging wall and footwall angles were considered in the DP, network flow and the geostatistical approach. It is evident that most of the heuristic approaches only considered the stope dimension as the constraint. However, most other factors that will have an impact on the optimality of each stope geometry were not considered. These parameters include, *inter alia*, the stope walls accounting for rock strength, *in-situ* stress tensor, and local structures such as faults and dykes.

Table 2.2 Comparison of exact and heuristic stope optimisation algorithms

Source: Adapted from Nhleko and Musingwini (2017)

Algorithm	Classification	Mining method(s)	Dimension limitation	Optimisation criteria	Geometric constraints	True optimality
MIP based algorithm	Exact	All	3D	Metal content	Grade uncertainty, number of rings, pillar size	No
Branch and bound	Exact	All	1D	Economic value	Stope dimension (stope length)	Yes
Dynamic programming	Exact	Block caving	2D	Economic value	Stope dimension; draw control	No
Downstream geostatistical approach	Exact	Cut-and-fill; sub-level stoping	2D	Geometry	Stope dimension; hanging wall, footwall slope angle	No
Floating stope	Heuristic	All	3D	Economic value	Stope dimension	No
Multiple-pass floating stope	Heuristic	All	3D	Economic value	Stope dimension	No
Maximum value neighbourhood	Heuristic	All	3D	Economic value	Stope dimension, mining level	No
Octree division	Heuristic	All	3D	Grade/vein	Stope dimension, economic constraints	No
Sens & Topal approach	Heuristic	Sub-level stoping	3D	Economic value	Stope dimension	No
Network flow algorithm	Heuristic	Sub-level stoping	3D	Grade/economic value	Footwall angle, hanging wall angle, distance from raise centreline and horizontal width	No
Sandanayake	Heuristic	Sub-level stoping	3D	Economic value	Stope dimension, pillar separation and level location	No
Hybrid algorithm	Hybrid	Sub-level stoping	3D	Economic value	Stope dimension	No

There are several metaheuristic algorithms that have been applied to optimise various problems. These algorithms have their own shortcomings ranging from unstable convergence to being unsuitable for handling complex problems. Table 2.3 presents the main shortcomings of other metaheuristic algorithms. Ezugwu *et al.* (2020) compared the performance of various metaheuristic algorithms and concluded that the DE, PSO and GA algorithms outperformed the other algorithms according to the computed percentage success ratio metrics used. Furthermore, the PSO and GA required a smaller number of function evaluations to reach an optimum solution. Therefore, the PSO and GA have a potential to solve the complex stope boundary optimisation problem in 3D space. Other benefits of using the PSO and GA are that they are easy to implement in 3D and simple to code in Python language.

Table 2.3 Main shortcomings of some common metaheuristic algorithms

Source: Nhleko and Musingwini (2019); Ezugwu *et al.* (2020); Mulani and Desai (2018); Rivera Letelier (2021)

Algorithm	**Disadvantages**
Ant colony optimisation (ACO)	ACO is challenging in theory because it is premised on structures of random decisions of distinct artificial ants. ACO gets trapped in local optima when it reaches a stagnant phase, and the solution cannot be recovered without modifying its parameters.
Artificial bee colony algorithm (ABC)	ABC has a premature convergence in the later stage of its search and the best value may not be high enough to meet the objective function.
Bat algorithm (BA)	The standard BA cannot efficiently handle discrete optimisation problems.
Cuckoo search (CS)	CS has low convergence rate, unstable convergence, and generates low solution accuracy.
Differential evolution (DE)	DE has unstable convergence and easily get trapped in local optima. In DE, applying the same parameter values may not produce the global best solution.
Firefly algorithm (FFA)	FFA is not suitable for handling complex problems because it can easily be trapped in local optima. Introducing diversification in FFA results in reduced search speed and low convergence rate.
Flower pollination algorithm (FPA)	FPA easily gets trapped in local optima and has a slow convergence rate and low precision.
Genetic algorithm (GA)	GA does not have a standard method for defining a good fitness function thus care must be exercised when defining the objective function because generating best solutions depends on it.
Invasive weed optimisation (IWO)	IWO is effective in solving problems with large search spaces because a larger number of seeds must be applied to each plant to examine the whole search space, thus exponentially increasing the computation time.
Particle swarm optimisation (PSO)	In PSO, different combination of parameters may lead to slower convergence rate or non-convergence at all. Therefore, appropriate combination of parameter values must be selected to achieve best convergence.
Simulated Annealing (SA)	The major shortcoming of the SA is that it is difficult to assess the quality of the solution and the time to generate an 'optimal' solution can be too long.
Symbiotic organisms search	It does not require any specific parameters; thus, the optimisation process may be time consuming because it only relies on iterative performance.
Tabu Search (TS)	TS has several tunable parameters, and it is difficult to find the correct parameters for a specific optimisation problem. In addition, the TS requires a high number of iterations which leads to high computational time if the number of constraints considered is also large.
Toposort	The Toposort algorithm sorts blocks in topological order. However, if there are many possible topological orderings available, it needs an 'optimal' solution of the LP relaxation to guide the sorting process. The LP relaxation optimisation process requires that all variables be integers, i.e., fractional values must be rounded off to obtain an integral solution. The rounded LP relaxation solution is undesired for stope boundary optimisation problem with an objective function of maximising the economic value because it violates optimality principles. The optimality of the solution is not guaranteed because the BEVs can be fractional values and for the application of the LP relaxation, they will have to be rounded off to integers.

None of the techniques presented in Tables 2.2 and 2.3 can guarantee an optimum stope envelope solution except the branch and bound approach, DP algorithm and downstream geostatistical approach. The branch and bound approach offers true optimality in 1D space, however, as one moves to 2D it fails to guarantee optimality. The DP algorithm and downstream geostatistical approach are the only exact methods that can generate optimum stope layouts in 2D and can be extended to 3D. However, in the three-dimensional space, they do not guarantee an optimum solution as discussed in Sections 2.3.3 and 2.3.4. The major shortcomings of the downstream geostatistical approach are that it does not consider economic profit of the stopes and does not always adhere to geometrical constraints. For this study, economic values of the stopes are important to be able to maximise the SLEV of the mineral deposit. Hence, the downstream geostatistical approach was not considered as an option to be explored further for this study.

The PSO algorithm can be used to solve 3D optimisation problems as explained in Section 2.4.8. Furthermore, Mmola *et al.* (2019) applied the PSO algorithm to optimise a conceptual orebody subject to stope dimension, level, non-overlapping, and unique stope constraints. The findings showed that the PSO algorithm is amenable to the stope layout optimisation problem, thus justifying further investigation.

It is evident that there is still a need to explore the opportunities for developing algorithms that will generate optimum stope layouts for underground mining. This research study, therefore, adopted the PSO algorithm and GA principles in developing a dual interchange algorithm for stope layout optimisation. The DIA works by iteratively interchanging the two algorithms as explained later in Chapters 3 and 4.

2.6 Summary of Chapter 2

There are numerous algorithms that have been developed for optimising the open pit limits and some of them can guarantee optimum solutions. There are four critical areas of underground mine planning optimisation i.e., development and infrastructure placement, equipment selection and deployment, stope boundary definition, and production scheduling. Several studies have been conducted focusing on the optimisation of underground development and infrastructure, and production

scheduling. However, it should be noted that further developments are still necessary as there is no algorithm that can guarantee an optimum solution in 3D space. On the contrary, there has been little research directed at stope boundary optimisation, hence this study aimed to develop an algorithm for stope layout optimisation.

Optimisation techniques fall under two categories which are heuristic and exact methods. Exact methods are supported by mathematical proof; however, they can only optimise a panel/section of the orebody within a reasonable time. When they are applied to solve the optimisation problem in 3D space, the computational time required may be expensive or even impossible to solve for large-scale block models. The DP algorithm and downstream geostatistical approach are the only exact methods that generate optimum stope layouts in 2D. However, when they are applied to 3D problems, they cannot guarantee an optimum solution.

On the contrary, heuristic approaches can generate 3D solutions for large-scale block models and some can provide details of the stope layout but will not guarantee true optimality as they are rule-based and not mathematically proven for optimality. Amongst the heuristic algorithms, the PSO algorithm has never been applied to the stope layout optimisation problem but, it is well-suited to solve the optimisation problem in 3D space. This premise is based on that it can be used to optimise SLEV in the long term (*Gbest*) and profit in the short term (*Pbest*) for a mining project.

The techniques discussed above have not been able to yield optimum stope boundary solutions in 3D. Consequently, there is still a need to develop an algorithm that can generate optimal stope layouts for underground mine planning in 3D. This study adopted the principles of the PSO and GA algorithms in developing a DIA to be applied in 3D stope layout optimisation phase of mine planning for mineral deposits. The next chapter discusses the methodology followed in developing the DIA and to validate its robustness.

3 RESEARCH METHODOLOGY

3.1 Overview of Chapter 3

This chapter presents the process followed in developing the DIA and briefly discusses the mining methods that can be used to extract the Platreef mineral deposit. It also presents the approach used in developing a synthetic Platreef geological block model and construction of the economic block model. The validation procedure for the DIA results is also described.

3.2 Extraction of the Platreef deposit

The Platreef deposit is amenable to extraction using surface mining and underground mining methods depending on the depth of reef below surface. For underground mining, various mining methods can be employed to different sections of the orebody. Examples include the drift and fill, drift and bench for thinner ore zones, and longhole stoping for thicker ore zones. A major part of the ore deposit is in the thicker ore zone category and will be extracted using longhole stoping, while sections of the orebody that are unsuitable for extraction using longhole stoping will be exploited using the other mining methods. Thus, this thesis developed an algorithm that can generate an optimal stope layout when using the longhole stope mining method.

The stope dimension consists of the width, height, and length, which are the parameters considered during the stope design. For stability of the mining area and production per stope, the strike length and height of the stope were kept constant. The stope width may vary depending on the thickness of the orebody and hydraulic radius. The following section discusses longhole stoping briefly.

3.3 Longhole stope mining method

The longhole stope mining method is also known as sub-level open stoping, sub-level stoping or blast hole stoping. The longhole stoping method is a bulk or massive mining method with good recovery and minimal dilution compared to other massive mining methods such as sub-level caving. This method requires that the orebody be almost regular in shape, have competent ore and stable host rock, and the footwall should

allow the ore to gravitate due to gravity. Longhole stoping requires a large amount of pre-development before stoping commences. The capital cost for this large amount of development is off-set by the fact that the development occurs within the orebody, thus, ore can be realised and processed at a profit to off-set development capital outlay. The location of the development and infrastructure is vital for the extraction of ore. However, the positioning of development is beyond the scope of this thesis. It was not considered because the stope boundary problem was selected as the first step in the optimisation process and including development placement optimisation would lead to an intractable optimisation problem.

Each stope consists of a drill drift and extraction drift at the top and bottom of the stope, respectively, leaving a block of ground between the drifts referred to as the bench. Once the drifts have been established, the stope is mined (see Figure 3.1). The broken ore is mucked from the bottom of the stope through the drifts using load-haul-dump (LHD) machines, which may tip the ore into orepasses, trucks or underground trains. The void created can be backfilled after a stope has been completely mined and the fill material should be allowed to cure before an adjacent stope can be extracted, otherwise pillars must be left in place to prevent stopes from collapsing.

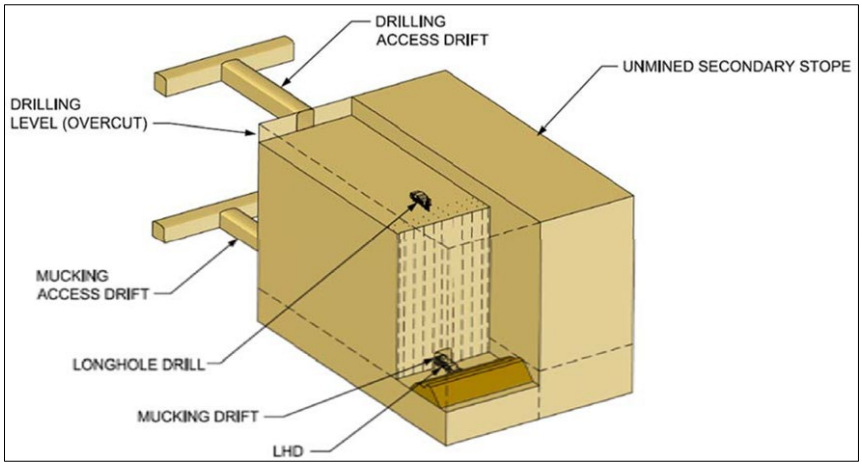


Figure 3.1 Generic longhole stope layout
Source: Ivanhoe Mines Ltd (2014)

3.4 Creation of a synthetic block model

The world supply of most platinum group elements (PGEs) is from the Bushveld Complex in South Africa, Stillwater Complex in the United States of America, Great

Dyke Complex in Zimbabwe and, Noril'sk and Talnakh Complexes in Russia. PGEs are important minerals that are used in several applications such as jewellery, automotive catalytic converters in vehicles, catalysts in chemical and petroleum processes and electrical devices (Cawthorn, 1999; Schouwstra *et al.*, 2000; Thormann *et al.*, 2017).

The Bushveld Complex consists of three limbs, namely the Western, Eastern and Northern limbs. It is the largest layered intrusion in the world. The Bushveld Complex hosts the Merensky reef, Upper Group Chromitite No.2 (UG2) and Platreef mineralisation. The ore types consist of platinum (Pt), palladium (Pd), rhodium (Rh), iridium (Ir), ruthenium (Ru), osmium (Os), gold (Au), copper (Cu), and nickel (Ni). The future of South Africa's PGE mining is in the Northern Limb as discussed in Section 1.5 and according to Cawthorn (2010), the Platreef is the third largest PGE deposit in the world after the Merensky and UG2 reefs. Therefore, a synthetic block model was created based on the Platreef deposit characteristics in terms of 4E grade distribution and prill split. The 4E grade refers to the combined content of the four most valuable precious metals namely platinum, palladium, rhodium, and gold.

The synthetic block model was created using Microsoft Excel as a csv file. The input values were generated in such a way that they mimic the characteristics of the Platreef deposit, and the steps followed were to:

- Create x, y and z block dimensions of 5m, 5m and 5m respectively. Block size and shape affect the stope layout optimisation process because large and non-cubic blocks limit the stope design in both the parallel and orthogonal directions of the block model. However, the optimisation performance maybe enhanced in terms of computational time. In contrast, smaller blocks provide better definition of the stope, consequently, resulting in a better stope layout in any direction (Thormann *et al.* 2017). Furthermore, it results in better management of internal dilution in the block model. However, as the block size becomes smaller, the optimisation process becomes slower, thus, increasing the computational time. Therefore, for this study, a block size of $25m^3$ was used to create a balance between block size and optimisation performance.
- Generate density values in tonnes per cubic metre (t/m^3).

- Randomly generate 4E grade values in grams per tonne (g/t) between 0.01 and 21.9. According to Thormann *et al.* (2017), the Platreef orebody has a high ore grade variation, thus a highly variable grade distribution was selected to mimic the Platreef mineral deposit.

According to Kinnard *et al.* (2005), Manyeruke (2007) and Armitage (2011), the Platreef deposit varies in thickness from approximately 50m in the north to about 400m in the south and strikes in the north to northwest direction and it is steeply dipping. Thus, the synthetic block model was created at a dip of 90˚ with dip direction due west and a strike direction due northwest.

3.5 Construction of the economic block model

In this study, a 3D geological block model was converted to a 3D economic block model where each ore block was assigned a BEV. To convert the geological block model to an economic block model several parameters were applied, *inter alia*, density, basket price, grade, dilution, recovery, cost of extraction, cost of processing and refining cost. Consequently, the objective function was set as the maximisation of the SLEV as it is the best proxy to measure the 'snapshot' economic value of the multi-commodity mining project. Thereafter, a minimum stope size was defined guided by the practicality of implementation during ore extraction.

A geological block model contains a set of blocks with estimates of grade values which are used to categorise blocks as either ore or waste based on a specific cut-off grade. However, it is important to express the content of each block in economic terms to determine its net worth for the mining optimisation process because most algorithms developed for the optimisation of the mining process for both open pit and underground mines are implemented on an economic block model of the mineral deposit with BEV as an attribute of the blocks (Ataee-Pour, 2005b).

The ultimate pit limit (UPL) optimisation principles for open pit mining were adopted for this study because the UPL is the largest possible envelope that contains all the blocks that are candidates for extraction, and it is based on the premise that all blocks within the UPL are extracted in a single period (Phillips *et al.*, 2021). However, depending on the mining method used to extract the underground reserves, transition to underground mining can occur earlier than UPL. The principles of the UPL are like

that of stope boundary optimisation problem because all the blocks in the economic block model are included in the optimisation process, but the time value of money is not accounted for. Therefore, the undiscounted UPL yields the net economic value of the mineral deposit because it does not account for the time value of money, which is applicable when optimisation of the production scheduling is considered.

The LG algorithm was mathematically modelled to solve the long-term open pit mine planning optimisation problem, and its approach was used as a base to develop analogies for underground stope layout optimisation adaptation. Table 3.1 presents the similarities drawn between open pit and stope boundary optimisation processes. The next phases will be scheduling the undiscounted UPL and stope layout which are beyond the scope of this study.

Table 3.1 Adaptation of the UPL optimisation principles for stope layout optimisation

LG algorithm	**DIA**
Open pit mining environment in 2D	Underground mining environment in 3D
BEV (B_{ij})	BEV (B_{ijk})
Block column value (M_{ij})	Stope economic value (SEV_{ijk})
Ultimate pit economic value (P_{ij})	Stope layout economic value ($SLEV_{ijk}$)

A 3D geological block model must be converted to a 3D economic model where each ore block is assigned a BEV. To convert the geological block model to an economic model several parameters are applied, *inter alia*, grade, dilution, metal price and cost of extraction and processing, and recovery. “Each block in the economic model can be identified by its x, y, and z co-ordinates. A typical block entry is represented by B_{xyz}, where x, y and z represent the sequential order of the block in X, Y, and Z directions, respectively” (Nhleko and Musingwini, 2019, p. 62). The resource model was valuated using Equation 3.1 to develop the economic model. For compatibility, it was necessary to convert the metal price in USD per ounce ($/oz) and grade in grams per tonne (g/t) to their equivalent USD per tonne ($/t) and percentage (%), respectively assuming 1t is equivalent to 32150.75 troy ounces (see Table 3.2).

Table 3.2 Conversion factors

Parameter	Conversion factor
Price	
US Dollar per tonne ($/t)	1
US Dollar per ounce ($/oz)	32150.75
Grade	
Percent (%)	1
Grade (g/t)	0.0001

The basket price, p_b, is applied where more than one mineral is considered for extraction purposes otherwise the price of the mineral concerned is used.

The conversion factors are:

- Metal price: Equation 3.2 shows the conversion from USD per ounce ($/oz) to USD per tonne ($/t) where a factor of 32150.75 is applied.
- Grade: conversion from grams per tonne (g/t) to percentage (%), a factor of 0.0001 was applied as shown in Equation 3.3.

$$BEV_r = \left[\left\{\left(p_f - r_r\right) \times g_f \times y_r\right\} - \left(e_r + c_r\right)\right] \times \left[t_r(1 - d_r)\right] \quad 3.1$$

$$p_f = 32150.75 \times p_b \quad 3.2$$

$$g_f = 0.0001 \times g_r \quad 3.3$$

Where:

BEV_r block economic value ($) for block *r*

p_b basket metal price ($/oz)

p_f metal price ($/t)

r_r refining cost of block *r* ($/t)

g_r grade of block *r* (g/t)

g_f grade of block *r* (%)

y_r recovery from block *r* (%)

e_r extraction or mining cost for block *r* ($/t)

c_r concentrating or processing cost of block *r* ($/t)

t_r tonnage of block *r* (t)

d_r dilution and mining losses incurred in block *r* (%)

3.6.1 Metal concentration

Platreef ore types consist of PGEs occurring in different concentrations; thus, others are not considered during feasibility studies. Section 1.5 stated that the Northern Limb has Platreef mineralisation with the platinum group elements, and gold, copper, and nickel. The mineralisation occurs in different ratios; therefore, it necessitates the use of the concept of average proportions. The concept of average proportions is called the prill split, which indicates the relative proportions of the metals contained in a tonne of ore. The average proportions of the metal volume slightly differ from mine to mine and with time (Musingwini, 2009; Chigumira and Raymond, 2017).

Musingwini (2009) and Mudd (2012) stated that most mining companies report elements contained in PGE ores as 4E or 6E grades while other mining companies only report PGE resource grade as 4E. The 4E grade is the sum of Pt, Pd, Rh, and Au grades and 6E grade is the sum of Pt, Pd, Rh, Ir, Ru, and Au grades in grammes per tonne. According to Thormann *et al.* (2017), the other PGE and base metals, namely nickel and copper contribute a minor share of the basket price. In addition, most mining companies consider 4E grade for feasibility studies. Therefore, only 4E grade was considered in this study.

The Northern Limb ore characteristics are shown in Table 3.3, where it can be discerned that the platinum and palladium ratio is approximately 1:1. Even though metal volume varies from one operation to another, it was deemed appropriate to assume the general prill split for the Northern Limb for representativity.

Table 3.3 Northern Limb prill split

Source: Chigumira and Raymond (2017)

Metal type	Proportion (%)
Platinum (Pt)	43
Palladium (Pd)	48
Rhodium (Rh)	3
Gold (Au)	6

3.6.2 The PGE basket price

PGE mining operations receive revenues from multiple metals such as Pt, Pd, Rh, Au, Cu, and Ni. Therefore, it is important to consider a basket price for all metals with significant contribution to the revenue when calculating the value of the deposit. This study is based on a 4E grade as stated in Section 3.6.1, hence, the basket price for 4E was used. The basket price was calculated using Equation 3.4 where p_b, and P respectively represent the basket price, and individual metal price in United States (US) dollars per ounce and ore type proportion (%) for each metal. The input values were adopted from a Platreef 2017 feasibility study by Ivanhoe Mines as shown in Table 3.4. The estimated long-term 4E basket price (p_b) was calculated to be US$1041.5/oz. The grade for each block was calculated using Equation 3.5, where P and g represent proportion (%) and grade (g/t) for platinum, palladium, rhodium, and gold (4E PGE), respectively.

$$p_b = \sum(P_{Pt} \times p_{Pt}) + (P_{Pd} \times p_{Pd}) + (P_{Rh} \times p_{Rh}) + (P_{Au} \times p_{Au}) \quad 3.4$$

$$g_r = \sum(P_{Pt} \times g_{Pt}) + (P_{Pd} \times g_{Pd}) + (P_{Rh} \times g_{Rh}) + (P_{Au} \times g_{Au}) \quad 3.5$$

Table 3.4 Input values for economic block model construction

Source: Ivanhoe Mines Ltd (2017)

Parameter		Unit	Value
Price	Platinum (Pt)	US$/oz	1250
	Palladium (Pd)	US$/oz	825
	Rhodium (Rh)	US$/oz	1000
	Gold (Au)	US$/oz	1300
Refining cost		US$/t	39.77
Extraction cost		US$/t	34.27
Concentrating cost		US$/t	15.83
Recovery		%	82

If the BEV (i.e., value to be realised if a block is mined) is positive, the block is regarded as ore, whereas, if the value is negative, it is regarded as waste. If a block has a BEV of zero, it means that only the cost of mining that block will be realised if extracted and the block is a marginal block. All the blocks in the economic model have a potential of being included in the optimum stope layout if they add to the realisation of the maximum economic value of the orebody or to ensure the stope dimension constraint is not violated. As indicated later in Section 5.4 all BEV blocks were considered for evaluation to not disrupt the optimality of the solution generated. The next section discusses the validation process for the dual interchange algorithm.

3.6 Dual interchange algorithm validation process

Any new stope boundary optimisation algorithm can be tested against algorithms such as the floating stope algorithm, maximum value neighbourhood algorithm, Sens and Topal approach, and Sandanayake and Topal approach. Some of these algorithms have been incorporated into commercial mining software (see Table 3.5). For example, the MVN algorithm has been implemented through the SLO software. There is a Mineable Reserves Optimiser (also known as Mineable Shape Optimizer), which is a tool based on the floating stope algorithm in Datamine commercial software (Erdogan *et al.,* 2016). The robustness of the DIA that was developed in this study was validated by benchmarking its results against the MSO module due to availability of Datamine software to the author.

Table 3.5 Stope optimisation algorithms in commercial software

Source: Nhleko *et al.* (2018)

Algorithm	Extension	Software
Octree Division algorithm	None	GEOCAD (BONANZA)
Floating Stope algorithm	Mineable Reserve Optimiser	Datamine
Floating Stope algorithm	Vulcan stope optimizer	Maptek
Multiple Pass Floating Stope process	None	Datamine
Maximum Value Neighbourhood algorithm	MSSTOPE	MineSight

3.7 Summary of Chapter 3

The dual interchange algorithm was developed using the principles of the PSO algorithm and GA techniques. The dual interchange algorithm was applied to a synthetic Platreef orebody amenable to the longhole stope mining method. The resource block model was converted to an economic model using parameters related to metal price, grade, and costs of mining, processing, and refining of the broken ore. The dual interchange algorithm was validated by benchmarking its results against the MSO. The next chapter presents the development process of the dual interchange algorithm.

4 DEVELOPMENT OF A DUAL INTERCHANGE ALGORITHM

4.1 Overview of Chapter 4

Chapter 4 presents the parameter setting procedure of the PSO algorithm and the mining constraints considered in this study. The adaptation process for the PSO and GA algorithm for underground stope layout optimisation is explained in detail. The programming language used to code the DIA is mentioned and justified.

4.2 Parameters of the PSO algorithm

This section discusses the selection of PSO parameters and their input values to ensure that an optimal solution could be generated. The PSO input parameters include the velocity of a particle, acceleration coefficients, number of particles in the swarm and maximum number of iterations, whichever provides better performance for the algorithm. The performance of the algorithm is affected by factors such as inertia weight, velocity clamping and constriction.

4.2.1 Particle velocity

In defining particle velocity Nhleko and Musingwini (2019, p. 59) stated that, “Velocity is a vector used to determine the speed and direction of a particle within a swarm. The velocity component has the following characteristics which are namely, ability to provide information about a particle’s previous movement, cognitive and social information”. The generic equation used to calculate the particle’s velocity was shown previously in Equation 2.2.

Nhleko and Musingwini (2019, p. 59) further stated that, “The velocity component acts as an inertia component where it provides memory for the particle’s previous movement. This information prevents the particle from drastically changing its direction and biases it towards the current direction. The cognitive component measures the performance of the particle relative to its previous solutions. It measures the propensity of a particle to return to its previous best-known position. It is represented by $c_1 r_1^t (Pbest_{id}^t - P_{id}^t)$. The social component is denoted by $c_2 r_2^t (Gbest_d^t - P_{id}^t)$, which measures the performance of the particle relative to other particles in the swarm.

Where parameters c_1 and c_2 represent the acceleration coefficients that regulate the length when flying to the most optimist individual particle and to the most optimistic particle of the whole swarm at time, t. Both r_1 and r_2 represent random factors ranging from 0 to 1. The parameter P_{id}^t represents the d-dimension quantity of the position of particle, i at time, t . The parameter $Pbest_{id}^t$ is the most optimistic position of the individual particle, i at time, t and d-dimension. The parameter $Gbest_d^t$ is the d-dimension quantity of the swarm at its most optimistic position at time, t.

This component ensures that each particle in the swarm draws towards the best position found in the search space (swarm)". The magnitude of the velocity needs to be managed to guarantee that the particles remain within the search space (Bai, 2010; Talukder, 2011), otherwise the particle will be too far from the search space, or the solution may converge to and be trapped in a local optimum. This is achieved by using parameter v_{id}^k as explained previously in Section 2.4.8.

The inertia weight and constriction factor are the commonly applied parameters to address the drawbacks of the PSO of converging to and getting trapped in local optima. Some researchers have applied mutation operators on *Pbest* and *Gbest* particles using different techniques to prevent the PSO from being trapped in local optima. The analysis of mutation operators is beyond the scope of this study since a standard PSO algorithm is being applied.

4.2.2 Acceleration coefficients

Acceleration coefficients maintain the stochastic influence of the cognitive and social capability of a particle by combining the acceleration coefficients with random values r_1 and r_2 as explained previously in Section 2.4.8. Cognitive and social components provide a level of confidence that a particle has on the solution it has produced and the solutions generated by its neighbours (Nhleko and Musingwini, 2019).

If the coefficient is too small, a particle may be too far away from the target field. If a coefficient that is too large it may cause particles in the swarm to fly to the target field suddenly or fly beyond the target field. The values of c_1 and c_2 can control the speed of the particles flying in the search space; hence, the solution generated by the particle will be an optimum solution. Cai *et al.* (2009) stated that empirical studies show that better solutions are generated when c_1 is equal to c_2 and they are both equal to two

(2). There are other options of initialising c_1 and c_2; the choice of the assigned values depends on the objective function of the optimisation process. However, incorrect initialisation may lead to divergence or cyclic behaviour. Table 4.1 shows the different options and their effect on the optimisation problem (Talukder, 2011).

Table 4.1 Acceleration coefficients and their effects on the optimisation solution

Source: Nhleko and Musingwini (2019)

Option	Qualities
$c_1 = c_2 = 0$	Particles continue at current speed and only stop once search space boundary has been reached. Velocity is calculated as $v_i^{t+1} = v_i^t$.
$c_1 > 0$ and $c_2 = 0$	Particles are independent and velocity is updated by $v_i^{t+1} = v_i^t + c_1 r_1^t (Pbest_i^t - P_i^t)$. There is no global best component, thus this option cannot work for global optimisation problems.
$c_1 = 0$ and $c_2 > 0$	Particles are only attracted to the global best position. $v_i^{t+1} = v_i^t + c_2 r_2^t (Gbest - P_i^t)$.
$c_1 = c_2 = 2$	Particles are attracted to the average values of the *Pbest* and *Gbest*.
$c_1 \gg c_2$	Each particle is strongly influenced by its *Pbest* position resulting in excessive wandering and may lead to non-convergence.
$c_2 \gg c_1$	Particles are strongly influenced by *Gbest* position resulting in premature convergence.

4.2.3 Inertia weight

Inertia weight is an agent that controls the speed of the current particle by weighing the contribution of the velocity of the particle's previous position. Equation 2.2 is therefore, modified to include inertia weight (w) as shown in Equation 4.1, which controls the exploitation and exploration capabilities of a particle in the search space. This parameter allows the convergence of the swarm to be more accurate and efficient than that of the original PSO approach. Table 4.2 shows the different inertia weight values that can be applied and their effects on the optimisation solution (Bai, 2010; Talukder, 2011).

$$v_{id}^{t+1} = w v_{id}^t + c_1 r_1^t (Pbest_{id}^t - P_{id}^t) + c_2 r_2^t (Gbest_d^t - P_{id}^t) \quad 4.1$$

Inertia weight can be applied as a fixed value, however dynamically changing values has produced superior results because this allows better control exploitation and exploration of the search space. The commonly used dynamically changing inertia weight decreases from 0.9 to 0.4 over the entire search space (Lim *et al.*, 2009; Bai, 2010; Talukder, 2011; Imran *et al.*, 2013; Alam, 2016). Therefore, assigning of large

values in the early stages of searching allows for short exploration time and more time is dedicated to the exploitation of the search space. Initialising the search process using a high value of the inertia weight guarantees that as w decreases, the speed decreases to search the entire solution space (Bai, 2010). The shortcoming of using a dynamically changing inertia weight is that once it decreases from 0.9 to lower values, it cannot increase to recover its exploitative function, thus, constraining the swarm from searching new areas (Talukder, 2011). The inability to increase w should not be a problem for stope layout optimisation since the search space is predefined before the implementation of the algorithm.

Table 4.2 Inertia weight options and their effects on the optimisation solution

Source: Nhleko and Musingwini (2019)

Option	Qualities
$w = 0$	Particle moves without knowledge of the previous velocity in each step, i.e., velocity component diminishes in Equation 4.1.
$w = 1$	There is a lack of searching ability throughout the search space.
$w \leq 1$	Quick changes in particle direction because there is little velocity influence from the previous step.
$w \geq 1$	Velocity increases over time and there is hardly any change in particle direction resulting in particle divergence.

4.2.4 Constriction factor

The constriction factor (χ) was introduced to guarantee particle convergence and prevent collapse in the search space resulting in the swarm being trapped in local optima. When the constriction factor is applied, the search procedure is implemented by using Eigen value analysis such that different regions are searched efficiently while preventing premature convergence (Lim *et al.,* 2009; Bai, 2010; Talukder, 2011). Equation 2.2 is therefore, modified as shown in Equation 4.2 when a constriction factor is applied.

$$v_{id}^{t+1} = \chi[v_{id}^{t} + c_1 r_1^t (Pbest_{id}^{t} - P_{id}^{t}) + c_2 r_2^t (Gbest_{d}^{t} - P_{id}^{t})] \quad 4.2$$

$$\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad 4.3$$

$$\varphi = c_1 + c_2 \quad 4.4$$

$$\varphi > 4 \qquad 4.5$$

Where φ is the sum of c_1 and c_2. Lim *et al.* (2009) and Alam (2016) stated that to guarantee the stability of the optimisation process, the value of φ must be greater than four (4). However, as the value of φ increases, χ decreases resulting in reduced diversification, resulting in a high computational time requirement. Generally, when $\varphi = 4.1$ then $\chi = 0.729$ (where $c_1 = c_2 = 2.05$), and the solutions generated have proved acceptable. Bai (2010) stated that when χ is used instead of w, the convergence to an optimum solution is much quicker. However, when appropriate values for w, c_1 and c_2 are selected, the results of using χ or w are identical. The major drawback of using a constriction factor instead of inertia weight is that it only considers the dynamic behaviour of one particle and the effect of the interaction among particles in the swarm. The equations were developed based on a static set of *Pbest* and *Gbest* positions. On the contrary, *Pbest* and *Gbest* positions change during the optimisation process in the standard PSO algorithm (Lim *et al.,* 2009; Bai, 2010; Talukder, 2011). Therefore, inertia weight was selected for controlling the speed of the particle for implementation in the DIA algorithm.

4.2.5 Initialisation

According to Nhleko and Musingwini (2019), appropriate initialisation will yield a balance between exploitation and exploration of the swarm during the search process. The common approach is to apply a uniform distribution for initialisation to ensure diversity in the amount of space covered and particle distribution. Optimal initial particle distribution in the search field must be confined between minimum and maximum ranges of the search space dimensions. The initial velocity of the particles must be zero to ensure random positioning and moving directions for particles. If non-zero velocity values are selected, care must be exercised to ensure that the particles do not spiral away from the feasible region during the optimisation process. Consequently, large number of iterations will be required before particles settle on the best solution (Talukder, 2011). According to Alam (2016), if a non-zero velocity is used, it should be within 10% of the particle position to guarantee efficient exploration and exploitation of the search space.

4.2.6 Number of iterations

The number of iterations required to obtain a good or near optimum solution depends on the objective function. Therefore, it is important to note that if the number of iterations is too small, the optimisation process may terminate prematurely, consequently, failing to generate an optimum solution. On the contrary, if the number of iterations is too large, there may be unnecessary computational time added, resulting in long solution times if convergence occurs. The number of functions evaluated for velocity adjustment, $Pbest, Gbest$ and the objective function evaluation during each iteration is equivalent to the number of particles in the swarm (Talukder, 2011). According to Alam (2016), the most used number of iterations ranges between 500 and 10000. Sandanayake (2014) applied 500 iterations for the stope boundary optimisation problem.

Given that the amount of computational time required increases exponentially as the number of constraints and number of iterations increase, 200 iterations were applied in this study. The DIA creates 25 stope layouts per run using the PSO algorithm which are modified through 200 iterations each, thus generating 5000 possible stope layout solutions. In addition, the GA modifies each stope layout several times until there is no improvement on the SLEV of the stope layout. A higher number of iterations may result in failure of the algorithm to converge to an optimum solution considering the number of constraints applied. Furthermore, a higher number of iterations require high computational power given the number of constraints applied in this study.

4.2.7 Stopping criteria

According to Talukder (2011) as cited in Nhleko and Musingwini (2019), stopping criteria are used to terminate the search process and there are three commonly used termination criteria which are:

- The maximum number of iterations which ensures that the algorithm terminates when the maximum number of iterations has been reached. If the number of iterations is small, the process may terminate prematurely.
- No significant improvement in the solution as a condition to ensure that if the average change of particle position is very small or if the average velocity of

the particle is approximately zero over several iterations, the search process terminates.

- Normalised swarm radius as a condition that ensures the search process terminates if the normalised swarm radius is approximately zero. Equations 4.6 to 4.8 show how to introduce this feature.

$$Normal\ swarm\ radius = Radius_{norm} = \frac{Radius_{max}}{Diameter} \quad 4.6$$

$$Radius_{max} = \|X_m - Gbest\| \quad 4.7$$

$$\|X_m - Gbest\| = \|X_i - Gbest\|, \qquad m, \forall\ i = 1, .., n \quad 4.8$$

As discussed previously in Section 4.2.6, the number of iterations was selected as a stopping criterion to terminate the algorithm.

4.3 Application of genetic algorithm

The GA is a metaheuristic algorithm that imitates the process of natural selection where the fittest individuals in a population are selected for reproduction of offspring in the next generation. The offspring inherit the genes of the parents. Therefore, the generated offspring will be better than its parents in terms of fitness and has a better chance of surviving. This reproduction process iterates until the criterion for termination is satisfied, consequently, producing a generation of fittest individuals in a new generation. Termination occurs when the population has converged meaning that there is insignificant difference between previous generation and the current generation in terms of fitness.

The generic stages of evolution in the GA approach are (Kurniawan, *et al.,* 2014):

- Step 1: Initial population selection – randomly select individuals to form a population.
- Step 2: Computation of the fitness function – valuate the objective function, ensuring that constraints are not violated.
- Step 3: Selection of the fittest parents – selection of the candidate parents for reproduction based on the fitness criterion. The best number of parents and

offspring populations are selected to form a new population by executing the following operations:

  - Crossover – allows for gene exchange between selected parents to produce an offspring of higher fitness than its parents.
  - Mutation – a certain number of individuals in the parent population are mutated and offspring population carrying hybrid genes from both parents is generated.
  - Elimination – individuals with best fitness function are identified and stored while those with poor fitness are eliminated.

- Step 4: Termination – stopping criterion is evaluated, if not satisfied, Step 2 is executed. Otherwise, once a criterion to stop the iterative process is met then Step 5 is executed.
- Step 5: Best fitness individuals are presented.

4.4 Mining constraints

This section discusses the mining constraints considered for the stope layout generation and how they were applied. These constraints are stope dimension, dilution, stope location (level), non-overlapping stopes and unique stopes. These constraints are important for the practical extraction of the orebody. The notations in Table 4.3 are used to explain the flow charts for mining constraints consideration.

Table 4.3 Notations of parameters used in the dual interchange algorithm

Notation	Explanation
X	location of a block along the x-axis
Y	location of a block along the y-axis
Z	location of a block along the z-axis
X_{max}	outmost location of a block along the x-axis
Y_{max}	outmost location of a block along the y-axis
Z_{max}	outmost location of a block along the z-axis
B_{xyz}	Block identifier in terms of x, y, and z location
nx_s	number of blocks consituting a stope along the x-axis
ny_s	number of blocks consituting a stope along the y-axis
nz_s	number of blocks consituting a stope along the z-axis
s	stope
t	iteration
p_s	set of all positive stopes
q_s	set of all positive stopes in a level
g_s	grade of stope s
ρ_s	density of stope s
m_s	tonnage of stope s
OLL_s	optimsed level stope layout
SEV_s	stope economic value for stope s
$SLEV$	stope layout economic value
OSL	optimised stope layout

4.3.1 Stope dimension considerations

There are several stope dimension combinations that can be used to generate stope layouts. However, the selection of the appropriate stope dimension is constrained by, *inter alia,* geological conditions, geotechnical environment, and type of equipment to be used. Therefore, it is important to consider these factors when determining the appropriate stope dimension. In this study various factors were considered such as the strike, dip, permissible hydraulic radius based on an existing prefeasibility study in the Northern Limb of the Bushveld Complex and operating mining constraints such as drilling machine capabilities.

The dual interchange algorithm randomly generates various stope dimensions given the input constraints of minimum and maximum dimensions of the stope width. Thereafter, it generates an appropriate stope dimension to be used for the optimisation process throughout the block model. It should be noted that the mine planner or the user can override the random generation of appropriate stope dimension by selecting the option to input the preferred stope dimension under the fixed tab in the interface as shown in Figure 4.1.

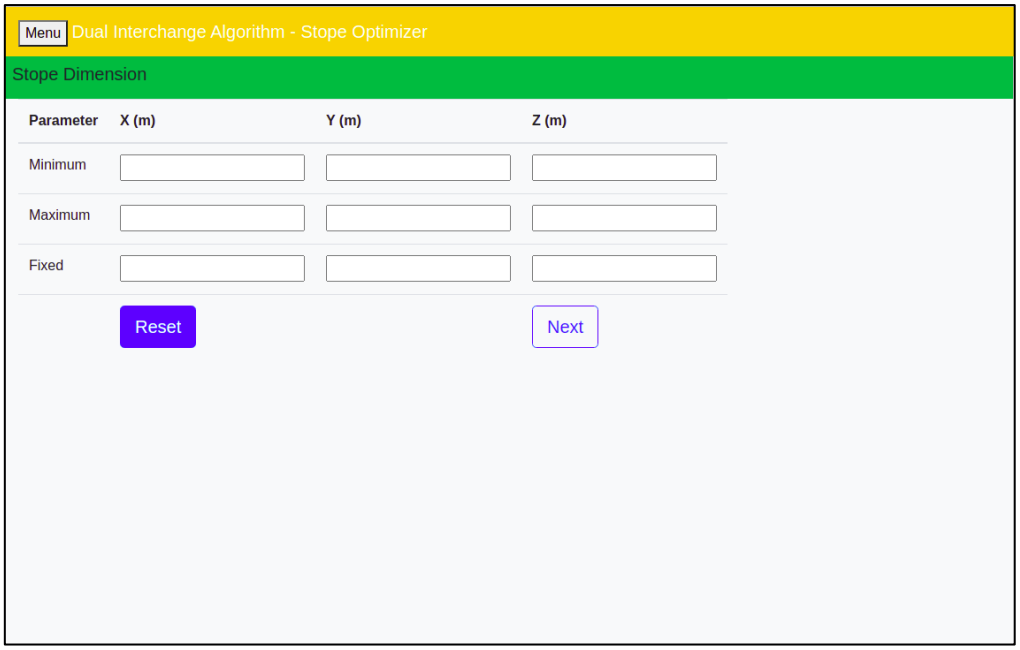


Figure 4.1 User interface for the stope dimension specifications

4.3.2 Stope generation process

The regularised economic block model is used as an input parameter for stope generation. Every block in the economic model is first assigned a unique identifier in terms of its X, Y, and Z location to facilitate the stope creation. The number of blocks along each axis determines the size of the stope in that specific direction. For example, if each block is 10m in X, Y and Z axes and $XINC = 2, YINC = 2, ZINC = 2$, the stope dimension will be $20m\ x\ 20m\ x\ 20m$. In this configuration, the stope size is fixed throughout the search space. Every possible stope is generated with its economic value being calculated by the summation of all the economic values of the blocks constituting the stope called the stope economic value, or SEV. The output of this process is a set of stopes that either have a positive or negative value depending on the sum of the values of the blocks in that stope.

The stope generation process ensures that all blocks in the economic block model are considered for inclusion in the final stope layout irrespective of the block economic value. This assurance is achieved by allowing each block in the model to generate a stope or a set of stopes. Practically, however, a block can be mined only once. Thus, if a block generates a set of stopes, only a stope yielding the highest stope economic value or SEV from a block is selected.

Figure 4.2 shows the flow chart for stope generation along the x, y, and z axes. After the stopes are generated, if the economic value of the stope was equal to or greater than zero, it was included in the optimisation process. However, if the stope economic value was negative it was excluded from the stope layout optimisation process since it does not add value. This elimination process reduces the number of stopes in the economic block model, subsequently, reducing the solution time for generating an optimum stope layout.

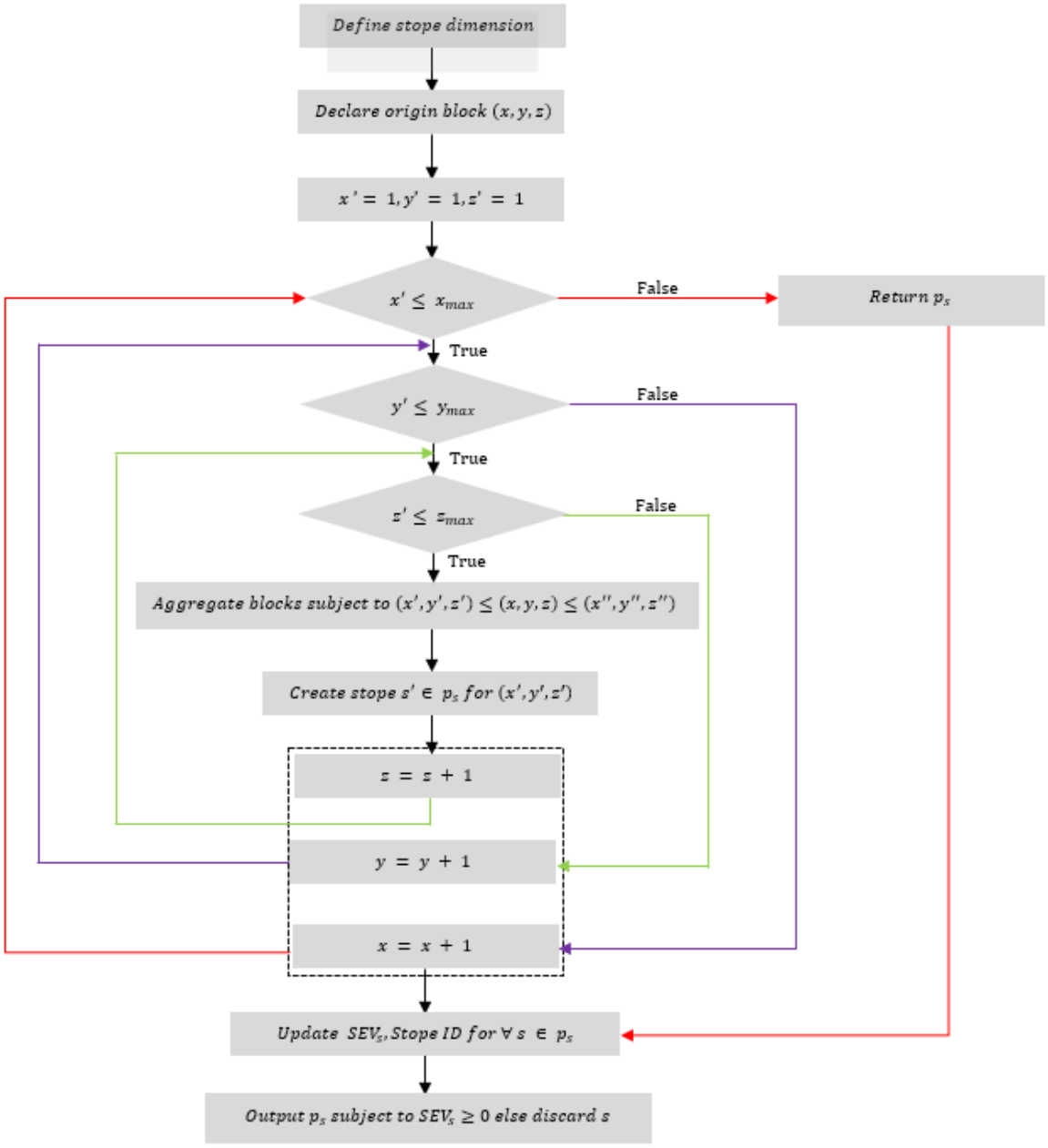


Figure 4.2 Flow chart of the stope generation process

(Adapted from Sandanayake, 2014)

4.3.3 Level constraint

The optimisation process is initiated randomly in the selection of stopes. The first stope (starting stope) selected is floated throughout the economic block model. Once the stope is selected, the height of the stope is fixed throughout the optimisation process. The selected (origin) stope is then floated along the x and y axes to determine the optimum stope combinations per level using the principles of the GA. Thereafter, the search process continues either upward or downward along the z-axis depending on the position of the starting (origin) stope to complete search process in the economic block model. Consequently, an optimum stope layout is generated subject to the constraints and the stopping criterion. Figure 4.3 shows a 2D example of the optimisation process along the z-axis where the stope dimension is set at 2 x 2 blocks. The bottom row of blocks along the z-axis are not included because they do not satisfy the stope dimension constraint. A schematic illustration of the stope generating algorithm incorporating the level constraint is depicted in Figure 4.4.

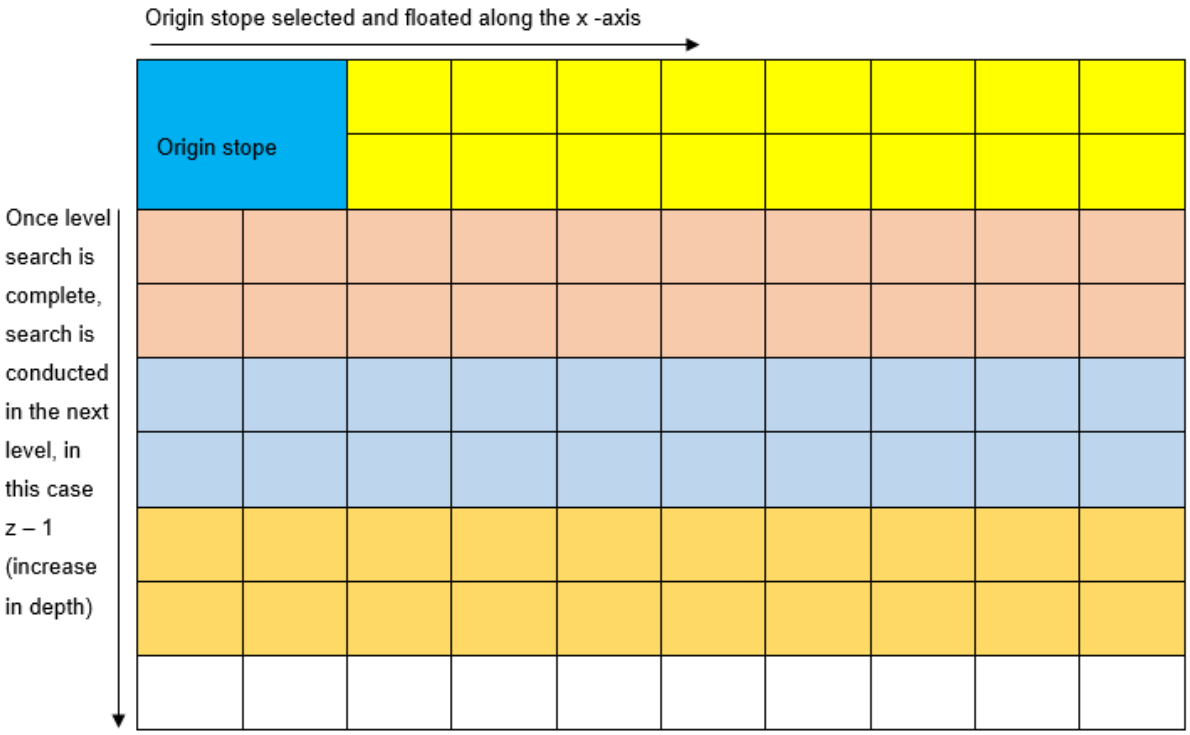


Figure 4.3 Stope optimisation along the z-axis in an x-z plane

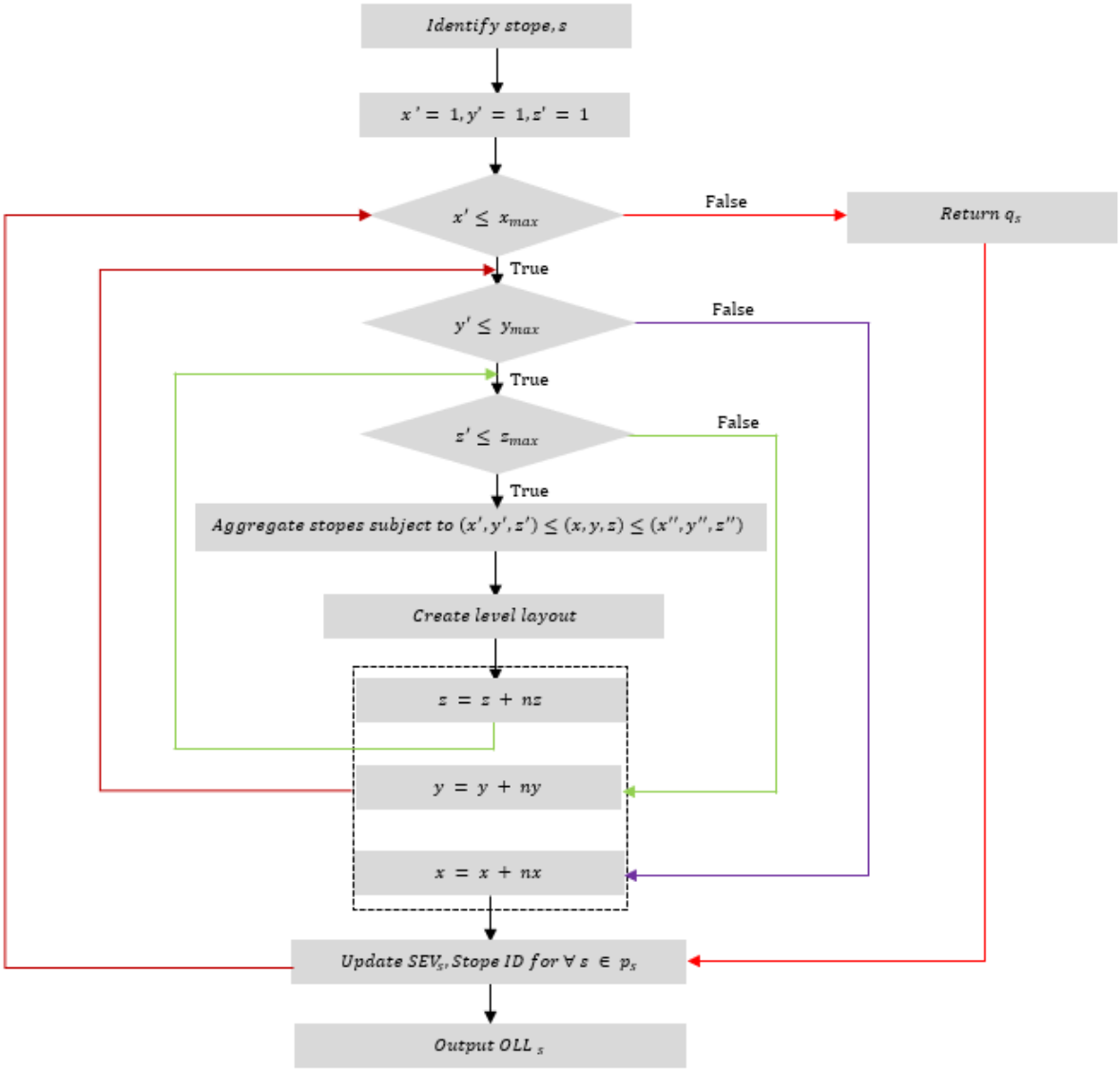


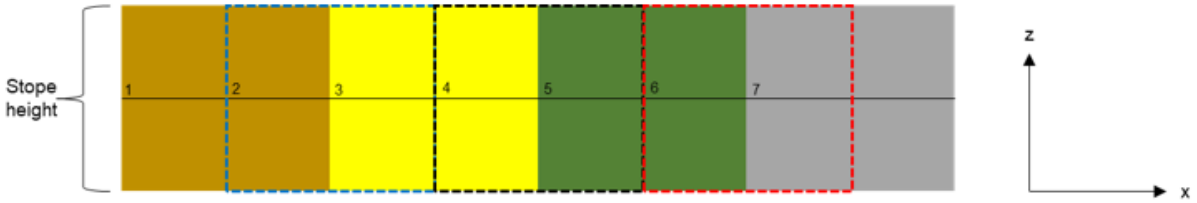
Figure 4.4 Level constraint incorporation in the optimisation algorithm

(Adapted from Sandanayake, 2014)

4.3.4 Non-overlapping and unique stopes

Once the origin stope has been selected it is floated along the x and y axes to generate a layout for that level. However, there are numerous possible combinations of stopes that need to be evaluated to determine the optimum combination of stopes. Figure 4.5 shows 2 x 2 blocks in 2D for a stope dimension that is floated along x-axis and y-axis in a level to determine optimum stope combinations. There are six unique stope combinations that are possible given the length of 8 blocks in a row. The selection of any of the possible combinations is governed by the SLEV, where the combination of

stopes with the highest sum of the stope economic values is desired irrespective of the number of stopes included.



Option	Possible stope combinations
A	1, 3, 5, 7
B	1, 4, 6
C	1, 4, 7
D	1, 3, 6
E	2, 4, 6
F	2, 4, 7
G	2, 5, 7

Figure 4.5 Example of a plan view of possible stope combinations in 2D

The generation of these possible stope combinations may be a cumbersome activity and may increase the solution time exponentially given the possible combinations when there are hundreds of thousands of stopes to be considered in a level, consequently, making the optimisation problem intractable. This constraint was eliminated by adapting the principles of the genetic algorithm in generating possible stope combinations and selecting the highly ranked combination for inclusion in the final stope layout.

Figure 4.6 presents the flow chart used to apply the genetic algorithm principles in overcoming overlapping stopes and ensuring unique stopes are included to avoid stope duplication in the generated stope layouts. The first step was to input all stopes in a specific level that is being optimised during a specific iteration of the PSO. From the set of stopes in a level, a population was generated randomly for input into the GA approach. Thereafter, the objective function was valuated to maximise the stope economic value in that level. Elitism was applied to ensure that best stopes are selected for inclusion in the next generation to avoid lack of diversity and a premature over-fitting in the optimisation process. However, only a small number of stopes were selected under the elitism process. The best fit stopes were selected as parents and mated to generate offspring through the crossover and mutation processes.

Generations (i.e., stopes) with inferior fitness were eliminated from the layout. If the termination criterion was not met, the iteration continued until termination was satisfied. Subsequently, the best stope layout for the level was stored. The optimisation continued to the next level along the z-axis. This process continued until the whole economic block model had been optimised. The individual level stope layouts were then aggregated to form an optimised stope boundary layout for the whole mining block model.

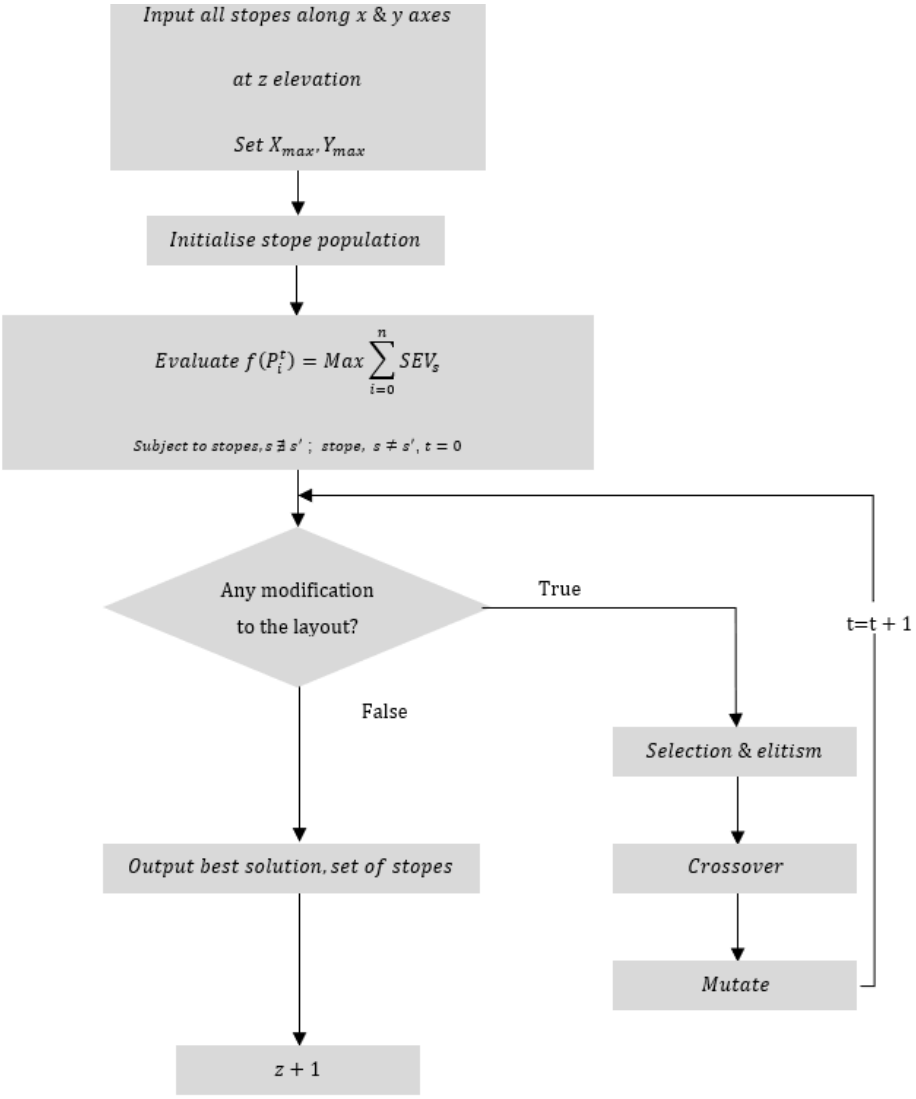


Figure 4.6 Non-overlapping and unique stope constraints incorporation

The GA algorithm uses the set of positive stopes as input to satisfy the non-overlap and unique constraint for the level optimisation stage of the DIA approach. Thereafter, the optimisation process is initialised as shown in the pseudocode starting from Step

1 and terminates once the stopping criterion is met (See Figure 4.7). The optimum solution per level is generated in Step 10.

Begin

1) Input stopes in level Z_0
2) Initialise input parameters

   Set:

   Generation = 0

   Population = 100

   Best fitness = 0.0

   Mutation = 0.1

   Elitism = 2

   Random = [0,1]
3) Initialise population
4) Evaluate objective function
5) Select parents
6) Select elite individuals
7) Crossover
8) Mutate
9) While layout modification == True:

   Go to step 5

   Else:

   Go to step 10
10) Output best level population

End

Figure 4.7 Pseudocode for GA adaptation

4.4 Adaptation of the PSO algorithm for stope optimisation

The basic PSO algorithm was adapted for optimisation of the underground stope layout. Table 4.4 depicts what the PSO parameters represent in a mining context.

Table 4.4 Adaptation of the PSO algorithm for optimising underground stope layout

Generic PSO algorithm (Adapted from Qi *et al. (2016)*	**Adaptation of PSO for a stope layout environment** (Nhleko and Musingwini, 2019)
Swarm: search space for particles	Set of positive stopes generated
Particle position, i, (P_i^t), at iteration, t	Stope layout, i, (P_i^t), at iteration, t
Particle best position, $Pbest_i^t$, at iteration, t	Personal best stope layout, P_i, at iteration t
Moving speed of a particle, (V_i^t), at iteration, t	Rate of change in stope layout, P_i, to resemble the current best layout solution
Particle position, i: $\left(P_i^{t+1}\right)$, at iteration, $t+1$	Stope layout, i: $\left(P_i^{t+1}\right)$, at iteration, $t+1$
Swarm best position, $Gbest^t$, at iteration, t	Global best stope layout from the set of layouts generated at iteration, t
w, c_1, c_2, r_1 and r_2 are the control parameters for the rate of change to prevent convergence on local optima and non-convergence of the algorithm during the search process.	

4.4.1 Dual interchange algorithm anatomy

The algorithm uses the set of positive stopes, s, as input. Thereafter, the optimisation process is initialised as shown in the pseudocode starting from Step 1 and terminates once the stopping criterion is met (see Figure 4.8). The optimum solution is generated in Step 10.

Begin

(1) Initialise input parameters

Set:

w = 0.8

c1 & c2 = 1.49445

r1 & r2 = [0, 1]

t = 0

N = 500

D = 5

(2) Initialise the swarm

Initialise the particles P(t) at t = 0

(3) Evaluate objective function:

$$f(P_i^t) = Maximise \sum V_s \qquad for\ s = s',\ s'', \dots, s^n$$

$$V_s = \sum B_{xyz} \qquad for\ B_{xyz} = B_{xyz}^1, B_{xyz}^1, \dots, B_{xyz}^n$$

Subject to these constraints:

Stope dimension

Level position

$$GA \begin{cases} Non-overlapping\ stopes \\ Unique\ stopes \end{cases}$$

(4) Set $f(P_i^t) = Pbest_i^t$ and $maximum\ f(P_i^t) = Gbest^t$

(5) Start

t= t+1

Update w

Update velocity

Update position

Go to step 3

(6) If $f(P_i^{t+1}) > Pbest_i^t\ then$

$f(P_i^{t+1}) = Pbest_i^{t+1}$

Else:

$Pbest_i^{t+1} = Pbest_i^t$

(7) If $Pbest_i^{t+1} > Gbest^t\ then$

$Pbest_i^{t+1} = Gbest^{t+1}$

Else:

$Gbest^{t+1} = Gbest^t$

(8) End

(9) While stopping criterion is unsatisfied:

Go to step 5

Else:

Go to step 10

(10) Output Gbest (stope layout)

End

Figure 4.8 Pseudocode for the dual interchange algorithm

Figure 4.9 presents a flowchart for the dual interchange algorithm where the input dataset is the set of positive stopes from the economic block model. Furthermore, Figure 4.10 shows a simple integration of the PSO and GA algorithms for the

optimisation of stope layout for an underground mine, showing how the PSO and GA iteratively interchange outputs as inputs until termination is achieved.

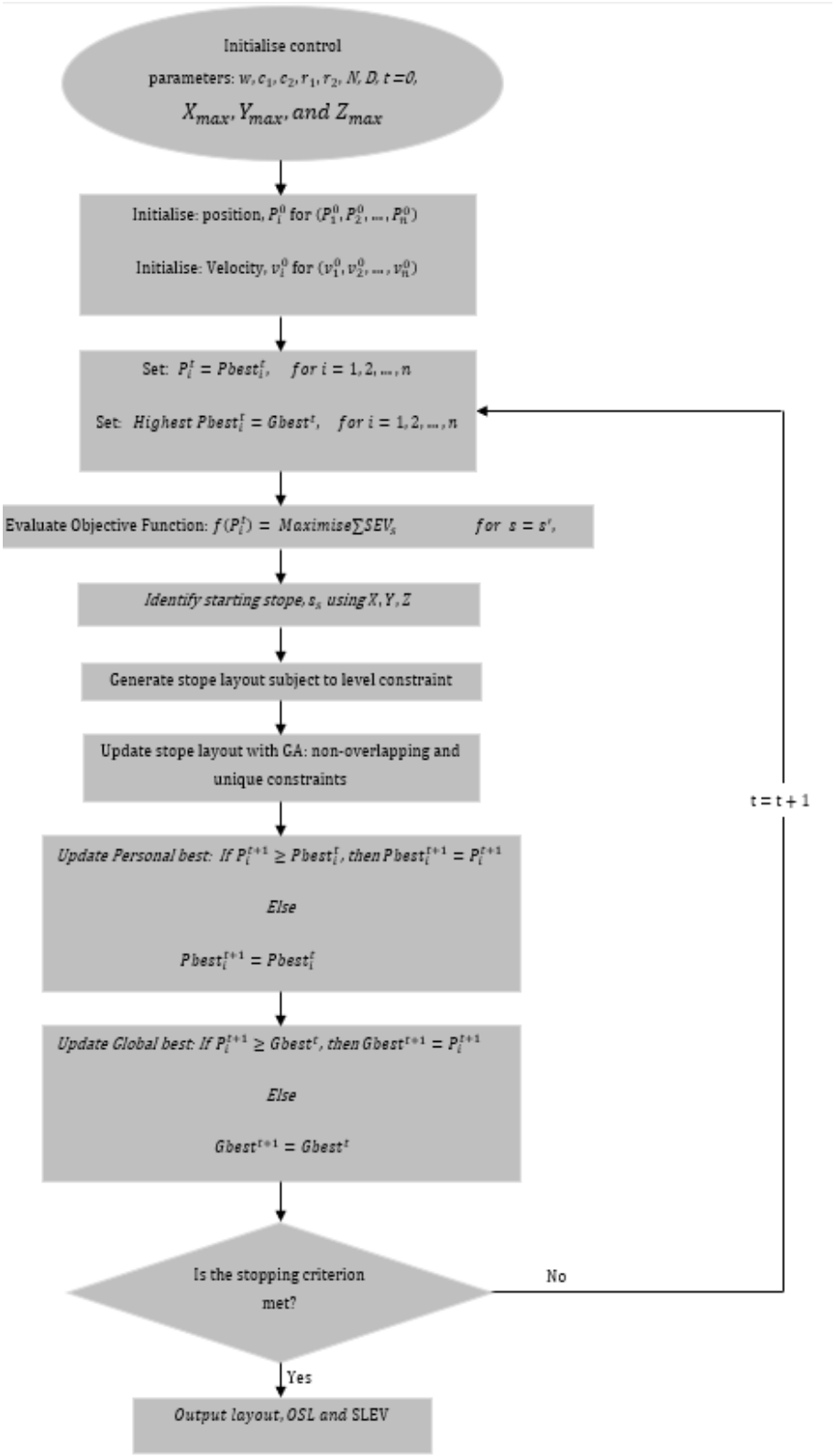


Figure 4.9 Dual interchange algorithm flowchart for stope layout optimisation

Figure 4.10 shows that the DIA interchanges between the modified PSO and GA techniques during the optimisation process. The DIA optimisation process for the stope boundary problem is as follows:

- Step 1, in the PSO, levels are generated based on the stope height constraint and orebody vertical extension.
- Step 2 evaluates the objective function which is to maximise the economic value of the mineral deposit.
- Step 3 generates stope layouts given the number of particles as a constraint.
- Step 4, in the GA, from the number of levels generated in Step 1, a level is selected for the optimisation process with the stope layouts as input.
- Step 5 evaluates the objective function.
- Step 6, for each generation, stope layouts are randomly selected from the population where the selection of each solution is related to its fitness value. The GA continues with the optimisation process following the crossover, mutation, and elimination process.
- Step 7 evaluates the termination criterion, if not satisfied Step 5 is repeated, otherwise the best level stope layout is selected and stored.
- Step 8 the GA evaluates if all the levels have been optimised, if not it selects the next level for optimisation and go to Step 4. If all levels have been optimised, the DIA interchanges to the PSO.
- Step 9, in the PSO, the termination criterion is evaluated. If the termination criterion is not satisfied Step 2 is executed, otherwise Step 10 is executed.
- Step 10 the PSO aggregates all the *Gbest* optimised level stope layouts and generates the global best stope layout for the mineral deposit.

The architecture of the DIA is such that it will best handle variable stope width which is ideal because the orebody periphery contour is not homogeneous. Therefore, varying stope width has the benefit of selective mining resulting in the reduction of the amount of internal dilution in the optimised stope layout. The developed DIA was applied to a synthetic Platreef economic block model to generate numerous possible optimal solutions. The solution with the highest SLEV was selected as the optimum solution for this orebody. To validate the results obtained from the DIA, it was

compared against the solutions generated by the MSO module in Datamine commercial software.

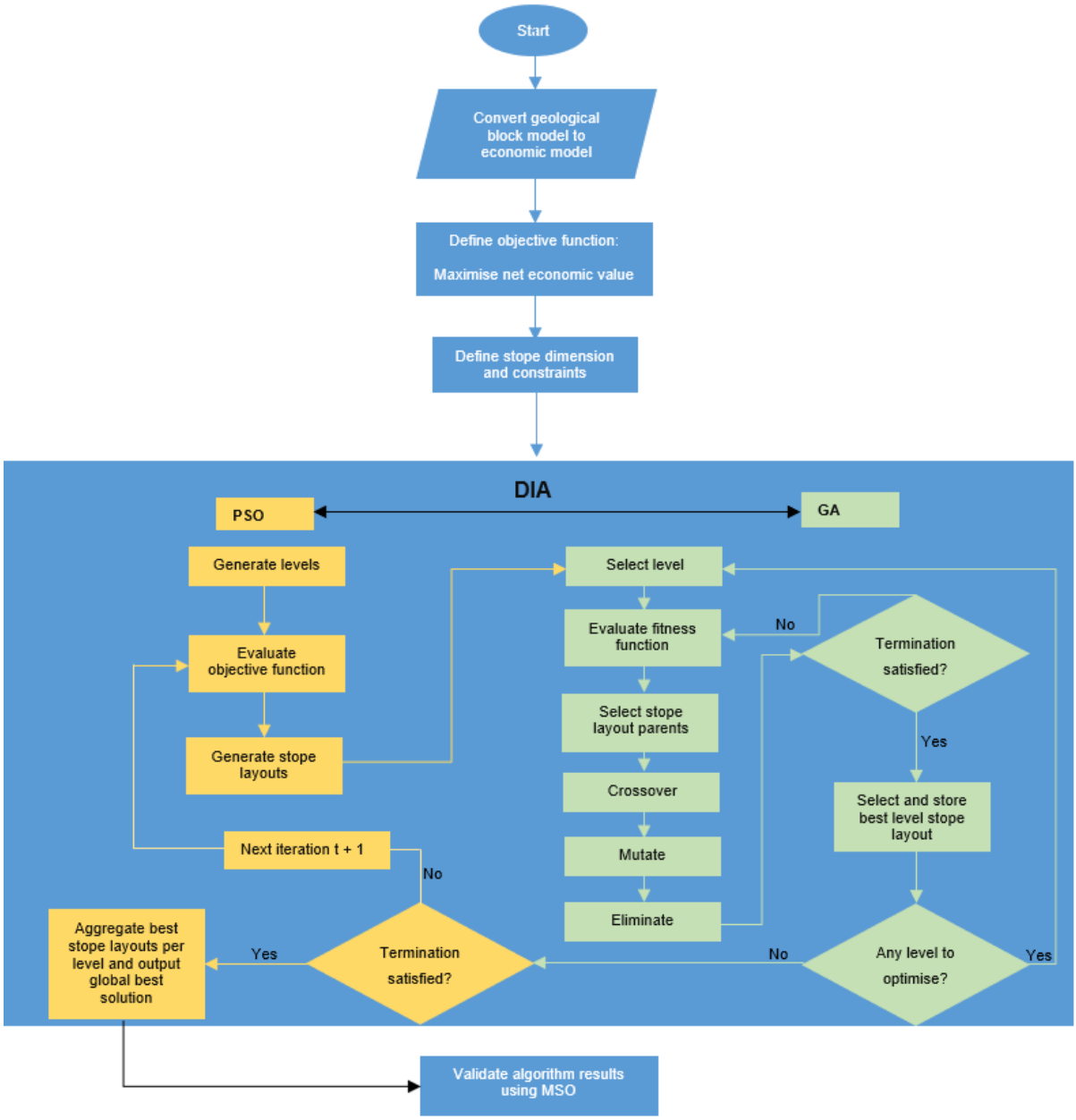


Figure 4.10 Schematic illustration of the proposed dual interchange algorithm

The dual interchange algorithm was coded using Python 3.7 programming language using an integrated development environment (IDE) called Spyder IDE which is an open-source platform. To run this programming on a computer, it needs to have Anaconda Distribution and Microsoft 2013 installed on it. The Python 3.7 programme allows for the development and application of a Graphical User Interface (GUI). Python programming allows for the implementation of the DIA and execution of the application

in a Windows-based computer. The output from this model is converted into a Microsoft Excel compatible (csv) format for further analysis (see Table 4.5). The Microsoft Excel file can be imported into Datamine software to create a block model to visualise the optimised stope layout.

Table 4.5 Sample of the csv file output for the global best layout

XC	YC	ZC	XINC	YINC	ZINC	XMORIG	YMORIG	ZMORIG	NX	NY	NZ	Stope Density (t/m^3)	SEV (US $)
-7642096	742846	-1092	20	40	25	-7642096	742846	-1092	20	40	25	3.1	194965783.30
-7642398	742733	-1092	20	40	25	-7642398	742733	-1092	20	40	25	3.1	295202997.80
-7642271	742781	-1092	20	40	25	-7642271	742781	-1092	20	40	25	3.1	235883272.30
-7642107	742748	-1092	20	40	25	-7642107	742748	-1092	20	40	25	3.1	191817722.20

4.4.2 Dual interchange algorithm source code

The source codes for the database, creation of economic block model, stope generation, introduction of constraints and application of the dual interchange algorithm are presented in Appendix 8.1. The next section presents the application interface for the dual interchange algorithm.

4.5 Application programming interface and output file

A user-friendly GUI was developed to allow the implementation of the dual interchange algorithm. The interface was developed using the Spyder IDE. Appendix 8.2 (Figures 8.1 to 8.5) presents the step-by-step instructions about how to use the interface. The first step is to upload a geological block model in a Microsoft Excel csv file format as shown in Figure 4.11.

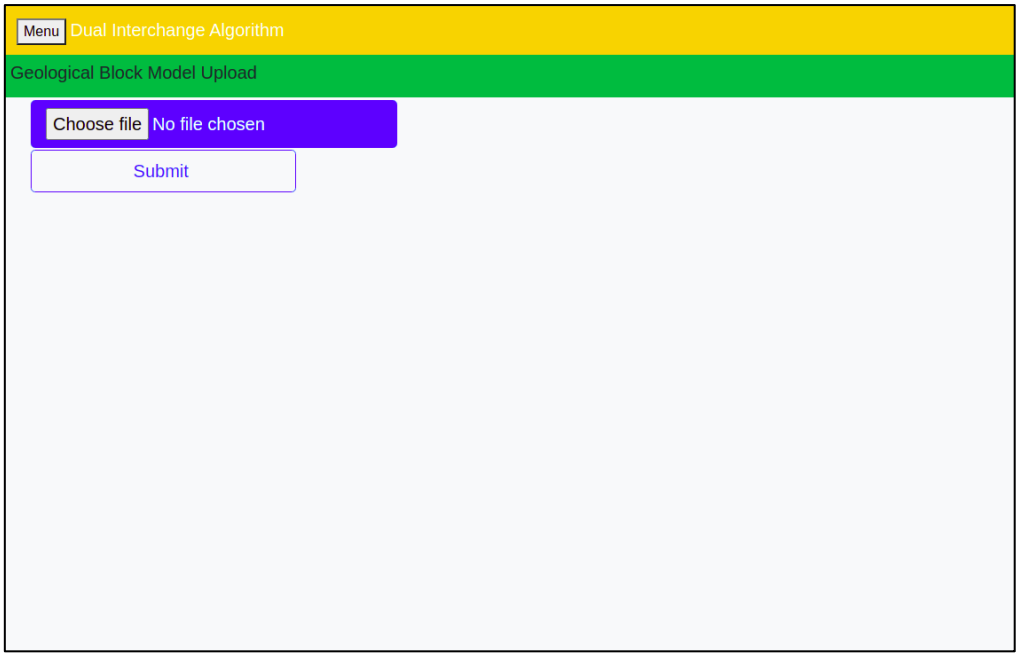


Figure 4.11 User interface for the dual interchange stope layout optimisation algorithm

4.6 Summary of Chapter 4

This chapter presented the selection of PSO parameters and the structure of the dual interchange algorithm. The mining constraints considered to ensure practical applicability of the dual interchange algorithm were also discussed. It is impractical to accommodate all mining constraints without making the optimisation problem intractable, thus, only the stope dimension, hanging wall and footwall orientation, level position, unique and non-overlapping stopes were accounted for as mining constraints. The dual interchange algorithm pseudocode and source code were also presented. The next chapter presents the application of the DIA to a synthetic Platreef block model and using results obtained for the validation process of the DIA.

5 RESULTS AND VALIDATION OF THE DUAL INTERCHANGE ALGORITHM

5.1 Overview of Chapter 5

There are several algorithms that are used for optimisation of stope layout for underground mines as discussed previously in Section 2.4. However, none of them can guarantee optimum solutions in three-dimensional space. The dual interchange algorithm was developed and applied to the synthetic Platreef deposit block model to evaluate its performance and benefits. The dual interchange algorithm was validated through comparing its results against solutions generated using the MSO module in Datamine commercial software. Figure 5.1 shows the validation process used in this study. Once the results from both the dual interchange algorithm and MSO were obtained, they were tabulated where possible for ease of comparison. The subsequent sections present comparison results for stope layout optimisation between MSO and the dual interchange algorithm.

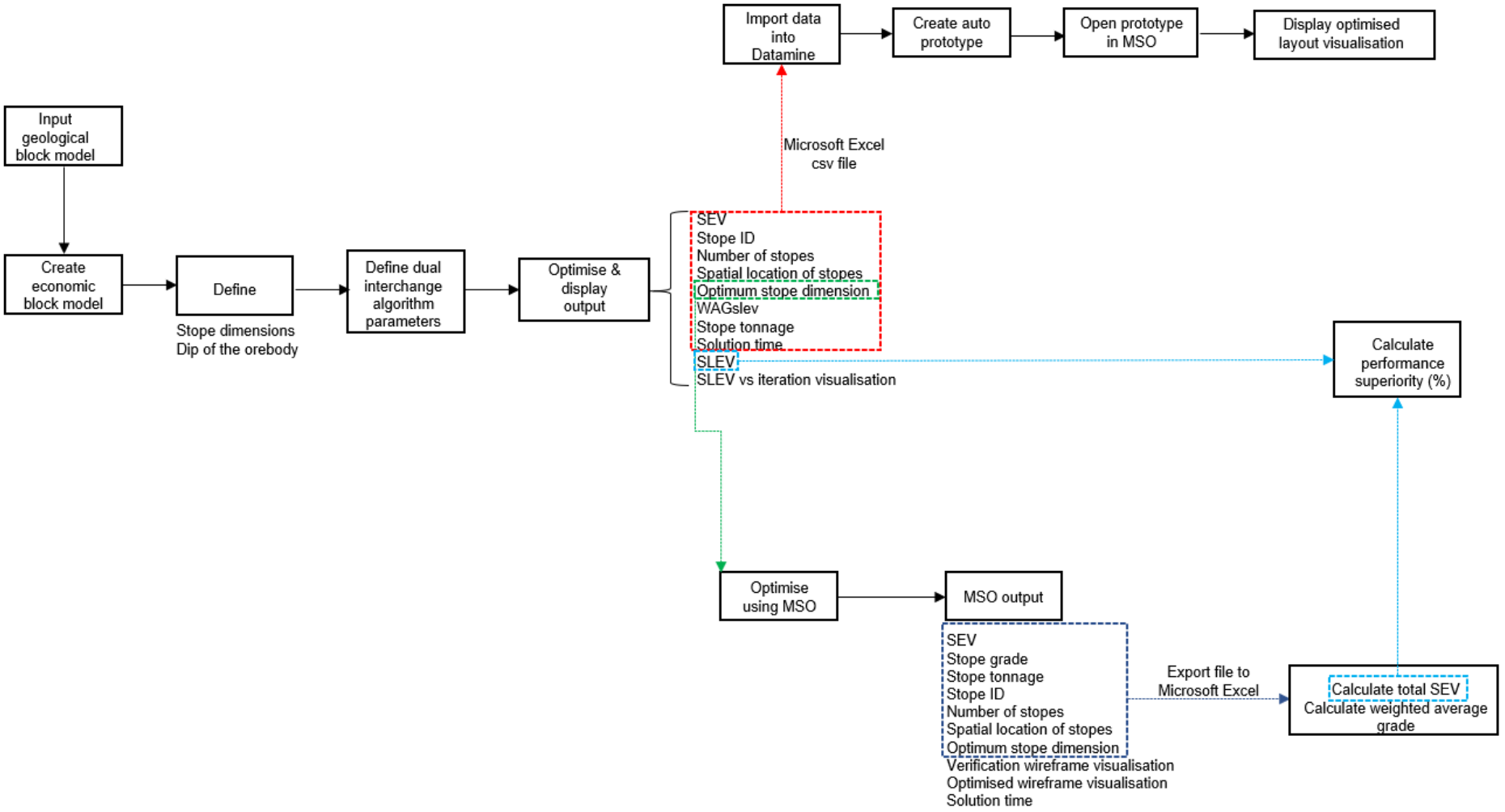


Figure 5.1 Validation process for the PSO based algorithm

5.2 Synthetic Platreef block model

The synthetic Platreef block model depicted in Figure 5.2 contains 50, 000 blocks with x, y and z-axes extents being 640m, 175m, and 55m, respectively. The characteristics of the synthetic orebody model were discussed in Section 3.5. The synthetic Platreef block model was used to demonstrate and validate the performance of the dual interchange algorithm and its qualities are shown in Table 5.1.

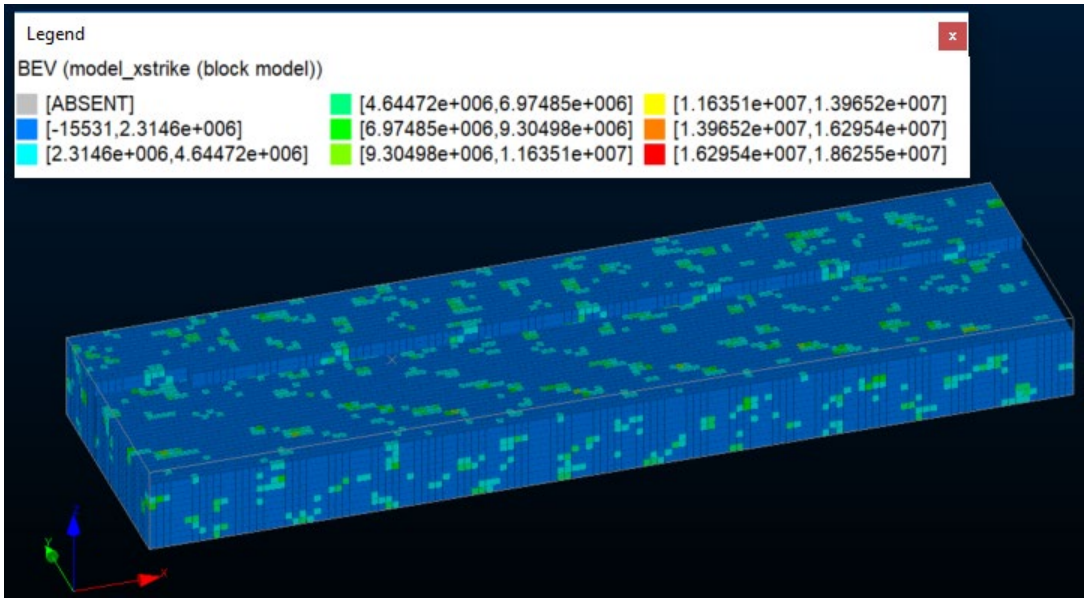


Figure 5.2 Synthetic Platreef block model with the legend showing BEV in USD

Table 5.1 Synthetic Platreef block model specifications

Parameter	Details
Spatial extent	X: -7642592.5 to -7641952.5
	Y: 742722.5 to 742897.5
	Z: -1097.5 to -1042.5
Number of blocks	50,000
Block size (x, y, z)	Fixed size: 5m x 5m x 5m
Density	3.1 t/m^3
4E PGE grade (g/t)	Variation: 0.01 to 21.9
	Average: 1.28
Blocks with positive BEV	43,060
Blocks with negative BEV	6,940
BEV (US$)	Lowest: -15,531.000
	Highest: 18,625,482.800
Total positive BEV (US$)	53,914,225,979.920
Total BEV (US$)	53,819,387,407.039

5.3 Procedure for implementing the Mineable Shape Optimizer

The Mineable Shape Optimizer is an algorithm developed to optimise underground stope design. The MSO requires input of cut-off grade or value for a block model to be able to define the optimal stope size, shape, and location of stopes for an underground stope layout. Stope optimisation in MSO starts with the creation of thin slices in the block model based on the slice interval set by the user. Each slice is a wireframe and is assessed against the block model. These slices are created in the transverse direction subject to dip and strike constraints. The model slices are used to create a seed wireframe (shape) subject to stope and pillar parameters. Thereafter, the seed wireframe is modified by stope shape annealing to create optimum stope shapes in the block model. There are numerous parameters that need to be set in the MSO optimisation process such as dip and strike control, stope and pillar dimensions, cut-off grade or value and slicing method. The seed wireframe is created above the set cut-off grade or value. The major setback of MSO is the specification of a cut-off, which disrupts the optimality of the solution generated in mine design and planning (Erdogan *et al.*, 2016).

Appendix 8.3 (Figures 8.6 to 8.11) shows an example of the parameters considered for the optimisation process using MSO. Once the run has completed, the results can be viewed under the ‘Review’ tab in the MSO. The results window displays outputs such as a detailed report, optimised wireframe, and other optional fields such as verification wireframe.

5.4 Synthetic Platreef block model optimisation

A preferred stope dimension is one that has low dilution to maximise the ore recovered. The main parameters considered during stope dimension design are the stope height and stope strike length. According to Henning (2007), stope heights of 20m, 30m and 40m with a strike length of less than or equal to 15m exhibited low dilution, whereas a sharp increase in dilution was observed when the stope length is between 15m and 20m, with a gradual increase in dilution being observed when the stope length is beyond 20m. Erdogan and Yavuz (2017) used a stope height of 25m for the sublevels citing limit of drilling accuracy of longhole drill rigs as the constraint. Furthermore, Lawrence (n.d.) stated that the stope height usually does not exceed 25m because

drill hole deviations start to occur beyond this length. The stope height is constrained by the capabilities of the drilling equipment and drilling accuracy because drilling inaccuracies increase as the stope height increases. Therefore, a strike length and height of 20m and 25m, respectively were selected for minimisation of dilution. According to Lawrence (n.d.), stope widths of less than 6m make drilling difficult for a longhole mining method. Therefore, the stope width variations in this study were limited to a minimum and maximum of 10m and 50m, respectively.

Table 5.2 presents the comparison of constraints considered by the MSO and the dual interchange algorithm. The main differences between these two algorithms are that:

- The MSO requires that minimum and maximum stope width be set, and it will be used in the optimisation process where the output may contain various stope widths. The dual interchange algorithm requires the setting of minimum and maximum stope width, and it runs through the block model and selects an appropriate stope width. The DIA then uses this appropriate stope width during the optimisation process. The use of fixed a stope width ensures that geotechnical considerations are not violated for extraction purposes and the scheduling process is simplified.
- The MSO requires that pillar dimensions be set while the DIA does not require pillars to be defined when backfill is used, thus allowing for maximum extraction.
- The MSO requires that the dip of the orebody be defined, and the dip of the stope may vary by a certain amount specified by the user. This dip variation may impact the layout orientation, thus negatively affecting the practicality of extraction. The DIA applies a fixed dip of the orebody, thus the stopes generated are aligned with the true dip of the orebody ensuring consistency in the layout generated.

Table 5.2 Main constraints of the MSO and dual interchange algorithm

MSO	Dual interchange algorithm
Set optimisation field: grade or value	Optimisation field: value
True Dip with varying orientation	True Dip
Cut-off grade/value	Cut-off set to zero measured as value
Set fixed/variable stope dimension	Set fixed/variable stope dimension
Set pillars between stopes along optimisation field	No pillars considered; skin-to-skin stopes desired
Stope orientations vary from true dip as per user input	Stope orientation is aligned to the true dip
Set accepted stope dilution (user dependent)	Dilution incorporated in the calculation of BEV (mine design dependent)
Unique stopes	Unique stopes
Non-overlapping stopes	Non-overlapping stopes
No specified number of iterations	User specified number of iterations
Algorithm parameters cannot be varied	Algorithm parameters can be varied
User can adjust the generated solution for further refinement	User cannot adjust the generated optimal solution
Local or global optima cannot be tested	Local or global optima can be tested by analysing the output *Gbest* and iteration graph

5.5 Scenarios to demonstrate the performance of the DIA

Four scenarios were used to demonstrate the performance of the DIA in different mining situations. The stope dimension constraints considered for the application of MSO and the dual interchange algorithm are shown in Table 5.3. A cut-off value of zero was selected to ensure that all stopes with positive stope economic value could be included in the final stope layout and not to disrupt the optimality of the solution generated. The other constraints considered in the optimisation process are as discussed in Section 4.3.1 to 4.3.4. The pillars were not considered for this optimisation validation because of the stope layout generated with pillars did not allow for practical extraction without the user adjustments to the layout. Once the user adjusts the layout for practical implementation, that may invalidate the optimality of the layout. Furthermore, the mining method does not need pillars since the mined-out stopes will be completely backfilled before adjacent stopes are extracted.

Table 5.3 Input parameters considered for the optimisation problem

Scenario	Length (m) on x-axis	Width (m) on y-axis	Height (m) on z-axis
A	20	40	25
B		10 – 40	
C		50	
D		10 – 50	

For each of the possible scenarios with stope dimensions listed in Table 5.3, runs were generated in the MSO to determine the consistency of the best layout economic value and it was found to be consistent for every run. Thereafter, the different possible stope layout solutions for each scenario were compared with the solutions generated from the DIA. The solution values for each scenario were plotted against the iterations to identify the convergence of the solution values.

The synthetic Platreef block model contains 6940 waste blocks (i.e., blocks with a grade value of zero). Using Equation 3.1 to calculate the BEV, if a number of waste blocks are included in the optimum stope layout, the economic contribution is US $ -15,531 per block given the economic constraints used in this study. The inclusion of any block with a negative BEV in a stope reduces the economic value of that stope. If a block is created outside of the spatial dimensions of the block model during the optimisation process it is classified as dummy block. A dummy block is attributed the same characteristics as a waste block. Figure 5.3 illustrates how dummy blocks get included during the stope layout optimisation process.

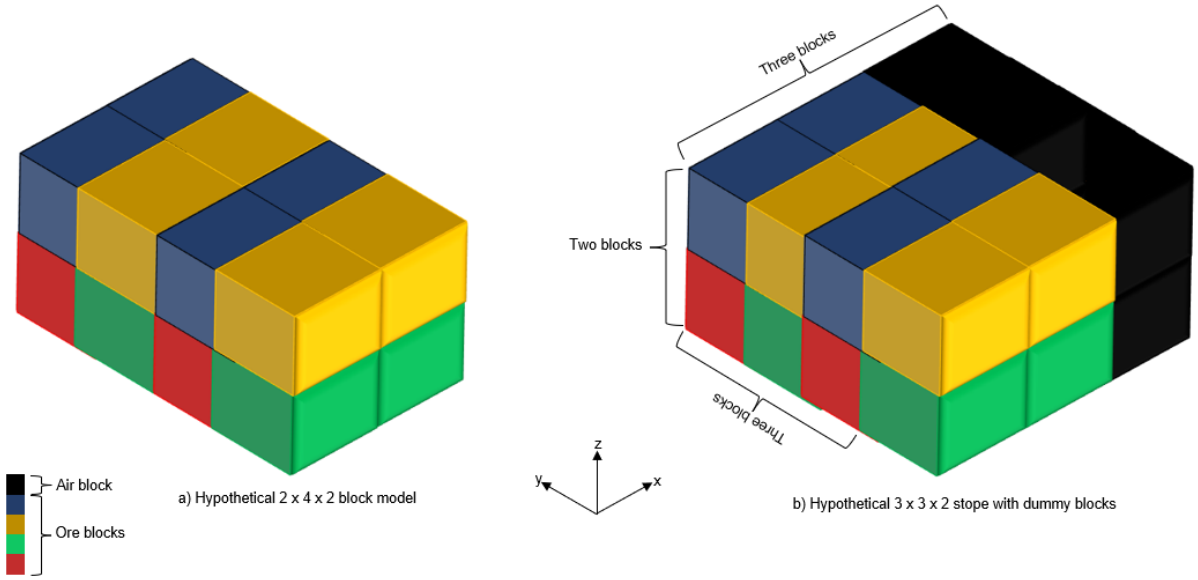


Figure 5.3 Hypothetical block model with dummy blocks

In an effort not to violate the stope dimension constraint, the MSO creates dummy blocks outside of the block model where additional blocks are needed to complete a stope. In Figure 5.3, a stope that has 3 x 3 x 2 blocks along the x, y, and z axes is required. Therefore, an extra set of blocks is needed along the x-axis to accommodate a stope length of 3 blocks, consequently creating dummy blocks. The dummy blocks represented waste blocks outside the economic block model. Thus, they attract costs associated with activities from extraction to refining if they are included in the optimum stope layout. However, the MSO does not account for costs associated with activities from extraction to refining for the dummy blocks. Thus, the MSO violates the optimality principles when it creates stopes outside of the block model because the SEV attributed to a stope will be less if the cost associated with mining dummy blocks is included.

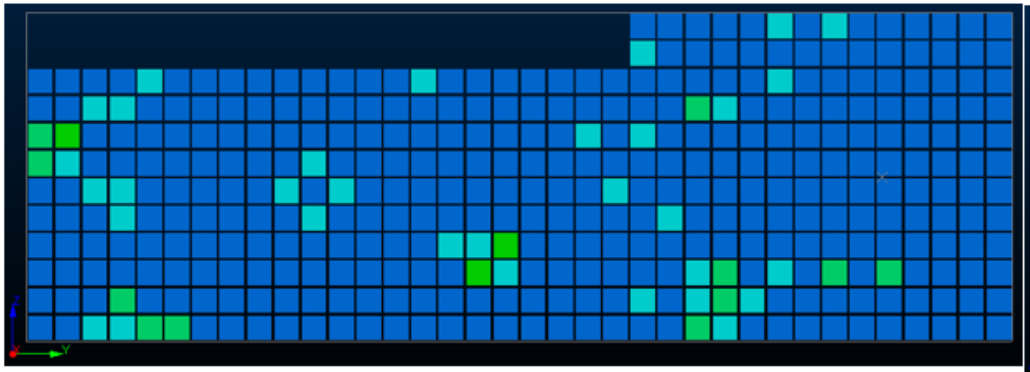
The following subsections present the outcomes of the optimisation process for each case as generated by the MSO and DIA algorithms.

5.5.1 Scenario A optimisation results

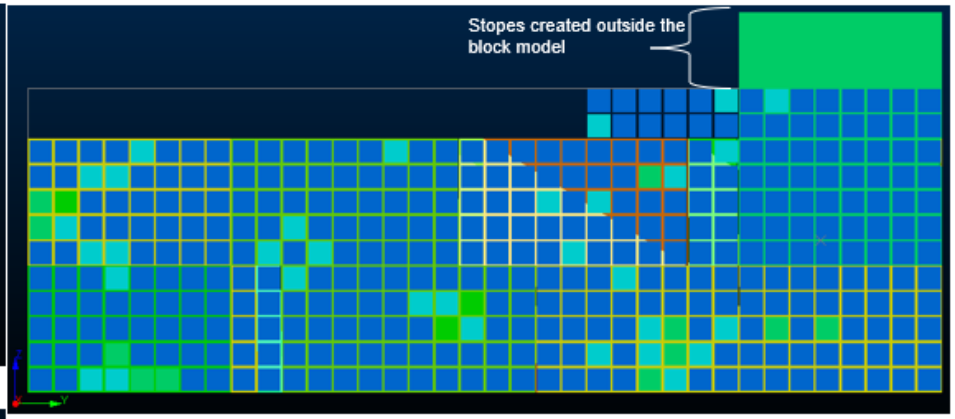
Figure 5.4 shows the stope layout solution generated using MSO for Scenario A where the stope has a strike length of 20m, height of 25m and stope width of 40m. Figure 5.4a represents the section view of the block model along the optimisation field. The MSO creates stopes in waste ground that results in the reduction of the maximum

value to be realised from mining the ore deposit (see Figure 5.4b). However, these stopes in the waste ground (where there are no blocks in the economic model) do not violate the cut-off value constraints. The MSO aggregates non-existing blocks to complete the stope shape, these blocks are assigned a value of zero. However, these blocks need to be extracted when the stope is mined. Thus, there are costs associated with activities from extraction to refining that are not accounted for. These costs may deem a stope uneconomic, consequently reducing the stope layout economic value.

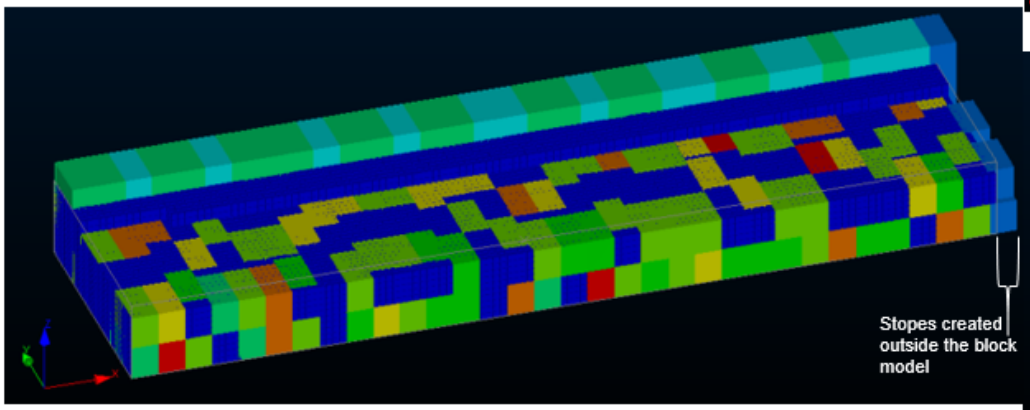
Figure 5.4c presents the 3D block model and the optimised 3D solution from the MSO. The block model has numerous blocks that have negative BEVs in the middle area, hence, not many stopes were created in the middle part of the block model. Nonetheless, the generated stopes pose a challenge in terms of planning the production tunnels due to the pronounced overlap positions. Figure 5.4d depicts the 3D stope layout visualisation and shows that the stopes beyond the block model will require a dedicated drilling tunnel which further increases the cost of extracting these stopes. The MSO generated a solution that has a stope layout economic value (SLEV) of US$ 35.87 billion from 217 stopes.



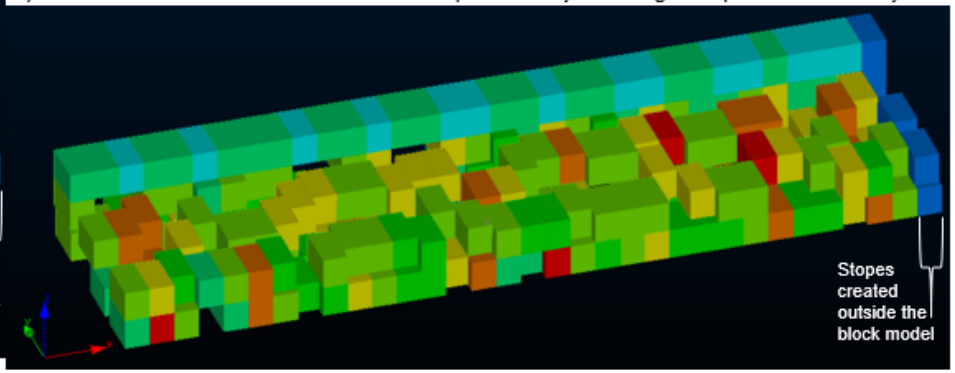
a) Section view of the block model along the optimisation field: y-axis



b) Section view of the block model and the optimised layout along the optimisation field: y-axis



c) 3D visualisation of the block model and the optimised layout



d) 3D visualisation of the optimised layout

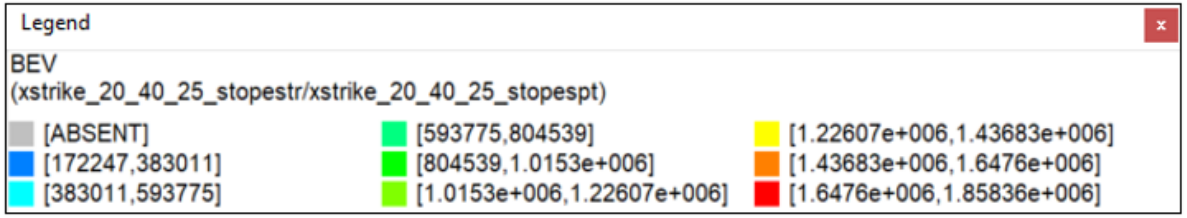


Figure 5.4 Scenario A: visualisations of the optimum stope layout solution using the MSO with the legend showing SEV in USD

The convergence of the solution of the DIA is shown in Figure 5.5 where the solution was generated after 200 iterations. Therefore, it can be concluded that the solution is not local optima, and the search space was well exploited. The visualisation of the optimum stope layout solution to Scenario A using the DIA is presented in Figure 5.6. The DIA generated a solution that has a SLEV of US$ 35.97 billion from 213 stopes. All the stopes were created within the block model perimeter.

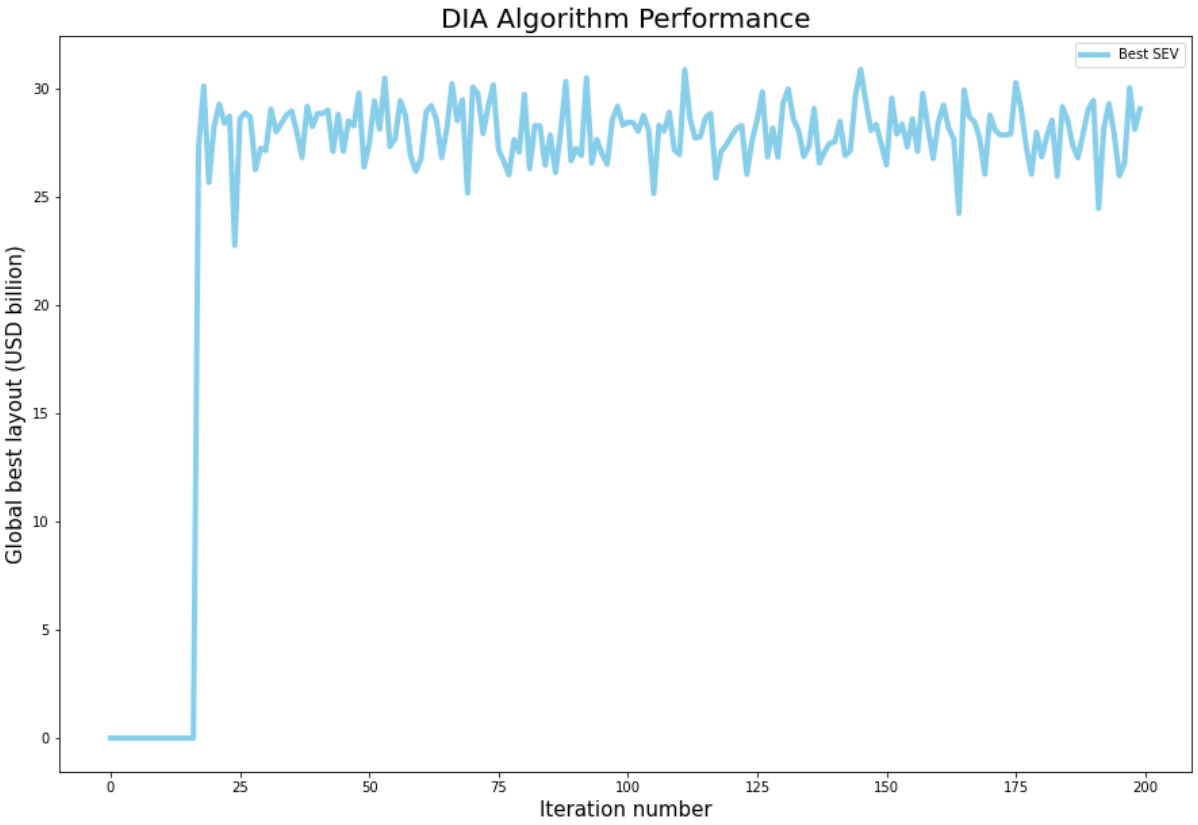


Figure 5.5 Scenario A: optimum solution convergence using the DIA

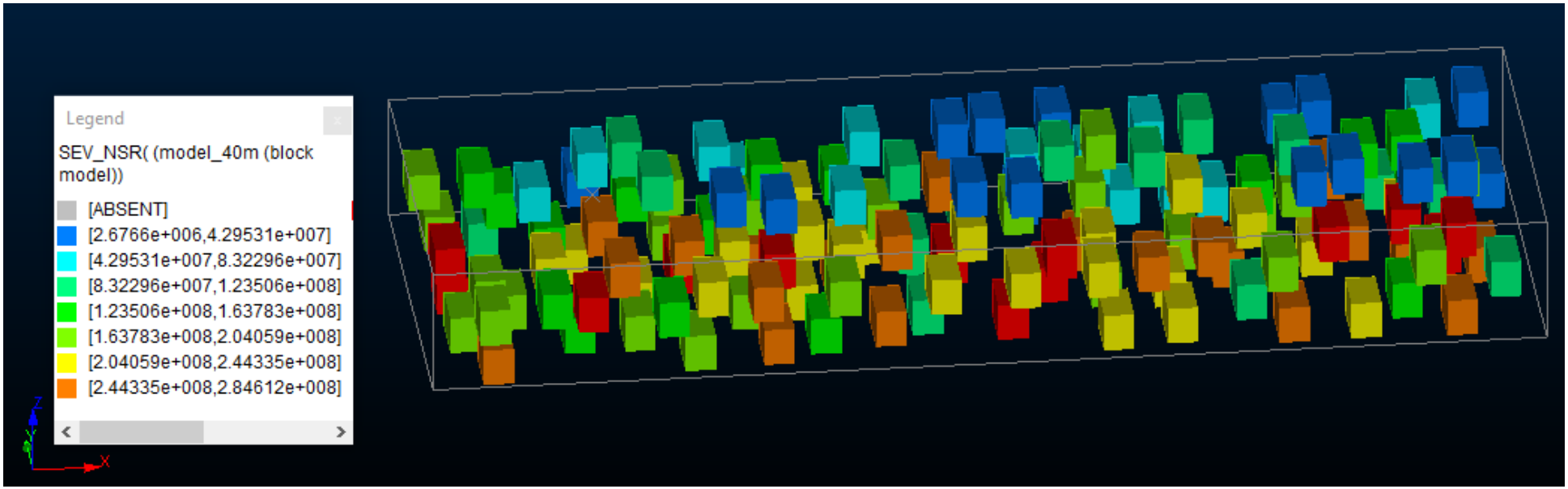


Figure 5.6 Scenario A: visualisations of the optimum stope layout solution using the DIA with the legend showing SEV in USD

Table 5.4 presents a comparison of the MSO and DIA where the DIA generates an optimum stope layout that is 0.3% more profitable than the MSO. The MSO generates a solution at much shorter solution time than the DIA; this is alluded to the fact that MSO only produces a single solution while the DIA generates several solutions before determining the optimum solution.

Table 5.4 Scenario A: comparison of the optimum solutions generated by the MSO and DIA

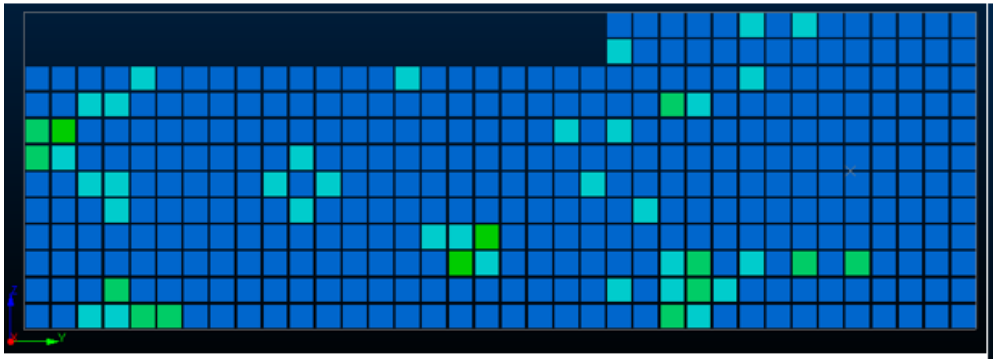
Parameter	**MSO**	**DIA**
Maximum stope layout economic value (US $)	35,866,231,792.854	35,972,381,189.708
Number of stopes	217	213
Solution time (hh:mm:ss)	00:00:47	05:39:34

5.5.2 Scenario B optimisation results

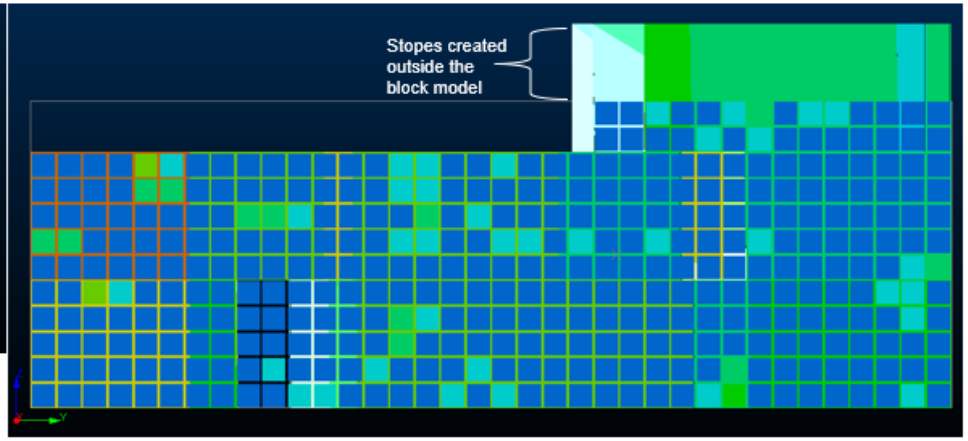
Figure 5.7 shows the stope layout solution generated using MSO for Scenario B where the stope has a strike length of 20m, height of 25m and varying stope width from 10 to 40m. The optimised layout has an average, minimum and maximum stope width of 37.9m, 14.1m and 40.4m, respectively. Figure 5.7a represents the section view of the block model along the optimisation field. The MSO creates stopes in waste ground that results in a reduction of the maximum value to be realised from mining the mineral deposit (see Figure 5.7b). However, these stopes do not violate the cut-off value constraints as they all have a positive stope economic value. The blocks outside the block model are assigned an economic value of zero. However, the actual economic value of these blocks is US$ -15,531 per block when costs associated with activities from extraction to refining are accounted for. The inclusion of these costs may deem the said stopes uneconomic and/or reduce the generated SLEV.

Figure 5.7c presents the 3D block model and the optimised 3D solution from the MSO. The block model has numerous blocks that have negative BEV values in the middle area, hence, not many stopes were created in the middle part of the block model. However, the varying stope width increased the number of stopes in the middle when compared to a fixed stope width of 40m. Nonetheless, the generated stopes present a challenge in terms of planning the production tunnels due to the pronounced overlap

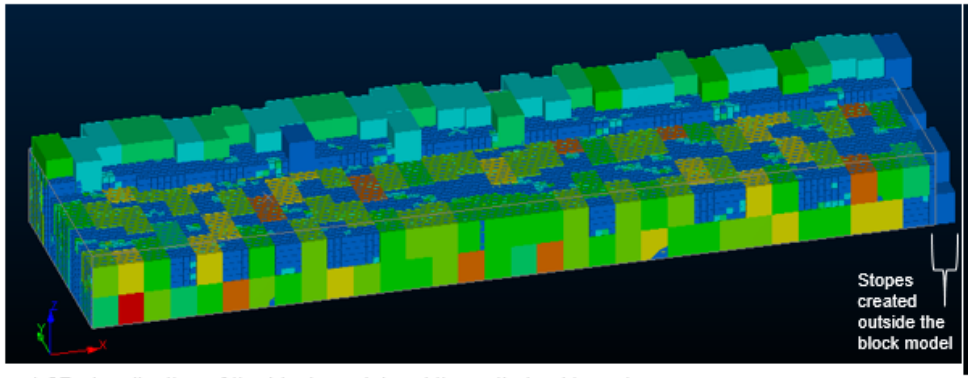
positions. Figure 5.7d depicts the 3D stope layout visualisation and shows that the stopes beyond the block model will require a dedicated drilling tunnel which will further increase the cost of extracting these stopes. The MSO generated a solution that has a SLEV of US$ 35.73 billion from 228 stopes.



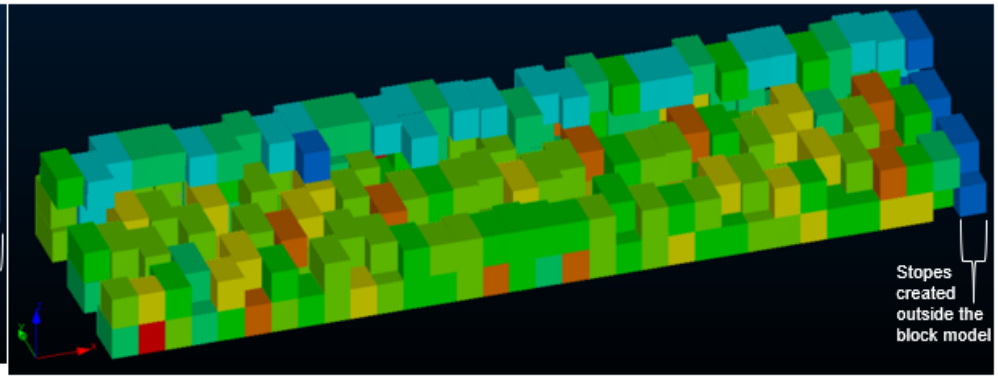
a) Section view of the block model along the optimisation field: y-axis



b) Section view of the block model and the optimised layout along the optimisation field: y-axis



c) 3D visualisation of the block model and the optimised layout



d) 3D visualisation of the optimised layout

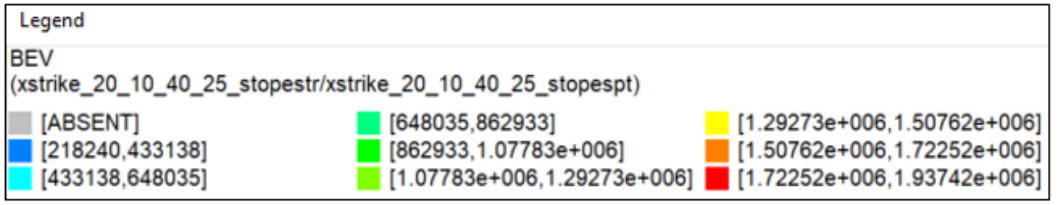


Figure 5.7 Scenario B: visualisations of the optimum stope layout solution using the MSO with the legend showing SEV in USD

The convergence of the solution of the DIA is shown in Figure 5.8. The solution was generated after 200 iterations. The visualisation of the optimum stope layout solution for Scenario B using the DIA is presented in Figure 5.9. The DIA generated a solution that has a SLEV of US$ 37.00 billion from 647 stopes. The DIA selected a stope width of 10m, hence the number of stopes in the optimised layout is high. All the stopes were created within the block model perimeter.

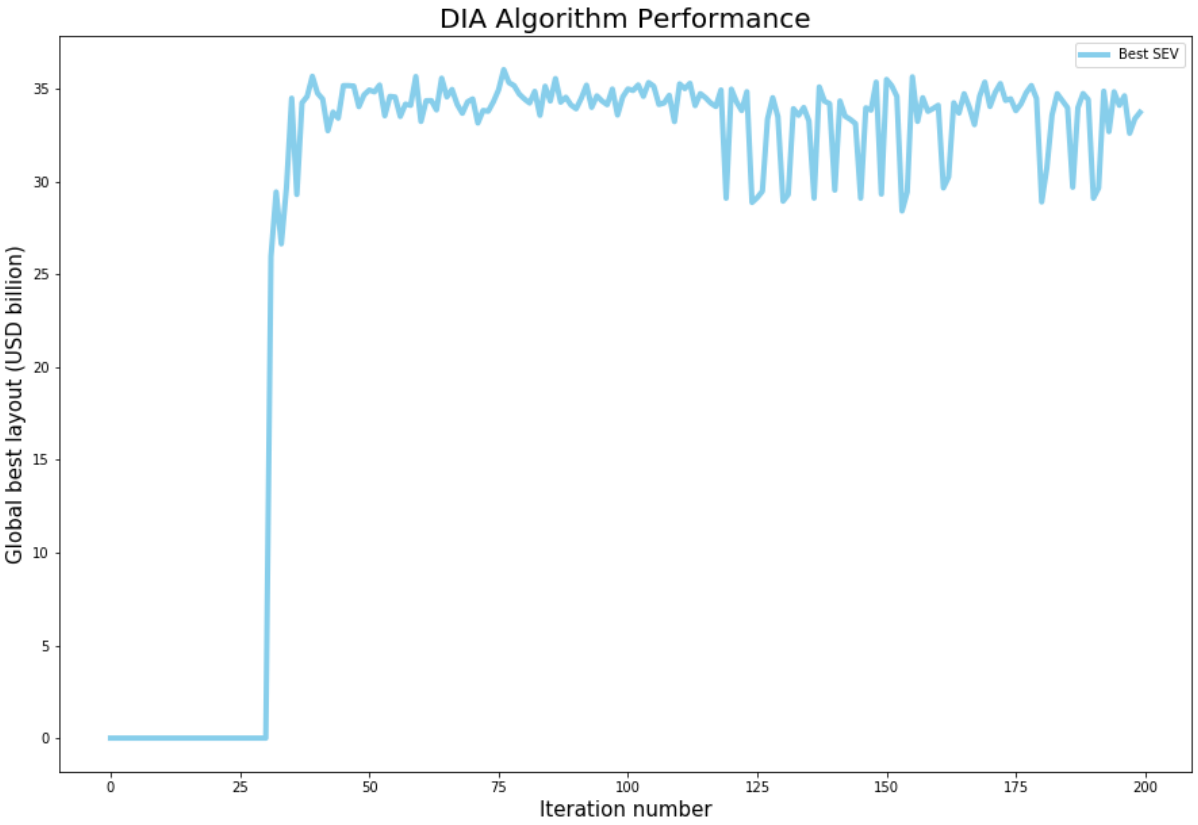


Figure 5.8 Scenario B: optimum solution convergence using the DIA

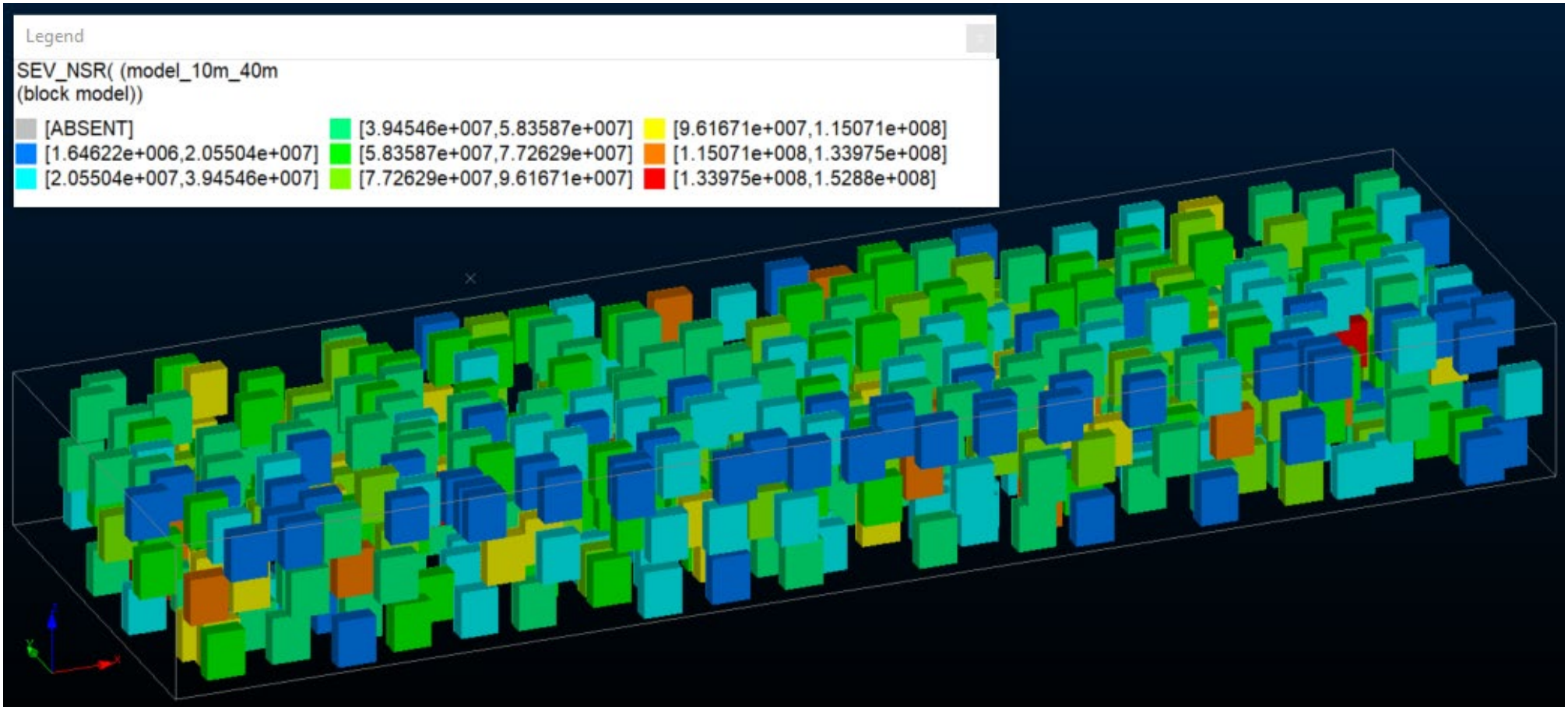


Figure 5.9 Scenario B: visualisations of the optimum stope layout solution using the DIA with the legend showing SEV in USD

Table 5.5 presents a comparison of the MSO and DIA where the DIA generated an optimum stope layout that is 3.4% more profitable than the MSO. Furthermore, the MSO generated a solution at much shorter solution time than the DIA; this is alluded to the fact that the MSO only produces a single solution while the DIA generates several solutions to determine the optimum solution.

Table 5.5 Scenario B: comparison of the optimum solutions generated by the MSO and DIA

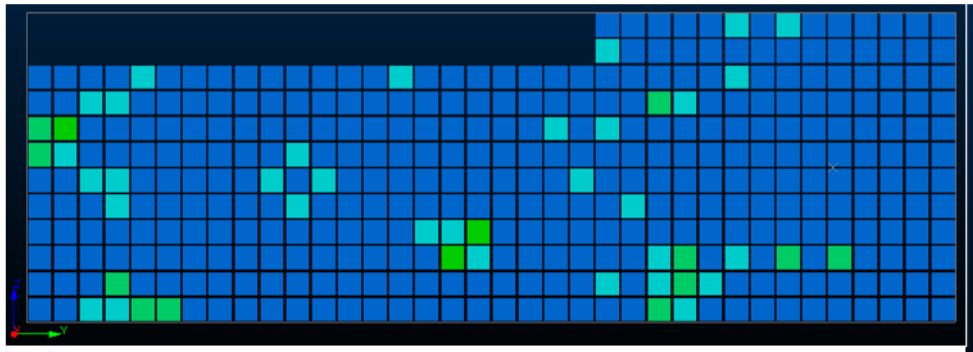
Parameter	**MSO**	**DIA**
Maximum stope layout economic value (US $)	35,730,081,740.688	37,004,613,547.534
Number of stopes	229	647
Solution time (hh:mm:ss)	00:02:58	17:15:24

5.5.3 Scenario C optimisation results

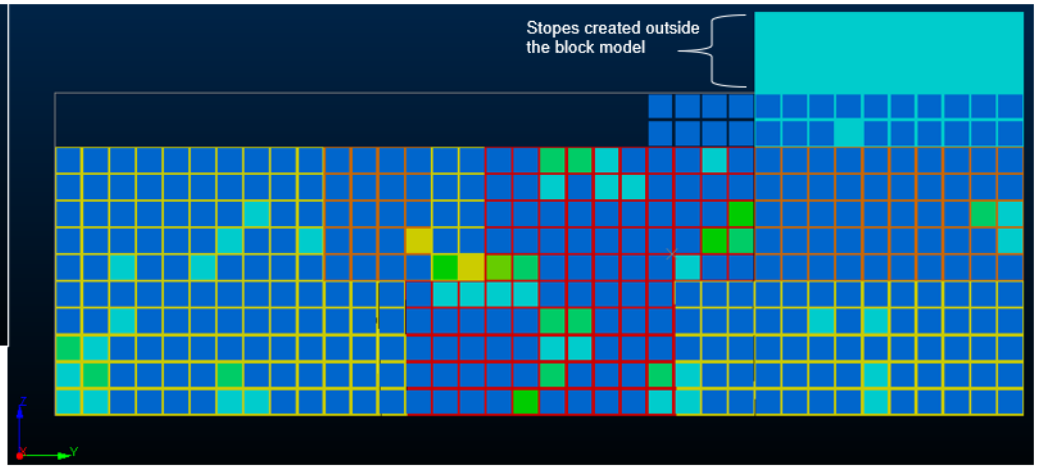
Figure 5.10 shows the stope layout solution generated using MSO for Scenario C where the stope has a strike length, height, and width of 20m, 25m and 50m, respectively. Figure 5.10a represents the section view of the block model along the optimisation field. The MSO creates stopes in waste ground that results in the reduction of the maximum value to be realised from mining the mineral deposit (see Figure 5.10b). However, these stopes do not violate the cut-off value constraints as they all have a positive stope economic value. The blocks outside the block model are assigned an economic value of zero. However, the actual economic value of these blocks is US$ -15,531 per block when costs associated with activities from extraction to refining are accounted for. The inclusion of these costs may deem the said stopes uneconomic and reduce the generated SLEV.

Figure 5.10c presents the 3D block model and the optimised 3D solution from the MSO. The block model has numerous blocks that have negative BEVs in the middle area, hence, not many stopes were created in the middle of the block model. However, the stope width of 50m created several stopes in the middle part of the block model because the width is fixed, thus some blocks with negative BEVs had to be included. Nonetheless, the generated stopes present a challenge in terms of planning the production tunnels due to the pronounced overlap positions. From Figure 5.10d, the

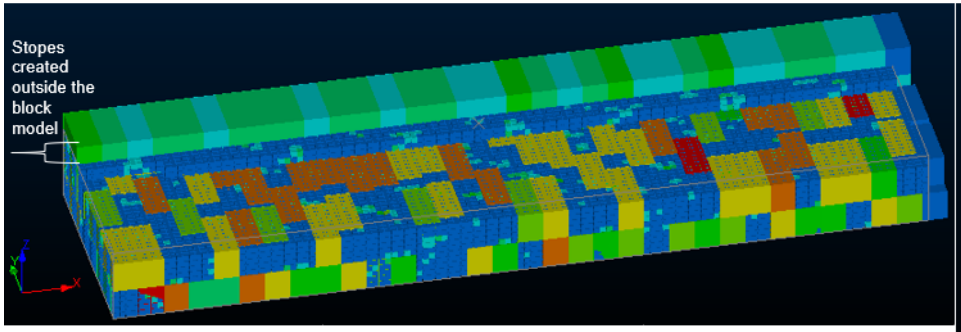
stope overlap in terms of position is pronounced which means some drifts will have to be driven longer into the orebody to reach the selected stopes. Furthermore, the stopes created outside of the block model will require a dedicated drilling tunnel during mining, which increase the extraction cost. The MSO generated a solution that has a SLEV of US$ 38.81 billion from 188 stopes.



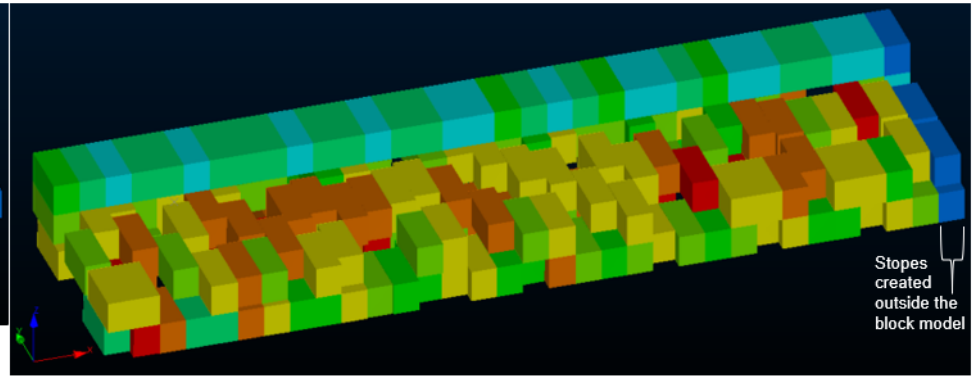
a) Section view of the block model along the optimisation field: y-axis



b) Section view of the block model and the optimised layout along the optimisation field: y-axis



c) 3D visualisation of the block model and the optimised layout



d) 3D visualisation of the optimised layout

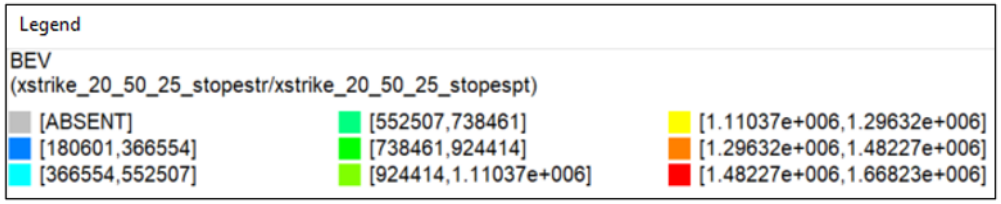


Figure 5.10 Scenario C: visualisations of the optimum stope layout solution using the MSO with the legend showing SEV in USD

The convergence of the solution of the DIA is shown in Figure 5.11. The solution was generated after 200 iterations. The visualisation of the optimum stope layout solution to Scenario C using the DIA is presented in Figure 5.12. The DIA generated a solution that has a SLEV of US$ 35.04 billion from 159 stopes.

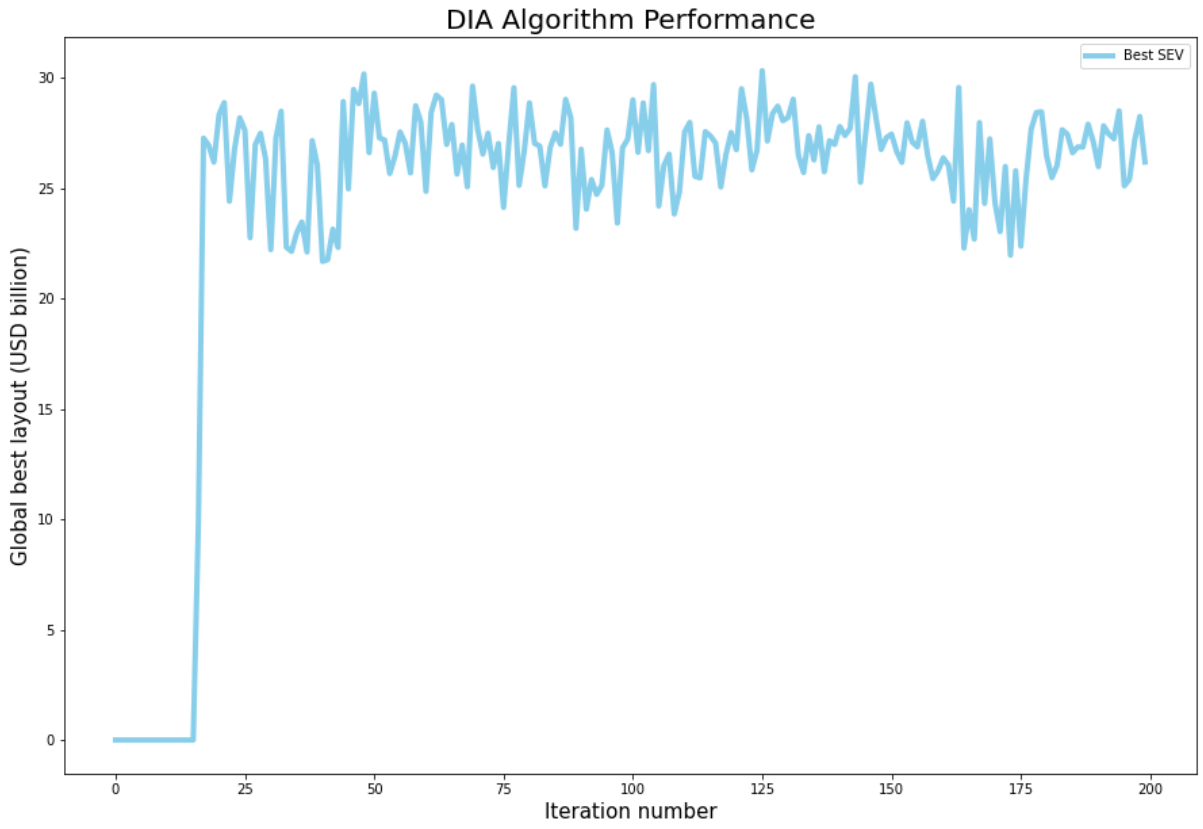


Figure 5.11 Scenario C: optimum solution convergence using the DIA

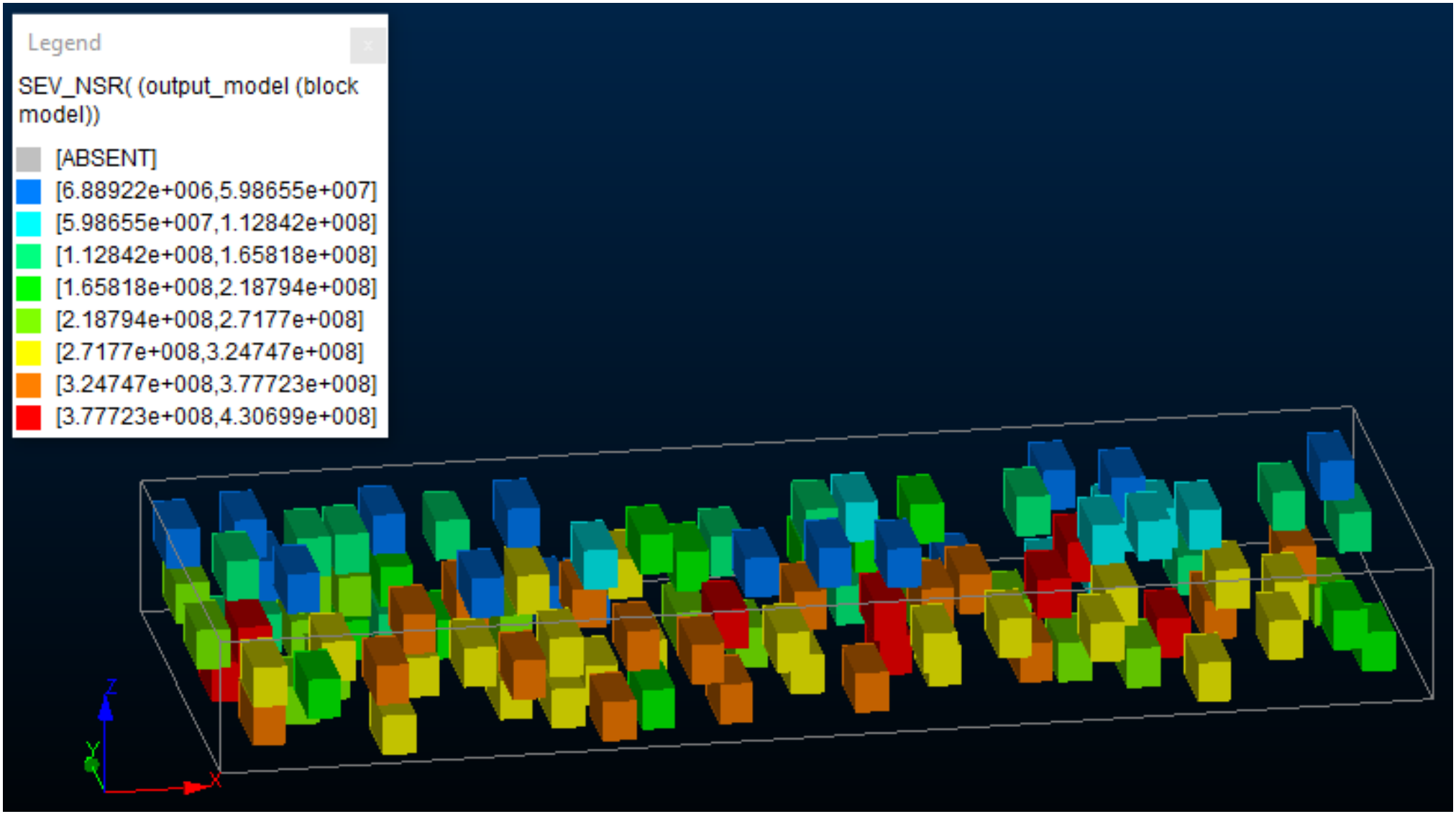


Figure 5.12 Scenario C: visualisations of the optimum stope layout solution using the DIA with the legend showing SEV in USD

Table 5.6 presents a comparison of the MSO and DIA where the DIA generated an optimum stope layout that is 9.7% less profitable than the MSO. Furthermore, the MSO generated a solution at much shorter solution time than the DIA; this is alluded to the fact that MSO only produced a single solution while the DIA generated several solutions used to determine the optimum solution.

Since there are waste blocks within the economic block model, the stope width of 50m made the DIA to generate numerous possible stope combinations to reduce the effect of dilution from the internal waste. In addition, the GA optimisation process will take longer because the termination criterion is based on lack of improvement in the solution not a set number of iterations. The poor the new generation in terms of fitness, the more generations are created to improve the quality of the best fitness value.

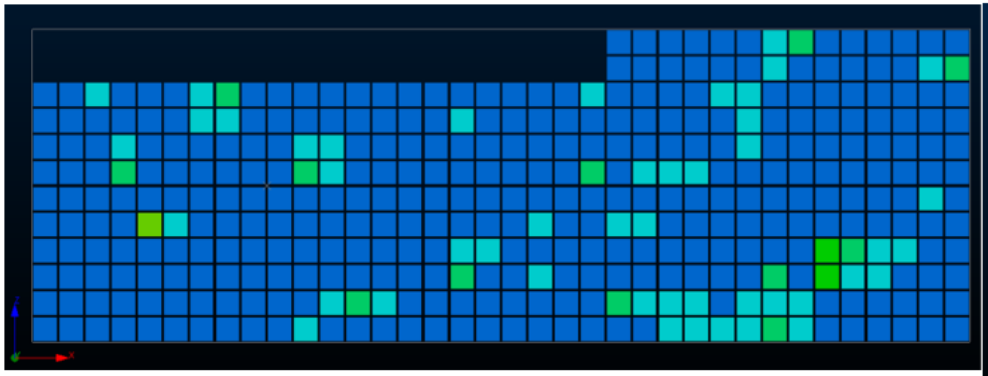
Table 5.6 Scenario C: comparison of the optimum solutions generated by the MSO and DIA

Parameter	**MSO**	**DIA**
Maximum stope layout economic value (US $)	38,805,614,807.051	35,042,470,315.314
Number of stopes	188	159
Solution time (hh:mm:ss)	00:00:39	09:25:10

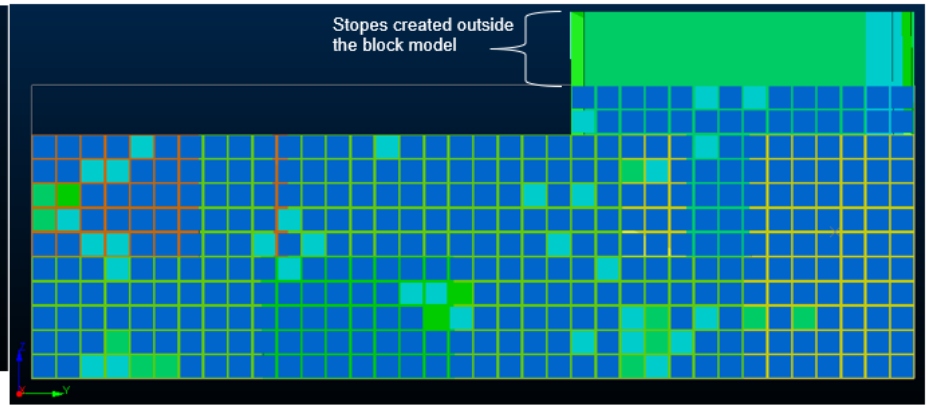
5.5.4 Scenario D optimisation results

Figure 5.13 shows the stope layout solution generated using MSO for Scenario D where the stope has a strike length of 20m, height of 25m and varying stope width from 10 to 50m. The optimised layout has an average, minimum and maximum stope width of 46.2m, 10.2m and 50.5m, respectively. Figure 5.13a represents the section view of the block model along the optimisation field. The MSO creates stopes in waste ground that results in the reduction of the maximum value to be realised from mining the mineral deposit (see Figure 5.13b). However, these stopes do not violate the cut-off value constraints as they all have positive stope economic values. The blocks outside the block model are assigned an economic value of zero. However, the actual economic value of these blocks is US$ -15,531 per block when costs associated with activities from extraction to refining are accounted for. The inclusion of these costs may deem these stopes uneconomic resulting in a reduction of the generated SLEV.

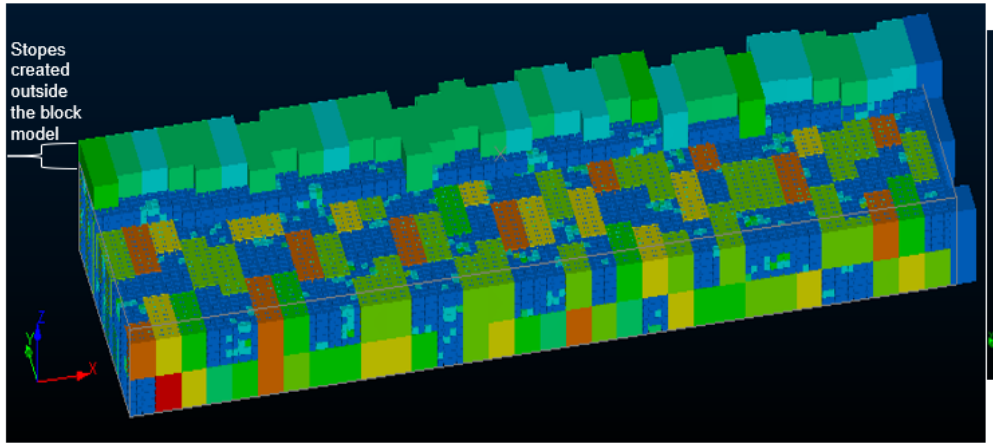
Figure 5.13c presents the 3D block model and the optimised 3D solution from the MSO. The block model has numerous blocks that have negative BEVs in the middle area, hence, not many stopes were created in the middle part of the block model. However, the varying stope width increased the number of stopes in the middle. Nonetheless, the generated stopes present a challenge in terms of planning the production tunnels due to the pronounced overlap positions. Figure 5.13d depicts the 3D stope layout visualisation and shows that the stopes beyond the block model will require a dedicated drilling tunnel which will further increase the cost of extracting these stopes. The MSO generated a solution that has a SLEV of US$ 34.04 billion from 181 stopes.



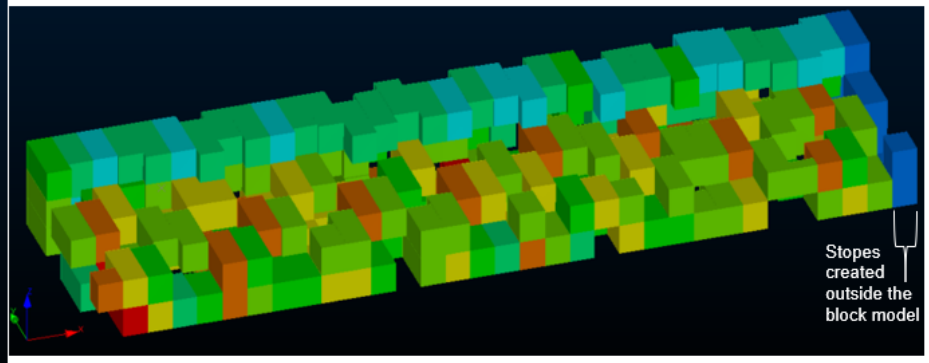
a) Section view of the block model along the optimisation field: y-axis



b) Section view of the block model and the optimised layout along the optimisation field: y-axis



c) 3D visualisation of the block model and the optimised layout



d) 3D visualisation of the optimised layout

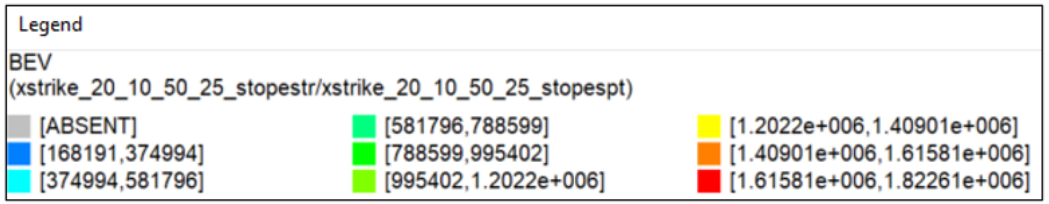


Figure 5.13 Scenario D: visualisations of the optimum stope layout solution using the MSO with the legend showing SEV in USD

The convergence of the solution of the DIA is shown in Figure 5.14 where the solution was generated after 200 iterations The visualisation of the optimum stope layout solution to Scenario D using the DIA is presented in Figure 5.15. The DIA generated a solution that has a SLEV of US$ 37,11 billion from 161 stopes. The DIA selected a stope width of 10m, hence the number of stopes in the optimised layout is high. All the stopes in the layout were created within the block model perimeter.

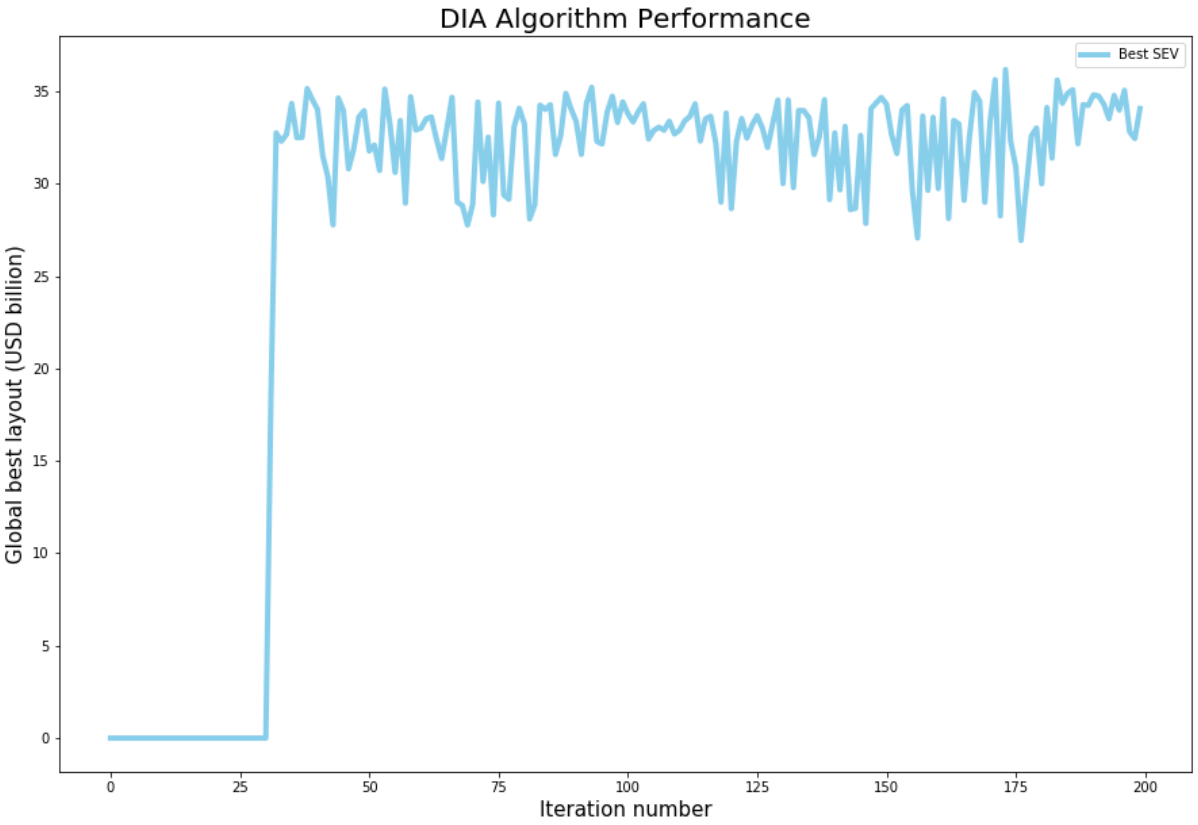


Figure 5.14 Scenario D: optimum solution convergence using the DIA

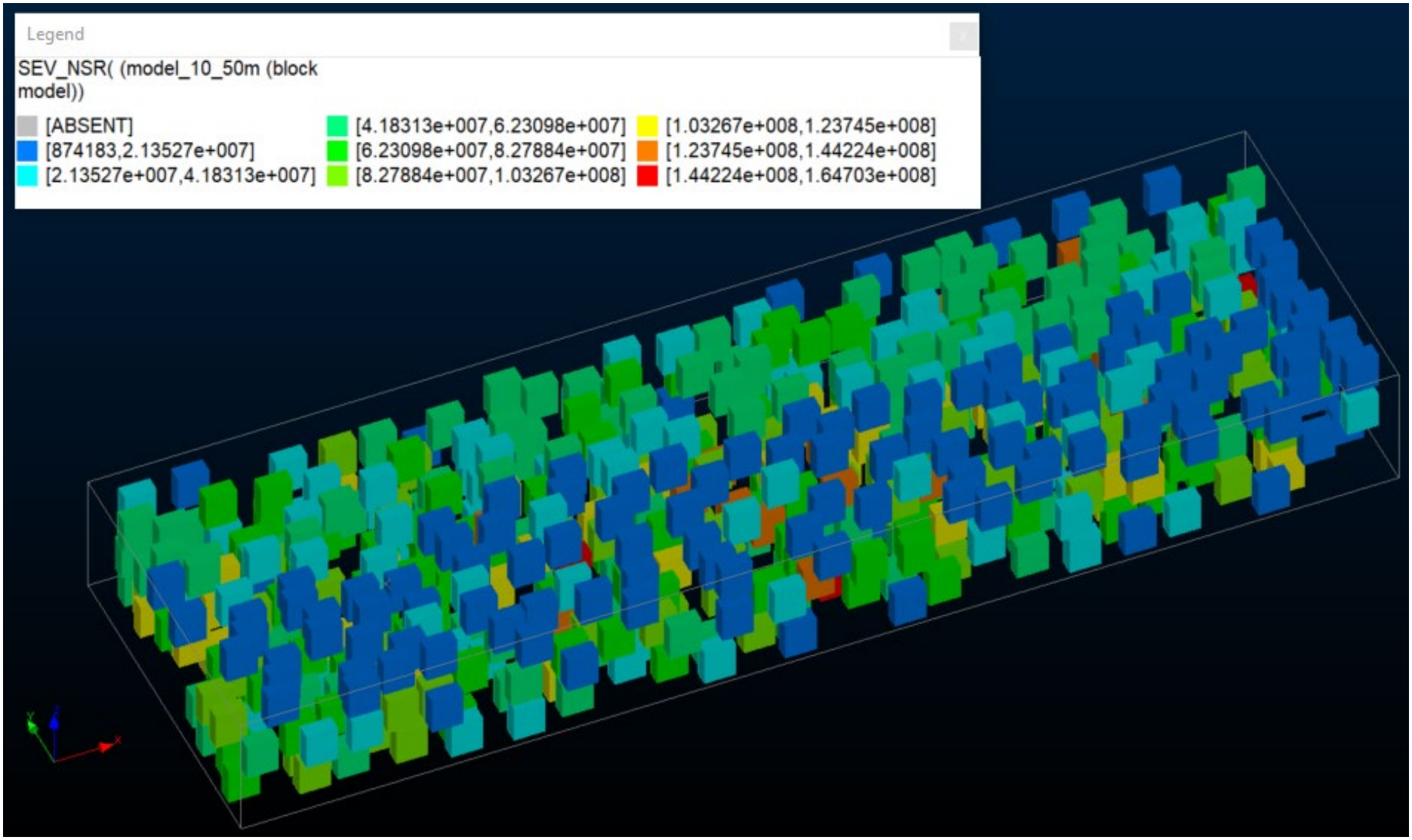


Figure 5.15 Scenario D: visualisations of the optimum stope layout solution using the DIA with the legend showing SEV in USD

Table 5.7 presents the comparison of the MSO and DIA where the DIA generated an optimum stope layout that is 8.3% more profitable than the MSO. Furthermore, the MSO generates a solution at much shorter solution time than the DIA; this is alluded to the fact that MSO only produced a single solution while the DIA generated several solutions used to determine the optimum solution.

Table 5.7 Scenario D: comparison of the optimum solutions generated by the MSO and DIA

Parameter	**MSO**	**DIA**
Maximum stope layout economic value (US $)	34,044,114,429.081	37,108,222,577.360
Number of stopes	181	677
Solution time (hh:mm:ss)	00:02:19	11:18:06

5.6 Summary of Chapter 5

The dual interchange algorithm was validated by comparing its results and performance with that of MSO. The validation process was conducted to ascertain if there has been any value added to the existing knowledge for mine planning and optimisation of stope layout optimisation problem. Four scenarios (i.e., Scenarios A, B, C and D) were used to test the robustness of the dual interchange algorithm. The results show that the dual interchange algorithm generated superior results compared to the MSO as summarised in Figure 5.16. The stope layout economic values generated using the dual interchange algorithm were more profitable than MSO results for Scenarios A, B and D, while the MSO produced superior results for Scenario C.

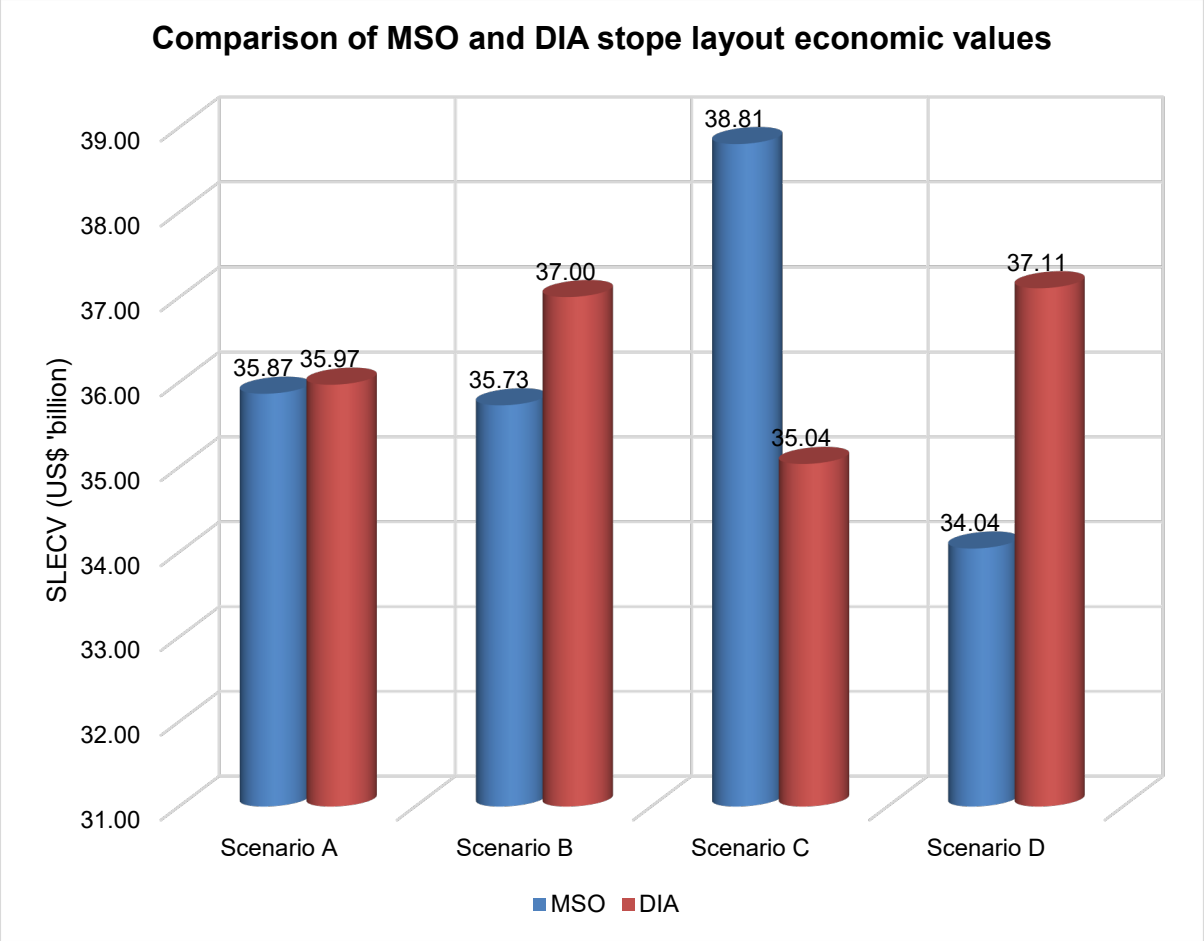


Figure 5.16 Comparison of the performance of the MSO and DIA

In Scenarios B and D, the stope width is variable, and the DIA tends to produce superior SLEV. The use of variable stope width is ideal in mine planning because the orebody width is not regular, so it is ideal to have variable stope widths. The results confirm that the architecture of the DIA is best suited for variable stope width as discussed in Section 3.2, and the solution times in Tables 5.4 to 5.7 in the order of several hours corroborate the argument presented earlier in Section 2.2 of this thesis.

The stope width for Scenarios A and C is fixed and the DIA tends to just out-perform the MSO as in Scenario A or underperform the MSO as in Scenario C. Fixed stope width is not ideal because the orebody is irregular, thus if the stope width is fixed, waste is taken in and geological contacts may be violated creating a safety issue during the ore extraction process. In addition, the DIA remained confined within the block model (i.e., the DIA does not create dummy blocks outside the economic block

model in its optimisation process). Since there are waste blocks within the economic block model, the stope width of 50m resulted in the DIA generating numerous possible stope combinations to reduce the effect of dilution from the internal waste. In addition, the GA optimisation process will take longer because the termination criterion is based on lack of improvement in the solution and not the number of iterations. If a new generation is poor in terms of fitness, then more generations are created to improve the quality of the best fitness value.

Although the DIA produced inferior SLEV for Scenario C, it should be noted that the MSO included dummy blocks in the optimum stope layout which represent waste blocks in a real mining environment. Therefore, the SLEV generated by the MSO will be less than the one reported if costs associated with extraction, processing and refining the dummy blocks are accounted for.

When analysing the four different scenarios, the solution time of the MSO is much shorter compared to the solution time of the DIA. However, Figures, 5.5, 5.8, 5.11, and 5.14 indicate that the highest SLEV is achieved after about 50 maximum iterations. This observation suggests that the DIA can have shorter running times by capping the number of iterations to a maximum of 50. The next chapter discusses the findings of the study and provide recommendations for future work.

6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Overview of Chapter 6

The aim of optimisation in an underground mining environment is to maximise the value derived from the Mineral Resource through an appropriate mine design and plan. However, the circular logic in underground mine planning makes it complex to generate an optimum solution that satisfies all the constraints for generating an optimum mining layout. There are various factors to consider when planning to exploit a sub-surface orebody and these include factors such as location and sizing of infrastructure and development, stope layout, production scheduling and equipment selection. These factors are interdependent, however, to handle these factors simultaneously will result in an intractable problem. Therefore, this study considered only the stope boundary optimisation problem to generate a stope layout using a dual interchange algorithm. This chapter presents conclusions from the optimisation undertaken using the DIA. It then makes recommendations for mine planning and optimisation of underground stope boundaries, presents the limitations of the study, and recommendations for future work.

6.2 Conclusions and contribution to knowledge

This study presented an innovative technique, the DIA, to handle the complexity of stope layout optimisation problem for underground mines. The dual interchange algorithm employs the metaheuristics principles of the particle swarm optimisation approach and the genetic algorithm; thus, it is metaheuristic by nature. A user-friendly application was developed using Python programming language to implement the DIA for different mining scenarios. This application allows the user to modify the input variables for the economic block model development and for the algorithm performance. However, manual changes of the source code are not permitted and do not affect the performance of the algorithm.

The algorithm was implemented for a resource model that mimics the Platreef deposit where four different types of minerals are to be extracted, i.e., 4E PGE. Four scenarios (A to D) were defined to demonstrate the performance of the algorithm in solving the

underground stope boundary optimisation problem. The solutions generated by the DIA were more profitable by 0.3%, 3.4%, and 8.3% for Scenarios A, B and D compared to those generated by the MSO, while the MSO generated a superior stope layout economic value for Scenario C where it was 9.7% more profitable than the DIA solution. Therefore, it can be concluded that the DIA obtained better solutions than MSO when applied to an underground stope boundary optimisation problem under variable stope width conditions as encountered in actual mining practice.

This thesis contributed to knowledge by firstly presenting an innovative way, the DIA, for exploiting the strengths of two existing algorithms, the PSO and GA. In addition, the DIA also presented an innovative way for handling variable stope width in stope boundary optimisation problems.

6.3 Recommendations

The DIA algorithm generates better solutions for underground stope boundary optimisation problem. Therefore, it is recommended that it is applied to other different case studies. In addition, the DIA is still amenable to improvements to make it more practical and acceptable in the mining industry. The author recommends that:

- Ways to reduce the computational time and computer memory requirements be explored to make the DIA amenable to mine planning requirements in terms of efficiency.
- The DIA be modified to be suitable for use in other mining methods because only longhole stoping was considered in this study.
- Modification be made on the DIA code to accommodate cut-off grade to enable optimisation process based on plant requirements.
- Stochastic techniques be used with this model to handle the input values related to geological and economic uncertainties to improve the quality of the optimised stope layout solution.

6.4 Limitations and recommendations for future research work

The major limitation of this study is that it considered the stope optimisation problem as the main step in mine design, however, the circular logic complexity presents a challenge is selecting the main area to optimise first. Furthermore, this study does not

incorporate the other three optimisation sub-problems, i.e., infrastructure and development layout, equipment selection and production scheduling.

Furthermore, there are numerous constraints to be considered for the extraction of underground stopes such as the hanging wall and footwall orientation, provision of pillars if no backfill is to be used, and changes in the dip of the orebody. Accommodating all possible constraints would have made the optimisation problem intractable and/or require large amount of computational capacity.

The number of iterations were also constrained to ensure that an acceptable solution is generated in a reasonable amount of computational time.

Based on the foregoing observations and limitations, it is suggested that future research work focuses on the following:

- Combination of all four areas of underground mine design in the optimisation process.
- The current algorithm was tailor-made for a 4E Platreef deposit, thus, it can be extended to include other mineral deposits.
- Scheduling of the optimised stope layout to generate net present value (NPV) and establish how the scheduling might change the ‘snapshot’ economic value to assess an extended planning exercise and determine which of the MSO and DIA optimal stope layouts would result in better NPV.
- Extend the current developed interface to be a stand-alone visualiser which would enhance the rapid visualisation process of the stope in 3D because the current interface relies on external mine design software for complete visualisation.
- In comparing the solution time in the four scenarios, the DIA took more time than the MSO to solve the optimisation problem. Therefore, the DIA should be modified to reduce the time taken to solve the optimisation problem to enable its implementation in mining software packages. The modification can be achieved by capping the number of iterations to a maximum of 50.

7 REFERENCES

Alam, M.N. (2016) Particle swarm optimisation: algorithm and its codes in MATLAB. INTERNET. doi:10.13140/rg.2.1.4985.3206 [Accessed 01 August 2016].

Alford, C. (1995) Optimisation in underground mine design, *APCOM XXV 1995 Conference, pp.* 213-218. INTERNET. http://alfordminingsystems.com/wordpress/wp-content/uploads/Papers/Alford_APCOM_1995.pdf [Accessed 12 September 2016].

Armitage, P.E.B. (2011) Development of the Platreef in the Northern Limb of the Bushveld Complex at Sandsloot, Mokopane District, South Africa. *PhD Thesis*, University of Greenwich.

Astolfi, A. (2006) Optimisation: an introduction. INTERNET. http://www3.imperial.ac.uk/pls/portallive/docs/1/7288263.PDF [Accessed 20 March 2016].

Ataee-Pour, M. (2000) A heuristic algorithm to optimise stope boundaries, *PhD Thesis*, Faculty of Engineering, University of Wollongong. INTERNET. http://ro.uow.edu.au/cgi/viewcontent.cgi?article=3923&context=theses [Accessed 15 February 2016].

Ataee-Pour, M. (2005a) A critical survey of the existing stope layout optimization techniques, *Journal of Mining Science,* Vol. 41, No. 5, pp. 447-466. INTERNET. http://link.springer.com/article/10.1007/s10913-006-0008-9?view=classic [Accessed 03 March 2016].

Ataee-Pour, M. (2005b) A Linear Model for Determination of Block Economic Values. *Proceedings of The 19th International Mining Congress and Fair of Turkey, IMCET 2005,* Izmir, Turkey, June 09- 12, pp. 289-29*4.*

Ataee-Pour, M. (2006) The MVN multiple pass algorithm for optimisation of stope boundaries, *Iranian Journal of Mining Engineering*, Vol. 1, No. 2, pp. 7-20. INTERNET. http://www.sid.ir/EN/VEWSSID/J_pdf/110220060201.pdf [Accessed 12 September 2016].

Bai, Q. (2010) Analysis of particle swarm optimisation algorithm, *Computer and Information Science*, Vol. 3, No. 1, pp. 180-184. INTERNET. http://ccsenet.org/journal/index.php/cis/article/viewFile/5131/4314 [Accessed 01 August 2016].

Bai, X. (2013) Optimization of underground stope with network flow methods. *PhD Thesis*, Universite de Montreal.

Cai, X., Cui, Z., Zeng, J. and Tan, Y. (2009) Individual parameter selection strategy for particle swarm optimization, pp. 89-114. DOI:10.5772/6742 [Accessed 10 November 2018].

Cawthorn, R.G. (1999) Platinum-group element mineralization in the Bushveld Complex – a critical reassessment of geochemical models. *South African Journal of Geology*, Vol. 103, No. 3, pp. 268-281.

Cawthorn, R.G. (2010) The Platinum Group Element Deposits of the Bushveld Complex in South Africa. *Platinum Metals Review*, Vol. 54, No. 4, pp. 205-215.

Chigumira, D. and Raymond, T. (2017) Platinum essentials. INTERNET. https://www.platinuminvestment.com/files/126548/WPIC_Platinum_Essentials_January_2018.pdf [Accessed 30 April 2019].

Dimitrakopoulos, R. (2011) Strategic mine planning under uncertainty, *Journal of Mining Science,* Vol. 47, No. 2, pp. 138-150.

Edorgan, G., Cigla, M., Topal, E., and Yavuz, M. (2016) Implementation and comparison of four stope boundary optimisation algorithms in an existing underground mine, *6th International Conference on Computer Applications in the Minerals Industries*, Istanbul, Turkey, 5-7 October 2016.

Erdogan, G. and Yavuz, M. (2017) Application of Three Existing Stope Boundary Optimisation Methods in an Operating Underground Mine. *IOP Conference Series: Earth and Environmental Science*, 95. pp. 1-10. doi :10.1088/1755-1315/95/4/042077 [Accessed 12 March 2019].

Ezugwu, A.E., Adeleke, O.J., Akinyelu, A.A. and Viriri, S. (2020) A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems. *Neural Computing and Applications* (2020) Vol. 32, pp. 6207-6251. INTERNET. https://doi.org/10.1007/s00521-019-04132-w [Accessed 20 October 2021].

Grieco, N. and Dimitrakopoulos, R. (2007) Managing grade risk in stope design optimisation: probabilistic mathematical programming model and application in sublevel stoping, *Mining Technology*, Vol. 116, No. 2, pp. 49-57. INTERNET. http://dx.doi.org/10.1179/174328607X191038 [Accessed 10 June 2016].

Hassan, R.; Cohanim, B., de Weck, O. and Venter, G. (2005) A comparison of particle swarm optimization and the genetic algorithm. *The Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference,* 18 - 21 April 2005, Austin, Texas, pp. 1-13. INTERNET. https://bee22.com/resources/Hassan%202004.pdf [Accessed 26 July 2016].

Henning, J.G. (2007) Evaluation of Long-Hole Mine Design Influences on Unplanned Ore Dilution. *PhD Thesis*, McGill University.

Imran, M., Hashim, R. and Khalid, N.E. A. (2013) An overview of particle swarm optimization variants. *Malaysian Technical Universities Conference on Engineering & Technology 2012, MUCET 2012 Part 4- Information and Communication Technology. Procedia Engineering,* Vol. 53 (2013), pp. 491-496. INTERNET. https://doi.org/10.1016/j.proeng.2013.02.063. https://ac.els-cdn.com/S1877705813001823/1-s2.0-S1877705813001823-main.pdf?_tid=3b148edb-fb58-44c6-896d-5597f625e613&acdnat=1552557505_c04aa1d7689a097a3ba1511c52e1bdb5 [Accessed 12 March 2016].

Ivanhoe Mines Ltd. (2014) Preliminary economic assessment (PEA), Platreef 2014. INTERNET. http://www.ivanhoemines.com/assets/docs/reports/2014-03_Platreef_PEA.pdf [Accessed 12 February 2016].

Ivanhoe Mines Ltd. (2017) Platreef 2017 feasibility study. INTERNET. https://www.ivanhoemines.com/site/assets/files/2981/platreef-feasibility-study-september-2017.pdf [Accessed 06 December 2018].

Kennedy, J. and Eberhart, R. (1995) Particle swarm optimisation, IEEE, pp. 1942-1948. INTERNET. http://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading6%201995%20particle%20swarming.pdf [Accessed 01 August 2016].

Kinnaird, J.A., Hutchinson, D., Schurmann, L., Nex, P.A.M. and de Lange, R. (2005) Petrology and mineralisation of the southern Platreef: northern limb of the Bushveld Complex, South Africa. *Mineralium Deposita,* Vol. 40, pp. 576-597. INTERNET. https://doi.org/10.1007/s00126-005-0023-9 [Accessed 11 December 2020].

Kok, D. and Lane, G.R. (2012) Case study: Using software optimization techniques for the optimal allocation of equipment in a mechanized underground mining environment, *The Journal of The Southern African Institute of Mining and Metallurgy*, Platinum 2012, pp. 35-48. INTERNET. http://www.saimm.co.za/Conferences/Pt2012/35-48_Kok.pdf [Accessed 01 August 2016].

Kurniawan, H., Sofianti, T.D., Pratama, A.T., and Tanaya, P.I. (2014). Optimizing Production Scheduling Using Genetic Algorithm in Textile Factory. *Journal of System and Management Sciences*, Vol. 4, No. 4, pp. 27-44.

Lawrence, B.W. (n.d.) Chapter 2. Considerations for Sublevel Open Stoping. INTERNET. https://www.onemine.org/document/document.cfm?docid=2670&docorgid=10 [Accessed 05 January 2021].

Lim, S.Y., Montakhab, M. and Nouri, H. (2009) A constriction factor based particle swarm optimization for economic dispatch. *The 2009 European Simulation and Modelling Conference (ESM2009),* Leicester, United Kingdom, 26-28 October 2009. http://eprints.uwe.ac.uk/13171/1/PSO_Final_Sept23.pdf [Accessed 12 October 2017].

Little, J. (2012) Simultaneous optimisation of stope layouts and production schedules for long-term underground mine planning. *PhD Thesis*, School of Mechanical and Mining Engineering, The University of Queensland.

Manyeruke, T.D. (2007) Compositional and lithological variation of the Platreef on the farm Nonnenwerth, northern lobe of the Bushveld Complex: Implications for the origin of Platinum-group elements (PGE) mineralization. *PhD Thesis,* University of Pretoria.

McCutcheon, S. (2012) Platinum-group mineral assemblages in the Platreef on Tweefontein, Northern Bushveld Complex, South Africa. *MSc dissertation,* Faculty of Science, University of the Witwatersrand.

Mmola, T. M., Nhleko, A. S. and Atherfold, J. M. (2019) Application of Particle Swarm Optimization Algorithm to Optimize Stope Layout for Underground Mines. *Proceedings of the 27th International Symposium on Mine Planning and Equipment Selection – MPES 2018*, Santiago, Chile, pp. 213-221.

Mudd, G.M. (2012) Key trends in the resource sustainability of platinum group elements. *Ore Geology Reviews*, Vol. 26 (2012), pp. 106-117.

Mulani, M. and Desai, V.L. (2018) Design and Implementation Issues in Ant Colony Optimization. *International Journal of Applied Engineering Research,* Vol. 13, No. 16, pp. 12877-12882.

Musingwini, C. (2009) Techno-economic optimisation of level and raise spacing range in planning a Bushveld Complex platinum reef conventional breast mining layout. *PhD Thesis*, University of the Witwatersrand.

Musingwini, C. (2016) Optimisation 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.

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. INTERNET. http://inside.mines.edu/~anewman/OR_Review.pdf [Accessed 01 August 2016].

Nhleko, A.S. and Musingwini, C. (2016) Estimating cost of equity in project discount rates: Comparison of the Capital Asset Pricing Model and Gordon's Wealth Growth Model. *The Journal of The Southern African Institute of Mining and Metallurgy,* Vol. 116, No. 3, pp. 215-220.

Nhleko, A.S. and Musingwini, C. (2017) Development of an algorithm for stope boundary optimization for underground mines. In *Proceedings of the 3rd Young Professionals Conference*, The Southern Institute of Mining and Metallurgy, 9-10 March 2017, Innovation Hub, Pretoria, pp. 241–252.

Nhleko, A.S. and Musingwini, C. (2019) Analysis of the particle swarm optimization (PSO) algorithm for application in stope layout optimisation for underground mines. In *Proceedings of the Mine Planner's Colloquium 2019*, The Southern African Institute of Mining and Metallurgy, Glenhove Conference Centre, Melrose Estate, Johannesburg, pp. 57-68.

Nhleko, A.S.; Tholana, T. and Neingo, P.N. (2018) A review of underground stope boundary optimization algorithms. *Resources Policy*, Volume 56, pp. 59-69.

Nikbin, V., Ataee-Pour, M., Shahriar, K. and Pourrahimian, Y. (2018) A 3D approximate hybrid algorithm for stope boundary optimization, *Computer and Operations Research.* INTERNET. https://doi.org/10.1016/j.cor.2018.05.012 [Accessed 11 January 2019].

Phillips, R.A., Ali, M.M. and Musingwini, C. (2021) A metaheuristic dual interchange algorithm (DIA) for optimising long-term production scheduling in open-pit mine planning. *Proceedings of the Application of Computers and Operations Research in*

the Minerals Industries, APCOM 2021, Online via Zoom, 30 August - 1 September 2021, The Southern African Institute of Mining and Metallurgy, pp. 167 - 180.

Rivera Letelier, O (2021). Applications of Integer Programming and Decomposition to Scheduling Problems: the Strategic Mine Planning Problem and the Bin Packing Problem with Time Lag. Optimization and Control [math.OC], *PhD Thesis*, University of Bordeaux; Universidad Adolfo Ibáñez.

Salama, A., Nehring, M. and Greberg, J. (2014) Operating value optimisation using simulation and mixed integer programming, *International Journal of Mining, Reclamation and Environment*, Vol. 28, No. 1, pp. 25-46. INTERNET. http://www.tandfonline.com/doi/abs/10.1080/17480930.2013.768019?needAccess=true [Accessed 20 June 2017].

Sandanayake, D.S.S. (2014) Stope boundary optimisation in underground mining based on a heuristic approach, *PhD Thesis*, Western Australian School of Mines, Curtin University.

Sandanayake, D.S.S., Topal, E. and Asad, M.W.A. (2015) A heuristic approach to optimal design of an underground mine stope layout, *Applied Soft Computing,* Vol. 30, pp. 595-603. INTERNET. DOI: 10.1016/j.asoc.2015.01.060 [Accessed 11 February 2016].

Schouwstra, R.P., Kinloch, E.D. and Lee, C.A. (2000) A Short Geological Review of the Bushveld Complex. *Platinum Metals Review,* Vol. 44, No. 1, pp. 33-39.

Sens, J. and Topal, E. (2009) A new algorithm for stope boundary optimisation, in *Proceedings New Leaders' 2009,* The Australasian Institute of Mining and Metallurgy, pp. 25-28.

Sens, J.J. (2011) Stope mine design optimisation using various algorithms for the Randgold Kibali project. *MSc Thesis,* Department of Geotechnology, Delft University of Technology. INTERNET. http://resolver.tudelft.nl/uuid:3d864408-d3fa-46c4-82f3-6a9270bd7edf [Accessed 12 March 2016].

Talukder, S. (2011) Mathematical modelling and application of particle swarm optimization. *Master's Thesis,* Blekinge Institute of Technology, Sweden. INTERNET. https://www.diva-portal.org/smash/get/diva2:829959/FULLTEXT01.pdf [Accessed 15 June 2018].

Thormann, L.; Buchspies, B.; Mbohwa, C.; Kaltschmitt, M. (2017) PGE Production in Southern Africa, Part I: Production and Market Trends. *Minerals* 2017, Vol. 7, No. 11:224, pp. 1-22. INTERNET. https://www.mdpi.com/2075-163X/7/11/224/htm [Accessed 06 November 2020].

Topal, E and Sens, J. (2010) A new algorithm for stope boundary optimization, *Journal of Coal Science and Engineering,* Vol. 16, No. 2, pp. 113-119.

8 APPENDICES

8.1 Python script for the dual interchange algorithm

Particle Swarm Optimisation algorithm script

```
import csv
import random
from operator import attrgetter
from datetime import datetime
from entities.block import Block
from entities.particle import Particle
from stope_layout import StopeLayout
import pandas as pd
import matplotlib.pyplot as plt

#Print starting time
start_time = datetime.now()

def read_csv():
    block_list = []
    with open("pseudoplatwithnegblocks_txt.csv", "r") as file:
        reader = csv.DictReader(file)
        i = 1
        for row in reader:
            block = Block(i, float(row["XC"]), float(row["YC"]), float(row["ZC"]), float(row["XINC"]),
float(row["YINC"]), float(row["ZINC"]), float(row["BEV"]))
            block_list.append(block)
    return block_list;

def write_csv(stopes):
    with open("output.csv", "w", newline='', encoding='utf-8') as file:
        writer = csv.writer(file)

writer.writerow(["XC","YC","ZC","XINC","YINC","ZINC","XMORIG","YMORIG","ZMORIG","NX","NY","NZ","Stope
Density (t/m^3)","SEV (US$)"])

        for stope in stopes:
            writer.writerow([stope.x_c, stope.y_c, stope.z_c, stope.x_inc, stope.y_inc, stope.z_inc,
stope.x_c, stope.y_c, stope.z_c, stope.x_inc, stope.y_inc, stope.z_inc, 0, 0, 0, stope.sev])

class PSO:

    def __init__(self):
        self.df = pd.DataFrame(columns =[ 'Epoch','Best SEV' ] )
        self.block_list = read_csv()
        self.min_x_c = min(self.block_list,key=attrgetter('x_c')).x_c
        self.min_y_c = min(self.block_list,key=attrgetter('y_c')).y_c
        self.min_z_c = min(self.block_list,key=attrgetter('z_c')).z_c
```

```
44
45          self.max_x_c = max(self.block_list,key=attrgetter('x_c')).x_c
46          self.max_y_c = max(self.block_list,key=attrgetter('y_c')).y_c
47          self.max_z_c = max(self.block_list,key=attrgetter('z_c')).z_c
48
49          print("\nBegin DIA Optimisation\n")
50          print("Goal is to maximise stope layout economic value")
51
52
53          dim = 5 #problem dimensions
54          num_particles = 25
55          max_epochs = 200
56
57          # Storing the Gbest for every iteration
58          self.epoch_list = []
59          self.best_sev_list = []
60
61          self.min_stope_length = 10.0
62          self.max_stope_length = 40.0
63          self.min_stope_width = 20.0
64          self.max_stope_width = 20.0
65          self.min_x_orig = self.min_x_c
66          self.max_x_orig = self.min_x_c
67          self.min_y_orig = self.min_y_c - 50.0
68          self.max_y_orig = self.min_y_c
69          self.min_z_orig = self.min_z_c - 20.0
70          self.max_z_orig = self.min_z_c
71
72          self.switcher_pos = {
73              0: self.zero_pos,
74              1: self.one_pos,
75              2: self.two_pos,
76              3: self.three_pos,
77              4: self.four_pos
78          }
79
80          self.switcher_vel = {
81              0: self.zero_vel,
82              1: self.one_vel,
83              2: self.two_vel,
84              3: self.three_vel,
85              4: self.four_vel
86          }
87
88          print(f"\nSetting problem dimension to {dim}")
89          print(f"Setting num_particles = {num_particles}")
90          print(f"Setting max_epochs = {max_epochs}")
91          print("\nStarting DIA");
92
93          best_particle = self.solve(dim, num_particles, self.min_x_orig, self.max_x_orig,
94 self.min_y_orig, self.max_y_orig, self.min_z_orig, self.max_z_orig, self.min_stope_length,
95 self.max_stope_length, max_epochs, self.min_stope_width, self.max_stope_width)
```

```
        print("Best position/solution found:")
        for i in range(len(best_particle.best_position)):
            print(f"x{i} = {best_particle.best_position[i]}")

        print(f"Final best SEV = {best_particle.best_sev}")
        write_csv(best_particle.stope_layout.stopes)
        print("\nEnd DIA\n")

    def sev(self, x):
        return StopeLayout(x[0], x[1], x[2], self.max_x_c, self.max_y_c, self.max_z_c, x[4], x[3], 20,
self.block_list)

    def zero_pos(self):
        return self.min_x_orig

    def one_pos(self):
        return random.randrange(0,(self.max_stope_length +
self.min_stope_length),self.min_stope_length) + self.min_y_orig

    def two_pos(self):
        return random.randrange(5,25,5) + self.min_z_orig

    def three_pos(self):
        return random.randrange(5, self.max_stope_length + self.min_stope_length,
self.min_stope_length)

    def four_pos(self):
        return random.randrange(5, self.max_stope_width + self.min_stope_width, self.min_stope_width)

    def zero_vel(self):
        return self.min_x_orig * 0.1

    def one_vel(self):
        return random.uniform(self.min_y_orig, self.max_y_orig) * 0.1

    def two_vel(self):
        return random.uniform(self.min_z_orig, self.max_z_orig) * 0.1

    def three_vel(self):
        return random.uniform(self.min_stope_length, self.max_stope_length) * 0.1

    def four_vel(self):
        return random.uniform(self.min_stope_width, self.max_stope_width) * 0.1

    def solve(self, dim, num_particles, min_x_orig, max_x_orig, min_y_orig, max_y_orig, min_z_orig,
max_z_orig, min_stope_length, max_stope_length, max_epochs, min_stope_width, max_stope_width):
        # assumes existence of an accessible sev function and a Particle class
        swarm = [None]*num_particles
        best_global_position = [None]*dim # best solution found by any particle in the swarm
        best_global_sev = 0 # bigger values better

        particle_to_return = None
```

```
148
149         print("\nSwarm Initialisation\n")
150         for i in range(len(swarm)):
151             random_position = [None]*dim
152
153             for j in range(len(random_position)):
154                 random_position[j] = self.switcher_pos.get(j, "nothing")()
155             print(f"\nInitial Particle {i + 1} position ~ z-min: {random_position[2]} ~ stope length:
156 {random_position[3]}\n")
157             print(f"\nNow: {datetime.now().strftime('%d/%m/%Y %H:%M:%S')}\n")
158
159             sl = self.sev(random_position)
160
161             error = sl.sev
162             random_velocity = [None]*dim
163             for j in range(len(random_velocity)):
164                 random_velocity[j] = self.switcher_vel.get(j, "nothing")()
165
166             swarm[i] = Particle(random_position, error, random_velocity, random_position, error)
167             swarm[i].stope_layout = sl
168
169             if swarm[i].sev > best_global_sev:
170                 particle_to_return = swarm[i]
171                 best_global_sev = swarm[i].sev
172                 best_global_position = swarm[i].position[:]
173
174         # prepare
175         w = 0.8 # inertia weight
176         c1 = 1.49445 # cognitive/local weight
177         c2 = 1.49445 # social/global weight
178         #double r1, r2; // cognitive and social randomizations
179         # double probDeath = 0.01;
180         epoch = 0
181
182         new_velocity = [None]*dim
183         new_position = [None]*dim
184         # double newSEV;
185
186         # main loop
187         begin_time = datetime.now()
188         while (epoch < max_epochs):
189             print(f"\nDIA iteration {epoch + 1}\n")
190             print(f"\nNow: {datetime.now().strftime('%d/%m/%Y %H:%M:%S')}\n")
191
192             for i in range(len(swarm)):
193                 curr_p = swarm[i] # for clarity
194                 # new velocity
195                 for j in range(len(curr_p.velocity)):
196                     r1, r2 = random.random(), random.random()
197
198                     new_velocity[j] = (w * curr_p.velocity[j]) + (c1 * r1 * (curr_p.best_position[j] -
199 curr_p.position[j])) + (c2 * r2 * (best_global_position[j] - curr_p.position[j]))
```

```
200
201                 curr_p.velocity = new_velocity[:]
202                 # new position
203                 for j in range(len(curr_p.position)):
204                     new_position[j] = curr_p.position[j] + new_velocity[j]
205
206                     if j == 0:
207                         if new_position[j] < min_x_orig:
208                             new_position[j] = min_x_orig
209                         elif new_position[j] > max_x_orig:
210                             new_position[j] = max_x_orig
211                     if j == 1:
212                         if new_position[j] < min_y_orig:
213                             new_position[j] = min_y_orig
214                         elif new_position[j] > max_y_orig:
215                             new_position[j] = max_y_orig
216                     if j == 2:
217                         if new_position[j] < min_z_orig:
218                             new_position[j] = min_z_orig
219                         elif new_position[j] > max_z_orig:
220                             new_position[j] = max_z_orig
221                     if j == 3:
222                         if new_position[j] < min_stope_length:
223                             new_position[j] = min_stope_length
224                         elif new_position[j] > max_stope_length:
225                             new_position[j] = max_stope_length
226                     if j == 4:
227                         if new_position[j] < min_stope_width:
228                             new_position[j] = min_stope_width
229                         elif new_position[j] > max_stope_width:
230                             new_position[j] = max_stope_width
231
232                 curr_p.position = new_position[:]
233                 print(f"\nMoving Particle {i + 1} to new position\n")
234                 print(f"\nNow: {datetime.now().strftime('%d/%m/%Y %H:%M:%S')}\n")
235                 print(f"\nParticle {i + 1} position ~ z-min: {curr_p.position[2]} ~ stope length:
236 {curr_p.position[3]}\n")
237
238                 stope_layout = self.sev(new_position)
239                 new_sev = stope_layout.sev
240                 curr_p.sev = new_sev
241                 curr_p.stope_layout = stope_layout
242                 print(str(curr_p))
243
244                 if new_sev > curr_p.best_sev:
245                     curr_p.best_position = new_position[:]
246                     curr_p.best_sev = new_sev
247
248                 if new_sev > best_global_sev:
249                     particle_to_return = curr_p
250                     best_global_position = new_position[:]
251                     best_global_sev = new_sev
```

```
252
253                     print(f"Iteration: {epoch + 1}")
254                     print("Best position/solution found so far:")
255                     for b in range(len(best_global_position)):
256                         print(f"x{b} = {best_global_position[b]}")
257                     print(f"Best SEV so far = {best_global_sev}")
258
259             self.df.loc[epoch,'Epoch'] = epoch
260             self.df.loc[epoch,'Best SEV'] = best_global_sev
261
262             epoch += 1
263
264             # Counting iterations
265             print("Remaining number of iterations", str(max_epochs - epoch))
266
267             # Appending the epoch and best sev lists
268             self.epoch_list.append(epoch)
269             self.best_sev_list.append(curr_p.best_sev)
270
271             # Storing Gbest per iteration
272             best_sev_file =pd.DataFrame()
273             best_sev_file["Epoch"] = self.epoch_list
274             best_sev_file["best sev"] = self.best_sev_list
275             best_sev_file.to_csv("Best_sev_output.csv", index = False)
276             print("Best sev output saved")
277
278         # show final swarm
279         print("\nProcessing complete")
280         end_time = datetime.now() - begin_time
281         print("Execution time is ", end_time)
282         print("\nFinal swarm:\n")
283         for i in range(len(swarm)):
284             print(str(swarm[i]))
285
286         return particle_to_return
287
288 pso_run = PSO()
289
290 df = pso_run.df
291 df['Epoch'] = pd.to_numeric(df['Epoch'])
292 df['Best SEV'] = pd.to_numeric(df['Best SEV']/1000000000)
293
294 #plt.close()
295 plt.figure(figsize=(15,10))
296 plt.ylim = None
297 plt.plot( 'Epoch', 'Best SEV', data=df, color='skyblue',linewidth=2)
298 # ============================================================================
299 plt.title('DIA Algorithm Performance', size=20,  y=1.)
300 plt.xlabel('Iteration number', size=15,  y=1.)
301 plt.ylabel('Global best layout (USD billion)', size=15)
302
303 plt.legend()
```

```
plt.savefig('DIA Algorithm Performance.png')
plt.show()

# Print completion time
end_time = datetime.now()

print("Optimisation initialisation time: ", start_time)
print("Optimisation completion time:", end_time)
Solution_time = end_time - start_time
print("Solution time: ", Solution_time)
```

Genetic algorithm scripts

```
import random

from .genome import Genome

class GeneticAlgorithm:
    def __init__(self, population_size, chromosome_id_and_sizes, get_random_gene, fitness_function, elitism,
                 mutation_rate, repair=None):
        self.population = []
        self.generation = 0
        self.best_fitness = 0.0
        self.average_fitness = 0.0
        self.elitism = elitism
        self.mutation_rate = mutation_rate
        self.repair = repair

        self.best_genome = Genome(chromosome_id_and_sizes, get_random_gene, fitness_function, False)

        for x in range(population_size):
            g = Genome(chromosome_id_and_sizes, get_random_gene, fitness_function, True)
            if self.repair is not None:
                self.repair(g)
            self.population.append(g)

    def new_generation(self):
        if len(self.population) <= 0:
            return

        if len(self.population) > 0:
            self.calculate_fitness()
            self.population.sort(key=lambda x: x.fitness, reverse=True)

        new_population = []

        for i in range(len(self.population)):
            if i < self.elitism:
                new_population.append(self.population[i])
            else:
```

```
38                # https://gamedev.stackexchange.com/questions/116832/random-number-in-a-range-biased-toward-the-low-end-of-
39 the-range
40                index1 = int(abs(random.random() - random.random()) * len(self.population))
41                parent1 = self.population[index1]
42                index2 = int(abs(random.random() - random.random()) * len(self.population))
43                parent2 = self.population[index2]
44
45                child = parent1.crossover(parent2)
46
47                for chromosome in child.chromosomes:
48                    chromosome.mutate(self.mutation_rate, i)
49
50                if self.repair is not None:
51                    self.repair(child)
52
53                new_population.append(child)
54
55        self.population = new_population
56
57        self.generation += 1
58
59    def calculate_fitness(self):
60        fitness_sum = 0
61        best = self.population[0]
62
63        for i in range(len(self.population)):
64            fitness_sum += self.population[i].calculate_fitness(i)
65
66            if self.population[i].fitness > best.fitness:
67                best = self.population[i]
68
69        self.best_genome.fitness = self.best_fitness = best.fitness
70        self.average_fitness = fitness_sum / len(self.population)
71
72        for chromosome in best.chromosomes:
73            chromosome_to_update = next((x for x in self.best_genome.chromosomes if x.id == chromosome.id), None)
74            if chromosome_to_update is not None:
75                chromosome_to_update.genes = chromosome.genes[:]
```

Chromosome script

```
1  import random
2
3
4 class Chromosome:
5
6     def __init__(self, id, size, get_random_gene, population_index, should_init_genes=True):
7
8         self.id = id
9         self.size = size
```

```
        self.genes = [None]*size
        self.get_random_gene = get_random_gene

        if should_init_genes:
            for i in range(self.size):
                self.genes[i] = self.get_random_gene()

    def mutate(self, mutation_rate, population_index):
        for i in range(self.size):
            if random.random() < mutation_rate:
                self.genes[i] = self.get_random_gene()
```

Genome script

```
#Genetic algorithm

# Genome class


import random
from .chromosome import Chromosome

class Genome:
    def __init__(self, chromosome_id_and_sizes, get_random_gene, fitness_function, should_init_genes=True):
        self.chromosome_id_and_sizes = chromosome_id_and_sizes
        self.get_random_gene = get_random_gene
        self.fitness_function = fitness_function
        self.chromosomes = []
        self.fitness = 0.0

        for chromosome_id, size in chromosome_id_and_sizes.items():
            new_chromosome = Chromosome(chromosome_id, size, get_random_gene, should_init_genes)
            self.chromosomes.append(new_chromosome)

    def calculate_fitness(self, index):
        self.fitness = self.fitness_function(index)
        return self.fitness

    def crossover(self, other_parent):
        child = Genome(self.chromosome_id_and_sizes, self.get_random_gene, self.fitness_function, False)

        for chromosome_id, size in self.chromosome_id_and_sizes.items():
            my_genes = next((x for x in self.chromosomes if x.id == chromosome_id), None).genes
            other_parent_genes = next((x for x in other_parent.chromosomes if x.id == chromosome_id),
None).genes
            child_genes = next((x for x in child.chromosomes if x.id == chromosome_id), None).genes

            for i in range(len(my_genes)):
                if random.random() < 0.5:
                    child_genes[i] = my_genes[i]
```

```
                else:
                    child_genes[i] = other_parent_genes[i]

        return child
```

Level optimiser

```
import random

from ga.genetic_algorithm import GeneticAlgorithm
from entities.point import Point
from entities.rectangle import Rectangle
from entities.stope import Stope

class LevelOptimiser:

    def __init__(self, max_stope_origin_points, min_x, max_x, min_y, max_y, level, stope_x_dimension, stope_y_dimension):
        self.max_points = max_stope_origin_points;
        self.min_x = min_x;
        self.max_x = max_x;
        self.min_y = min_y;
        self.max_y = max_y;
        self.level = level;
        self.stope_x_dimension = stope_x_dimension;
        self.stope_y_dimension = stope_y_dimension;

        self.population_size = 100;
        self.mutation_rate = 0.2;
        self.elitism = 2;

        self.ga = GeneticAlgorithm(self.population_size, { 1: self.max_points }, self.get_random_point,
self.fitness_function, self.elitism, self.mutation_rate)

    def run(self):
        can_continue = True
        while(self.ga.generation and can_continue):
            self.ga.new_generation()

            g = self.ga.best_genome
            s = self.create_stopes(g)
            can_continue = False if not s else True
            print(f"Optimising level:{3} - Gen:{0} | Best Fitness: {1} | Average Fitness: {2}", self.ga.generation,
self.ga.best_fitness, self.ga.average_fitness, self.level.z)

        genome = self.ga.best_genome
        stopes = self.create_stopes(genome)
        return stopes

    def get_random_point(self):
        x = random.uniform(self.min_x, self.max_x)
```

```
        y = random.uniform(self.min_y, self.max_y)
        return Point(x, y)

    def fitness_function(self, index):
        genome = self.ga.population[index]
        stopes = self.create_stopes(genome)
        return sum(s.sev for s in stopes)

    def create_stopes(self, genome):
        stopes = []
        rectangles = []
        local_blocks = self.level.blocks[:]

        for gene in [g for c in genome.chromosomes for g in c.genes]:
            rect = Rectangle(gene.x, gene.y, self.stope_x_dimension, self.stope_y_dimension)
            if any(r.intersects_with(rect) for r in rectangles):
                continue

            rectangles.append(rect)
            stope = Stope(gene.x, gene.y, self.level.z, self.stope_x_dimension, self.stope_y_dimension, self.level.z_inc)

            for lb in local_blocks:
                overlap = lb.xy_rectangle.intersect(stope.xy_rectangle)
                if overlap.area >= 0.0 and overlap.area > lb.xy_rectangle.area * 0.3:
                #if overlap.area > 0.0:
                    stope.blocks.append(lb)

            #local_blocks = [x for x in local_blocks if x not in stope.blocks]
            if stope.sev > 0.0:
                stopes.append(stope)

        return stopes
```

Stope optimiser

```
# Stope optimisation
import math
from entities.level import Level
from entities.rectangle import Rectangle
from level_optimiser import LevelOptimiser

class StopeLayout:

    def __init__(self, x_orig, y_orig, z_orig, x_max, y_max, z_max, x_inc, y_inc, z_inc, all_blocks):
        self.x_orig = x_orig
        self.y_orig = y_orig
        self.z_orig = z_orig
        self.x_max = x_max
        self.y_max = y_max
```

```
        self.z_max = z_max
        self.x_inc = x_inc
        self.y_inc = y_inc
        self.z_inc = z_inc
        self.all_blocks = all_blocks
        self.levels = []
        self.stopes = []

        self.generate_levels()
        self.optimise_levels()

    def generate_levels(self):
        local_blocks = self.all_blocks[:]
        level_count = math.ceil((self.z_max - self.z_orig) / self.z_inc)
        print(f"Level Count {level_count}")
        print(f"local_blocks Count {len(local_blocks)}")
        for i in range(level_count):
            level = Level(self.z_orig + float(self.z_inc * i), self.z_inc, [], Rectangle(self.x_orig,
self.z_orig + float(self.z_inc * i), self.x_max + self.x_inc - self.x_orig, self.z_inc))

            for lb in local_blocks:
                overlap = lb.xz_rectangle.intersect(level.xz_rectangle)
                if overlap.area > 0.0 and overlap.area > lb.xy_rectangle.area * 0.1:
                    level.blocks.append(lb)
                    lb.appended_to_level = True

            print(f"level blocks Count {len(level.blocks)}")

            local_blocks = [x for x in local_blocks if x not in level.blocks]
            self.levels.append(level)

        #find first block not appended
        first_not_appended = next((x for x in local_blocks if not x.appended_to_level), None)
        #find out why not added to one of the levels
        if first_not_appended:
            print(f"Found! x_c: {first_not_appended.x_c} z_c: {first_not_appended.z_c} x_inc:
{first_not_appended.x_inc} z_inc: {first_not_appended.z_inc}")
            for l in self.levels:
                print(f"x: {l.xz_rectangle.x} y: {l.xz_rectangle.y} length: {l.xz_rectangle.length} width:
{l.xz_rectangle.width}")
                overlap = first_not_appended.xz_rectangle.intersect(l.xz_rectangle)
                print(f"Overlap area: {overlap.area}")

            exp_l = next((x for x in self.levels if first_not_appended.z_c - x.xz_rectangle.y > 0.0), None)
            print(f"level blocks Count {len(exp_l.blocks)}")
            for a in exp_l.blocks:
                print(f"x_c: {a.x_c} z_c: {a.z_c}")
            pass

        for b in local_blocks:
            if not b.appended_to_level:
                print(f"Found! x_c: {b.x_c} y_c: {b.y_c} x_inc: {b.x_inc} z_inc: {b.z_inc}")
```

```
67
68      def optimise_levels(self):
69          print("optimising levels")
70          max_x_origin_points = math.ceil((self.x_max - self.x_orig) / self.x_inc)
71          max_y_origin_points = math.ceil((self.y_max - self.y_orig) / self.y_inc)
72          max_stope_origin_points = max_x_origin_points * max_y_origin_points
73          i = 1
74          for level in self.levels:
75              print(f"optimising level{i}")
76              i += 1
77              level_optimiser = LevelOptimiser(max_stope_origin_points, self.x_orig, self.x_max, self.y_orig,
78  self.y_max, level, self.x_inc, self.y_inc)
79              self.stopes.extend(level_optimiser.run())
80
81      @property
82      def sev(self):
83          return sum(s.sev for s in self.stopes)
```

8.2 DIA interface layout

This section presents the guidelines to using the stope optimisation application based on the dual interchange algorithm. Figure 8.1 shows that the first step in the optimisation process is to upload the geological block model file which must be in a csv format. Table 8.1 shows the data required in the geological block model in each column and its data type.

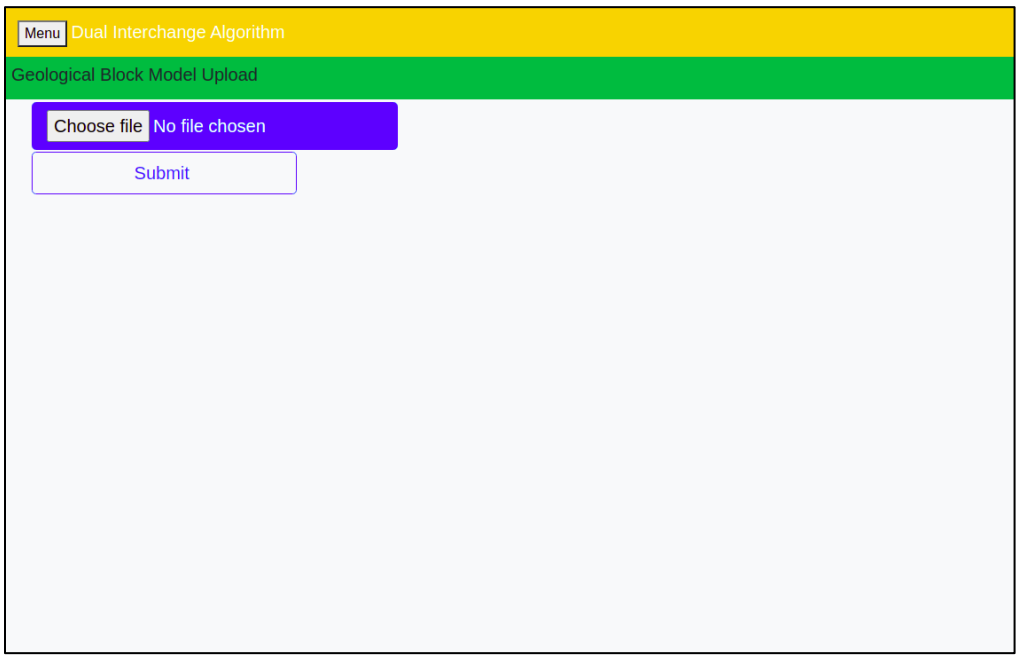


Figure 8.1 Geological block model upload

Table 8.1 Description of columns in the geological block model

Column Heading	SI Unit	Description	Data Type
XC	m	X coordinate of a block in the block model	Float
YC	m	Y coordinate of a block in the block model	Float
ZC	m	Z coordinate of a block in the block model	Float
XINC	m	Block width	Float
YINC	m	Block length	Float
ZINC	m	Block height	Float
XMORIG	m	Origin of the block model along the X-axis	Float
YMORIG	m	Origin of the block model along the Y-axis	Float
ZMORIG	m	Origin of the block model along the Z-axis	Float
NX	-	Number of cells needed in X-axis	Float
NY	-	Number of cells needed in Y-axis	Float
NZ	-	Number of cells needed in Z-axis	Float
VOLUME	m^3	Space occupied by a block in 3D	Float
DENSITY	$t.m^{-3}$	Material density of a block	Float
SEV	US $	Stope economic value	Float

After the geological block model file has been uploaded, its content is immediately displayed on the screen below the “Geological Block Model Upload” tab. Figure 8.2 depicts the input parameters required to create an economic block model by calculating economic values for individual blocks. Once the user has input the required parameters, the user must click on the “Create economic block model” tab and after the creation process is complete, the user can display the created block model for visualisation purposes.

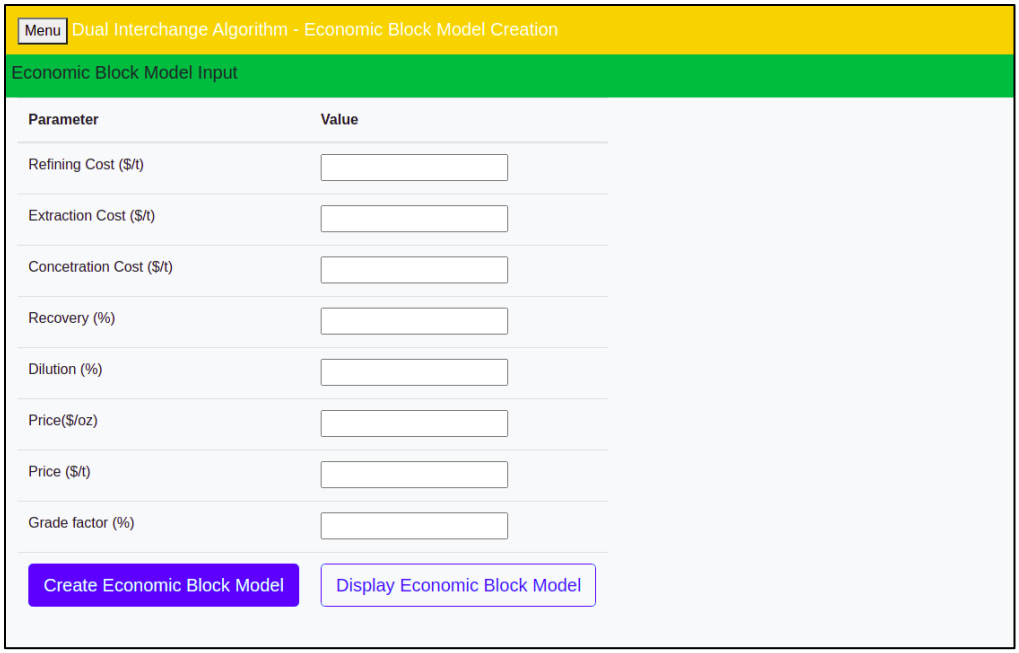


Figure 8.2 Economic block model creation

Figure 8.3 shows the interface for implementation of the stope dimension and dip of the orebody. The stope dimensions may be fixed or varied along the x- and z-axes and are fixed along the z-axis to ensure that the level constraint is not violated.

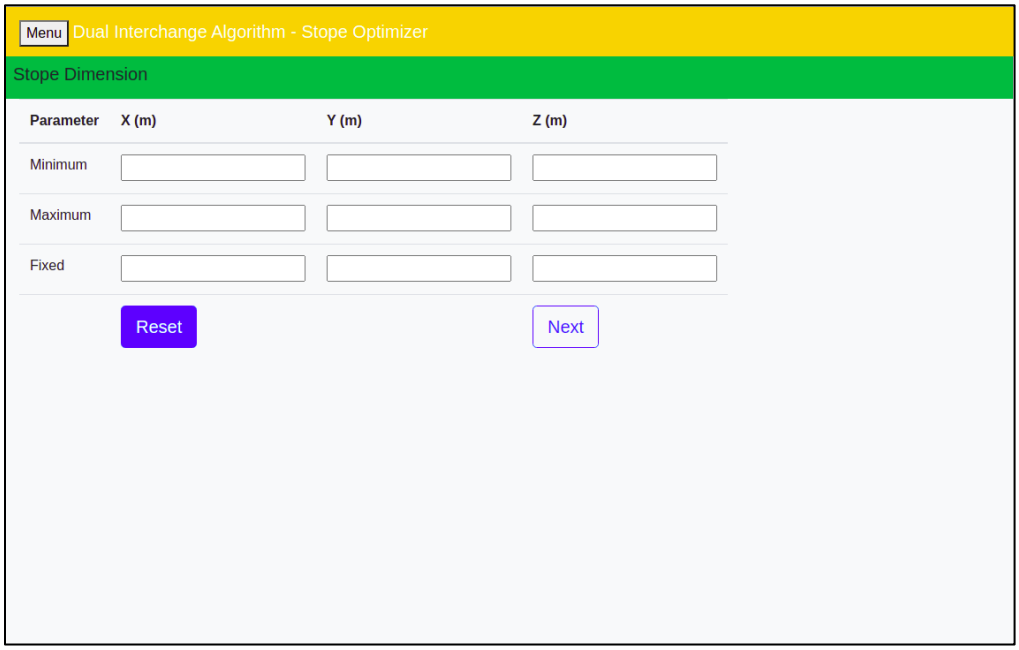


Figure 8.3 Stope dimension input

Figure 8.4 illustrates the stope optimiser implementation where the user has to specify the input parameters for the dual interchange algorithm. After the user has defined the parameters according to specific requirements, the user must click on the “Optimise” tab to initialise the optimisation process to generate the optimum stope layout. The stope optimiser generates all possible stopes and extract stopes with positive stope economic values (SEVs) while constrained by the set constraints such as level, non-overlap and unique stopes constraints.

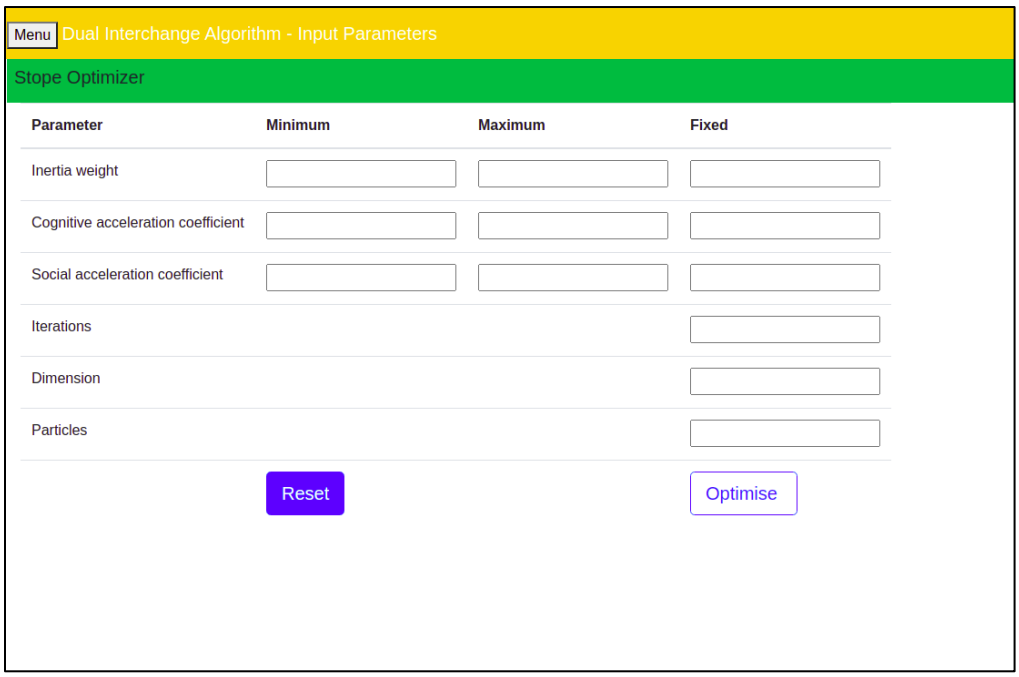


Figure 8.4 Specification of the DIA optimisation parameters

After the optimisation process is complete, the generated global best economic stope layout output is summarised as shown in Figure 8.5. The detailed output of the global best solution is saved as a csv file in the database. The user can use the generated csv file to create a prototype of the block model which can be visualised in the MSO module in Datamine software.

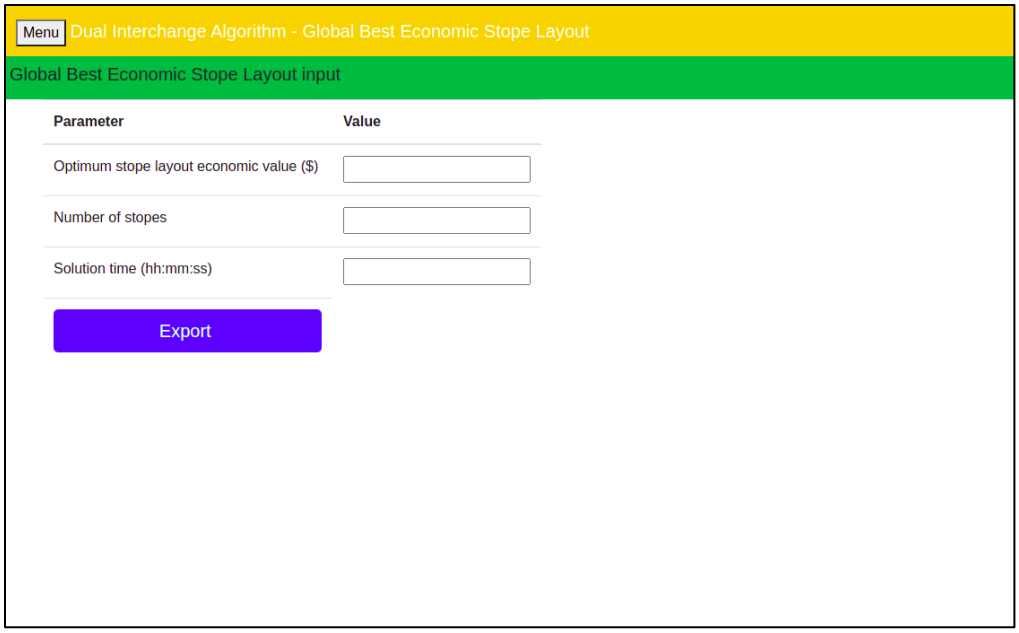


Figure 8.5 Global best economic layout solution

8.3 MSO application procedure

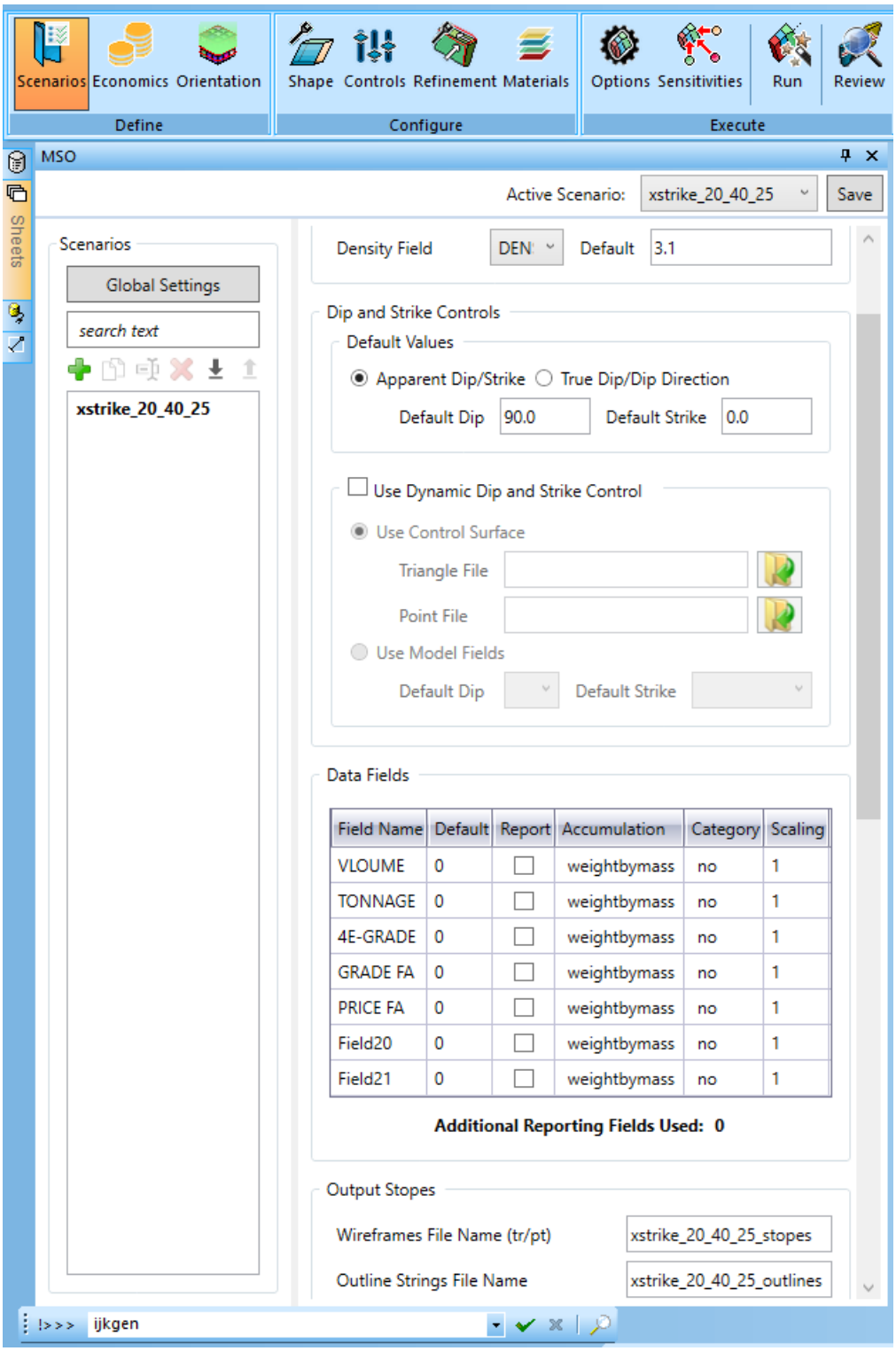


Figure 8.6 Stage 1: stope optimisation scenario definition

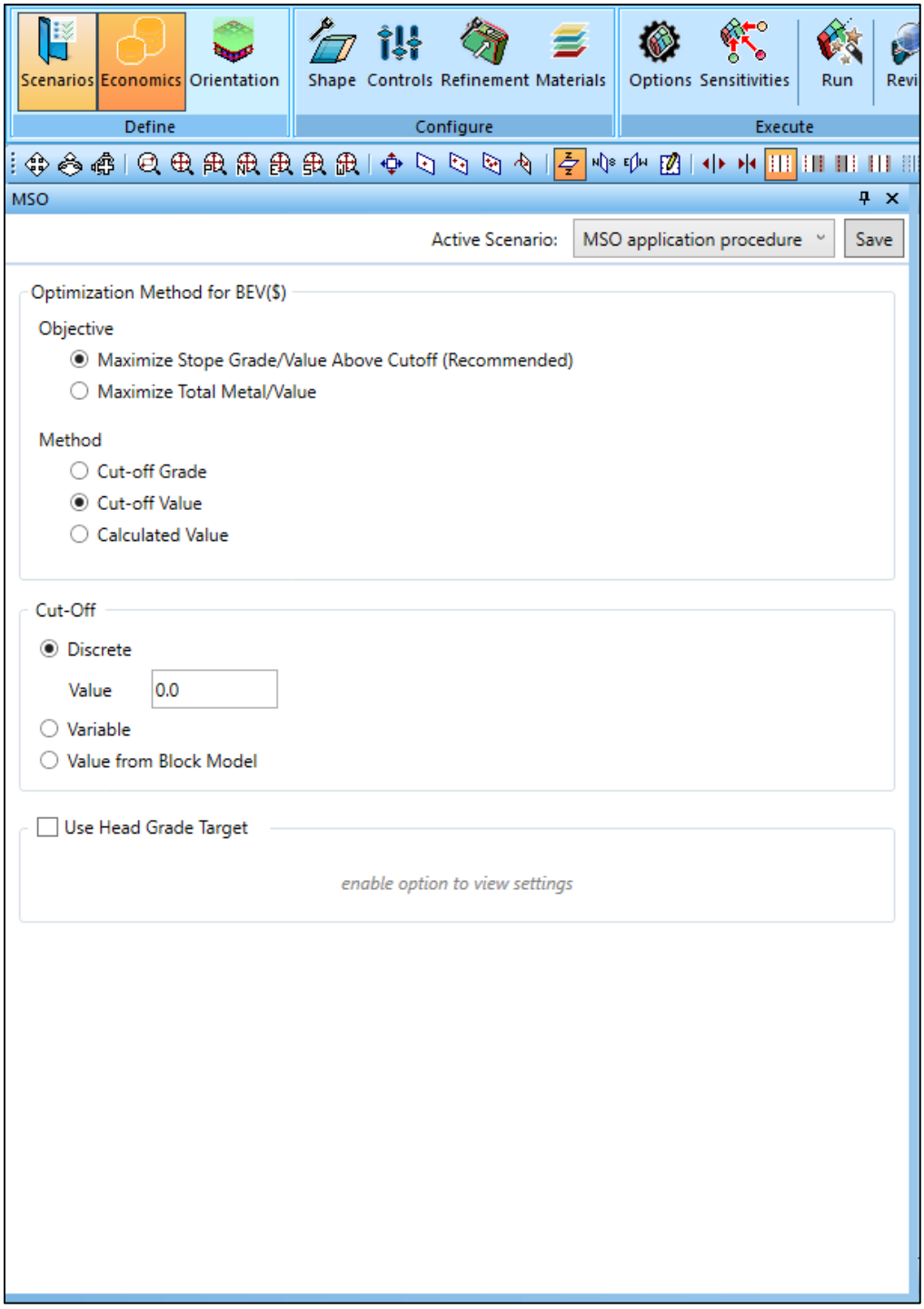


Figure 8.7 Stage 2: optimisation objective function definition

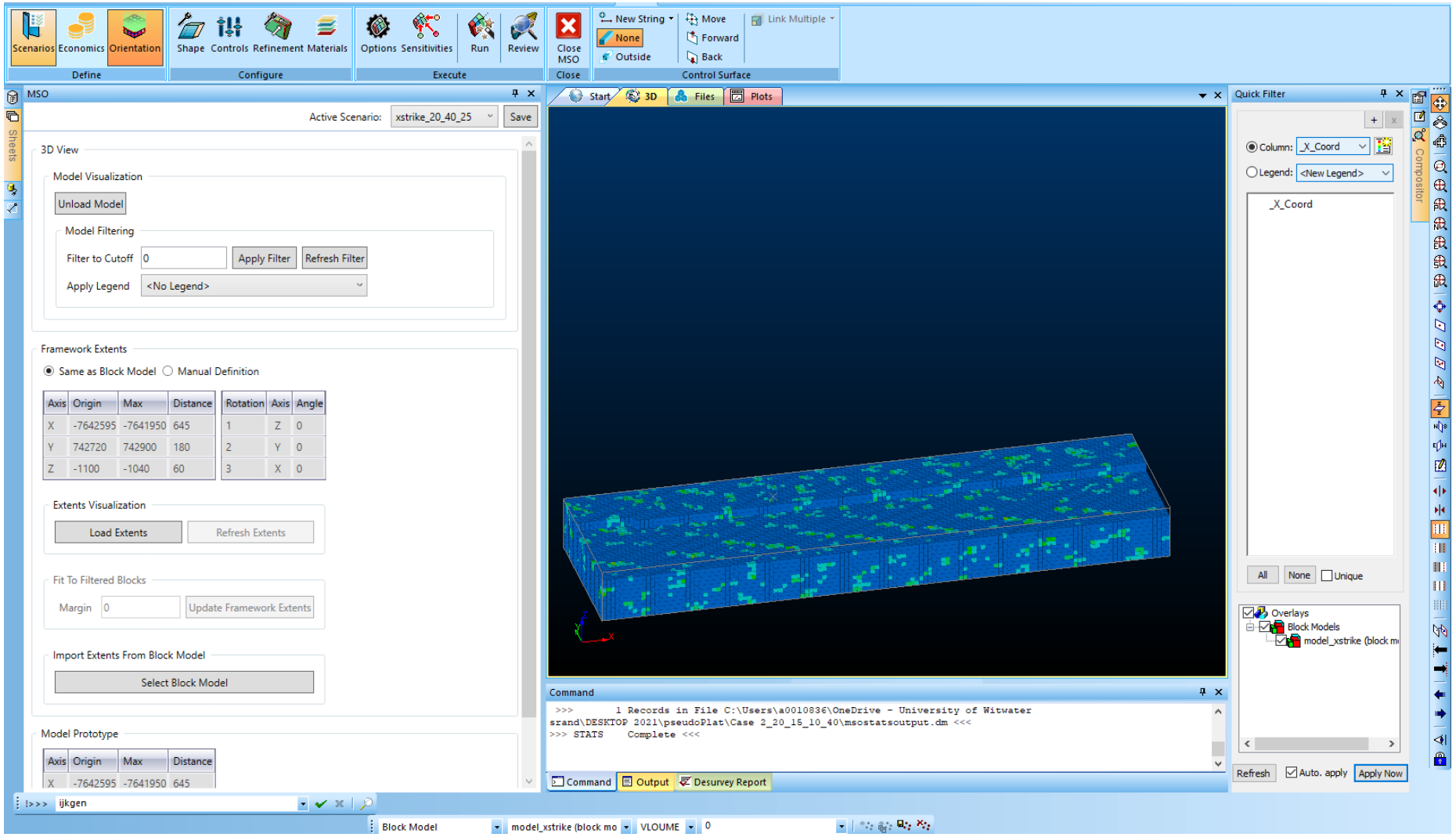


Figure 8.8 Stage 3: parameters for stope-shape framework specification

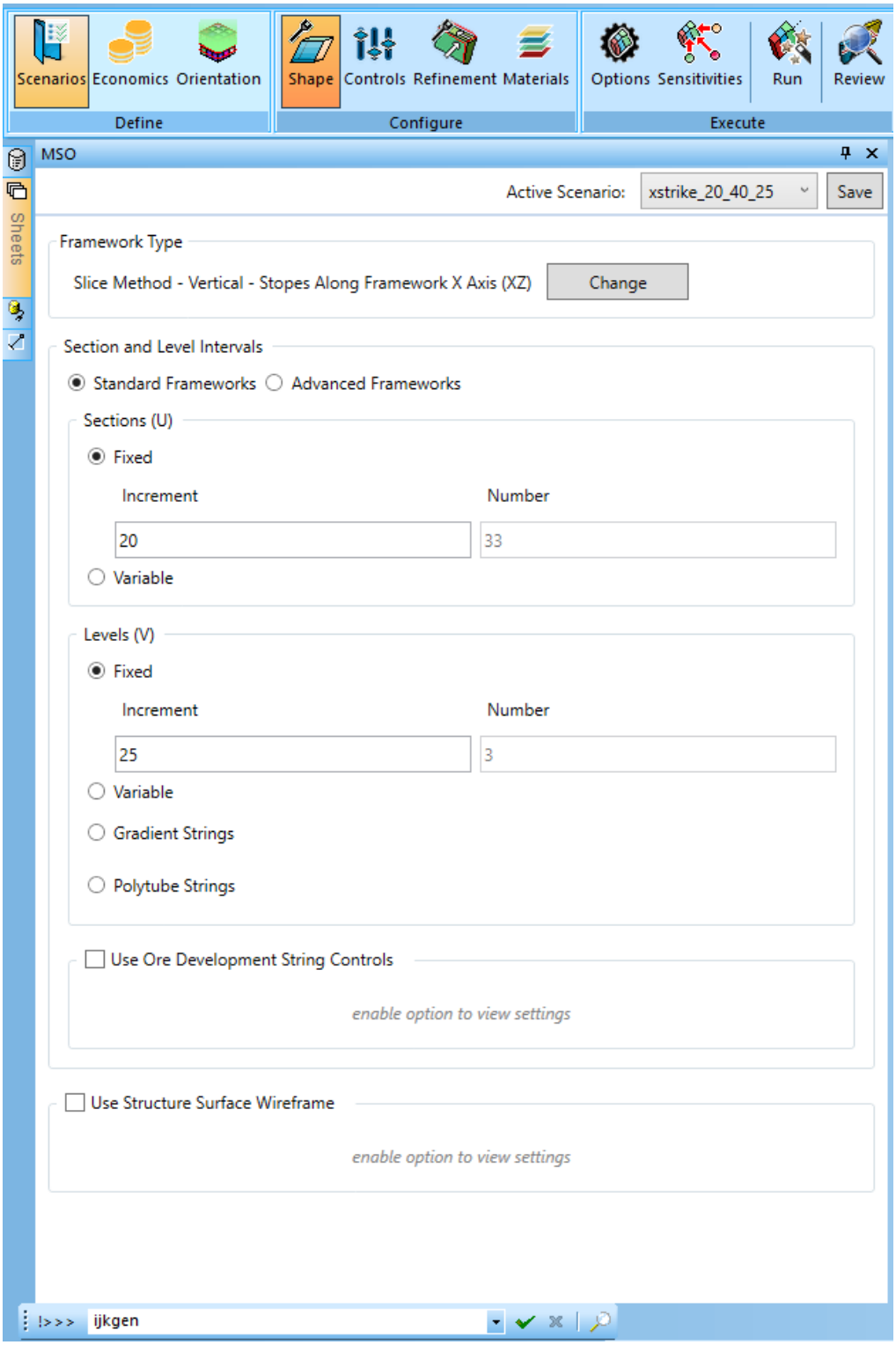


Figure 8.9 Stage 4: stope generation process definition

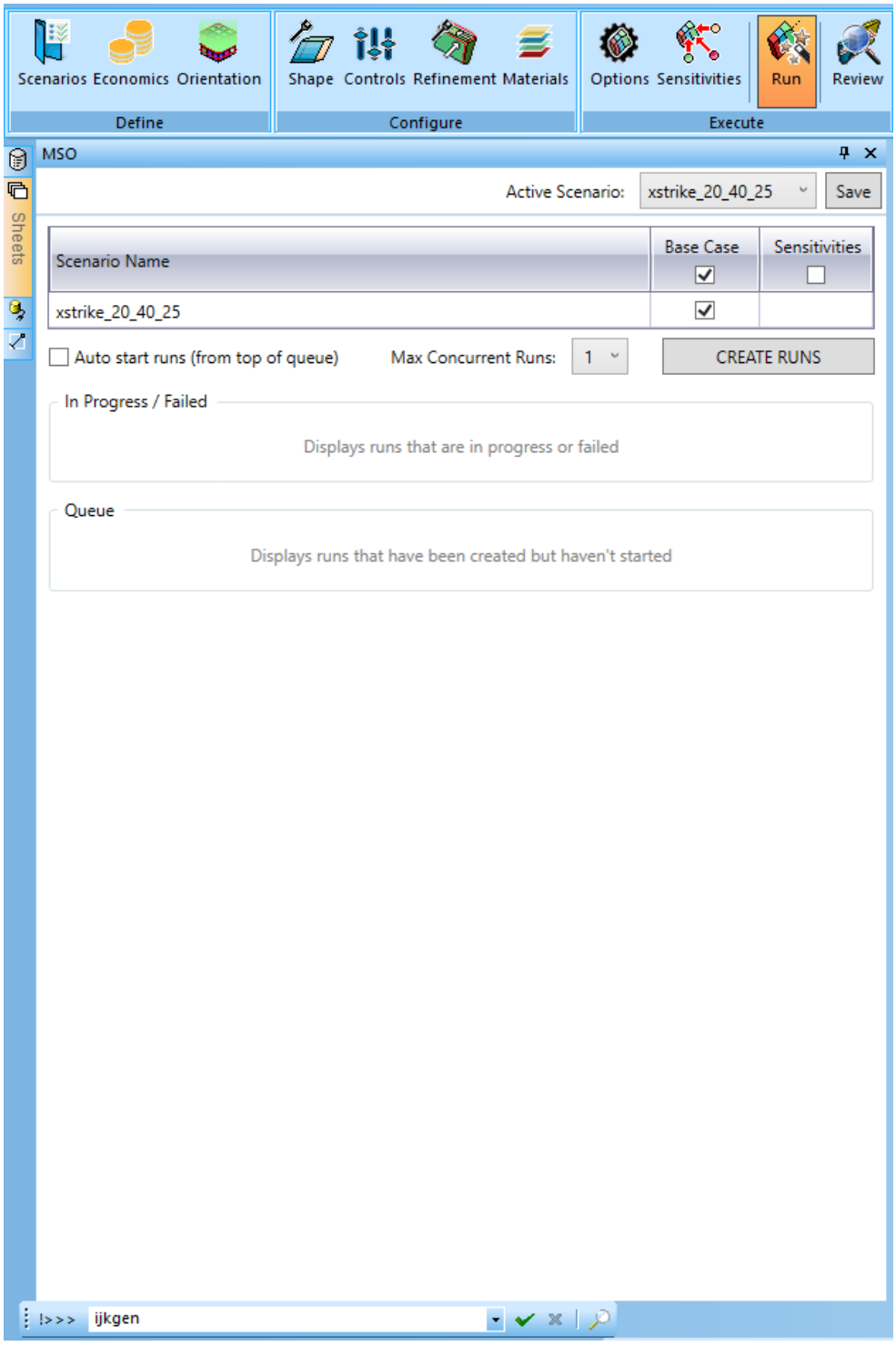


Figure 8.10 Stage 5: stope-shape framework specification

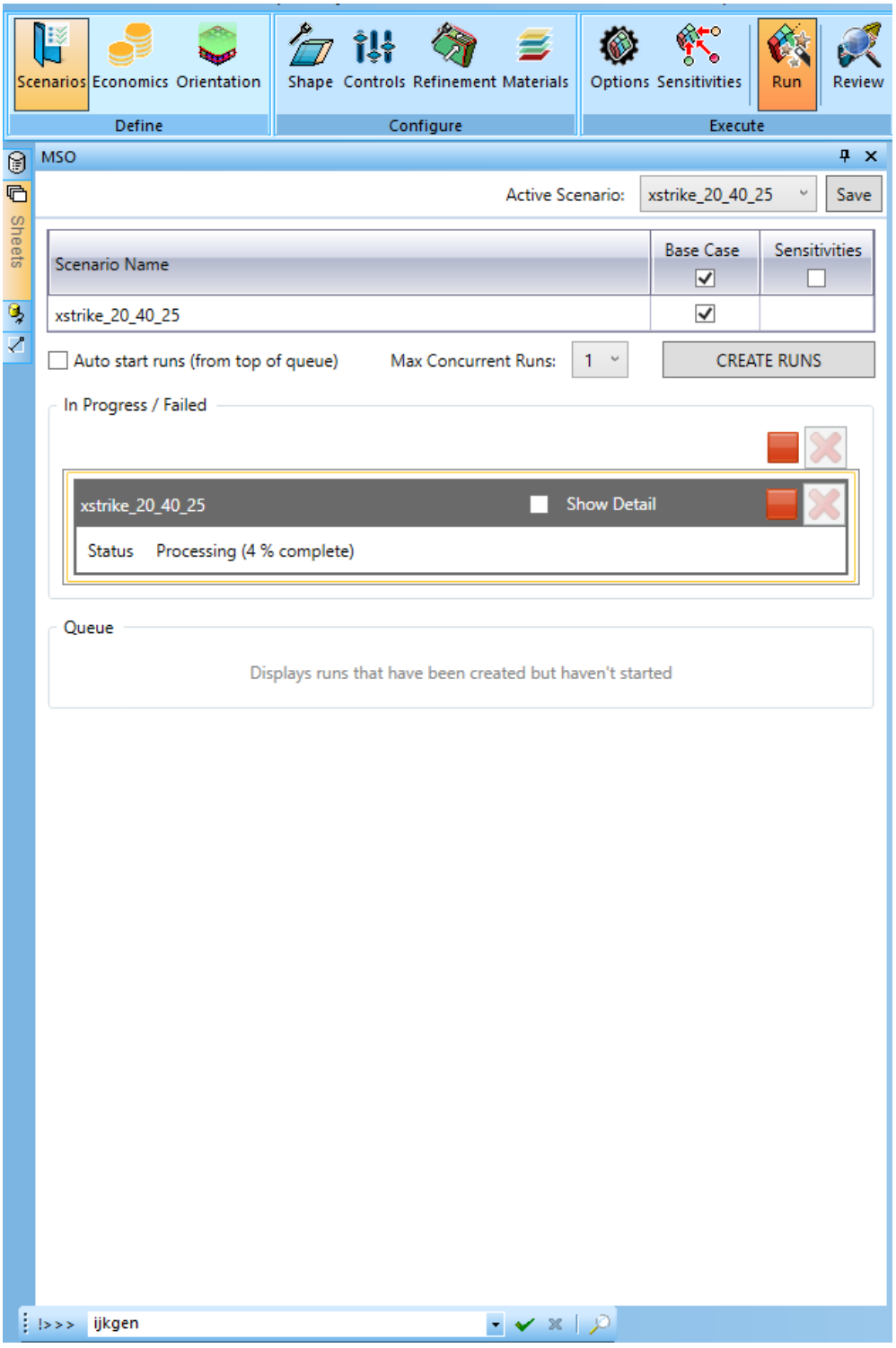


Figure 8.11 Stage 6: execution of an MSO run based on the defined scenario