Aggregation and Mathematical Programming for Long-Term Open Pit Production Planning

by

Mohammad Tabesh

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in
Mining Engineering

Department of Civil and Environmental Engineering
University of Alberta

ABSTRACT

The objective of this thesis is to develop, implement and verify a theoretical framework based upon aggregation and mathematical programming for solving the long-term open pit production planning problem. The goal is to closely estimate the maximum net present value of the operation by providing an optimum and practical mining, processing and stockpiling schedule for the open pit mining operation while respecting the technical and operational constraints. As stated by many researchers in the area and illustrated in the forth chapter of this thesis, using blocks as units of planning results in over-estimation of the operation's profitability and flexibility. Thus, we introduced a clustering algorithm along with a mathematical formulation that can result in good production plans that result in high NPV, are practical and do not under- or over-estimate the value of the operation.

In this thesis, we introduced, implemented and verified a specifically-designed clustering technique based on agglomerative hierarchical clustering, in order to aggregate blocks into mining-units that are homogenous in rock type and grade, and have mineable shape and size. We designed the algorithm, developed the codes and implemented and tested the algorithm on small test datasets and large real-size deposits to evaluate the performance of the algorithm. We showed that we can balance the clustering control parameters to obtain clusters of blocks aligned with the clustering purpose such as long-term planning units, ore polygons and blast patterns.

Moreover, we formulated, implemented and verified a mixed integer linear programming model, for long-term open pit production planning problem, which can use two different sets of units for making mining and processing decisions. Our model is able to simultaneously determine the optimum stockpiling strategy and the optimum mining and *processing schedule in reasonable processing time. We implemented our model on a small test dataset as well as real-size deposits to understand the effects of using different units of planning for making mining and processing decisions. We showed that we can obtain practical and optimum production schedules for real-size open pit mines in a reasonable time by benefiting from the clustering technique we introduced. Moreover, we showed how incorporating stockpile optimization in long-term production scheduling can increase the net present value of the operation. We benchmarked our model against commercial scheduling software to illustrate the flexibility and accuracy of our planning approach.*

The main contributions of this thesis to the mining body of knowledge are (i) introducing a clustering algorithm that creates aggregates of blocks homogenous in rock type and grade, with controlled shape and size, and respects the mining direction as well as other constraints and boundaries, (ii) introducing an MILP formulation for the long-term open pit production planning problem, with dynamic cut-off grade, that maximizes the net present value of the mine, by using two different units for making mining and processing decisions, while respecting operational and technical constraints, and (iii) incorporating stockpiling in the long-term scheduling to simultaneously optimize the mining, processing and stockpiling strategies.

PREFACE

This thesis is an original work by Mohammad Tabesh. Parts of this work are published as Tabesh, M., and Askari-Nasab, H. (2011). Two-stage clustering algorithm for block aggregation in open pit mines. Transactions of the Institutions of Mining and Metallurgy, Section A: Mining Technology, 120(3), 158-169; Tabesh, M., and Askari-Nasab, H. (2013). Automatic Creation of Mining Polygons using Hierarchical Clustering Techniques. Journal of Mining Science, 49(3), 426-439. I was responsible for designing the algorithms, running the case-studies, documenting the results and writing the manuscripts. H. Askari-Nasab was the supervisory author and was involved with concept formation and manuscript composition.

The case-study in Section 4.10 of this thesis has been published as Tabesh, M., Mieth, C., and Askari-Nasab, H. (2014). A Multi-Step Approach To Long-Term Open-Pit Production Planning. International Journal of Mining and Mineral Engineering, 5(4), 273-298. The publication has been a collaborative work from two research projects. I was responsible for designing the clustering algorithm, running the case-studies on clustering and MILP, documenting the results and writing the manuscript. C. Meith was responsible for designing the pushback determination procedure which was used in the case-studies. H. Askari-Nasab was the supervisory author and was involved with concept formation and manuscript composition.

This Thesis is Proudly Dedicated:

To my wonderful parents, Abolfazl and Saideh
who supported, nurtured and encouraged me to get to this point,

To my lovely wife, Rezvan
with whom I share the ups and downs of my life,

And, to my best friend in the world, Masoud
without whom my life would have been different.

AKNOWLEDGMENT

Obtaining a PhD in Engineering major has been a goal in the past few years of my life. Now that I have defended my thesis and accomplished the task, I like to thank my family, friends, colleagues and professors who helped me get to this point of my life.

First of all, I appreciate what my parents have done for me. They have helped me overcome the obstacles on my way and be sure that there is always someone I can count on. Second, I want to thank my wife for being patient with me when I was overwhelmed with my research and had less time to be with her.

I acknowledge my supervisor’s help and guidance and thank him for providing this opportunity for me to do research in a prestigious university. Dr. Askari-Nasab helped me learn mining engineering methods and concepts and helped me direct my research to a valuable goal. Moreover, I would like to thank my colleagues Dr. Ben-Awuah and Dr. Pourrahiman for their collaborations and consultations when I was new to the mining engineering area. I want to specially thank my friend Dr. Badiozamani for suggesting the hierarchical clustering idea and for years of working together and sharing ideas and expertise. In addition, I am grateful to my colleagues Ibrahim, Elmira, Samira, Shiv, Ali, Firouz, Saha, Nisarg and Luisa for being good friends that made my office a nice place to work in. There are many others inside and outside this university who helped me finish this program and I appreciate their role in my personal and academic life.

Mohammad Tabesh

September 2015

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BIP	Binary Integer Programming
CV	Coefficient of Variation
DCF	Discounted Cash Flow
DDF	Destination Dilution Factor
DP	Dynamic Programming
EPGAP	Relative MIP Gap Tolerance
GP	Goal Programming
LP	Linear Programming
LTOPP	Long-Term Open Pit Production Planning
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
NPV	Net Present Value
QP	Quadratic Programming
RU	Rock Unity
TS	Tabu Search

LIST OF NOMENCLATURE

Sets

B_c	Set of blocks in cluster c
S^m	For each mining unit m, there is a set of mining units (S^m) that have to be extracted prior to extracting mining unit m to respect slope and precedence constraints
U^m	Each mining unit m is divided into a set of processing units. U^m is the set of processing units that are contained in mining unit m
SC	Set of stockpile destinations

Indices

$d \in \{1,...,D\}$	Index for material destinations
$m \in \{1,...,M\}$	Index for mining units
$p \in \{1,...,P\}$	Index for processing units
$s \in \{C+1,...,C+S\}$	Index for stockpiles
$c \in \{1,...,C\}$	Index for processing plants
$e \in \{1,...,E\}$	Index for elements
$t \in \{1,...,T\}$	Index for scheduling periods

Parameters

R_{ij}	Penalty assigned if blocks are from different rock types
C_{ij}	Penalty assigned to blocks not located above the same cluster
D_{ij}	Penalty assigned if blocks are not assigned to the same destination
$\tilde{L}_{ij}$	Normalized distance value between blocks i and j
$\tilde{G}_{ij}$	Normalized grade difference between blocks i and j
W_D	Distance weight
W_G	Grade difference weight
IC_c	Intra-cluster similarity of cluster c
n_c	Number of blocks in cluster c
$\overline{IC}$	Average intra-cluster similarity of the bench

NA	Number of precedence arcs between clusters of this bench and the lower bench
W_c	Weight for intra-cluster similarity measure
W_n	Weight for number of precedence arcs
D	Number of material destinations (including processing plants, waste dumps and stockpiles)
M	Total number mining units
P	Total number of processing units
S	Number of stockpiles
E	Number of elements in the block model
T	Number of scheduling periods
$\overline{MC}^t$	Upper bound on the mining capacity in period t
$\underline{MC}^t$	Lower bound on the mining capacity in period t
$\overline{PC}_c^t$	Maximum tonnage allowed to be sent to plant c in period t
$\underline{PC}_c^t$	Minimum tonnage allowed to be sent to plant c in period t
$\overline{G}_c^{t,e}$	Upper limit on the allowable average grade of element e at processing plant c in period t
$\underline{G}_c^{t,e}$	Lower limit on the allowable average grade of element e at processing plant c in period t
$\overline{PC}_s^t$	Maximum tonnage allowed to be sent to stockpile s in period t
$\underline{PC}_s^t$	Minimum tonnage allowed to be sent to stockpile s in period t
$\overline{G}_s^{t,e}$	Upper limit on the allowable grade of element e to be sent to stockpile s in period t
$\underline{G}_s^{t,e}$	Lower limit on the allowable grade of element e to be sent to stockpile s in period t
$G_s^{t,e}$	Average predetermined reclamation grade of element e from stockpile s in period t
$G^{t,e}$	Average calculated reclamation grade of element e from the stockpile in period t
s_m	Number of predecessors of mining unit m (number of members of S^m)
o_m	Total ore tonnage in mining unit m
w_m	Total waste tonnage in mining unit m
o_p	Total ore tonnage in processing unit p
w_p	Total waste tonnage in processing unit p
c_m^t	Unit discounted cost of mining material from mining unit m in period t
$r_{p,c}^t$	Unit discounted revenue of sending material from processing unit p to processing destination c in period t minus the processing costs

$r_{s,c}^{t,e}$	Unit discounted revenue of processing every unit of element e from stockpile s to processing destination c in period t minus the processing and rehandling costs
$r_{c}^{t,e}$	Unit discounted revenue of processing every unit of element e from the stockpile to processing destination c in period t minus the processing and rehandling costs
g_{p}^{e}	Average grade of element e in processing unit p

Decision Variables

$y_m^t \in [0,1]$	Continuous decision variable representing the portion of mining unit m extracted in period t
$x_{p,d}^t \in [0,1]$	Continuous decision variable representing the portion of ore tonnage in processing unit p extracted in period t and sent to destination d
$b_m^t \in \{0,1\}$	Binary decision variable indicating if all the predecessors of mining unit m are completely extracted by or in period t
$f_{s,c}^t \geq 0$	Continuous decision variable representing the tonnage sent from stockpile s to processing plant c in period t
$f_{c}^t \geq 0$	Continuous decision variable representing the tonnage sent from the stockpile to processing plant c in period t

INTRODUCTION

1. INTRODUCTION

1.1. Introduction

Mining is a multistage process from exploration to processing ore in order to make ready-to-sell final product from earth resources. Mine planning is a step in which the block model of the resource, developed in the estimation step using Geostatistical methodologies, is used and the decision on the order of extraction of blocks, over a specific time horizon, is made (T. B. Johnson, 1969). The main goal of this process is usually considered to be maximizing net present value (NPV) of the mine while respecting constraints imposed by technical and operational characteristics of the project. However, a wide range of objectives and constraints has been introduced in the literature. This area has attracted lots of attention from researchers in mining, operations research, and mathematics. Various models and techniques are developed to satisfy the need for having an efficient and feasible production schedule. Linear programming (LP), integer programming (IP), mixed integer linear programming (MILP), dynamic programming (DP), graph theories, heuristics and meta-heuristics are all used during decades of research on this topic.

Open pit mining is one of the most common ways of extracting valuable material from the ground. The operations deal with billions of dollars of cost and revenue. That makes a single percent reduction in costs or increase in revenues worth hundreds of millions. Consequently, operations research has been widely implemented for optimizing various aspects of the mining operation. Production scheduling is one of these areas and has attracted lots of attention from the researchers during the past decades. This production scheduling is usually categorized into three main groups based on the time horizon: long-term, mid-term and short-term. Long-term open pit production planning (LTOPP) is the process of making decision on the order of extraction of blocks, or other units of planning, while satisfying operational constraints over the mine life. When a block model is created using Geostatistical methodologies, it usually contains millions of blocks. Most of the values assigned to these blocks are uncertain due to scarcity of information. On the other hand, some of the constraints and operational characteristics can only be determined as the mining goes along. Therefore, mine planning is usually performed in a top-down fashion based on different time horizons. In the first step, the life of mine plan, also referred to as the long-term plan, is determined. This plan is then broken into mid-term plans which are usually on monthly basis. Short-term or operational planning is then undertaken by breaking the mid-term plan into weekly or daily schedules.

It is not possible and reasonable to take all the details into long-term plans. Therefore, more details are taken into account as shorter time horizons are considered. The decision in a long-term plan usually consists of the period of extraction of blocks and whether they should be processed as ore or dumped as waste. The most common constraints considered are the yearly mining and production capacities as well as head grades of processes and deleterious material. As shorter term plans are being developed, more information on estimated grades, equipment availability and utilization, market demands and prices etc. become available. On the other hand, short-term planning is done on a smaller block model restricted by the result of the long-term planning i.e. short-term planning of the first year is done only on blocks assigned to the first year through the long-term planning. Therefore, more details such as stockpiles, equipment selection, process selection, mining and re-handling issues are taken into account (Osanloo, Gholamnejad, & Karimi, 2008).

1.2. Statement of the Problem

The long-term open pit production planning problem is the problem of scheduling material extraction from an open pit mine on a yearly basis with respect to technical and operational constraints. Since it is defined over the whole mine life, LTOPP is also known as the strategic mine planning. The objective of LTOPP is usually to maximize net present value of the mine. However, other objectives such as having smooth production levels, minimizing deviations from desired head grades and maximizing the utilization of trucks and shovels are also considered in the literature. Moreover, risks and costs of uncertainties can be added to the objective function if the problem is going to be addressed as a stochastic problem. The decision on the destination of blocks can also be made along with the extraction period. This leads to the multi-destination LTOPP in which the common predetermined cut-off approach is not used. Instead, the decision on the destination of planning unit i.e. processes, waste dump or stockpiles is made for each individual planning unit based on its grade, economic value and available capacities in the period of extraction. The planning procedure and the difference between the fixed and dynamic cut-off approaches are illustrated in Figure 1.1.

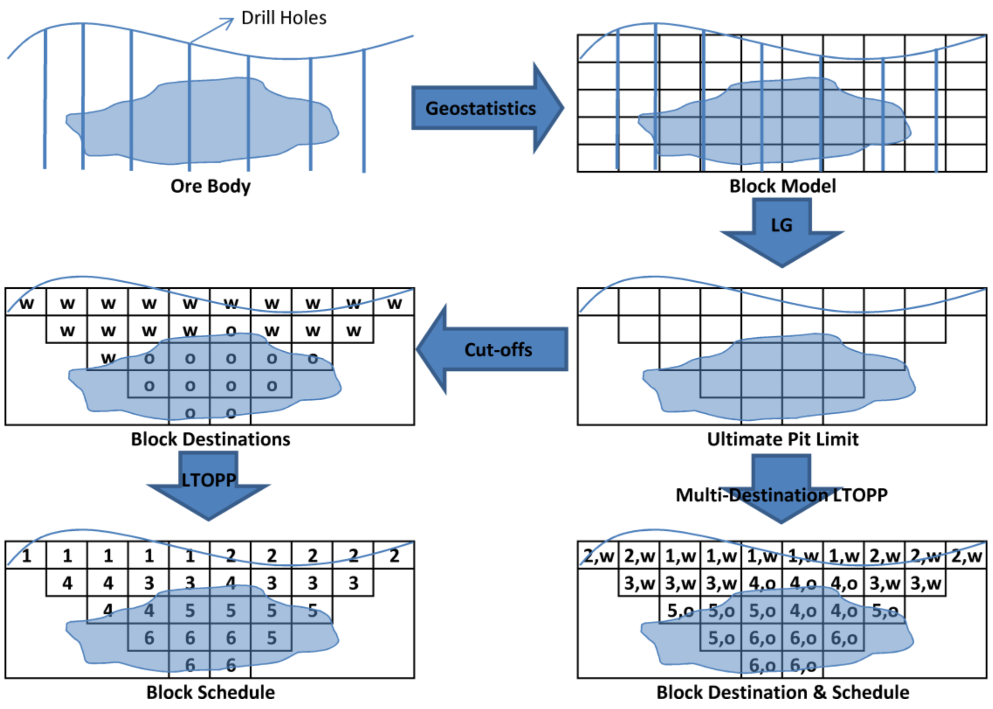


Figure 1.1.Dynamic vs. Fixed Cut-Off Approach

The problem addressed in this project is the multi-destination long-term open pit planning problem. Various mathematical models and solutions for LTOPP have been proposed in literature and are reviewed in the next chapter. However, there exists no globally recognized technique that can determine the optimum extraction sequence with reasonable computing time. In this study, we try to answer this research question:

> *Can a multi-destination long-term open pit production schedule which (1) results in near-optimal net present value, (2) is practical and realistic, (3) is not based upon predetermined cut-off grades, (4) includes stockpiling, be determined in a reasonable time by aggregating blocks into larger units, forming a mathematical model and solving with currently available computing resources and software?*

In this study, our goal is to develop a multi-step approach to determine the multi-destination long-term open pit production plan. The long-term production plan must:

- determine the best order of extraction of material from the final pit,
- determine destination of extracted material,

- be able to work based on different units of mining and production,
- maximize the net present value of the operation over the mine life,
- determine the best stockpiling strategy aligned with the mining plan to satisfy processing blend requirements and increase net present value of the operation,
- generate realistic and practical production schedule,
- not over- or under-estimate the value of the operation,
- be solved in reasonable CPU time.

Moreover, we need to develop an aggregation technique that:

- creates mining and processing units homogenous in rock type, grade and other characteristics,
- has control over the shape and size of the generated aggregates,
- can generate aggregates with respect to the direction of mining.

In this research, we are developing a mine planning technique based on the following assumptions:

- Our multi-step approach starts by taking the block model of the deposit as an input. The block model holds all the attributes for imaginary 3D cubes discretized over the deposit. Rock types, tonnages, grades and recoveries are some of block attributes determined using Geostatistical methodologies prior to production planning.
- Block values are calculated based on mine life, mining, processing and haulage costs as well as block attributes.
- Mine life, mining and processing capacities are deterministic and determined by running multiple scenarios. The optimization models do not directly determine these parameters.

- All block attributes, costs, prices and equipment availabilities are known and deterministic.
- The ultimate pit limit is determined using the well-known Lerchs and Grossmann algorithm prior to planning.
- Production pushbacks (mining phases) are determined prior to production planning.
- Processing units are contained within mining units (no overlaps). In case of creating clusters, the clusters are formed with respect to pushback boundaries.
- When mining a mining unit we have access to all processing units within that mining unit.

The next step is to form mining cuts based on block similarities and use them to make processing and stockpiling decisions. These cuts need to be homogenous in rock type and grade to provide reliable solutions. On the other hand, they have to have controlled shape and size since they are going to be marked for different destinations. We then propose a mathematical formulation for long-term open pit production planning that uses mining-panels and mining-cuts to make extraction, processing and stockpiling decisions. The process is illustrated in Figure 1.2.

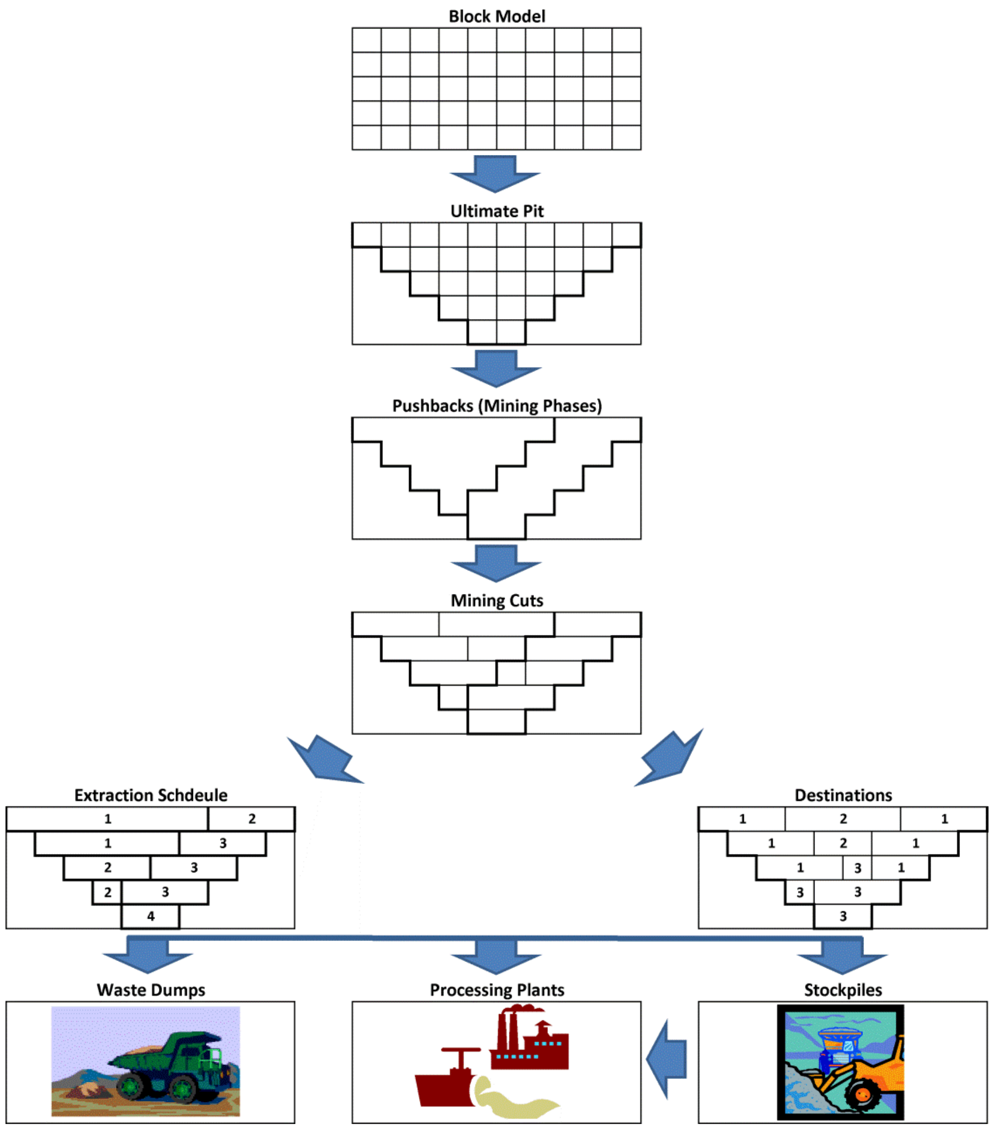


Figure 1.2. Multi-Step Production Scheduling Approach

1.3. Summary of the Literature Review

As we thoroughly discuss in Chapter 2 of the thesis, Operations Research (OR) has been used by scientists and problem solvers for centuries to improve operations, manage resources and find best possible ways of conducting tasks. However, the modern area of Operations Research was introduced in the World War II era to tackle with resource allocation problems in the military. Since then, various areas of engineering and science have been affected by the new OR techniques including the mining industry.

Johnson (1969) is believed to be the first one to bring OR into mining industry. He formulated a mathematical planning model to determine the optimum sequence of block extraction and proposed a decomposition approach to solve the model. However, the first proposed model by Johnson (1969) used linear variables to control the extraction sequence of blocks and failed to properly control the constraints. Hence, linear variables were replaced by integer ones to make sure a block is completely extracted before extraction of its successors start. Although this solves the partial extraction problem, adding integer variables makes the problem NP-Hard and extremely hard to solve for real-size problems. Therefore, researchers have focused on finding alternative methods to formulate the problem or to solve it with near-optimal solutions.

A common way of dealing with large problems is to aggregate the variables and solve the problem in aggregated level. Busnach et al. (1985) use zero-one model to maximize NPV and aggregate blocks into horizontal layers in order to make the problem tractable. Klingman and Phillips (1988) also aggregate blocks into horizontal layers in a phosphate mine before creating the production planning model. Pursuing the same goal, Gershon and Murphy (1989) create aggregated layers of blocks which are totally mined as ore or waste and solve the model using Dynamic Programming. Another noticeable study is proposed by Ramazan and Dimitrakopoulos (2004) and Ramazan (2007). The authors create sub-graphs of the block model using a so called fundamental tree algorithm based on blocks precedence and block value. Samanta et al. (2005) take the same layering approach as in Busnach et al. (1985) but solve the problem using Genetic algorithm. Similarly, Zhang (2006) combined Genetic algorithm with a block aggregation technique to reduce the number of variables in the model. Boland et al. (2009) propose a solution procedure based on aggregation and disaggregation but do not discuss the aggregation method to use. Weintraub et al. (2008) and Askari-Nasab et al. (2010) group blocks into larger planning units using k-means and fuzzy c-means clustering respectively. Epstein et al. (2012) use expansions of the open pit or underground mine as their aggregated units of planning.

Newman et al. (2010) summarizes the state of literature on open-pit mine planning as:

> *"Open-pit block sequencing is a heavily studied area, largely because of the many open-pit mines in existence today and the somewhat generic nature of the mines. Researchers are able to solve increasingly large models, which has enabled progress from solving an ultimate pit model and subsequently sequencing blocks within small nested pits to solving a monolith sequencing problem containing up to hundreds of thousands of blocks. However, work remains in terms of incorporating fidelity such as variable cutoff grades and inventory constraints into large models that produce optimal solutions in a reasonable amount of time."*

Various mathematical formulations and aggregation techniques are proposed by researchers to tackle the long-term production planning problem but share a few similar drawbacks.

1. Most of the proposed aggregation techniques do not account for block similarities when combining them into larger units. This can introduce material dilution and create discrepancies between the solutions provided and the real outputs of the operation.

2. In cases where the aggregation techniques consider similarities and similarities between blocks, the algorithm does not have control over the size and shape of the mining units. This can result in production schedules that are not practical from the mining operations point of view.

3. The common approach in open-pit mine planning is to use a fixed cut-off grade to determine the destination of material. It is required to have models that determine

the destination of material along with the schedule since fixed cut-off grade does not look into operational constraints and can result in over- or under-utilizing the plants.

4. Most of the models assume deterministic values for grades, revenues, costs and operational parameters.
5. Many of the proposed techniques do not look into practicality of the solutions they provide from operational point of view and result in under- or over-estimation of operation value.
6. Proposed mathematical models require intensive processing power and have long solution times where heuristics and meta-heuristics lack flexibility and optimality measures.

1.4. Objectives of the Study

The main goal of the research is to develop, verify and validate a mathematical model together with a solution technique for the multi-destination long-term open pit production planning problem. The objective of the mathematical model is to maximize the net present value of the mining operation with respect to operational constraints such as grade blending, average deleterious material levels, lower and upper bounds on mining and production levels, extraction precedence, slope constraints, etc. Moreover, we are trying to answer another important question in this thesis: what is the proper choice of mining and processing units. As pointed out by Johnson (1969), using blocks as units of planning will not result in a practical schedule and consequently, over-estimate the value of the operation. Therefore, we try to create a reasonable schedule by forming mining-cuts and mining-panels and using them in an MILP formulation for long-term open pit production planning. The multi-step approach proposed in this research focuses on:

- Maximizing the NPV of the operation while respecting technical and operational constraints and avoiding unpractical schedules,
- Providing best stockpiling strategy for long-term blending,
- Forming homogenous mining-cuts with controlled size and shape to be used in formulating LTOPP and other steps of mine planning.

1.5. Scope and Limitations of the Research

The objective of this research is to form a mixed integer linear programming formulation for long-term open pit production planning. In order to achieve this goal, we created a hierarchical clustering technique that can form mining-cuts homogenous in grade and rock type with minable shapes. These mining-cuts along with mining-panels are used to formulate the MILP. Therefore, this project involves two separate steps: forming mining-cuts and developing the MILP.

The mining-cuts are created using an agglomerative clustering technique with regard to the characteristics of the block model.

- All the attributes considered in the clustering algorithm have deterministic values and are predetermined using Geostatistical and Geological tools. Techniques implemented to create the block model and to estimate the attribute values are not within the scope of thesis.
- Mining-cuts are formed within the boundaries of the ultimate pit limit, and if required, the generated pushbacks. Although, two different methods for determining the UPL

and generating the pushbacks are compared, the details of the methods are not within the scope of this thesis.

- The clustering algorithm developed in the thesis is sensitive to the control parameters defined, and therefore, their impacts on the quality of the created mining-cuts are evaluated. However, the proposed algorithm is not capable of self-tuning and all the control parameters are manually tuned for each case.

We propose an MILP formulation that uses different mining and processing units to determine the optimum long-term production schedule for any open pit mining operation.

- The objective of the model is to maximize the net present value of the operation.
- It includes the profit made at plants by processing the valuable elements in the material sent directly from mine or reclaimed from stockpiles, the costs associated with removing deleterious material from ore, the costs of mining and hauling material from the pit to the plants, stockpiles or waste dumps, and the re-handling costs associated with reclaiming material from the stockpile and sending to the plant.
- Actual grade of elements of the stockpiles in each period are not variables in the model. They are replaced by fixed predetermined values assuming that material reclaimed from the stockpile has fixed predetermined grades.
- When modeling the relationship between the mining and processing units, it is assumed that the scheduler has access to all the processing units within each mining unit and does not need to follow the same ore-waste ratio as in the mining unit.
- All the prices and costs, availability of the equipment, element grades and contents in the mineralized material and recoveries are assumed known and fixed.
- Tactical and operational planning, shovel assignments and truck dispatching are not considered in this thesis.

1.6. Research Methodology

As mentioned earlier, this study tries to achieve two goals: creating homogenous minable mining-cuts and developing a mathematical model for finding the optimal solution to the LTOPP. To do so, a MATLAB® application with GUI is developed which can guide the user through various steps of preparing the dataset, running clustering algorithm, forming and solving the MILP, and interpreting and plotting results.

The first part of this study involved reviewing the literature on the clustering techniques and their implementations, operations research and mathematical programming and their implementations on open pit production planning problem. Afterwards, a number of different block models from different open pit mining operations were prepared for evaluating the performances of clustering techniques as well as the MILP formulations. The corresponding analysis can be found in Tabesh and Askari-Nasab (2011), Tabesh and Askari-Nasab (2013) and Tabesh et al. (2014). In order to present the capabilities of the proposed model and algorithm in this document, a small standard test dataset, called Marvin, is chosen to illustrate the details of the proposed approach and to study the effects of solving the MILP in different resolutions.

We used the Marvin block model from Whittle™ (GEOVIA, 2014b) to determine the ultimate pit limits and pushbacks. The GUI has forms to input each type of block model into MATLAB® and perform clustering on them. Then the user sets the clustering parameters and runs the chosen algorithm. In the next step, the MILP problem is set up using another GUI. Technical and operational constraints, mine life and interest rates are assigned next. Then the program forms the matrices required, does the preprocessing to reduce problem size and runs TOMLAB/CPLEX (Tomlab Optimization AB, 2011) to solve the problem to the specified tolerance gaps. Two types of tolerance gaps can be defined in TOMLAB/CPLEX: EPAGAP and EPGAP. These two can be used to define the optimality gap to accept a solution as the optimal solution and stop the mixed integer optimization. Another interface is designed to interpret the MILP results, plot required graphs and export results into a text format to import into other software packages.

After the programs and GUIs were developed in MATLAB®, two groups of experiments were taken to evaluate the performance of the clustering algorithms as well as mathematical formulations. The clustering algorithms and MILP formulation are evaluated on real-size datasets (Tabesh & Askari-Nasab, 2011, 2013; Tabesh, et al., 2014). However, it is not possible to solve a real-size LTOPP in block level. Therefore, in order to evaluate the effects of clustering on the MILP results, the Marvin dataset is clustered into mining-cuts and solved in four different resolutions that use:

- Mining-panels for extraction and mining-cuts for processing decisions
- Mining-cuts for extraction and processing decisions
- Mining-cuts for extraction and blocks for processing decisions
- Blocks for extraction and processing decisions

Three different clustering methods are developed: agglomerative hierarchical, agglomerative hierarchical with Tabu search and K-means. These methods are tested on Marvin block model and compared based on homogeneity of the generated clusters, cluster shapes and processing time of the algorithms. Since the agglomerative hierarchical clustering showed to result in better clusters we used this algorithm in further steps of the research. We studied the clustering parameters and implemented various scenarios to identify the effects of changing the clustering parameters on the mining-cuts and consequently MILP results. Afterwards, the model coded in MATLAB® is solved for various scenarios for various datasets.

1.7. Scientific Contribution and Industrial Significance of the Research

The goal of this research is to focus on the long-term production planning in open pit mines and forming mining-cuts using hierarchical clustering algorithm. This includes developing and programming clustering algorithms and forming mathematical formulation to solve the LTOPP. The scientific contributions and motivations behind this study can be summarized in the following:

- The main advantage of creating clusters is to represent the mining cut concept which provides more realistic schedules (T. B. Johnson, 1969) and increase practicality of the solutions (Osanloo, et al., 2008). Moreover, the homogeneity of the created mining-cuts is a concern pointed out by Osanloo et al. (2008) and is addressed in this project.

- The solution time of the LTOPP is significantly decreased by aggregating blocks into larger mining and processing units making it possible to evaluate multiple scenarios in a short time. In addition, decreasing the solution time can lead to incorporating uncertainty in LTOPP by using simulation-optimization techniques.

- Our mathematical formulation simultaneously decides on the extraction period and destination of material which corresponds to the dynamic cut-off grade concept and increases the NPV of the operation (Osanloo, et al., 2008). It is designed to handle multiple elements at the same time and decides on the destination of material by evaluating all the grades and the profits made at each destination.

- The developed formulation not only provides the long-term extraction and production schedule of the operation, but also develops the optimum stockpiling strategy to increase the NPV of the operation and provide an acceptable blending to the processing plants.

Mathematical programming models for LTOPP are used since 1960s, and since then, hundreds of research projects has been undertaken to improve the models and provide solution procedures. Despite all these effort no globally recognized technique is in place which can satisfy all the needs of this subject matter. This study focuses on two stages of the LTOPP. The first part of this project is an area which is not investigated as widely as it deserves: creating mining-cuts (a.k.a. aggregates or clusters of blocks). Mining-cuts are usually determined by hand which needs hours of work by a well trained expert and causes significant reduction in the practicality and profitability of the generated schedule. Forming the mining cuts with reasonable time and resource usage and with consideration of geological and technical properties, directions and other mining constraints is a main industrial contribution of this research project. Moreover, shape control procedure included in the clustering algorithm results in clusters with minable shape and controlled size. This can contribute significantly to the mine planning software industry where the mining cuts are considered given; except than a few cases where semi-automatic procedures are implemented (e.g. MineSight ® 3D Advanced CAD (Mintec, 2014)).

In the next step the generated cuts are used for developing mathematical models of LTOPP. The proposed mathematical formulation uses aggregated units for making mining and processing decisions. Therefore, the time and resource required to solve the model is less than other common formulations. This makes mine planners able to benefit from the accuracy and flexibility of mathematical programming without having to spend hours to get the production schedules.

1.8. Organization of Thesis

Chapter 1 of this thesis is an introduction into the research project. It explains the motives and objectives behind this study and illuminates on the boundaries of the project. Moreover, a brief summary of literature review and an introduction into the research methodology are included in this chapter.

Chapter 2 reviews the literature related to this project. The review starts by looking into definitions of operations research and continues by reviewing the applications of operations research in the mining industry. It reviews the relevant clustering literature and then focuses on the long-term open pit production planning problem. The chapter concludes by having a glimpse at other areas of research in mining that are attracting operations researchers.

Chapter 3 is the theoretical framework of the study. The chapter includes the details of the clustering algorithms and how they are implemented. The mathematical formulation for LTOPP is also explained in this chapter. The details of programming and using the GUI are provided in Appendix I.

Chapter 4 is the implementation of the clustering algorithm and the MILP formulation. A small test dataset is used in this chapter to formulate the model and study the effects of clustering on the MILP results. Moreover, we used the test dataset to illustrate how the clustering and MILP work together and how we can tune the parameters. The test dataset is a small standard block model, called Marvin, and is taken from the MineLib block model library (Espinoza, Goycoolea, Moreno, & Newman, 2013). It is chosen because it is small enough to be solved in block level and also because it is the demo dataset in Whittle™ (GEOVIA, 2014b) mine planning software as it enables us to compare our planning results to commercial software.

Chapter 5 is the concluding chapter that provides the conclusions of this research project and directions for future work. It restates the goals achieved in this study and elaborates on the strengths and limitations of the proposed approach.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1. Operations Research (OR)

It is commonly believed that Operations Research was born during World War II in the U.S. air-force base in England. However, Gass and Assad (2005) define OR as the "Mathematical or scientific analysis of systematic efficiency and performance of manpower, equipment, machinery and policies used in a governmental, military or commercial operation". Therefore, they claim that the first known instance of OR is when "Joseph aided Pharaohs and Egyptians live through seven fat years followed by seven lean years by the application of lean-year programming". Following the same approach, Murty (1995) defines Operations Research as "the branch of science dealing with techniques for optimizing the performance of systems"; and concludes that ancient cave men, finding the shortest route from their cave to river based on try and error, were the first to implement OR. Accordingly, all the scientific endeavors to analyze and improve human and machine efficiencies can be considered as parts of Operations Research: from Girlamo Cardano's and Blaise Pascal's work on gambling and chances in 16th century to the very modern computer-based intelligent optimization algorithms. Among these some may look more similar to what we today call Operations Research: the graph-based solution to Konigsberg bridge problem by Leonhard Euler (1707-1783), Lagrange multipliers by Joseph-Louis de Lagrange (1736-1813), Pareto optimal solution by Vilfredo Pareto (1848-1923), Time Studies by Fredrick W. Taylor (1856-1915), Markov process and chain by Andrei Markov (1856-1922), representation of convex polyhedra by Hermann Minkowski (1864-1909), etc. The interested readers can refer to Gass and Assad (2005) for a comprehensive annotated timeline of Operations Research.

Despite all the aforementioned scientific work on Operations Research, the first organized effort to optimize the performance of a system using OR techniques, is found in the Bawdsey Manor Research Station in 1936. Bawdsey was established by British Air Ministry to study how the newly developed radar system can be used to increase the efficiency of the British air defense system. A. P. Rowe, the head of the Bawdsey station in 1938, is believed to be the first to use the term “Operational Research” (Gass & Assad, 2005). The same term later adapted by American scientists as “Operations Research”. In October 1942, a group of scientists, led by John M. Harlan, traveled from U.S. to England to form the Operations Research Section in the Air Force’s Eighth Bomber Command. This group of scientists was asked to work on improving the bombing accuracy of the Air Force bombers and managed to increase the number of bombs on target by 1000 percent (Gass & Assad, 2005). Nonetheless, the aforementioned OR groups solved their problems by quantitative studies of the past data instead of modeling the problem as a mathematical formulation and solving the model for the best solution.

In 1947, the most influential step in OR, the project SCOOP (Scientific Computation of Optimal Programs) was established. This research group was formed in the U.S. Air Force and led by economist Marshall K. Wood. However, George B. Dantzig, the most recognized name in Operations Research, was the chief mathematician of the group. Project SCOOP was where linear programming was born and many famous problem structures and solution procedures were introduced. We will take a brief look at some early problem structures and solution procedures in the next section.

2.1.1 *Mathematical Programming*

The first known representation of reality in a mathematical model, which awarded Wassily Leontief a Noble prize, was formulating the inputs and outputs of an economic system in a table (matrix) of coefficients in 1936 (Leontief, n.d.). Another well-known instance of early mathematical formulations is the Traveling Salesman Problem (TSP) brought to attention by Merrill M. Flood in 1937 (G. Dantzig, Fulkerson, & Johnson, 1954). Although TSP was studied in 19th century as a graph theory problem, its representation in mathematical formulation with variables and objectives is first known to be published in Dantzig et al. (1954). However, the authors acknowledged that the TSP was promoted by M. Flood. Afterwards, Flood (1956) offered a solution procedure that can solve the problem to optimality and implemented it on the same instance of TSP as of Dantzig et al. (1954). The problem instance consisted of a tour of the lower United States and the District of Columbia with 48 nodes. While Dantzig et al. (1954) did not claim that their solution was optimal, Flood (1956) provided a better solution and proved its optimality. TSP and its variations are still among the most discussed problems (Bao & Liu, 2012; Bontoux, Artigues, & Feillet, 2010; Letchford, Nasiri, & Theis, 2013; Nagata & Soler, 2012).

In the same era, the Personnel Assignment Problem was studied widely because it was simple and possible to be generalized to various situations. This problem was widely studied by various researchers in the first half of the 20th century. Nonetheless, Kuhn (1955) solved the problem to optimality with an interesting approach called the "Hungarian" method, which is still taught in OR courses and textbooks. The diet problem was another interesting problem definition posed and solved by, Noble prizewinner, George Stigler in 1943. The diet problem is to determine the optimum combination of food that can provide required nutrition for an average person with minimum cost. Later in 1947, George Dantzig solved the problem to optimality and showed that Stigler's solution was very close to optimality (Gass & Assad, 2005).

In the summer of 1947, Linear Programming (LP) as a unified mathematical framework along with its solution procedure, the Simplex algorithm, took OR a huge step forward. Introduction of LP, coincided with development of electronic digital computers, helped researchers from various areas form mathematical models and solve them easily. Furthermore, it offered researchers a common language to share their problem structures and solution procedures (G. B. Dantzig, 1963). Although various attempts to propose a unified mathematical modeling format were previously made by researchers, especially by Russian mathematician Leonid V. Kantorovich who suggested the term "Linear Programming", Dantizg's definition and solution procedure attracted global attention. However, Leonid V. Kantorovich and Tjalling C. Koopmans won the 1975 Economics Noble prize for their work on optimal allocation of resources and Dantzig was left out. Since Kantorovich and Koopmans were friends with Dantzig and worked together for a while, they explicitly expressed their regret about the decision of selection committee (Gass & Assad, 2005).

10 years after linear programming was introduced, Dantzig (1957) promoted the idea of modeling and solving linear problems with discrete solution space: Integer Programming (IP). Dantzig (1957) modeled and solved a number of discrete problems. However, Dantzig (1963) refers to Gomory (1958) as the founder of the Integer Programming theory. Among the problems introduced by Dantzig (1957) is the Knapsack Problem (KP). KP is one of the most famous OR problems and is similar to the diet problem but with binary

variables. The Knapsack Problem is to decide which items a hiker should carry in his/her knapsack to maximize the total value in his/her knapsack while not exceeding a certain weight. Dantzig (1957) proposed the problem and solved it using a graphical approach. We will discuss KP and its variations in Section 3 in more details since the problem addressed in this project can be considered an extension of KP.

Although Dantzig'a simplex method, along with branch and bound and Gomory's cutting planes, enabled researchers model and solve various real-life problems, the complexity and size of the optimization problems as well as limitations in computational power forced OR specialists to look into problem specific heuristics and meta-heuristics that do not guarantee optimality but help providing a "good enough" solution.

2.1.2 *Heuristics and Meta-Heuristics*

The word heuristic is usually used to describe a solution method that is based on "intuitive and plausible arguments" but not guaranteed to find the optimal solution (Murty, 1995). In this sense, it is not an easy job to argue about the start of heuristic algorithms. However, among OR specialists, the word heuristic refers to the "methods for intelligent search" (Murty, 1995). On the other hand, Gass and Assad (2005) define heuristics as the ways to "program a computer to be a thinking machine". Therefore, heuristic algorithms we review in this section are the ones born after computers were born. While heuristics are plausible rules for searching the solution space and finding good solutions, "meta-heuristics are high level procedures that coordinate simple heuristics, such as local search, to find solutions that are of better quality than those found by the simple heuristic alone" (Resende, de Sousa, & Viana, 2004).

As mentioned earlier some linear programming problems are based on discrete variables, such as the assignment problem, the traveling salesperson problem and the knapsack problem. Dantzig (1957) calls these problem “Discrete-Variable Extremum Problems”. However, in today’s literature these problems are usually referred to as “Combinatorial Optimization” problems (Rayward-Smith, Osman, Reeves, & Smith, 1996). The nature of these problems makes it hard for linear programming solvers to find optimal or sometimes even feasible solution to them, especially when the size grows. Therefore, groups of heuristic and meta-heuristic algorithms are developed to deal with such problems. Here we review some of the most famous ones that have previously been implemented on mine planning problems: Genetic Algorithms (GA), Simulated Annealing (SA), Tabu Search (TS) and Ant Colony Optimization (ACO). We will briefly mention the origins of these algorithms and provide useful resources in this section. Their implementations on mining problems are covered in the upcoming section.

Holland (1975), inspired by the evolution of biological systems and the way genes interact to form more evolved species, introduced a group of algorithms called Genetic Algorithms (GA) (Rayward-Smith, et al., 1996). Genetic Algorithms are chance-based search algorithms that use a population of solutions (chromosomes) and use random mutation and mating operators to sweep the solution space and provide better (more evolved) solutions. Since 1975, these algorithms have been and still being widely implemented on various optimization problems. However, the genetic operators and selection procedures are modified by every author based on their problem structure. Moreover, there are valuable instances that combine GA with other heuristics, such as local search, to obtain better solutions. Investigating the numerous applications of GA is beyond the scope of this document but interested readers can refer to Haupt and Haupt (2004) and Resende et al. (2004) for applications of Genetic Algorithms and more details on how they work.

Another famous meta-heuristic algorithm, Simulated Annealing (SA), was introduced by Kirkpatrick et al. (1983). SA is developed based on a computational algorithm by Metropolis et al. (1953) designed to simulate the cooling of material in a heat bath by randomly changing the state and calculating the energy level (Rayward-Smith, et al., 1996). SA, on the other hand, searches the solution space similar to a local search heuristic but with the ability to avoid local optima. A simple local search heuristic gets an initial solution and searches the solution neighborhood for better solutions (e.g. smaller value for minimization problems), until reaches a point that no better neighbor solution is found. This point, which might be different from the optimal solution, is called a “local optima”. However, SA uses a control parameter, temperature, to determine the possibility of moving to a worse solution (e.g. larger value for minimization problems), if the algorithm is in early stages. The temperature drops as the algorithm iterates and lets the algorithm stop at the best solution found. Since its introduction in 1983, SA has been widely implemented on optimization problems because it is simple to implement and provides good solutions in reasonable time. Dowsland and Thompson (2012) explores theories and practices of SA and reviews some of its applications.

Tabu Search (TS) is another local search based meta-heuristic that tries to avoid local optima by preventing the search from exploring an already visited region or structurally unwanted solutions. TS was first introduced by Glover (1986) and has become popular because of its simplicity (Gass & Assad, 2005). However, the performance of the algorithm significantly relies on the initial solution and how the Tabu list is created and updated. TS is a deterministic search method opposed to SA and GA which are chance-based search methods (Rayward-Smith, et al., 1996). Therefore, it is usually combined with other heuristic techniques to cover larger regions of the solution space. Some recent examples can be found in: TS and SA (Mousavi & Tavakkoli-Moghaddam, 2013), TS and GA (Garai & Chaudhurii, 2013) and TS and ACO (Lin, Yeh, & Huang, 2013). Glover and Laguna (1997) investigate the details of the algorithm and some modifications proposed by other authors to improve its performance.

Ant Colony Optimization (ACO) is another chance-based meta-heuristic algorithm inspired by the nature. It was first developed by Dorigo (1992) in his PhD thesis. It was then improved by various authors and implemented on problems from various disciplines. Dorigo and Stützle (2004) provides a comprehensive resource on ACO, its variations and implementations. ACO is based on how real ants communicate when they find a food source: they leave a trace, pheromone, on their path. Borrowing the this idea from real ants, Dorigo (1992) defined search agents that explore the solution space and leave pheromone on the paths they take. The more frequent a path is taken the more pheromone is left on the path, and consequently, the probability that other ants take the same path is increased. Therefore, neighborhoods with higher objective values are more thoroughly explored.

2.2. Aggregation and Disaggregation in Operations Research

Aggregation techniques are used to combine data, reduce the size of the problem and analyze the effects of simplifying large scale optimization problems (Litvinchev & Tsurkov, 2003). In this section, we are going to review a number of clustering techniques used to combine data.

Clustering is defined as the process of grouping similar entities together in a way that maximum intra-cluster similarity and inter-cluster dissimilarity is achieved. This can be modeled and solved as a mathematical programming problem; however, the difficulty lies in the amount resources and time required to solve the problem. The clustering problem is proven to be NP-Hard (Gonzalez, 1982). Therefore, a wide range of non-exact algorithms has been developed in the literature. These algorithms work based on defining a measure of similarity or dissimilarity between the objects. These techniques can be categorized into two major groups: hierarchical and partitional clustering. As its name implies, hierarchical clustering is performed by creating a hierarchy of clusters. On the other hand, partitional clustering is performed by partitioning data objects into a number of groups. Hierarchical clustering is known to result in better clusters comparing to partitional algorithms but by taking more CPU time (Feng, et al., 2010).

Hierarchical clustering algorithms are divided into two main groups: agglomerative vs. divisive clustering. In the former group, clusters are formed from merging smaller ones. This means, at the beginning, the number of clusters is the same as the number of objects. As the algorithm goes on, more similar clusters are merged together until the stopping criterion is met or all the objects fall into the same cluster. The procedure is the other way round for the divisive clustering techniques; i.e. all the objects are considered to belong to the same cluster in the beginning of the algorithm, and clusters are divided into two in each step of the algorithm. Finding the target cluster to split and the way they are divided is a tricky step. Therefore, this method of clustering has not attracted as much attention as the agglomerative one.

A famous example of partitional algorithms is the k-means, which attempts to find cluster means and assign data points to the closest mean. Finding the exact optimal solution to this problem is also proven to be NP-Hard (Mahajan, Nimbhorkar, & Varadarajan, 2010), but heuristics can be used to find good partitions on data. Another shortcoming of the k-means algorithm is dealing with problems in which not all the attributes of data points are numerical. Therefore, modifications have been done on k-means in order to incorporate non-numerical data properties (He, Xu, & Deng, 2008).

Clustering techniques can also be categorized based on their usage: specific purpose vs. general-purpose algorithms. The main difference between these two sets is in the objective function. General purpose clustering algorithms deal with a set of attributed objects and try to create a number of clusters in a way that they maximize pre defined intra-cluster similarity or inter-cluster dissimilarity while the other type of algorithms try to create clusters aligned with the objective function of the problem. A famous and widely studied example in the second group is the cell formation problem (S. Johnson, 1967; Vakharia & Wemmerlöv, 1995).

2.3. Operations Research in Mining Industry

Mining industry is a vast area that consists of various operations such as exploration, planning, extraction, transportation and processing. All these operations share two common features: they all need huge investments and they all deal with large amounts of material. An average mining operation requires millions of dollars in capital investments and has millions of dollars in annual cashflow. Therefore, a few percent improvements in every step of the operation lead to hundreds of thousands of dollars in costs and profits. On the other hand, mining companies are facing lower quality and scarce deposits since most of the good available deposits are already extracted. In this sense, optimization can play a major rule in success and failure of a mining operation. As mentioned earlier, Operations Research is the science of systematically studying and improving the efficiencies and performances of operations, equipment and other resources. Therefore, OR techniques have been widely used, in various stages of mining operations, to improve and optimize the plans and techniques as well as the performances of equipment and human resources. In this section, we review the literature on open pit and underground long- to short-term production planning. We also name a few emerging areas and valuable references that readers can use to find out about the latest trends in the applications of operations research in mining industry.

2.3.1 *Open Pit Production Planning*

Long-term open pit production planning (LTOPP) is commonly defined as the process of deciding on the sequence of extracting blocks from the mine in order to gain the highest NPV subject to a set of constraints. Two stages are usually defined in the open pit mine planning process. The first stage is to decide on the ultimate pit limit. In the second stage the order of extraction of blocks falling inside the ultimate pit is determined. There are techniques in which the two stages are combined together seeking higher profitability. However, Caccetta and Hill (2003) prove that, in many cases, trying to simultaneously find the ultimate pit limit and the production schedule does not increase the chance for having higher NPV; because the final pit found based on the schedule will fall inside the optimal pit found without considering the order of extraction. Two famous techniques are used for determination of the ultimate pit limit (UPL). The floating cone method developed in 1961 by Kennecott Copper (Kim, Cai, & Meyer, 1988) and the graph theory based approach by

Lerchs and Grossmann (1965) which is known as the LG algorithm. Other techniques are also proposed which are described and compared in Hochbaum and Chen (2000).

The implementation of mathematical programming on the mine production scheduling backs to 1960s but it was not the dominant way of planning for the next 2 decades because of the computational complexities in solving linear programming models. The earliest well known mathematical model is proposed by Johnson (1969). Johnson (1969) defines variables to decide on the extraction of the block in each period and to determine its destination. The model considers all the possible alternatives for the block if it contains more than one material type. Consequently, Johnson (1969) introduces dynamic cut-off concept by making simultaneous decision on the period of extraction and the destination of the material. It also considers capacity constraints on the required resources for mining, processing, etc. Bounds on the produced valuable and deleterious materials are also taken into account. The concept used for modeling slope and precedence constraints in Johnson (1969) is the one still in use by various authors. However, Johnson (1969) used continuous variables instead of integer ones. Johnson (1969) offers to solve the proposed model by the means of Dantzig-Wolfe decomposition technique. The author decomposes the main model into T sub-problems and a master problem which is a multi period planning problem and satisfies the given constraints. Sub-problems develop plans for each period based on block values determined by the master problem using optimum pit limit technique.

Since the model proposed by Johnson (1969) results in a very large and intractable mathematical problem, other approaches are tested by researchers. The traditional approach was to define production levels (in tonnages) of ore and waste. Afterwards, Gershon (1983) used two different mathematical models: LP and IP. Gershon (1983) proposed an IP model with zero-one variables indicating whether a block is going to be mined in a period or not. On the other hand, he proposed an LP with continuous variables that represent the portion of block extracted in each period and studied the differences. The problem with the zero-one variables lies in the computational complexities where the problem with continuous variables is not meeting the slope and access constraints (Busnach, et al., 1985).

Attempts have been made to overcome the curse of dimensionality of the long-term production scheduling combinatorial optimisation problems. Busnach et al. (1985) use zero-one model for maximizing NPV and aggregate blocks into horizontal layers in order to make the problem tractable. Their model also decides on the method of mining for each land unit (shallow vs. deep mining). However, the model proposed is non-linear which makes it difficult to solve. Another weakness of the model is that it does not consider precedence constraints. Klingman and Phillips (1988) also aggregate blocks into horizontal layers in a phosphate mine before creating the production planning model. Their model decides on mining and processing of each stratum simultaneously while respecting precedence relationships. They also take mining and processing constraints into account. However, variables defined in Klingman and Phillips (1988) do not take time ranges into account and only decide on the extraction and processing of strata in the mine life. The authors also decide to use the total produced material as their objective function.

Pursuing the same goal, Gershon and Murphy (1989) create aggregated layers of blocks which are totally mined as ore or waste and solve the model by the means of DP. The quality of ore in their case is a determined to be a function of depth. Therefore, Gershon and Murphy (1989) use DP to decide to which depth extraction should continue. They also decide on the destination of the extracted material: ore or waste. The objective function in their model is the total present and future return from the mine. However, their model does not actually construct a mining schedule but tries to find the pit limit while deciding on the cut-off grade. Additionally, the model is very simple and does not consider operation constraints and cannot handle multiple elements either. DP is used in Elevli (1995) too, where the authors present simultaneous problems of finding the final pit limit and the extraction sequence as a single dynamic programming problem. In order to avoid enumerating all the possible states, some probabilities of occurrences are assigned to each state. These values are not known at the beginning but are estimated using machine learning techniques as the algorithm goes on.

Another work based on DP can be found in Tolwinski and Underwood (1996) where the authors combine a stochastic search method with DP in order to avoid enumerating all the possible states. Tolwinski and Underwood (1996) consider each possible pit as a state of the problem and the decision on extracting another block as the transition between the states. They use the depth of mining in each column of the projection grid of the block model on the XY surface as the decision variable. The algorithm is then implemented on a very small block model that does not provide evidence that it can be used for real size models. The processing and head grade constraints are also missing in the DP procedure proposed in Tolwinski and Underwood (1996).

Caccetta and Hill (2003), going back to using integer programming, change the definition of the binary decision variables from “being extracted in period t” to “being extracted by period t” and propose a problem specific branch and cut algorithm for solving the problem. Their model considers different ore types, maximum vertical depth, minimum pit bottom width and stockpiles. For the solution procedure, the authors try to benefit from the structure of the problem in performing the branch and cut technique. Precedence constraints are strongly used for branching while knapsack capacity constraints are added to the model as cutting planes. In addition, since they prove that the optimal pit falls inside the UPL found by LG algorithm, all blocks falling outside the UPL are eliminated from the model. However, authors state that their algorithm generates tight bounds but is unable to find the optimum solution especially as the size grows.

Another noticeable study is proposed by Ramazan and Dimitrakopoulos (2004) and Ramazan (2007). The author creates sub-graphs of the block model using a so called fundamental tree algorithm. These sub-graphs are created using LP models and the production scheduling problem is solved according to these trees instead of the blocks (nodes in the tree) themselves. Each fundamental tree (FT) is a set of blocks which can be extracted without violating the precedence constraints and has a positive value. The third characteristic of FTs is that they cannot be decomposed into smaller FTs. Although solving the mine planning problem with FTs can reduce the number of zero-one variables in the model, it is not applicable to large deposits because of the large number of trees required. On the other hand, the complexity of this technique has reduced its popularity (Osanloo, et al., 2008). Another drawback of this method is the averaging of the mining costs of the blocks of an FT which reduces the accuracy of the calculations.

While Ramazan (2007) proposes a method for aggregating blocks for making mining and processing decisions, Boland et al. (2009) propose a solution procedure based on aggregation and disaggregation. In their work, excavation decisions are made based on some given aggregates through an MIP. Afterwards, disaggregation techniques are proposed to enable the model make processing decisions in block level by introducing the concept of bins. The main goal is to solve the MIP in aggregate level and to use an LP to model the processing stage with higher resolution. Two major improvements over other techniques are claimed in Boland et al. (2009). The first one is making excavation and processing decisions in different spatial resolutions which significantly contribute to the efficiency of the solution procedure. The second one is the dynamic cut-off approach in which the decision on the cut-off grade and block extraction sequence are made at the same time by the optimizer. The modeling and solution approach introduced in Boland et al. (2009) is implemented on two block models with 8,513 and 96,821 blocks. Various disaggregation techniques are implemented and compared for the resulted NPV and the CPU time. However, they do not propose any aggregation technique and do not discuss how different aggregation techniques can affect the solution quality.

Meta-heuristics are also used in mine planning. Kumral and Dowd (2005) propose a simulated annealing approach toward mine planning. The authors consider minimization of three objective functions: deviation from the desired tonnage, material content and the fluctuations in production. The multi-objective mathematical model is then formed with regard to precedence constraints. An initial feasible solution is found using Lagrangean relaxation and improved using simulated annealing. In each step of perturbation, a random number of blocks are moved to another neighbour period, either before or after their current period, based on the precedence relationships. The algorithm is tested on a very small dataset and its performance is shown but not compared to any other technique.

Samanta et al. (2005) also mention the complexity of the open pit mine production planning problem. They take the same layering approach as in Busnach et al. (1985) but solve the problem using a genetic algorithm and with a different objective function. Samanta et al. (2005) try to minimize deviation of monthly average grade from the specified values for two elements. Another significant aspect of their work is that the algorithm provides five best schedules that the planner can select from. However, the procedure works only on single bench mines and fails to address deeper ones because of the precedence relationships. The authors tested the algorithm on a very small block model with 98 blocks over 24 months which cannot provide any information on its efficiency on large models. Genetic algorithm is also used by Zhang (2006) combined by a block aggregation technique to reduce the number of variables in the model.

Sattarvand (2009) and Sattarvand and Niemann-Delius (2013) use ant colony optimization to solve the scheduling problem. The authors use a 2-dimensional test block model and determine the depth of extraction in each column in order to maximize the net present value of the mine. Their test block model consists of 1000 blocks on a cross sectional plane. However, they do not implement their algorithm on a real-size block model. They only mention the memory required for running the algorithm and do not predict the processing-time required for getting a good solution to the planning problem.

Sayadi et al. (2011) solve the ultimate pit limits problem using artificial neural networks. They focus on the pit limit optimization instead of the planning problem and add impurity constraint to problem compared to the original pit limit problem. The authors compare their result against LG algorithm via a case study of a phosphate mine. They provide almost the same pit limit as LG when the only set of constraints is the slope constraint. However, when adding impurity constraints they obtain higher profits than LG algorithm.

For readers interested in meta-heuristics, Sattarvand and Niemann-Delius (2008) review and provide a perspective to the possible uses of meta-heuristics in open pit mine planning. The common disadvantage of meta-heuristic algorithms is that there is no measure of optimality for the acquired solution.

Bley et al. (2010) is one of the most recent articles on mine production planning. The authors of this work try to reduce the number of decision variables based on precedence and production constraints. In their variable elimination procedure, if a block is not possible to be extracted at a specified period – because the summation of an attribute in the cone above it exceeds the cumulative allowed amount of that attribute – the variable is set to zero. The study implements and compares various techniques for single-block, multiple-block and block conflict scenarios and concludes the first two techniques have contributed significantly to the CPU time required by the solver. They also add some constraints for combination of blocks which their cones violate production capacities if extracted at the same period. These are cutting planes reducing the size of the feasible area. The important point is that these constraints make the LP relaxation of the problem stronger. This means less iteration of branching are required to converge to the optimum solution. In addition, stronger LP relaxation means having stronger bounds for the optimum result when comparison is required.

Bienstock and Zuckerberg (2010) focus on solving the LP-Relaxation of the mine planning problem by benefiting from the structural properties of the problem. The authors propose a hybrid technique based on Lagrangean relaxation and column generation that can find optimal continuous solutions to large instances of the precedence constrained production scheduling problem; and consequently, the open pit production planning problem. They implement their algorithm on instances with up to 178,000 blocks and provide the LP-Relaxation solutions in less than half an hour. However, Bienstock and Zuckerberg (2010) try to provide theoretical means for calculating LP-Relaxations and upper bounds. Therefore, Bienstock and Zuckerberg (2010) do not deal with the practicality and quality of the provided solutions.

Pursuing the same goal as Bienstock and Zuckerberg (2010), Chicoisne et al. (2012) use decomposition technique to solve the LP-relaxation of the open pit production planning problem. However, the authors use heuristics to find an integer solution to the problem based on the relaxed solution they found. They manage to solve large instances of the problem in reasonable time using their proposed technique. Nevertheless, they solve the simplified version of the problem with only one or two fixed capacity constraints in each period. They also mention that solving the problem in block level may not result in practical solutions from the mining operation point of view.

Souza et al. (2010) form and solve the mine planning problem with a different approach. Their work is focused on the operational (short-term) plan of a mine based on production levels. The objective of their model is to minimize the number of trucks and shovels used for extracting and hauling minerals while meeting the required production levels. Their proposed model also covers the multi-pit operations where multiple pits serve one processing plant. Souza et al. (2010) combine two known meta-heuristic algorithms to solve their model. The first heuristic, Greedy Randomized Adaptive Search Procedures (GRASP), is used to build up the initial solution. Then the solution is improved using another algorithm called General Variable Neighborhood Search (GVNS). They compare their solutions against the ones obtained from the CPLEX solver and report that most of the solutions fall within 1% gap of the optimal solution. However, their approach is to determine the production rate from each pit and does not provide the production schedule of the blocks.

Another recent study is undertaken in Cullenbine et al. (2011). The authors use a sliding time window approach to decompose the long-term open pit production planning problem into smaller models. In their approach, the planning problem is formed and solved on a limited number of periods, for example 5 years. Then the time window is moved forward for one year and another planning problem is formulated and solved. Cullenbine et al. (2011) run their proposed algorithm on sample datasets with 10,819 to 25,620 blocks over 15 periods. The smallest problem is solved in less than half an hour where solving the model with 25,620 blocks takes up to 3 hours. The authors mention that their algorithm is unable to develop a mine plan for a model with 53,668 blocks, which is not a large block model. Another aspect of their work is that they solve the problem in block resolution. As mentioned by Johnson (1969), this does not lead to a practical extraction plan. Moreover, the model proposed in Cullenbine et al. (2011) is based on predetermined cut-off grade.

Epstein et al. (2012) propose a mathematical model for optimizing the production schedule from open pit and underground mining operations at the same time. Their model is based on a general capacitated multi-commodity network flow and models rocks flow through various stages of mining and processing. They consider equipment capacities in each stage such as drilling, blasting, loading etc. as the constraints on the flow. Accordingly, Epstein et al. (2012) look at the mining operation as a chain of processes with limited capacities and try to determine the optimum flow of material through the network. The authors use bench-phases as their unit of planning in open pit operations to make the problem tractable.

They also introduce a set of network flow constraints based on the expansions (phases) of the pit to strengthen the formulation and obtain tighter LP-Relaxation. Epstein et al. (2012) implement their model on Coldeco Copper mine, with around 3 billion tonnes of material to be mined, and report 8% improvement in the NPV compared to the current production plan. However, their model is solved using rounding heuristics and the gap between the obtained solution and the optimal solution is not reported.

In order to compare the mathematical formulation with predetermined and dynamic cut-off grades, Kumral (2012) uses two different MILP formulation. The first model is based fixed destination for blocks opposed to the second model that uses one decision variable for each destination. The author builds the models based on a gold deposit with 7,020 blocks and schedules the blocks over 5 periods. The results show 5% improvement when using dynamic cut-off grade. Kumral (2012) focuses on comparing the fixed and dynamic cut-off approaches and does not propose any solution procedures or modeling tweaks that make large-scale block models solvable.

For a more complete review on the mine planning literature Osanloo et al. (2008) and Newman et al. (2010) can be useful. Fytas et al. (1993) can also be used if a review on older models and techniques is needed.

In conclusion, there is no globally recognized mine planning procedure which can determine the optimum mine plan with reasonable time and resource usage. In addition, most the work done do not consider the possibility of multiple destinations or the dynamic cut-off concept. The few which consider multiple destinations or dynamic cut-off grades are restricted to very small problems. There seems to be a need for more work on finding more efficient models and solution procedures. Moreover, smarter algorithms for creating mining cuts, in order to reduce the size of the problem and to increase the practicality of the generated schedule, seem to be necessary.

2.3.2 *Tactical and Operational Production Planning*

After developing the LTOPP, the plan has to be broken down to operational plans. The annual productions have to be studied in more details and with higher resolutions to develop the mid- and short-term production plans. Mid-term plans are usually monthly plans, which are then broken into short-term (weekly or shift-by-shift) production plans. Since more exploration data becomes available as the operation goes on, the production schedules become more accurate and in higher resolutions. On the other hand, instead of maximizing the NPV, operational production plans try to optimize the utilization of the available fleet and to make sure enough ore with acceptable quality is delivered to the processing plant. These objectives are pursued by scheduling material extraction, allocating trucks and shovels to mining faces and determining the optimal dispatching technique. Simulation and queuing theory are widely used in this step but our focus is on reviewing the optimization models for determining the operational production plans. Reader can refer to Newman et al. (2010) for a review on simulation and other techniques implemented on operational production plans.

Goodman and Sarin (1988) use a two-stage modeling approach to determine the truck and loader movements in a surface coal mine. In the first stage, they form an IP model that determines the schedule of extracting lifts in order to minimize the total time required. They consider the precedence of the lifts as well as the available fleet. In the next stage, another IP is formed to decide on the destination of the extracted overburden. Since there are multiple cells, with different distances to the lifts and limited capacities, to dump the overburden in, Goodman and Sarin (1988) use an optimization model allocate loaders to lifts and to determine the destination cell for each truck based on the velocities and times required for loading and dumping. They test their models on a single-seam production data from a coal mine and report improvements in the utilization of the fleet and the transportation time. Similarly, Soumis et al. (1989) break down the operational planning problem into smaller steps. Their three-step approach consists of determining the shovel locations, allocating trucks to shovels and a real-time dispatching model. Soumis et al. (1989) report significant decrease in truck waiting times at shovels and material destinations and increase in the productivity of the system while meeting production goals.

Smith (1998) compares manual scheduling methods to MIP based techniques and provides guidelines on how to use AMPL/CPLEX to develop short-term production plan. Smith (1998) investigates various constraints and objective functions that may be applied to the short-term production plan. Gholamnejad (2008) develops a zero-one integer programming model for short-production planning seeking to maximize the production of contaminants in the early periods with respect to grade blending, ore production, mining capacity and accessibility constraints. Gholamnejad (2008) mentions that the problem size is increased because of the zero-one variables and does not provide any case-studies to evaluate the performance of the proposed model.

Finally, Eivazy and Askari-Nasab (2012) propose an extensive MILP model that minimizes the total cost of mining, processing, rehabilitation, rehandling and haulage. Their MILP formulation decides on the period of extraction of blocks, their destination (process, stockpile or waste dump), the ramps and exits the loads should be taken through and the tonnage of material retrieved from the stockpiles in each period. Their model is capable of handling multiple processing facilities as well as multiple stockpiles with grade limitations. However, the size of the model is beyond the capabilities of commercial MILP solvers and they use aggregation to reduce the number of variables in model.

2.3.3 *Underground Mine Planning*

Although open pit mining the most common mining method, underground mining is used to mine deposits with high waste to ore ratio. Furthermore, some mining operations switch from open pit to underground when they reach a specific depth that open pit mining is not economical anymore. Underground mining is performed with a variety of techniques, such as room and pillar, sublevel caving, sublevel stopping, longwall mining and block caving (Newman, et al., 2010). Because of this variety, the production planning models are usually developed for a single technique or even a single mining operation. In this section, we review some of the mathematical formulations proposed for underground mine planning. Newman et al. (2010) and Alford et al. (2007) offer good comprehensive reviews of the underground mine planning literature.

Similar to open pit mining, underground mine production planning starts by strategic planning. In this step, the model decides on the tonnage of ore and waste extracted from the mine in each period, the extent of ore body to be extracted and the location of the tunnels and haul roads. Yun et al. (1990) deal with the problem of determining the number of the openings and their placement in a sublevel caving mine. They solve their problem by implementing GA and try to minimize the total costs. Brazil et al. (2003), followed by Brazil and Thomas (2007) propose an integrated model to design declines, ramps and drives with an objective of minimizing the total cost of development and material handling. They propose a heuristic that determines the three-dimensional geometry of haulage ramps. They implement their algorithm on a real-size mine and report AU$1.5 million of saving from their design compared to the currently in place design.

Newman and Kuchta (2007) form a mathematical formulation to determine the starting time of each machine placement in an underground mine. They aggregate periods into phases to reduce the size of the problem. Their objective function is to minimize the deviation from the targeted production. Newman and Kuchta (2007) solve the aggregated model and disaggregate the solution to determine the machine placements in each period. In addition, they calculate the worst-case performance of the aggregated model and propose an upper bound for the solution to the primary problem. Few years later, Martinez and Newman (2011) model short- and long-term plans for the same mining operation simultaneously and decompose the problem into five sub-problems using a heuristic. In addition, they use Critical Path Method (CPM) to determine the earliest start time of each machine placement and eliminate the variables for prior periods. They also break the symmetry of the objective function and show that the model with unsymmetrical objective function is solved faster.

Nehring et al. (2012) integrate the short- and mid-term production plans of a sublevel stopping mine with two different objective functions: minimizing deviation from target head grade for the short-term plan and maximizing the net present value for the mid-term plan. Nehring et al. (2012) implement their model on a conceptual case-study and report a slight improvement in the integrated model compared to the two models solved separately.

Rahal et al. (2003) propose a mathematical formulation to determine the production schedule in block caving operations. Their objective function consists of minimizing deviations from the targeted production and, at the same time, minimizing deviations from a given drawpoint depletion rates. Pourrahimian et al. (2012) formulate and solve an MILP model to maximize the NPV in block caving operations. They first solve their model in draw point level and determine the draw rates in each period for each draw point. Next, they form aggregates of draw points, solve their model in aggregated level, and compare the processing times and resulting NPVs. Pourrahimian et al. (2012) consider various constraints such as the maximum and minimum draw rates, upper and lower bounds on the extracted material and feed to the plant, number of open draw points in each period and the precedence between the draw points.

In addition to open pit and underground planning, the transition from open pit to underground has attracted researcher's attention. By having open pit operations completed, mining companies investigate their options for switching to underground operations and extract the remaining mineralized rock. Chen et al. (2003) study the effects of pit limit optimization on a combined operation and conclude that including underground operations can significantly affect the results. They propose a model to maximize the profit from the whole deposit while recovering as much mineral as possible. Moreover, Chen et al. (2003) develop a solution algorithm by modifying the moving cone algorithm to find the optimum pit limits while considering underground mining as the next phase of the operation.

Bakhtavar et al. (2009) propose a combined algorithm to determine the optimum transition from open pit operation to underground mining. They formulate two separate mathematical formulations for open pit and underground planning and combine the two for finding the optimum transition that maximizes the NPV of the whole operation. Roberts et al. (2013) study the transition from open pit to underground mining in a copper mine and evaluate various scenarios of providing ore to the processing plant via open pit expansion or underground operation. They conclude that in many areas of the operation, switching to underground mining can increase the NPV of the project.

2.3.4 *Other Emerging Areas*

Operations research in mining industry is not limited to production planning. Mining operations consist of various stages that improvements in each step can significantly affect the profitability of the operation. Moreover, faster and better computers along with advanced solvers have made it easier for researchers to model and to solve larger problems. Therefore, various researchers have used OR techniques to solve mining problems from explorations to processing plants. Equipment selection problem, mining methods selection,

Equipment selection problem is the decision of choosing the appropriate fleet of truck and loaders for a mining operation with respect to the operation constraints. The problem objective is to choose the best purchase policy with minimum cost. Burt and Caccetta (2013) provide a comprehensive review on the equipment selection literature and various modeling and solution procedures proposed by the scholars in the area.

Another emerging area in mining that is attracting operation research scholars is the stochastic optimization of mine operations. Most of the literature reviewed in the mine planning section is considering predetermined fixed values for ore grades, availability of the equipment and product prices. However, uncertainty exists in every mining operation and many researchers are devoting their efforts to include that in the planning procedure. Interested reader can refer to Dimitrakopoulos (2011) for review on the stochastic optimization of mine plans.

2.4. Rationale for PhD Research

The open-pit production planning problem has attracted numerous researchers in the past few decades. This is due to the fact that open-pit mining is still the first choice of mining even though resources are becoming more scare and lower in grade and quality. The open-pit production planning problem is a very important step that can significantly affect the profitability of the operation. On the other hand, the problem is very complex and modeling it in details usually results in extremely large mathematical problems. Therefore, researchers are divided in different groups trying to deal with the problem via different means. Some of the literature in the area is dedicated to using heuristics and meta-heuristics for planning in order to avoid large time consuming models. On the other hand, there are other researchers who decided to use dynamic programming to address the dynamic nature of the planning problem. However, the largest group of researchers used mathematical programming, and in particular linear and integer programming, to address the planning problem since mathematical programming is more flexible and provides optimal solutions.

Since Johnson (1969) proposed his linear programming model for LTOPP, various researchers struggled with including more details and finding solution techniques that can handle large-scale models. Among the various proposed methodologies, aggregation is more promising technique as it is able to solve larger problems and results in more practical and implementable solutions. However, the proposed aggregation techniques share one or more of these shortcomings:

1. Most the proposed aggregation techniques do not account for block similarities when combining them into larger units. This can introduce material dilution and

create discrepancies between the solutions provided and the real outputs of the operation.

2. In cases where the aggregation techniques consider similarities and similarities between blocks, the algorithm does not have control over the size and shape of the mining units. This can result in production schedules that are not practical from the mining operations point of view.

Therefore, there is a need to create a mathematical model along with an aggregation technique that can include better details of the mining operation and result in more practical implementable schedules. This research introduces a clustering technique that can form mining aggregates based on material similarities with controlled size and shape and a mathematical formulation that can handle different units for making mining and processing decisions and controls extraction schedule, material destination, grade blending and stockpiling at the same time.

2.5. Summary and Conclusion

In this chapter, we reviewed the origins of Operations Research and how it affected the mining industry. Various implementations of OR in mining industry were reviewed to point out the growing trends and vastness of opportunities OR can bring to the industry, among those is the area of mine planning. Many researchers in the past few decades have allocated their time and resources to improve the production plans for open pit and underground mines in short- to long-term. Since this project is dealing with the long-term open pit production planning problem, we spent the majority of this chapter reviewing the past work in this area.

Although mathematical models for LTOPP have been in place for years, the common industrial practice is to use the heuristic algorithms to avoid the intense resources required by mathematical models. This is despite the fact that heuristics not only cannot guarantee the quality of the obtained solutions but also are problem specific and cannot be easily modified to account for technical and case-specific features. Meta-heuristics such as Genetic Algorithm, Simulated Annealing and Ant Colony Optimization are also used but suffer the same drawbacks as heuristic methods.

On the other hand, the MILP formulations require lots of computational power and are not solvable in real-scale mining problems. Therefore, researchers looked for ways of breaking down the LTOPP into smaller problems, decreasing the problem size, developing problem-specific solution procedures and aggregating blocks and periods. Among the reviewed workarounds, aggregation seems to be producing more promising results as (i) the generated schedules are more practical, (ii) the formulations are faster to solve, (iii) they can be easily modified to account for case-specific features, and (iv) they are easier to implement. However, how these aggregates are formed can significantly affect the quality of generated solutions. In this sense, we focused on using clustering techniques to create aggregates homogenous in block characteristics.

Theoretical Framework

3. THEORETICAL FRAMEWORK

3.1. Introduction

Long-term open pit production planning aims at determining the optimum sequence of extracting material from the mine and sending them to different destinations. The material extraction equipment is expensive and limited. The destinations can be processing plants, waste dumps and stockpiles. Processing plants have limited capacities to process material. On the other hand, some processing and extraction techniques require specific tonnage and head grades to be able to extract minerals with desired efficiencies. Waste dumps can also have limited capacities due to geographical or technical restrictions. Stockpiling is also a complicated procedure in which valuable material is deposited in a temporary dump and reclaimed and fed to plant in later periods. Stockpiling is usually performed to feed the plant with smoother, better-controlled head grades or to keep low-grade ore for processing in later periods when the mine runs out of high-grade ore. An optimal production plan has to account for these restrictions and maximizing the net present value of the operation. Although NPV is the most common goal for long-term mine production planning, other objectives such as minimizing deviations from desired production, and maximizing the reserves. Formulating this complicated problem with mathematical programming creates an extensively large model that cannot be solved using current solvers and computational power. Therefore, finding ways of simplifying the model and making the problem tractable is another challenge on the way.

This chapter describes the formulation developed for determining the optimum long-term multi-destination open pit production plan. It also describes the clustering procedure developed to group blocks into larger units. The performance of the clustering procedure and the mathematical model are then illustrated via a small block model. This allows us to run various scenarios on the planning problem in reasonable time and comment on the advantages and shortcomings of the proposed approach.

3.2. Long-term Open pit Production Planning

Long-term open pit production planning is a multi-step procedure that starts by having the estimated block model of the deposit and provides the best extraction and processing sequence of the blocks with respect to technical and geological constraints. The input block model and the technical parameters affecting the schedule can have stochastic or deterministic values. However in this project, we dealt with deterministic production planning assuming all the block properties and scheduling parameters are predetermined. We have developed a MATLAB® application with GUI to facilitate planning. The planning procedure can be summarized to the following steps:

1. Create the block model using Geostatistical tools and techniques such as GSLib (Deutsch & Journel, 1992) or Gems (GEOVIA, 2014a)
2. Determine the operation parameters and calculate the costs and revenues of extracting and processing blocks
3. Determine the ultimate pit limits using LG algorithm (Lerchs & Grossmann, 1965)
4. Import the blocks within the UPL into the application
5. Assign pushback IDs to the blocks. The pushbacks are determined using Whittle™ (GEOVIA, 2014b) or the application created by Mieth (2012).

6. Aggregate blocks into larger units using clustering algorithm. Two clustering algorithms with various control parameters are developed and are discussed in details in Section 3.3.
7. Prepare MILP inputs by setting up planning parameters and choosing desired resolutions.
8. Preprocess the matrices and remove unnecessary columns and rows as explained in Section 3.4.
9. Form the matrices and solve the problem using TOMLAB/CPLEX (Tomlab Optimization AB, 2011) solver.
10. Interpret and plot the results and evaluate the quality of the generated schedule.
11. If required, modify planning parameters and redo planning procedure until a satisfying schedule is produced

The first step is to create a block model which discretizes the area into a 3D grid called blocks. Each block is then assigned various properties such as tonnage, rock type and grade. These properties are estimated based on available drillhole and seismic data using Geostatistical methods. Various software packages are available for performing this task and details of the procedure are out of this project scope. Afterwards, the costs and revenues of extracting and processing each block is calculated based on element selling prices as well as extraction and processing costs. The value of each block is calculated as the total revenue earned by processing the material in that block and selling the final product minus the costs associated with extracting and processing it. Equation (3.1) illustrates the block value calculation for a multi-element deposit.

$$BV_i = \sum_{e \in E} R^e \times r_i^e \times g_i^e \times o_i - (o_i + w_i) \times MC_i - o_i \times PC_i \tag{3.1}$$

Where

- R^e is the unit revenue of selling element e
- r_i^e is the recovery of element e in the processing plant based on rock type and other properties of block i
- g_i^e is the grade of element e in block i
- o_i is the total ore tonnage of block i
- w_i is the total waste tonnage of block i
- MC_i is the unit mining cost adjusted based on rock type and other properties of block i
- PC_i is the unit processing cost adjusted based on rock type and other properties of block i

For example, a block value from a Gold, Silver and Copper deposit can be calculated as in equation (3.2):

Table 3.1. Block Value Calculation Example

Ore Tonnage	Waste Tonnage	Adjusted Mining Cost	Adjusted Processing Cost	Elements	Unit Revenue	Recovery	Grade
4,025 tonnes	0 tonnes	$1.93 /tonne	$6.47 /tonne	Au	$1,330 /oz	72%	0.008 oz/tonne
				Ag	$21.5 /oz	38%	0.094 oz/tonne
				Cu	$3.2 /lb	46%	0.21 lb/tonne

$$\begin{aligned} BV_i &= (1,330 \times 0.72 \times 0.008 + 21.5 \times 0.38 \times 0.094 + 3.2 \times 0.46 \times 0.21) \times 4,025 \\ &-1.93 \times 4,025 - 6.47 \times 4,025 = \$1,360 \end{aligned} \tag{3.2}$$

The block value is then used to determine the UPL using the LG algorithm. Blocks falling outside the UPL are removed from the model to decrease the size of the model. The next step is to determine the pushbacks. The common approach to pushback design is called parameterization approach. In this process, the selling prices of the elements are iteratively changed and the UPL is determined based on the modified block values. The resulting UPLs form nested pits. Afterwards, an experienced user or an automatic algorithm chooses a number of the pits as the pushbacks. The other approach used in this project is the BIP approach proposed by Mieth (2012). In this approach, a binary integer programming model is formed in order to determine pushbacks with equal tonnage of ore and waste in each. The model is then solved using a commercial solver or by the heuristic algorithm proposed. After determining the pushbacks, mining-panels or bench-phases are formed as the intersections of the pushback boundaries and benches. Afterwards, the clustering algorithm is used to aggregate blocks into mining-cuts while respecting the mining-panel boundaries. The details of the clustering algorithms developed are presented in the next section. The generated mining-cuts and –panels are then used to form the matrices required for the MILP model. In this step, a pre-processing algorithm is used to remove unnecessary variables without losing optimality. This algorithm is explained in more details in Section 3.4. The matrices are then normalized and sent to the TOMLAB/CPLEX (Tomlab Optimization AB, 2011) solver where the optimal solution of the problem can be determined. The result is then imported back to the program for interpretation and plotting. After interpreting the results and plotting the schedules, there might be a need to modify planning parameters and run the model again until a satisfying schedule is generated.

3.3. Clustering

Clustering is the process of grouping similar objects into clusters while trying to maximize the intra-cluster similarity and at the same time inter-cluster dissimilarity. Various clustering algorithms are developed in the literature and are reviewed in section 2.2. These algorithms can be categorized based on how they form clusters and based on their purpose. From the methodological point of view, clustering algorithms are divided into partitional and hierarchical algorithms. Partitional algorithms work by iteratively partitioning the dataset into clusters and evaluating a performance measure. Their variation comes from how they try to improve the performance measure in each step. Hierarchical algorithms, as their name imply, work by forming a hierarchy of the objects. This can be by starting from considering each object as a cluster and merging them into larger ones or by considering the whole dataset as one cluster and breaking it down into smaller clusters. The former group is called agglomerative hierarchical clustering opposed to the later one which is called divisive hierarchical clustering. The other categorization is general- versus specific-purpose algorithms. In this project, we have developed 2 specific-purpose clustering algorithms. The first one is an agglomerative hierarchical algorithm and the second one is a k-means algorithm which is a partitional clustering algorithm.

3.3.1 *Agglomerative Hierarchical Clustering Algorithm*

Hierarchical clustering algorithms usually work based on a similarity index. We defined a similarity index based on our purpose of clustering with the following factors affecting the similarity between blocks:

1. Distance: since the created clusters are going to be used as units of scheduling, blocks closer to each other have to be grouped together

2. Rock types: we are going to assign mining and processing costs to mining-cuts and these costs vary based on rock types. Therefore, the more homogenous mining-cuts are more favorable as they introduce less error into calculations.
3. Grade: grades are also going to be calculated for each cluster and more homogenous clusters give us more accurate results. Since a block model usually accounts for more than one element, the grade of the major elements of interest is used.
4. Destination: if the clusters are going to be used as units of planning in a predetermined cut-off scenario where the destination of the blocks is determined prior to planning, the clusters have to be formed from blocks with the same destination.
5. Beneath cluster: if the clusters are used as units of mining, they have to be formed in a way that the scheduler can get to high grade blocks at the bottom of the pit faster. This means lower number of precedence arcs between clusters of consequent benches. Having this factor in determining similarity provides better initial solutions for Tabu search procedure explained in section 3.3.2.

The similarity between two blocks can now be determined based on these factors. There are various methods for calculating similarity or dissimilarity between two values. The Minkowski distance is a common method of calculating the dissimilarity between two numeric values and is used in this project. This includes the well known Euclidean and Manhattan distance measures. However, this method cannot be used for categorical attributes such as rock type and destination. There are other methods that can be used to calculate the difference between categorical values. The simplest method is to check if the values are equal or not equal. This is called simple matching and has been used in clustering objects with categorical attributes in Huang (1997) and Huang (1998). The shortcoming of this method is that it cannot account for the degree of dissimilarity between the values. Another way of dealing with categorical attributes is to perform binary encoding on the categorical variable. Nevertheless, binary encoding does not account for dissimilarities between categories either (Hsu, Chen, & Su, 2007). Consequently, the idea of distance hierarchy is proposed by Hsu et al. (2007) to be able to calculate similarities between categorical variable. In this method, the categorical are placed on a weighted tree and the distance between two values is calculated from the links between the two values. Distance hierarchy is an accurate comparison but requires a great deal of experience and knowledge of the data to tune the weights. Another method for calculating the dissimilarity between categorical values is to use penalties as proposed by Dósea et al. (2008). This is similar to simple matching used by Huang (1997) and Huang (1998) but the penalties can be calibrated to account for the extent of dissimilarity between the values.

In order to calculate the similarity between the blocks in the block model, we have used a combination of Euclidean distances and penalty values. Euclidean distance is used to determine the dissimilarity between numerical attributes (distance and grade). The calculated distances are normalized with the maximum values to be able to combine them with other attributes. This results in having differences between zero and one. The normalized differences are then powered by their weights and multiplied together. Since the normalized differences are between zero and one, the powered values are still going to be between zero and one. On the other hand, penalties are used for categorical attributes (rock type, destination and beneath cluster). Penalty values have to between zero and one in order to be in the same scale as numerical differences. Accordingly, the similarity between two blocks can be determined based on equation (3.3). Note that S_{ij} is not calculated for $i = j$ and is equal to zero.

$$S_{ij} = \frac{R_{ij} \times C_{ij} \times D_{ij}}{\tilde{L}_{ij} \qquad {}_{ij}} \tag{3.3}$$

Where:

- R_{ij} : is the penalty assigned if blocks i and j are from different rock types
- C_{ij} : is the penalty assigned to blocks i and j not located above the same cluster
- D_{ij} : is the penalty assigned if blocks i and j are not assigned to the same destination
- $\tilde{L}_{ij}$: represents the normalized distance value between blocks i and j
- $\tilde{C}_{ij}$: represents the normalized grade difference between blocks i and j
- W_D : is the distance weight
- W_G : is the grade difference weight

As mentioned earlier, the penalties are assigned if the categorical values are different. Since they are multiplied by other factors, a value of 1 is the neutral values while smaller penalties mean higher effect on similarity index. The categorical dissimilarities R_{ij}, C_{ij} and D_{ij} are calculated as in equations (3.4) to (3.6) respectively. The penalty values $r \in (0,1]$, $c \in (0,1]$ and $d \in (0,1]$ are calibrated via try and error.

$$R_{ij} = \begin{cases} 1 & \textit{if blocks i and j are from the same rock type} \\ r & \textit{otherwise} \end{cases} \tag{3.4}$$

$$C_{ij} = \begin{cases} 1 \; \textit{if blocks i and j are located above same cluster} \\ c \qquad\qquad\qquad\qquad\qquad\qquad \textit{otherwise} \end{cases} \tag{3.5}$$

$$D_{ij} = \begin{cases} 1 \; \textit{if blocks i and j are assigned to the same destination} \\ d \qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{otherwise} \end{cases} \tag{3.6}$$

The distance factor is the normalized Euclidean distance between center points of blocks i and j (equation (3.7)). Since the clustering is performed bench by bench, the elevation difference is not included in distance calculations. In the same way, the grade difference is calculated using Euclidean difference between grade values (equation (3.8)). However, we usually use only one major element grade. On the other hand, it is possible to have zero values for grade difference which can cause division by zero errors or infinite similarities. Therefore, zero grade differences are replaced by a very small value.

$$\tilde{L}_{ij} \quad \frac{\sqrt{\left(x - x_j\right)^2 + \left(y_i - y_j\right)^2}}{D_{\max}} \tag{3.7}$$

$$\tilde{C}_{ij} \quad \begin{cases} \dfrac{\sqrt{\sum_e \left(g_i^e - g_j^e\right)^2}}{G_{\max}} & \text{if} \sum_e \left(g_i^e - g_j^e\right)^2 \neq 0 \\ \varepsilon & \text{if} \sum_e \left(g_i^e - g_j^e\right)^2 = 0 \end{cases} \tag{3.8}$$

The clustering algorithm is design to cluster blocks bench by bench since the resulted clusters are going to be used in solving the production planning problem. It starts by clustering blocks from the lowest bench where high grade value blocks are usually located to the surface of the mine. The algorithm starts by calculating the similarity matrix (S) of the bench holding similarity indexes between all pairs of blocks. The similarity values for similarity between each block and itself (S_{ii}) is not calculated and is assumed zero. Since each mining-cut is a unit of planning, fragmented clusters are not favorable. The mining units are preferably continuous aggregation of blocks that can be mined without relocating the shovel. Therefore, we calculate an adjacency matrix (A) which holds zeros and ones indicating if two blocks are adjacent or not. Two blocks are considered adjacent if they share a side. In this definition, every block has a maximum of 4 adjacent blocks. Therefore, we start clustering in each bench by calculating two $N \times N$ matrices (assuming there are N blocks on the bench).

As mentioned earlier, the developed algorithm is an agglomerative hierarchical clustering method. It starts by considering every block as a cluster and merges the most similar ones into a larger cluster. If multiple pairs of blocks share the most similar position, the one with the largest index is chosen. The process can continue until all the clusters are merged into one large cluster. However, for our purpose, we have to respect 3 constraints: continuous clusters, number of clusters and maximum cluster size. To control the first constraint, we multiply the similarity matrix by the adjacency matrix. The similarity between not adjacent blocks will become 0 and they will not be chosen for merging. For the next constraints, we stop the merging process when we reach the desired number of clusters on the bench. Finally, the sizes of the clusters are checked every time two clusters are chosen to be merged. If the size of the merged cluster exceeds a predetermined threshold, the merging is avoided and the two clusters are marked as not adjacent to prevent them from being chosen again.

After choosing the most similar blocks (namely i and j) and merging them into a larger cluster, the similarity and the adjacency matrices have to be updated. There are 3 different methods used to calculate the similarity between the merged cluster and other clusters:

single link, complete link and average link (Jain & Dubes, 1988). The first method calculates the similarity between the two clusters as equal to the similarity between the two most similar objects from the two clusters. In opposite, the complete link method calculates cluster similarity as the similarity between the most dissimilar objects of the clusters. The average link method calculates the similarity between clusters as the average of similarities between all pairs of objects from the two clusters. Having tested the 3 methods, we used the complete link approach in this project as in equation (3.9). The updated similarities are going to be placed in the row and column associated with the cluster with smaller index and the other cluster is removed from the set by setting its row and column to 0.

$$S_{(i,j),k} = \min\left(S_{ik}, S_{jk}\right) \qquad \forall k \in \{1,...,N\}, \quad k \notin \{i,j\} \qquad (3.9)$$

The adjacency matrix also needs to be updated. The merged cluster is considered adjacent to other clusters if at least one of the blocks in that cluster is adjacent to one of the blocks in the other cluster. Therefore, the adjacency matrix can be updated using a max operator (equation (3.10)). The pseudo code of the process is illustrated in Figure 3.1.

$$A_{(i,j),k} = \max\left(A_{ik}, A_{jk}\right) \qquad \forall k \in \{1,...,N\}, \quad k \notin \{i,j\} \qquad (3.10)$$

```
SimilarityMatrix = Calculate_Similarity_Matrix()
AdjacencyMatrix = Create_Adjacency_Matrix()
NumberofClusters = NumberofBlocks
WHILE NumberofClusters > MaximumNumberofClusters
    (i,j) = Find_Most_Similar_Adjacent_Clusters()
    IF length(Clusters(i)) + length(Clusters(j)) <= MaximumClusterLength THEN
        SimilarityMatrix(i,:) = Min(SimilarityMatrix(i,:), SimilarityMatrix(j,:))
        SimilarityMatrix(j,:) = 0
        AdjacencyMatrix (i,:) = Max(AdjacencyMatrix (i,:), AdjacencyMatrix (j,:))
        AdjacencyMatrix (j,:) = 0
        Clusters(i) = Clusters(i) + Clusters(j)
        NumberofClusters = NumberofClusters – 1
    ELSE
        AdjacencyMatrix (i,j) = 0
    ENDIF
ENDWHILE
```

Figure 3.1. Hierarchical Clustering Pseudo Code

The hierarchical clustering process is illustrated in Figure 3.2. Assume similar colors in the Figure 3.2 (a) represent the main similarity factor e.g. grade and the blocks are clustered based on their color. The algorithm is set to form 8 clusters not exceeding 10 blocks in size. In the first step, the most similar blocks are merged together to form a cluster. The process continues in Figure 3.2 (b) to (d) by merging blocks 1 and 2, 3 and 4, and 5 and 6. In Figure 3.2 (e) the similarity between block 7 and the cluster formed by 5 and 6 is the highest similarity and 7 is a neighbor of cluster 5-6. Therefore, a cluster with 3 blocks is formed. Afterwards, blocks 8 and 9 are merged to form a cluster. In the next iteration the highest similarity is between clusters 1-2 and 8-9 and block 2 from cluster 1-2 is a neighbor of block 8 from cluster 8-9. Therefore, the two clusters are merged. The process continues until the clustering scheme Figure 3.2 (z) is created.

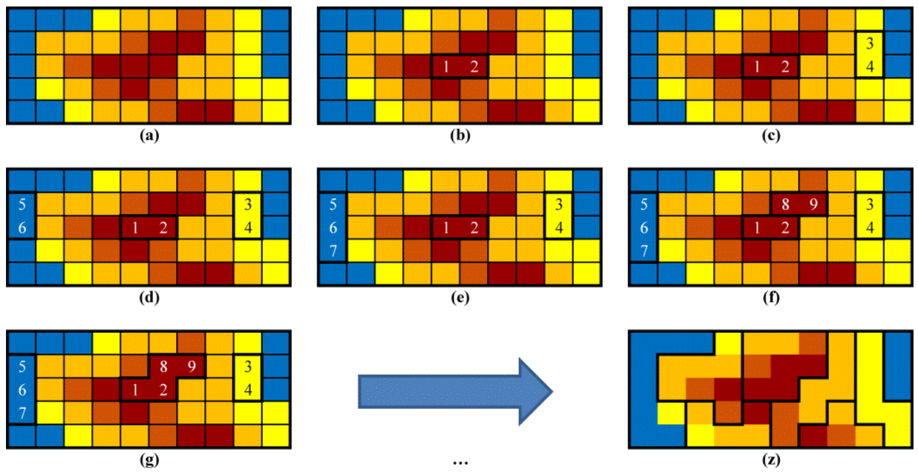


Figure 3.2. Hierarchical Clustering Illustration

After forming the clusters two post-processing steps can be called to improve the clustering results. The first method is a Tabu Search procedure that is designed to decrease the number of precedence constraints between the generated clusters. The second one is a shape refinement procedure that tries to improve the shape of the generated clusters by removing very small clusters as well as removing sharp corners.

3.3.2 *Tabu Search*

Tabu Search (TS) is a modified local search algorithm that holds a Tabu list to avoid revisiting same neighbor solutions in consequent iterations. The procedure starts by having an initial solution and improving it based on a measure of goodness. The measure of goodness is the objective of the search. The initial solution here comes from the hierarchical clustering step. However, our goal of using TS is not improving the clustering objective which, as mentioned earlier, is maximizing the similarity within the objects of the same cluster and dissimilarity between objects of different clusters. In this project, we are using TS to decrease the number of precedence arcs between generated clusters of different benches. However, we try to preserve the intra-cluster similarity by defining the measure of goodness as in equation (3.12).

$$IC_c = \frac{1}{n_c^2} \sum_{i=1}^{n_c} \sum_{j=1}^{n_c} S_{ij} \qquad \forall i, j \in B_c \tag{3.11}$$

$$GM = \frac{\overline{IC}^{W_c}}{NA^{W_n}} \tag{3.12}$$

Where:

- IC_c: is the intra-cluster similarity of cluster c
- n_c: is the number of blocks in cluster c
- B_c: is the set of blocks in cluster c
- $\overline{IC}$: is the average intra-cluster similarity of the bench. The intra-cluster similarities are calculated based on equation (3.11)
- NA: is the number of precedence arcs between clusters of this bench and the lower bench
- W_c: is the weight for intra-cluster similarity measure
- W_n: is the weight for number of precedence arcs

Tabu search, similar to hierarchical clustering, is implemented bench by bench starting from the lowest bench. The TS procedure starts by calculating the number of precedence constraints and measure of goodness for the current clustering scheme in each bench. This calculation consists of determining the number of precedence arcs between clusters of the current bench as well as the blocks of the current bench with the clusters of the lower bench. In every iteration of the search, k clusters are chosen to form the candidate list. These are the clusters that have the maximum number of precedence arcs with the lower bench clusters. Then, l blocks with the highest number of precedence arcs with the lower bench clusters are chosen from each candidate. This gives us $k \times l$ candidate blocks. Each block has a maximum of 3 neighbor clusters as illustrated in Figure 3.3 (block 4 has 3 neighbour clusters compared to block 1 without any neighbor clusters). In the next step, we determine the possible modifications by detaching the block from its cluster and attaching it to a neighbor cluster. This can result in a maximum of $k \times l \times 3$ solutions. However, not all the modifications can produce valid solutions as some of them may create fragmentation in the clusters. For example, detaching block number 3 in Figure 3.3 and attaching it to either of the adjacent clusters can cause a fragmentation in the cluster. In contrast, blocks 2, 4 or 5 in Figure 3.3 can be detached from their current clusters and be attached to either of the neighbor clusters. After creating the list of candidates, all the valid neighbor solutions are evaluated for measure of goodness and the one with the maximum improvement is selected. This neighbor solution is then compared to the Tabu list to make sure that it has not been tried before. If the solution is an unvisited solution we choose it as the current clustering scheme and add it to the Tabu list. It is kept in the Tabu list for a predetermined number of iterations called the lock out period. The solution along with its measure of goodness is kept in a separate list called good solutions.

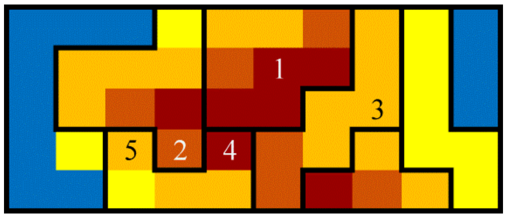


Figure 3.3. Number of Neighbor clusters

The TS procedure is repeated for a predetermined number of iterations. At the end, the best solution with the highest measure of goodness is chosen from the good solution list and replaces the initial bench clustering scheme from hierarchical clustering. Afterwards, the algorithm moves on to the next bench until it reaches the surface of the mine. The pseudo code of the algorithm can be found in Figure 3.4. Glover and Laguna (1997, 2013) can be consulted for more details on Tabu search and its implementations.

```
IterationNum = 1
WHILE IterationNum < MaxNumberofIterations
    IterationNum = IterationNum + 1;
    FOR ClusterLoop=1 TO NumberofClusterstoAlter
        ClusterID = Find_Cluster_With_Highest_No_of_Arcs()
        ListofClustersToAlter += ClusterID
        FOR BlockInClusterLoop=1 TO NumberofBlockstoAlter
            BlockID = Find_Block_With_Highest_No_of_Dependent_Clusters (ClusterID)
            ListofBlocksToAlter += BlockID
        ENDFOR
    ENDFOR
    FOR ClusterLoop=1 TO NumberofClusterstoAlter
        FOR BlockInClusterLoop=1 TO NumberofBlockstoAlter
            IF Is_Block_Dettachable(NumberofBlockstoAlter(BlockInClusterLoop),
            NumberofClusterstoAlter(ClusterLoop)) THEN
                AlteredClusterings += Generate_All_Neighbor_Soultions
            ENDIF
        ENDFOR
        BestSoultion = Select_Max_State_Measure(AlteredClusterings)
        IF NOT Is_In_List(TABUList, BestSoultion) THEN
            TABUList += BestSoultion
            Set_Best_Solution_As_Current_State()
        ENDIF
    ENDFOR
ENDWHILE
```

Figure 3.4. Tabu Search Pseudo Code

3.3.3 *Shape Refinement Procedure*

The clustering algorithm developed in this project is aimed at creating aggregates of blocks that can be used as units of mining and processing in open pit mines. These units need to be homogenous in rock type and grade to decrease the errors in disaggregation of the solutions back to block level. On the other hand, the aggregates have to be minable from the operations point of view. Very small aggregates will increase the number of drop cuts in the resulted schedules. In addition, very large aggregates limit the flexibility of the scheduler. Moreover, the shapes of the generated clusters are important. Very narrow clusters or clusters with sharp corners will decrease the practicality of the generated schedule as different shovels can mine with different selectivity and require a reasonable operating area. Therefore, we developed a post processing procedure that improves the sizes and shapes of the generated clusters. We call this step the shape refinement procedure.

As mentioned in section 3.3.1, we can define a maximum cluster size in the developed clustering algorithm. The maximum cluster size along with the number of clusters on the bench, to some extent, controls the size of the generated clusters. However, the algorithm can create very small clusters. The first step in shape refinement is to remove very small clusters. The minimum cluster size is an input to the procedure and is determined based on the purpose of clustering and the mining equipment used. The first step in the shape refinement procedure is finding clusters smaller than the minimum acceptable size and breaking them down into single block clusters. For instance, assuming the minimum cluster size is set to 4, the blue cluster in Figure 3.5 (a) is broken into 3 clusters in Figure 3.5 (b). This idea is borrowed from Barca and Rumantir (2007) where the authors break down small clusters after they cluster objects with their k-means algorithm. Next, the corner blocks of clusters are identified. Based on our definition, corner blocks are the ones that only have 1 neighbor from their own cluster and have 2 or more neighbors from another cluster. The identified corner blocks are marked with an "X" sign in Figure 3.5 (b). The corner blocks are then detached from their original cluster and attached to the neighbor cluster. Afterwards, the cluster sizes are checked again. If a cluster size falls below the minimum acceptable size after detaching a block from it, it will be broken down into blocks. On the other hand, it is worth mentioning that the maximum cluster size constraint may be violated when a block is detached from one cluster and attached to the other one. The process continues for a predetermined number of iterations. However, the process stops if there are no refinement actions identified in the iteration.

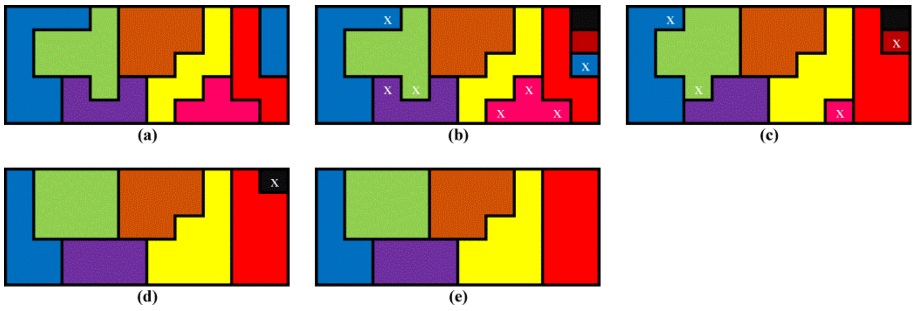


Figure 3.5. Shape Refinement Illustration

The number of shape refinement iterations and the minimum acceptable cluster size are predetermined and can affect the result of shape refinement. The more iterations used the smoother clusters are formed. However, the shape refinement procedure does not take similarity into account. Therefore, extensive shape refinement can sabotage the intra-cluster similarity gained via hierarchical clustering. The trade-off between the smoothness of the clusters and their homogeneity largely depends on the purpose of clustering and the type of operation is place. Mining operations on stratified or porphyry deposits using large excavation equipment require smoother larger clusters. In contrast, hard metal selective mining can deal with small jagged mining units but requires high homogeneity in the mining units. Therefore, minimum and maximum sizes as well as number of refinement iterations are decisions that have to be made by an expert based on the mining operation properties.

3.3.4 *Clustering with Direction*

The developed clustering algorithm is not only useful for creating long-term planning mining unit, but also can be used to create other mining polygons that are generally drawn by hand. Short-term scheduling polygons, ore control polygons and blasting units are other aggregates that can be formed using the developed clustering algorithm. The similarity index weights, minimum and maximum size and shape refinements can used to calibrate the clustering algorithm based on its purpose. However, including the direction of mining in forming the clusters is a key feature when dealing with short-term mining and blasting polygons. We have developed two different direction factors to be included in the similarity index and account for the mining direction. We call them straight and spherical direction factors. The straight direction factor is useful for directional mining of stratified deposits where the shovels start from one side of the pit and advance to the other side. On the other hand, the spherical direction factor is more useful for deep hard metal mines where the expansion starts from the pit center and shovels push back the pit walls bench by bench. Figure 3.6 (a) illustrates a right to left mining direction and how the polygons are formed perpendicular to the mining direction. In contrast, Figure 3.6 (b) shows the clusters required in a central expansion operation. Note that in both cased, the clusters have to follow the formations of rock and distributions of the grades and they cannot be arbitrarily drawn. Identifying these polygons is currently a manual procedure that an expert has to spend hours drawing polygons based on the available information.

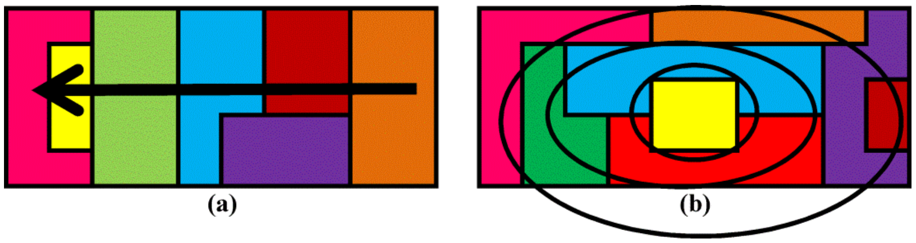


Figure 3.6. Directional Clustering

The first mining direction factor is based on a direction vector provided by the user. Since the direction of mining can be affected by the shape of the bench and the location of the ramps, this vector has to be specified separately for each bench. The two end points of the vector are called clustering reference points. Afterwards, the mining direction factor is calculated as in equation (3.14) and used in determining block similarity with other factors as in equation (3.13). M_i^1 and M_i^2 are the distances from block i to the reference points 1 and 2 respectively. The $Sign()$ function returns $+1$ if the value is positive and -1 if the value is negative. M_{ij} is the normalized Euclidean distance between values of M_i and M_j. If the two values are equal the difference is replaced by a very small value to avoid division by zero.

$$S_{ij} = \frac{R_{ij} \times C_{ij} \times D_{ij}}{\tilde{L}_{ij} \quad \text{[illegible]}_{ij} \quad \tilde{\text{[illegible]}}_{ij}} \qquad (3.13)$$

$$M_i = sign\left(\left(M_i^1\right)^2 - \left(M_i^2\right)^2\right) \times \sqrt{\left|\left(M_i^1\right)^2 - \left(M_i^2\right)^2\right|} \qquad (3.14)$$

$$M_i^1 = \sqrt{(X_i - X^1)^2 + (Y_i - Y^1)^2} \qquad (3.15)$$

$$M_i^2 = \sqrt{(X_i - X^2)^2 + (Y_i - Y^2)^2} \quad (3.16)$$

Similarly, two reference points have to be determined to form spherical clusters. The 2 reference points represent the starting points of the expansion operation. They can be in the center of the pit or the entrance to the bench based on the type clusters required. M_i^1 and M_i^2 are again calculated as the distances from blocks to the reference points. Afterwards, the M_i factor is calculated based on equation (3.17) and used in the similarity index with the appropriate weight.

$$M_i = \sqrt{\left|\left(M_i^1\right)^2 + \left(M_i^2\right)^2\right|} \quad (3.17)$$

3.3.5 *Clustering within Predetermined Boundaries*

There are situation when the created clusters have to respect predetermined boundaries i.e. one cluster is not allowed to be extended to the two sided of a boundary line. The best example is when the clusters have to be formed within the phases of the production. The boundaries can also be arbitrary lines drawn by the planner or the results of a higher level scheduling step. We can easily force clusters to be formed within the boundaries by manipulating the adjacency matrix. We set the adjacency value of two blocks to 0 if they fall on two different sides of the boundary line. Consequently, the two blocks cannot be merged together to form a cluster. This is very useful when we create processing units within mining-panels using the clustering algorithm.

Another implementation of the clustering within boundaries feature is when an absolute distinction between different rock types or predetermined destinations is required. For example, the mine planner may need mining units that are completely made of one rock type or are in their whole sent to one destination. The same boundary logic can be implemented to replace the rock type or destination penalty values defined in 3.3.1 and create clusters of 100% same rock type. However, the shape refinement procedure can be exempted from the strict boundaries to improve the shapes of the created clusters.

3.3.6 *K-Means Clustering*

As mentioned earlier, clustering algorithms can be divided into two categories: hierarchical and partitional clustering. We have developed a k-means partitional clustering algorithm to compare against our hierarchical algorithm. Hierarchical clustering is known to result in better clusters than partitional algorithms but by consuming more time (Feng, et al., 2010). The term k-means is first used in MacQueen (1967) but covers a large series of heuristics proposed in literature which all partition data points by selecting mean points for clusters and assigning each data point to the nearest mean. One extension to k-means clustering which is relevant to this project is the kernel k-means which is developed to be able to partition data points which are not linearly separable by mapping them into a kernel space (Dhillon, Guan, & Kulis, 2004). The modification addressed in this article is to add lower and upper bounds on the number of objects in each cluster. Barca and Rumantir (2007) add a lower bound to the k-means algorithm. In their approach each cluster with less than specific number of members is deleted and its members are assigned to other clusters. The approach in this paper is to add both lower and upper bounds and to preserve the predefined number of generated clusters.

In order to apply k-means clustering on our dataset, we have implemented a variation of k-means algorithm based on support vector machines (SVM) and a first-degree polynomial kernel function. The formulations and theoretical background can be found in Linli Xu et al. (2005). However, we briefly explain the algorithm's basis and implementation. Similar to the hierarchical algorithm, the k-means implementation in this project is performed on each bench independently. The first step for this algorithm is to form the feature matrix, which holds all the important properties of all objects. To be consistent with the hierarchical clustering technique, the same parameters are used with the same weighting approach. However, rock-type is not treated as a categorical variable but a numerical variable. This is necessary to the k-means implementation but damaging to rock-type homogeneity. Afterwards, the feature matrix has to be kernelized in order to get better results. Kernel functions are used to map data points from the initial space to the kernelized space when objects are not linearly separable in their initial space. The same map is used in returning to the initial space with all the objects labeled as belonging to various clusters. Having tested various kernel functions and parameters, a first-degree polynomial kernel function is used in this implementation. Afterwards, k initial cluster centers are randomly selected in the kernelized space and objects are assigned to the nearest mean. In the next step, the objective function, which is a summation of Euclidean distances between all objects and cluster means, is calculated. Cluster means are then manipulated in an iterative manner based on gradient descent until a local minimum is found. This is stored as a solution to the clustering problem and a new replication starts with another initial random definition of cluster means. Finally, all of the replications are compared, and the one with the lowest objective function is selected as the solution to the clustering problem on that bench.

The k-means algorithm implemented in this project suffers a number of setbacks and requires significant improvements to become a useful clustering technique for creating mining units. As mentioned in section 3.3.1, the hierarchical clustering procedure is modified to respect the average and maximum cluster size and to create non-fragmented continuous clusters of blocks. However, the k-means clustering procedure does not control the maximum cluster size and may create very large and very small clusters. On the other hand, k-means clustering is performed in a kernelized space and does not respect the adjacencies of blocks. Therefore, shape refinement and Tabu search procedures could not tested on k-means results. Moreover, k-means clustering deals with numerical attributes and categorical attributes such as rock-type have to be either excluded from the procedure or be included as numbers.

3.3.7 *Evaluating Clustering Methods*

We have developed two different clustering algorithms with various control parameters. There are various parameters that have to be determined by the user and can affect the created clusters. Therefore, there is a need to define measures of evaluating the quality of the generated clusters. These measures have to be aligned with the purpose of clustering. However, not all the required characteristics of the clusters can be quantified. Therefore, we have defined quantitative and qualitative measures for evaluating the performance of the clustering algorithms.

As mentioned earlier, the generated clusters need to be homogenous in grade and rock type. Since grade is a numerical attribute it is easy to define a measure to calculate the homogeneity in grade. It is common to evaluate numerical attributes by the use of squared error criterion (Hsu, et al., 2007). This is the square distance of each data point from the cluster mean. For a single numerical attribute this is the same as the standard deviation. However, a normalized dimensionless measure can be more useful when comparing different element grades or different datasets. Therefore, we divide the standard deviation by the mean and calculate the coefficient of variation (CV) of the grade for each cluster. The average CV of all the generated clusters is then used as a performance measure. Higher grade CV means the algorithm is creating more homogenous clusters.

Rock type is another important attribute of blocks and we usually prefer clusters homogenous in rock type. It is not possible to use CV for rock type since rock type is a categorical attribute. Various homogeneity measures for categorical attributes are introduced in the literature to evaluate clustering algorithm performances. One measure introduced by Hsu et al. (2007) is the categorical utility which is defined as the probability of having two objects from same category in one cluster. Another measure is the relative frequency of categories in clusters proposed by Huang (1997). Since our goal is to have clusters that are mostly formed from one rock type, we define a new measure that is aligned with our goal. We call this rock unity and calculate it as the largest percentage of blocks in a cluster with the same rock type. In other words, rock unity is the percentage of most dominant rock type in the cluster. The average rock unity of all clusters is a measure of evaluating the rock homogeneity of the clustering method.

Another important factor in evaluating the fluctuations in cluster sizes. In many case, the created clusters should be of equal size since they are being used as units of planning. Very small clusters will result in scattered production schedules. On the other hand, very large clusters limit accessibility and flexibility in planning. Therefore, we prefer to have clusters of roughly the same size. Consequently, we use CV of the cluster sizes as a measure of evaluating the clustering scheme. Lower CV is desired as it implies less fluctuation in cluster sizes.

In addition to the aforementioned quantitative performance measures, shapes of the generated clusters have to be evaluated. Mineablity of the created clusters plays an important role in their usability when used as excavation or blasting units. Narrow clusters or clusters with jagged and sharp corners cannot be used as mining units since they do not satisfy excavation equipment requirements. On the other hand, geographically fragmented clusters can create drop cuts in the generated production schedules. Although shape of the created clusters is an important measure it is not possible to be quantified. Therefore, we plot the clusters and comment on the shapes based on the operations characteristics and mining equipment in use.

Finally, the time required to form the clusters is considered a performance measure. As mentioned earlier, tuning the clustering control parameters is a case-specific iterative procedure and is performed by running the clustering algorithm multiple times and evaluating the results. Therefore, faster algorithms are easier and faster to tune and the planner can evaluate more combinations of control parameters in a limited time.

3.4. Multi-Destination Long-Term Open pit Production Planning MILP

In this section, we present a general form of our mathematical formulation that aims to maximize the NPV of the operation while respecting technical and operational constraints. Our proposed model consists of integer and linear variables. Therefore, it is called a mixed integer linear programming (MILP) model. The model is presented in a general notation with mining and processing units. These units can be block, clusters and mining-panels and different combinations of them can be used based on the stage of planning and the type and size of the operation. On the other hand, we know that planning based on smaller units results in higher NPV as it gives us more flexibility in choosing what to process and what to dump as waste. Therefore, solving the model in block level results in over-estimation of the achievable NPV as this is not practical in real mining operations. The question here is what resolution is closer to the real outcome of the operation and better represents the reality of the operation. We will study the differences through case-studies in the forth chapter of this thesis.

The MILP model is designed to determine the extraction period of the mining units as well as the destination of each processing unit. The model is called multi-destination since the sequence of extraction and destination of material is determined simultaneously within the model. The other approach is to determine the destination of material based on a cut-off grade prior to the scheduling step. Since our model is not using a fixed cut-off grade, it can also be called LTOPP with dynamic cut-off grade.

3.4.1 *Model Assumptions*

Mathematical models are used to formulate a real problem and find an optimum solution to the problem. However, every mathematical model is limited by some assumptions and simplifications compared to the real world problem. Therefore, there is a need to describe the assumption that we have prior to presenting the formulation. These assumptions and limitations are:

1. The MILP is formed based on a given Geostatistical block model. Each block is uniquely marked with its coordinates and indices in the block model. The block model is a 3D presentation of the ore body and its surrounding wastes and the height of the blocks is equal to the bench height. Each block is assigned a rock type, a tonnage and multiple element grades and recoveries using Geostatistical methods prior to the planning step.

2. The cost of mining and processing material and the revenues generated from selling the final product is calculated based on fixed deterministic values and assigned to each block. In cases where the costs and revenues of extracting and processing a block is different if extracted in different periods, the costs and revenues can be adjusted when applying discount factors.
3. The mine life, mining fleet, processing and re-handling capacities are assumed given for every period of mining operation and the MILP model tries to find the optimum solution with respect to these constraints. However, it is possible to run multiple scenarios and determine a good combination of capacities.
4. The capital investment, salvage and maintenance costs are not directly taken into the MILP model. They can be deducted from or added to the model output to calculate the actual NPV of the operation.
5. Multiple material destinations and stockpiles can be defined in the MILP model. However, dilution of material extracted and deterioration of material in stockpiles is not directly incorporated into the MILP and should be reflected in the block recoveries (if blocks are used as processing units) and stockpile reclamation revenues.
6. We assume having simultaneous access to all the material within a block/cluster/panel when used as planning unit. We assume all the tonnage extracted from one planning unit has the same grades and recoveries as the weighted average of all the material within that unit.

7. The pit slope, ramps and berms are all accounted for in the block precedence calculations i.e. the defined precedence between the blocks of different benches is the only constraint restricting access to material in lower benches.
8. The processing plant recoveries are not affected by the head grade or the total tonnage delivered to the plant but only affected by the material properties.
9. The material in the stockpile does not deteriorate or lose its value by time. However, the re-handling cost of reclaiming material from the stockpile and sending them to the processing plant can be applied. Each stockpile has its upper and lower bounds on the input grade and the average output grade. The average output grade is used as the grade of material sent to the plant for revenue calculations. Therefore, it is assumed that the stockpile is homogenous in grade and material is reclaimed at stockpile average grade at all times. Recovery factors can be multiplied by the stockpiles revenues if different recoveries for stockpiles need to be considered.

3.4.2 *Modeling Approach*

The MILP model is a representation of the problem using linear and integer variables and linear equations. The objective function is a linear combination of those variables that has to be minimized or maximized. The constraints are linear inequalities of the linear and integer variables. Our model uses linear variables to make extraction and processing decisions and integer (binary) variables to control the precedence between different units of mining. The model is designed to be able to work in different resolutions of mining and processing units. The mining and processing units can be blocks, clusters and panels. Therefore, we will not point to either of these units when describing the model but use the terms mining unit and processing unit. Although blocks, clusters and panels can be used as either of mining and processing units, our assumption is that the mining unit is larger or equal to the processing unit.

3.4.3 *Objective Function*

There are various objective functions used in the literature for long-term open pit production planning. Maximizing the total production of ore or total revenue, minimizing deviations from desired production and deviations from the head grade or maximizing the utilizations of the available fleet are among the common objective functions for LTOPP. However, the most common objective function which we used in our model is maximizing the net present value of the operation. Since the LTOPP deals with long-term strategic plans of production it has to provide financial justifications for the investors and help make capital investment decisions. Higher net present value for an operation means higher return of investment and short payback period which are all attractive to investors. Therefore, the net present value is a reliable measure on the quality of the long-term production schedule.

The net present value (NPV) of a mining operation is the revenue made from selling valuable material within mining units discounted to the first period minus the costs associated with mining and processing the mining unit discounted to the same period. In addition, we include the revenue made from reclaiming material from the stockpiles and the associated re-handling cost in the objective function.

3.4.4 *Constraints*

There are numerous constraints that can be included in an LTOPP formulation based on the level of details required and the type of operation. However, we had to choose a number of constraints that apply to most mining operations to include in our model. These constraints are:

1. Mining limits
2. Processing limits
3. Mining and processing tonnage control
4. Slope and precedence control
5. Stockpile tonnage and material content
6. Grade blending
7. Reserve constraints

3.4.4.1. Mining Limits

The common mining capacity constraint is the maximum available truck-shovel fleet and the total tonnage of material they can move in each period. This limit can vary from period to period. In many operations, buying new equipment during the mining life time or salvaging older ones may change the available capacity from period to period. Therefore, we use a vector of values with one capacity per period as the input to our model.

On the other hand, enforcing a minimum on the total production of the mine is required to justify the overhead costs of the operation. The overhead cost can skyrocket if the production in the mine falls below certain levels and hence we include a minimum of the total production in each period of the operation. This minimum can also vary from period to period and can be used to force the scheduler to avoid under production even if it is contributing to the NPV.

3.4.4.2. Processing Limits

The processing plant is an important part of every mining operation. It is a significant part of capital investment and the goal of mining operation is to feed the plant. Therefore, it is important to provide enough material to fully utilize the plant. On the other hand, the plant has a limited capacity which is affected by both technical constraints and selling constraints. Thus, we include lower and upper limits on the total tonnage of material sent to the plant in each period. Since the model is a multi-destination LTOPP model, the limits can be defined for each plant separately. The same type of lower and upper bounds can be defined for the tonnage of material sent to stockpiles in each period.

3.4.4.3. Mining and Processing Tonnage Control

Since the model is designed to use two different sets of units for making mining and processing decisions, it is required to control the relationship between mining and processing. This set of constraints makes sure that the tonnage processed from each processing unit does not exceed the tonnage extracted from the corresponding mining unit.

3.4.4.4. Slope and Precedence Control

An important set of constraints in open pit production planning problem is the slope constraints. The open pit excavation schedule needs to be performed such that the slope of pit walls does not exceed a predetermined threshold. This threshold, called the safety slope, is determined based on geotechnical properties of the mine and is the maximum angle of the walls that is stable for mining. On the other hand, the order of extraction of material has to respect the accessibility constraints i.e. upper level units have to be extracted prior to the lower level to provide access to the material. These two constraints are modeled using integer variables and the precedence arcs defined between mining units of consequent benches. Figure 3.7 (a) and (b) show how this concept can be applied to a 45 degree slope and a 26.6 degree slope consequently. Figure 3.7 (c) shows how this can be applied to mining-cuts or –panel with the same 45 degree slope. It is worth mentioning that the size of the blocks and their rock type should be considered when determining the precedence arcs between them.

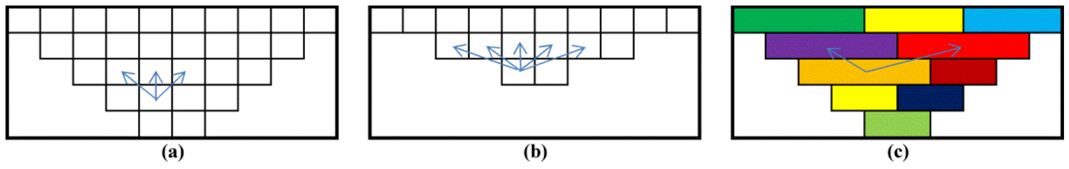


Figure 3.7. Precedence and Slope Constraints

3.4.4.5. Stockpile Tonnage and Material Content

A set of constraints is required to control the tonnage that is sent into the stockpile and the tonnage reclaimed from that stockpile. At the same time, it is needed to control the element content of the stockpile at every period since we are using an average grade for reclamation of material from the stockpile. The constraint limits the tonnage reclaimed from the stockpile in each period based on the reclamation grade and the element content of the stockpile. In other words, the maximum tonnage that can be reclaimed from the stockpile equals to tonnage that will have the equal element content to the stockpile content if reclaimed at the reclamation grade. For example, if a stockpile is defined with average reclamation grade of 4% and two units with 20,000 tonnes of 2% grade and 10,000 tonnes of 5% grade are sent to the stockpile, the total tonnage is 30,000 tonnes. However, with an average reclamation grade of 4%, the total element content of the stockpile will be 900 units, which is reclaimed at 22,500 tonnes. Therefore, reclaiming 30,000 tonnes at 4% grade from the stockpile is infeasible. This is happening due to linearization assumption and can be avoided if we calculate the actual average grade of the stockpile in every period. However, calculating the actual average grade will introduce nonlinearity to the problem and makes it very hard to solve. We will demonstrate the effects of this linearization using case-studies in the upcoming sections and document it as the limitations of the model in conclusions.

3.4.4.6. Grade Blending

Grade blending is another important constraint on the processing plant. Every plant has limitations on the grade of material it can process and needs the average grade of material to be in a reasonable range to perform efficiently. This is one of the main reasons behind using stockpiles. Reclaiming from high- or low-grade stockpiles can balance the average grade of material sent to the plant. Therefore, we define a set of constraints to control the average grade of material sent to the plant with lower and upper bounds. The average grade is simultaneously affected by the material extracted from the mine and reclaimed from the stockpiles. The stockpiles also need lower and upper bounds on the type of material they can accept. The lower and upper bounds are especially important in our model since we are using fixed average reclamation grades for the stockpiles and not calculating actual average grades. Tighter bounds on the lower and upper limits of acceptable grade at the stockpiles will decrease the linearization error in our stockpiling model.

3.4.4.7. Reserve Constraints

Reserve constraints can be defined in two ways. As mentioned earlier, our assumption in this model is that the UPL is determined prior to scheduling. Therefore, one way of defining reserve constraints is to make sure that all the material inside the UPL is extracted by the end of mine life. However, it is possible that not extracting all the material would contribute to the NPV by not extracting some of the lower value blocks.

3.4.5 *MILP Formulation*

Sets

S^m	For each mining unit m , there is a set of mining units (S^m) that have to be extracted prior to extracting mining unit m to respect slope and precedence constraints
U^m	Each mining unit m is divided into a set of processing units. U^m is the set of processing units that are contained in mining unit m
SC	Set of stockpile destinations

Indices

$d \in \{1,...,D\}$	Index for material destinations
$m \in \{1,...,M\}$	Index for mining units
$p \in \{1,...,P\}$	Index for processing units
$c \in \{1,...,C\}$	Index for processing plants
$s \in \{C+1,...,C+S\}$	Index for stockpiles
$e \in \{1,...,E\}$	Index for elements
$t \in \{1,...,T\}$	Index for scheduling periods

Parameters

D	Number of material destinations (including processing plants, waste dumps and stockpiles)
C	Number of processing plants
M	Total number mining units
P	Total number of processing units
S	Number of stockpiles
E	Number of elements in the block model
T	Number of scheduling periods
$\overline{MC}^t$	Upper bound on the mining capacity in period t
$\underline{MC}^t$	Lower bound on the mining capacity in period t

$\overline{PC_c^t}$	Maximum tonnage allowed to be sent to plant c in period t
$\underline{PC_c^t}$	Minimum tonnage allowed to be sent to plant c in period t
$\overline{G}_c^{t,e}$	Upper limit on the allowable average grade of element e at processing plant c in period t
$\underline{G}_c^{t,e}$	Lower limit on the allowable average grade of element e at processing plant c in period t
$\overline{PC_s^t}$	Maximum tonnage allowed to be sent to stockpile s in period t
$\underline{PC_s^t}$	Minimum tonnage allowed to be sent to stockpile s in period t
$\overline{G}_s^{t,e}$	Upper limit on the allowable grade of element e to be sent to stockpile s in period t
$\underline{G}_s^{t,e}$	Lower limit on the allowable grade of element e to be sent to stockpile s in period t
$G_s^{t,e}$	Average predetermined reclamation grade of element e from stockpile s in period t
$G^{t,e}$	Average calculated reclamation grade of element e from the stockpile in period t
s_m	Number of predecessors of mining unit m (number of members of S^m)
o_m	Total ore tonnage in mining unit m
w_m	Total waste tonnage in mining unit m
o_p	Total ore tonnage in processing unit p
c_m^t	Unit discounted cost of mining material from mining unit m in period t
$r_{p,c}^t$	Unit discounted revenue of sending material from processing unit p to processing destination c in period t minus the processing costs
$r_{s,c}^{t,e}$	Unit discounted revenue of processing every unit of element e from stockpile s to processing destination c in period t minus the processing and rehandling costs
$r_c^{t,e}$	Unit discounted revenue of processing every unit of element e from the stockpile to processing destination c in period t minus the processing and rehandling costs
g_p^e	Average grade of element e in processing unit p

Decision Variables

$y_m^t \in [0,1]$	Continuous decision variable representing the portion of mining unit m extracted in period t
$x_{p,d}^t \in [0,1]$	Continuous decision variable representing the portion of ore tonnage in processing unit p extracted in period t and sent to destination d
$b_m^t \in \{0,1\}$	Binary decision variable indicating if all the predecessors of mining unit m are completely extracted by or in period t

$f_{s,c}^{t} \geq 0$	Continuous decision variable representing the tonnage sent from stockpile s to processing plant c in period t
$f_{c}^{t} \geq 0$	Continuous decision variable representing the tonnage sent from the stockpile to processing plant c in period t

3.4.5.1. Original Model

The original model does not have stockpiling variables. It uses two different sets of variables for making mining and processing decisions. $y_m^t \in [0,1]$ is used to decide on the portion of each mining unit m to be extracted in each period while $x_{p,d}^t \in [0,1]$ is used to decide on the destination of material extracted from processing unit p. The objective function is to maximize the net present value of the operation over the mine life.

Objective Function

$$\max \sum_{t=1}^{T}\left(\sum_{p=1}^{P}\sum_{c=1}^{C}(r_{p,c}^{t} \times o_p \times x_{p,c}^{t}) - \sum_{m=1}^{M}(c_m^t \times (o_m + w_m) \times y_m^t)\right) \tag{3.18}$$

Constraints

$$\underline{MC}^{t} \leq \sum_{m=1}^{M}\left((o_m + w_m) \times y_m^t\right) \leq \overline{MC}^{t} \qquad \forall t \in \{1,...,T\} \tag{3.19}$$

$$\underline{PC}_c^{t} \leq \sum_{p=1}^{P}\left(o_p \times x_{p,d}^{t}\right) \leq \overline{PC}_c^{t} \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, d = c \tag{3.20}$$

$$\sum_{p \in U^m}\sum_{d=1}^{D}\left(o_p \times x_{p,d}^{t}\right) \leq (o_m + w_m) \times y_m^t \qquad \forall t \in \{1,...,T\}, \forall m \in \{1,...,M\} \tag{3.21}$$

$$\underline{G}_c^{t,e} \leq \frac{\sum_{p=1}^{P}\left(o_p \times g_p^e \times x_{p,c}^{t}\right)}{\sum_{p=1}^{P}\left(o_p \times x_{p,c}^{t}\right)} \leq \overline{G}_c^{t,e} \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, \forall e \in \{1,...,E\} \tag{3.22}$$

$$\sum_{t=1}^{T} y_m^t = 1 \qquad \forall m \in \{1,...,M\} \qquad (3.23)$$

$$\sum_{d'=1}^{D} \sum_{t=1}^{T} x_{p,d'}^t = 1 \qquad \forall p \in \{1,...,P\} \qquad (3.24)$$

$$\sum_{i=1}^{t} y_m^i \le b_m^t \qquad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T\} \qquad (3.25)$$

$$s_m \times b_m^t \le \sum_{i \in S^m} \sum_{j=1}^{t} y_i^j \qquad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T\} \qquad (3.26)$$

$$b_m^t \le b_m^{t+1} \qquad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T-1\} \qquad (3.27)$$

The objective function in equation (3.18) consists of 2 major parts summed over mine life. The first part is the total revenue made from sending portions of processing units to various destinations, processing them and selling the final product extracted. The second part is the mining cost of the material scheduled to be extracted in each period. With respect to costs and revenues mentioned here it is worth noting that:

1. c_m^t is the unit cost of mining material in mining unit m as waste and sending them to the waste dump. If mining unit m consists of different rock types with different mining costs, the weighted average of the costs is used as the unit cost of mining. In addition, this mining cost is discounted to period 1 of the mine life and there is no need for including the discount rate in the model. The discounted mining cost can be calculated using equation (3.28) where i is the discount rate.

$$c_m^t = \frac{1}{(1+i)^t} \times (\text{cost of mining}) \qquad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T\} \qquad (3.28)$$

2. $r_{p,c}^{t}$ is the unit discounted revenue of processing material from processing unit p in processing plant c in period t. The processing costs, selling costs or extra mining cost due to mining material as ore is included in this revenue. However, the reference mining cost is not included in this calculation. $r_{p,c}^{t}$ is calculated prior to formulating the MILP using equation (3.29).

$$r_{p,c}^{t}=\frac{1}{(1+i)^{t}}\times\left(\begin{array}{l}\text{revenue from processing}\\ \text{- processing costs}\\ \text{- selling costs}\\ \text{-}\left(\begin{array}{l}\text{cost of mining as ore}\\ \text{- cost of mining as waste}\end{array}\right)\end{array}\right)\quad \forall p\in\{1,...,P\},\forall t\in\{1,...,T\},\forall c\in\{1,...,C\} \tag{3.29}$$

3. If the mining operation has more than one waste dump with different mining costs or different rock types, the waste dumps can be defined as a dummy processing destination with negative revenue. For example, if a specific rock type is sent to a waste dump with higher mining and haulage cost, we can define a destination c' with $r_{p,c'}^{t}$ calculated as in equation (3.30).

$$r_{p,c'}^{t}=\frac{1}{(1+i)^{t}}\times\left(\text{-}\left(\begin{array}{l}\text{cost of mining}\\ \text{and haulage to c}'\\ \text{- cost of mining as waste}\end{array}\right)\right)\quad \forall p\in\{1,...,P\},\forall t\in\{1,...,T\} \tag{3.30}$$

Equation (3.19) is the mining capacity equation. It accounts for the lower and upper bounds on the total material mined in each period. Equation (3.20) is the processing capacity constraint and makes sure that the total tonnage of extracted from mine and sent to the plant in each period is within the acceptable range. Equation (3.21) controls the relationship between the tonnage mined from mining unit and the tonnage processed from the processing units within that mining unit. Equation (3.22) is the grade blending constraint. Although it seems to be a non-linear equation we can rearrange the equation to make it linear by multiplying both sides by $\sum_{p=1}^{P}\left(o_p \times x_{p,d}^{t}\right)$. Accordingly, we can rearrange the equation (3.22) into equations (3.31) and (3.32) for the lower and upper bound on the head grade respectively.

$$0 \le \sum_{p=1}^{P}\left(o_p \times \left(g_p^e - \underline{G}_c^{t,e}\right) \times x_{p,c}^{t}\right) \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, \forall e \in \{1,...,E\} \qquad (3.31)$$

$$\sum_{p=1}^{P}\left(o_p \times \left(g_p^e - \overline{G}_c^{t,e}\right) \times x_{p,c}^{t}\right) \le 0 \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, \forall e \in \{1,...,E\} \qquad (3.32)$$

Equation (3.23) is called the reserve constraint and makes sure all the material in the UPL is extracted in the mine life. However, it is possible to let the MILP decide whether to extract all the material from the UPL or leave some if it increases the NPV. For the latter case, equation (3.23) is replaced with the equation (3.33).

$$\sum_{t=1}^{T} y_m^t \le 1 \qquad \forall m \in \{1,...,M\} \qquad (3.33)$$

Binary integer variable $b_m^t \in \{0,1\}$ is introduced to control the mining precedence and to satisfy the slope constraints. Equation (3.25) ensures that the mining of unit m in period t does not happen unless variable b_m^t is equal to 1. Although, it is enough to use $y_m^t \le b_m^t$ to prevent mining unless variable b_m^t is equal to 1, tests have shown that $\sum_{i=1}^{t} y_m^i \le b_m^t$ is a stronger formulation. Equation (3.26) is the precedence constraint in which variable b_m^t is forced to be equal to 0 until all the predecessor mining units have been completely extracted. Equation (3.27) makes sure that the binary variable is equal to 1 if it has been set to 1 in an earlier period.

3.4.5.2. Non-linear Model

We can modify the original LTOPP model to add stockpiling to the operation. To do so, we introduced a new set of variables $f_c^t \geq 0$ to decide on the tonnage of material reclaimed from the stockpile in each period and sent to plant. We also added the stockpile to the material destinations and updated the objective function and constraints as follows. The objective function is updated by adding the profit made from reclaiming material from the stockpile and sending to the plant. On the other hand, a new destination (c') is added to send material from the processing units for the stockpile. Since no revenue is made from sending material to the stockpile $r_{p,c'}^t$ can be used to add any extra costs inflected by sending material to the stockpile to the objective function. $G^{t,e}$ is the variable holding the reclamation grade of element e in period t and $r_c^{t,e}$ is the unit discounted revenue of processing one unit of element e from stockpile in processing destination c in period t minus the processing and rehandling costs. $G^{t,e}$ is calculated using equation (3.38) based on the material sent to the stockpile and the tonnage and grade reclaimed from the stockpile. It is assumed that material is stockpiled at the end of the period and reclaimed at the beginning of the period. Equation (3.38) is a non-linear equation and makes the problem hard to solve. Equation (3.20) is replaced with equation (3.35) to account for the tonnage reclaimed from the stockpile and sent to each destination. Equation (3.37) ensures that the summation of tonnages reclaimed from the stockpile from the first period to the current period does not exceed the summation of tonnages sent to the stockpile by the current period. The constraint is formed in this way to avoid defining variables for the tonnage of material in the stockpile in each period. Moreover, the destination head grade constraint (equation (3.22)) has to replaced by equation (3.36) to account for the tonnage and grade of material sent from the stockpile to the destination.

Objective Function

$$\max \sum_{t=1}^{T}\left(\sum_{p=1}^{P}\sum_{d=1}^{C}(r_{p,c}^{t}\times o_{p}\times x_{p,d}^{t})-\sum_{m=1}^{M}(c_{m}^{t}\times\left(o_{m}+w_{m}\right)\times y_{m}^{t})+\sum_{e=1}^{E}\sum_{c=1}^{C}\left(f_{c}^{t}\times G^{t,e}\times r_{c}^{t,e}\right)\right) \tag{3.34}$$

Constraints

$$\underline{PC}_{c}^{t}\leq\sum_{p=1}^{P}\left(o_{p}\times x_{p,c}^{t}\right)+f_{c}^{t}\leq\overline{PC}_{c}^{t} \qquad \forall t\in\{1,...,T\},\forall c\in\{1,...,C\} \tag{3.35}$$

$$\underline{G}_{c}^{t,e}\leq\frac{\sum_{p=1}^{P}\left(o_{p}\times g_{p}^{e}\times x_{p,c}^{t}\right)+\left(f_{c}^{t}\times G^{t,e}\right)}{\sum_{p=1}^{P}\left(o_{p}\times x_{p,c}^{t}\right)+f_{c}^{t}}\leq\overline{G}_{c}^{t,e} \qquad \forall t\in\{1,...,T\},\forall c\in\{1,...,C\}, \forall e\in\{1,...,E\} \tag{3.36}$$

$$\sum_{i=1}^{t}\sum_{c=1}^{C}f_{c}^{i}\leq\sum_{i=1}^{t-1}\sum_{p=1}^{P}\left(o_{p}\times x_{p,c'}^{i}\right) \qquad \forall t\in\{2,...,T\} \tag{3.37}$$

$$G^{t,e}=\frac{\sum_{p=1}^{P}\left(o_{p}\times g_{p}^{e}\times x_{p,c'}^{t}\right)-\sum_{t'=1}^{t-1}\sum_{c=1}^{C}f_{c}^{t'}\times G^{t',e}}{\sum_{p=1}^{P}\left(o_{p}\times x_{p,c'}^{t}\right)+\sum_{t'=1}^{t-1}\sum_{c=1}^{C}f_{c}^{t'}} \qquad \forall t\in\{1,...,T\},\forall e\in\{1,...,E\} \tag{3.38}$$

Note that r_{c}^{t} is the unit discounted revenue of reclaiming material from stockpile and sending to processing plant c in period t. r_{c}^{t} accounts for the revenue generated from processing material minus the costs of processing and selling, and reclamation. Therefore, r_{c}^{t} can be calculated using equation (3.39).

$$r_c^t = \frac{1}{(1+i)^t} \times \begin{pmatrix} \text{revenue from processing} \\ \text{- processing costs} \\ \text{- selling costs} \\ \text{- rehandling costs} \end{pmatrix} \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\} \qquad (3.39)$$

3.4.5.3. Linearized Model

As mentioned earlier, calculating the element grade of material in the stockpile for each period results in a non-linear model with non-linear constraints. Therefore, we modified the model based on a linearization technique inspired by piecewise linearization to avoid non-linearity. In order to avoid non-linearity, we consider s stockpiles in the model with different ranges of element grades. Each stockpile has lower and upper bounds on the average grade of material sent to the stockpile and an average reclamation grade. Accordingly, we can replace $G^{t,e}$ with a predetermined grade ($G_s^{t,e}$) for each stockpile and rewrite the objective function and constraints.

Objective Function

$$\max \sum_{t=1}^{T} \left(\sum_{p=1}^{P} \sum_{d=1}^{C} (r_{p,c}^t \times o_p \times x_{p,d}^t) - \sum_{m=1}^{M} (c_m^t \times (o_m + w_m) \times y_m^t) + \sum_{s=1}^{S} \sum_{c=1}^{C} \sum_{e=1}^{E} (r_{s,c}^{t,e} \times G_s^{t,e} \times f_{s,c}^t) \right) \qquad (3.40)$$

Constraints

$$\underline{MC}^t \le \sum_{m=1}^{M} \left((o_m + w_m) \times y_m^t \right) \le \overline{MC}^t \qquad \forall t \in \{1,...,T\} \qquad (3.41)$$

$$\underline{PC}_c^t \le \sum_{p=1}^{P} \left(o_p \times x_{p,c}^t \right) + \sum_{s=1}^{S} f_{s,c}^t \le \overline{PC}_c^t \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\} \qquad (3.42)$$

$$\sum_{p \in U^m} \sum_{d=1}^{D} \left(o_p \times x_{p,d}^t \right) \le (o_m + w_m) \times y_m^t \qquad \forall t \in \{1,...,T\}, \forall m \in \{1,...,M\} \qquad (3.43)$$

$$\underline{G}_c^{t,e} \le \frac{\sum_{p=1}^{P}\left(o_p \times g_p^e \times x_{p,c}^t\right) + \sum_{s=C+1}^{C+S}\left(f_{s,c}^t \times G_s^{t,e}\right)}{\sum_{p=1}^{P}\left(o_p \times x_{p,c}^t\right) + \sum_{s=C+1}^{C+S} f_{s,c}^t} \le \overline{G}_c^{t,e} \quad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, \forall e \in \{1,...,E\} \tag{3.44}$$

$$\underline{G}_d^{t,e} \le \frac{\sum_{p=1}^{P}\left(o_p \times g_p^e \times x_{p,d}^t\right)}{\sum_{p=1}^{P}\left(o_p \times x_{p,d}^t\right)} \le \overline{G}_d^{t,e} \quad \forall t \in \{1,...,T\}, \forall e \in \{1,...,E\}, \forall d \in SC \mid d = C+s \tag{3.45}$$

$$\sum_{t=1}^{T} y_m^t = 1 \quad \forall m \in \{1,...,M\} \tag{3.46}$$

$$\sum_{d'=1}^{D}\sum_{t=1}^{T} x_{p,d'}^t = 1 \quad \forall p \in \{1,...,P\} \tag{3.47}$$

$$\sum_{i=1}^{t} y_m^i \le b_m^t \quad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T\} \tag{3.48}$$

$$s_m \times b_m^t \le \sum_{i \in S^m}\sum_{j=1}^{t} y_i^j \quad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T\} \tag{3.49}$$

$$b_m^t \le b_m^{t+1} \quad \forall m \in \{1,...,M\}, \forall t \in \{1,...,T-1\} \tag{3.50}$$

$$\sum_{i=1}^{t}\sum_{c=1}^{C} f_{s,c}^i \le \sum_{i=1}^{t-1}\sum_{p=1}^{P}\left(o_p \times x_{p,d}^i\right) \quad \forall s \in \{C+1,...,C+S\}, \forall t \in \{2,...,T\}, \forall d \in SC \mid d = C+s \tag{3.51}$$

$$\sum_{i=1}^{t}\sum_{c=1}^{C} G_s^{t,e} \times f_{s,c}^i \le \sum_{i=1}^{t-1}\sum_{p=1}^{P}\left(o_p \times g_p^e \times x_{p,d}^i\right) \quad \forall s \in \{C+1,...,C+S\}, \forall t \in \{2,...,T\}, \forall e \in \{1,...,E\}, \forall d \in SC \mid d = C+s \tag{3.52}$$

Equation (3.51) controls the tonnage sent to the stockpile versus the tonnage extracted from the stockpile. It is assumed that sending material to the stockpile happens at the end of the period and reclaiming from the stockpile happens at the beginning of each period. Therefore, the tonnage reclaimed from the stockpile in each period has to be less than or equal to the tonnage sent to stockpile in the previous periods. The constraint is defined using the cumulative sent and reclaimed tonnages to avoid introducing extra variables for keeping track of the stockpile inventory. On the other hand, equation (3.52) controls the material content of stockpile to make sure that the average reclamation grade assumption does not violate the feasibility of the stockpiling strategy. In order to keep the model linear, we rewrite equations (3.44) and replace them with equations (3.53) and (3.54).

$$0 \le \sum_{p=1}^{P}\left(o_p \times \left(g_p^e - \underline{G}_c^{t,e}\right) \times x_{p,c}^t\right) + \sum_{s=C+1}^{C+S}\left(f_{s,c}^t \times \left(G_s^{t,e} - \underline{G}_c^{t,e}\right)\right) \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, \forall e \in \{1,...,E\} \tag{3.53}$$

$$\sum_{p=1}^{P}\left(o_p \times \left(g_p^e - \overline{G}_c^{t,e}\right) \times x_{p,c}^t\right) + \sum_{s=C+1}^{C+S}\left(f_{s,c}^t \times \left(G_s^{t,e} - \overline{G}_c^{t,e}\right)\right) \le 0 \qquad \forall t \in \{1,...,T\}, \forall c \in \{1,...,C\}, \forall e \in \{1,...,E\} \tag{3.54}$$

3.5. Preprocessing

In order to improve the performance of the solution procedure, we implemented a preprocessing stage. In this stage, we formulate the MILP and determine the variables whose values can be determined prior to solving the model. Five types of variable and constraint elimination techniques are implemented from which three do not affect the optimal solution.

3.5.1 *Removing Unnecessary Variables*

The MILP is formulated to use two different sets of variables for making extraction and processing decisions. Despite mining units that all require variables to determine their period of extraction, not all the processing units need a variable to determine their destination. Since the clustering stage tries to create processing units that are homogenous in rock-type, many of them are completely made of one rock type. Therefore, the destination of processing units with all waste rock-type can be determined prior to solving the model. Especially, we can remove all the $x_{p,d}^{t}$ variables for waste units when we do not distinguish between waste dumps. Moreover, the processing destinations can be limited to a specific rock-type based on technical requirements. Thus, we can eliminate all the $x_{p,d}^{t}$ variables based on the possibility of sending one rock-type to one destination in cases where the processing unit only contains that rock-type without losing optimality.

3.5.2 *Predecessor Cone*

Another variable elimination technique that does not sacrifice optimality is borrowed from Bley et al. (2010). In this technique, the cumulative tonnage of material that has to be removed prior to accessing a mining unit is compared against the cumulative available mining capacity. For example, if the tonnage of material that has to be removed prior to accessing mining unit m is greater than the summation of the mining capacities in the first two periods, y_{m}^{t} variables for the first two periods can be set to zero without affecting the optimal solution. Consequently, all the $x_{p,d}^{t}$ variables corresponding to the processing units within that mining unit can be set to zero for the first two periods. We call this the predecessor cone method referring to the cone of material that has to be extracted prior to accessing each mining unit.

3.5.3 *Successor Cone*

As mentioned earlier, two different types of reserve constraints are defined for this project. The first type ensures all the material within the final pit is extracted some time during the mine life whereas the second type lets the optimizer choose what to extract and what to leave in the pit. In case of the former type, we can form a successor cone for each mining unit similar to the predecessor cone but based on the tonnage material that can only be extracted after that mining unit is extracted. Similar to the predecessor cone example, if the tonnage of material that can only be extracted after extraction of mining unit m exceeds the summation of mining capacities for the last two periods, we can be sure that mining unit m has to be extracted prior to $T-2$ and consequently set y_m^t variables for zero for the last two periods. The two-cone concept illustrated in Figure 3.8 can also be found in the Simulated Annealing approach by Kumral and Dowd (2005).

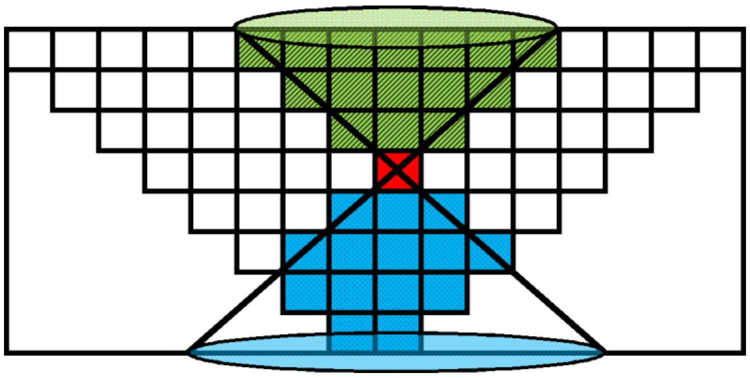


Figure 3.8. Predecessor and Successor Cones

3.5.4 *Final Pit Limits*

The common approach in open pit production planning is to determine the optimum pit prior to the scheduling stage. On the other hand, Caccetta and Hill (2003) prove that including blocks outside the final pit does not contribute to the NPV of the operation if all the elements of the coefficient matrix are non-negative. It means that in cases where the blending constraints are not binding, we can eliminate the blocks outside the final pit without sacrificing the optimality of the solution. Although including stockpiles and blending constraints in the MILP disproves Caccetta and Hill's assumption, we limit our model to final pit limits as it is the common practice and significantly reduces the size of the model.

3.5.5 *Using Initial Solution*

Another way of reducing the problem size is to use an initial solution and limit the search space to initial solution neighborhood. This becomes handy for very large problems where an initial solution is obtained through other means such as heuristic algorithms, aggregated units and aggregated periods. For example, we can solve the MILP for a 30 year production plan in six five-year periods and use the solution obtained to limit the extraction periods of units to reduce problem size. To do so, we accumulate the mining and processing capacities for every five year into one period and solve the model. If a unit is scheduled to be extracted in the second aggregated period, we can eliminate variables corresponding to years 1-5 and 11-30 from the original model. Although this neighborhood limitation affects the optimal solution it can be helpful when the size of the model is extremely large.

3.5.6 *Removing Unnecessary Constraints*

Along with removing unnecessary variables we can identify and remove non-binding and redundant constraints from the MILP before solving it with the optimizer. These constraints include:

1. Processing capacity and head grade constraints for pre-stripping periods (equations (3.20) and (3.22))

2. Mining and processing units tonnage control, precedence constraints and binary constraints for the periods and units that are predetermined based on predecessor and successor cone techniques (equations (3.21), (3.25), (3.26) and (3.27))

3.6. Summary and Conclusion

In this chapter, we presented a clustering algorithm along with an MILP formulation for solving the long-term open pit production planning problem. The clustering algorithm creates aggregates of blocks for short- to long-term planning. The aggregates created via clustering along with other units of planning can then be used in MILP formulation to maximize the NPV of the operation subject to technical and operational constraints. Moreover, the MILP formulation simultaneously provides optimal stockpiling strategies for the long-term production plan.

The clustering algorithm presented in this chapter is a modification of the agglomerative hierarchical clustering technique that is designed to serve in the mine planning area. We showed how the algorithm works by calculating the similarity matrix and merging similar blocks in an iterative manner. Moreover, we introduced a Tabu search technique that modifies the clustering schemes to reduce the number of precedence arcs between clusters without losing intra-cluster similarity. Next, we added a post-processing step that smoothens the shapes of the generated clusters to provide clusters with mineable shapes. In addition to the similarities, we modified our clustering algorithm to account for mining direction and predetermined boundaries while forming the clusters. Afterwards, we implemented a variation of k-means clustering to be able to compare against our hierarchical approach in the upcoming chapters. Finally, we introduced four evaluation measures to be able to compare different clustering outcomes and decide on the proper settings to use.

The MILP formulation presented in this chapter is built upon the current state of the art models used by various researchers in the area. In this chapter, we first reviewed the assumptions we made for formulating the MILP. Then, we presented the MILP formulation with the objective and constraints. We explained the details defining variables and formulating the objective function and constraints. Our MILP formulation maximizes the NPV of the operations subject to mining, processing, slope and precedence, grade blending, and reserve constraints. Moreover, we added stockpiling to our model to simultaneously determine the optimal stockpiling strategy while looking for the optimum production schedule. We presented the stockpiling model with quadratic objective function and constraints and used linearization to make the problem solvable with the available solvers.

Finally, in this chapter, we presented six different techniques for reducing the size of the MILP problem prior to using a solver to decrease the computational time required for solving the model. These techniques are based upon the structure of the problem as well as preprocessing and initial solutions obtained prior to solving the model.

VERIFICATION, IMPLEMENTATION AND DISCUSSION OF RESULTS

4. VERIFICATION, IMPLEMENTATION AND DISCUSSION OF RESULTS

4.1. Introduction

In this chapter, we implement the clustering algorithms and mathematical formulations on a standard test dataset called Marvin (Espinoza, et al., 2013) in order to illustrate how the algorithms work and how clustering affects the mathematical formulation and its run time. Real-size case-studies and the performance evaluations of the algorithms on real-size datasets can be found in the following sections.

4.2. Marvin Dataset

Marvin is a well-known test dataset used as the demo dataset in Whittle™ (GEOVIA, 2014b) mine planning software and presented as a standard dataset in MineLib (Espinoza, et al., 2013). Marvin dataset consists of 53,271 blocks with four different rock types and two element grades. Each block is 30 meters in all dimensions. The main rock types are Mixed (MX), Oxide (OX) and Primary (PM) along with the undefined waste denoted here with UND. Marvin mine is a gold and copper mine with a thin layer of overburden which makes planning easier and provides access to ore in the very first periods of extraction.

As mentioned earlier, Marvin dataset exists in both Whittle™ (GEOVIA, 2014b) and MineLib (Espinoza, et al., 2013). Despite using the same cost and profit parameters the optimum pit determined by Whittle™ (GEOVIA, 2014b) is different from the optimum pit determined in MineLib (Espinoza, et al., 2013). The former has 9,381 blocks (575 million tons) in the final pit compared to the latter with 8,516 blocks (527 million tons) in the final pit. We have used the outputs from Whittle™ (GEOVIA, 2014b) in our case studies to be able to compare our results to a commercial software used by many companies in the mining industry.

In all cases, the block economics are calculated based on a mining cost of $1.5 per ton and a processing cost of $6.25 per ton for all rock types. The gold and copper recoveries in the mill, selling costs and prices are summarized in Table 4.1. The histograms and descriptive statistics of the two elements can be found in Figure 4.1, Figure 4.2 and Table 4.2. Sample plan views of rock-type and grade distribution within the final pit are presented in Figure 4.3 to Figure 4.8. For the purpose of calculating clustering measures, we use the initial destinations determined by Milawa NPV in Whittle™ (GEOVIA, 2014b) as the destination of blocks in the clustering step.

Table 4.1. Marvin Element Economics

Element	Unit	Recovery	Selling Cost	Price
Au	gram	0.6	4.80	38.6
Cu	%m	0.8	11.03	33.1

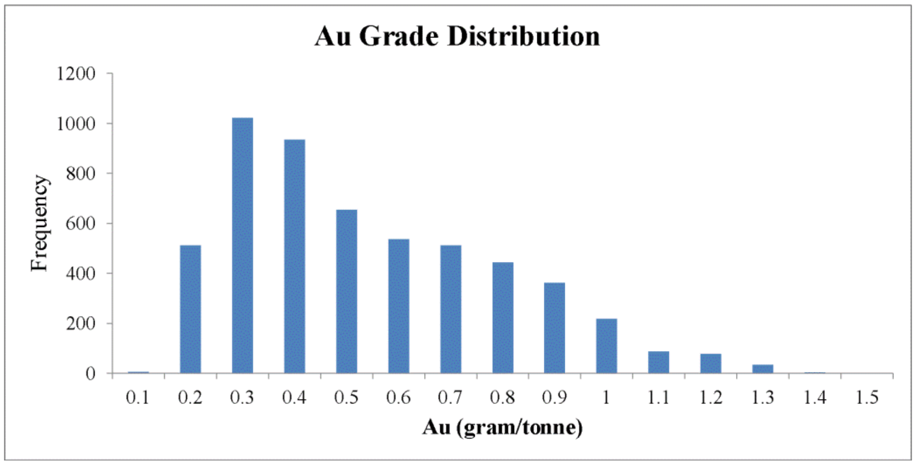


Figure 4.1. Original Au Grade Distribution

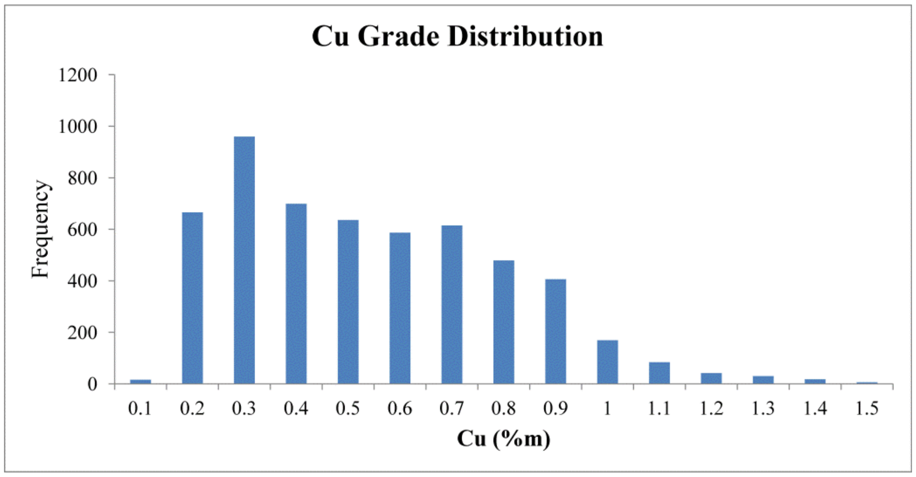


Figure 4.2. Original Cu Grade Distribution

Table 4.2. Original Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	5,413	5,413
Mean	0.494	0.493
Median	0.433	0.456
Standard Deviation	0.257	0.259
Sample Variance	0.066	0.067
Kurtosis	-0.243	-0.196
Skewness	0.716	0.611
Range	1.342	1.387
Minimum	0.075	0.077
Maximum	1.417	1.464

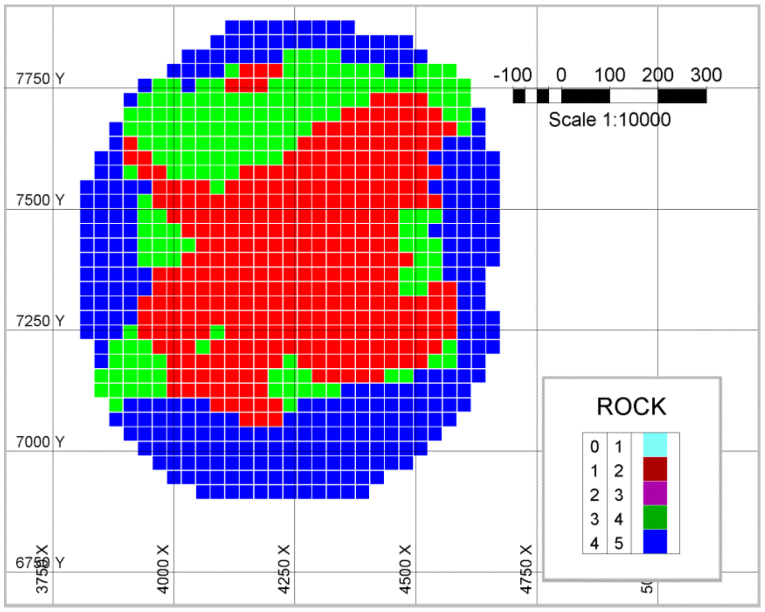


Figure 4.3. Rock-type Distribution Plan View at 600m Elevation

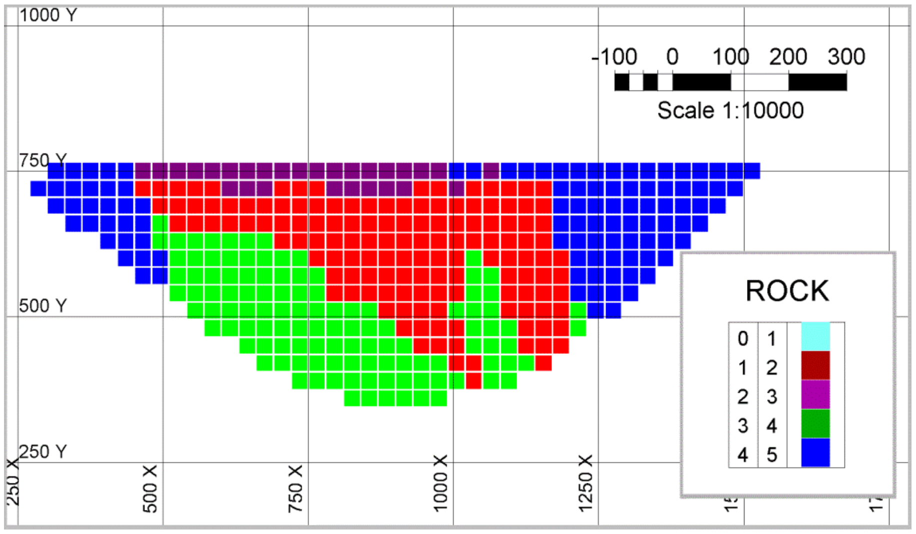


Figure 4.4. Rock-type Distribution at 4100m Easting

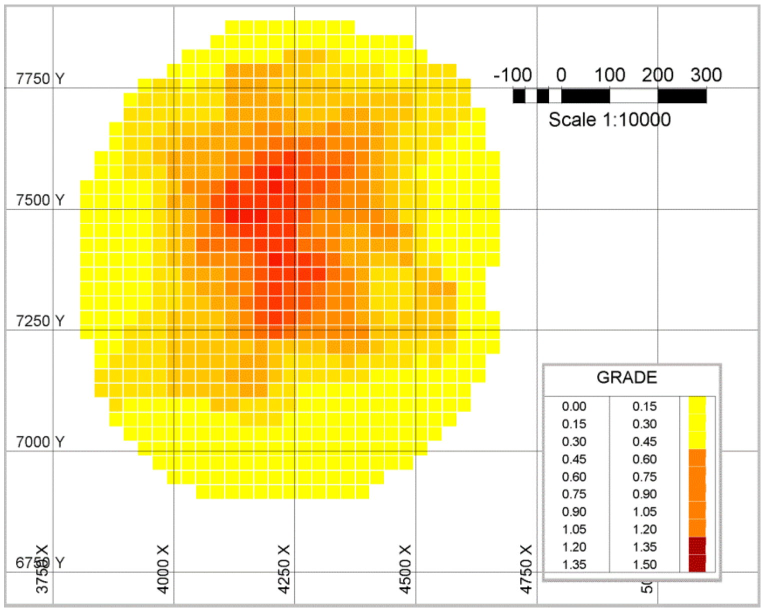


Figure 4.5. Gold Grade Distribution Plan View at 600m Elevation

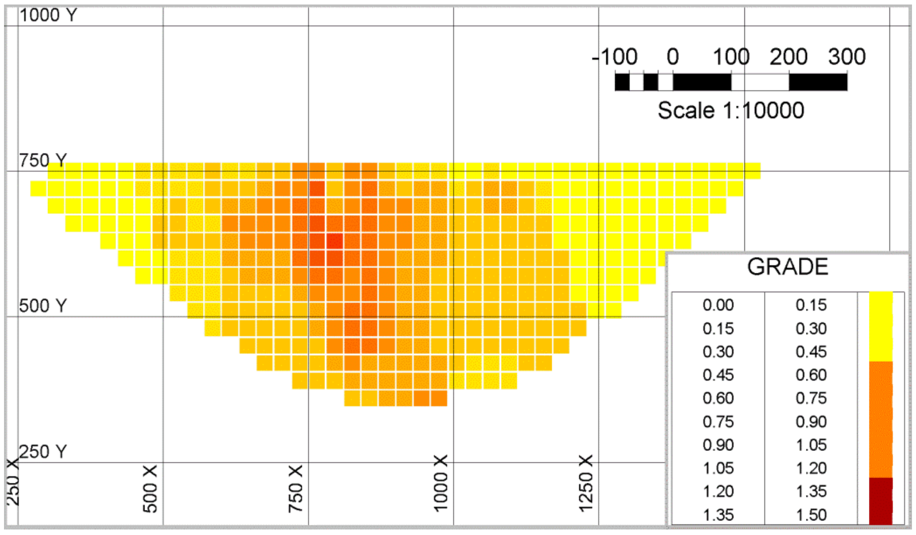


Figure 4.6. Gold Grade Distribution at 4100m Easting

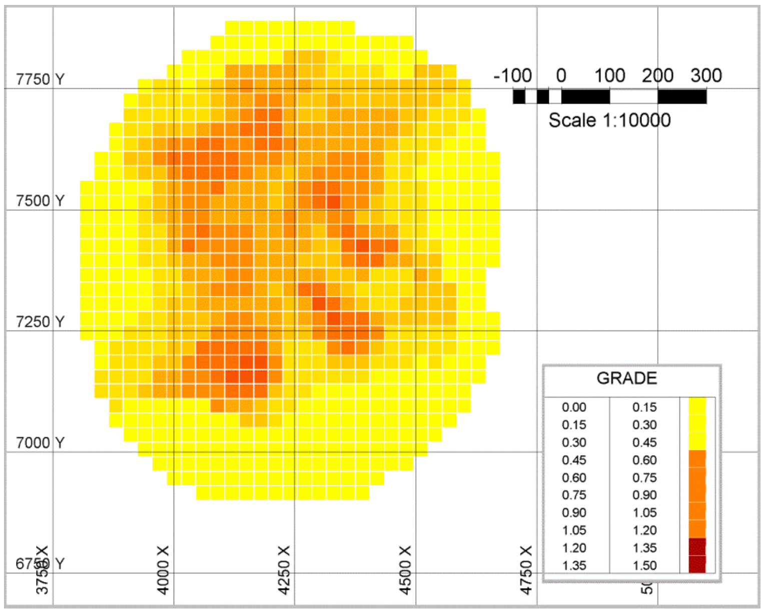


Figure 4.7. Copper Grade Distribution Plan View at 600m Elevation

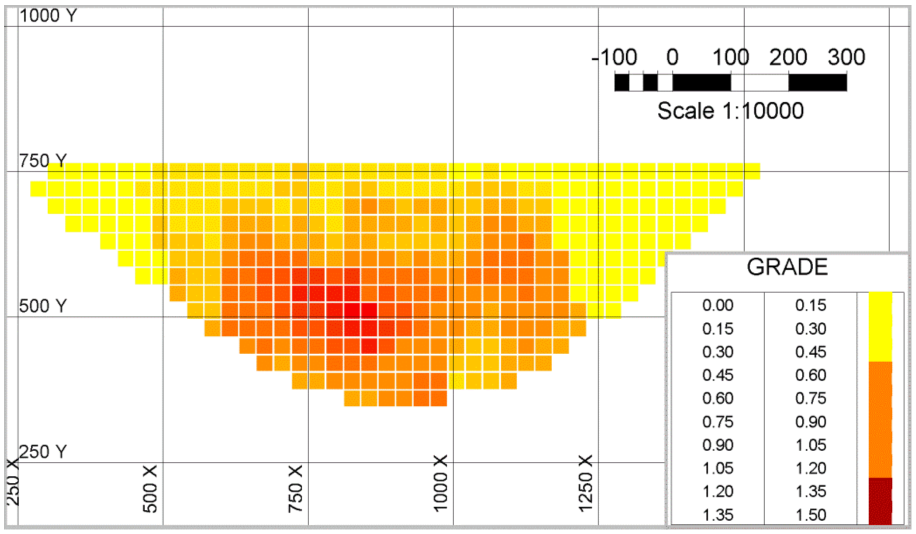


Figure 4.8. Copper Grade Distribution at 4100m Easting

4.3. Hierarchical Clustering

In this section, we study the effects of different clustering parameters on the clustering results based on the defined quality measures as in section 3.3.7. We run multiple scenarios with the clustering algorithm on the Marvin dataset by changing the clustering parameters. We use different weights and penalties for the similarity measures and different control parameters on the cluster size and shape. Our goal is not to tune the parameters and find out the best clustering scheme but to show how the parameters affect the clustering outcome. The clustering scenarios to be tested on Marvin dataset in this section are summarized in Table 4.3. Better tuned and more practical case-studies will follow in sections 4.8 to 4.10.

Table 4.3. Clustering Scenarios

ID	W_D	W_G	r	c	Size			Shape Refin.	With. Bound.	Min. Dir.
					Avg.	Max	Min			
D	1	0	1	1	15	5000	0	0	-	-
G	0	1	1	1	15	5000	0	0	-	-
R	0	0	0.1	1	15	5000	0	0	-	-
DG	0.5	0.5	1	1	15	5000	0	0	-	-
DR	0.5	0	0.5	1	15	5000	0	0	-	-
DGR	0.5	0.5	0.5	1	15	5000	0	0	-	-
TS	0.5	0	0.5	0.5	15	5000	0	0	-	-
DR30	0.5	0	0.5	1	30	5000	0	0	-	-
DR60	0.5	0	0.5	1	60	5000	0	0	-	-
DR90	0.5	0	0.5	1	90	5000	0	0	-	-
DRM	0.5	0	0.5	1	15	20	10	1	-	-
SR	0.5	0	0.5	1	15	20	5	3	-	-
BND	0.5	0	0.5	1	15	20	0	3	PB	-
DV	0.4	0	0.2	1	50	60	0	0	-	V
DS	0.4	0	0.2	1	50	60	0	0	-	S
DSR	0.4	0	0.2	1	50	60	5	3	-	S

4.3.1 *Distance Similarity*

The simplest way of grouping blocks together is to use distance as the sole similarity index and group blocks based on their coordinates. In this sense, the deposit characteristics are ignored and the generated clusters would lack homogeneity in grade or rock-type. Although, blocks closer to each other are usually more similar than blocks far away, this method does not distinguish between different rock-types and grades and cannot be used to draw the boundaries between ore and waste. The clustering quality measures for having distance as the sole similarity factor is presented in Table 4.6. Figure 4.9 and Figure 4.10

present the histogram of gold and copper average grades of the generated clusters. The algorithm settings for this run are presented in Table 4.4.

Table 4.4. Clustering Parameters

Parameter	Value
Distance Weight	1
Grade Weight	0
Rock-type Penalty	1
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

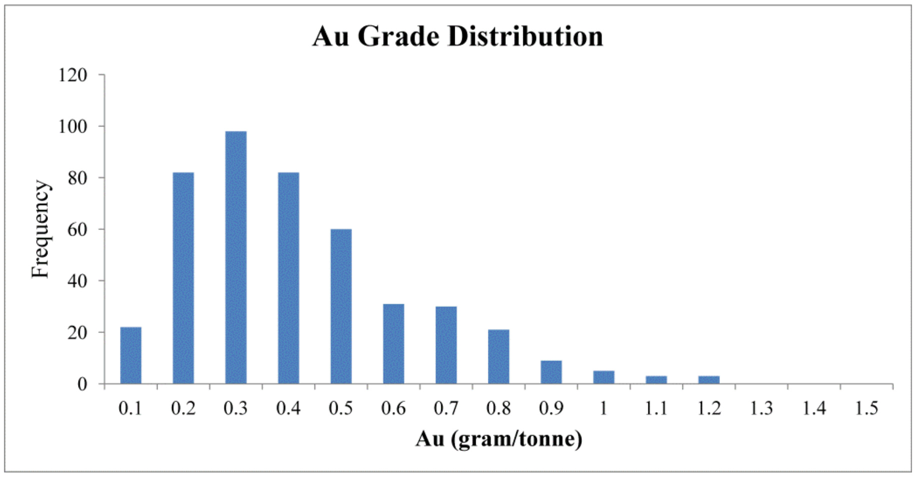


Figure 4.9. Clustered Au Grade Distribution

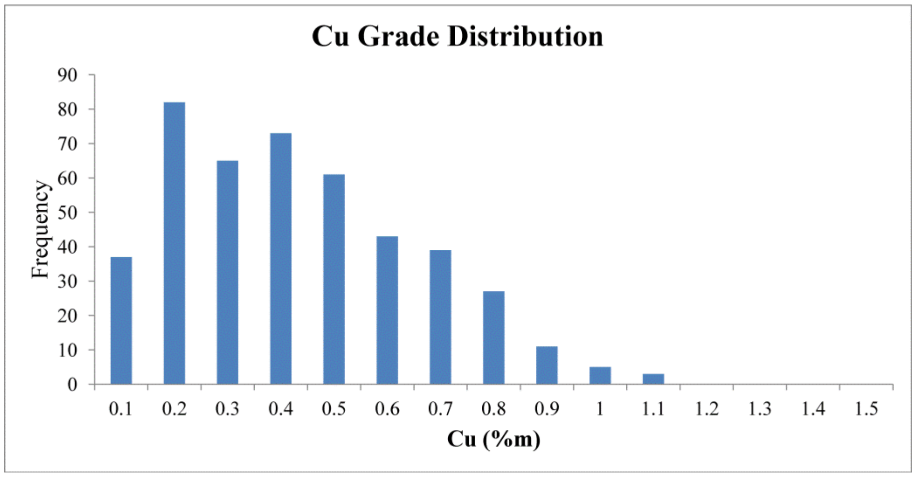


Figure 4.10. Clustered Cu Grade Distribution

Table 4.5. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	446	446
Mean	0.368	0.381
Median	0.332	0.344
Standard Deviation	0.217	0.225
Sample Variance	0.047	0.051
Kurtosis	0.825	-0.391
Skewness	0.983	0.557
Range	1.135	1.058
Minimum	0.016	0.014
Maximum	1.151	1.072

Table 4.6. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	382
Rock Unity (%)	85.8	17.1
DDF (%)	92.6	13.6
Au CV (%)	48.9	42.6
Cu CV (%)	47.7	41.3

Despite not contributing to the homogeneity of the generated clusters, we usually include distance in similarity index in order to generate clusters more uniform in shape and size. Figure 4.11 illustrates the clustering results for a sample plan view.

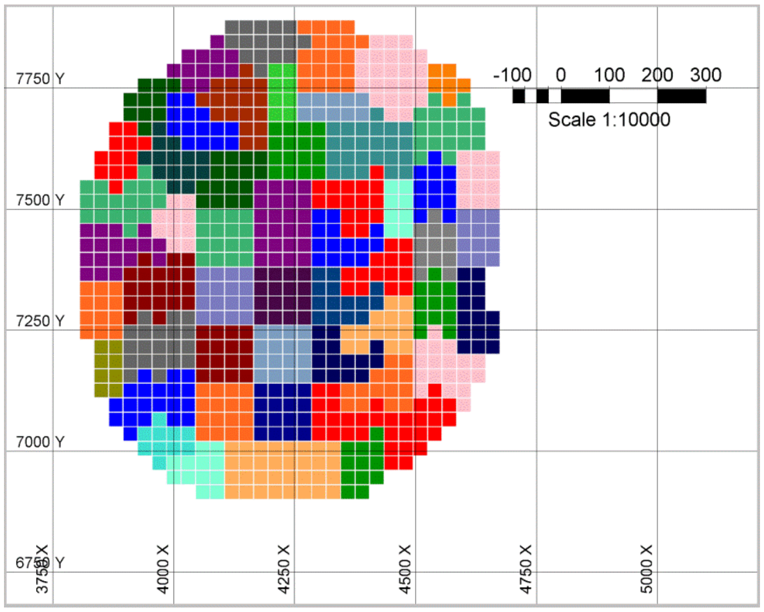


Figure 4.11. Cluster IDs Plan View at 600m Elevation

4.3.2 *Grade Control*

In this section, we only used element grade as the deciding factor in calculating similarities between blocks. Therefore, we set other weights to zero and penalties to one to explore the effects of having grade as the sole similarity index. Grade of gold is used as the major element grade in clustering. Other settings are presented in Table 4.7. As presented in Table 4.9 average gold grade variation in clusters is lower than the previous case. Moreover, the variation in copper grade is decreased due to correlations between the two grades. However, the shape and size of the generated clusters are out of control and variable (Figure 4.14). The clustered grade distribution of the elements are presented in Figure 4.12 and Figure 4.13.

Table 4.7. Clustering Parameters

Parameter	Value
Distance Weight	0
Grade Weight	1
Rock-type Penalty	1
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

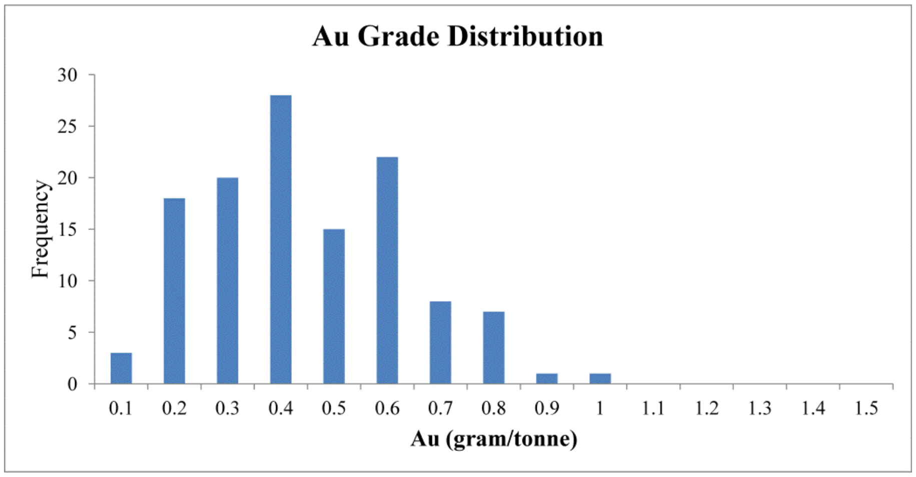


Figure 4.12. Clustered Au Grade Distribution

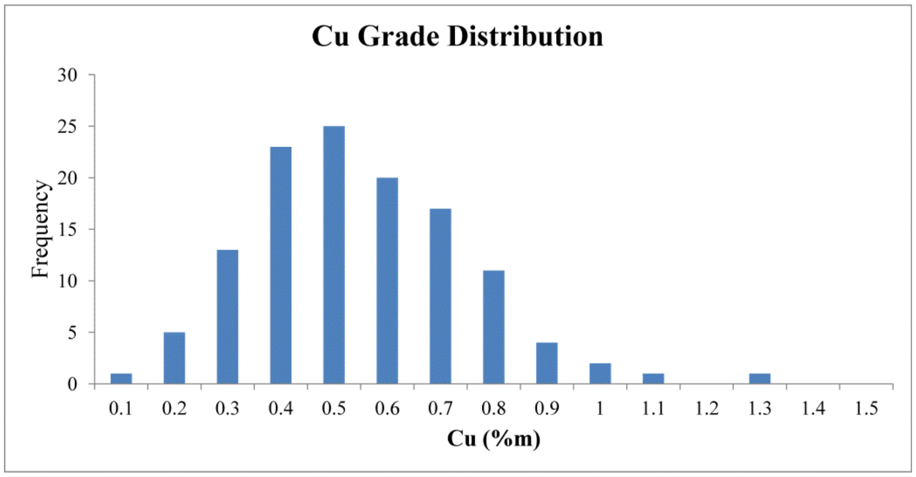


Figure 4.13. Clustered Cu Grade Distribution

Table 4.8. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	123	123
Mean	0.395	0.495
Median	0.382	0.479
Standard Deviation	0.184	0.199
Sample Variance	0.034	0.040
Kurtosis	-0.314	0.671
Skewness	0.428	0.505
Range	0.893	1.174
Minimum	0.033	0.034
Maximum	0.926	1.207

Table 4.9. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	5,662
Rock Unity (%)	96.0	11.9
DDF (%)	99.4	4.5
Au CV (%)	23.6	19.4
Cu CV (%)	26.9	17.2

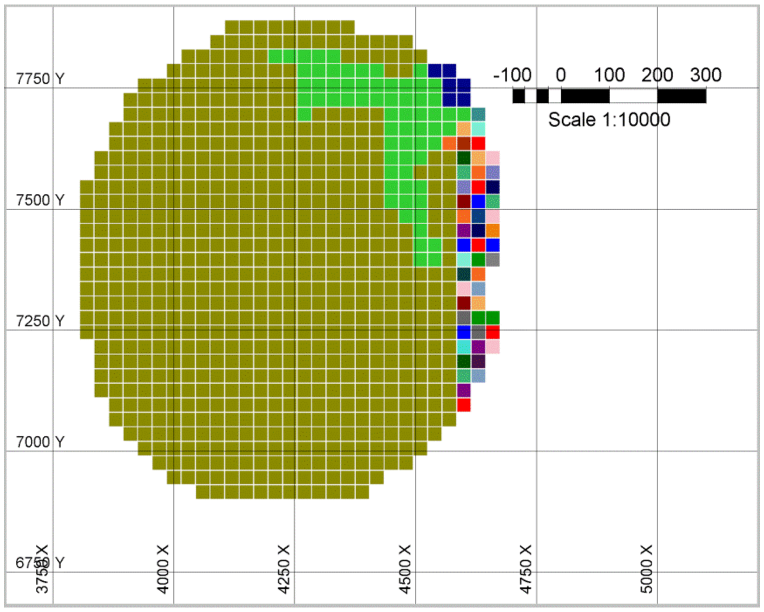


Figure 4.14. Cluster IDs Plan View at 600m Elevation

4.3.3 *Rock-type Control*

Another important characteristic of a block is its rock-type especially when different rock-types are processed in different plants. Since rock-type is a categorical variable, the rock-type similarity between two blocks is modeled with a penalty value. It this section, we aim at creating clusters with homogenous rock-types. Since smaller values for penalties means higher impact on the similarity measure we use 0.1 for rock-type penalty as the sole similarity measure (Table 4.10). As a result, the generated clusters are 98% formed of the same rock-type. However, using rock-type as the sole similarity measure suffers the same setbacks as mentioned in section 4.3.2. The clustered element grade distributions and the sample plan view can be found in Figure 4.15 to Figure 4.17.

Table 4.10. Clustering Parameters

Parameter	Value
Distance Weight	0
Grade Weight	0
Rock-type Penalty	0.1
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

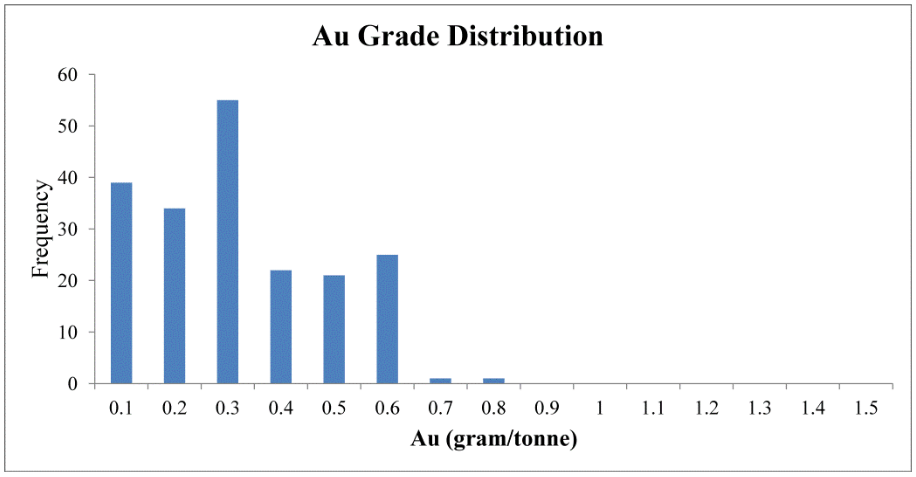


Figure 4.15. Clustered Au Grade Distribution

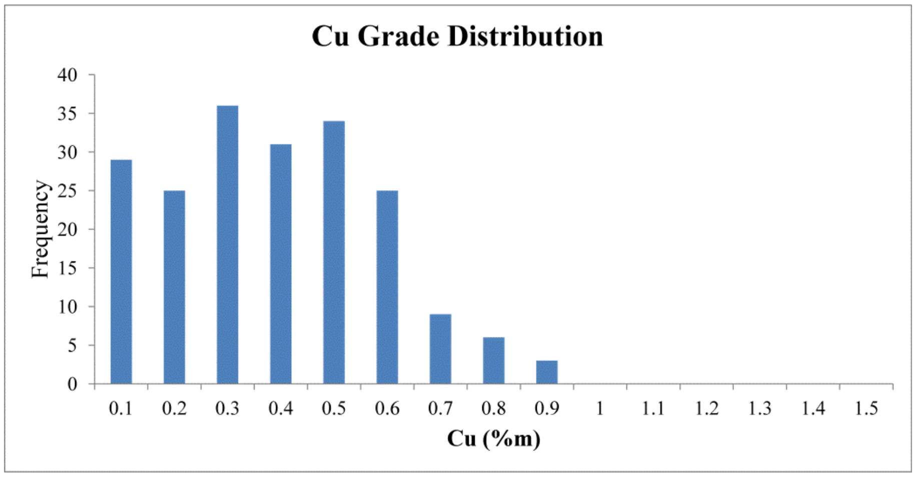


Figure 4.16. Clustered Cu Grade Distribution

Table 4.11. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	198	198
Mean	0.269	0.337
Median	0.244	0.331
Standard Deviation	0.165	0.200
Sample Variance	0.027	0.040
Kurtosis	-0.553	-0.554
Skewness	0.450	0.304
Range	0.756	0.815
Minimum	0.012	0.020
Maximum	0.768	0.835

Table 4.12. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	3,942
Rock Unity (%)	98.0	7.5
DDF (%)	100.0	0.0
Au CV (%)	21.7	40.7
Cu CV (%)	20.3	39.5

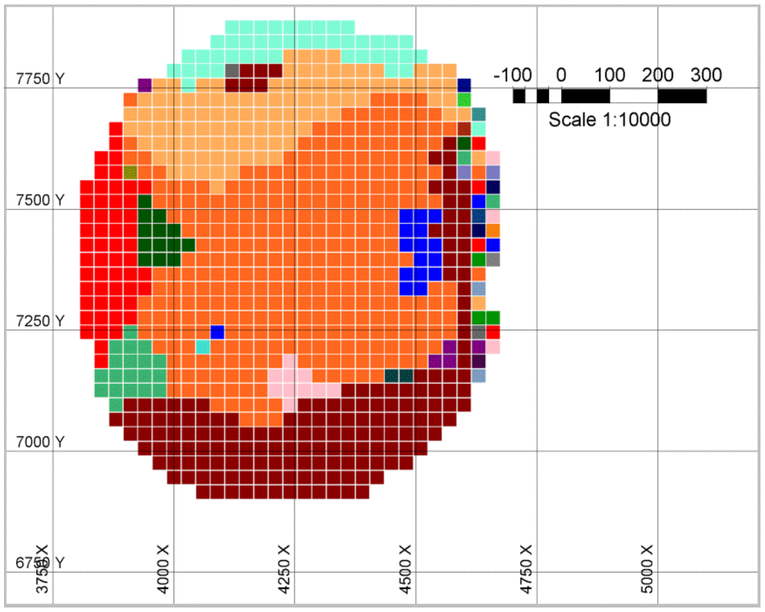


Figure 4.17. Cluster IDs Plan View at 600m Elevation

4.3.4 *Distance and Grade*

As shown in section 4.3.2, clustering blocks solely with regard to grade results in impractical clusters that have extremely irregular shapes following the patterns of high and low grade in the deposit. In order to form more reasonable clusters we use both the distance and grade as the similarity measures. The clustering parameters are summarized in Table 4.13. With this setting, we create clusters with better shape while conserving the gold grade homogeneity within clusters. However, the variation in cluster size is making the clusters unusable as planning units. The standard deviation of cluster tonnage from Table 4.15 clearly shows the variation in cluster size. The clustered grade histograms can be found in Figure 4.18 and Figure 4.19.

Table 4.13. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0.5
Rock-type Penalty	1
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

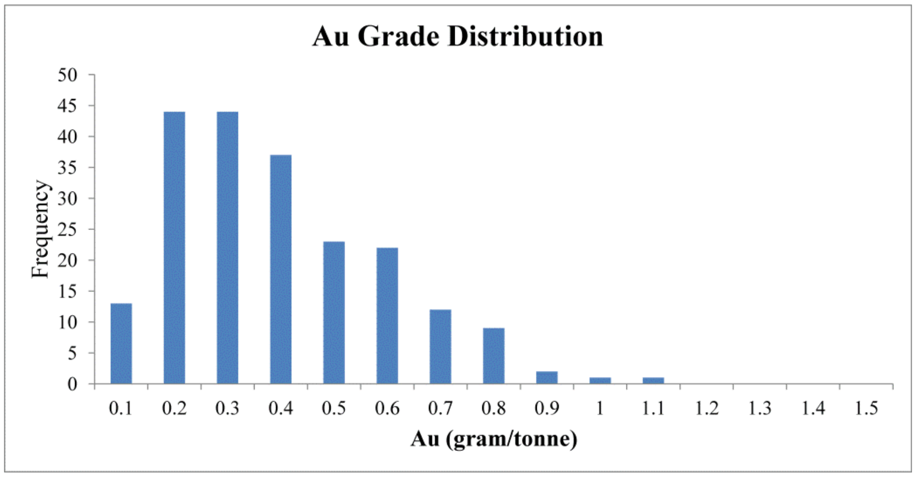


Figure 4.18. Clustered Au Grade Distribution

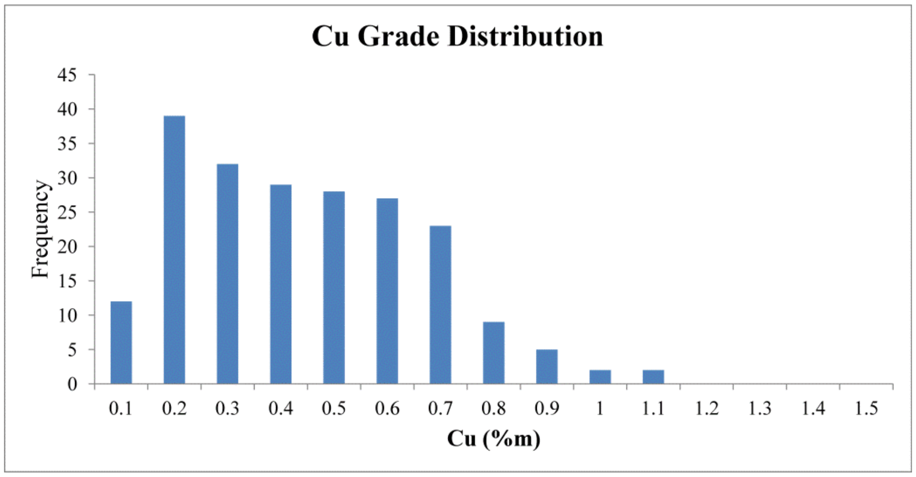


Figure 4.19. Clustered Cu Grade Distribution

Table 4.14. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	208	208
Mean	0.347	0.397
Median	0.308	0.375
Standard Deviation	0.197	0.223
Sample Variance	0.039	0.050
Kurtosis	0.398	-0.133
Skewness	0.828	0.547
Range	1.061	1.077
Minimum	0.020	0.016
Maximum	1.081	1.094

Table 4.15. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	1,520
Rock Unity (%)	91.8	14.9
DDF (%)	95.4	11.0
Au CV (%)	41.8	37.2
Cu CV (%)	44.9	35.4

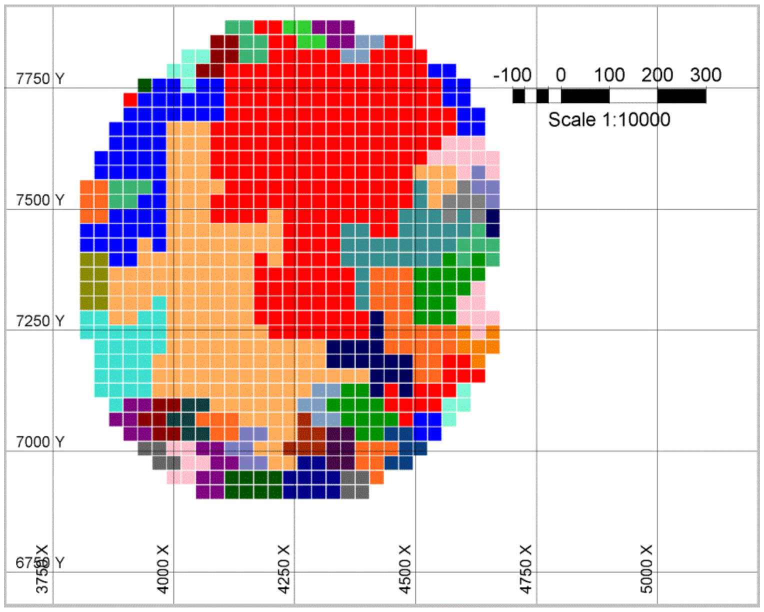


Figure 4.20. Cluster IDs Plan View at 600m Elevation

4.3.5 *Distance and Rock-type*

Similar to section 04.3.4, we can combine the rock-type with distance to form clusters that are both homogenous in rock-type and practical from the mining operations point of view. Table 4.16 shows the parameters used in this clustering scheme. Since the rock-types have less variation compared to grades, the generated clusters are more similar to distance only clustering (Figure 4.23). The average rock unity of clustering results is between the two rock unities gained from distance only and rock-type only clustering (Table 4.18). The clustered element grade distributions can be found in Figure 4.21 to Figure 4.22.

Table 4.16. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

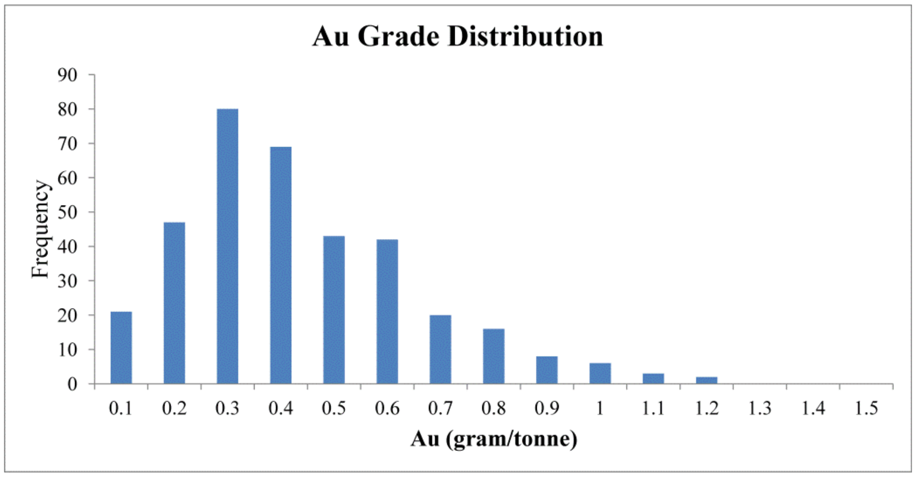


Figure 4.21. Clustered Au Grade Distribution

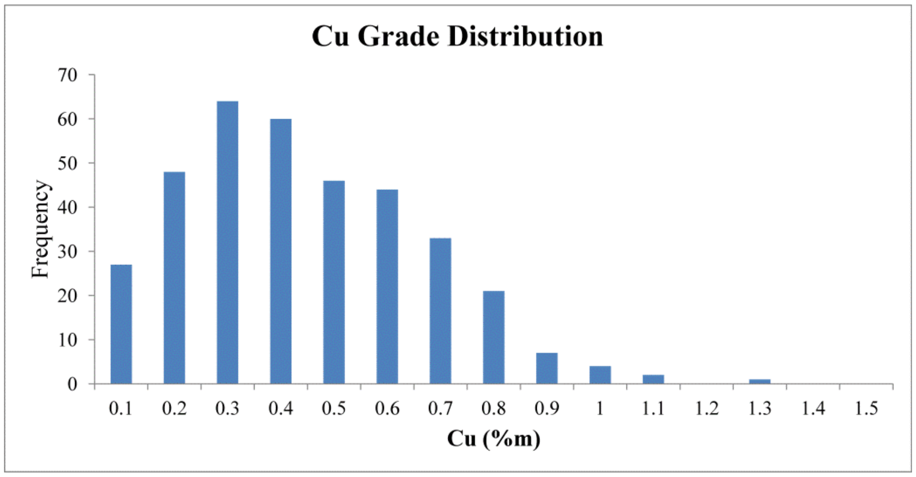


Figure 4.22. Clustered Cu Grade Distribution

Table 4.17. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	357	357
Mean	0.385	0.395
Median	0.337	0.374
Standard Deviation	0.219	0.222
Sample Variance	0.048	0.049
Kurtosis	0.706	-0.022
Skewness	0.923	0.568
Range	1.145	1.226
Minimum	0.012	0.020
Maximum	1.157	1.246

Table 4.18. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	584
Rock Unity (%)	94.2	12.0
DDF (%)	100.0	1.3
Au CV (%)	36.6	36.6
Cu CV (%)	36.0	36.3

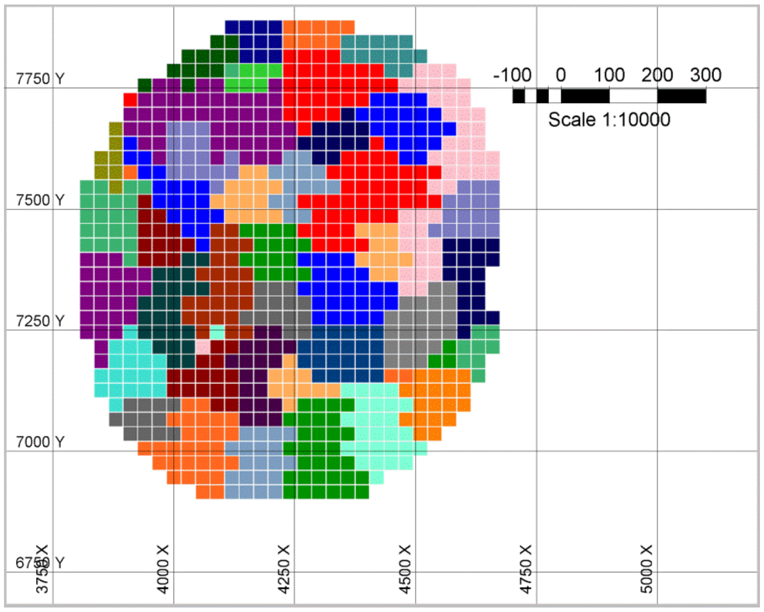


Figure 4.23. Cluster IDs Plan View at 600m Elevation

4.3.6 *Distance, Grade and Rock-type*

How the similarity index is defined in this project enables us to include as many similarity factors as we want in the calculations of the similarity matrix. However, if we include too many similarity factors in the similarity definition, the similarity index becomes noisy and may lose its credibility in determining similar blocks. Therefore, we usually avoid using correlated factors in determining the similarity between the blocks. In this section, we tried to balance the weights on grade and distance with a penalty on rock-type to respect grade and rock-type pattern along with controlling the shape of the formed clusters (Table 4.19). However, having distance, grade and rock-type with equal emphasis in the similarity index undermines the quality of the generated clusters and significantly disturbs grade variation in the clusters compared to the distance and rock-type case. The clustered element grade distributions can be found in Figure 4.21 to Figure 4.22. A sample plan view for the clustering scheme is presented in Figure 4.26. The summary of clustering evaluation is presented in Table 4.21.

Table 4.19. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0.5
Rock-type Penalty	0.5
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

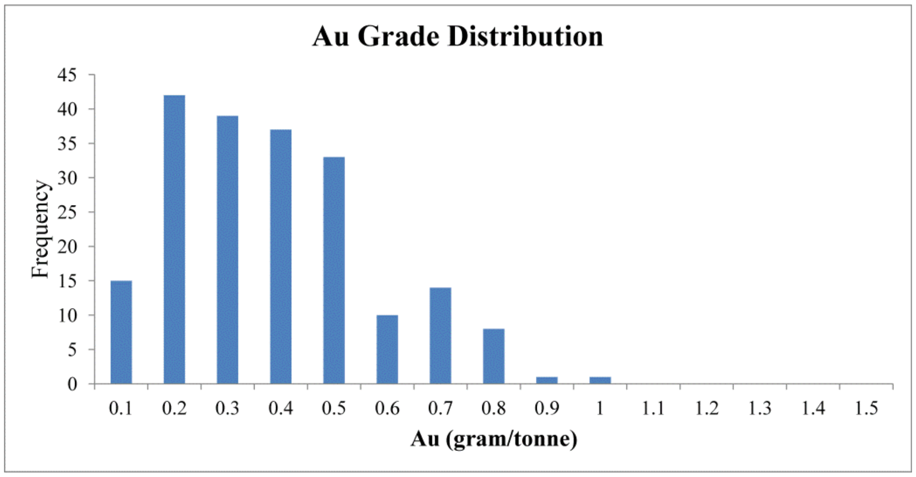


Figure 4.24. Clustered Au Grade Distribution

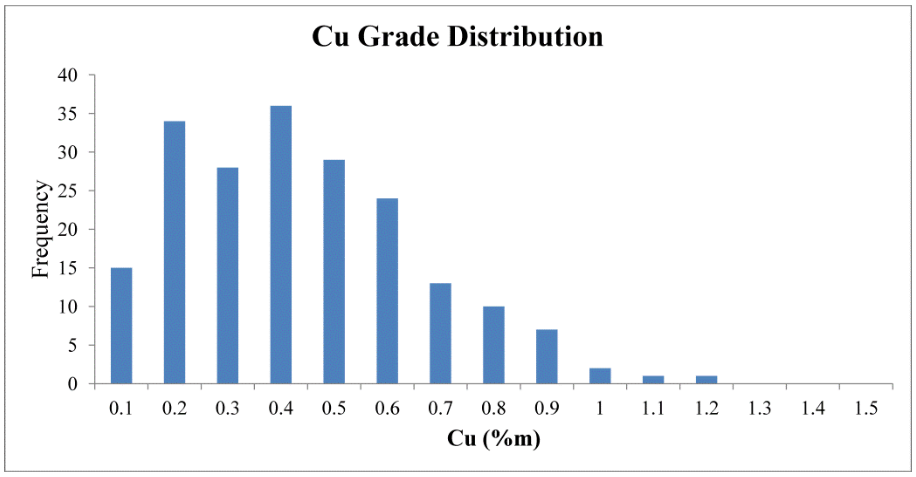


Figure 4.25. Clustered Cu Grade Distribution

Table 4.20. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	200	200
Mean	0.333	0.387
Median	0.309	0.358
Standard Deviation	0.190	0.227
Sample Variance	0.036	0.051
Kurtosis	-0.089	0.060
Skewness	0.677	0.623
Range	0.916	1.174
Minimum	0.012	0.013
Maximum	0.928	1.186

Table 4.21. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	1,528
Rock Unity (%)	93.8	13.6
DDF (%)	98.2	7.1
Au CV (%)	54.0	99.5
Cu CV (%)	56.0	98.1

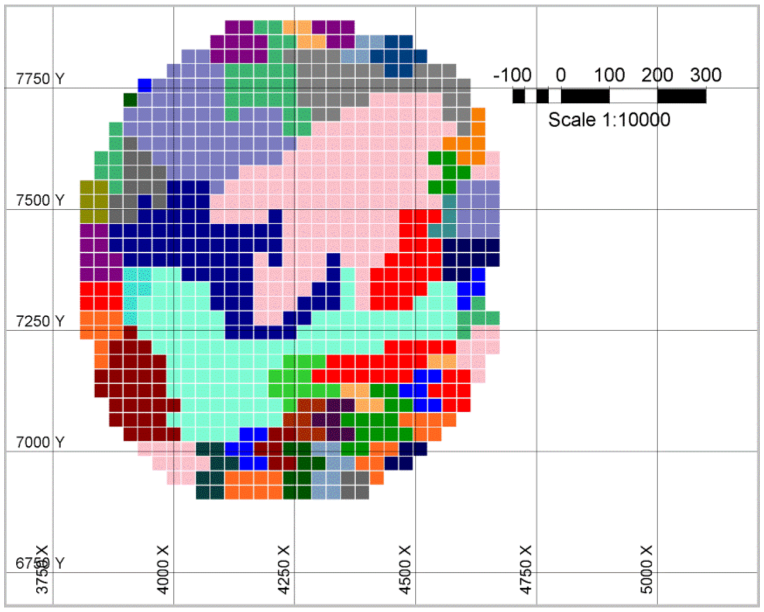


Figure 4.26. Cluster IDs Plan View at 600m Elevation

4.3.7 *Tabu Search*

We have developed a Tabu search procedure to decrease the number of precedence arcs between the generated clusters of consequent benches. This procedure becomes handy when the clusters are going to be used as mining units. Since we are trying to decrease the number of precedence arcs, we will use beneath cluster as a similarity measure in order to form clusters of same shape on consequent benches. The algorithm settings are similar to section 4.3.5 with the addition of a penalty for beneath cluster (Table 4.22). The clustered grade histograms are presented in Figure 4.27 and Figure 4.28. The clustering evaluation summary is presented in Table 4.24. Figure 4.29 and Figure 4.30 show the sample views for before and after Tabu search respectively. The effects of using Tabu search on the MILP processing time and results are studied in section 4.7.5.

Table 4.22. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Beneath Cluster Penalty	0.5
Avg. Blocks per Cluster	15
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

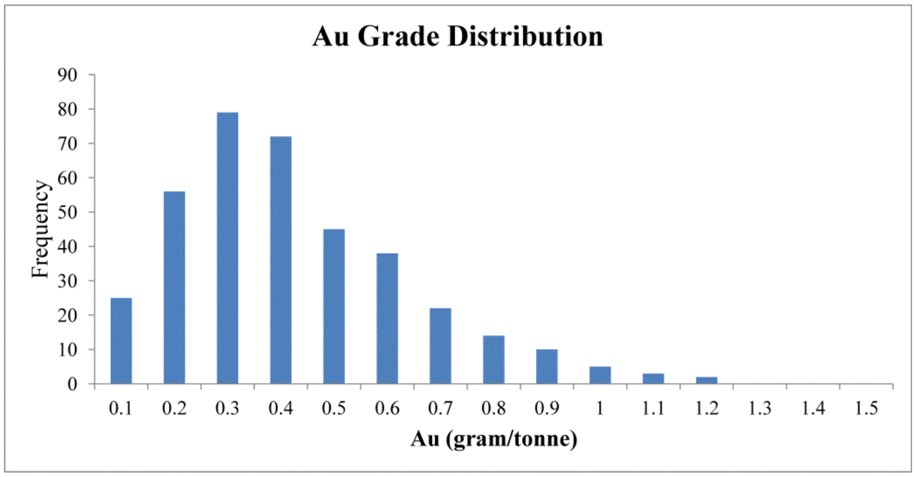


Figure 4.27. Clustered Au Grade Distribution

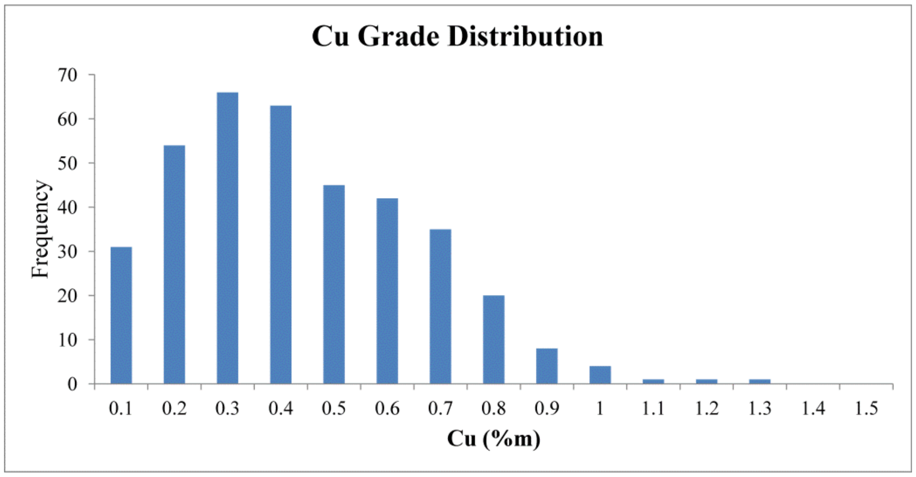


Figure 4.28. Clustered Cu Grade Distribution

Table 4.23. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	371	371
Mean	0.375	0.386
Median	0.328	0.360
Standard Deviation	0.220	0.225
Sample Variance	0.049	0.051
Kurtosis	0.759	0.022
Skewness	0.947	0.616
Range	1.133	1.236
Minimum	0.012	0.020
Maximum	1.145	1.255

Table 4.24. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	909	1,205
Rock Unity (%)	90.0	15.4
DDF (%)	96.0	10.4
Au CV (%)	49.6	57.9
Cu CV (%)	48.2	54.2

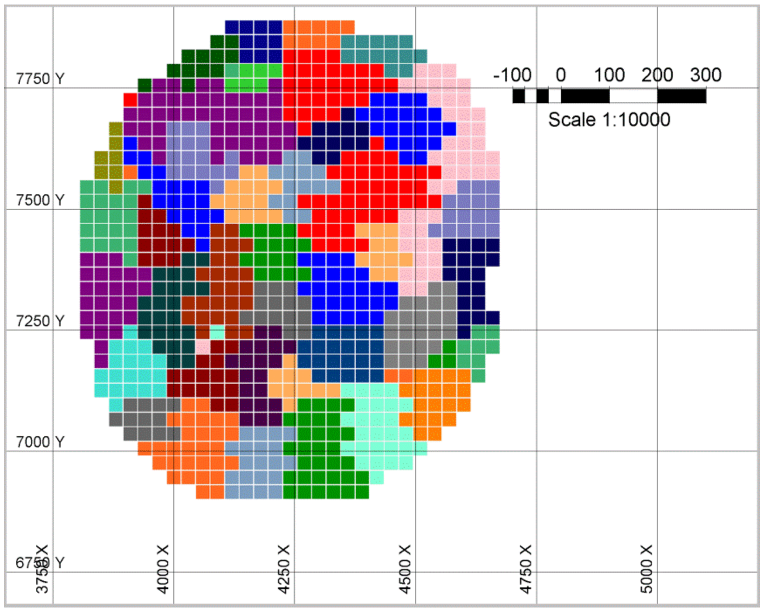


Figure 4.29. Cluster IDs Plan View at 600m Elevation (before Tabu Search)

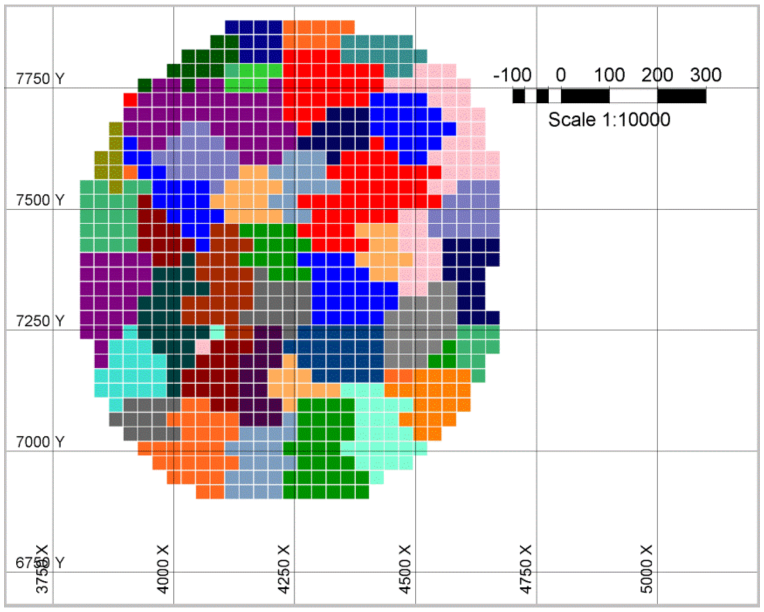


Figure 4.30. Cluster IDs Plan View at 600m Elevation (after Tabu Search)

4.3.8 *Size Control*

One of the most important features of our clustering algorithm is its capability to control the size of the generated clusters. In this section, we ran a few different scenarios to show how the control on the cluster size can change the outcome.

1. The first scenario is to use the same weights as in section 4.3.5, without limits on the cluster size. The scenario in section 4.3.5 was run with 15 blocks per cluster and here we double the size and run with 30 blocks per cluster to evaluate the effects of having larger clusters on the outcome (Table 4.26). Although the rock unity is not significantly affected, the grade variation grows as it is not considered in the similarity index. On the other hand, the shape and size of the clusters show unexpected disturbance (Figure 4.31) compared to section 4.3.5 which portrays the need for better control on the shape and size of the generated clusters.

Table 4.25. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Avg. Blocks per Cluster	30
Max. Blocks per cluster	5000
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

Table 4.26. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	1,790	1,703
Rock Unity (%)	94.5	11.6
DDF (%)	99.8	2.8
Au CV (%)	42.4	38.8
Cu CV (%)	39.5	38.8

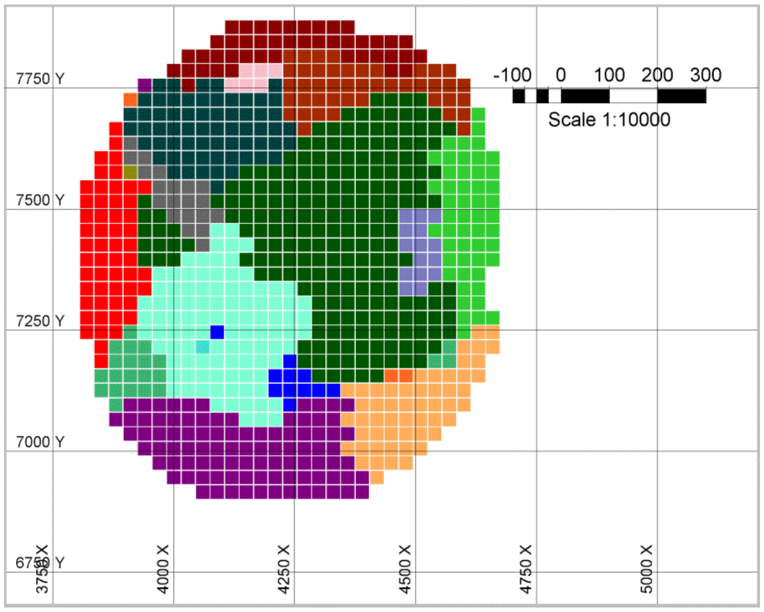


Figure 4.31. Cluster IDs Plan View at 600m Elevation

2. Following the same settings we doubled the size of the clusters to 60 blocks per cluster and below is the results. As expected, the rock unity did not significantly change but the grade variations rose again. The variation in size has also increased as there no constraint on the size yet. The clustering outcome is presented in Table 4.27. A sample plan view in shown in Figure 4.32.

Table 4.27. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	3,504	4,489
Rock Unity (%)	93.9	12.6
DDF (%)	98.1	7.7
Au CV (%)	50.2	50.7
Cu CV (%)	45.3	49.4

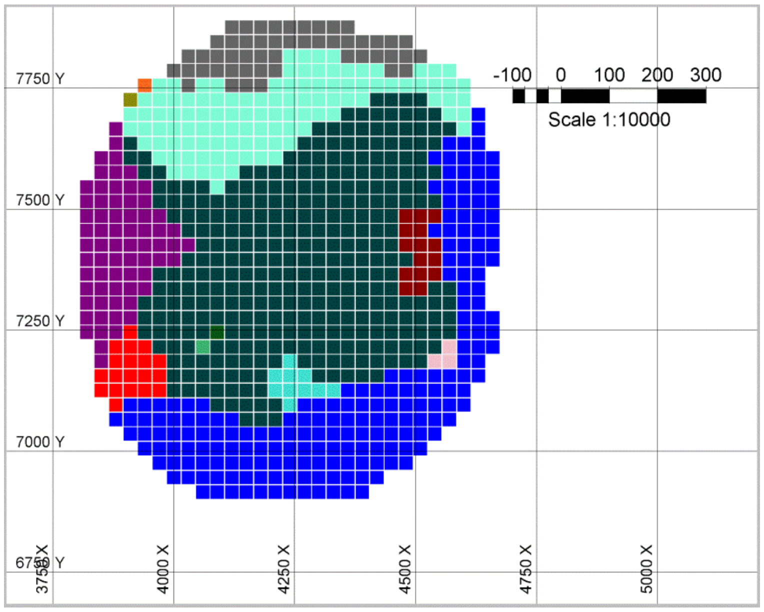


Figure 4.32. Cluster IDs Plan View at 600m Elevation

3. Increasing the cluster size to 90 blocks per clusters further affects the clustering outcome and starts to sabotage the rock unity even though it is included in calculating the similarity index. The clustering evaluation summary is presented in Table 4.28. Figure 4.33 is a sample plan view from the clustering scheme.

Table 4.28. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	5,177	6,783
Rock Unity (%)	93.1	13.0
DDF (%)	97.1	9.6
Au CV (%)	51.7	52.6
Cu CV (%)	47.0	52.8

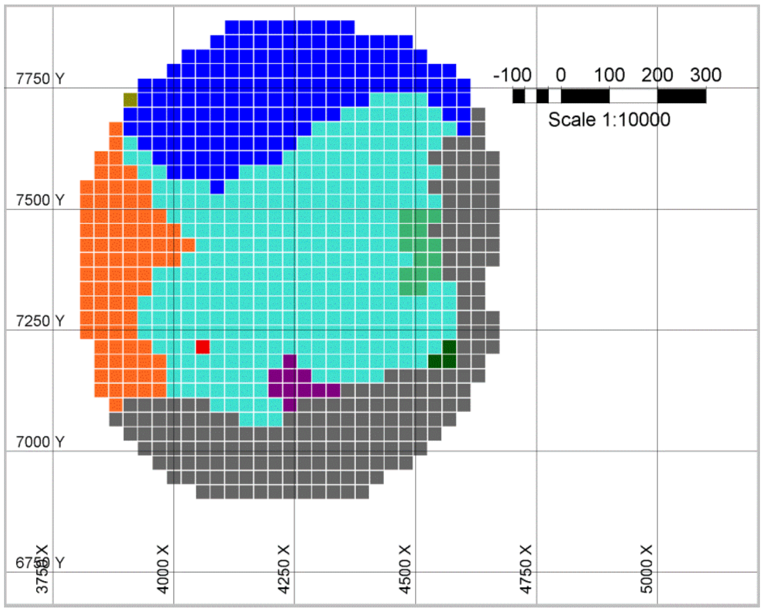


Figure 4.33. Cluster IDs Plan View at 600m Elevation

4. In order to decrease variations in cluster size and generate more uniform useful clusters, we apply minimum and maximum cluster size constraints (Table 4.29). The following scenario is a clustering scheme based on rock homogeneity with controlled cluster size. Note that the minimum constraint is enforced in the post-processing shape refinement procedure and can violate the maximum size constraint. On the other hand, since the shape refinement is an iterative procedure that attaches single block clusters to the neighbor clusters, having only one iteration will not remove all the single block clusters. A better setting is used in the next section to examine the shape refinement and minimum size enforcement. The clustering evaluation is summarized in Table 4.30. A sample plan view is presented in Figure 4.34.

Table 4.29. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Avg. Blocks per Cluster	15
Max. Blocks per cluster	20
Min. Blocks per Cluster	10
Number of Shape Refinement Iterations	1

Table 4.30. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	674	529
Rock Unity (%)	92.7	13.1
DDF (%)	98.0	5.9
Au CV (%)	30.3	28.1
Cu CV (%)	29.9	26.9

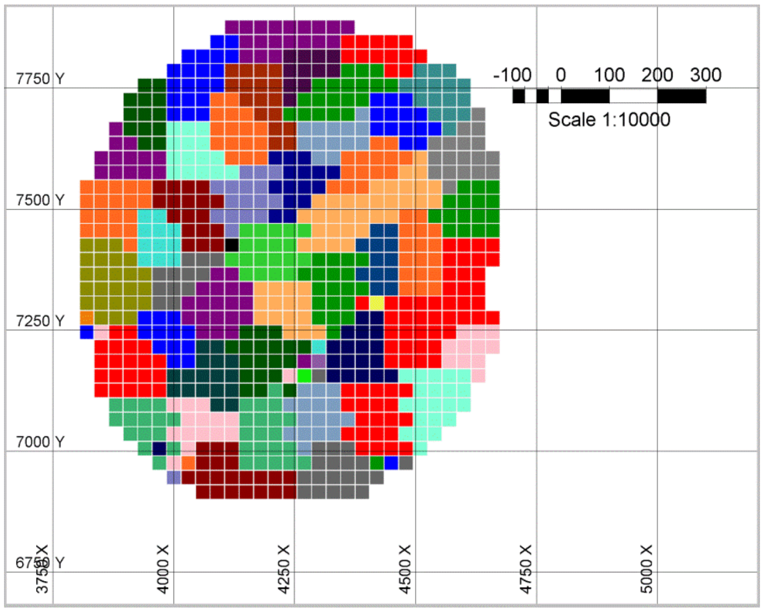


Figure 4.34. Cluster IDs Plan View at 600m Elevation

4.3.9 *Shape Refinement*

As explained in section 3.3.3, we developed a shape refinement post processing procedure to be used when jagged shaped clusters with sharp corners are not desirable. The same procedure is responsible for removing clusters smaller than the specified threshold. Since the procedure is designed to improve the shape of the generated clusters it may sabotage the intra-cluster similarity by attaching non-similar blocks to the cluster. As we can see in Table 4.33, the rock unity drops and both grade variations increase. On the other hand, the sharp corners and small clusters are removed to improve the cluster shapes (Figure 4.38). the clustered grade histograms are plotted in Figure 4.35 and Figure 4.36.

Table 4.31. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Avg. Blocks per Cluster	15
Max. Blocks per cluster	20
Min. Blocks per Cluster	5
Number of Shape Refinement Iterations	3

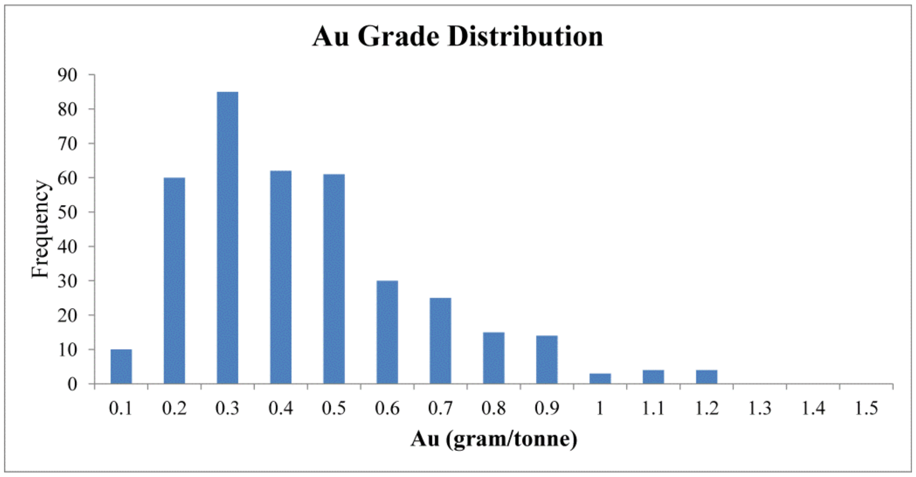


Figure 4.35. Clustered Au Grade Distribution

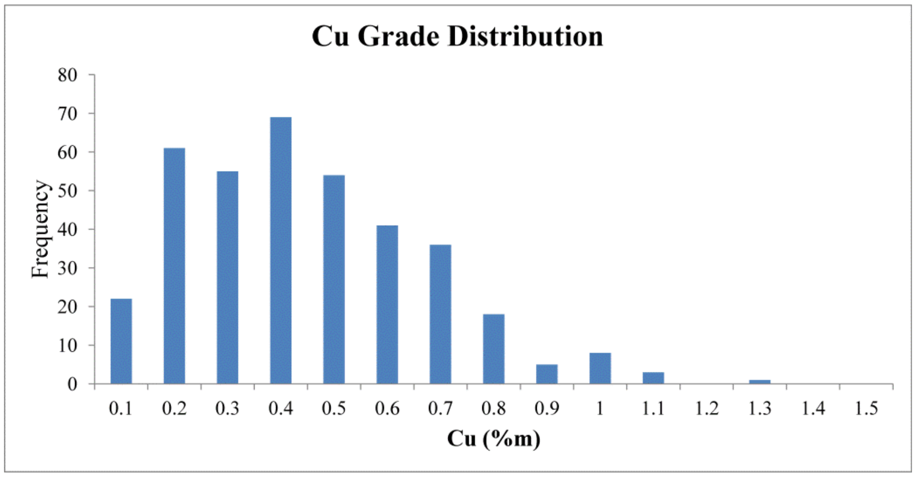


Figure 4.36. Clustered Cu Grade Distribution

Table 4.32. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	373	373
Mean	0.397	0.400
Median	0.344	0.375
Standard Deviation	0.223	0.225
Sample Variance	0.050	0.051
Kurtosis	0.762	0.216
Skewness	1.004	0.679
Range	1.131	1.215
Minimum	0.033	0.018
Maximum	1.164	1.233

Table 4.33. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	862	333
Rock Unity (%)	91.5	13.5
DDF (%)	97.9	6.2
Au CV (%)	38.0	27.2
Cu CV (%)	37.8	25.7

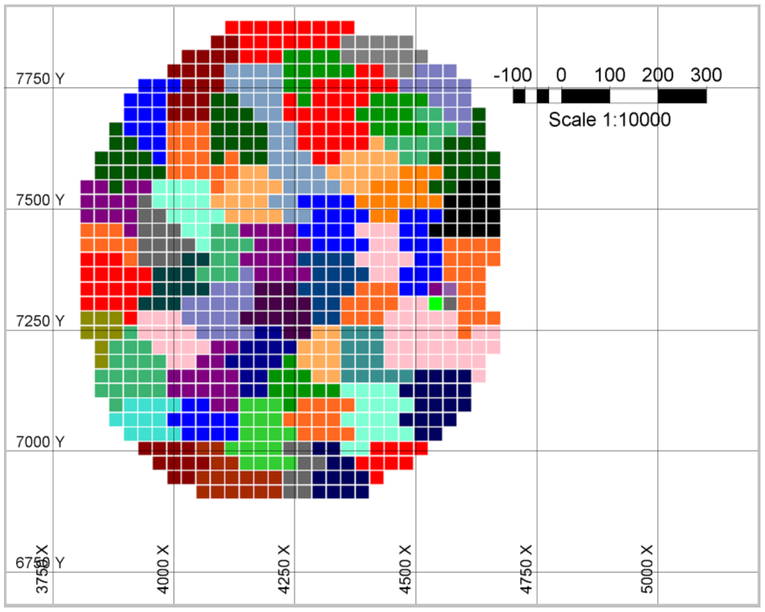


Figure 4.37. Cluster IDs Plan View at 600m Elevation (before Shape Refinement)

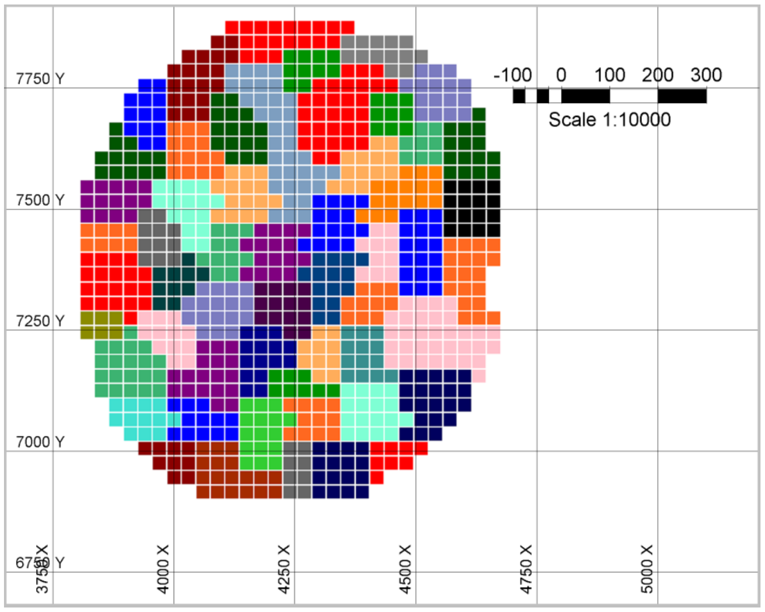


Figure 4.38. Cluster IDs Plan View at 600m Elevation (after Shape Refinement)

4.3.10 *Clustering within Boundaries*

In many cases, especially when the formed clusters are going to be used as processing units within bench-phases, we required the clusters to be formed within predetermined boundaries. Therefore, we included a feature in our algorithm that can tweak the adjacency matrix such that blocks from two different sides of a boundary are never grouped together.

In this section, we present a clustering scheme that is respecting pushback boundaries and can be used in MILP formulation when the processing units are clusters and mining units are bench-phases. As can be seen in Figure 4.41 and Figure 4.42, the clusters do not overlap pushbacks. However, enforcing more constraints (pushback boundaries) will decrease the rock unit and increase grade variation as shown in Table 4.36.

Table 4.34. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Avg. Blocks per Cluster	15
Max. Blocks per cluster	20
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	3

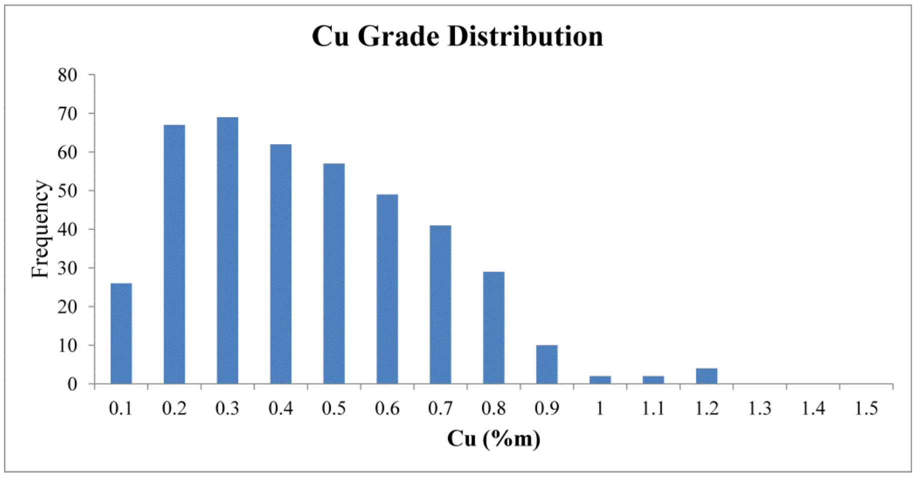


Figure 4.39. Clustered Au Grade Distribution

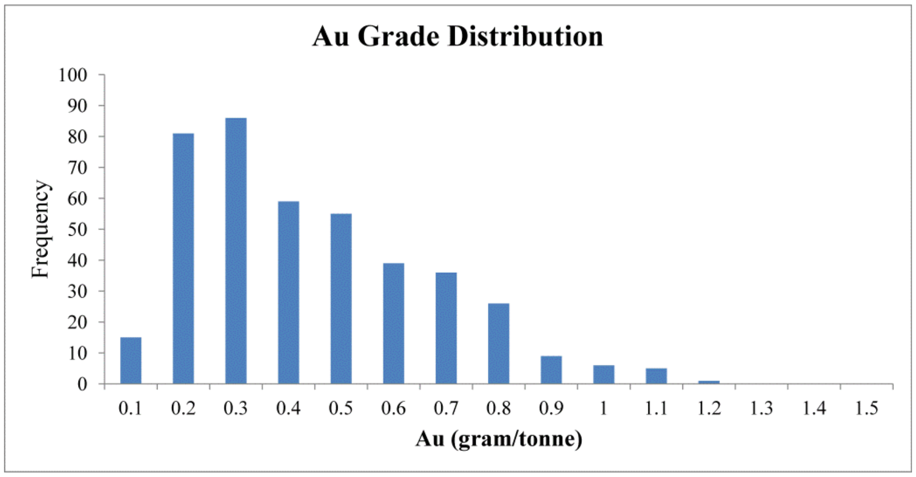


Figure 4.40. Clustered Cu Grade Distribution

Table 4.35. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	418	418
Mean	0.393	0.404
Median	0.337	0.382
Standard Deviation	0.227	0.230
Sample Variance	0.051	0.053
Kurtosis	-0.019	-0.049
Skewness	0.771	0.600
Range	1.094	1.173
Minimum	0.023	0.020
Maximum	1.118	1.193

Table 4.36. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	819	349
Rock Unity (%)	91.1	13.5
DDF (%)	97.4	7.2
Au CV (%)	37.1	28.8
Cu CV (%)	37.4	27.3

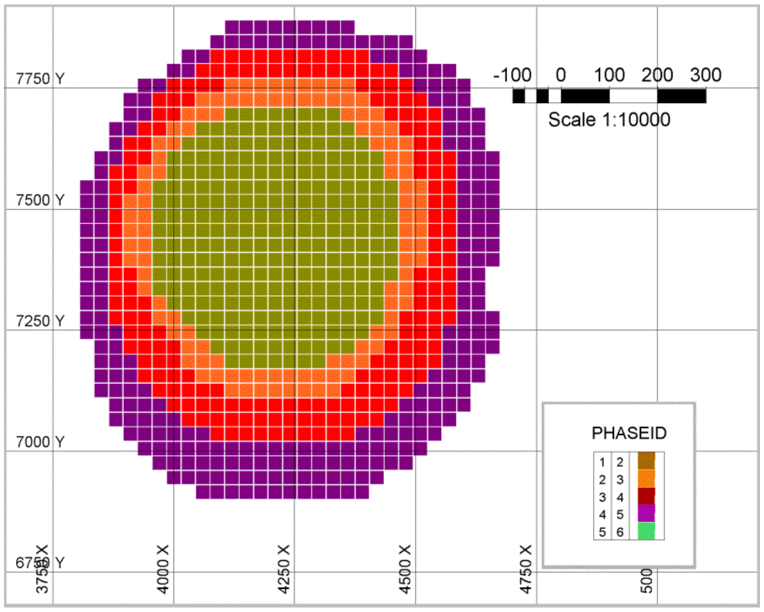


Figure 4.41. Phase IDs Plan View at 600m Elevation

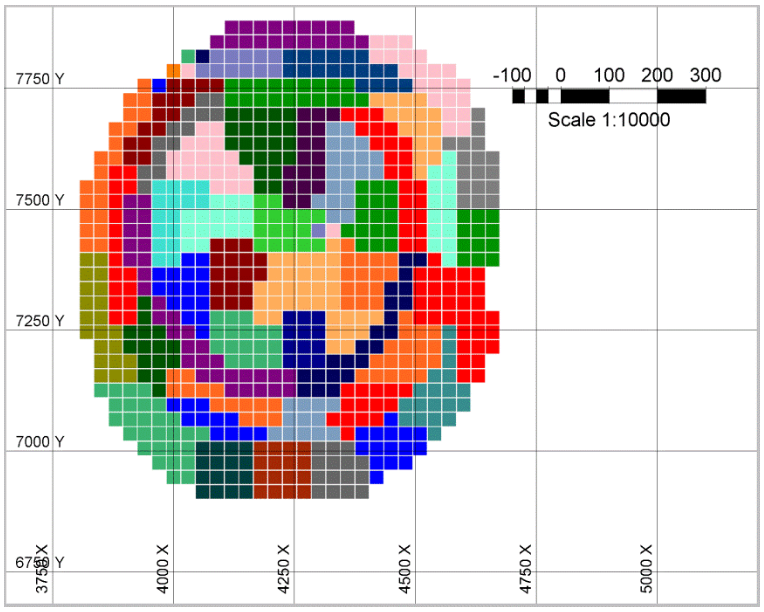


Figure 4.42. Cluster IDs Plan View at 600m Elevation

4.3.11 *Clustering based on Mining Direction*

Since the clustering algorithm is not only designed to be used in long-term production planning but also in any planning problem that requires mining polygons, we added the mining direction to the clustering features. In order to use this feature we require two clustering reference points (CRPs) for each bench. In this section, we illustrate how the algorithm can create mining faces based on CRPs. The first scenario is using the two CRPs as the two ends on the mining direction vector and creates polygons that are perpendicular to this vector. The second scenario is using the CRPs as expansion centers and creates circular sectors aligned with the two center points. In both cases, the goal is to have clusters that are based on the mining direction and respect grade or rock-type homogeneity at the same time. Table 4.37 summarizes the clustering parameters used to create the upcoming clustering schemes.

Table 4.37. Clustering Parameters

Parameter	Value
Distance Weight	0.4
Grade Weight	0
Rock-type Penalty	0.2
Avg. Blocks per Cluster	50
Max. Blocks per cluster	60
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

Direction Vector: this type of clustering is useful when a directional mining operation is in place. Especially, when dealing with large mining operations with in-pit disposal, such as oil sands, clustering based on mining direction helps not only create clusters homogenous in rock-type but also designed to respect mining direction and make room for the in-pit tailing dumps. As expected, combining the direction with rock-type similarity would not produce as high rock unity as in clustering solely based on rock-type (Table 4.39). However, comparing Figure 4.45 and Figure 4.46 shows that the resulted clustering scheme follows the boundaries of different rock-types to some extent. The clustered grade histograms are presented in Figure 4.43 and Figure 4.44.

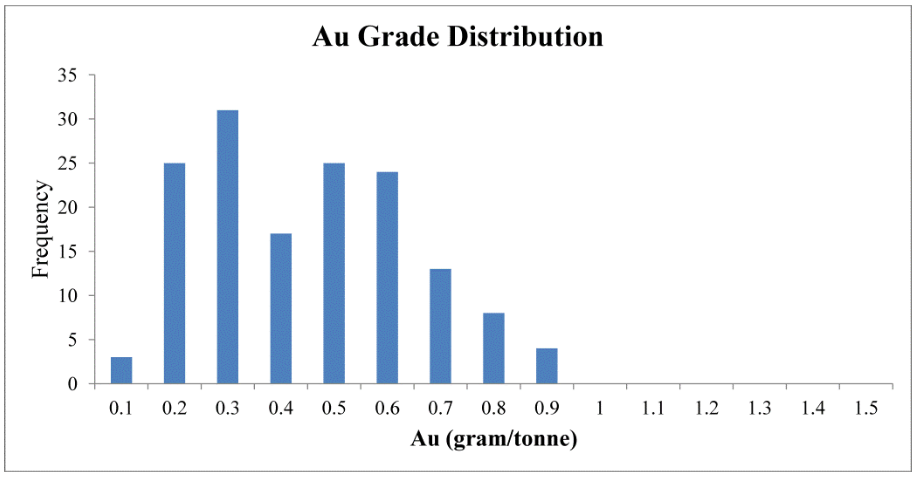


Figure 4.43. Clustered Au Grade Distribution

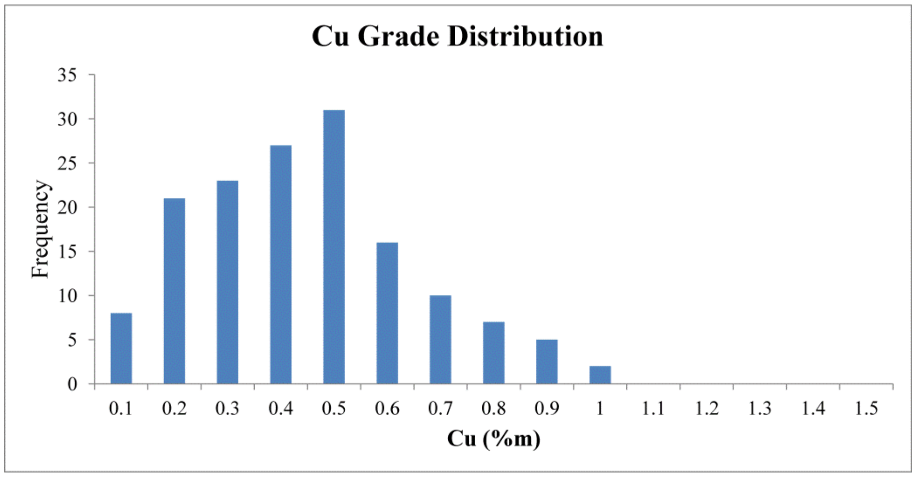


Figure 4.44. Clustered Cu Grade Distribution

Table 4.38. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	150	150
Mean	0.402	0.397
Median	0.389	0.391
Standard Deviation	0.194	0.205
Sample Variance	0.037	0.042
Kurtosis	-0.780	-0.181
Skewness	0.346	0.478
Range	0.815	0.932
Minimum	0.043	0.034
Maximum	0.857	0.966

Table 4.39. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	2,624	1,004
Rock Unity (%)	85.9	17.0
DDF (%)	93.7	12.5
Au CV (%)	50.3	20.8
Cu CV (%)	46.6	21.0

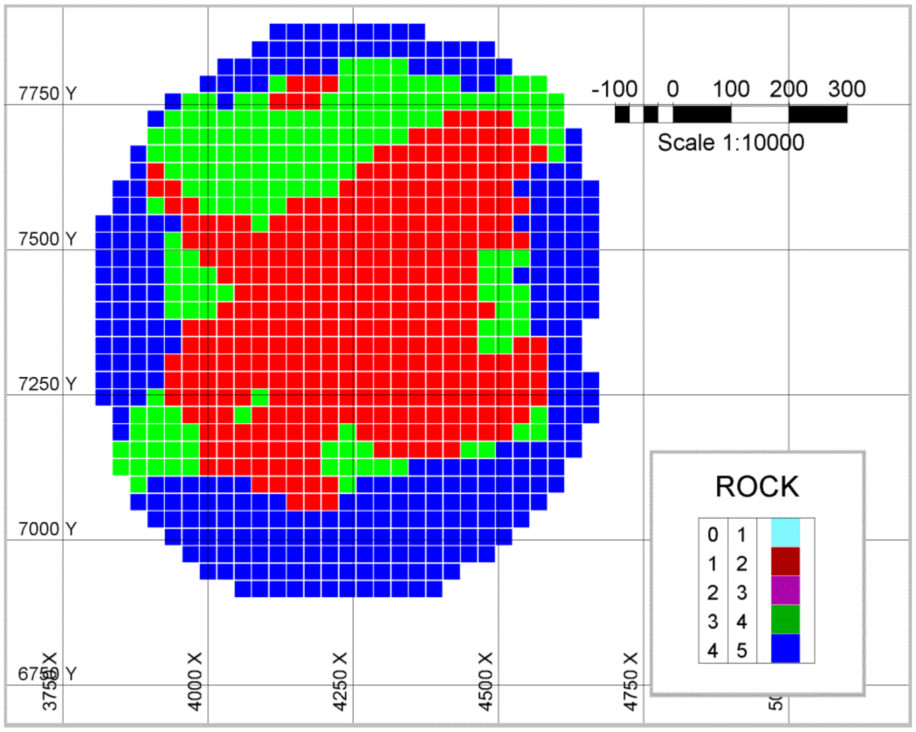


Figure 4.45. Rock-type Distribution Plan View at 600m Elevation

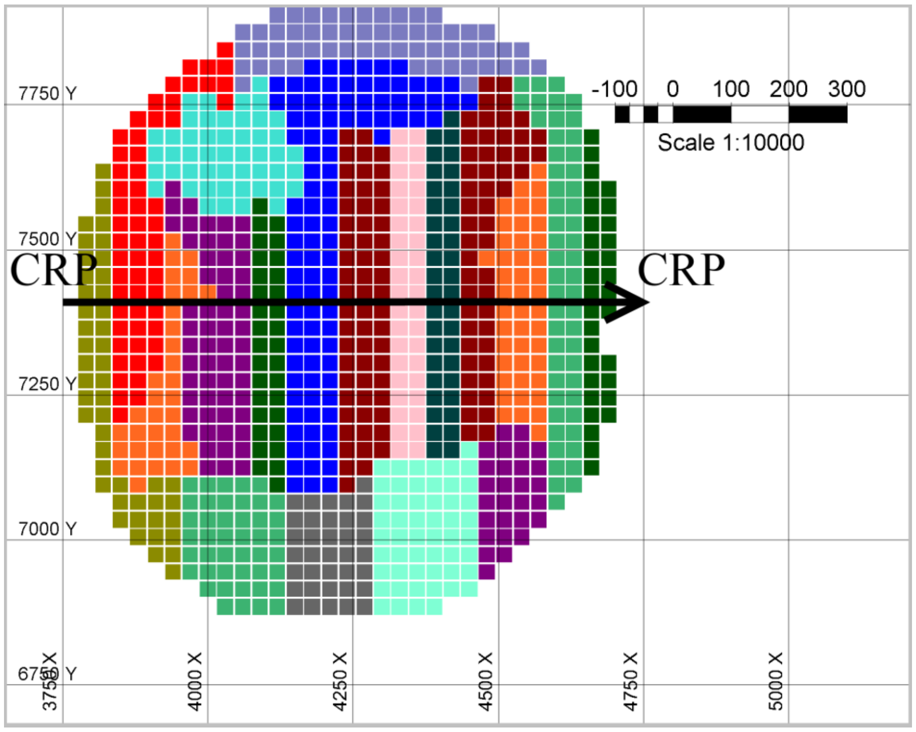


Figure 4.46. Cluster IDs Plan View at 600m Elevation

Spherical: changing the CRPs from Figure 4.46 to Figure 4.49 and changing the definition of mining direction factor in similarity index gives us clusters based on spherical expansion. In this sense, the created clusters are aligned with oval expanding from the CRPs. This is useful for central expansion operations and the CRPs can be ramp entrance points or arbitrary points above the high grade ore body. Similar to the previous example, the generated clusters respect the rock-type distribution to a good extent. The clustered grade histograms are presented in Figure 4.47 and Figure 4.48. The clustering evaluation summary is presented in Table 4.41.

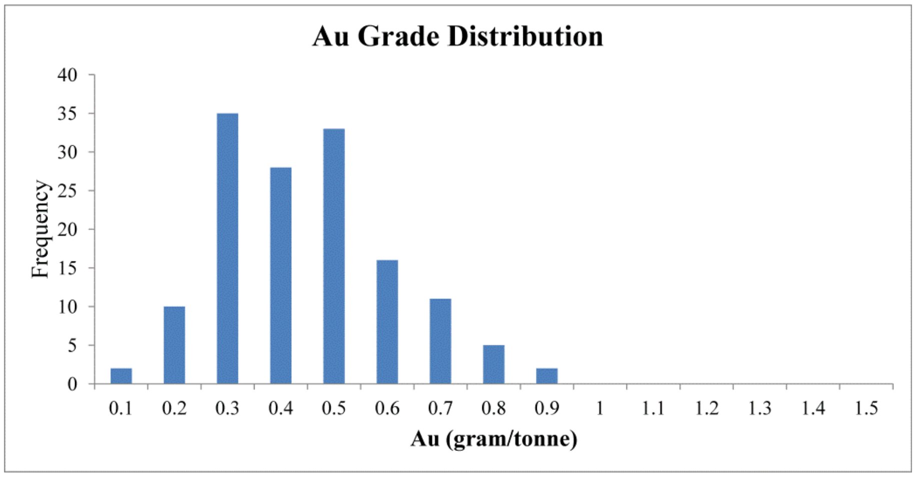


Figure 4.47. Clustered Au Grade Distribution

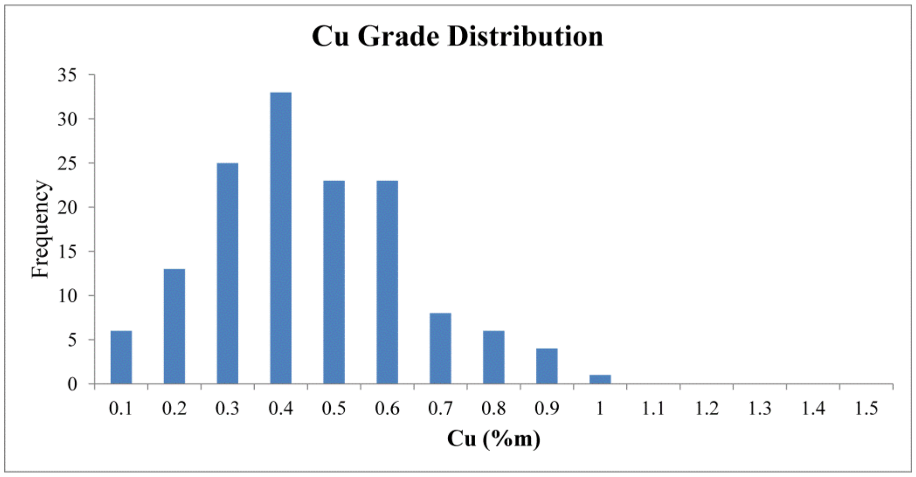


Figure 4.48. Clustered Cu Grade Distribution

Table 4.40. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	142	142
Mean	0.397	0.405
Median	0.393	0.376
Standard Deviation	0.167	0.194
Sample Variance	0.028	0.038
Kurtosis	-0.167	0.115
Skewness	0.407	0.531
Range	0.854	0.960
Minimum	0.032	0.032
Maximum	0.886	0.992

Table 4.41.Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	2,698	1,149
Rock Unity (%)	89.2	15.2
DDF (%)	96.6	10.2
Au CV (%)	48.5	22.6
Cu CV (%)	43.6	22.4

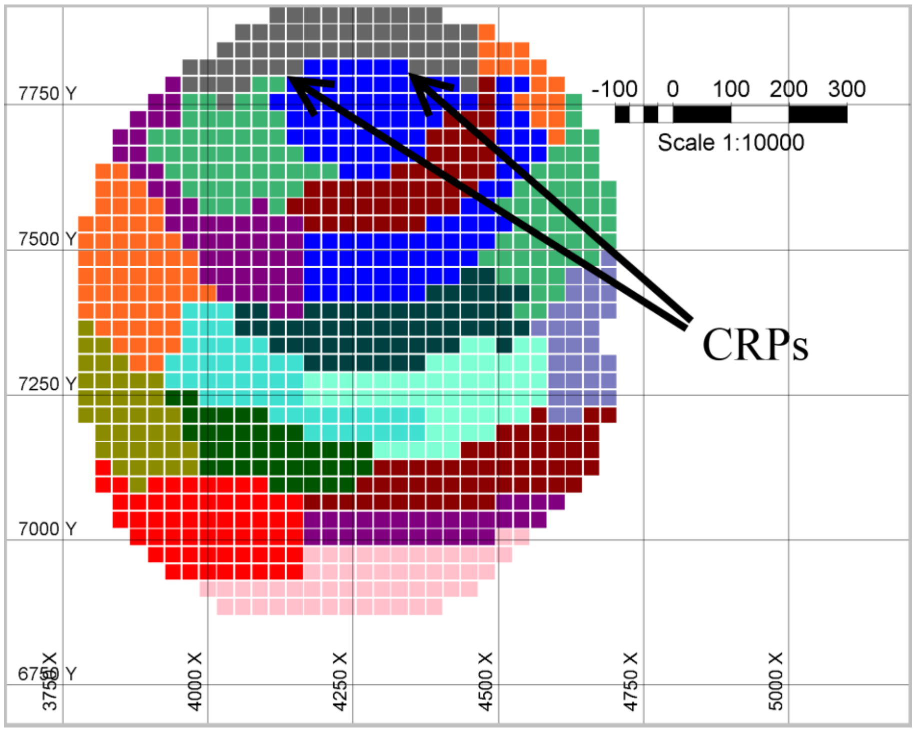


Figure 4.49. Cluster IDs Plan View at 600m Elevation

1. Spherical with Shape Refinement: in order to generate clusters with better mineability, we now run the shape refinement procedure on the same clustering scheme as the previous example with three iterations. The result is shown in Figure 4.52. The clustered grade histograms and clustering evaluation summary are presented in Figure 4.50, Figure 4.51 and Table 4.44

Table 4.42. Clustering Parameters

Parameter	Value
Distance Weight	0.4
Grade Weight	0
Rock-type Penalty	0.2
Avg. Blocks per Cluster	50
Max. Blocks per cluster	60
Min. Blocks per Cluster	5
Number of Shape Refinement Iterations	3

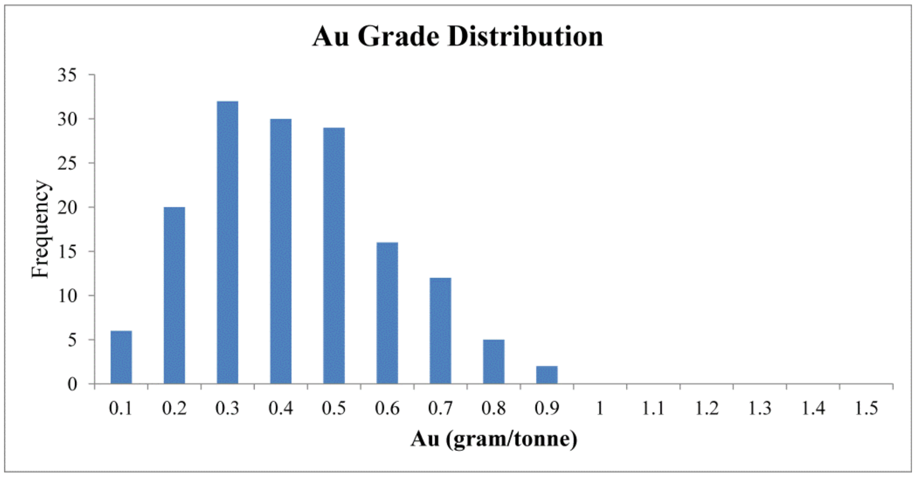


Figure 4.50. Clustered Au Grade Distribution

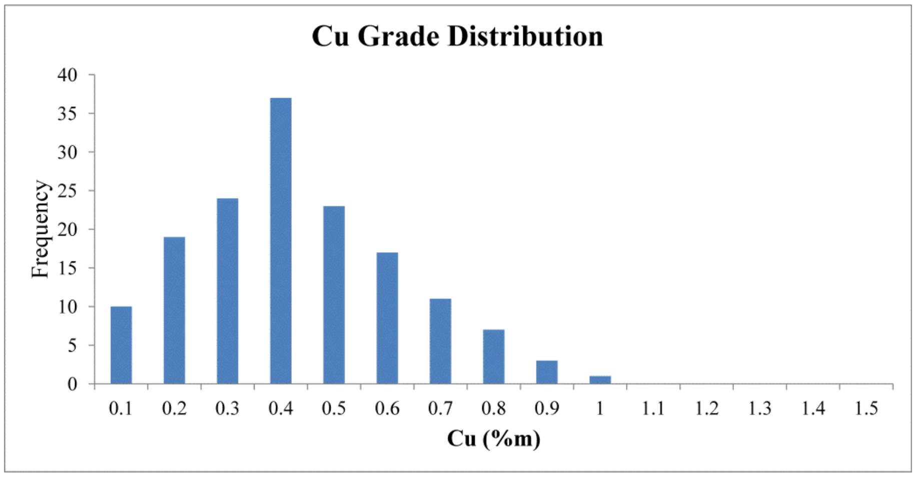


Figure 4.51. Clustered Cu Grade Distribution

Table 4.43. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	152	152
Mean	0.380	0.386
Median	0.379	0.365
Standard Deviation	0.175	0.198
Sample Variance	0.031	0.039
Kurtosis	-0.264	-0.104
Skewness	0.426	0.526
Range	0.840	0.925
Minimum	0.046	0.051
Maximum	0.886	0.976

Table 4.44. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	2,737	1,141
Rock Unity (%)	88.5	15.2
DDF (%)	95.8	10.3
Au CV (%)	47.4	22.3
Cu CV (%)	42.8	21.3

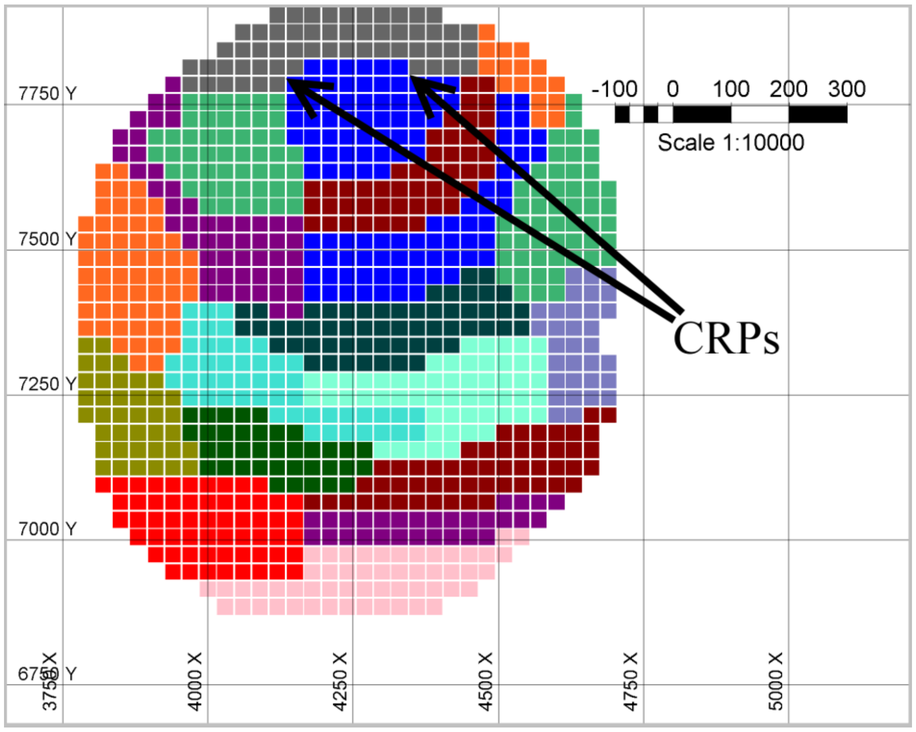


Figure 4.52. Cluster IDs Plan View at 600m Elevation

4.4. K-Means Clustering

In this section, we ran the k-means clustering algorithm on the same dataset with equal weights on distance, grade and rock-type. Note that the rock-type is treated as a numerical attribute and penalties are not applied. The k-means algorithm is run for 10 replications of 1,000 iterations and the outcome in summarized in Table 4.47. The gold and copper grade histograms are presented in Figure 4.53 and Figure 4.54. The clustering scheme for the sample bench is presented in Figure 4.55. As expected, clusters are fragmented and variable in size.

Table 4.45. Clustering Parameters

Parameter	Value
Distance Weight	1
Grade Weight	1
Rock-type Penalty	1
Avg. Blocks per Cluster	30

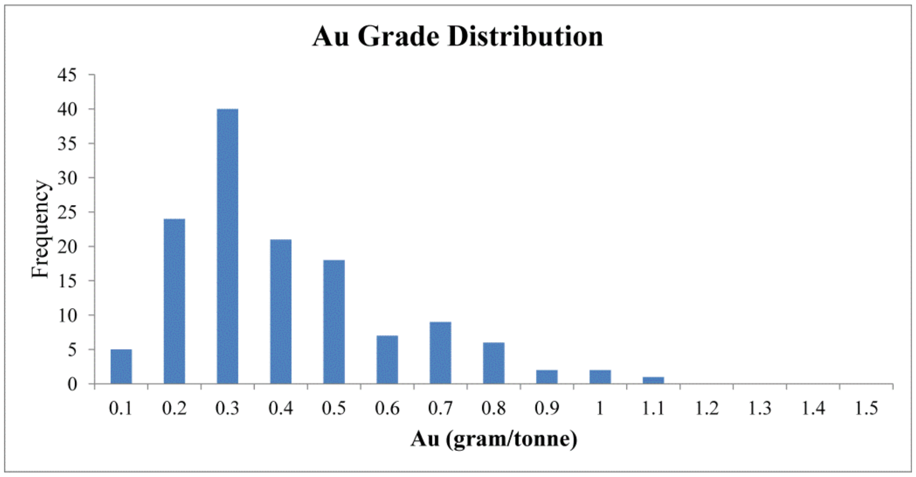


Figure 4.53. Clustered Au Grade Distribution

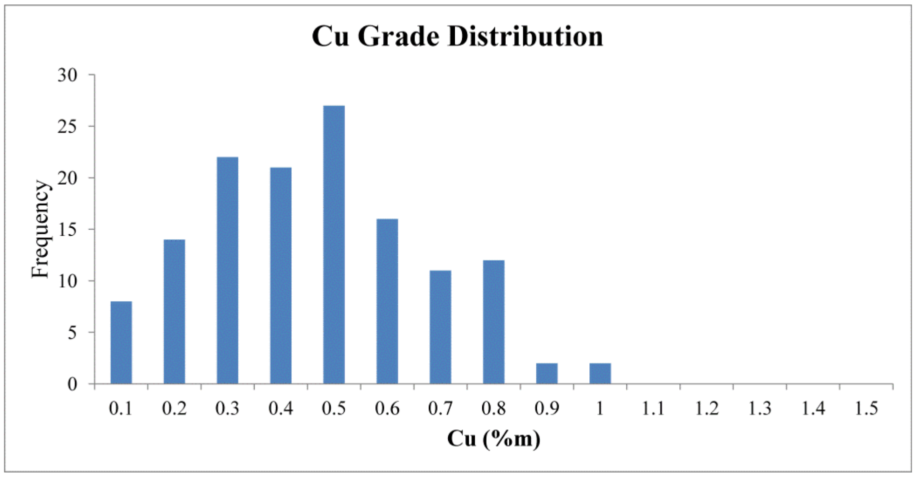


Figure 4.54. Clustered Cu Grade Distribution

Table 4.46. Clustered Grade Descriptive Stats

	Au (gram/tonne)	Cu (%m)
Count	135	135
Mean	0.360	0.411
Median	0.297	0.405
Standard Deviation	0.204	0.209
Sample Variance	0.042	0.044
Kurtosis	0.777	-0.505
Skewness	1.053	0.311
Range	0.955	0.912
Minimum	0.054	0.032
Maximum	1.009	0.943

Table 4.47. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	3,041	1,784
Rock Unity (%)	94.1	7.0
DDF (%)	99.4	2.6
Au CV (%)	41.8	37.2
Cu CV (%)	44.6	36.1

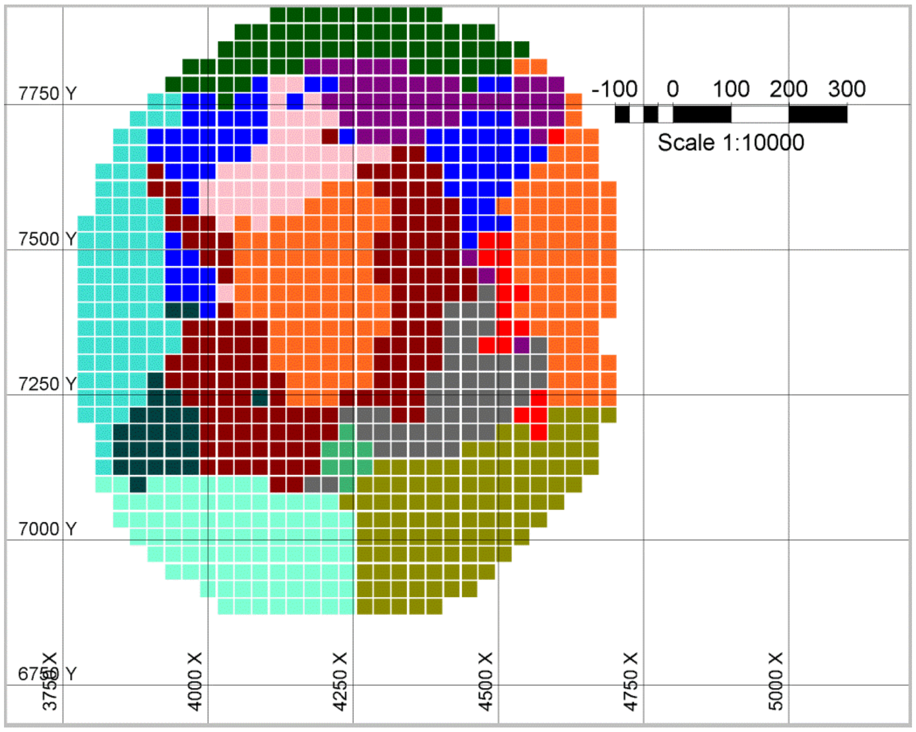


Figure 4.55. Cluster IDs Plan View at 600m Elevation

4.5. Grade Distribution Study

Since the element grades are averaged over clusters, we studied the effects of clustering on grade variations in this section. We expect the variance to drop as we increase the size of the clusters due to averaging out of high grade and low grade blocks. However, according to Rossi and Deutsch (2013), “the averaging is affected by the size and shape of the volume, the continuity of the variable, and the averaging process”. Therefore, we ran multiple clustering settings based on different cluster sizes to understand the effects of our clustering approach on overall variance of grades. In order to be able to compare the histograms, the frequency of occurrence in each histogram bin should be in the same scale. Therefore, after we create the clusters, we apply the average cluster grade to each block and analyse the statistics in block level. Since we deal with mineralized and waste tonnage of each cluster separately, we only apply this to mineralized blocks in each cluster and leave waste blocks with zero grades. In this section, we focus on gold grade as the major element for both clustering and statistical analysis. The same analysis can be applied to copper grade.

The following scenario is based on gold grade and location similarities and is performed based on the settings summarized in Table 4.48. The settings are determined such that clusters with controlled size and mineable shapes are generated. Moreover, we reblock the model based on $5 \times 5 \times 1$ blocks and compare the outcome to our clustering results with 20 blocks per cluster. Following the same logic as in clustering, we distinguish between mineralized and waste blocks when averaging the grade values after reblocking. Figure 4.56 compares the distributions of gold grade for the original block model, clustering results and reblocked model. We removed the first bin (zeros and very small values) in Figure 4.57 to rescale the graph and be able to better see the differences in the distributions.

Table 4.48. Clustering Parameters

Parameter	Value
Distance Weight	1
Grade Weight	1
Rock-type Penalty	1
Avg. Blocks per Cluster	20
Max. Blocks per cluster	30
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	3

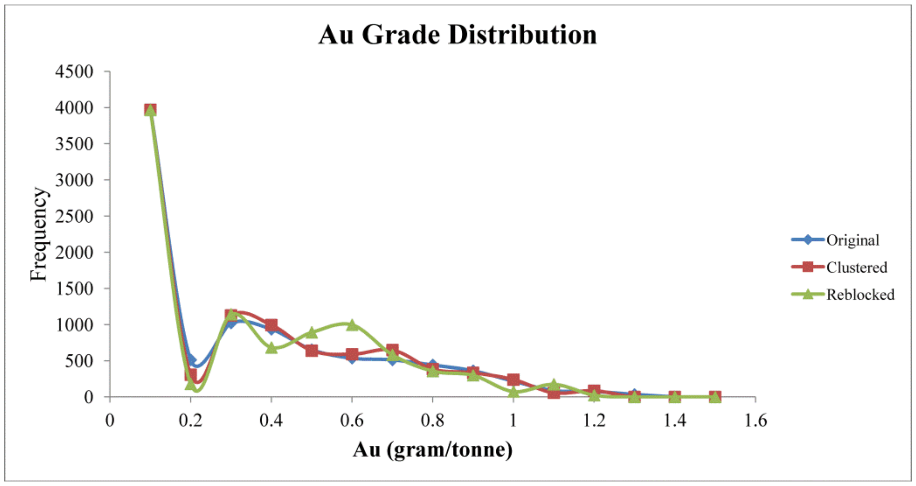


Figure 4.56. Clustered Au Grade Distribution

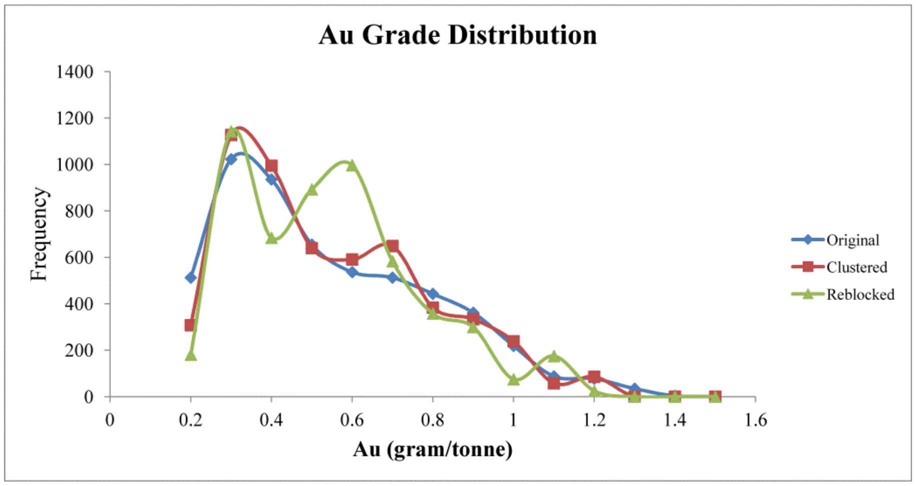


Figure 4.57. Clustered Au Grade Distribution

As we expected, there are less instances of high grade (larger than 1.2%) and low grade (smaller than 0.2%) are occurring in both clustering and reblocked results compared to the original block model. Similarly, there are more instances of around the average (0.3%) in the aggregated blocks than the original blocks. However, the reblocked model is showing more changes in the shape of the grade distribution and more fluctuations in the values. In contrast, the clustered distribution is smoother and, to some extent, following the original distribution of the grade values. Moreover, the drop in the grade variance is 5% in clustering results compared to 11% in the reblocked model. The statistical summary of the clustering results and reblocked model can be found in Table 4.49. Note that the mean of the distribution did not change since averaging of the grade values is a linear operation. Changes in other statistical measures support the idea that clustering is following the original distribution better than reblocking. A sample plan view of the clustering scheme is presented in Figure 4.58. We can compare sample plan views of the gold grade values in the block level, clustered level and reblocked level in Figure 4.59, Figure 4.60 and Figure 4.61 respectively.

Table 4.49. Clustered and Reblocked Grade Descriptive Stats

	Au (gram/tonne)				
	Original	Clustered	Diff%	Reblocked	Diff%
Count	9,381	9,381	0%	9,381	0%
Mean	0.285	0.285	0%	0.285	0%
Median	0.223	0.236	6%	0.247	11%
Standard Deviation	0.313	0.305	-3%	0.295	-6%
Sample Variance	0.098	0.093	-5%	0.087	-11%
Kurtosis	-0.200	-0.405	102%	-0.591	195%
Skewness	0.866	0.773	-11%	0.648	-25%
Range	1.417	1.199	-15%	1.111	-22%
Minimum	0.000	0.000	0%	0.000	0%
Maximum	1.417	1.199	-15%	1.111	-22%

Table 4.50. Clustering Summary

	Mean	Standard Deviation
Cut Tonnage (K)	1,225	608
Rock Unity (%)	85.8	17.5
DDF (%)	93.5	12.3
Au CV (%)	40.6	43.6
Cu CV (%)	43.5	42.4

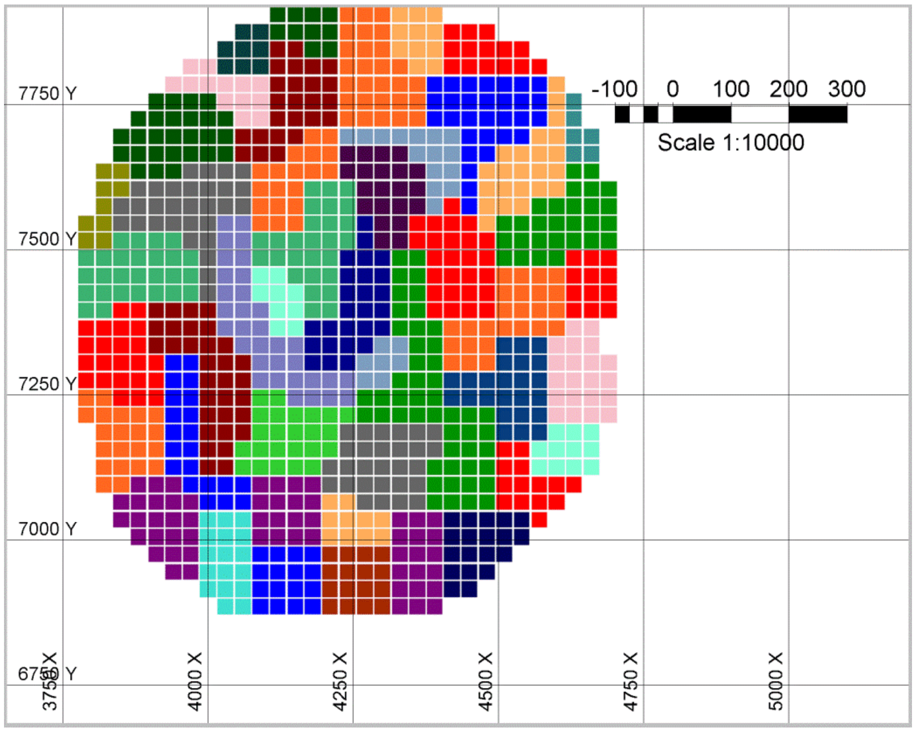


Figure 4.58. Cluster IDs Plan View at 600m Elevation

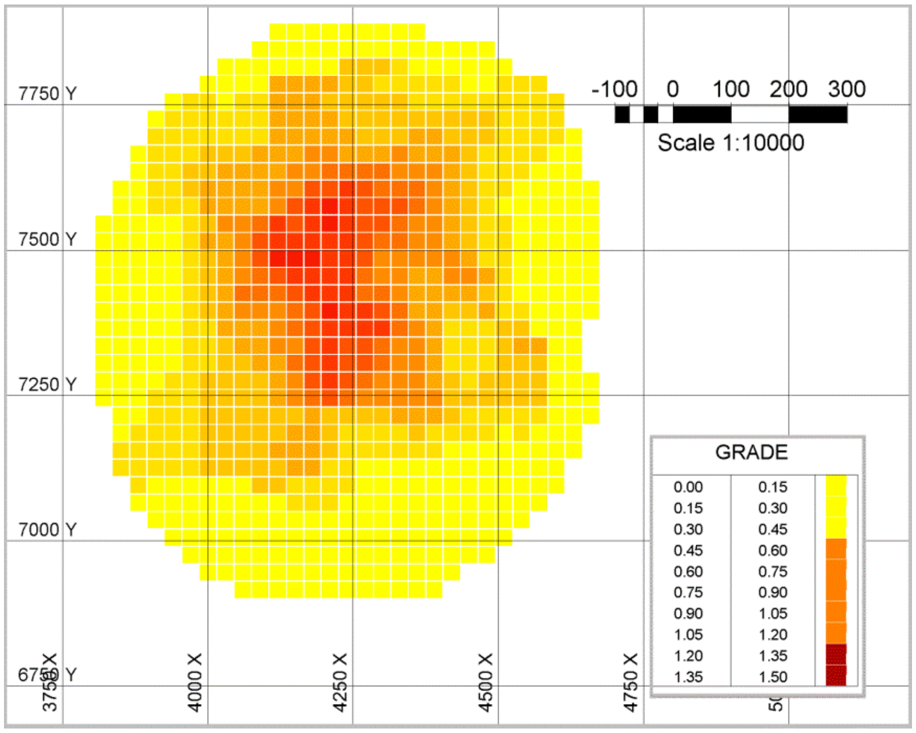


Figure 4.59. Gold Grade Distribution Plan View at 600m Elevation

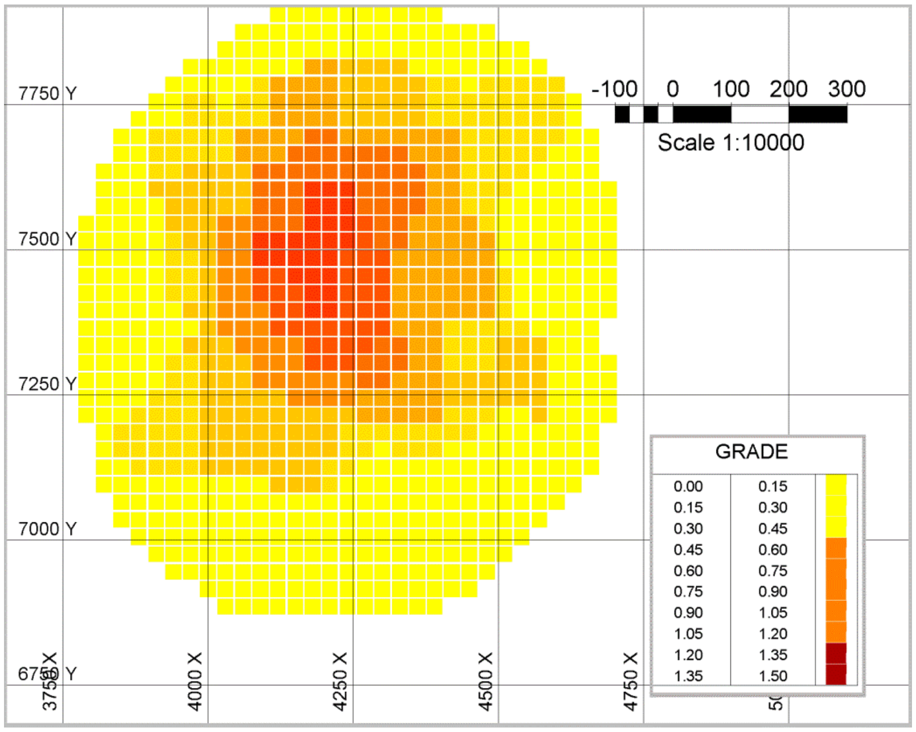


Figure 4.60. Clustered Gold Grade Distribution Plan View at 600m Elevation

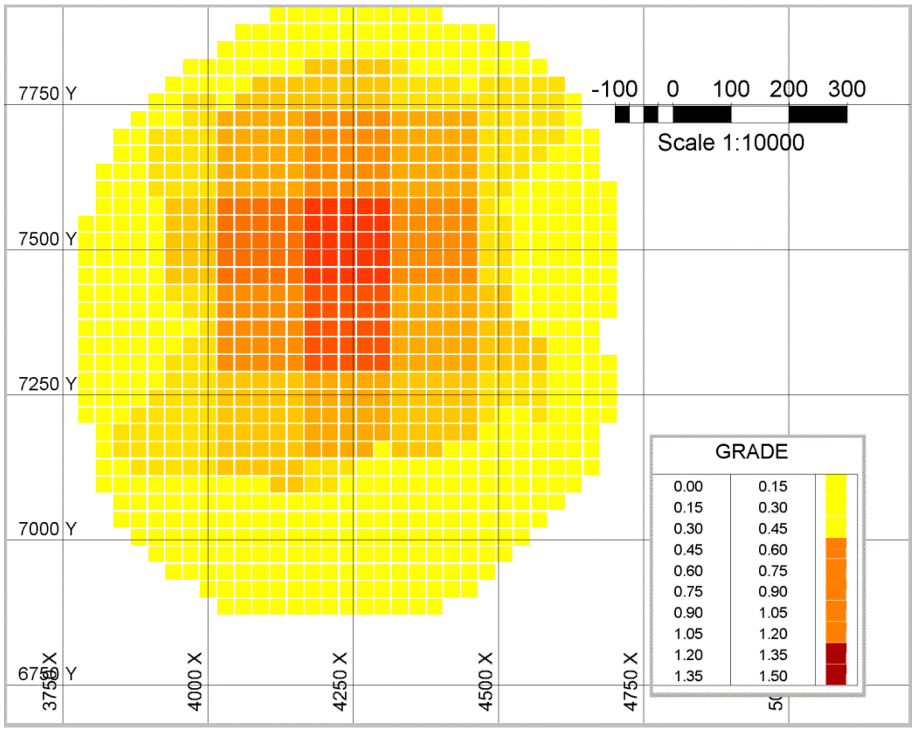


Figure 4.61. Reblocked Gold Grade Distribution Plan View at 600m Elevation

Afterwards, we increase the size of clusters from 20 blocks per cluster to 40, 60, 80 and 100 blocks per cluster. The grade variance is calculated for each setting and divided by the original (block level) grade variance to calculate the Variance Correction Factor (VCF) proposed by Rossi and Deutsch (2013). Similarly, we performed reblocking based on $6 \times 6 \times 1$, $8 \times 8 \times 1$, $9 \times 9 \times 1$ and $10 \times 10 \times 1$. The changes in VCF are plotted against average blocks per cluster and reblocking size to show the trends and differences between clustering and reblocking. As can be seen in Figure 4.62, the difference between clustering and reblocking becomes more significant by increasing the size of the aggregates and clustering results in smaller drops in grade variance in all scenarios.

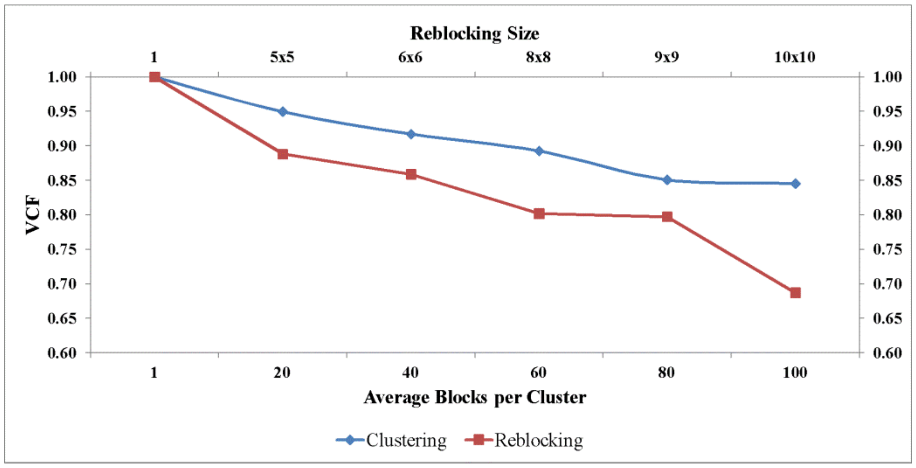


Figure 4.62. VCF Changes for Clustering and Reblocking

4.6. Clustering Summary

We studied various clustering scenarios with different settings on the Marvin dataset. In this section, we summarize the studies and the effects of various parameters on the clustering outcome. Table 4.51 represents the summary of the clustering outcomes. The first column is the scenario ID used in Table 4.3 to define the parameters of clustering for various scenarios. The next three columns are the average, standard deviation and coefficient of variation for cluster tonnages. Average rock unity and grade coefficients of variation are presented in columns five to seven.

Table 4.51. Clustering Scenarios Summary

ID	Tonnage			RU (%)	Au CV (%)	Cu CV (%)
	Avg.	Std.	CV (%)			
D	909	382	42	86	49	48
G	909	5,662	623	96	24	27
R	909	3,942	433	98	22	20
DG	909	1,520	167	92	42	45
DR	909	584	64	94	37	36
DGR	909	1,528	168	94	54	56
TS	909	1,205	133	90	50	48
DR30	1,790	1,703	95	94	42	40
DR60	3,504	4,489	128	94	50	45
DR90	5,177	6,783	131	93	52	47
DRM	674	529	79	93	30	30
SR	862	333	39	91	38	38
BND	819	349	43	91	37	37
DV	2,624	1,004	38	86	50	47
DS	2,698	1,149	43	89	48	44
DSR	2,737	1,141	42	88	47	43
K-Means	3,041	1,784	59	94	42	45

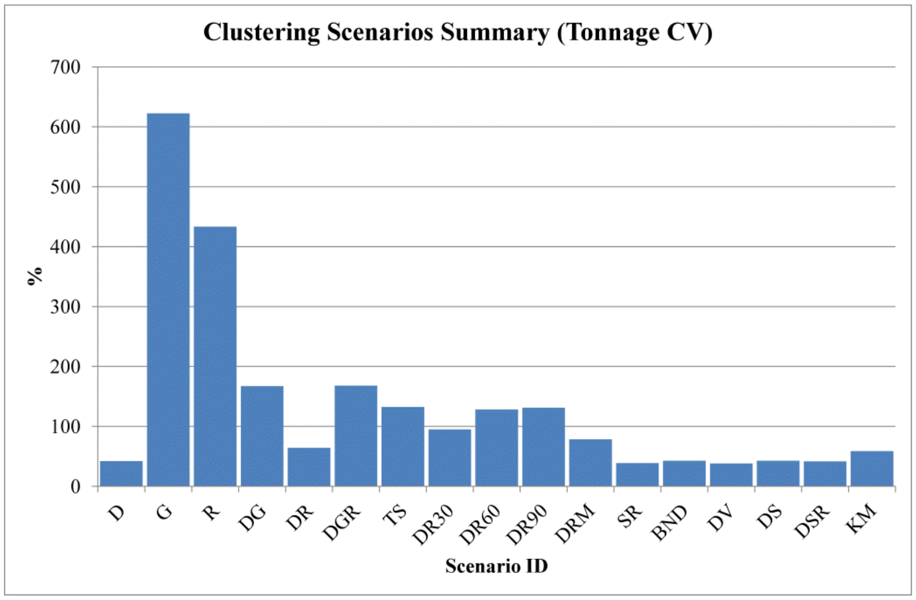


Figure 4.63. Clustering Scenarios Summary (Tonnage CV)

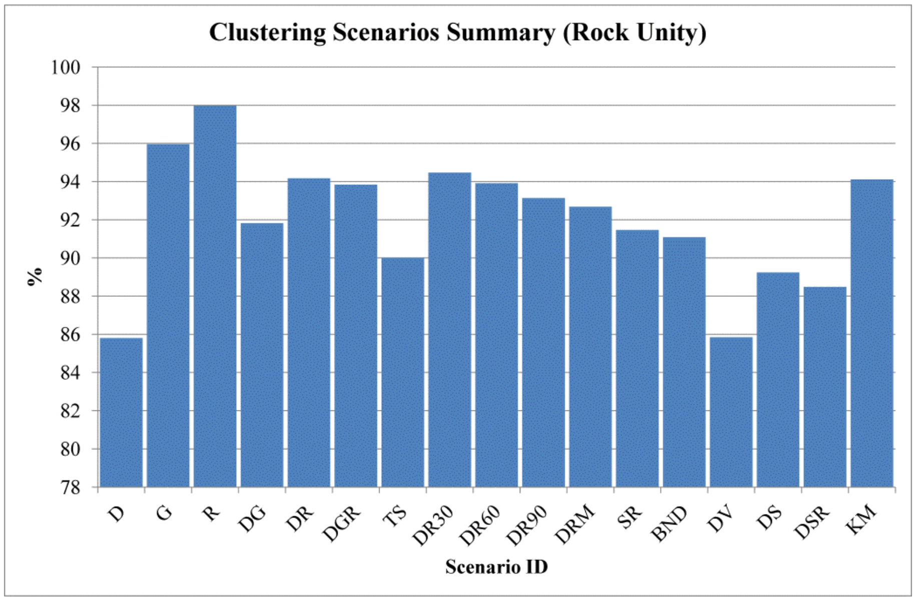


Figure 4.64. Clustering Scenarios Summary (Rock Unity)

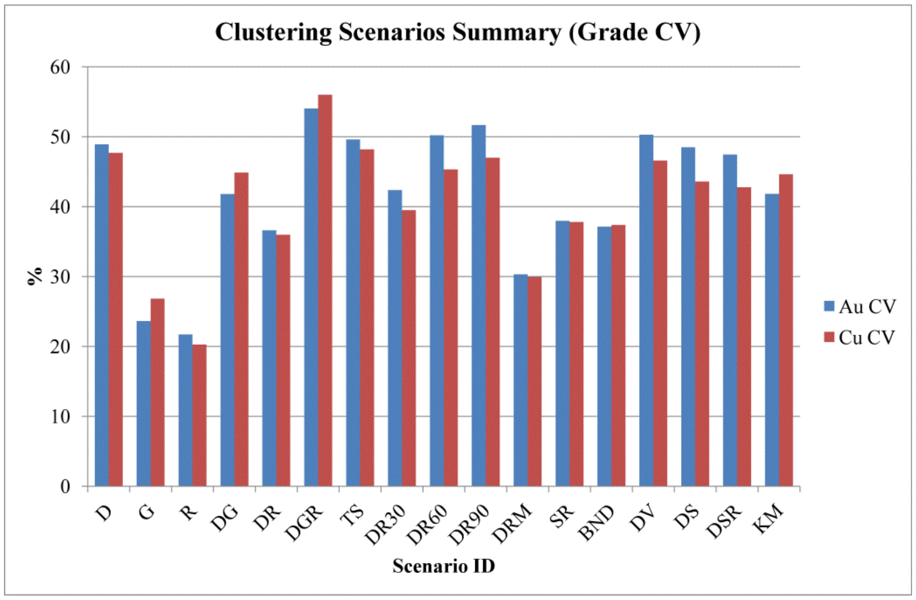


Figure 4.65. Clustering Scenarios Summary (Grade CV)

Based on comparing the outcomes of various clustering scenarios we can understand the effects of different parameters on the outcome of clustering. Although these understandings are specific to the Marvin deposit, they can be generalized to some extent. Here are some of the observations that we expected and can be generalized:

1. Not including block distance in clustering results in significant variations in size and shape of the generated clusters (cases G and R in Figure 4.63).
2. Clustering based on either grade or rock type results in better clustering measure compared to including both without tuning the parameters (compare DG and DR against DGR in Figure 4.63 to Figure 4.65).
3. Even though intra-cluster similarity is considered in defining Tabu search measure of goodness, running Tabu search damages the clustering evaluation measures (compare DR against TS in Figure 4.63 to Figure 4.65).
4. Increasing the size of the generated clusters will decrease the outcome quality (compare DR30, DR60 and DR90 in Figure 4.63 to Figure 4.65).
5. Adding constraints on the size and shape of the generated clusters decreases the quality measure we aimed for (DRM and SR have lower rock unity and higher tonnage variation than DR even though all have rock type and distance as the similarity factor).
6. Similarly, adding direction to clustering parameters damages the quality measures of clustering (compare DV and DS against DR, and DSR against SR in Figure 4.63 to Figure 4.65).
7. Similarly, adding boundary constraints for clustering will decrease the quality of the generated clusters (compare DR against BND in Figure 4.63 to Figure 4.65).

On the other hand, there are observations that we believe are specific to this case-study and should not be generalized.

1. Clustering based on rock type resulted in less variation in grade than clustering based on grade. This can be due to the fact that Marvin is a synthetic dataset and does not have significant spatial variations in grade as other real deposits may have.
2. Including more than two features in the definition of the similarity index results in lower performance measures for both features. However, fine tuning the weights and penalties (as shown in the upcoming case-studies) may result in clustering schemes that have relatively good performances on all measures.

4.7. Long-term Open pit Production Planning

In this section we are solving the LTOPP problem in different resolutions and comparing the results with Whittle™ (GEOVIA, 2014b) mine planning software. This is a comparative analysis to have a benchmark against a tool that is commonly used in the industry. Although Whittle™ is commercial software and we do not fully know the details of their heuristic scheduling algorithms, we use it as a benchmark of what is common in the industry. Firstly, we present the results we obtained from Whittle™ (GEOVIA, 2014b) mine planning software. Since our MILP works with different units for making mining and processing decisions we will use six different sets of mining and planning units for the MILP. In this section, our goal is to understand the effects of using different resolutions for planning and find out which resolution is more practical and closer to reality of the operation. The first setting is to use blocks for making processing decisions and bench-phases for making mining decisions (section 4.7.2). This is the scenario that resembles the decision making in Whittle™ (GEOVIA, 2014b). Afterwards, we create clusters within pushback boundaries and use bench-phases as the mining units (section 4.7.3). In this setting, the processing decisions are made based on the clusters within those bench-phases. The third setting, in section 4.7.4, is using the same clusters for making both mining and processing decision. In the next section, we compare two sets of the results obtained based on using the clustering algorithm and the Tabu search. In order to improve the performance of the Tabu search algorithm, we consider the beneath cluster as one of the similarity indices and then obtain the results before and after running the Tabu search. In section 4.7.6, mining decisions are made based on clusters of blocks created with the agglomerative hierarchical clustering algorithm and processing decisions are made based on blocks within those clusters. The same clusters from section 4.7.5 are used for making mining decisions. We used the after Tabu search clusters to decrease the number of precedence arcs generated and decrease the solution time for the MILP. Afterwards, in section 4.7.7, we use blocks for making mining and processing decisions in order to evaluate the processing time required to solve the model and comment on the quality of the generated solution in the original block level modeling. Since using clusters as processing units and bench-phases as mining units creates models that are easier and faster to solve, we test the stockpiling scenario on this resolution (section 4.7.8). Moreover, using these sets of processing and mining units is closer to reality and does not over-estimate the value of the operation.

4.7.1 *GEOVIA Whittle™ Schedule*

We obtained two production schedules based on four pushbacks from Whittle™. The first schedule is based on Milawa NPV algorithm and the second one is based on Milawa Balanced. These algorithms are heuristics designed to use pushbacks as mining units and maximize NPV of the operation. Milawa NPV focuses on maximizing NPV where Milawa

Balanced looks into maximizing NPV and having a balanced schedule at the same time. The Milawa balanced algorithm results in \$2,166M of NPV and the schedule in Figure 4.66. The Milawa NPV algorithm results in \$2,240M of NPV and the schedule in Figure 4.69. As expected, the Milawa NPV algorithm resulted in higher NPV by sacrificing the balance in utilizing resources.

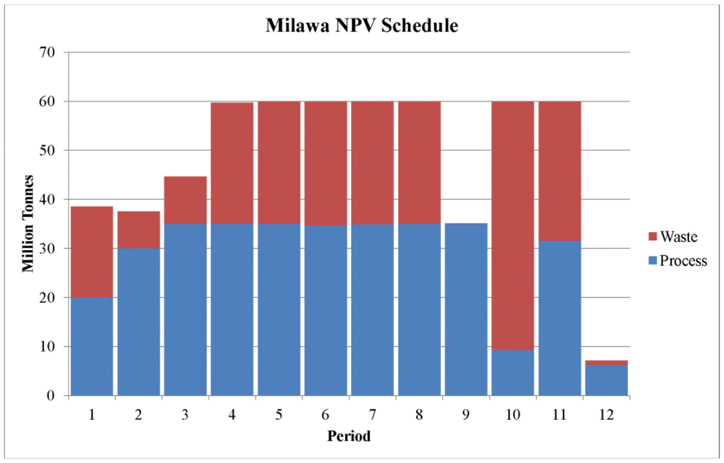


Figure 4.66. Milawa NPV Schedule

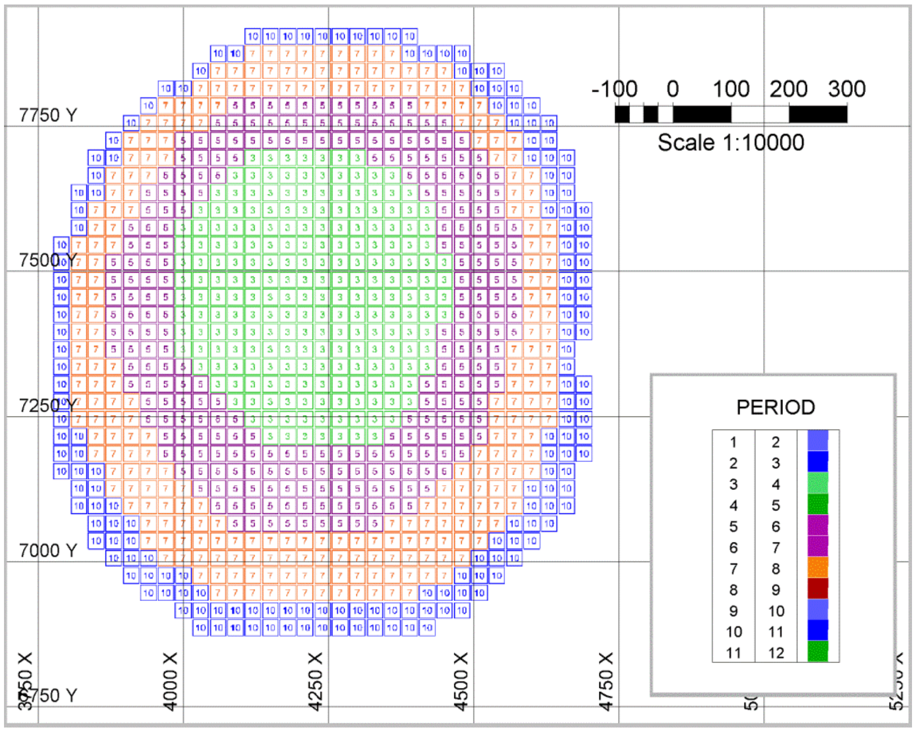


Figure 4.67. Milawa NPV Schedule Plan View at 600m Elevation

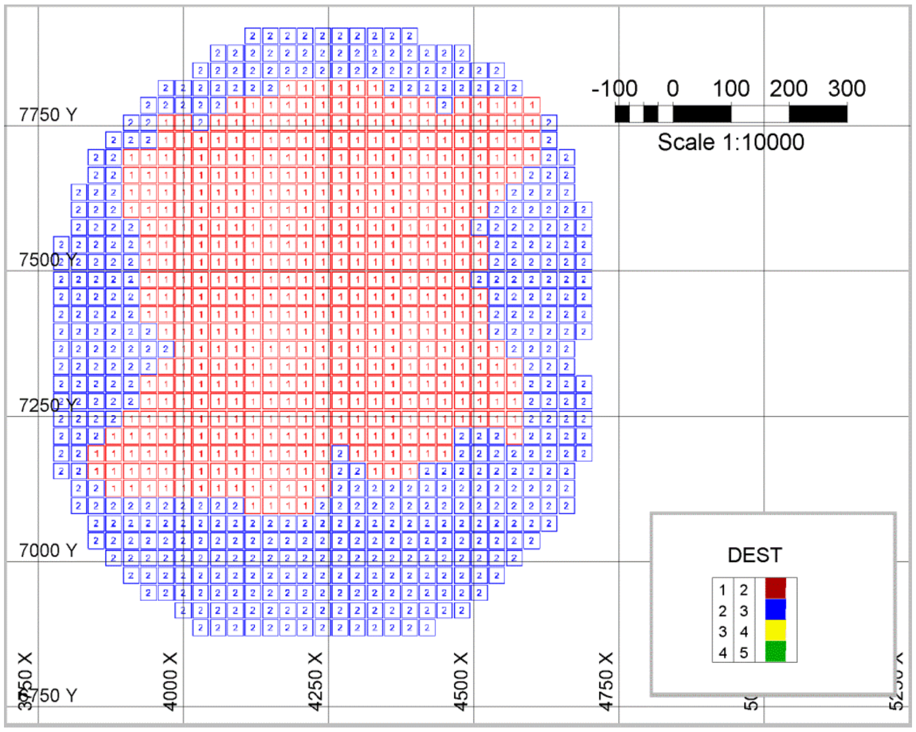


Figure 4.68. Milawa NPV Destination Plan View at 600m Elevation

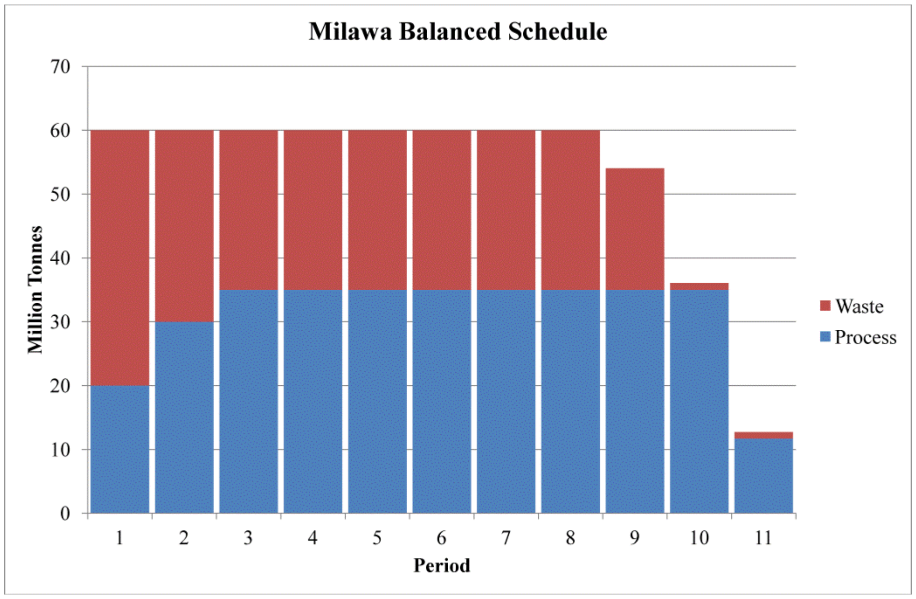


Figure 4.69. Milawa Balanced Schedule

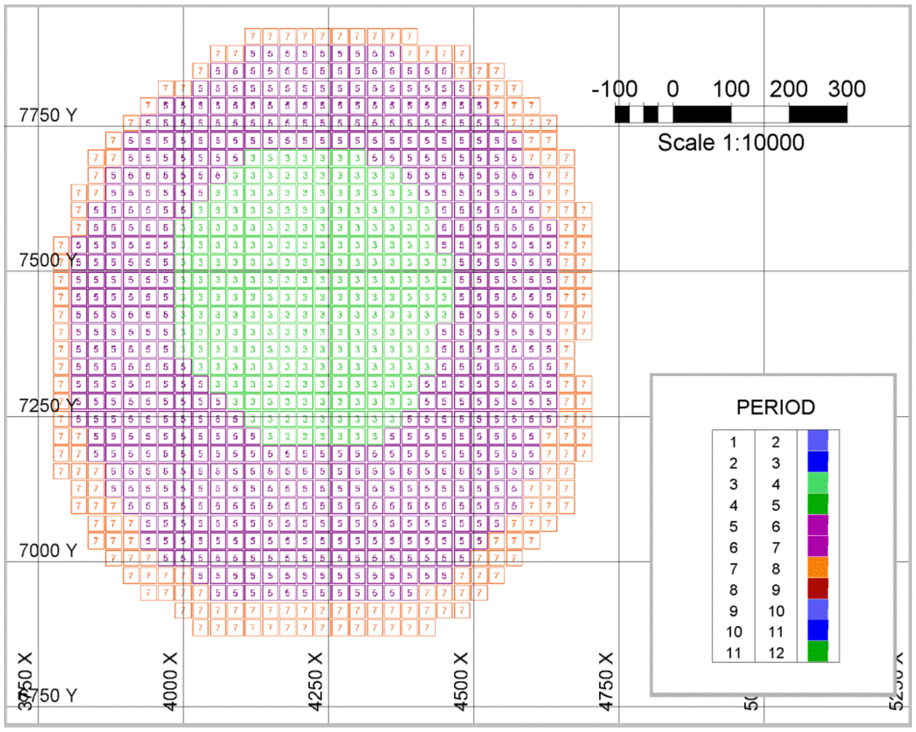


Figure 4.70. Milawa Balanced Schedule Plan View at 600m Elevation

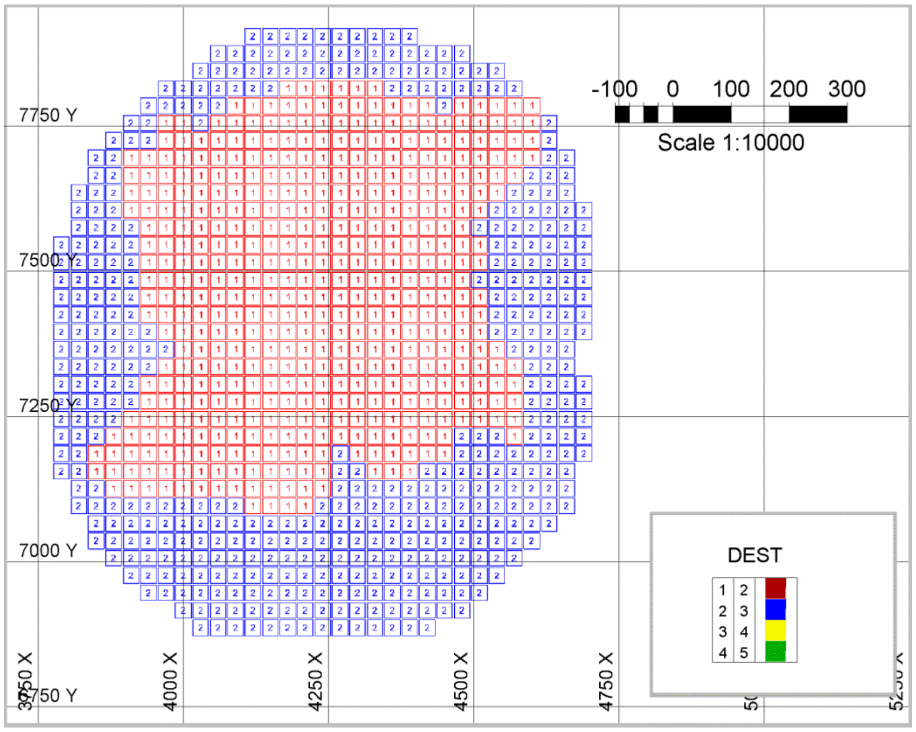


Figure 4.71. Milawa Balanced Destination Plan View at 600m Elevation

4.7.2 *Mining Units: Bench-phases, Processing Units: Blocks*

Since the Whittle™ scheduling is performed based on using bench-phases as mining units and blocks as processing units, we used the same resolution for the first case-study. We first ran the MILP model to 5% optimality gap. The results were obtained in 7 seconds and an NPV of $2,653M is reached. Since the runtime was short we ran the model to optimality and obtained the optimal solution in this resolution in 322 seconds. The optimal NPV is $2,660M and the corresponding schedule graph and plan views are presented in Figure 4.72 to Figure 4.74. Although the block destinations are very similar to the Whittle™ results, the NPV of the operation shows 18.8% improvement over the highest NPV obtained by implementing Milawa NPV scheduling algorithm.

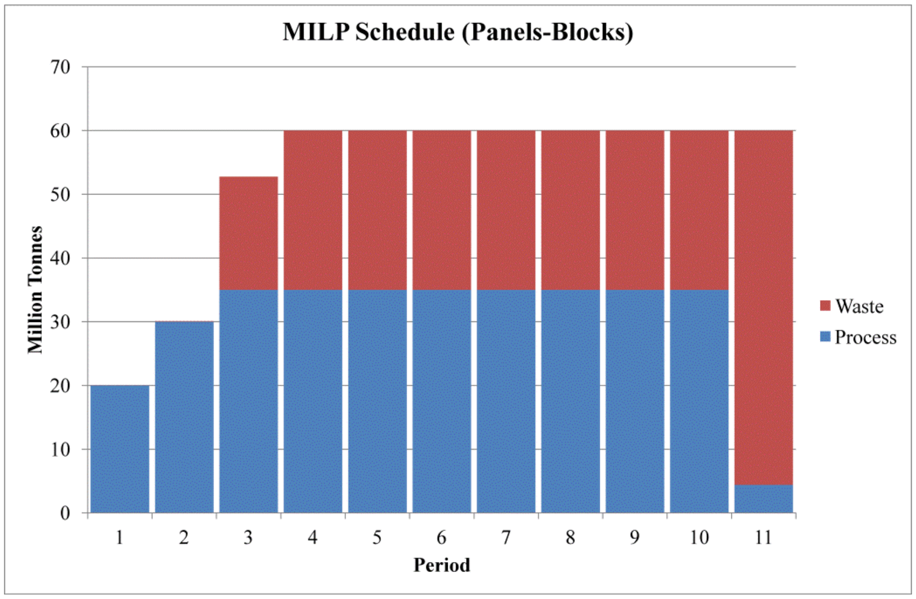


Figure 4.72. MILP Schedule

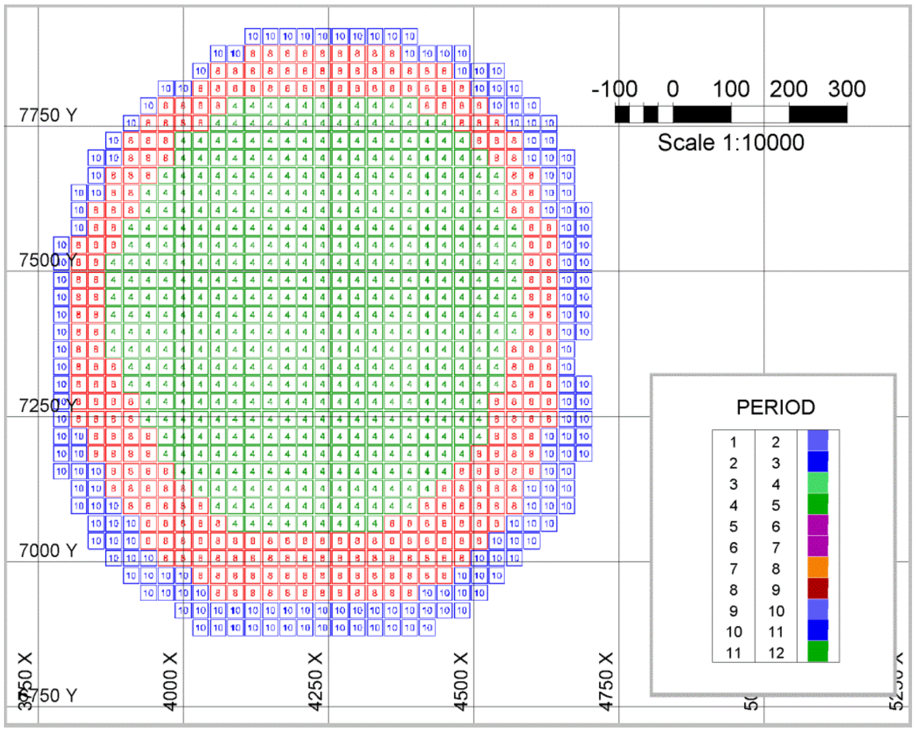


Figure 4.73. MILP Schedule Plan View at 600m Elevation

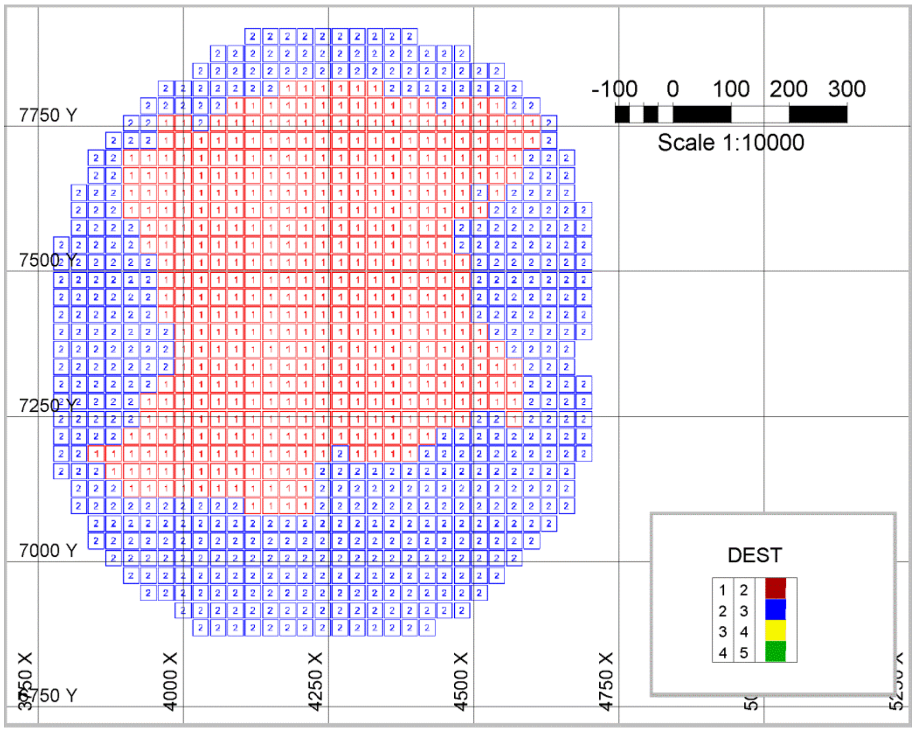


Figure 4.74. MILP Destination Plan View at 600m Elevation

4.7.3 *Mining Units: Bench-phases, Processing Units: Clusters*

The focus of this project is on creating clusters of blocks with homogenous grade and rock-type and using the clusters as mining and processing units. Therefore, we use bench-phases as mining units and clusters as processing units in this scenario to evaluate the effects of clustering on the mine planning outcomes. The hierarchical clustering algorithm in this scenario is performed based on the parameters summarized in Table 4.52. It takes the solver 1.6 seconds to solve the MILP formulation to 10% gap with an NPV of $2,136M. Solving the MILP to optimality takes 30.5 seconds and results in an NPV of $2,185M which is 3.5% less than the Milawa NPV algorithm but with a more balanced schedule. The schedule in presented in Figure 4.75 and the clusters, extraction periods and destination plan views follow in Figure 4.76 to Figure 4.78.

Table 4.52. Clustering Parameters

Parameter	Value
Distance Weight	0.5
Grade Weight	0
Rock-type Penalty	0.5
Avg. Blocks per Cluster	20
Max. Blocks per cluster	25
Min. Blocks per Cluster	5
Number of Shape Refinement Iterations	3

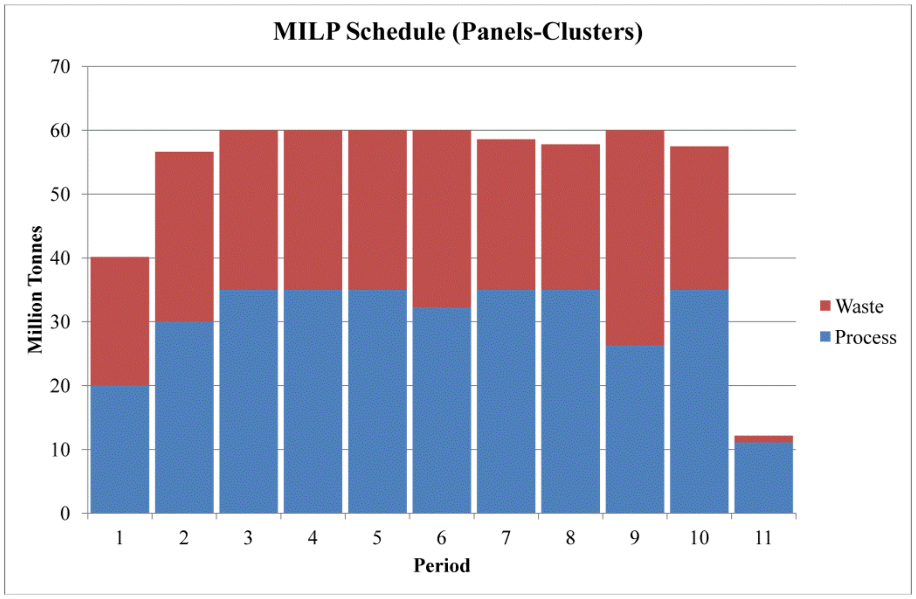


Figure 4.75. MILP Schedule

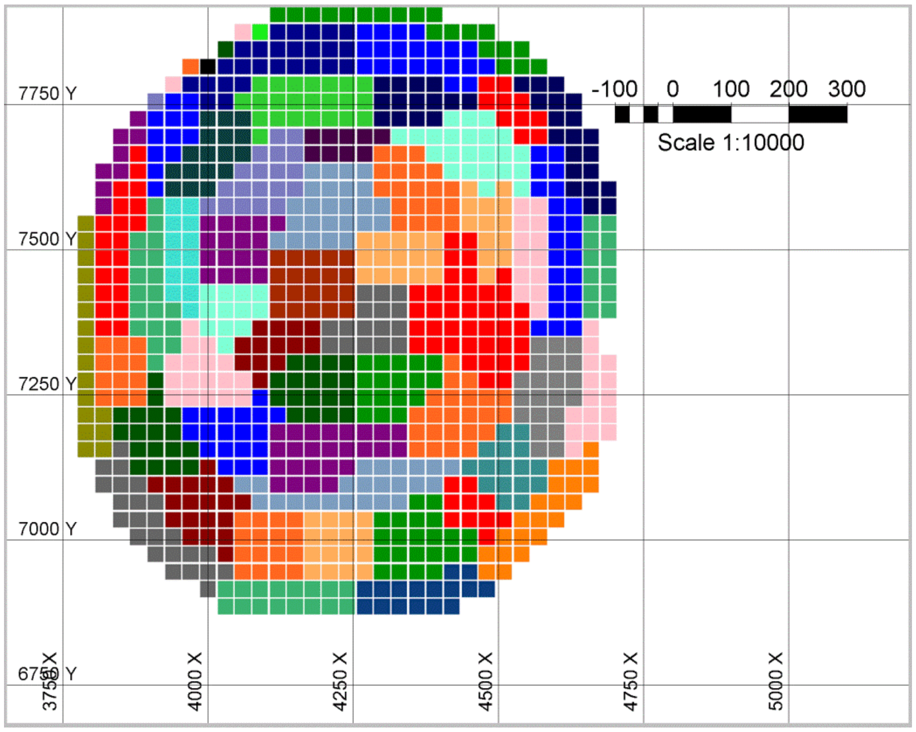


Figure 4.76. Cluster IDs Plan View at 600m Elevation

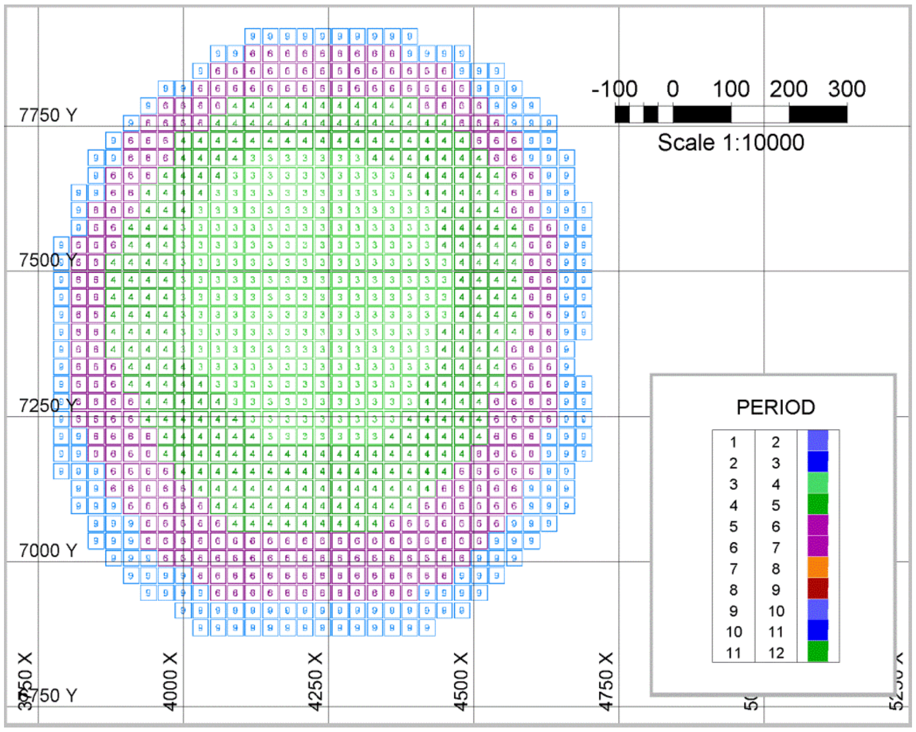


Figure 4.77. MILP Schedule Plan View at 600m Elevation

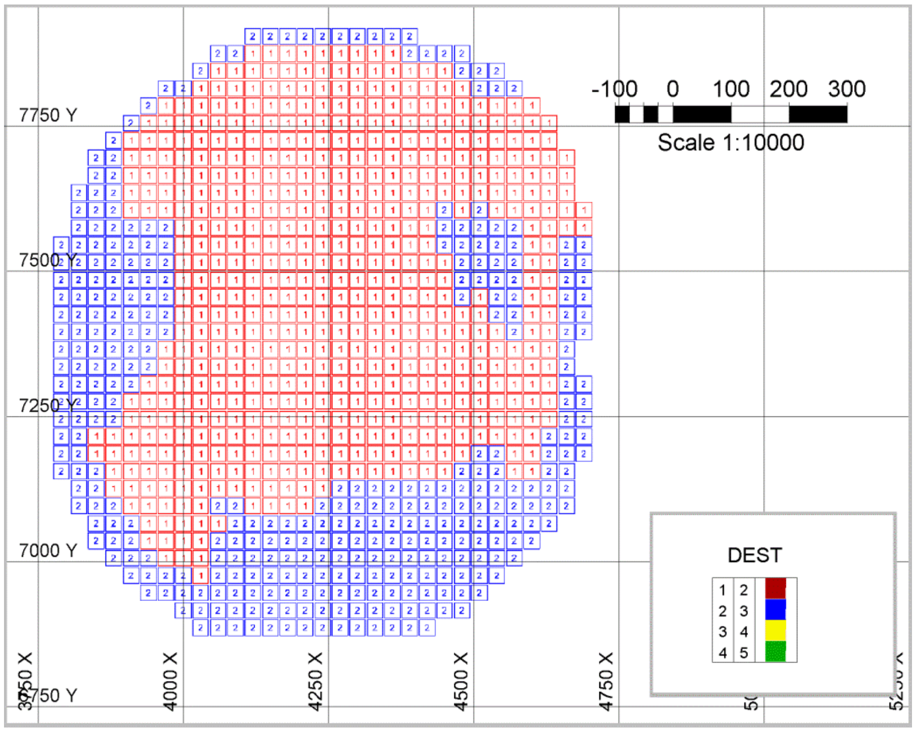


Figure 4.78. MILP Destination Plan View at 600m Elevation

4.7.4 *Mining Units: Clusters, Processing Units: Clusters*

We used the same clusters as in the previous section as both mining and processing units. The clustering parameters are the same as Table 4.52. We solved the MILP based on this resolution and obtained a 10% gap solution within 1,435 seconds. The resulted NPV is $2,006M which is 10.4% less than Milawa NPV. The resulted schedule is plotted in Figure 4.79 and the schedule and destination plots are presented in and respectively.

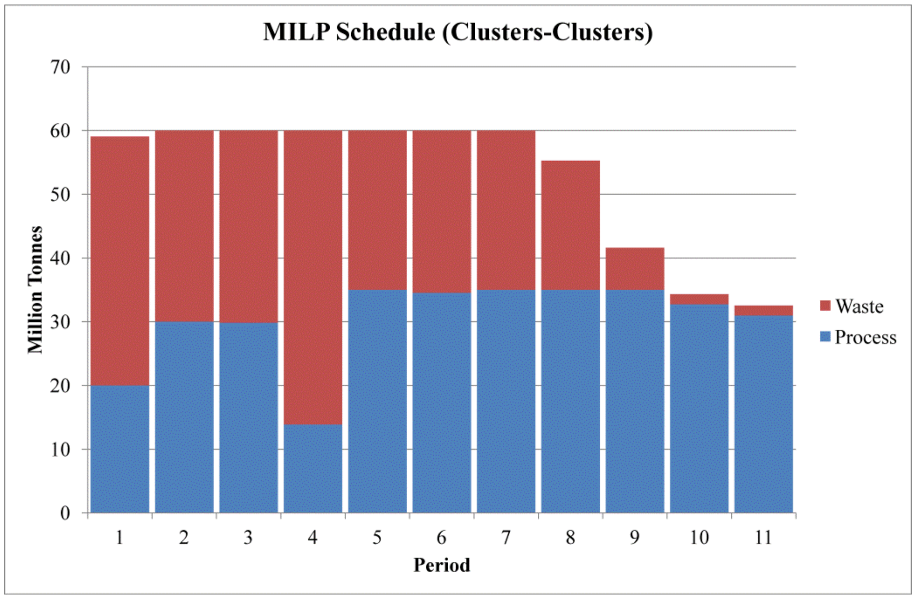


Figure 4.79. MILP Schedule

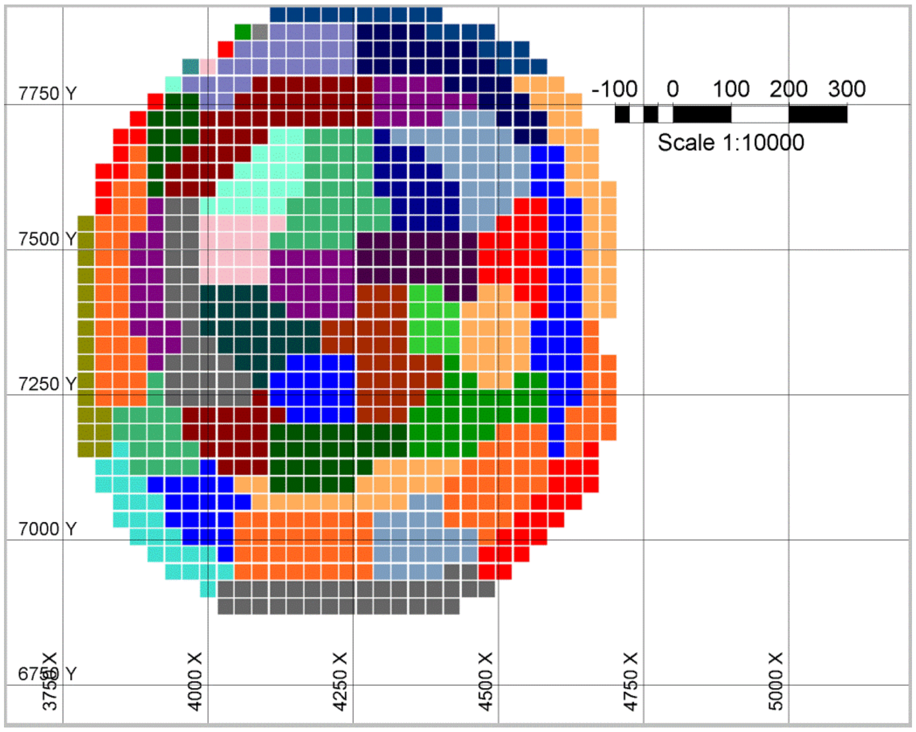
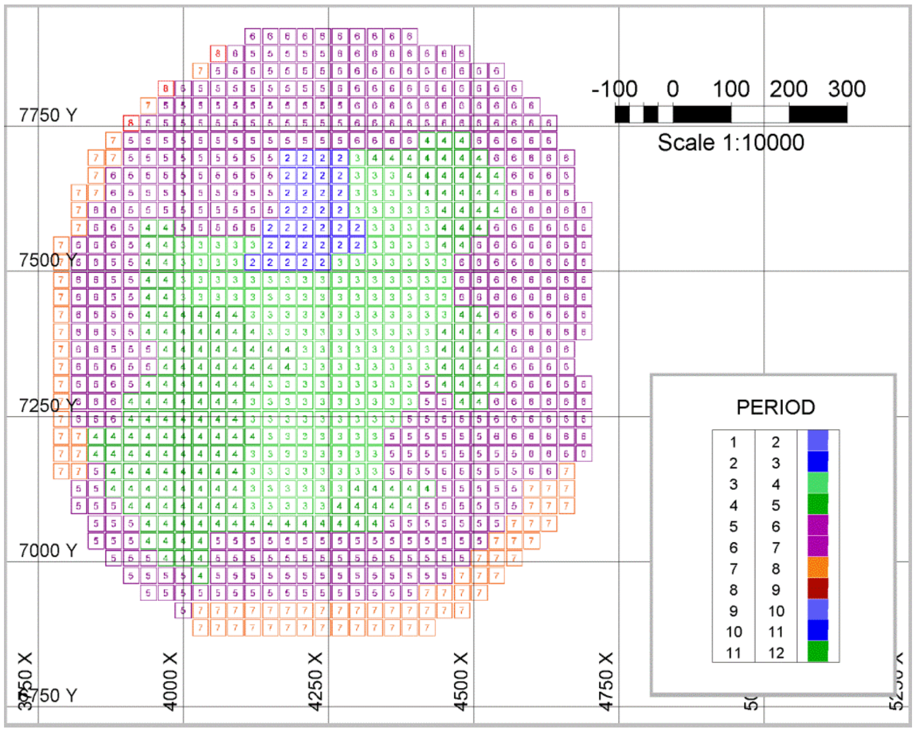


Figure 4.80. Cluster IDs Plan View at 600m Elevation



1	2	🟦
2	3	🟦
3	4	🟩
4	5	🟩
5	6	🟪
6	7	🟪
7	8	🟧
8	9	🟥
9	10	🟦
10	11	🟦
11	12	🟩

Figure 4.81. MILP Schedule Plan View at 600m Elevation

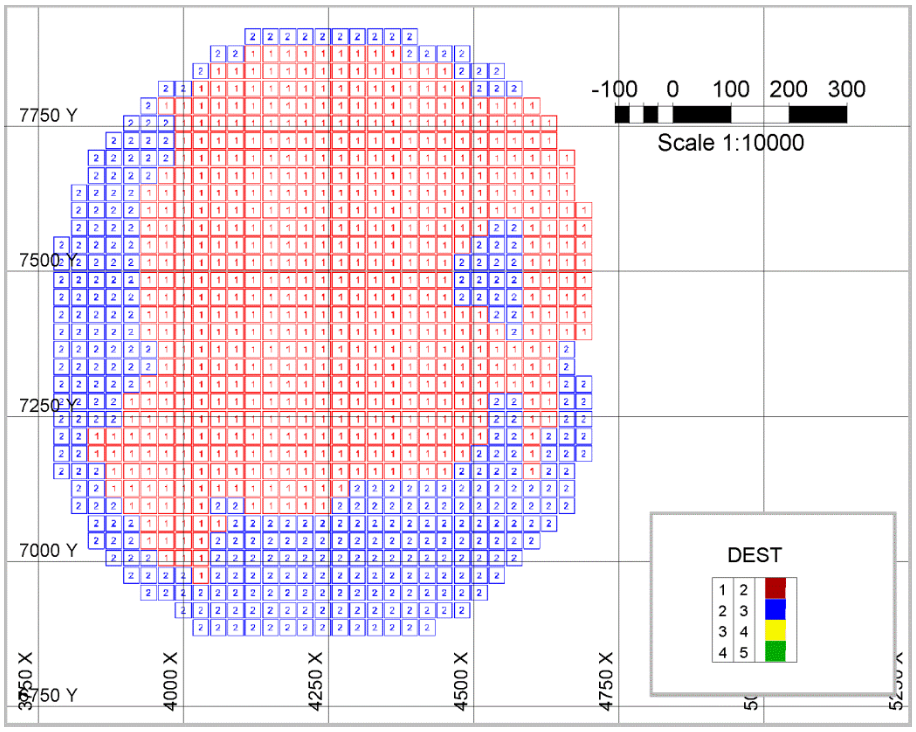


Figure 4.82. MILP Destination Plan View at 600m Elevation

4.7.5 *Mining Units: Clusters, Processing Units: Clusters, With Tabu Search*

The clustering algorithm in the previous section is performed with emphasis on homogeneity of the clusters and is not looking at the relationship between clusters of consequent benches. Therefore, it can result in numerous precedence arcs between the mining units and increase the solution time. In this section, we included beneath cluster as a similarity factor to create clusters on top of each other (Table 4.53). The MILP was solved to 10% gap in 804 seconds which is around two times faster than the previous run. The resulted NPV has also increased slightly to $2,051M. The resulted schedule and the plan views are presented in Figure 4.83 to Figure 4.86. Afterwards, we ran the Tabu search procedure on the same clusters, formulated and solved the MILP model. Solving the MILP model to 10% gap took 239 seconds and resulted in an NPV of $2,042M. The Tabu search procedure decreases the solution time for the model to get to 10% gap. The resulted schedule is presented in Figure 4.87 and the plan views are in Figure 4.88 to Figure 4.90.

Table 4.53. Clustering Parameters

Parameter	Value
Distance Weight	0.3
Grade Weight	0
Rock-type Penalty	0.3
Beneath Cluster Penalty	0.3
Avg. Blocks per Cluster	20
Max. Blocks per cluster	25
Min. Blocks per Cluster	0
Number of Shape Refinement Iterations	0

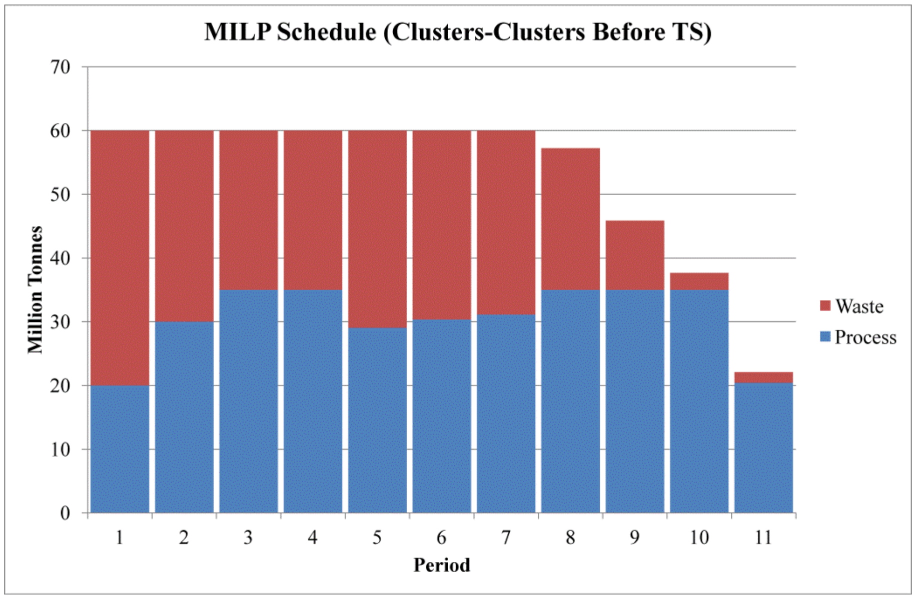


Figure 4.83. MILP Schedule

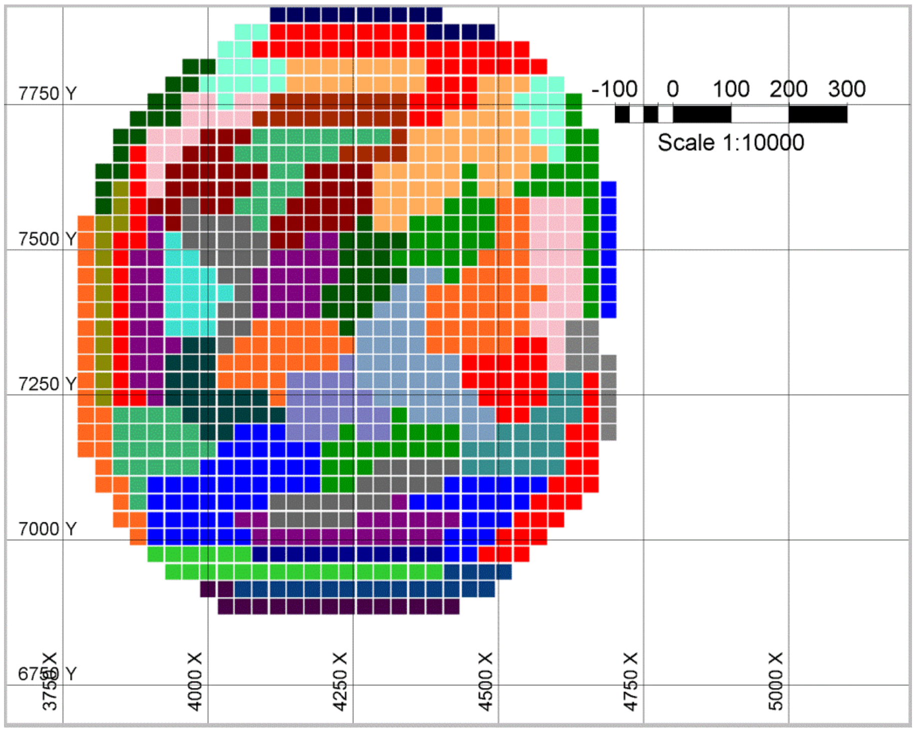


Figure 4.84. Cluster IDs Plan View at 600m Elevation

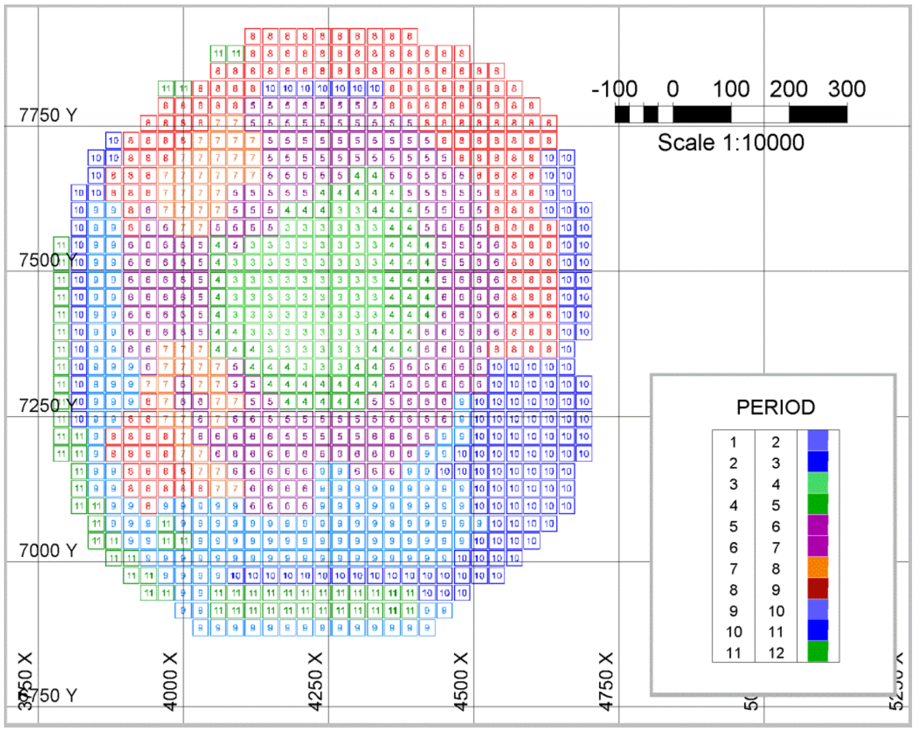


Figure 4.85. MILP Schedule Plan View at 600m Elevation

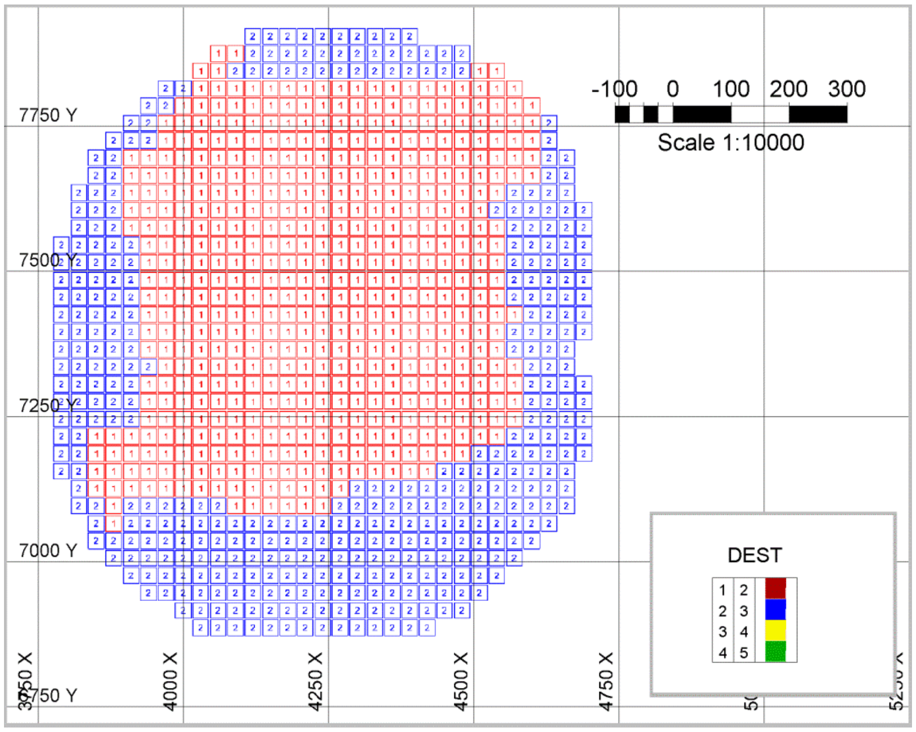


Figure 4.86. MILP Destination Plan View at 600m Elevation

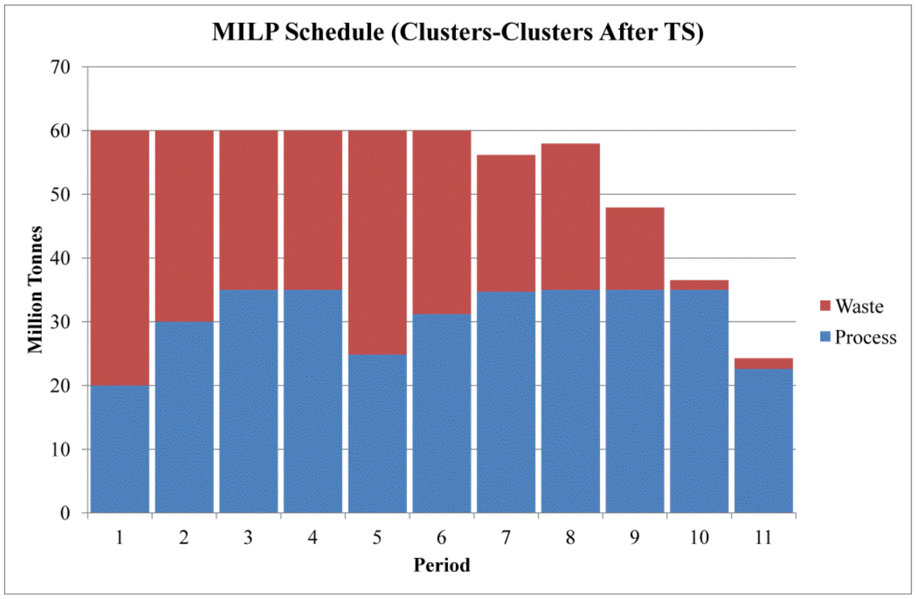


Figure 4.87. MILP Schedule

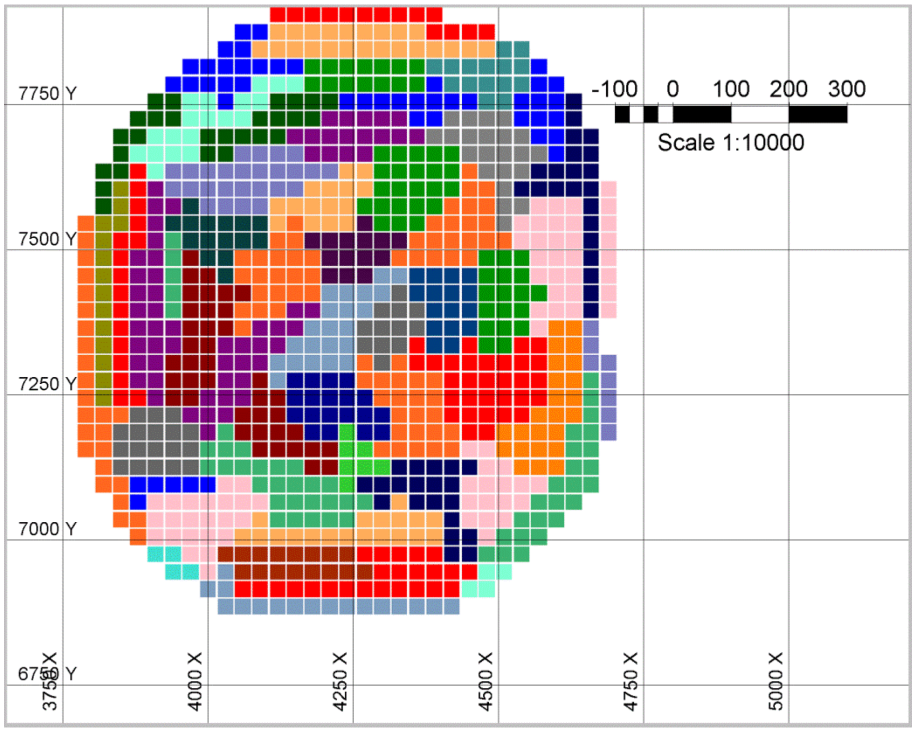


Figure 4.88. Cluster IDs Plan View at 600m Elevation

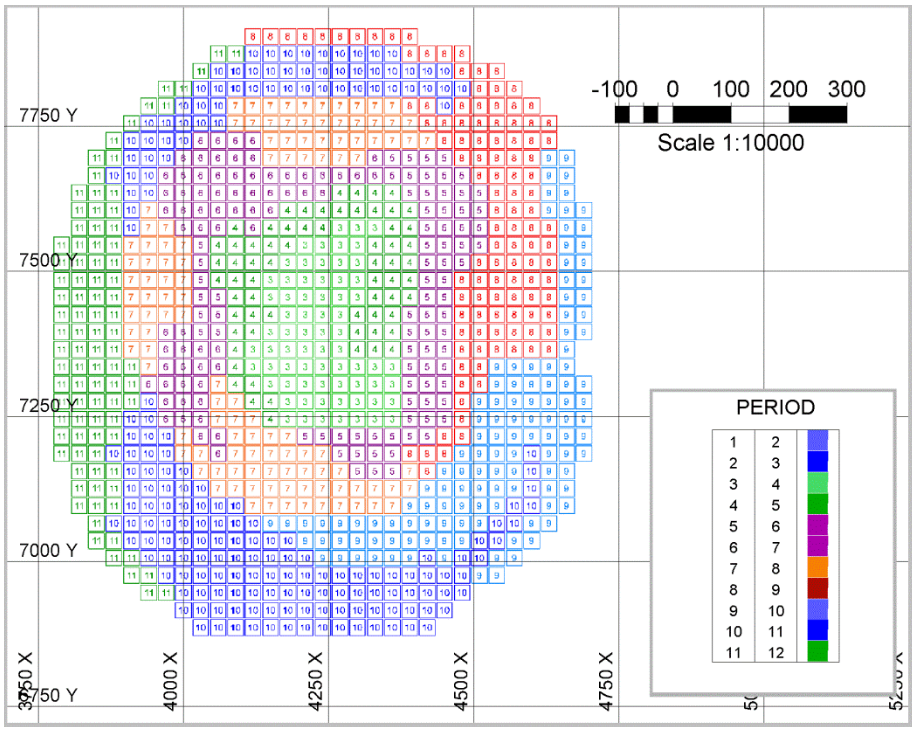


Figure 4.89. MILP Schedule Plan View at 600m Elevation

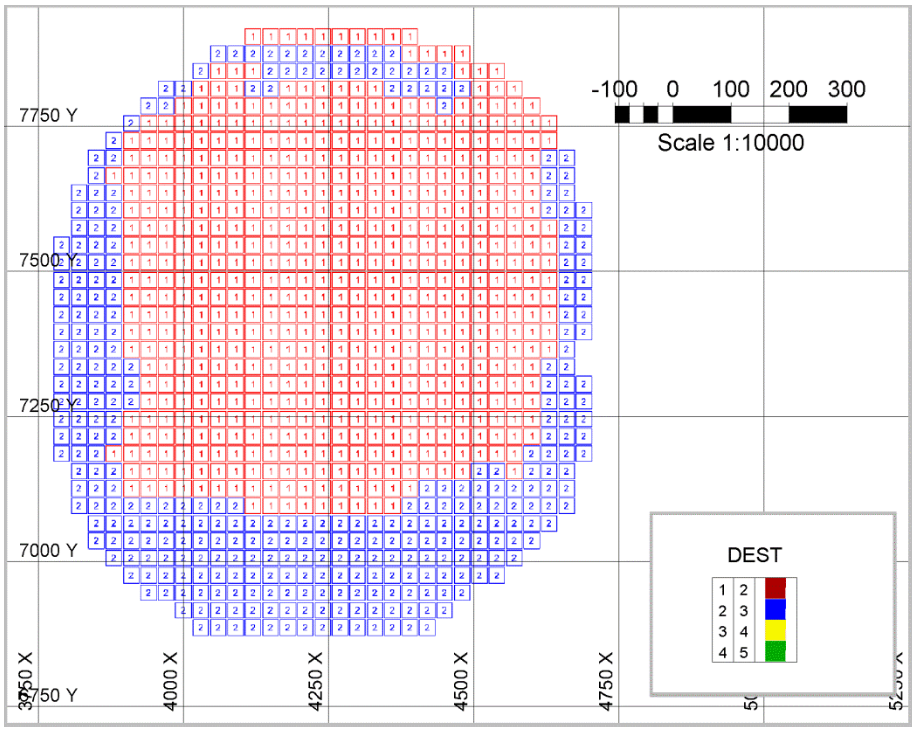


Figure 4.90. MILP Destination Plan View at 600m Elevation

4.7.6 *Mining Units: Clusters, Processing Units: Blocks*

In this section, we ran the MILP model with the Tabu search clusters generated in the previous section as the mining units and blocks as processing units. Despite the higher resolution compared to using bench-phases as mining units, the generated NPV is much lower. Running the MILP solver to 10% gap takes 2,238 seconds and results in $2,035M which is 9.2% less than the NPV generated from Milawa NPV schedule. The resulted schedule and plan views follow in Figure 4.91 to Figure 4.94.

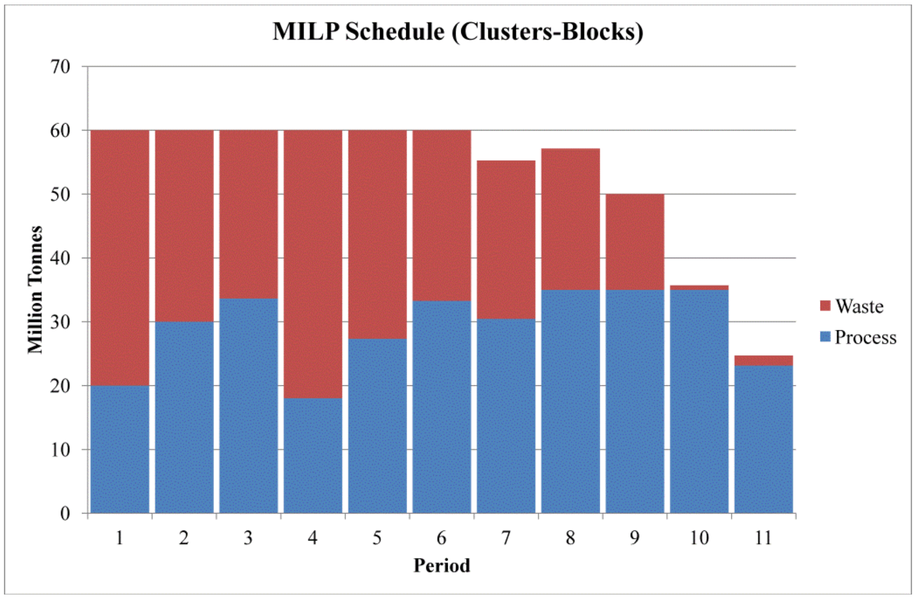


Figure 4.91. MILP Schedule

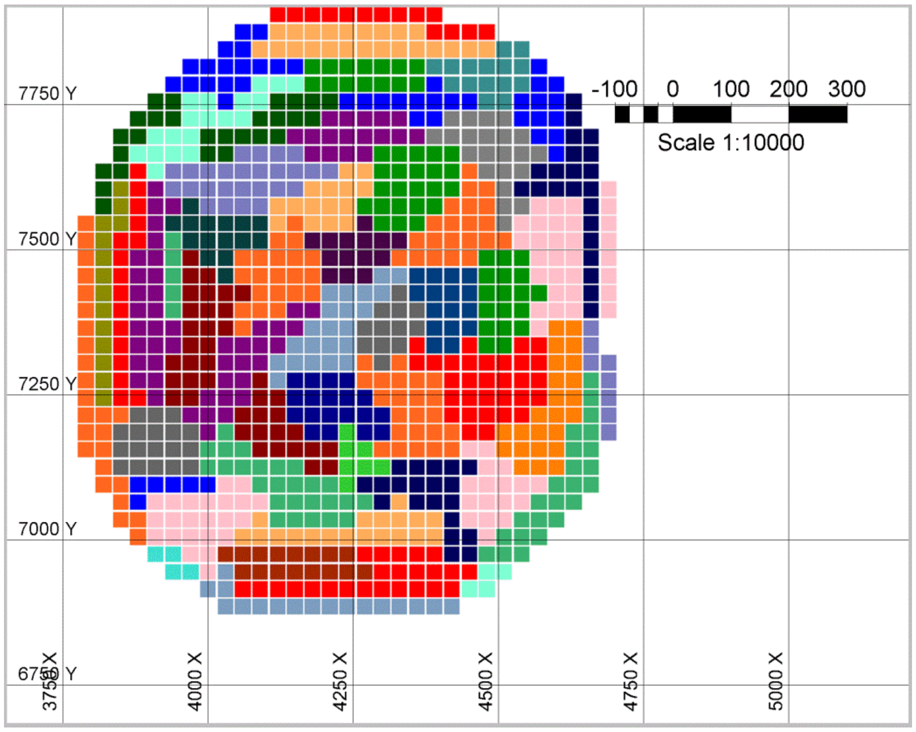


Figure 4.92. Cluster IDs Plan View at 600m Elevation

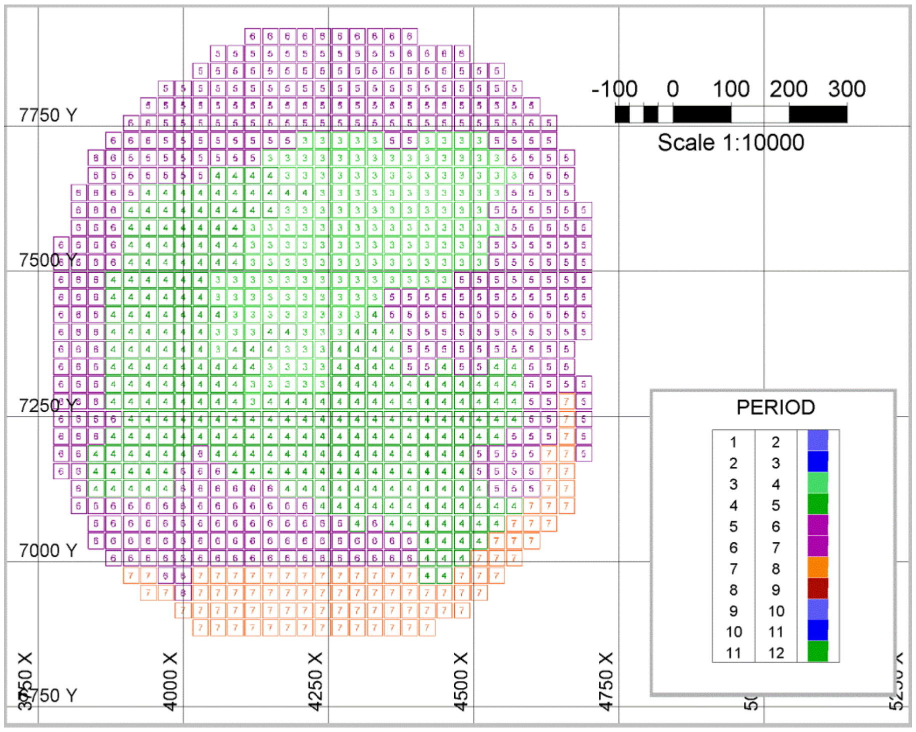


Figure 4.93. MILP Schedule Plan View at 600m Elevation

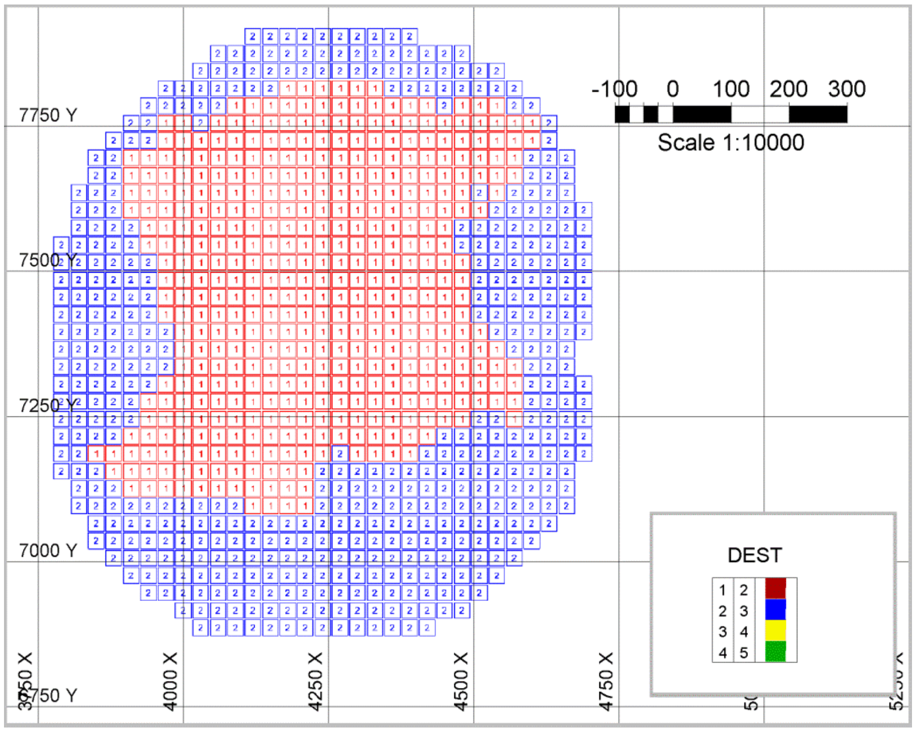


Figure 4.94. MILP Destination Plan View at 600m Elevation

4.7.7 *Mining Units: Blocks, Processing Units: Blocks*

We tried to directly solve the model with blocks as both mining and processing units for this section. However, after 30 days of running the solver no feasible solution was found and we had to kill the process. Therefore, we decided to use a three-stage approach based on period aggregation to solve the problem. First, we solved the problem with three periods of four years and use the solution to limit the decision variables. The mining and processing capacities for each four-year period are presented in Table 4.54. We used a discount rate of 40% per period. The problem was solved in 4.81 hours and the resulted schedule is plotted in Figure 4.95.

Table 4.54. Aggregated Four-Year Periods Capacities

Period	1	2	3
Equivalent Years	1-4	5-8	9-12
Mining Capacity (Mt/period)	240	240	180
Processing Capacity (Mt/period)	120	140	105

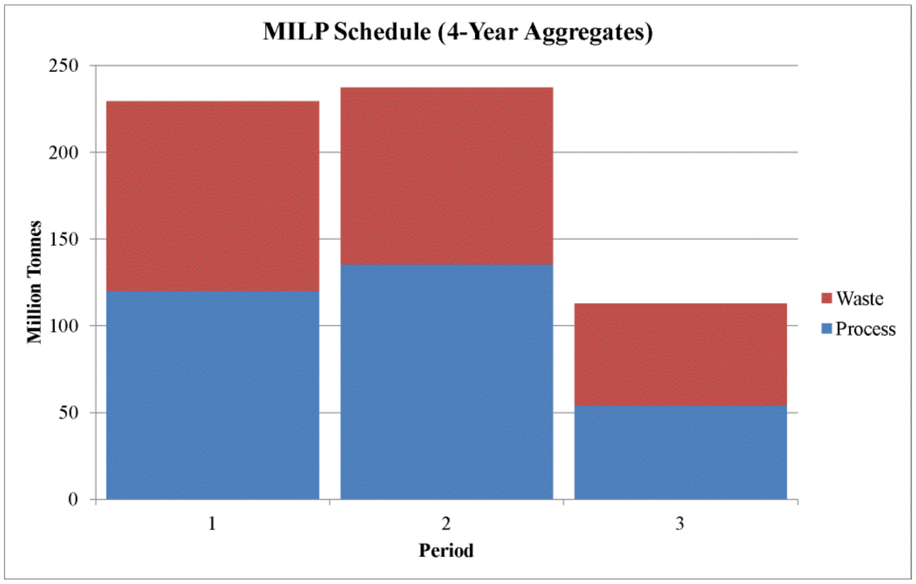


Figure 4.95. MILP Schedule (4-Year Aggregates)

We solved the aggregated problem to 5% gap in block resolution and used the solution to restrict extraction periods with a multiplier of two and solved the model for six periods of two years. For example, if a block is scheduled to be extracted in the first four-year period it now will be decided to be extracted in the first two-year period or the second two-year period. The mining and processing capacities for each two-year period are presented in Table 4.55. We used a discount rate of 20% per period. The problem was solved to 5% gap in 29.1 hours and the resulted schedule is plotted in Figure 4.96.

Table 4.55. Aggregated Two-Year Periods Capacities

Period	1	2	3	4	5	6
Equivalent Years	1-2	3-4	5-6	7-8	9-10	11-12
Mining Capacity (Mt/period)	120	120	120	120	120	60
Processing Capacity (Mt/period)	50	65	70	70	70	35

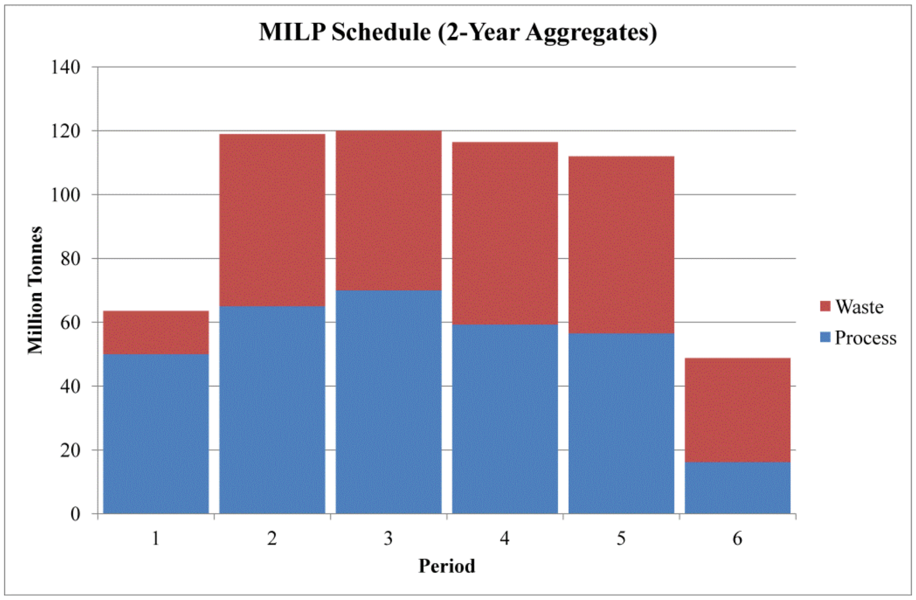


Figure 4.96. MILP Schedule (2-Year Aggregates)

Finally, we used the solution from aggregated two-year periods and restricted the original problem to the extraction periods from this model by adding a tolerance of one year. For example, if a block was scheduled to be extracted in the second two-year period, the corresponding extraction variables will be limited to years 2, 3, 4 and 5. The problem was solved to 5% gap in 34.4 hours. The resulted schedule and sample plan views are presented in Figure 4.97 to Figure 4.99. The resulted NPV is $2,566M.

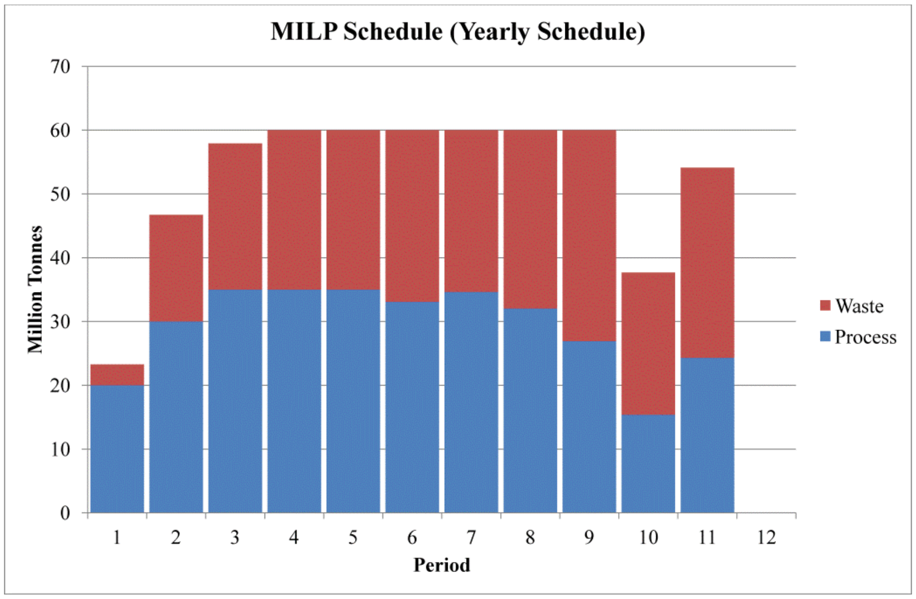


Figure 4.97. MILP Schedule (Yearly Schedule)

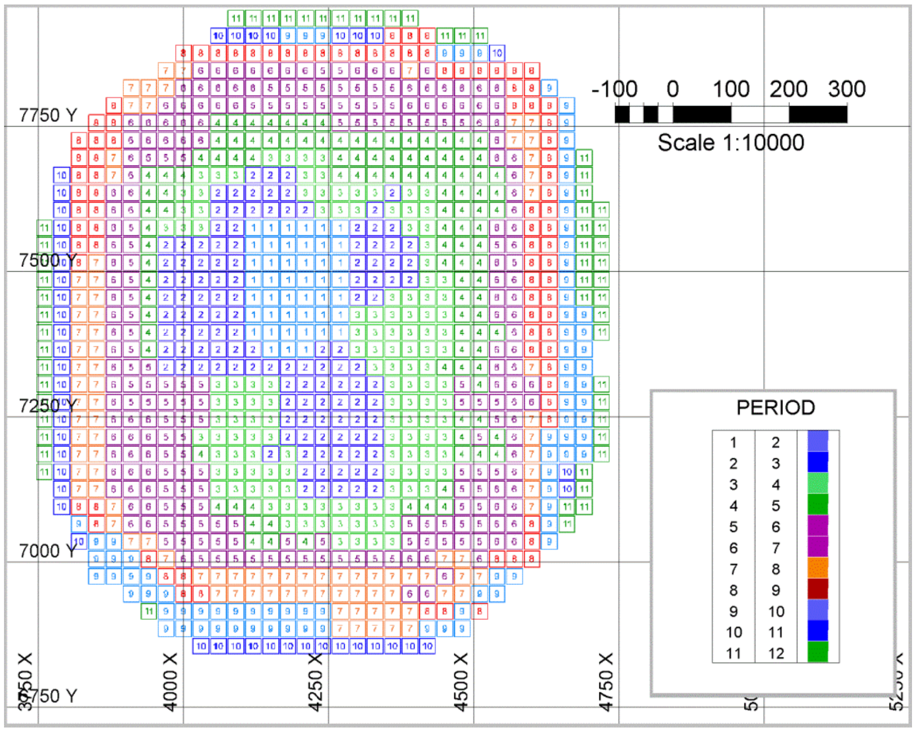


Figure 4.98. MILP Schedule Plan View at 600m Elevation

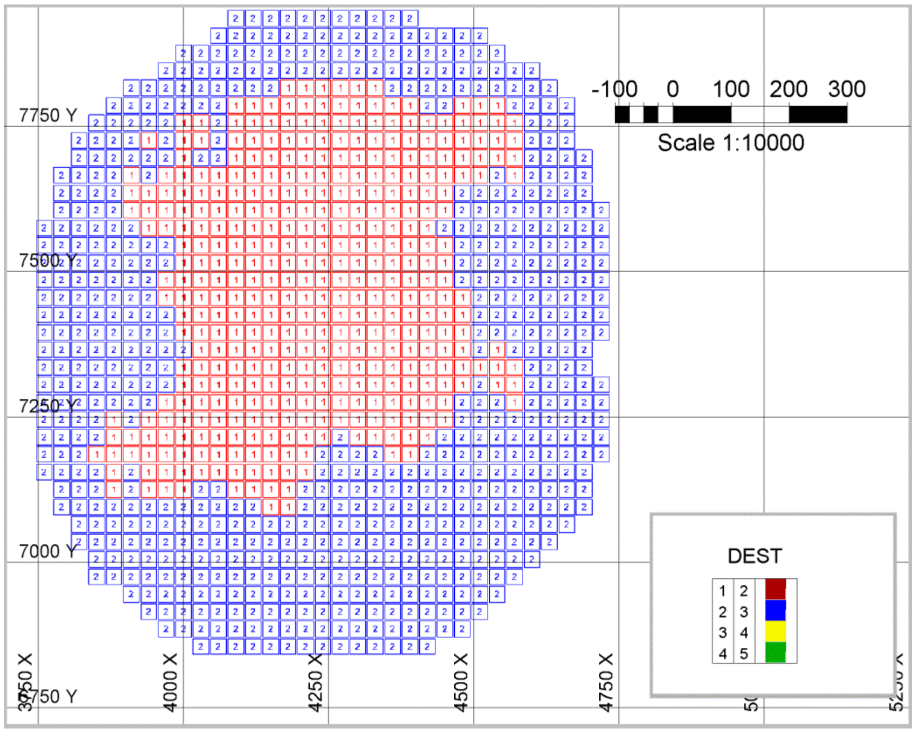


Figure 4.99. MILP Destination Plan View at 600m Elevation

4.7.8 *Mining Units: Bench-phases, Processing Units: Clusters, With Stockpiles*

In this section, we study how we can add stockpiles to the case-study. For this purpose, we add three stockpiles with unlimited capacity to Whittle™ scheduler and our own model. A rehandling cost of 0.4 $/tonne is applied for reclaiming material from stockpiles and sending to the mill. Moreover, it is assumed that the recovery of material reclaimed from stockpiles is 2% less than original recoveries. The fleet required to reclaim material from stockpiles and send to the mill is considered to be independent of the available mining capacity. Figure 4.103 shows the schedule generated with Whittle™ based on the listed assumptions. As can be seen in the figure, Whittle™ uses the stockpile to make sure that the plant has enough feed in every period. However, the NPV of the operation based on this schedule is $2,224M which is 0.8% less than the original Milawa NPV algorithm. Figure 4.104 is the schedule generated from the MILP formulation with the same assumptions. Since the MILP formulation requires a fixed reclamation grade for each stockpile, we averaged the grade values for the clusters with each rock-type and used as the reclamation grade. The reclamation grades presented in Table 4.56 are used to calculate the revenue generated from reclaiming material from each stockpile and sending to the plant in the MILP formulation.

Table 4.56. Stockpile Parameters

	Rock-type	Au Grade (gram/tonne)			Cu Grade (%m)		
		Min	Max	Avg	Min	Max	Avg
SP1	PM	0	0.89	0.45	0	0.42	0.20
SP2	MX	0	1.14	0.58	0	1.29	0.50
SP3	OX	0	1.13	0.42	0	1.29	0.58

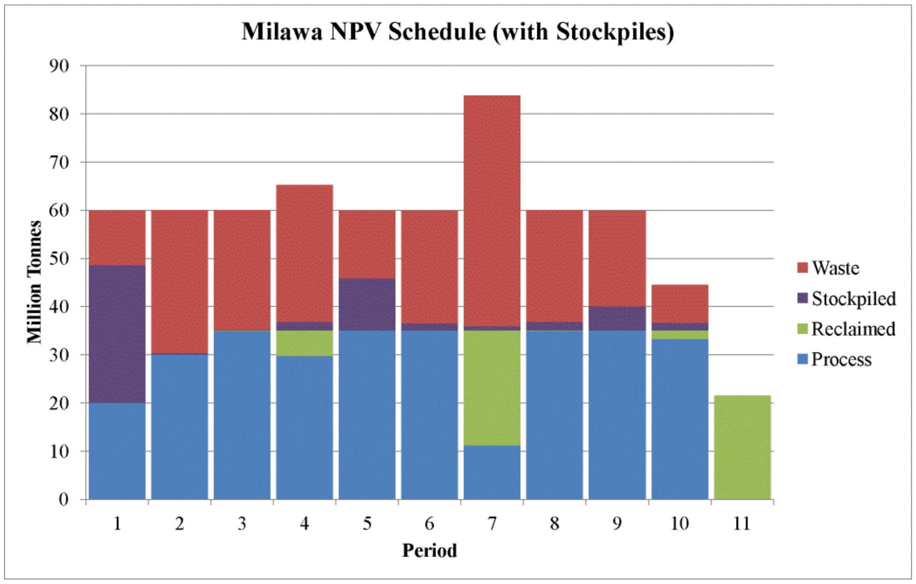


Figure 4.100. Milawa NPV Schedule (with Unrestricted Stockpiles)

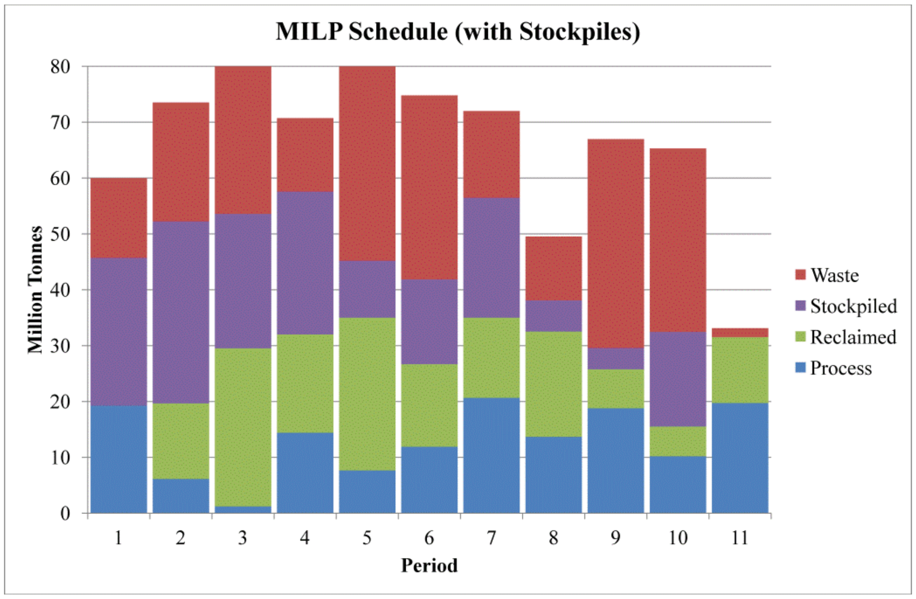


Figure 4.101. MILP Schedule (with Unrestricted Stockpiles)

The NPV resulted from the MILP schedule is $2,747M which is significantly higher than other scenarios. However, this NPV is resulted from approximating the reclamation grade of the stockpiles with the average grade of the material in the pit. Therefore, we calculated the actual grade of material in the stockpile based on the proposed schedule and the actual revenue generated from reclaiming material from stockpiles and sending to the plant. Figure 4.102 shows the approximated cashflow based on fixed reclamation grade minus the actual generated revenue for each period. As can be seen in Figure 4.102, the difference between the approximated revenue and the actual revenue is significant in most of the periods and has resulted in overestimation of the final NPV. In total, the generated NPV is $516M more than the actual NPV that can be generated with this schedule. Therefore, we restricted each stockpile to accept one rock-type with limited grade range to reduce the difference between the assumed average reclamation grade and the actual stockpile grade. The summary of stockpile definitions is provided in Table 4.57. We calculated the weighted average of grade values in each rock-type within the acceptable ranges and used as the reclamation grades for each stockpile. We used the same ranges for Milawa NPV and compared the outcomes of both schedulers.

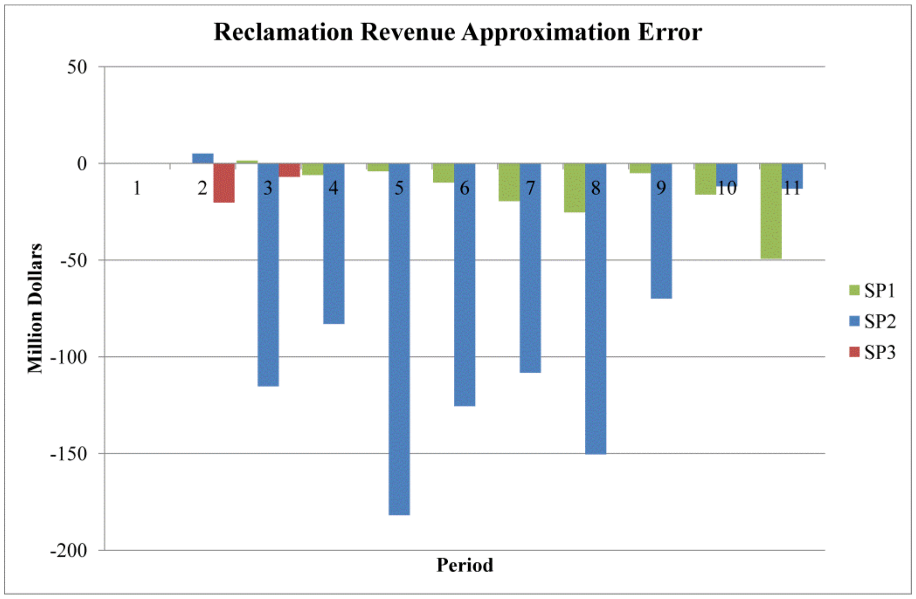


Figure 4.102. Reclamation Revenue Approximation Error

Table 4.57. Stockpile Parameters

	Rock-type	Au Grade (gram/tonne)			Cu Grade (%m)		
		Min	Max	Avg	Min	Max	Avg
SP1	PM	0.1	0.3	0.25	0.15	0.45	0.18
SP2	MX	0.2	0.5	0.31	0.1	0.3	0.23
SP3	OX	0.1	0.4	0.15	0.1	0.2	0.17

As mentioned earlier, Milawa NPV algorithm uses the stockpiles to feed the plant when enough ore cannot be extracted from the mine and extends the mine life to the 11th period in this case. The resulted NPV is \$2,155M which is 3.8% less than the original Milawa NPV schedule and 0.6% less than the original Milawa balanced schedule. However, the plant is fully utilized in all periods except than the last period. On the other hand, the MILP model uses stockpiles more frequently and increases the NPV of the operation to \$2,432M which is 11.3% more than the original panel-cluster scenario. The plant is also better utilized compared to not using the stockpiles. The generated schedules from Milawa NPV and MILP are presented in Figure 4.103 and Figure 4.104 respectively. Similar to the unrestricted stockpile case, we plotted the approximation error in cashflows in Figure 4.105. The total overestimation in calculating the NPV of the operation is $27M which is a 1.1% error. Moreover, it is possible to decrease the error by using tighter bounds on the stockpile grades or by calibrating the reclamation grades based on the resulted schedule.

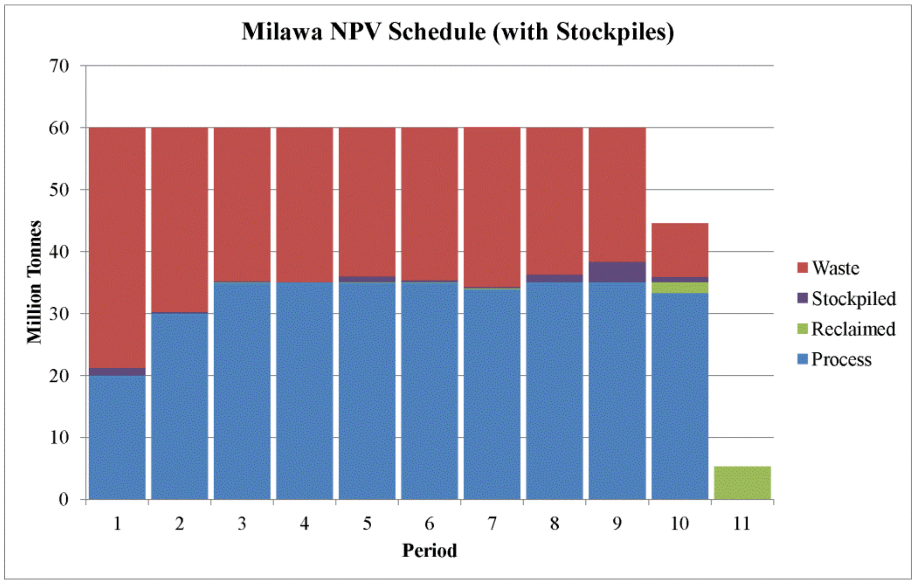


Figure 4.103. Milawa NPV Schedule (with Stockpiles)

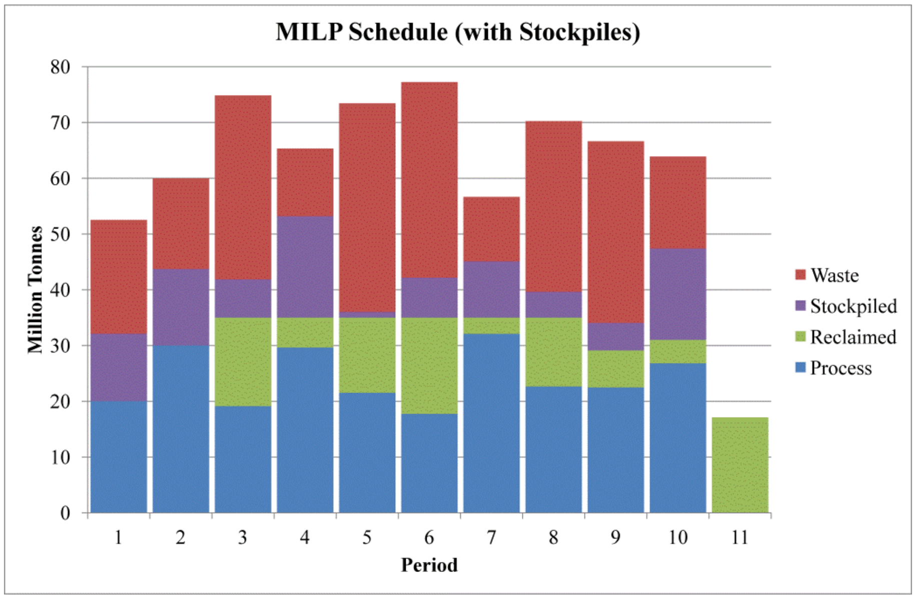


Figure 4.104. MILP Schedule (with Stockpiles)

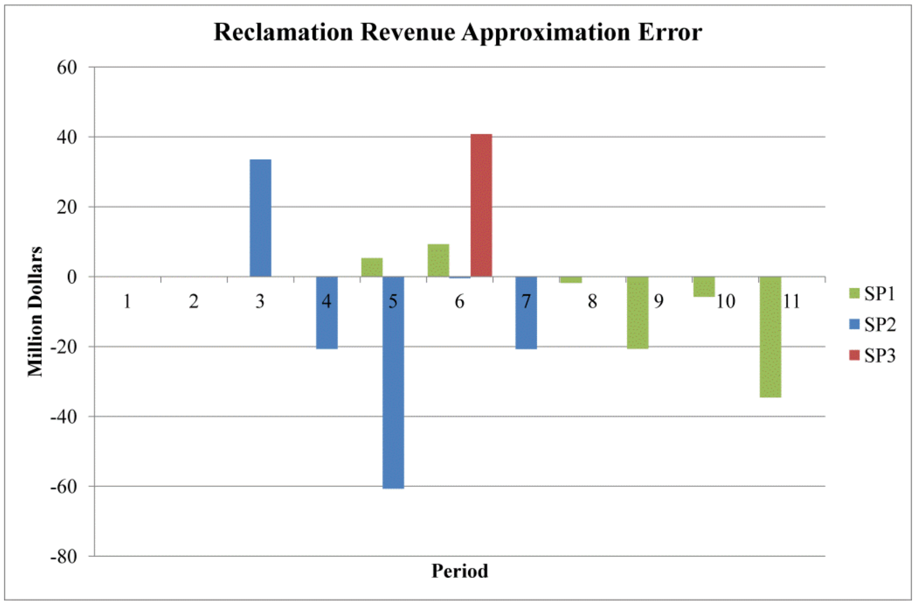


Figure 4.105. Reclamation Revenue Approximation Error

4.7.9 *Mining Units: Bench-phases, Processing Units: Clusters, Blend Control*

Blending is another important aspect of production planning. As mentioned in 3.4.4.6, our model is capable of controlling the head grade of material sent to processing destinations. This features works for both models with and without stockpiles. In this section, we add lower and upper bounds to the head grade of material sent to the mill. The gold and copper grade lower and upper bounds are presented in Table 4.58. The rest of the parameters are the same as before.

Table 4.58. Head Grade Control Parameters

Au Grade (gram/tonne)		Cu Grade (%m)	
Min	Max	Min	Max
0.3	0.7	0.3	0.7

First, we solve the model by adding grade control constraints and removing stockpiles. The model is solved to optimality in 1,290 seconds and results in an NPV of $2,085M. The production schedule is presented in Figure 4.106 and the head grade of material sent to the process is presented in Figure 4.107.

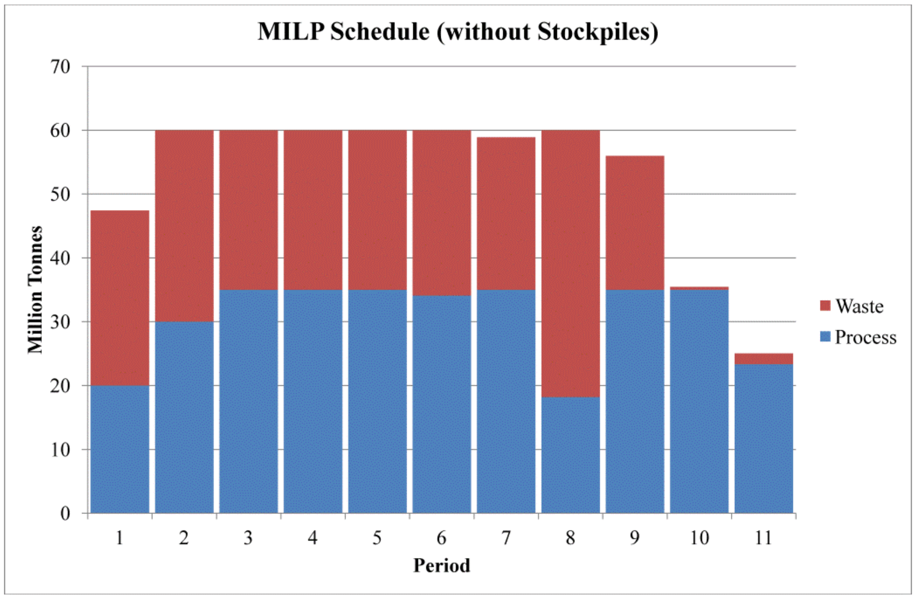


Figure 4.106. MILP Schedule (without Stockpiles)

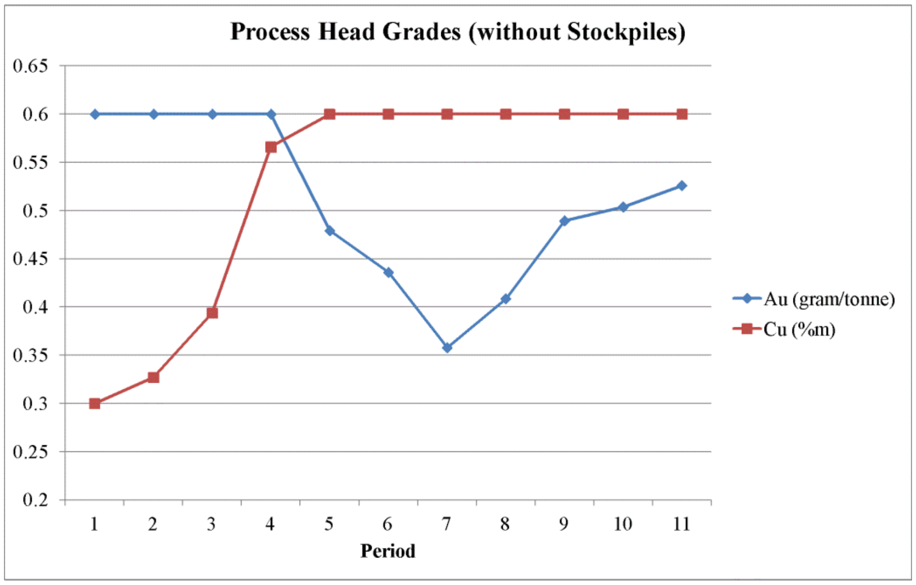


Figure 4.107. Process Head Grades (without Stockpiles)

Afterwards, we add the same stockpiles as in the previous section to increase the flexibility of the model and test its performance. Although the mathematical formulation uses fixed reclamation grades for stockpiles, we used the actual grade of stockpiles in calculating the head grades. We solved the model by adding the same stockpile settings as the previous section and applying the head grade constraints. Solving the model to optimality takes 613 seconds and results in an NPV of $2,394M which is 1.3% less than not having constraints on the head grades. We expect the NPV to drop more if tighter bounds on the head grade are applied. The production schedule and head grades are plotted in Figure 4.108 and Figure 4.109 respectively. As can be seen in Figure 4.109, the actual head grades violate the upper bounds in only one instance due to approximation of reclamation grade with a fixed number.

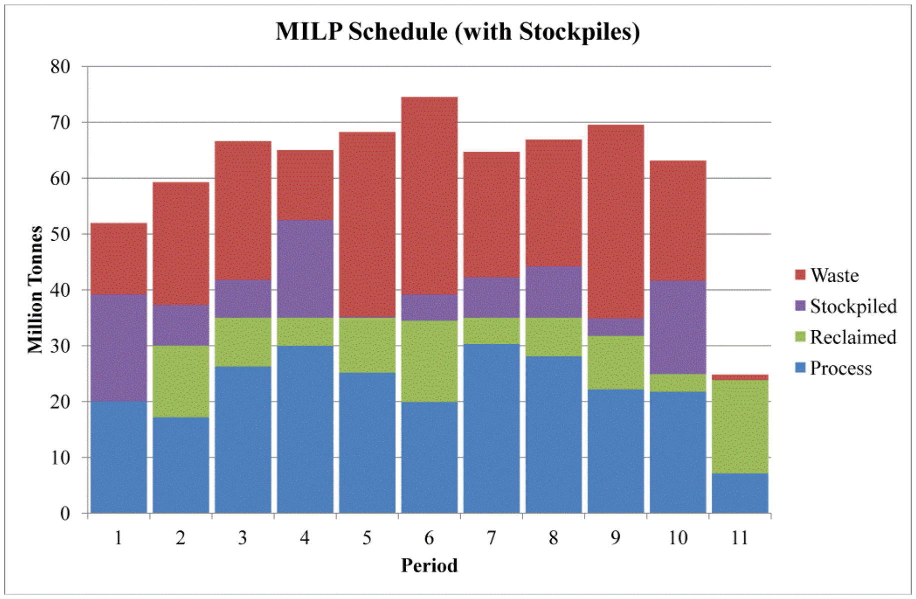


Figure 4.108. MILP Schedule (with Stockpiles)

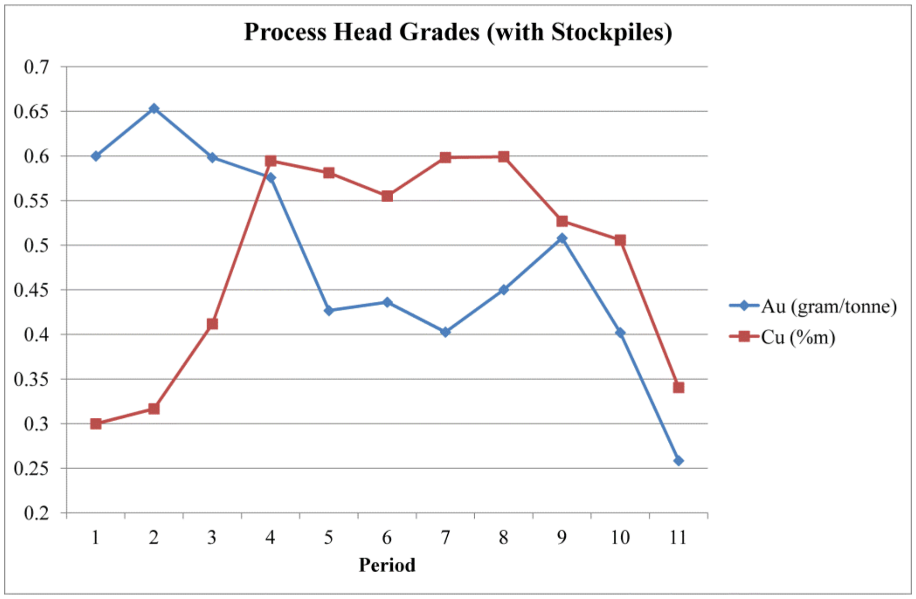


Figure 4.109. Process Head Grades (with Stockpiles)

4.7.10 *Summary and Conclusion*

In this section, we implemented the MILP formulation on different resolutions and scenarios on the Marvin dataset to study the performance of formulation and decide on the proper resolution of mining and processing units. We ran different scenarios on the LTOPP MILP formulation with different combinations of blocks, clusters and panels as mining and processing units. We compared the results with GEOVIA Whittle™ Milawa algorithms to be able to benchmark against a popular commercial mine scheduling software package. We started by using blocks as processing units and mining-panels as mining units to get a solution in the similar resolution as GEOVIA Whittle™. Whittle™ uses parcels (single rock type units within blocks) for making processing decisions and we keep track of different rock types within each block. Our MILP formulation resulted in around 19% improvement over Milawa NPV showing the benefits of using a mathematical formulation. However, both the Whittle™ and MILP solutions in this resolution suffer the same setback: they are not practical from the operations' point of view as the real mining operation does not have the selectivity to pick the blocks independent of their neighbors. A mining dilution factor is usually used to overcome this shortcoming and account for the dilution that happens by blasting and mining. Another way to tackle this issue is to create larger mining units that represent the selective mining unit of the operation and perform planning based on these units. For this purpose, we developed a clustering algorithm to form mining cuts based similar rock types and used the mining cuts as processing units. The model runs to optimality in 31 seconds and the resulted NPV is 2.5% less than the one from Milawa NPV. However, the MILP based on clusters has two advantages over the Milawa algorithm: the MILP has a measure of optimality and using the clusters provides a more practical and implementable schedule. Note that Marvin dataset is a synthetic uniform dataset and the differences will be more drastic in real deposits as can be seen in the following sections. Next, we solved the MILP with different combinations of clusters and blocks as mining and processing units. As can be seen in sections 4.7.4 to 4.7.7, using clusters and blocks not only significantly increases the processing time but also results in unpractical schedules. The number of drop cuts in the generated schedules makes the schedule impractical and since the cost of relocating the shovel is not included the NPV calculations, the resulted NPV is not really achievable. The resulted NPVs and processing times for the different scenarios are summarized in Table 4.59. The upper bound is calculated based on the optimality gap and represents the maximum possible NPV if solved to optimality. The difference percentage column is calculated based on Milawa NPV result.

Table 4.59. Marvin MILP Run Summary

Section	Mining Unit	Processing Unit	NPV ($M)	GAP (%)	CPU Time (s)	Upper Bound	Diff%
4.7.1 M-NPV	Panels	Parcels	2,240	-	7	-	-
4.7.1 M-Balanced	Panels	Parcels	2,166	-	7	-	-3.3%
4.7.2	Panels	Blocks	2,660	0	322	2,660	18.8%
4.7.3	Panels	Clusters	2,185	0	31	2,185	-2.5%
4.7.4	Clusters	Clusters	2,006	10	1,435	2,207	-1.5%
4.7.5 Before TS	Clusters	Clusters	2,051	10	804	2,256	0.7%
4.7.5 After TS	Clusters	Clusters	2,042	10	239	2,246	0.3%
4.7.6	Clusters	Blocks	2,035	10	2,238	2,239	-0.1%
4.7.7	Blocks	Blocks	2,566	5	250K	2,694	20.3%

Another set of scenarios are designed to test the stockpiling model with linearization. Since we concluded that panels as mining units and clusters as processing units are the best options for scheduling, we ran three different stockpiling scenarios on this resolution. The first scenario was presented to show the error that linearization can introduce into the NPV calculations. Afterwards, we restricted the stockpiles to certain grades and the error dropped significantly. We compared the outcome to Whittle™ scheduler to show how MILP can outperform Whittle™ when the model gets more complicated by introducing stockpiles. The last stockpiling scenario included the grade blending in the mill. The processing times and generated NPVs are summarized in Table 4.60. The case studies for the stockpile verify that the error caused by assuming a fixed reclamation grade can be controlled by restricting the stockpiles to certain lower and upper bounds on the input grade.

Table 4.60. MILP with Stockpiling Summary

Section	Description	NPV ($M)	CPU Time (s)	Error ($M)	Diff%
4.7.8	Milawa	2,224	9	-	-
4.7.8	MILP	2,747	2	516	23.5%
4.7.9	Milawa	2,155	15	-	-
4.7.9	MILP	2,432	232	27	12.9%
4.7.9	MILP with Blending	2,085	1,290	-	-3.2%

4.8. Gold Deposit

The second case-study is a gold, silver and copper deposit with 292,000 blocks in the final pit. The total tonnage of material in the final pit is 1,068 Mt. Two processing methods are used in the operation: mill and leach pad. The rock type tonnages and their destinations are summarized in Table 4.61. The dataset is from an ongoing mining operation and the presented tonnages are the material to be mined from the final pit limit. A 3D representation of the deposit and its rock types is presented in Figure 4.110. The deposit is divided into 21 pushbacks. A sample plan view and a sample cross section with the pushback IDs are presented in Figure 4.111 and Figure 4.112 respectively.

Table 4.61. Gold Deposit Summary

Rock Code	Rock Type	Tonnage (Million)	Possible Destination
1	CUOX (copper leach oxide)	13.741	Leach
2	PL (partial leach)	252.310	Leach
3	EN (enriched)	37.696	Leach & Mill
4	HYPO (hypogene)	555.710	Mill
5	TR (transitional)	21.504	Waste Dump
6	OX (oxide)	49.895	Waste Dump
7	UND (undefined)	87.530	Waste Dump

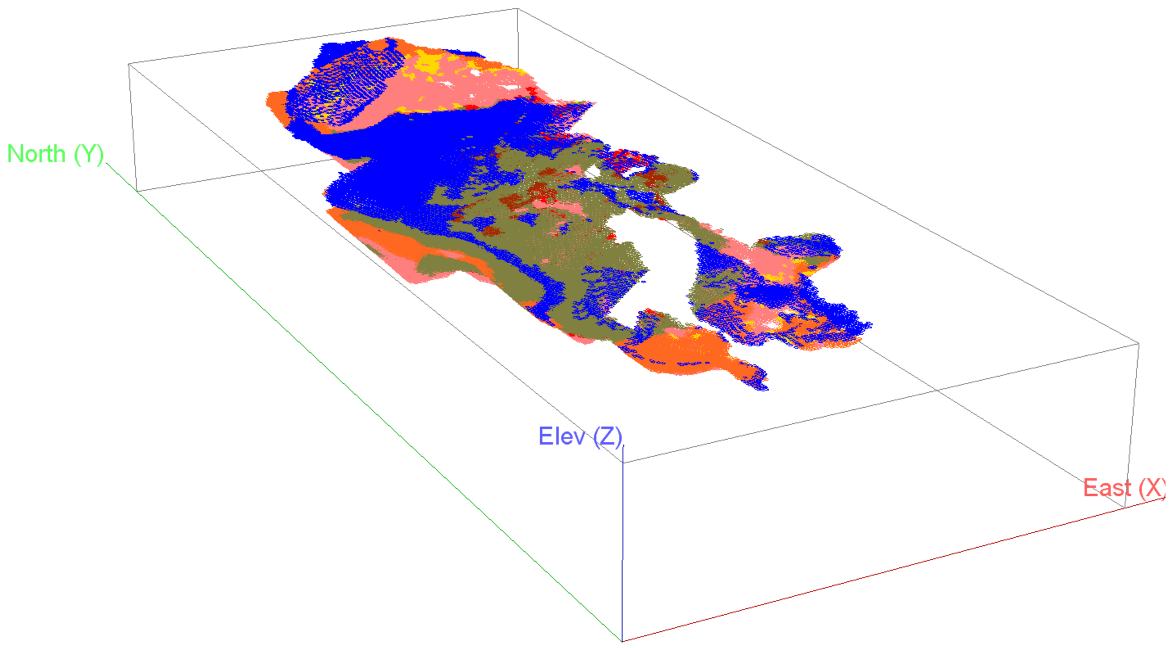


Figure 4.110. 3D view of the deposit

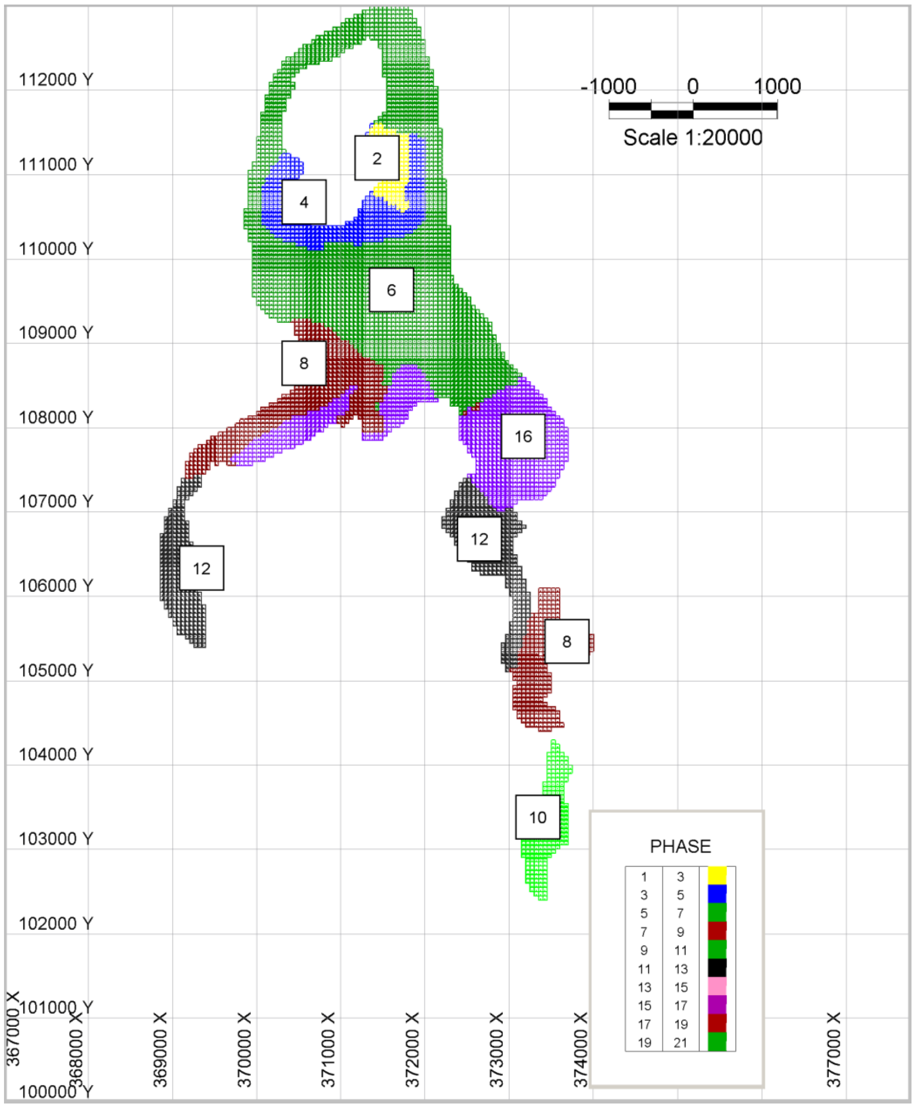


Figure 4.111. Sample Plan View of pushbacks

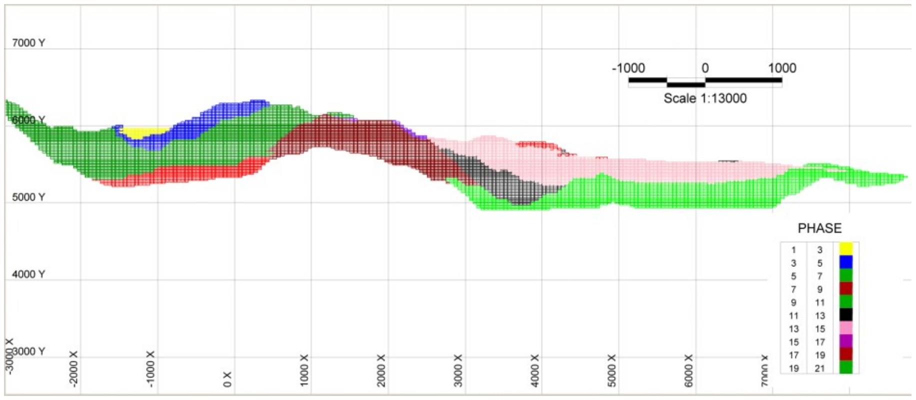


Figure 4.112. Sample Cross Section of pushbacks

4.8.1 *Clustering*

We ran our hierarchical clustering algorithm on this deposit with different parameters to evaluate the performance of the algorithm on a large deposit. The main element of interest is gold and we used gold grade in clustering and calculating the grade variations.

Table 4.62. Clustering Scenarios

Scenario Code	Distance Weight	Grade Weight	Rock type Penalty	Destination Penalty	Cluster Size	
					Avg	Max
G15	0.8	1	1	1	15	22
G30	0.8	1	1	1	30	40
G100	0.8	1	1	1	100	120
R15	0.8	0	0.2	1	15	22
R30	0.8	0	0.2	1	30	40
R100	0.8	0	0.2	1	100	120
D15	0.8	0	1	0.2	15	22
D30	0.8	0	1	0.2	30	40
D100	0.8	0	1	0.2	100	120
B15	0.6	0.2	0.2	0.2	15	22
B30	0.6	0.2	0.2	0.2	30	40
B100	0.6	0.2	0.2	0.2	100	120

Table 4.63. Clustering Summary

Scenario Code	Number of Cuts	Rock Unity (%)		DDF (%)		Au Grade CV (%)		CPU Time (s)
		Avg	Std	Avg	Std	Avg	Std	
G15	6,167	89.4	16.4	90.0	15.0	37.7	62.6	608
G30	3,336	87.0	17.4	87.7	15.7	45.2	79.7	600
G100	1,158	82.3	19.4	81.1	16.9	58.9	26.3	612
R15	6,144	97.1	8.7	91.6	13.9	50.8	29.1	608
R30	3,316	95.1	11.0	89.7	14.6	56.5	28.9	610
R100	1,187	91.1	14.3	87.1	15.1	63.2	28.1	601
D15	6,155	93.2	13.3	96.5	8.9	45.9	30.3	615
D30	3,342	90.9	14.9	94.7	10.4	51.3	28.4	612
D100	1,208	87.7	16.7	91.6	12.3	58.6	28.6	602
B15	6,167	96.6	9.6	96.0	9.9	39.7	29.3	625
B30	3,358	94.5	11.8	94.2	11.1	47.3	28.9	541
B100	1,225	90.7	14.9	90.6	13.3	57.7	29.0	531

The clustering scenarios are presented in Table 4.62. The clustering performance measures as well as processing times are presented in Table 4.63.

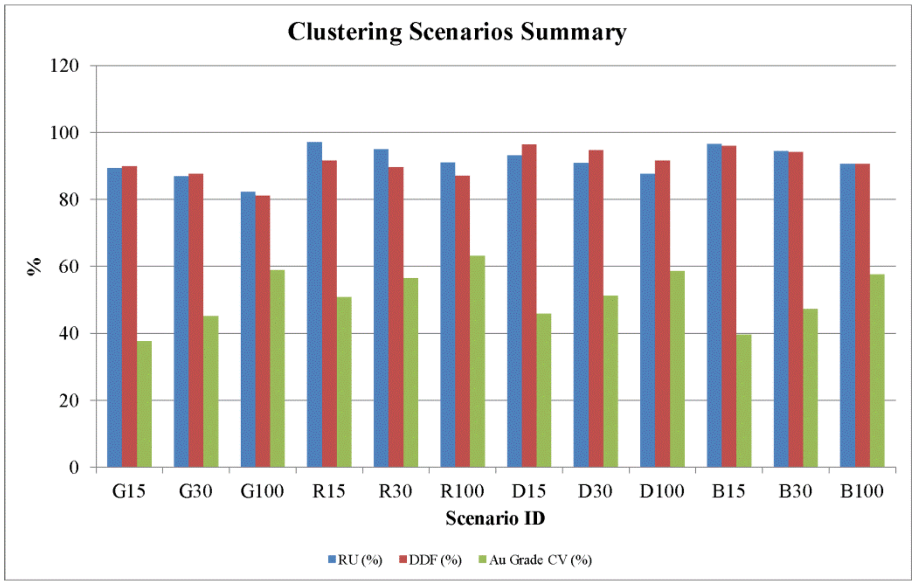


Figure 4.113. Clustering Scenarios Summary

As clearly shown in Figure 4.113, the clustering performance measures deteriorate as the size of the clusters increase in every group of scenarios. On the other hand, it is shown that clustering based on grade results in lowest variation in grade. Similarly, clustering based on destination and rock type results in highest homogeneity in block destination and rock type. However, clustering based on grade fails to account for destination and rock type homogeneity and vise versa. Thus, the balanced scenarios (B15, B30 and B100) are tested to tune the parameters such that all the performance measures are relatively improved. The weights and penalties in these scenarios are acquired via try and error. Average grade variation in scenario B15 is 5% more than G15 while its RU and DDF are 8% and 6% higher than G15 respectively. Similarly, average DDF in B15 is only 0.5% less than D15 while its Au CV is 14% less and its RU is 4% higher than D15. Likewise, average RU in

B15 is 0.6% less than R15 while its AU CV is 22% less and its DDF is 5% higher than R15.

4.8.2 *MILP*

We ran multiple scenarios of production planning on the gold deposit with different mining and processing capacities. In the first group of experiments, our goal was to illustrate how we can improve the production schedule by modifying constraints and rerunning the MILP to different optimality gaps. In the second group, we aimed at showing the differences between maximizing NPV and maximizing reserves by changing the mine life.

As mentioned earlier, there 292,000 blocks in the final pit which are divided into 21 pushbacks resulting in 725 panels. We clustered blocks based on distance, grade and rock type to form the processing units. The total tonnage of material in the final pit is 1,068 Mt. In the first group of scenarios we form the MILP based on 25 years mine life with an initial mining capacity of 52.5 Mt per year. The mill capacity is 12.5 Mt/yr and leach capacity is 9 Mt/yr.

1. Case 1: this is our initial scenario based on clusters with 100 blocks per cluster on average. We ran the optimizer to the first feasible solution and obtained the results as shown in Table 4.64. As Figure 4.114 shows, the generated schedule is fully utilizing the mill but is not practical as it is not following a proper trend in utilizing mining equipment.

Table 4.64. Case 1 Summary

Case	Total Ore (Mt)	Total Mill (Mt)	Total Leach (Mt)	Total Waste (Mt)	NPV ($Million)
1	427.05	310.89	116.16	640.94	$843.60
Cuts - Panels	Gap %	CPU Time (S)	No of nonzero elements in A	No of Integer Variables	No of B&B nodes visited
3119-725	12.97	4,821	2,014,977	18,650	1,810

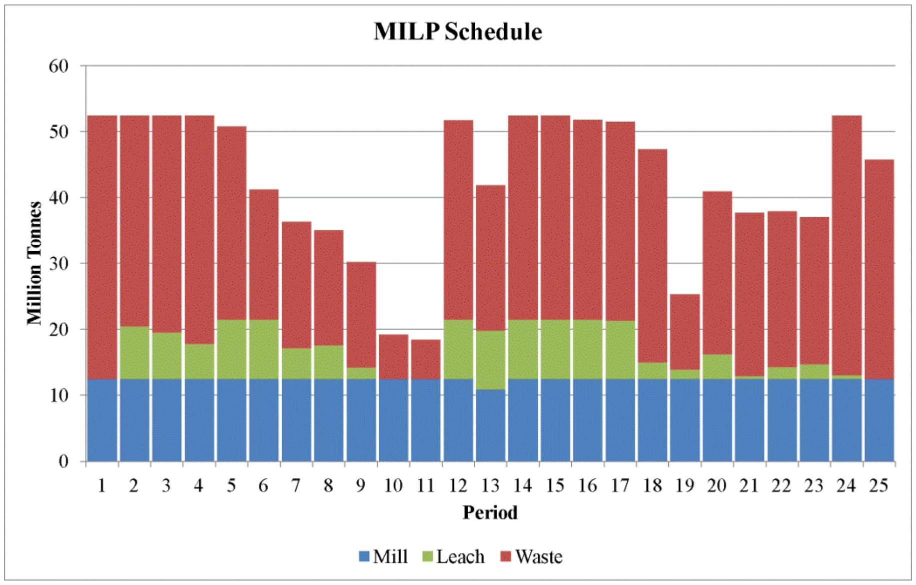


Figure 4.114. MILP Schedule

2. Case 2: for the second case, we clustered blocks with 30 blocks per cluster on average and changed the mine life to 24 years. We added a minimum mining constraint of 40 Mt/yr to periods 1-19 to obtain a more practical schedule. Afterwards, we ran the model to 5% gap and obtained the results as shown in Table 4.65 and Figure 4.115.

Table 4.65. Case 2 Summary

Case	Total Ore (Mt)	Total Mill (Mt)	Total Leach (Mt)	Total Waste (Mt)	NPV ($Million)
2	433.82	294.78	139.04	634.18	$1,095.40
Cuts - Panels	Gap %	CPU Time (S)	No of nonzero elements in A	No of Integer Variables	No of B&B nodes visited
9775-725	5.09	144,000	3,907,600	17,904	8906

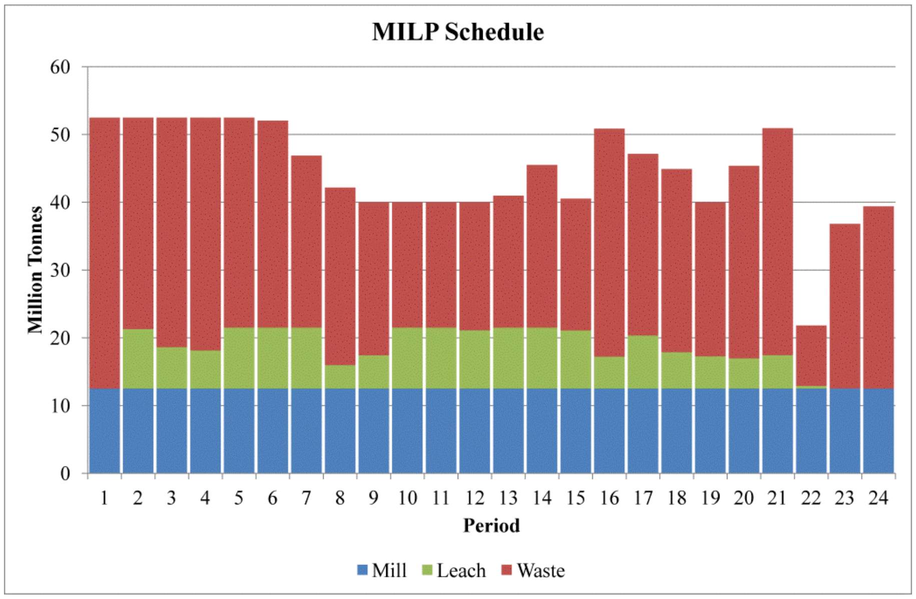


Figure 4.115. MILP Schedule

3. Case 3: after running case 2 to 5% gap, we used the solution we obtained and limited the extraction period of the panels to this solution ±2 years. We modified the matrices by removing the variables corresponding to the periods outside the allowable range (as explained in section 3.5.5) and solved the model to 1% gap. The summary of the outcome and the schedule are presented in Table 4.66 and Figure 4.116.

Table 4.66. Case 3 Summary

Case	Total Ore (Mt)	Total Mill (Mt)	Total Leach (Mt)	Total Waste (Mt)	NPV ($Million)
3	429.96	291.50	138.46	638.03	$1,099.71
Cuts - Panels	Gap %	CPU Time (S)	No of nonzero elements in A	No of Integer Variables	No of B&B nodes visited
9775-725	0.98	3146	3,907,600	17,904	520

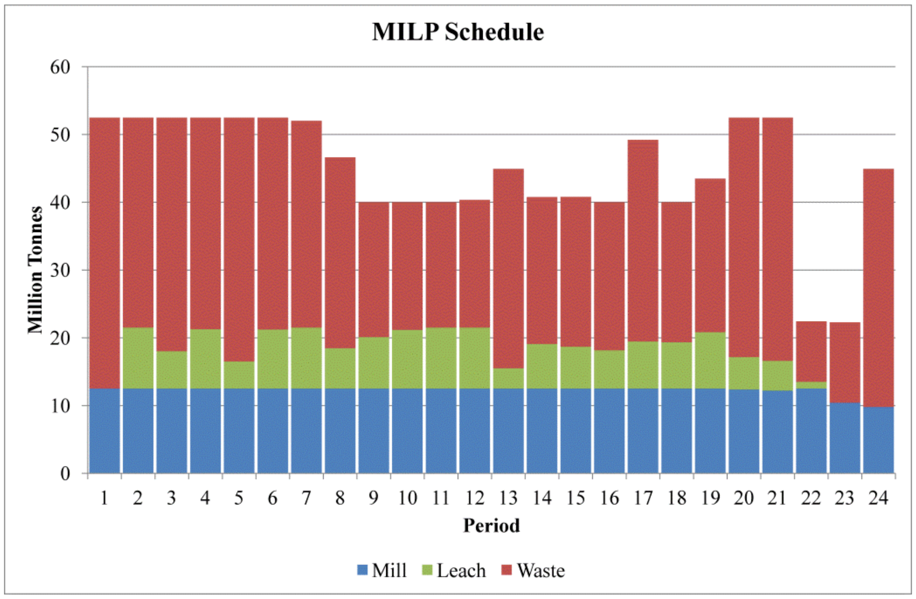


Figure 4.116. MILP Schedule

4. Case 4: in this case, we increase the mine life to 30 years. In the following cases we tried to create a practical schedule for 30 years mine life and compare the outcome against the 24 year schedule. We decreased the mining capacity to 50 Mt/yr for the first five years and 35 Mt/yr for next years. We did not enforce any minimum constraint on the total mining tonnage. We ran the model to the first feasible solution and obtained the results. The summary of the scheduling and the schedule plot are presented in Table 4.67 and Figure 4.117.

Table 4.67. Case 4 Summary

Case	Total Ore (Mt)	Total Mill (Mt)	Total Leach (Mt)	Total Waste (Mt)	NPV ($Million)
4	516.67	374.79	141.88	551.33	$1,029.38
Cuts - Panels	Gap %	CPU Time (S)	No of nonzero elements in A	No of Integer Variables	No of B&B nodes visited
9775-725	12.77	49,998	5,082,502	22,380	2331

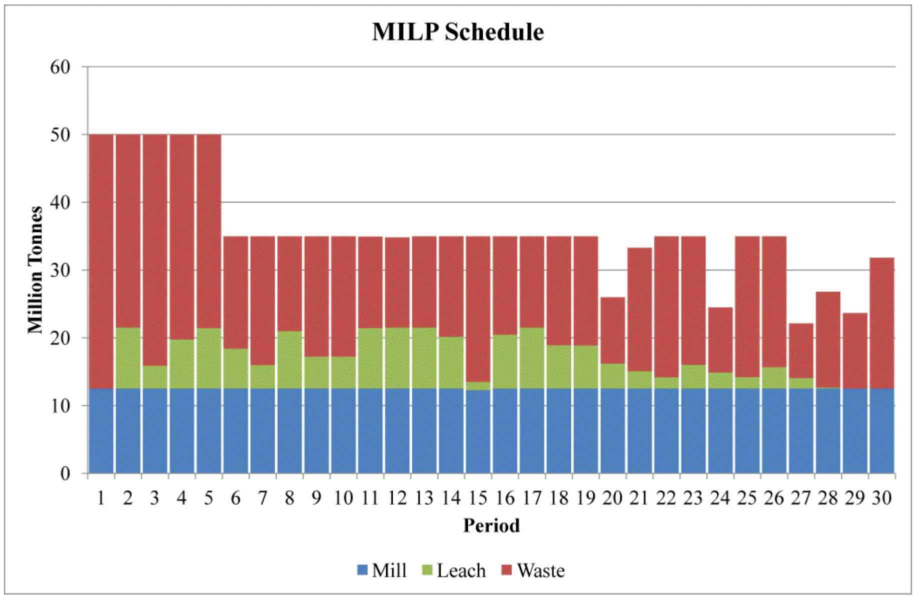


Figure 4.117. MILP Schedule

5. Case 5: in this case, we ran the same model as the previous case to 2% gap to obtain a solution closer to optimality. The summary of the generated schedule and the corresponding schedule plot can be found in Table 4.68 and Figure 4.118.

Table 4.68. Case 5 Summary

Case	Total Ore (Mt)	Total Mill (Mt)	Total Leach (Mt)	Total Waste (Mt)	NPV ($Million)
5	498.78	361.51	137.27	569.22	$1,067.67
Cuts - Panels	Gap %	CPU Time (S)	No of nonzero elements in A	No of Integer Variables	No of B&B nodes visited
9775-725	1.39	43,997	5,082,502	22,380	8393

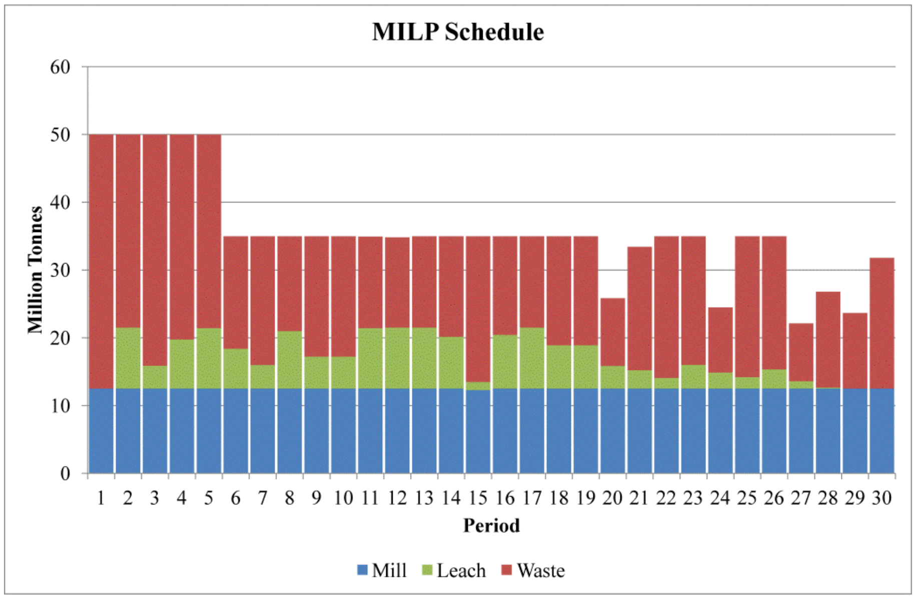


Figure 4.118. MILP Schedule

We ran multiple scheduling scenarios on the same dataset to understand the process of obtaining a reasonable schedule by changing constraints and parameters. In the first case, we solved for an initial solution to understand the deposit and come up with ideas to improve the schedule. Afterwards, we changed the mine life by only one year and enforced minimum mining constraint. With these changes and running the solver to 5% optimality gap, we obtained 29% improvement in the NPV and a more practical schedule. We used the solution obtained from case 2 to limit our extraction period decision making and solve the model to a tighter gap in a short time. The result showed 30% improvement on the NPV compared to case 1. Comparing case 3 to case 1 we can see that the optimizer has decided to decrease the total mill production from 311 Mt to 291 Mt and increase total leach production from 116 Mt to 138 Mt: a decision that is not obvious in the first place as production in mill usually results in higher revenues than the leach pad. Table 4.69 summarized the changes in mill and leach production as well as the NPV for different cases. The difference percentages are calculated based on difference from case 1.

Another important observation is the difference between case 3 and case 5 as the best schedules obtained for 24 and 30 years respectively. The 24 year schedule results in higher NPV while the 30 year schedule results in more reserves. The decision on which schedule to pursue relies on various technical, financial and management conditions and has to be made with care. Table 4.70 shows that increasing the mine life to 30 years results in 24% more reserves and 3% less NPV. Table 4.71 summarizes the total elements recovered in each case.

Table 4.69. Mill, Leach and NPV Comparison

Case #	Total Mill (Mt)	Diff%	Total Leach (Mt)	Diff%	NPV ($Million)	Diff%
1	311	0%	116	0%	844	0%
2	295	-5%	139	20%	1,095	30%
3	292	-6%	138	19%	1,100	30%
4	375	21%	142	22%	1,029	22%
5	362	16%	137	18%	1,068	27%

Table 4.70. Reserve versus NPV Comparison

Mine Life	Total Ore (Mt)	Total Mill (Mt)	Total Leach (Mt)	Total Waste (Mt)	NPV ($Million)
24	429.96	291.50	138.46	638.03	$1,099.71
30	498.78	361.51	137.27	569.22	$1,067.67
Diff (%)	16%	24%	-1%	-11%	-3%

Table 4.71. Total Elements Recovered

	24 Years		30 years		Diff		Diff (%)	
Element	MILL	Leach	MILL	Leach	MILL	Leach	MILL	Leach
Au (M oz)	4.168	-	4.73	-	0.56	-	13.45	-
Ag (M oz)	24.138	-	28.49	-	4.36	-	18.05	-
Cu (K ton)	291.32	180.11	341.31	177.10	49.99	-3.01	17.16	-1.67

4.9. Small Iron-Ore Deposit

The next case-study is an iron ore deposit which is processed in a magnetic separation plant. We implemented our clustering algorithm and MILP formulation on two different subsets of the deposit. The first subset is a small pit generated by lowering revenues with a revenue factor and the second subset is the optimal pit.

4.9.1 *Clustering and MILP*[1]

The algorithm is coded in Matlab® and implemented on an iron ore mine dataset consisting of 19,492 blocks in 19 benches. After creating the mining-cuts the MILP mine-life production schedule model is decided for 20 years. Various values have been tested for weight factors and they are calibrated accordingly. The calibration of weights are carried out in a way that the created mining-cuts follow the grade distribution and rock type patterns, while they have a shape that are reasonable for later extraction steps. The major emphasis in TS is to reduce the number of arcs generated. It is seen that the only option which significantly contributes to this factor is to consider a very small weight for intra-cluster similarity. The set of final weight parameters decided are summarised in Table 4.72.

Table 4.72. Algorithm parameters

Distance Factor Weight	Grade Factor Weight	Rock Type Penalty	Beneath Cluster Penalty	Intra-Cluster Similarity Weight	Number of Arcs Weight
W_D	W_G	r	c	W_S	W_N
2	0.5	0.1	0.8	0	1

The CPU time required for aggregating blocks and the number of arcs generated, which determines the number of binary constraints in the final model, are compared in Table 4.73 for different cluster sizes. The Tabu Search is run for a maximum of 100 iterations. Since this is an agglomerative hierarchical approach, the processing time for creating clusters gets longer as larger clusters are to be created. The Tabu Search CPU time, on the other hand, does not change regularly based on the cluster size.

[1] This section is published in: Tabesh, M. and Askari-Nasab, H., (2011), "Two-stage clustering algorithm for block aggregation in open pit mines", Transactions of the Institutions of Mining and Metallurgy, Section A: Mining Technology, © Maney Publishing, Leeds, LS3 1AB, United Kingdom, Vol. 120, No. 3, pp. 158-169. DOI: 10.1179/1743286311y.0000000009.

Table 4.73. Clustering algorithm run summary

Maximum Cluster Size	Number of Clusters Created	Hierarchical CPU Time (s)	TABU Search CPU Time (s)	Number of Arcs before TS	Number of Arcs After TS
25	1052	91.8	177.8	5133	4931
35	767	95.5	175.9	3625	3437
55	502	245.9	159.8	2215	2120
75	354	255.3	165.1	1464	1416
95	289	263.1	170.1	1186	1142

The resulted clustering schemes for the maximum size of 35 before and after TS are presented in Figure 4.119 and Figure 4.120. These can be compared against the rock type and grade distribution in the same level presented in Figure 4.121 and Figure 4.122.

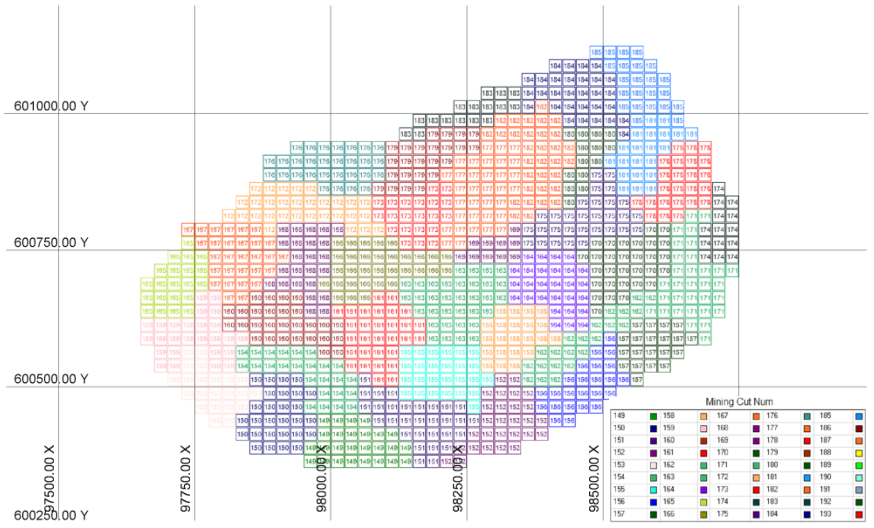


Figure 4.119. Clustering Scheme before TS (Plan View 1580)

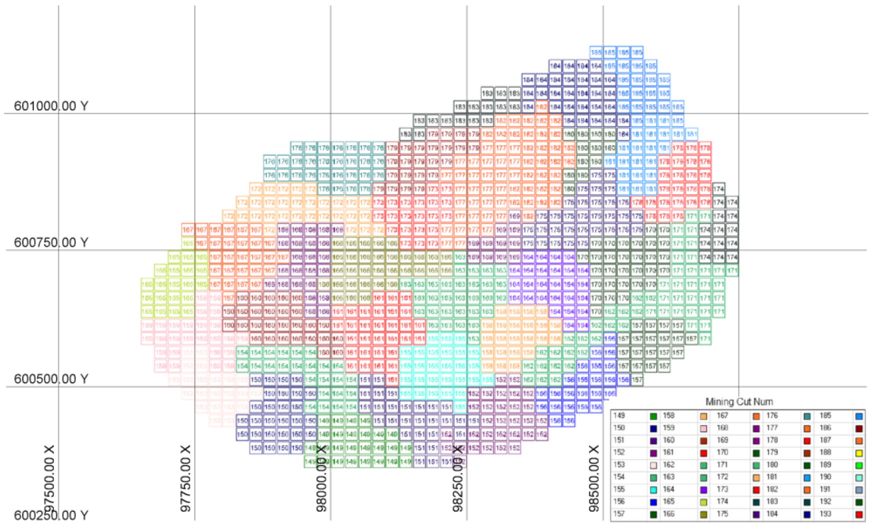


Figure 4.120. Clustering Scheme after TS (Plan View 1580)

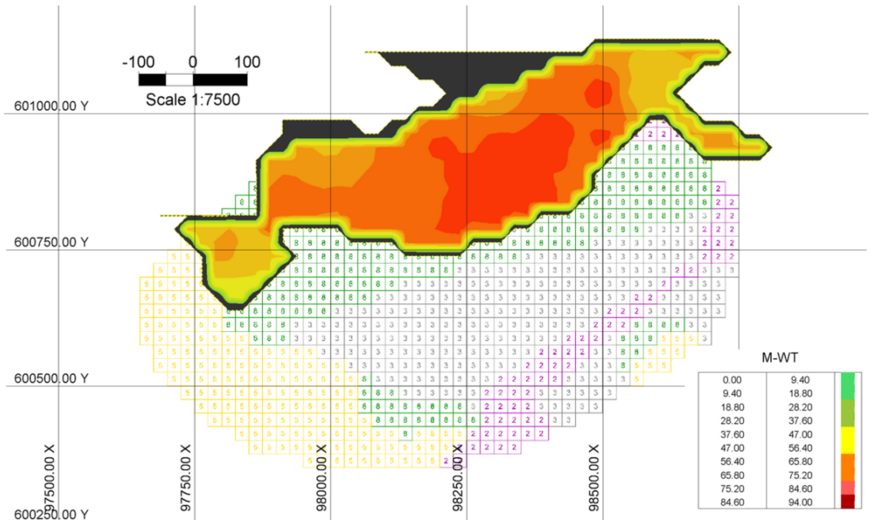


Figure 4.121. Ore-body and Grade Distribution (Plan View 1580)

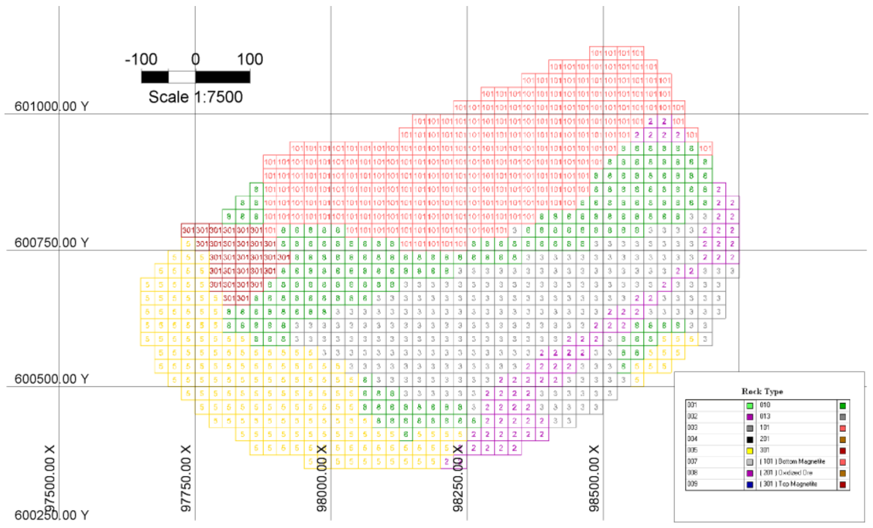


Figure 4.122. Rock-type Distribution (Plan View 1580)

The final pit contains 427.33 million tonnes of rock and 119.72 million tonnes of ore. The average grade of magnetic weight recovery of iron (MWT) in the ore body is 72.9 percent. The mining and processing capacity are considered 26 and 8 million tonnes per year, respectively. The yearly averages of grades are also constrained for the deleterious and elements of interest. Average phosphor and sulphur grades sent to the mill are constrained to a maximum of 0.2 and 1.9 percent, respectively. The average MWT grade is also bounded to be between 60 and 80 percent each year.

Since there are 19,492 blocks within the final pit, there would be 370,348 binary variables to solve the scheduling problem over 20 periods at the block level. An MILP model at this size is almost impossible to be solved. Hence, different cluster sizes have been considered and compared according to the resulted NPV, practicality of the generated schedules, size of the mathematical formulations, the number of binary variables, and computational time required for convergence. Another advantage of using the mining-cuts for solving the scheduling problem becomes apparent from the mining operation point of view. The generated schedule at the block resolution is not usually practical because: (1) the selective mining unit (SMU) usually is not equal to the block size, and (2) the generated schedule aims at maximising NPV so it chases high grade ore, this usually results in a schedule that requires frequent equipment movement and many number of drop-cuts. The mining-cuts generated could represent the selective mining unit (SMU), which will generate a more practical schedule from mining point of view.

The case studies are tested on a Dell Precision T3500 machine with a Dual Core 2.40 GHz CPU and 4 GB of RAM and the run times are presented accordingly. Although creating larger clusters takes slightly more time, the reduction in MILP solution time is significant. On the other hand, solving the MILP with larger clusters would result in schedules with lower NPV comparing to higher resolution clustering schemes. The MILP model is run on both sets of mining-cuts before and after implementing TS in order to be able to study improvements that TS can contribute to the final result and the computational time usage.

Table 4.74. MILP run summary for mining-cuts before implementing TS

ID	Number of Cuts	No. Of Binary Variables	MILP NPV (Million Dollars)	MILP CPU Time(s)	Coefficient Matrix	Non-Zero Elements
1	1,052	19,080	$1,914	166,740	164808*63120	1,553,726
2	767	13,900	$1,799	18,843	117833*46020	1,105,866
3	502	9,070	$1,746	11,638	73998*30120	687,666
4	354	6,440	$1,680	6,011	50246*21240	436,073
5	289	5,225	Infeasible	-	34923*11785	353,061

Table 4.75. MILP run summary for mining-cuts after implementing TS

ID	Number of Cuts	No. Of Binary Variables	MILP NPV (Million Dollars)	MILP CPU Time (s)	Coefficient Matrix	Non-Zero Elements
6	1,052	19,080	$1,926	103,878	160768*63120	1,507,946
7	767	13,900	$1,804	46,809	114793*46020	1,071,826
8	502	9,070	$1,752	24,334	72089*30120	666,616
9	354	6,440	$1,679	4,513	49286*21240	451,992
10	289	5,225	$1,622	2,641	39971*17340	365,832

Table 4.74 represents the run summary before implementing TS whereas Table 4.75 shows the summary of MILP run after implementing TS on the created mining-cuts. As can be

seen in both tables, the resulted NPV has a direct relationship with the resolution of the model. On the other hand, higher resolution models take more CPU time to converge to the optimal solution. Since reaching the optimal point takes a long time, the runs are restricted to a predefined gap of 2 percent in CPLEX solver. The purpose here is to have an understanding of the results of each clustering scheme.

The obvious point is that the Tabu Search has significantly contributed to the size of the coefficient matrix as well as the number of non-zero elements in it. Consequently, the CPU time required for solving the model is decreased in most cases. On the other hand, one cannot be sure judging whether TS has positive or negative influence on the NPV although its effect on CPU time is obvious. Since there is an assumption in the model that the extraction is going to be done uniformly from all the blocks in the mining-cut, it is required to have mining-cuts homogenous in grade and rock types. Consequently, the created mining-cut scheme is evaluated using two separate measures for grade and rock type, which are numerical and categorical attributes respectively. It is common to evaluate numerical attributes by the use of squared error criterion (Hsu, et al., 2007). This is the square distance of each data point from the cluster mean. Since there is only one numerical attribute having effect on the quality of the results, this distance is the same as the standard deviation of that attribute. Therefore, the first criterion is the average coefficient of variation (CV) of block grades in each mining-cut which is a normalized version of the standard deviation. Mining-cuts with lower grade variation are more effective in practice. The same thing applies to the rock type distribution. Since it is a categorical variable standard deviation or CV cannot be used. One measure used for categorical attributes is the categorical utility which is the probability of having two objects from same category in one cluster (Hsu, et al., 2007). Another measure is the relative frequency of categories in clusters. This measure puts more weight on rare categories if considered as the objective of clustering (Huang, 1997). Based on the structure of the problem and existing criteria for categorical variables, a new index is defined as the percentage of blocks in a mining-cut belonging to the most dominant rock type in that mining-cut. The effects of different mining-cut sizes as well as the Tabu Search on these indices are presented in Figure 4.123. As can be seen, CV values for S and P grades are following the same pattern as MWT although they were not considered when defining the similarity index. Another conclusion is that the manipulation by Tabu Search is making the mining-cuts less homogenous in grade and rock type.

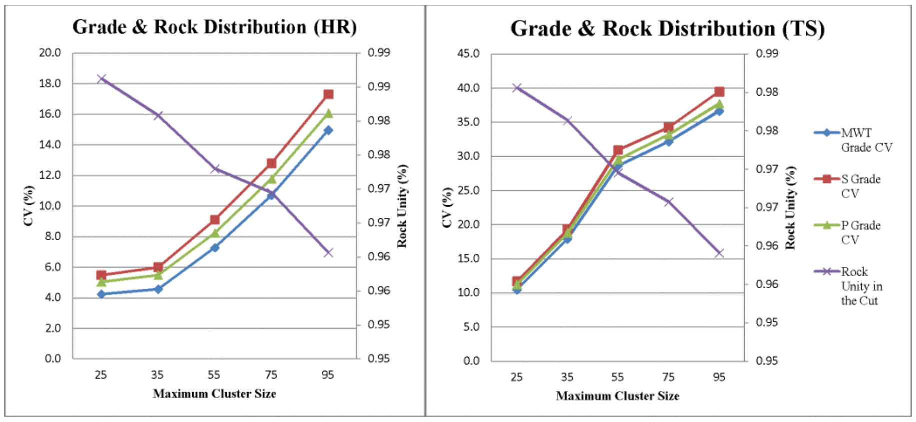


Figure 4.123. Grade and Rock Type Distribution in Generated Clustering Schemes

Since the best NPV belongs to the clustering scheme number 6, the generated mining and processing schedule, expected cash flow and average yearly grade fluctuations for all elements are presented in Figure 4.124 to Figure 4.127. The resulted schedule for plan 14 is also plotted in Figure 4.128. The bench is scheduled to be extracted totally in years 9 to 15.

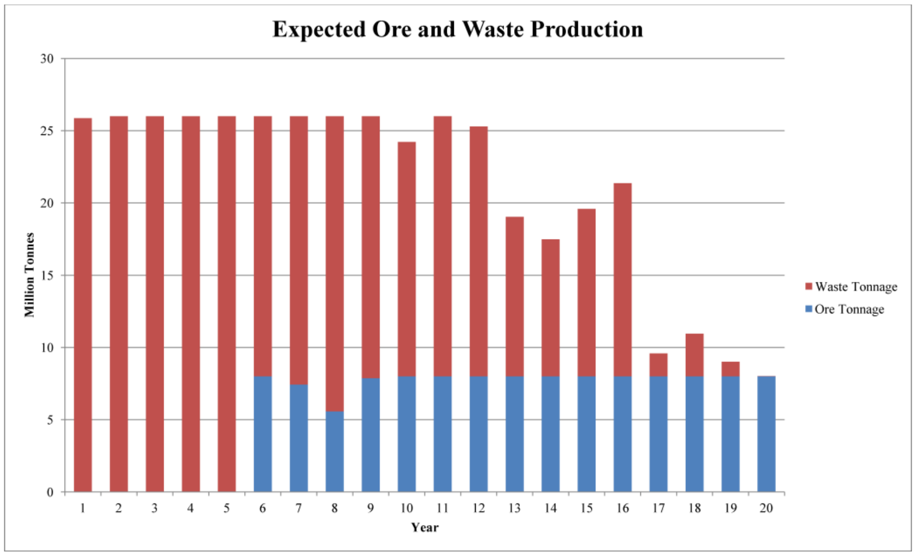


Figure 4.124. Scheduled Mining and Processing

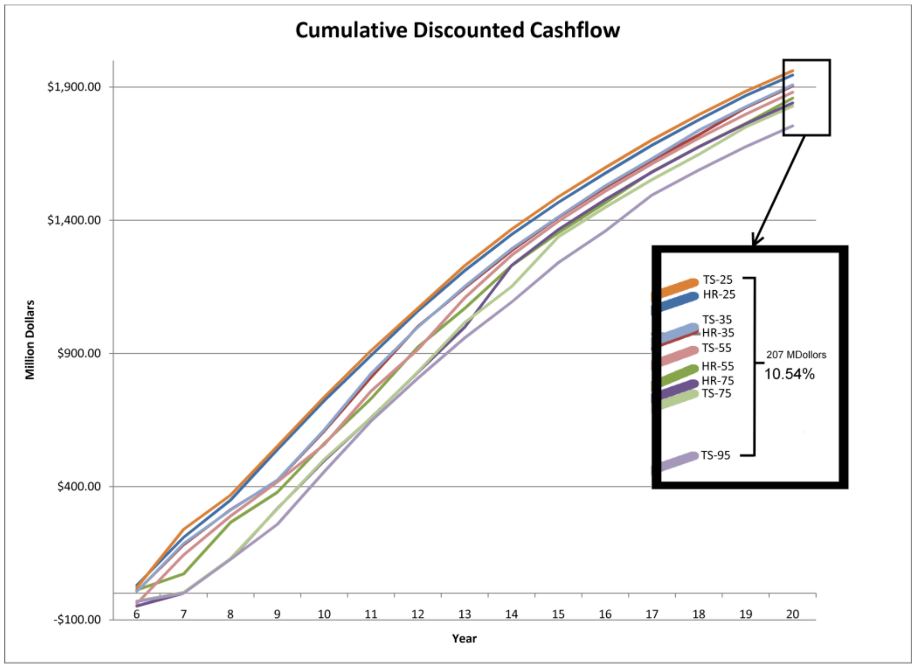


Figure 4.125. Expected Cumulative Discounted Cash Flow

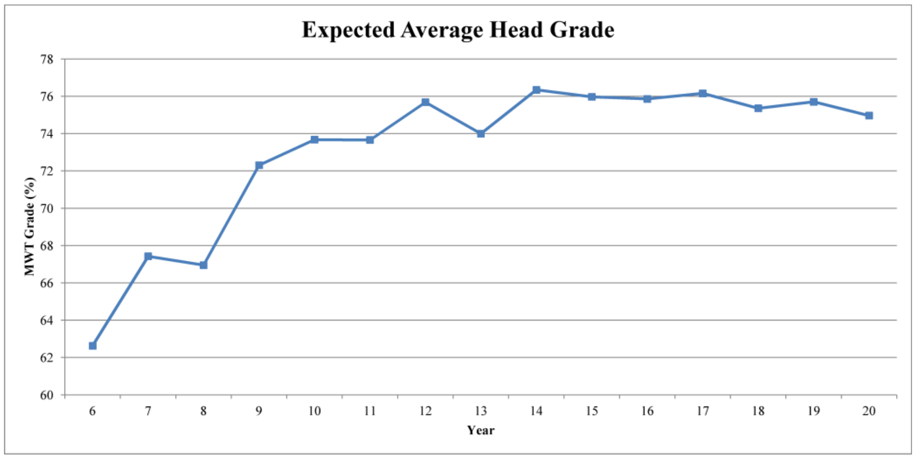


Figure 4.126. Expected Average Head Grade

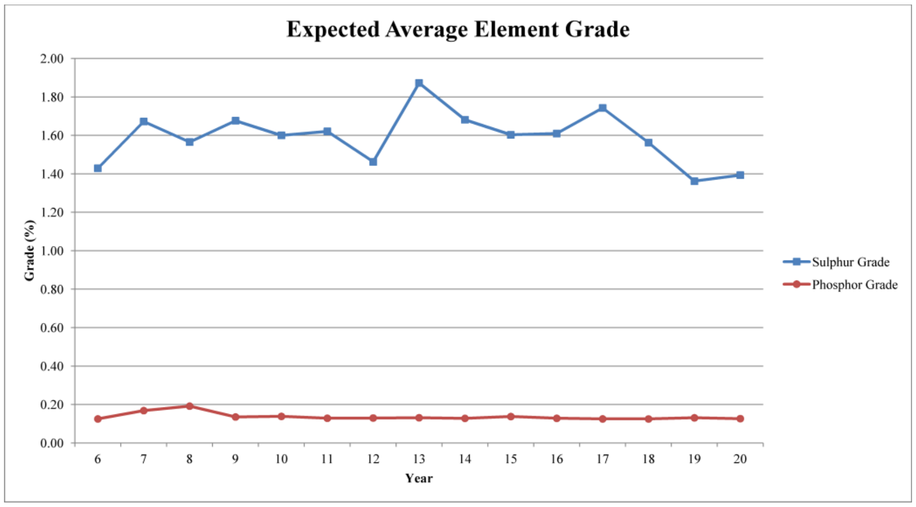


Figure 4.127. Average Expected P and S Grades

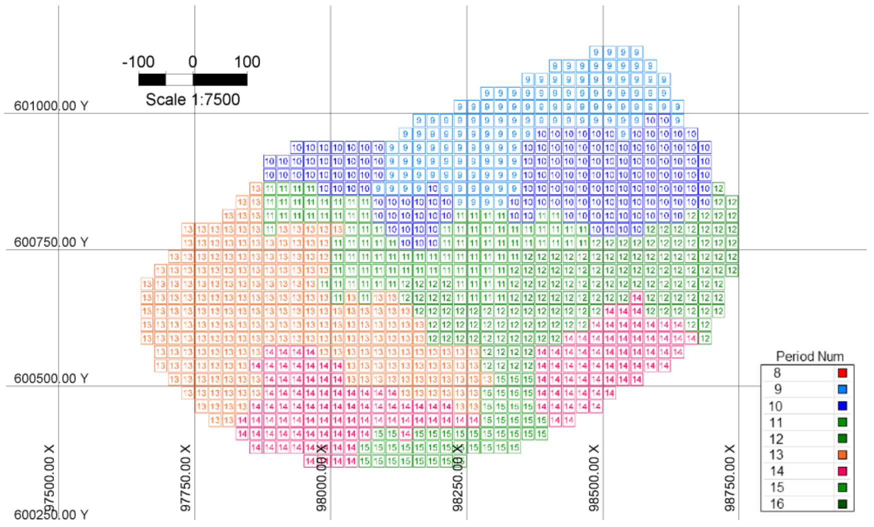


Figure 4.128. Schematic Resulted Schedule (Plan View 1580)

4.9.2 *Pushback Study*[2]

In this section we perform clustering and formulate MILP based on different pushbacks on the same iron-ore deposit. We use the parameterization approach from Whittle™ (GEOVIA, 2014b) for clustering as well as the pushback design algorithm by Mieth (2012) to study the effects of using different pushbacks on the clustering and MILP outcome. Although Mieth (2012) approach consists of multiple steps from formulating a binary integer program to solving the relaxation and using heuristics for pushback design, we call this the BIP approach in this section. The clustering is performed based on the similarity measures with distance weight and rock type penalty equal to 0.8 and 0 for other weights. The minimum, average and maximum blocks per cluster are set to 5, 30 and 35 respectively.

We used the parameterization approach with revenue multipliers starting from 0.1 and with a step size of 0.001. The parameterization approach resulted in 66 nested pits and we used Whittle™ auto pushback chooser to choose 3 and 5 pushbacks from the generated pits. However, the huge gap in rock and ore tonnage contained in pits 8 and 9 makes it difficult to find pushbacks with the same tonnage of rock or ore. Comparing the pushbacks generated with the parameterization approach versus the ones generated with the BIP approach in Figure 4.129 and Figure 4.130, the total rock tonnage is uniformly distributed among the pushbacks in the BIP approach. In contrast, the parameterization method creates a very large pushback and consequent smaller ones. Another important aspect of pushback design is its capability of providing pushbacks with reasonable and minable shapes. Figure

[2] This section is published in: Tabesh, M., Mieth, C., and Askari-Nasab, H., (2014). "A Multi-Step Approach To Long-Term Open-Pit Production Planning", International Journal of Mining and Mineral Engineering, © Inderscience Enterprises Ltd, Geneve, Switzerland, Vol. 5, 5, pp. 273-298.

4.131 to Figure 4.133 are sample plan views comparing the resulting pushbacks from the parameterization approach and the BIP approach. We also tried to perform scheduling based on 5 and 8 pushbacks but the parameterization approach results in the same 3 pushbacks that we got earlier, as opposed to our algorithm, which can provide as many pushbacks as desired. Note that this problem, with around 20,000 blocks, is solved using the relaxation bounds and exact solver without requiring the heuristic stage. The processing times are presented in Table 4.76.

Table 4.76. Summary of pushback design step

No. of Pushbacks	Maximum Rock Tonnage (MT) per Pushback	Maximum Ore Tonnage (MT) per Pushback	CPU Time (s)
3	145	40	1432
5	85	30	1196
8	55	18	1340

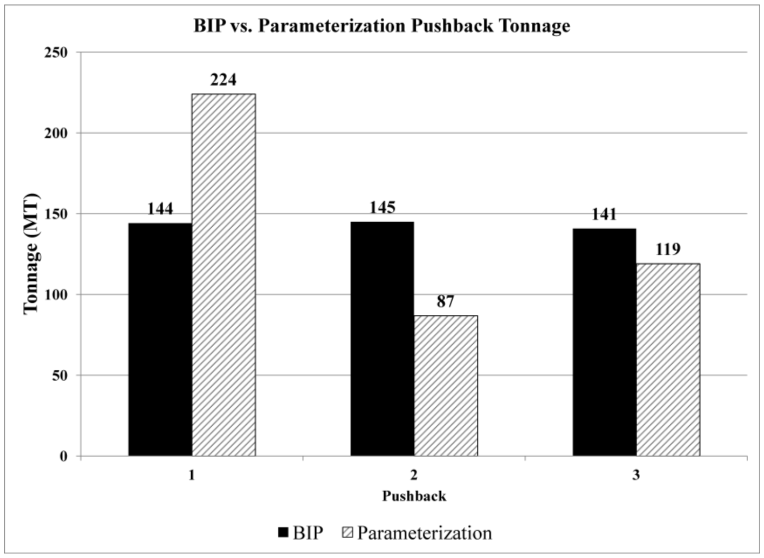


Figure 4.129. BIP vs. Parameterization (3 pushbacks)

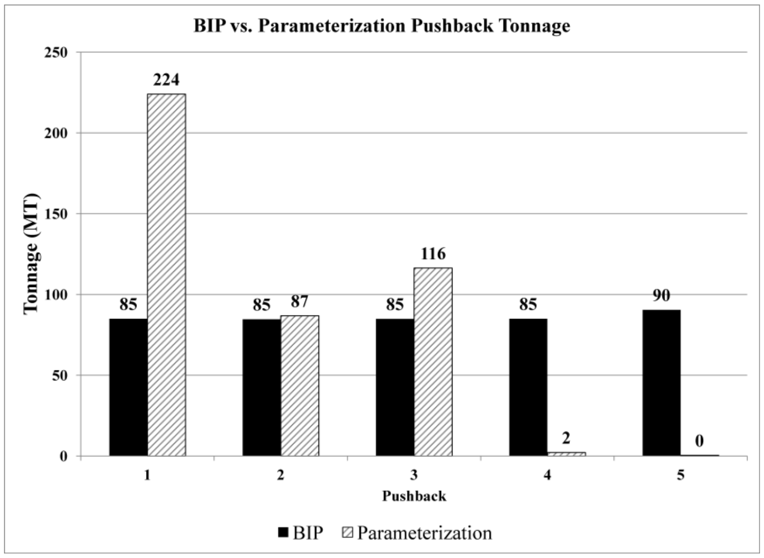


Figure 4.130. BIP vs. Parameterization (5 pushbacks)

The next step is to cluster blocks within the bench-phases and prepare the planning units for the MILP. Figure 4.134(b) shows the clustering result after performing 3 shape refinement iterations which can be compared to Figure 4.134(a) in order to see the difference. Both figures present the clustering performed based on the 3 pushbacks we found using the BIP approach. Comparing the figures against Figure 4.132 we can see that the clusters are formed strictly within the pushbacks. This restriction helps when formulating the MILP since the relationship between clusters and bench-phases as the units of planning has to be modeled.

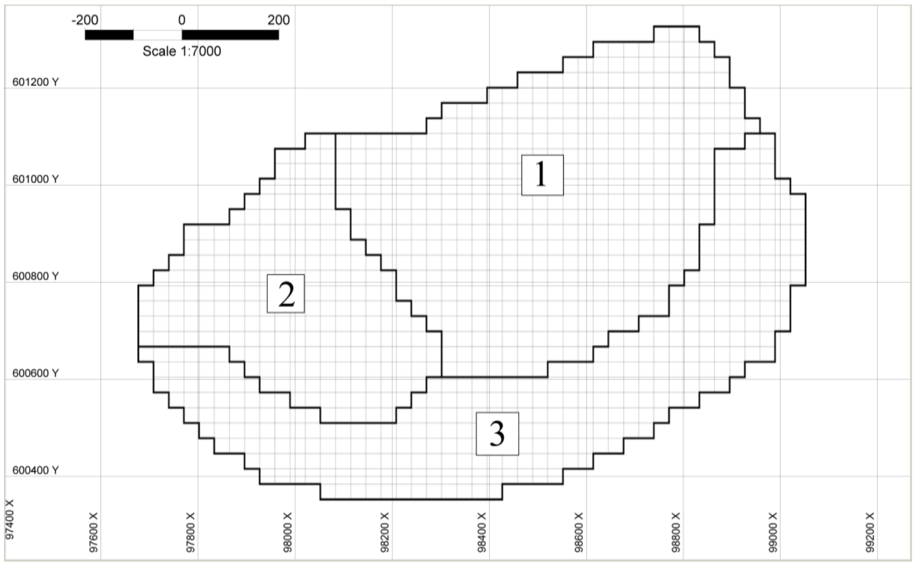


Figure 4.131. Parameterization Pushbacks (3)

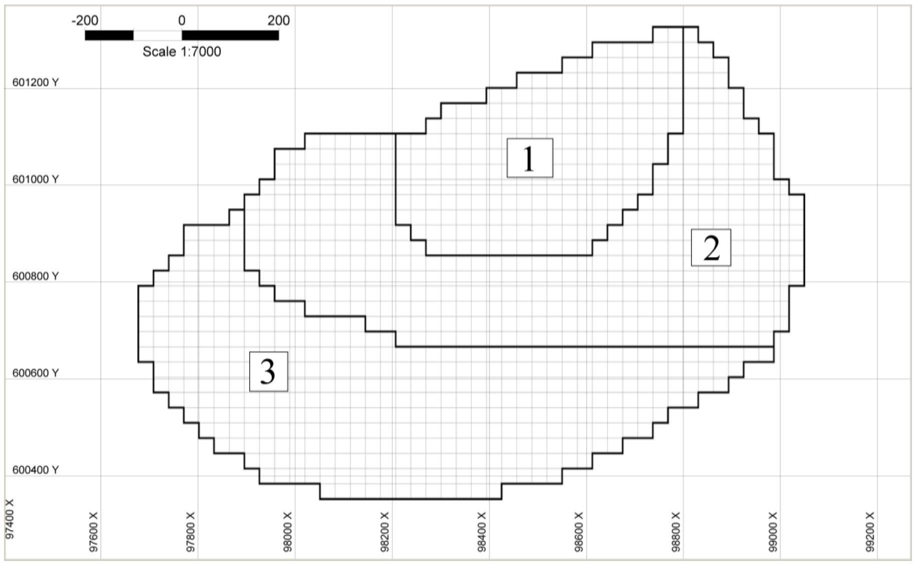


Figure 4.132. BIP Pushbacks (3)

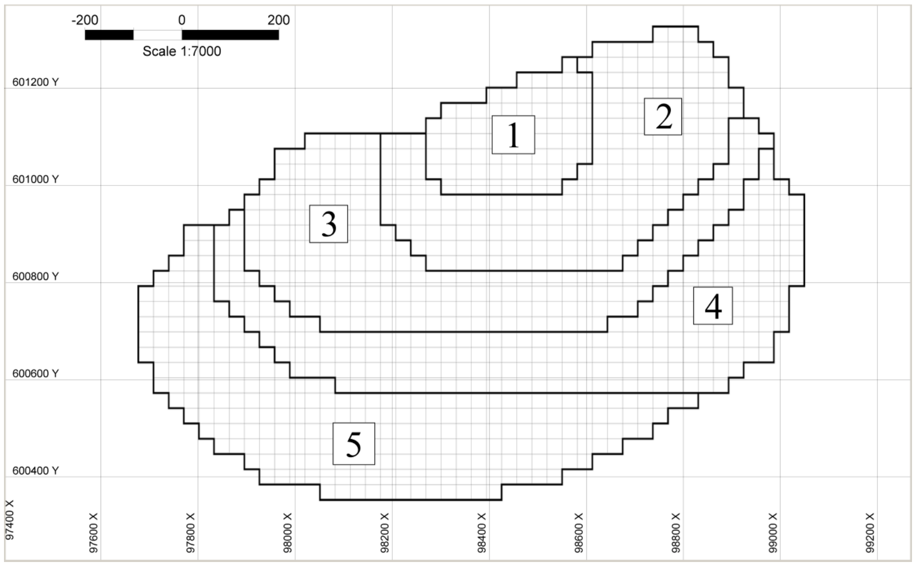


Figure 4.133. BIP Pushbacks (5)

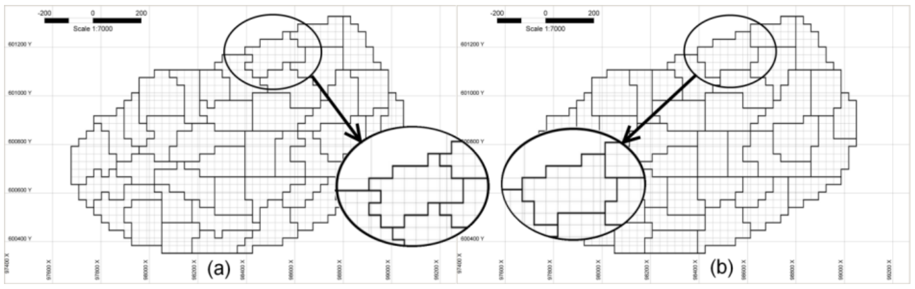


Figure 4.134. Clustering with Shape Refinement

Two sets of planning units are formed to be used in the MILP. The first set consists of the panels (bench-phases) determined using the BIP method and is used for making extraction decisions. The second set consists of clusters (mining-cuts) determined to divide the panels into minable units of planning with homogeneity in rock type and grade to result in a good estimate of the final revenue. Note that the tonnage processed from each cluster in each period is controlled by the tonnage extracted from its containing panel in the mathematical formulation. The MILP is formulated to maximize NPV (Net Present Value) based on 5 years of pre-stripping with 26 MT/yr mining capacity. A mining capacity of 25 MT/yr and a mill with 8MT/yr capacity from period 6 to the end of the planning horizon are considered. The resulting NPVs and computational processing times are presented in Table

4.77. It can be seen that formulating and solving the MILP with our pushbacks results in 2 to 4 percent increase in the NPV of the project.

Table 4.77 – NPV from MILP with different number of pushbacks

No. of Pushbacks	Parameterization pushbacks ($M)	BIP pushbacks ($M)	BIP NPV Improvement (%)
3	2163.4	2209.9	2.1%
5	2118.1	2161.2	2.0%
8	2118.1	2216.8	4.7%

Table 4.78 – CPU time from MILP with different number of pushbacks

Algorithm	No. of Pushbacks	No. of Panels	No. of Cuts	CPU Time (s)	No. of Integer Variables	No. of Non-Zeros in Coefficient Matrix
Parameterization	3	56	789	16.3	1120	192,250
BIP		56	804	7.9	1120	192,050
Parameterization	5	58	790	10.6	1160	194,166
BIP		100	823	21.1	2000	230,782
Parameterization	8	58	790	10.6	1160	194,166
BIP		150	947	90.3	3000	303,902

Another important factor in designing the pushbacks is their role in the production schedule. Pushbacks have to be determined such that they give the MILP more flexibility in accessing ore and waste blocks in all periods. It can be seen that the production schedules determined based on the pushbacks we designed are more favorable to mine engineers. BIP pushback design (Figure 4.135) is capable of providing ore to the processing plant in all periods and does not suffer from shortfalls as in the parameterization approach (Figure 4.136).

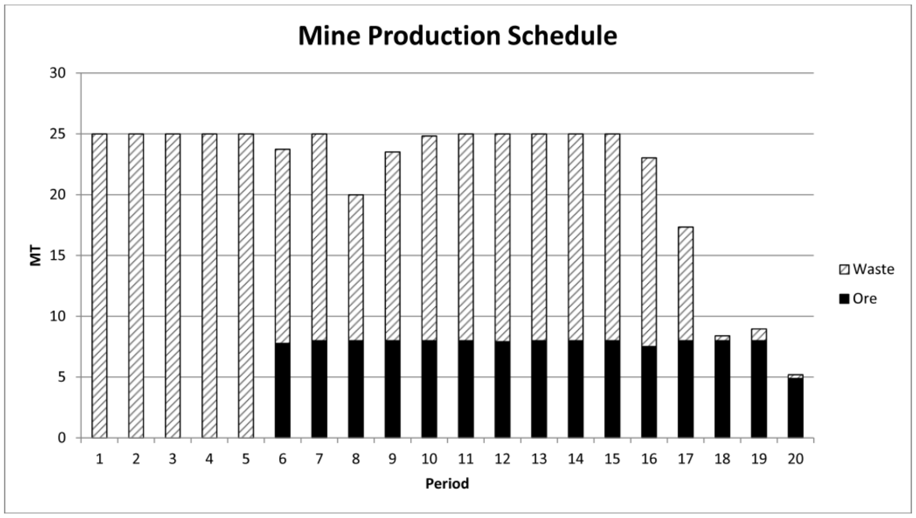


Figure 4.135. Production Schedule with 3 BIP Pushbacks

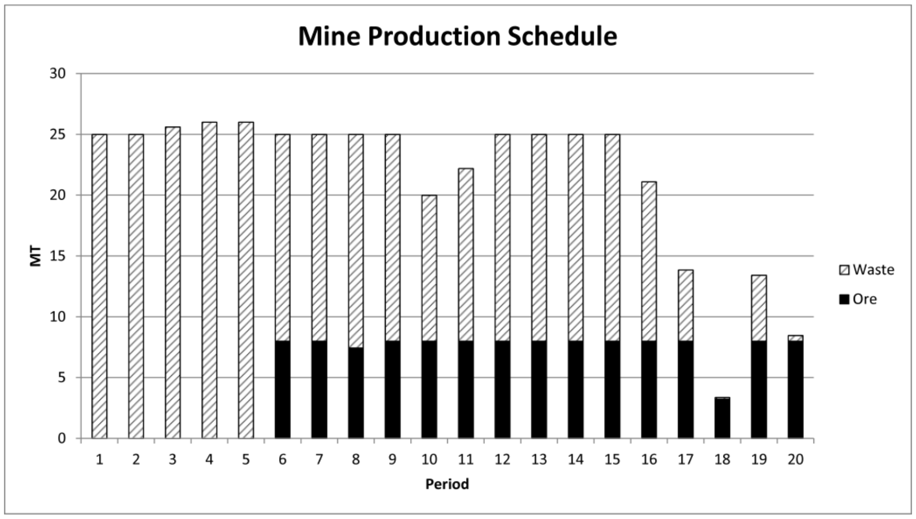


Figure 4.136. Production Schedule with 3 Parameterization Pushbacks

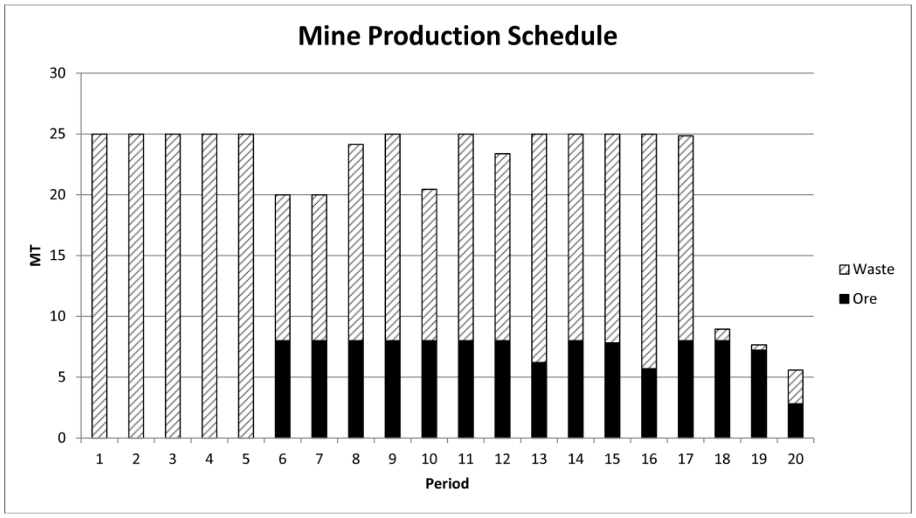


Figure 4.137. Production Schedule with 5 BIP Pushbacks

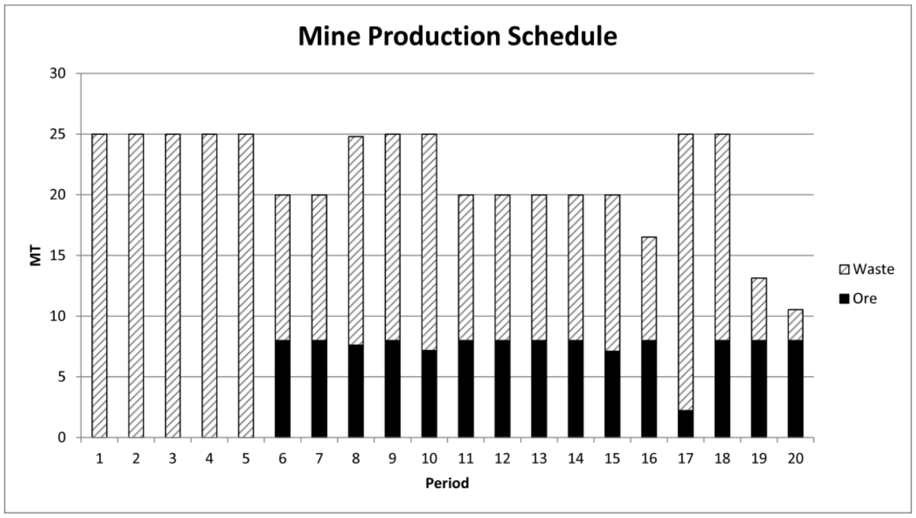


Figure 4.138. Production Schedule with 5 Parameterization Pushbacks

4.10. Large Iron-Ore Deposit[3]

Another case study is based upon the same deposit as in section 4.9 but with a higher metal price that results in a larger pit with 277,000 blocks in the ultimate pit limit. The ultimate pit limit contains around 4 billion tonnes of material with around 700 thousand tonnes of

[3] This section is published in: Tabesh, M., Mieth, C., and Askari-Nasab, H., (2014). "A Multi-Step Approach To Long-Term Open-Pit Production Planning", International Journal of Mining and Mineral Engineering, © Inderscience Enterprises Ltd, Geneve, Switzerland, Vol. 5, 5, pp. 273-298.

ore. The ore body is tabular and goes deep (Figure 4.141), which makes it very hard for the parameterization approach to find pushbacks with the same tonnages.

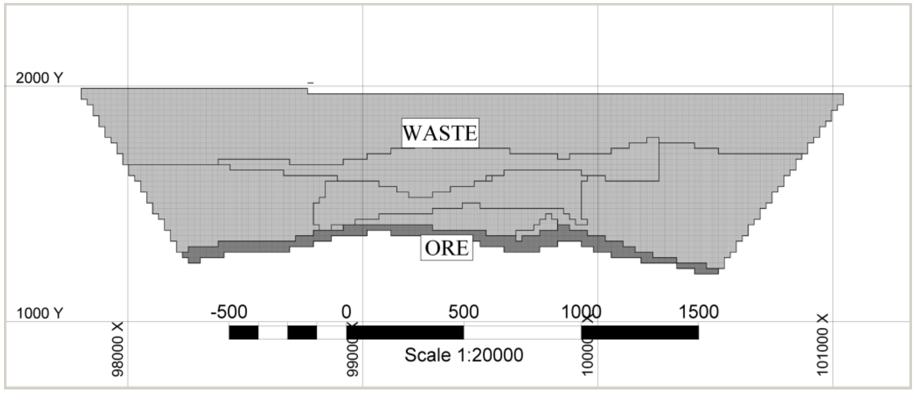


Figure 4.139. Sample vertical section looking North at 600140 Northing

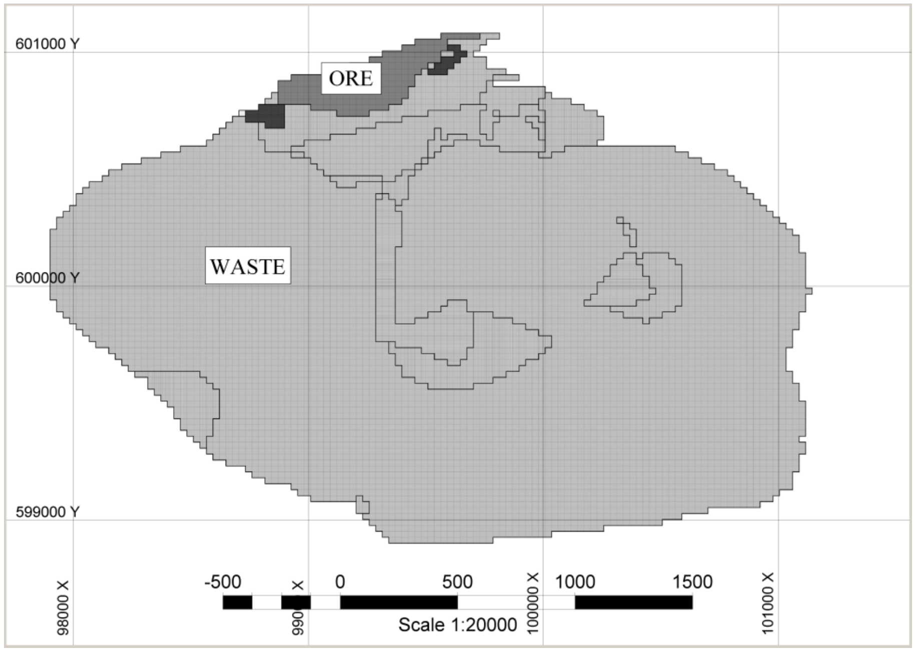


Figure 4.140. Sample plan view at 1580 meters elevation

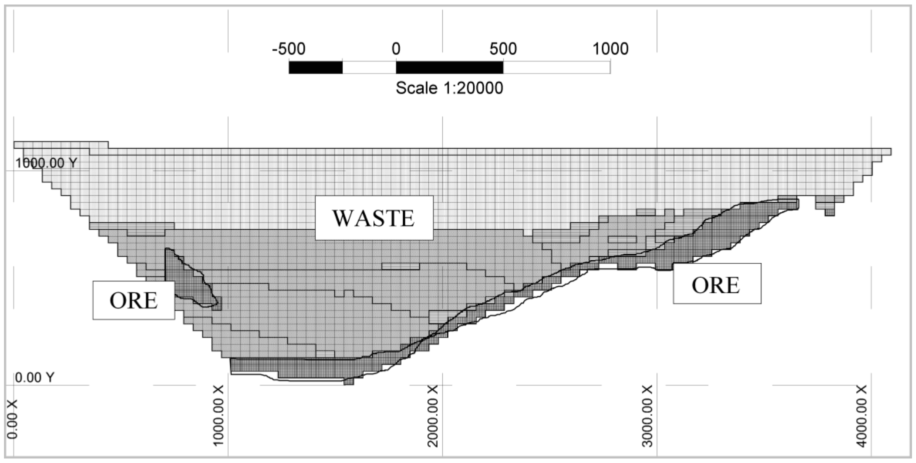


Figure 4.141. Sample vertical section looking West at 98100 meters Easting

4.10.1 *Pushback Study*

Figure 4.142 shows the comparison of tonnages when we ask for 5 pushbacks from our algorithm and parameterization approach. Figure 4.143 compares the results for 8 pushbacks. We set the maximum rock tonnage in each pushback to 800 thousand and 500 thousand tonnes for the two figures respectively. The processing times for finding the pushbacks on a machine with two quad-core CPUs at 2.8 GHz are 2.7 and 3.8 hours. Pushbacks generated with the parameterization technique vary significantly in size. The largest pushback (pushback number 3) in Figure 4.142 is 16 times larger than the smallest one (pushback number 1). The same large pushback appears when we try to find 8 pushbacks (pushback number 2 in Figure 4.143). In contrast, the maximum difference between the tonnages of rock in our pushbacks is 31% for 5 pushbacks and 11% for 8 pushbacks.

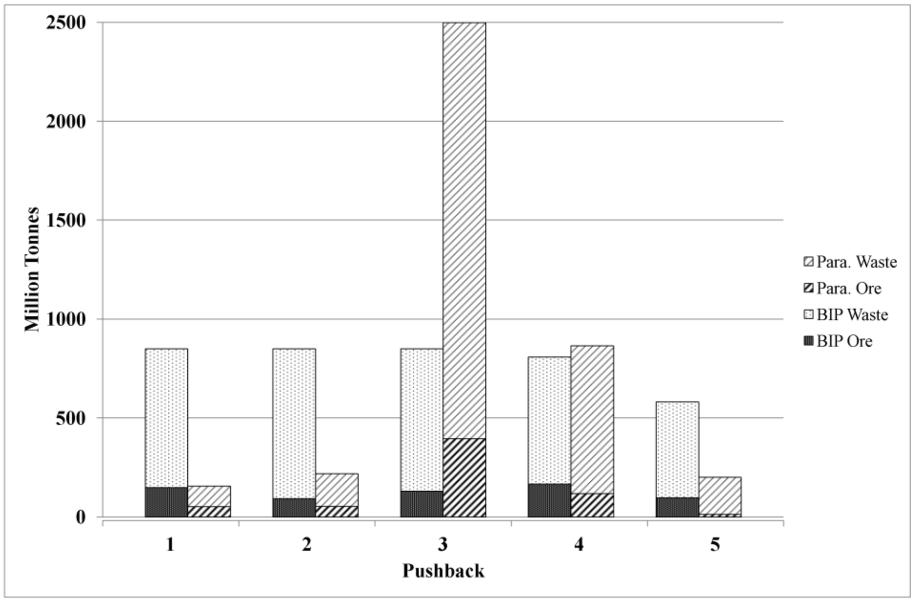


Figure 4.142. Comparison of 5 pushbacks

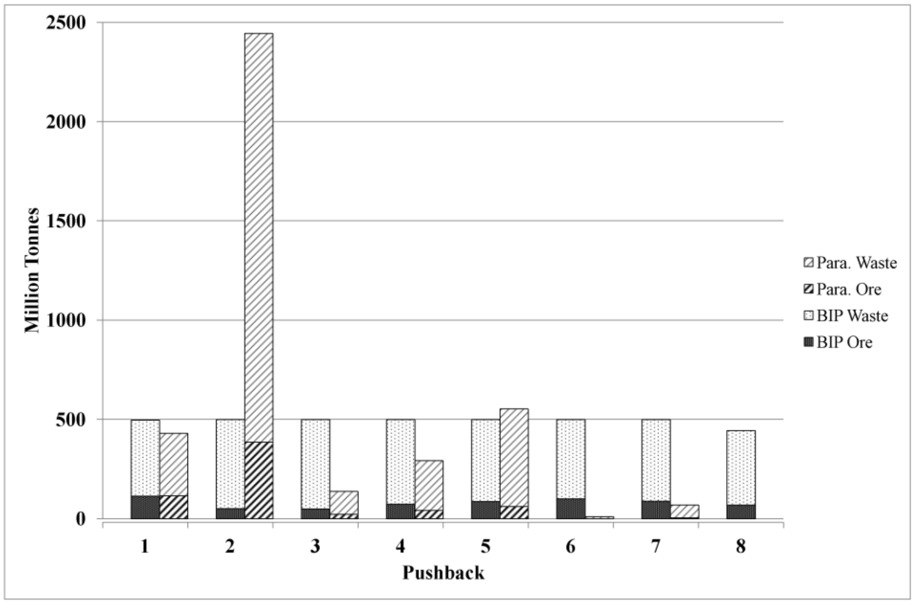


Figure 4.143. Comparison of 8 pushbacks

The shapes of the generated pushbacks are also important from the mining operations point of view. Very narrow pushbacks (e.g. pushback 5 in Figure 4.145) are not favorable since they cannot be practically mined and they do not satisfy minimum mining width requirements. On the other hand, very large pushbacks result in large panels and prevent the schedule from getting to the ore faster.

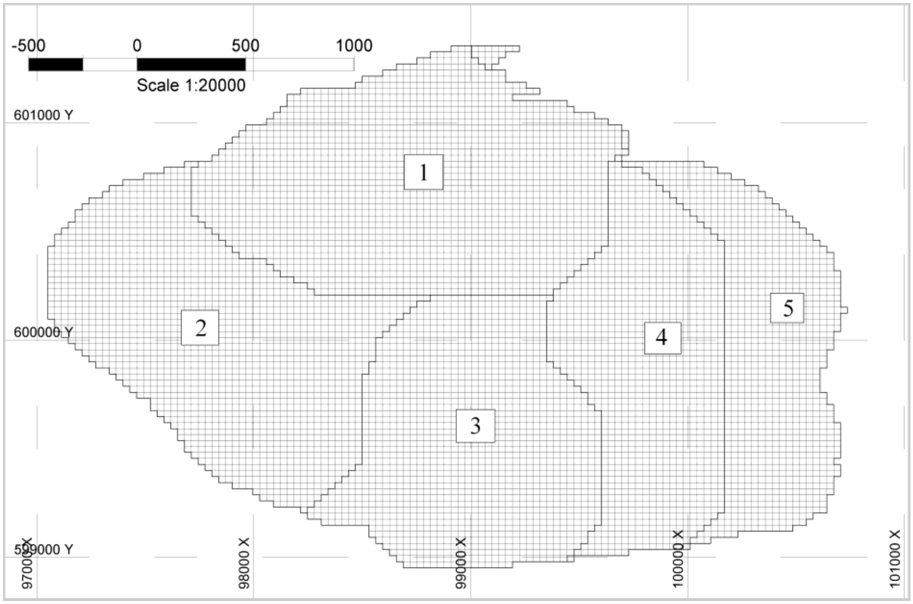


Figure 4.144. Sample plan view from the BIP/Heuristic pushbacks

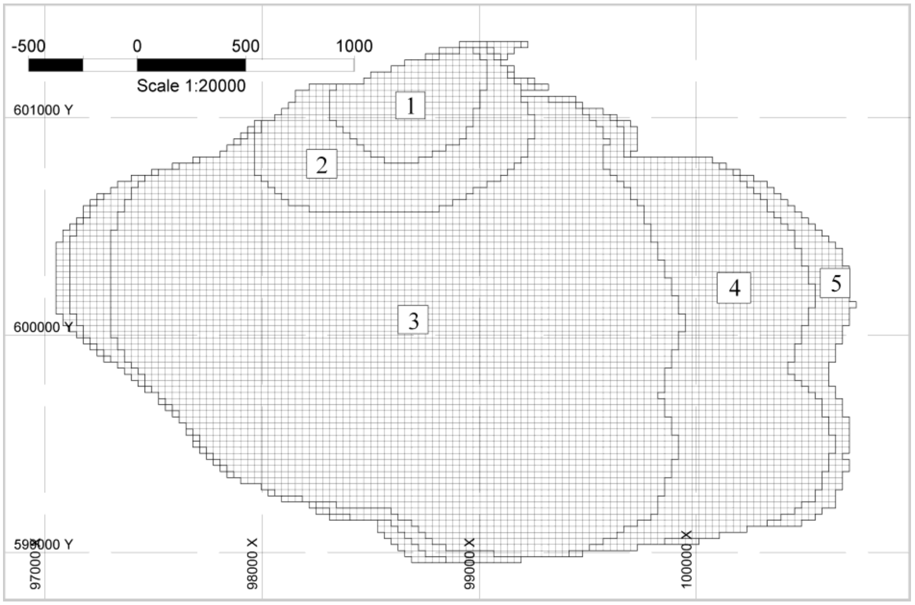


Figure 4.145. Sample plan view from the parameterization pushbacks

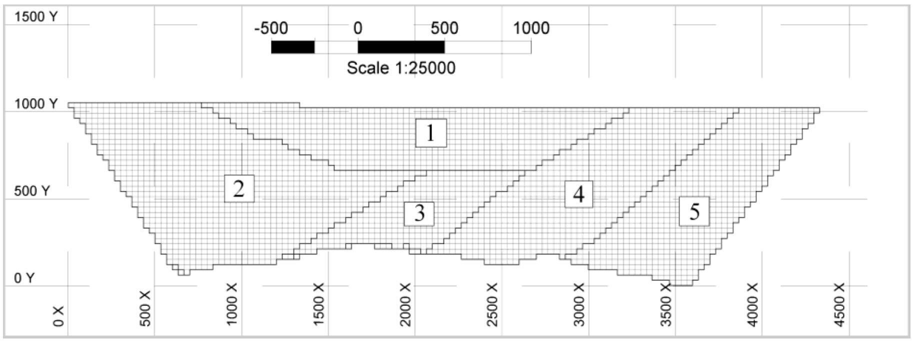


Figure 4.146. Sample vertical section from the BIP/Heuristic pushbacks

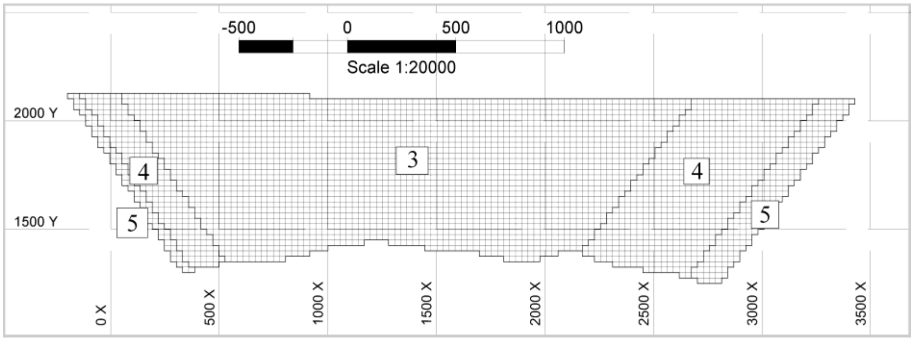


Figure 4.147. Sample vertical section from the parameterization pushbacks

4.10.2 *Clustering and MILP*[4]

In this section, the hierarchical clustering algorithm is used to form mining-cuts which are the basis for making ore and waste decisions. The determined pushbacks have to be taken into account during the clustering algorithm so that we do not end up with clusters overlapping two or more panels. An MILP is then formulated to determine the optimum life-of-mine production plan using the generated cuts and panels. We set the mining capacity to 120 MT/yr for the first 9 years, 150 MT/yr from year 10 to 20 and 100 MT/yr from year 20 to 35. The processing capacity is 10 MT/yr from year 4 to 14 and 20 MT/yr from year 15 to 35. We ran the solver to 2% optimality gap. The results are summarized in

[4] This section is published in: Askari-Nasab, H., Mieth, C., and Tabesh, M., (2013), "A Hybrid Local Search/Mixed Integer Programming Approach to Open Pit Controlled Phase-Design," in Applications of Computers and Operations Research in Mineral Industry - 36th APCOM, Porto Alegre, Brazil, November 4-8, 2013, pp. 502-514.

Table 4.79. The highest NPV belongs to the MILP based on BIP/Heuristic pushbacks which is \$6.34 billion. However, having 5 pushbacks results in 0.5% smaller NPV but offers a time saving of 82%. MILPs based on 8 and 5 pushbacks from the parameterization approach have lower NPVs and do not contribute to the processing times. We used the same parameter settings in Whittle™ (GEOVIA, 2014b) and used the Milawa algorithms to get the production schedule. The result is shown in Table 4.80. It can be seen that the MILP results in significantly higher NPV.

Table 4.79. MILP Summary

Pushback Algorithm	No. of Pushbacks	No. of Panels	No. of Cuts	NPV (\$B)	MILP CPU Times (s)
BIP/Heuristic	5	173	3,764	6.31	383
	8	263	3,642	6.34	2,161
Parameterization	5	151	4,605	5.83	303
	8	244	10,555	6.29	2,314

Table 4.80. Whittle production scheduling results

Pushback Algorithm	Scheduling Algorithm	NPV (\$B)	
		5 Pushbacks	8 Pushbacks
BIP/Heuristic	MILP	6.31	6.34
Parameterization		5.83	6.29
Parameterization	Milawa NPV	1.85	4.73
Parameterization	Milawa Balanced	1.85	2.86

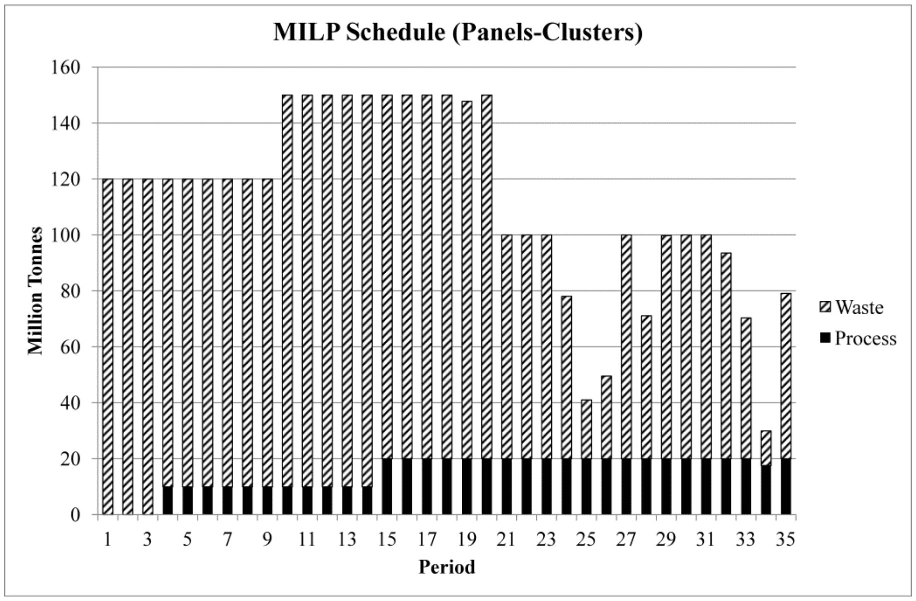


Figure 4.148. MILP Production Schedule with 5 Pushbacks

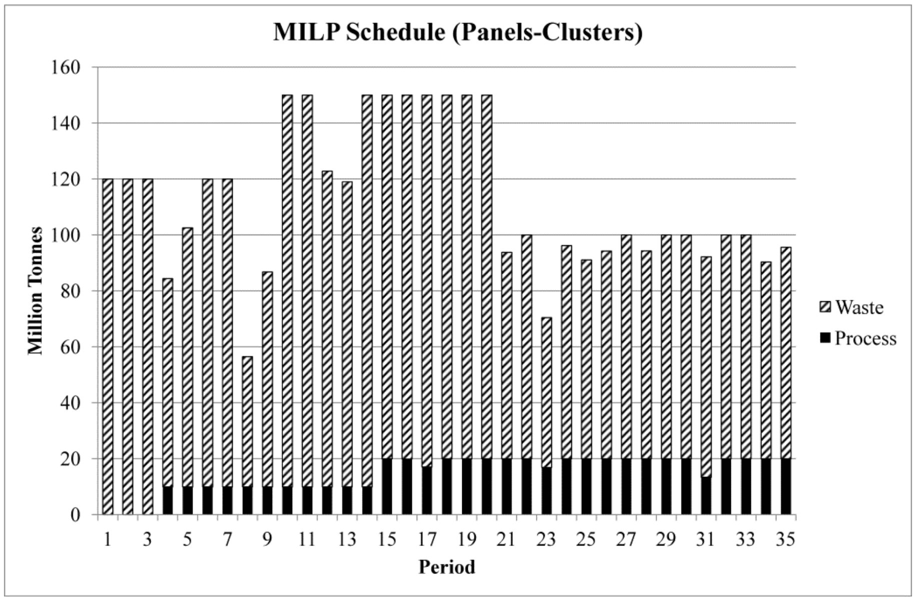


Figure 4.149. MILP Production Schedule with 8 Pushbacks

4.11. Summary and Conclusion

This chapter covered the case studies for the clustering algorithms as well as the MILP formulations. First, we used Marvin dataset, a very small synthetic dataset used by GEOVIA Whittle™ and provided in MineLib (Espinoza, et al., 2013), to study the effects of different parameters on the clustering algorithms and solve the MILP in different resolutions in order to find the best combination of mining and processing units. We ran the agglomerative hierarchical clustering algorithm with 11 different settings on the Marvin dataset to show the capabilities of the algorithm. We used different weights and similarity parameters to understand how the resulted clusters change based on the similarity factors involved. We showed that our algorithm can produce clusters homogenous in grade and rock type while respecting size and shape requirements. Moreover, we showed how mining direction and explicit boundaries can be applied to the clusters if they are being used for different purposes such as creating ore polygons for short-term planning, creating blast patterns and creating processing units within mining units. In addition, we implemented the k-means clustering algorithm on the Marvin dataset and illustrated the weaknesses of a generic clustering technique compared to our specifically designed algorithm.

Afterwards, we tested the MILP formulations by using different units of mining and processing from the Marvin dataset in order to determine the proper resolution to solve the LTOPP. We concluded that using mining-panels as mining units and clusters as processing units not only makes the problem tractable and easy to solve but also provides practical implementable schedules that do not over-estimate the value and flexibility of the operation. Next, we used the same dataset and resolution to test the stockpiling feature of the MILP formulation. Since we are applying linearization in our formulation by assuming a fixed reclamation grade for the stockpile, there will be an error in determining the expected NPV of the operation and the head grade of material fed to the plant. However, we showed that we can control both errors by restricting the stockpiles to certain input grades, which is a common practice in mining operations as well. Moreover, we showed how our approach can outperform commercial schedulers by providing more robust and realistic production schedules with or without stockpiling.

After evaluating the performances of the clustering algorithm and MILP formulation on a small synthetic dataset, we used three different real datasets to verify how our approach performs on real-size datasets. The first real case study in section 4.8 was a gold, silver and copper deposit with 292,000 blocks in the final pit. Our clustering algorithm managed to produce different sets of clusters with different settings in around 10 minutes on this dataset. Afterwards, we showed how we can use the generated clusters to formulate the LTOPP MILP model and solve it in reasonable time. In addition, we illustrated how we can use the same MILP formulation to maximize the NPV or the reserves.

In section 4.84.9, we studied the clustering algorithm and MILP on a small iron ore deposit with 19,000 blocks in the final pit. First in section 4.9.1, we implemented the clustering algorithm to create clusters with different sizes and compared the MILP results with different cluster sizes. As expected, having larger clusters decreased the processing time and resulted in lower NPV. It was shown that changing cluster size and implementing Tabu search can change the processing time in some cases by 63% and NPV by 19%. However, the decision on the proper size of the clusters has to be made based on the operation's equipment and characteristics. Moreover, since we are using the intersections of pushbacks and benches as mining units, we studied the effects of using different pushback generation algorithms on the MILP outcomes in section 4.9.2. It was shown that even in a small dataset with only 8 pushbacks using a better pushback design algorithm can result in 5% improvement in the resulted NPV.

In section 4.104.10, we studied the clustering algorithm and the MILP on larger iron-ore deposit with 277,000 blocks. We first compared different pushback generation techniques in section 4.10.1 and evaluated the pushbacks, clusters and MILP in section 4.10.2. It was shown again that a better pushback design algorithm can result in 1% to 8% improvement in the NPV of the operation. On the other hand, it was shown that using smaller pushbacks increases the processing time required to solve the MILP and the NPV of the operation. However, choosing the proper number of pushbacks is a complicated decision that is affected by multiple operation characteristics and technical considerations.

CONCLUSIONS AND FUTURE WORKS

5. CONCLUSIONS AND FUTURE WORKS

5.1. Summary of research

As high grade mineral resources become depleted and the mining industry seeks better optimized techniques and operations, better mine planning techniques play a crucial role in improving the bottom line of mining ventures. Changes in the value of the operation are significant with different production schedules over different time horizons. Among those, long-term production plans are responsible for determining the future of the operation, the revenues and costs that can be promised to stakeholders and finding out the possibilities of improving the fleet or modifying the operation. Therefore, it is important for the long-term production plans to be aligned with the real outcomes of the operation. As stated by many researchers in the area and illustrated in the forth chapter of this thesis, using blocks as units of planning results in over-estimation of the operation's profitability and flexibility. Thus, we introduced a clustering algorithm along with a mathematical formulation that can result in good production plans that are optimal, practical and do not over estimate the value of the operation.

In this thesis, we developed a clustering algorithm based on agglomerative hierarchical clustering to create mining units based on similarities between blocks. The algorithm is able to create non-fragmented mining units by merging similar adjacent blocks. Moreover, it can create clusters with controlled size and shape and aligned with the direction of mining. On the other hand, we defined four different quality measures to be able to evaluate the quality of the generated clusters. These measures are rock unity, destination dilution factor, tonnage coefficient of variation and grade coefficient of variation.

It was shown that the clustering algorithm is capable of producing units homogenous in grade, rock type and other block properties based on different settings. We implemented the clustering algorithm on Marvin, a synthetic small dataset used in GEOVIA Whittle™ and presented in MineLib (Espinoza, et al., 2013), in order to illustrate the effects of different parameters on the algorithm results. We showed that we can achieve up to 98% homogeneity in cluster rock types and less than 25% variations in element grades within the clusters by changing the weights and penalties for similarity factors. Moreover, we showed how we can balance the weights and penalties in order to achieve a clustering scheme with respect to our goals. However, since Marvin is a synthetic dataset and does not have the variability commonly found in real deposits, we implemented our clustering algorithm on three real-size block models. We also showed that our clustering algorithm can perform as good on real deposits and can cluster blocks in large datasets in reasonable time.

After evaluating the clustering algorithm, we formulated various MILP models to evaluate the performance of our MILP formulation. In this thesis, we introduced an MILP formulation for solving the LTOPP problem that uses two different sets of units for making mining and processing decisions. Our model is not based upon a fixed cut-off grade and can dynamically determine the destination of material based on values generated in each destination. Moreover, the model determines the optimal stockpiling strategy for the long-term plan at the same time it is determining the mining and processing schedules.

First, we implemented the MILP formulation on Marvin dataset and solved it in different resolutions by considering blocks, clusters and mining-panels as units of mining and processing. The goal of this step was to understand the effects of using different units for mining and processing. Since the block resolution model was not solvable in reasonable time directly, we solved it in a three-stage approach and showed that the generated solution is not practical. Moreover, we compared other possible scenarios of using blocks, clusters and mining-panels based on practicality and solution time and concluded that using clusters as processing units and panels as mining units is the best combination balancing practicality and high NPV. Afterwards, we tested the stockpiling model by allowing material to be sent to the stockpiles and reclaimed for processing. We showed that the error caused by our fixed reclamation grade assumption can be controlled by restricting the average grade of material sent to the stockpile which is the common practice in the industry. We showed that we can decrease the error in NPV and plant head grade to 1% by restricting the stockpile input grade range.

In addition to Marvin dataset, we implemented the MILP formulation on three real-size models to evaluate the performance of our formulation. We solved most models in less than few hours of processing and showed the benefits of using MILP for long-term production planning. Moreover, we illustrated the effects of using different cluster sizes in the outcome of planning and the processing time required for solving the problem. However, as mentioned earlier, the decision on the size of the clusters should be made based upon the mining equipment used in the operation and technical characteristics of the mine to be able to avoid over- or under-estimation of NPV.

Finally, we studied the effects of pushback design on MILP performance as we are using intersections of pushbacks and mining benches as mining-units. We compared pushbacks generated via the parameterization approach against the ones generated by mathematical programming and heuristics proposed by (Mieth, 2012). We concluded that using better controlled pushbacks can increase the NPV of the operation by up to 5% for the instances studied.

5.2. Conclusions

In this thesis, we reviewed the literature related to the long-term open pit production planning problem and showed the strength and weaknesses of the current body of knowledge. One of the most important shortcomings in the current LTOPP literature is the difference between the theoretical schedules and what can be delivered in the mining operation. On the other hand, the proposed planning techniques are either very time consuming or heuristics without the robustness of mathematical programming. The concept of stockpiling is also missing from most the LTOPP model and techniques. Therefore, we introduced a specific-purpose clustering algorithm along with a mathematical formulation to address these issues. We conclude this thesis by looking at the objectives and scope of research provided in Chapter 1 and present the following conclusions drawn from the theoretical modeling and implementations of the proposed approach:

- We proposed and tested an MILP formulation for LTOPP with stockpiling that maximizes the NPV of the operation while respecting technical and operational constraints.
- We proposed and tested a clustering algorithm that creates clusters of blocks homogenous in rock type and grade.
- Our clustering algorithm creates clusters that have controlled shape and size and can be used as mining-units in different planning stages.

- We created schedules, which are practical from operational point of view and obtainable in reasonable time by using the clustering algorithm along with the MILP formulation.

- Our MILP formulation simultaneously provides optimal stockpiling strategy for the long-term production plan in open pit mines.

- Our MILP formulation can be solved within a reasonable computational time, in order to evaluate multiple strategies and even realizations of the block model.

5.3. Contributions of PhD Research

This research has developed a clustering technique for aggregating mining blocks into larger units based similarities between the blocks as well as an MILP formulation for simultaneous determination of mining, processing and stockpiling strategies in an open pit mine. The resulting techniques and formulations improve the current literature on open-pit production planning by:

- Introducing a clustering algorithm that creates mining-units homogenous in rock type and grade, with controlled shape and size, and respects the mining direction as well as other constraints and boundaries, useful for creating mining aggregates in various stages of planning from long-term mining units to ore polygons and blast patterns;

- Introducing an MILP formulation for the long-term open pit production planning problem that maximized the net present value of the mine, by using two different units for making mining and processing decisions, with respect to operational and technical constraints;

- Incorporating stockpiling in the long-term open pit production planning to simultaneously optimize the mining, processing and stockpiling strategies;

- Determining the proper resolution of mining and processing units for long-term open pit production planning, by implementing the clustering algorithm and MILP formulation on multiple datasets in different resolutions and balancing the practicality of schedules with high NPV.

5.4. Recommendations for Future Work

In this thesis, we presented three interrelated contributions to the open pit production planning problem: hierarchical clustering algorithm, long-term open pit production planning MILP and stockpiling. In this section, we will provide recommendations for future work for each area separately if other researchers are going to improve upon this work and make it more valuable.

5.4.1 *Clustering*

We designed and implemented an agglomerative hierarchical clustering technique that creates mining cuts homogenous in rock type and grade with controlled size and shape. However, we believe the algorithm can be significantly improved if the following areas are studied further.

1. Defining a measure to be able to objectively evaluate the shape of the generated clusters: it will be beneficial to automate the shape evaluation by defining a quantitative measure based on mineablity of the generated clusters.
2. Including uncertainty in the clustering algorithm: when multiple realizations of the block model are available, one can run clustering multiple times and define a membership function to assign blocks to clusters; or, the similarity index can be altered to include more than one value per similarity index included. As shown in Figure 5.1, while dealing with uncertain attributes using the mean values does not completely represent similarities between the two objects.

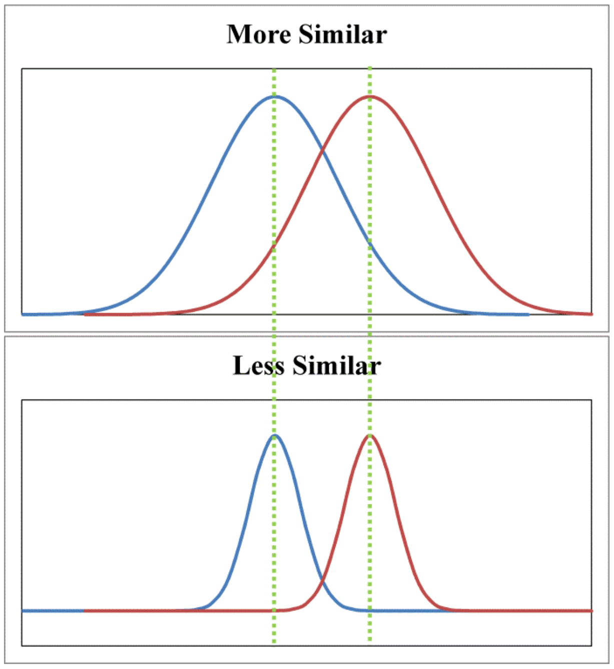


Figure 5.1.Uncertainty in Clustering

3. Add self-tuning feature: adding a self-tuning feature to the clustering algorithm can avoid repetitive implementations and evaluations of the clustering outcomes and provide better optimized weights and penalties based on defined goals and performance measures. However, this feature requires the first recommendation (shape measure) to be developed.

4. Clustering for mining-panels: it is a good idea to modify the current clustering technique to form the mining-panels the same ways mining-cuts are formed to avoid using bench-phases.

5. Correlation study: our similarity index definition looks into independent similarity factors based on their weights and penalties. However, it was shown that including too many parameters in the definition of similarity decreases the quality measures. It seems beneficial to add a correlation study prior to clustering to find the non-correlated attributes and include them in the definition of similarity index.

6. Multiple CRPs: our clustering based on mining direction is designed based on two clustering reference points per bench. However, if the bench does not have a regular shape, the mining direction has to be defined with more than two points. Adding the ability to work with multiple points or find a way to use pairs of CRP for different blocks can increase the flexibility of clustering based on mining direction.

5.4.2 ***Long-Term Open Pit Production Planning MILP***

We proposed an MILP formulation for LTOPP based upon the current literature that can uses two different units for making mining and processing decisions. The objective of the model is maximizing the NPV of the operation and includes constraints on the mining and processing capacity, head grade of material sent to destinations and precedence of extraction of mining units. However, the model can be improved and here are some recommendations for improvement.

1. Include uncertainty: one of the most discussed areas in mine planning is including uncertainty in production plans. Our goal in this thesis was to propose a formulation that runs fast so we can run it multiple times with different realizations of the block model or different values for uncertain parameters. However, it is a good idea to directly include uncertainty in the model.
2. Objective functions: our proposed model is aiming to maximize the NPV of the operation and we showed how we can run different scenarios to maximize reserve or change constraints to maximize utilization of the equipment. However, having a model that has different (or multiple) objective functions can result in improved schedules.

3. Uniform extraction assumption: an important shortcoming of our model is the uniform extraction assumption meaning that when mining a mining unit we have access to all processing units within that mining unit at all times. However, this can result in drop cuts and not respecting horizontal direction of mining, if mining-cut sizes are not defined properly. Therefore, adding constraints for horizontal mining direction or including the drop cuts in cost calculations can increase the accuracy of the model.
4. Include pushback design in model: we used intersections of pushbacks and mining benches as units of mining. However, it seems useful to incorporate pushback design phase in the planning problem to avoid the two step approach of designing pushbacks and creating the production schedule.
5. Compare by and at, compare fraction and tonnage: our model was designed based on fraction of mining- or processing-unit extracted at each period. It seems useful to test how changing the variable definitions from at period to by period and from fraction of unit to tonnage from unit can change the solution time.
6. Upper bound calculations for aggregation: since we are using aggregates for making mining and processing decisions, to have a mathematical measure for the upper bounds on the difference between aggregated and non-aggregated solutions can help decision makers to decide on the proper size of the aggregates.
7. Add constraints for market, shipping, legal, environmental impact, waste and tailings disposal.

5.4.3 *Stockpile*

In this thesis, we proposed an MILP formulation for LTOPP that considers stockpiling in the long-term plan. We used linearization to make the problem solvable with MILP solvers. Future researchers can focus on the following points to improve the formulation.

1. Solve Quadratic Programming problem: the original model we proposed for stockpiling is a quadratic programming problem but we solved a linearized version of it. The model is hard to solve as it has both quadratic objective function and constraints. Removing the quadratic constraints by relaxation or penalty methods can make the problem easier to solve and make researchers understand the differences between the original model and linearized version.

2. Change $f_{s,c}^{t} \geq 0$ variables to be in the same scale as others: we have used variables for the tonnage of material reclaimed from the stockpile and sent to processing destinations as our decisions variables. However, other variables we have in the model are fractions and restricted to be between zero and one. Therefore, changing the variable definition to a zero-one scale, for example fraction of the stockpile capacity, can help the model be solved faster.

3. Account for material losing value in stockpile: our assumption in building the LTOPP model with stockpiling was that the material recovery does not change as it remains in the stockpile for years. However, processing recoveries drop as exposed to air in the stockpile or more expensive rehandling and processing is required if remain in the stockpile for long periods. Therefore, accounting for the changes in the value and grade of material in the stockpiles can increase the accuracy of the production schedule.

REFERENCES

6. REFERENCES

[1] Alford, Christopher, Brazil, Marcus, & Lee, DavidH. (2007). Optimisation in Underground Mining. In Andres Weintraub, Carlos Romero, Trond BjÃ¸rndal, Rafael Epstein & Jaime Miranda (Eds.), *Handbook Of Operations Research In Natural Resources* (99 ed., pp. 561-577): Springer US.

[2] Askari-Nasab, Hooman, Awuah-Offei, Kwame, & Eivazy, Hesameddin. (2010). Large-scale open pit production scheduling using Mixed Integer Linear Programming. *International Journal of Mining and Mineral Engineering, 2*(3), 185-214.

[3] Bakhtavar, E., Shahriar, K., & Oraee, K. (2009). Transition from open-pit to underground as a new optimization challenge in mining engineering. *Journal of Mining Science, 45*(5), 485-494.

[4] Bao, Xiaoguang, & Liu, Zhaohui. (2012). An improved approximation algorithm for the clustered traveling salesman problem. *Information Processing Letters, 112*(23), 908-910.

[5] Barca, J.C., & Rumantir, G. (2007, 11-13 July 2007). *A Modified K-means Algorithm for Noise Reduction in Optical Motion Capture Data.* Paper presented at the 6th IEEE/ACIS International Conference on Computer and Information Science, Melbourne.

[6] Bienstock, Daniel, & Zuckerberg, Mark. (2010). Solving LP Relaxations of Large-Scale Precedence Constrained Problems. In Friedrich Eisenbrand & F.Bruce Shepherd (Eds.), *Integer Programming and Combinatorial Optimization SE - 1* (6080 ed., pp. 1-14): Springer Berlin Heidelberg DA - 2010/01/01.

[7] Bley, Andreas, Boland, Natashia, Fricke, Christopher, & Froyland, Gary. (2010). A Strengthened Formulation and Cutting Planes for the Open Pit Mine Production Scheduling Problem. *Computers & Operations Research, 37*(9), 1641-1647.

[8] Boland, Natashia, Dumitrescu, Irina, Froyland, Gary, & Gleixner, Ambros M. (2009). LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. [doi: DOI: 10.1016/j.cor.2007.12.006]. *Computers & Operations Research, 36*(4), 1064-1089.

[9] Bontoux, Boris, Artigues, Christian, & Feillet, Dominique. (2010). A Memetic Algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem. *Computers & Operations Research. Metaheuristics for Logistics and Vehicle Routing, 37*(11), 1844-1852.

[10] Brazil, M., Lee, D. H., Leuven, M. Van, Rubinstein, J. H., Thomas, D. A., & Wormald, N. C. (2003). Optimising declines in underground mines. *Transactions of the Institution of Mining and Metallurgy. Section A, Mining industry, 112*(3), 164-170.

[11] Brazil, M., & Thomas, D.A. (2007). Network optimization for the design of underground mines. *Networks, 49*(1), 40-50.

[12] Burt, Christina, & Caccetta, Lou. (2013). Equipment Selection for Surface Mining: A Review. http://www.optimization-online.org.

[13] Busnach, E., Mehrez, A., & Sinuany-Stern, Z. (1985). A Production Problem in Phosphate Mining. *Operational Research Society, 36*(4), 285-288.

[14] Caccetta, Louis, & Hill, Stephen P. (2003). An Application of Branch and Cut to Open Pit Mine Scheduling. *Journal of Global Optimization, 27*(2-3), 349-365.

[15] Chen, Jian hong, Gu, De sheng, & Li, Jian xiong. (2003). Optimization principle of combined surface and underground mining and its applications. *Journal of Central South University of Technology, 10*(3), 222-225.

[16] Chicoisne, Renaud, Espinoza, Daniel, Goycoolea, Marcos, Moreno, Eduardo, & Rubio, Enrique. (2012). A New Algorithm for the Open-Pit Mine Production Scheduling Problem. *Operations Research, 60*(3), 517-528.

[17] Cullenbine, Christopher, Wood, R., & Newman, Alexandra. (2011). A sliding time window heuristic for open pit mine block sequencing. *Optimization Letters, 5*(3), 365-377.

[18] Dantzig, G., Fulkerson, R., & Johnson, S. DO - 10.2307/166695. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America, 2*(4), 393-410.

[19] Dantzig, George B. DO - 10.2307/167356. (1957). Discrete-Variable Extremum Problems. *Operations Research, 5*(2), 266-277.

[20] Dantzig, George Bernard. (1963). *Linear Programming and Extensions*. Santa Monica, CA: RAND Corporation.

[21] Deutsch, Clayton V., & Journel, Andre G. (1992). *GSLIB: Geostatistical Software Library and User's Guide*. New York: Oxford University Press.

[22] Dhillon, Inderjit, Guan, Yuqiang, & Kulis, Brian. (2004, 2004///). *Kernel k-means: spectral clustering and normalized cuts.* Paper presented at the Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining SE - KDD '04, Seattle, WA, USA.

[23] Dimitrakopoulos, R. (2011). Strategic Mine Planning under Uncertainty. *Journal of Mining Science, 47*(2).

[24] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms [in Italian].* Politecnico di Milano, Milan.

[25] Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*: Bradford Book.

[26] Dósea, Marcos, Silva, Leila, Silva, Maria A., & Cavalcanti, Sócrates C.H. (2008). Adaptive Mean-Linkage with Penalty: A new algorithm for cluster analysis. [doi: DOI: 10.1016/j.chemolab.2008.05.007]. *Chemometrics and Intelligent Laboratory Systems, 94*(1), 1-8.

[27] Dowsland, KathrynA., & Thompson, JonathanM. (2012). Simulated Annealing. In Grzegorz Rozenberg, Thomas BÃ¤ck & JoostN. Kok (Eds.), *Handbook of Natural Computing SE - 49* (pp. 1623-1655): Springer Berlin Heidelberg.

[28] Eivazy, H., & Askari-Nasab, H. (2012). A Mixed Integer Linear Programming Model for Short-Term Open Pit Production Scheduling. *Transactions of the Institution of Mining and Metallurgy. Section A, Mining industry, 121*(2), 97-108.

[29] Elevli, Birol. (1995). Open pit mine design and extraction sequencing by use of OR and AI concepts. *International Journal of Surface Mining, Reclamation and Environment, 9*(4), 149 - 453.

[30] Epstein, Rafael, Goic, Marcel, Weintraub, Andrés, Catalán, Jaime, Santibáñez, Pablo, Urrutia, Rodolfo, et al. (2012). Optimizing Long-Term Production Plans in Underground and Open-Pit Copper Mines. *Operations Research, 60*(1), 4-17.

[31] Espinoza, Daniel, Goycoolea, Marcos, Moreno, Eduardo, & Newman, Alexandra. (2013). MineLib: a library of open pit mining problems. *Annals of Operations Research, 206*(1), 93-114.

[32] Feng, Liang, Qiu, Ming-Hui, Wang, Yu-Xuan, Xiang, Qiao-Liang, Yang, Yin-Fei, & Liu, Kai. (2010). A Fast Divisive Clustering Algorithm Using an Improved Discrete Particle Swarm Optimizer. *Pattern Recognition Letters, 31*(11), 1216-1225.

[34] Flood, Merrill M. DO - 10.2307/167517. (1956). The Traveling-Salesman Problem. *Operations Research, 4*(1), 61-75.

[35] Fytas, K., Hadjigeorgiou, J., & Collins, J. L. (1993). Production scheduling optimization in open pit mines. *International Journal of Surface Mining, Reclamation and Environment, 7*(1), 1 - 9.

[36] Garai, Gautam, & Chaudhurii, B.B. (2013). A novel hybrid genetic algorithm with Tabu search for optimizing multi-dimensional functions and point pattern recognition. *Information Sciences, 221*(0), 28-48.

[37] Gass, Saul I., & Assad, Arjang A. (2005). *An Annotated Timeline of Operations Research* (Vol. 75): Springer US.

[38] GEOVIA. (2014a). Gems (Version 6.4).

[39] GEOVIA. (2014b). Whittle (Version 4.4.0).

[40] Gershon, M. E. (1983). Optimal Mine Production Scheduling: Evaluation of Large Scale Mathematical Programming Approaches. *International Journal of Mining Engineering, 1*, 315-329.

[41] Gershon, Mark E., & Murphy, Frederic H. (1989). Optimizing Single Hole Mine Cuts by Dynamic Programming. [doi: DOI: 10.1016/0377-2217(89)90468-2]. *European Journal of Operational Research, 38*(1), 56-62.

[42] Gholamnejad, J. (2008). A zero-one integer programming model for open pit mining sequences. *South African Institute of Mining and Metallurgy, 108*(12), 759-762.

[43] Glover, F.W., & Laguna, M. (1997). *Tabu Search*: Kluwer Academic Publishers.

[44] Glover, F.W., & Laguna, M. (2013). *Tabu Search*: Springer US.

[45] Glover, Fred. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533-549.

[46] Gomory, Ralph. (1958). Outline of an Algorithm for Integer Solutions to Linear Programs. *Bulletin of the American Mathematical Society, 64*(5), 275-278.

[47] Gonzalez, Teofilo F. (1982). On the Computational Complexity of Clustering and Related Problems *System Modeling and Optimization* (Vol. 38, pp. 174-182): Springer Berlin / Heidelberg.

[48] Goodman, Gerrit V.R., & Sarin, Subhash C. (1988). Using mathematical programming to develop optimal overburden transport strategies in a surface coal mining operation. [doi: 10.1080/09208118808944137]. *International Journal of Surface Mining, Reclamation and Environment, 2*(1), 51-58.

[49] Haupt, R.L., & Haupt, S.E. (2004). *Practical Genetic Algorithms* (2 ed.): John Wiley & Sons.

[50] He, Zengyou, Xu, Xiaofei, & Deng, Shengchun. (2008). k-ANMI: A Mutual Information Based Clustering Algorithm for Categorical Data. [doi: DOI: 10.1016/j.inffus.2006.05.006]. *Information Fusion, 9*(2), 223-233.

[51] Hochbaum, Dorit S., & Chen, Anna. (2000). Performance Analysis and Best Implementations of Old and New Algorithms for the Open-Pit Mining Problem. [doi: 10.1287/opre.48.6.894.12392]. *Operations Research, 48*(6), 894-914.

[52] Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*: University of Michigan Press.

[53] Hsu, Chung-Chian, Chen, Chin-Long, & Su, Yu-Wei. (2007). Hierarchical clustering of mixed data based on distance hierarchy. [doi: DOI: 10.1016/j.ins.2007.05.003]. *Information Sciences, 177*(20), 4474-4492.

[54] Huang, Zhexue. (1997, 1997///). *A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining.* Paper presented at the In Research Issues on Data Mining and Knowledge Discovery.

[55] Huang, Zhexue. (1998). Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining & Knowledge Discovery, 2*(3), 283-304.

[56] Jain, A., & Dubes, R. (1988). *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall.

[57] Johnson, Stephen. (1967). Hierarchical Clustering Schemes. [10.1007/BF02289588]. *Psychometrika, 32*(3), 241-254.

[58] Johnson, T. B. (1969). Optimum Open Pit Mine Production Scheduling. In A. Weiss (Ed.), *A Decade of Digital Computing in the Mineral Industry* (pp. 539-562). New York: AIME.

[59] Kim, Y. C. , Cai, W. , & Meyer, W. L. (1988). Comparison of microcomputer-based optimum pit limit design algorithms Algorithms. *AIME Transactions, 284*, 1827-1830.

[60] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*(4598), 671-680.

[61] Klingman, Darwin, & Phillips, Nancy. (1988). Integer Programming for Optimal Phosphate-Mining Strategies. *The Journal of the Operational Research Society, 39*(9), 805-810.

[62] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*(1-2), 83-97.

[63] Kumral, M., & Dowd, P. A. (2005). A Simulated Annealing Approach to Mine Production Scheduling. *The Journal of the Operational Research Society, 56*(8), 922-930.

[64] Kumral, Mustafa. (2012). Production planning of mines: Optimisation of block sequencing and destination. [doi: 10.1080/17480930.2011.644474]. *International Journal of Mining, Reclamation and Environment, 26*(2), 93-103.

[65] Leontief, Wassily (Ed.) (n.d.) The Concise Encyclopedia of Economics. Library of Economics and Liberty.

[66] Lerchs, H., & Grossmann, I. (1965). Optimum Design of Open Pit Mines. *Canadian Mining and Metallurgical Bulletin, 58*, 17-24.

[67] Letchford, Adam N., Nasiri, Saeideh D., & Theis, Dirk Oliver. (2013). Compact formulations of the Steiner Traveling Salesman Problem and related problems. *European Journal of Operational Research, 228*(1), 83-92.

[68] Lin, Yi-Kuei, Yeh, Cheng-Ta, & Huang, Pei-Sheng. (2013). A hybrid ant-tabu algorithm for solving a multistate flow network reliability maximization problem. *Applied Soft Computing, 13*(8), 3529-3543.

[69] Linli Xu, James Neufeld, Bryce Larson, & Schuurmans, Dale. (2005). Maximum margin clustering. *Advances in Neural Information Processing Systems, 17*, 1537-1544.

[70] Litvinchev, Igor, & Tsurkov, Vladimir. (2003). Aggregation in Large-Scale Optimization: Springer US.

[71] MacQueen, J. B. (1967). *Some methods for classification and analysis of multivariate observations.* Paper presented at the Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.

[72] Mahajan, Meena, Nimbhorkar, Prajakta, & Varadarajan, Kasturi. (2010). The Planar K-means Problem is NP-hard. [doi: DOI: 10.1016/j.tcs.2010.05.034]. *Theoretical Computer Science, In Press, Corrected Proof.*

[73] Martinez, Michael A., & Newman, Alexandra M. (2011). A solution approach for optimizing long- and short-term production scheduling at LKABâ€™s Kiruna mine. *European Journal of Operational Research, 211*(1), 184-197.

[74] Metropolis, Nicholas, Rosenbluth, Arianna W., Rosenbluth, Marshall N., Teller, Augusta H., & Teller, Edward. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics, 21*(6), 1087-1092.

[75] Mieth, Clemens. (2012). *Pushback-design using mixed integer linear optimization.* Unpublished Master of Science, TU Bergakademie Freiberg, Freiberg.

[76] Mintec, Inc. (2014). MineSight ® 3D Advanced CAD.

[77] Mousavi, S. Meysam, & Tavakkoli-Moghaddam, Reza. (2013). A hybrid simulated annealing algorithm for location and routing scheduling problems with cross-docking in the supply chain. *Journal of Manufacturing Systems, 32*(2), 335-347.

[78] Murty, K.G. (1995). *Operations Research: Deterministic Optimization Models*: Prentice Hall PTR.

[79] Nagata, Yuichi, & Soler, David. (2012). A new genetic algorithm for the asymmetric traveling salesman problem. *Expert Systems with Applications, 39*(10), 8947-8953.

[80] Nehring, M , Topal, E , Kizil, M. , & Knights, P. (2012). Integrated short- and medium-term underground mine production scheduling. *Journal of the Southern African Institute of Mining and Metallurgy, 112*, 365-378.

[81] Newman, Alexandra M., & Kuchta, Mark. (2007). Using aggregation to optimize long-term production planning at an underground mine. *European Journal of Operational Research, 176*(2), 1205-1218.

[82] Newman, Alexandra M., Rubio, Enrique, Caro, Rodrigo, Weintraub, Andrew, & Eurek, Kelly. (2010). A Review of Operations Research in Mine Planning. *Interfaces, 40*(3), 222-245.

[83] Osanloo, M., Gholamnejad, J., & Karimi, B. (2008). Long-term Open Pit Mine Production Planning: A Review of Models and Algorithms. *International Journal of Mining, Reclamation and Environment, 22*(1), 3 - 35.

[84] Pourrahimian, Yashar, Askari-Nasab, Hooman, & Tannant, Dwayne. (2012). Mixed-Integer Linear Programming formulation for block-cave sequence optimisation. *International Journal of Mining and Mineral Engineering, 4*(1), 26-49.

[85] Rahal, D. C., Smith, M. L., van Hout, G., & van Johannides, A. (2003, 14-16 May, 2003). *The use of mixed integer linear programming for long term scheduling in block caving mines.* Paper presented at the 31st International Symposium on Application of Computers and Operations Research in the Minerals Industries (APCOM), Cape Town, South Africa.

[86] Ramazan, S., & Dimitrakopoulos, R. (2004). Recent applications of operations research in open pit mining. *SME Transactions, 316*, 73-78.

[87] Ramazan, Salih. (2007). The New Fundamental Tree Algorithm for Production Scheduling of Open Pit Mines. [doi: DOI: 10.1016/j.ejor.2005.12.035]. *European Journal of Operational Research, 177*(2), 1153-1166.

[88] Rayward-Smith, V. J., Osman, I. H., Reeves, C. R., & Smith, G. D. (1996). *Modern Heuristic Search Methods*. the University of Michigan: Wiley.

[89] Resende, M.G.C., de Sousa, J.P., & Viana, A. (2004). *Metaheuristics: Computer Decision-Making*: Springer.

[90] Roberts, B, Elkington, T, van Olden, K, & Maulen, M. (2013). Optimising combined open pit and underground strategic plan. *Mining Technology, 122*(2), 94-100.

[91] Rossi, M.E., & Deutsch, C.V. (2013). *Mineral Resource Estimation*: Springer.

[92] Samanta, B., Bhattacherjee, A., & Ganguli, R. (2005). A Genetic Algorithms Approach for Grade Control Planning in A Bauxite Deposit *Application of Computers and Operations Research in the Mineral Industry* (pp. 337-342): Taylor & Francis.

[93] Sattarvand, Javad. (2009). *Long-term open-pit planning by ant colony optimisation.* Unpublished PhD, RWTH Aachen University, Aachen.

[94] Sattarvand, Javad, & Niemann-Delius, Christian. (2008). *Perspective of Metaheuristic Optimization Methods in Open Pit Production Planning.* Paper presented at the World Mining Congress, katowice, Poland.

[95] Sattarvand, Javad, & Niemann-Delius, Christian. (2013). A New Metaheuristic Algorithm for Long-Term Open-Pit Production Planning. *Archives of Mining Sciences, 58*, 107.

[96] Sayadi, Ahmad Reza, Fathianpour, Nader, & Mousavi, Amin Allah. (2011). Open pit optimization in 3D using a new artificial neural network. *Archives of Mining Sciences, 56*(3), 389-403.

[97] Smith, Martin L. (1998). Optimizing short-term production schedules in surface mining: Integrating mine modeling software with AMPL/CPLEX. [doi: 10.1080/09208118908944038]. *International Journal of Surface Mining, Reclamation and Environment, 12*(4), 149-155.

[98] Soumis, F., Ethier, J., & Elbrond, J. (1989). Truck dispatching in an open pit mine. [doi: 10.1080/09208118908944263]. *International Journal of Surface Mining, Reclamation and Environment, 3*(2), 115-119.

[99] Souza, M.J.F., Coelho, I.M., Ribas, S., Santos, H.G., & Merschmann, L.H.C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. [doi: 10.1016/j.ejor.2010.05.031]. *European Journal of Operational Research, 207*(2), 1041-1051.

[100] Tabesh, M., & Askari-Nasab, H. (2011). Two-stage clustering algorithm for block aggregation in open pit mines. *Transactions of the Institutions of Mining and Metallurgy, Section A: Mining Technology, 120*(3), 158-169.

[101] Tabesh, M., & Askari-Nasab, H. (2013). Automatic Creation of Mining Polygons using Hierarchical Clustering Techniques. *Journal of Mining Science, 49*(3), 426-439.

[102] Tabesh, M., Mieth, C., & Askari-Nasab, H. (2014). A Multi-Step Approach To Long-Term Open-Pit Production Planning. *International Journal of Mining and Mineral Engineering, 5*(4), 273-298.

[103] The MathWorks, Inc. (2013b). Matlab. Natick, Massachusetts, United States.

[104] Tolwinski, Boleslaw, & Underwood, Robert. (1996). A scheduling algorithm for open pit mines. *IMA Journal of Management Mathematics, 7*(3), 247-270.

[105] Tomlab Optimization AB. (2011). Tomlab (Version 7.3).

[106] Vakharia, Asoo J., & Wemmerlöv, Urban. (1995). A Comparative Investigation of Hierarchical Clustering Techniques and Dissimilarity Measures Applied to the Cell Formation Problem. [doi: DOI: 10.1016/0272-6963(95)00017-M]. *Journal of Operations Management, 13*(2), 117-138.

[107] Weintraub, Andres, Pereira, Marianela, & Schultz, Ximena. (2008). A Priori and A Posteriori Aggregation Procedures to Reduce Model Size in MIP Mine Planning Models. [doi: 10.1016/j.endm.2008.01.051]. *Electronic Notes in Discrete Mathematics, The IV Latin-American Algorithms, Graphs, and Optimization Symposium, 30*(0), 297-302.

[108] Yun, Q., Liu, J., Chen, Y., & Huang, G. (1990). *Optimization of planning and design in underground mines.* Paper presented at the 22nd International Conference on Applications of Computers & Operations Research in Mineral Industry (APCOM), Berlin.

[109] Zhang, M. (2006). *Combining Genetic Algorithms and Topological Sort to Optimise Open Pit Mine Plans.* Paper presented at the 15th International Symposium on Mine Planning and Equipment Selection (MPES), Torino, Italy.

APPENDIX

7. APPENDIX I - IMPLEMENTATION

7.1. Introduction

The clustering algorithms along with the mathematical programming formulation are explained in Chapter 3. In this appendix, we describe how we implemented the algorithms and formulations in a computer application with graphical user interface. The developed application is then used to verify and evaluate the algorithms and mathematical formulations.

The application is developed using MATLAB® (The MathWorks, 2013b) and uses the TOMLAB/CPLEX (Tomlab Optimization AB, 2011) toolbox for solving the MILP models.

7.2. Preprocessing

The first step of mine planning is to have a block model representative of the deposit. Various software packages are available to create the block model. They use the drill-hole data and estimate the geology of the deposit as well as the grades of the elements and economics of mining and processing material. Since the focus of this project is on mine planning, we consider the block model as given and skip the steps required to estimate the values. We have developed multiple functions to import the block model into MATLAB® and prepare it for planning. However, they will all create a structure array designed to hold the block information and clustering and planning results. The structure array is designed such that it can hold information for blocks, mining-cuts and mining-panels without major differences so that the mathematical programming codes can use either of the units for planning. The blocks structure is summarized in Table 7.1. The block structure is created based on the following parameters:

N	Total Number of blocks/mining-cuts/mining-panels
D	Total number of possible destinations (processing plants, waste dumps and stockpiles)
R	Number of rock types in the block model
E	Number of elements in the block model
T	Number of scheduling periods

Table 7.1. Blocks Structure Array

Field Name	Field Type	Description		
		Field Name	Field Type	Description
ID	Integer	Unique ID of the block/mining-cut/-panel		
IDinBench	Integer	Block ID in the bench (required for clustering step)		
BenchID	Integer	Unique ID of the bench		
Index	Struct (3×1)	Block indices (used only for blocks)		
		XI	Integer	Block X index
		YI	Integer	Block Y index
		ZI	Integer	Block Z index
Coordinates	Struct (3×1)	Block/mining-cut/-panel coordinates		
		X	Integer	Center point Easting
		Y	Integer	Center point Northing
		Z	Integer	Center point Elevation
PitID	Integer	Pit ID of the block/mining-cut/-panel (required if the block model includes multiple pits)		
PhaseID	Integer	Phase ID of the block/mining-cut/-panel (pushback ID)		
PanelID	Integer	Panel ID of the block/mining-cut/-panel (bench-phase ID)		
Blocks	Integer Array	Block IDs of the blocks in the aggregated unit (used if referring to a mining-cut, cluster IDs if referring to a mining-panel)		
numBlocks	Integer	Number blocks in the aggregated unit (used if referring to a mining-cut or -panel)		
numRockTypeinBlock	Integer	Number of rock-types forming the block/mining-cut/-panel		
RockTypes	Binary ($R\times1$)	Binary array indicating if a rock-type exists in the block/mining-cut/-panel		
DominantRockType	Integer	Index of the dominant rock-type in the block/mining-cut/-panel		
RockUnity	Real	Rock unity of the block/mining-cut/-panel (defined in 3.3.7)		
DominantDestination	Integer	Index of the dominant destination of material from the block/mining-cut/-panel		
DDF	Real	Destination Dilution Factor of the block/mining-cut/-panel (defined in 3.3.7)		
BlockTonnage	Real	Total block/mining-cut/-panel tonnage		
MineralizedTonnage	Real ($R\times1$)	Total mineralized tonnage of each rock-type in the block/mining-cut/-panel tonnage		
WasteTonnage	Real ($R\times1$)	Total waste tonnage of each rock-type in the block/mining-		

		cut/-panel tonnage		
PossibleDestinations	Binary ($D \times 1$)	Binary array indicating if it is allowed to send material from block/mining-cut/-panel to each destination		
PossiblePeriods	Binary ($T \times 1$)	Binary array indicating if it is allowed to extract material from block/mining-cut/-panel in each period		
BlockSchedule	Real ($T \times 1$)	Fractions of block/mining-cut/-panel tonnage to be extracted in each period		
ExtractionPeriod	Integer	Period number when the largest portion of block/mining-cut/-panel is extracted		
BlockDestination	Integer	Destination ID where the largest portion of block/mining-cut/-panel is sent		
Destinations	Struct ($D \times 1$)	Struct array with a row for each destination		
		DestinationName	Char	Name of the destination
		DestinationCode	Integer	Destination ID
		RTOreTonnage	Real ($R \times 1$)	Recoverable ore tonnage if sent to this destination for each rock-type
		RTWasteTonnage	Real ($R \times 1$)	Waste tonnage if sent to this destination for each rock-type
		Revenue	Real	Total revenue if sent to this destination (undiscounted value)
		MiningCost	Real	Total mining cost if sent to this destination (undiscounted value)
		ProcessingCost	Real	Total processing cost if sent to this destination (undiscounted value)
		BlockValue	Real	Revenue – MiningCost - ProcessingCost
		DestSchedule	Real ($T \times 1$)	Fractions of block/mining-cut/-panel scheduled to be sent to this destination in each period
Elements	Struct ($E \times 1$)	Struct array with a row for each destination		
		Element	Char	Name of the element
		Quantity	Real	Total quantity of the element in the block/mining-cut/-panel
		Grade	Real	Grade of the element in the block/mining-cut/-panel
		Recoveries	Real ($D \times 1$)	Recovery of the

				element if sent to each destination
ClusterIDinBench	Integer	ID of the cluster containing this block in the bench		
ClusterID	Integer	Unique ID of the cluster containing this block		
IntraClusterSimilarity	Integer	Unique ID of the cluster containing this block		
BeneathBlock	Integer	Unique ID of the block directly underneath this block		
AboveBlock	Integer	Unique ID of the block directly above this block		
LowerDependentBlocks	Integer Array	Unique IDs of the blocks that can only be extracted if this block is extracted		
UpperDependentBlocks	Integer Array	Unique IDs of the blocks that have to be extracted prior this block		
BeneathCluster	Integer	Unique ID of the mining-cut directly underneath this block/mining-cut/-panel		
AboveCluster	Integer	Unique ID of the mining-cut directly above this block/mining-cut/-panel		
LowerDependentClusters	Integer Array	Unique IDs of the mining-cuts that can only be extracted if this block/mining-cut/-panel is extracted		
UpperDependentClusters	Integer Array	Unique IDs of the mining-cuts that have to be extracted prior this block/mining-cut/-panel		

After the definition the blocks variable, the block model data is imported from text files into MATLAB® and default field values are replaced with the values read from the text files. In cases where the block model file contains parcels, splitting block data into smaller segments with different rock types or phase IDs, the import function sums up the tonnages and assigns different ore and waste tonnages to different rock types. Moreover, the block model file may hold scheduling information which can later be used as guidelines for scheduling to speed up the process.

In addition, a secondary interface is designed to update the block model based on new cost and revenue factors as well as new destinations. This interface is capable of changing mining and processing costs, recoveries at plants and selling costs of elements which determines the block revenues.

7.3. Pushback Design

As mentioned earlier, the block model data may include phase or pushback IDs that can be used in further steps. However, two algorithms are coded in our software to determine the pushbacks from the blocks data. Since we are going to use the intersections of benches and pushbacks as mining units, this step affects the outcome of clustering and planning as we will explain in the upcoming chapters. The first algorithm is the common parameterization approach which iteratively changes the block revenues and determines the optimum pit limits based on LG (Lerchs & Grossmann, 1965) algorithm. The second algorithm is a hybrid algorithm based on binary integer programming and local search heuristics by Mieth (2012) and is capable of uniformly distributing the ore and waste tonnages among the required number of pushbacks. The mathematical formulations, heuristic techniques and case-studies for the pushback design algorithm can be found in Mieth (2012) and Tabesh et al. (2014).

7.4. Clustering Algorithms

Two clustering algorithms are developed for this project. The first algorithm is an agglomerative hierarchical clustering algorithms based on similarity factors explained in Section 3.3.1. The second algorithm is a k-means variation based on gradient descent explained in Section 3.3.6. Here we discuss the programming and implementation of the algorithms in Matlab. The algorithms have been implemented, tested and improved multiple times and what we present here is the outcome of numerous changes and performance improvements based on MATLAB® capabilities.

7.4.1 *Agglomerative Hierarchical Clustering Algorithm*

Since the blocks are only allowed to be grouped together only if they are located on the same bench, the clustering algorithm is performed on the blocks of each bench separately. This helps reduce the memory required and speed up the algorithm. The clustering algorithm starts from the lowest bench and continues until it reaches the topography. After clustering blocks on each bench, the program updates the relationships between the generated clusters and the blocks on the upper bench, evaluates the quality of the generated clusters, and calls the post-processing and Tabu search procedures if required.

The first step in clustering is to calculate the similarity and adjacency matrices. Assuming that there are n blocks on the bench, the similarity matrix is an $n \times n$ matrix with double precision values where the adjacency matrix is of the same size but binary values. The matrices are calculated using MATLAB® matrix operations instead of loops in order to decrease the processing time. For readability, the vectors are presented with lower case letters and matrices are presented with upper case letters. The clustering procedure can be broken down into the following steps.

1. Reading the Inputs: the block information as well as clustering parameters is required for the operation. The block data is input to the function as a number of vectors holding grade, rock-types, etc. There are also clustering parameters including weights, penalties and the available clustering options such as maximum cluster size and clustering within pushback and destination boundaries. If a directional clustering scheme is required, the clustering reference points for the bench have to be provided. The details of the inputs are summarized below.

n	Number of blocks on the bench (1×1)

l	Block length (1×1)
w	Block width (1×1)
dw	Distance weight in similarity matrix (1×1)
gw	Grade weight in similarity matrix (1×1)
rp	Rock-type difference penalty in similarity matrix (1×1)
dp	Destination difference penalty in similarity matrix (1×1)
mcs	Maximum number of blocks in a cluster (1×1)
mnc	Maximum number of clusters (1×1)
wpb	Clustering within pushback boundaries (1×1)
wsb	Clustering within predetermined schedule boundaries (1×1)
wdb	Clustering within predetermined destination boundaries (1×1)
$CRPs$	Clustering reference points (2×2)
dct	Directional clustering type: 1:Spherical / 0:Vector (1×1)
x	X coordinate of the blocks ($n\times1$)
y	Y coordinate of the blocks ($n\times1$)
p	Phase IDs the blocks ($n\times1$)
t	Predetermined periods of extraction ($n\times1$)
d	Predetermined destinations ($n\times1$)
g	Major element grades of the blocks ($n\times1$)
r	Block rock-types ($n\times1$)
c	Cluster IDs in the lower bench (beneath cluster) ($n\times1$)

2. Calculating the similarity matrix: in order to increase the performance of the calculation step, all the block data vectors are replicated to create two $n\times n$ matrices which one of them is a transposed version of the other one. Therefore, comparing the two using $\sim=$ operator or deducting one from another results in a matrix with all the pair-wise differences in that parameter. For example, the distance matrix and rock-type difference calculation are illustrated in Table 7.2 and Table 7.3 respectively. The weights and penalties are then applied to these matrices and the similarity matrix is calculated accordingly.

Table 7.2. Distance calculation in Matlab

```
X1 = repmat(x,[1,n]);
X2 = X1';
Y1 = repmat(y,[1,n]);
Y2 = Y1';
DS = sqrt((X1 - X2).^2 + (Y1 - Y2).^2);
NDS = DS./max(max(DS));
```

Table 7.3. Rock-type difference calculation in Matlab

```
R1 = repmat(r,[1,n]);
R2 = R1';
R = R1 ~= R2;
```

3. Calculating the adjacency matrix: the initial adjacency matrix is calculated based on the distance matrix. Blocks with a distance of one block are considered to be adjacent. However, to account for difference in the length and width of the blocks, the x and y vectors are divided by block length and width respectively. Accordingly, a block can have a maximum of 4 adjacent blocks as illustrated in Figure 7.1.

Table 7.4. Adjacency calculation in Matlab

```
X1 = repmat(x/l,[1,n]);
X2 = X1';
Y1 = repmat(y/w,[1,n]);
Y2 = Y1';
DS = sqrt((X1 - X2).^2 + (Y1 - Y2).^2);
A = DS <= 1;
```

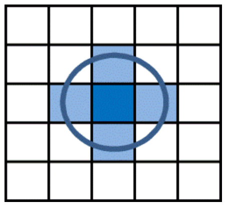


Figure 7.1. Block adjacency

4. In the next step, the adjacency matrix is multiplied by any boundary matrix required. For example, if the clustering has to be performed within pushback boundaries, the adjacency matrix is updated as in Table 7.5. Other boundaries can be applied using the same technique. Therefore, two blocks on the two sides of a boundary line will not be considered as adjacent and will not be merged together in any future step.

Table 7.5. Updating adjacency matrix based on pushback boundaries in Matlab

```
If $wpb == 1$
    $P1 = repmat(p,[1,n]);$
    $P2 = P1';$
    $A = A.*(P1 == P2);$
End
```

5. Initialize clusters: as is common among all agglomerative hierarchical clustering algorithms, the clustering starts by assuming every object is a cluster. Therefore, we initialize the clusters by considering every block a cluster and set the number of clusters equal to number of blocks on the bench.

6. Finding maximum similarity: in this step, we created a while-loop to choose clusters to merge until the number of clusters drops below the number of clusters required or there are no clusters to merge. In every iteration of the loop, we find the maximum similarity in the similarity matrix, for example, value in (i, j) cell of the matrix. If merging the two clusters i and j does not violate the maximum cluster size constraint the two clusters are merged. Otherwise, the adjacency and similarity of the two clusters will be set to zero to avoid being selected again. Afterwards, the adjacency and similarity matrices are updated accordingly. The pseudo code is presented in Table 7.6.

Table 7.6. Clustering loop

```
$NumberofClusters = NumberofBlocks$
$ClustersSize = 1(NumberofBlocks)$
While $NumberofClusters > MaximumNumberofClusters$
    $[i, j, MaxSimilarity] = Max(S)$
    If $MaxSimilarity == 0$
        $break;$
    End
    If $ClusterSize(i) + ClusterSize(j) > MaxClusterSize$
        $A(i, j) = 0; S(i, j) = 0;$
    Else
        $S(i,:) = Min(S(i,:), S(j,:)); S(:,i) = Min(S(:,i), S(:,j));$
        $S(j,:) = 0; S(:,j) = 0;$
        $A(i,:) = Max(A(i,:), A(j,:)); A(:,i) = Max(A(:,i), A(:,j));$
        $A(j,:) = 0; A(:,j) = 0;$
        $ClusterSize(i) = ClusterSize(i) + ClusterSize(j);$
        $ClusterSize(j) = 0;$
        $NumberofClusters = NumberofClusters - 1;$
    End
End
```

7. Update precedence relations: we update the relationships between blocks of the bench and lower and upper bench, after creating the clusters. The blocks and clusters located above and below each blocks is determined and based on that, the precedence relationships between clusters.

7.4.2 *Tabu Search*

If we decide to run the Tabu search procedure, it will be called after clustering each bench and updating the relationships between blocks and clusters of the consequent benches. The first step in Tabu search is to evaluate the current clustering scheme and calculate the measure of goodness for it. We initialize the temporary blocks and clusters variables to the original blocks and clusters variable; and afterwards, we get in a loop for the determined number of iterations. The following steps are taken in every iteration of Tabu search:

1. Identify clusters and blocks to alter: there are two parameters set by the user that control the number of clusters and number of blocks in each cluster to alter. We used two nested loops to identify the clusters with maximum number of precedence arcs and blocks within those clusters that cause the maximum number of arcs. Table 7.7 presents the pseudo code for this step.

Table 7.7. Determine clusters and blocks to alter

```
For $iLoop = 1 : NumberofClusterstoAlter$
    $i = Max(TempClustersNumPrecedenceArcs);$
    $ListofClusterstoAlter(iLoop) = i;$
    For $jLoop = 1 : NumberofBlockstoAlter$
        $j = Max(TempBlocksNumPrecedenceArcs(Blocks(i)));$
        $ListofBlockstoAlter(iLoop) = j;$
        $TempBlocksNumPrecedenceArcs(j) = 0;$
    End
    $TempClustersNumPrecedenceArcs(i) = 0;$
End
```

2. Altering clusters: after preparing the list of candidates for alterations, we will prepare the alternate solutions (clustering schemes). For this step, we first check if we can detach the candidate block from its cluster without causing fragmentation in the cluster. To do so, a subroutine, based on graph theory, is developed that we will discuss in the next step. If the candidate block is detachable, we check for the number of clusters adjacent to the candidate block. Afterwards, we detach the candidate block from its original cluster and attach it to the adjacent clusters to create alternative solutions. We evaluate the measure of goodness for all proposed alternative solutions and choose the one with the maximum measure. Then, we

check the alternative solution with highest measure against Tabu list. We replace the original clustering scheme with the alternative solution if it does not exist in the Tabu list. Finally, the solution is added to the Tabu list and the algorithm moves on to the next iteration. The pseudo code is provided in Table 7.8.

Table 7.8. Create alternative solutions

```
For iLoop = 1 : NumberofClusterstoAlter
    For jLoop = 1 : NumberofBlockstoAlter
        If IsBlockDetachable(iLoop, jLoop)
            AdjacentBlocks = ListofAdjacentBlocks(iLoop);
            For kLoop = 1 : NumberofAdjacentBlocks
                NewClusterID = AdjacentBlocks(kLoop).ClusterID;
                AlternateSolution(AltSolutionEnumerator) = OriginalSolution;
                AlternateSolution(AltSolutionEnumerator)
                     .Blocks(iLoop).ClusterID = NewClusterID;
                UpdateAlternateSolutionMeasure(AltSolutionEnumerator);
            End
        End
    End
End
i = max(AlternateSolutionMeasure);
If ~ IsInTabuList(AlternateSolution(i))
    OriginalSolution = AlternateSolution(i);
    AddtoTabuList(AlternateSolution(i));
End
```

3. Check if block is detachable: we have developed a function that uses the neighbor blocks as well as graph theory functions to determine if a block is detachable from its cluster or if detaching the block causes fragmentation in the cluster. In order to use graph theory functions in Matlab, we assume every block is a node and the cluster is a bidirectional graph of these nodes. Afterwards, we create a sparse matrix where the value of element (i, j) is one if blocks i and j are adjacent. We

then check for four different scenarios regarding the number of adjacent blocks from the same cluster.

a) If a block has four adjacent blocks from the same cluster (block 3 in Figure 4.1), it is definitely not detachable as it is in the middle of a cluster and there is no adjacent cluster to attach the block to.

b) If a block has only one adjacent block from the same cluster (block 1 in Figure 4.1), it is definitely detachable.

c) If a block has two adjacent blocks from the same cluster, we use graph shortest path function to determine if the block is detachable by removing the arcs connecting the block to its neighbors. If there is a path from the first neighboring block to the second neighboring block, the block is detachable (block 5 in Figure 4.1). If there is not a path from the first neighboring block to the second neighboring block, the block is not detachable (block 4 in Figure 4.1).

d) If a block has three neighbor blocks from the same cluster, we repeat the mentioned procedure by checking for paths from the first neighbor to the second neighbor and from the first neighbor to the third neighbor. In Figure 4.1, both blocks 2 and 6 have three neighbors but block 2 is detachable while block 6 is not.

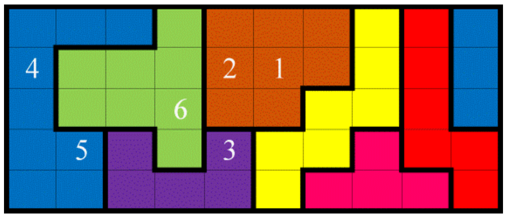


Figure 7.2. Detachable Block Scenarios

7.4.3 *Shape Refinement*

The shape refinement is a post-processing iterative procedure that can be called after clustering for a specific number of iterations. The procedure removes small clusters and smoothes the corners of the generated clusters for each bench. In every iteration of the procedure, corner blocks from all clusters are identified. A corner block is a block that has one adjacent block from the same cluster or it is a cluster by its own. The corner block will be detached from its original cluster and attached to an adjacent cluster if it has more than one adjacent block from the other cluster. Next, clusters with less than threshold number of blocks are exploded into single-block clusters. The process continues for a predetermined number of iterations.

7.5. Mathematical Programming

In this section, we explain how we use the outputs from clustering and use them to form the matrices required for MILP modeling. Four sets of units are used in this section: blocks, cuts, panels and phases. All these units have the same structure as explained earlier. Blocks are the smallest planning units that are imported into the program from other software. Cuts are the clusters formed using different clustering algorithms and are going to be used as processing units. Panels are the intersections of pushbacks (phases) and benches and are going to be used as mining units. Phases are mine pushbacks determined prior to import or by Mieth algorithm. Phases are not being used now but are considered for future expansions of the model if pushback precedence is going to be implemented.

7.5.1 *Prepare Units*

The first step is to choose the units of planning and production. There are four options available: block-block, block-cut, cut-cut and cut-panel. We have developed four different functions for preparing the matrices and name them cuts and panels for ease of use. In order to avoid having multiple codes for creating MILP matrices we will name the chosen processing unit cuts and mining units panels not looking at the unit itself. For example, if we want to solve the block-block resolution model we will make a copy of the blocks and name it cuts and another copy with the name panels. We save the output of this stage into two variables: "inputToMILP_Cuts.mat" and "inputToMILP_Panels.mat". The contents of the files are presented in Table 7.9.

Table 7.9. Variable Information

File Name	Variable Name	Data Type	Description
inputToMILP_Cuts.mat	Cuts	Struct $(1\times n)$	Structure array holding cuts information
	numCuts	Double (1×1)	Number of cuts
	cutsAbove	Cell $(n\times 1)$	IDs of the predecessor cuts
	miningCuts	Cell $(n\times 1)$	IDs of blocks within each cut
inputToMILP_Panels.mat	Panels	Struct $(1\times n)$	Structure array holding panels information
	numPanels	Double (1×1)	Number of panels
	panelsAbove	Cell $(n\times 1)$	IDs of the predecessor panels
	panelCuts	Cell $(n\times 1)$	IDs of cuts within each panel

7.5.2 *Setup Problem*

We have developed a GUI for setting the parameters required for forming the MILP model. Mining and processing capacities, stockpile settings, head grade and blending and other parameters required for planning are set in this step and saved into two variables:

"inputToMILP_Parameters .mat" and "SPParams.mat". The first file contains planning parameters and variables required for the setup GUI and the second file contains the variables for defining stockpiles.

7.5.3 *Update Stockpile Destinations*

If stockpiles are defined in the setup stage, they are added to the possible destination for cuts and the corresponding revenues and costs have to be updated. Therefore, we developed a function for updating the cuts and panels variables for including the stockpile called "updateDestinations4SP". In this function, we add stockpiles to the destinations of cuts. These dummy destinations do not have any revenue or cost associated with them. However, the possibilities of sending material to each of these destinations are updated based on the stockpile settings for different rock types. The rehandling costs and revenues are included in later stages for reclamation variables.

7.5.4 *Matrix Structure*

Since we use Tomlab/Cplex to solve the MILP model, we have to prepare the model in a matrix format. Seven matrices are required as the inputs for Tomlab/Cplex:

1. C: is a $(n \times 1)$ matrix holding all the variable coefficients for the objective function
2. A: is a $(m \times n)$ matrix holding all the variable coefficients in the constraints
3. b_L: is a $(m \times 1)$ matrix holding all the lower bound values for the constraints
4. b_U: is a $(m \times 1)$ matrix holding all the upper bound values for the constraints
5. x_L: is a $(n \times 1)$ matrix holding all the lower bound values for the variables
6. x_U: is a $(n \times 1)$ matrix holding all the upper bound values for the variables

7. IntVars: is a row matrix holding the indices of the integer variables

The sizes of the first six matrices have to comply with each other and, more importantly, they have to follow the same order. We will use matrix A to explain the order of definitions as it has to comply with both variables and constraints. The order can then be extended to matrices related to variable (C, x_L, x_U and IntVars) and matrices related to constraints (b_L and b_U). As explained in section ….., there are four groups of variables defined in the MILP: $x_{p,d}^{t}$, y_{m}^{t} , b_{m}^{t} and $f_{s,c}^{t}$.

8. The first group of variables is $x_{p,d}^{t}$ which is the fraction of processing units sent to each destination in each period. Therefore, there are $P \times CS \times T$ variables representing this group where P is the number of processing units, CS is the number of processing destinations plus the number of stockpiles, and T is the number of planning periods. Consequently, columns 1 to $P \times CS \times T$ in the A matrix correspond to $x_{p,d}^{t}$ variables. The variables in this group are defined as in Table 7.10.

Table 7.10. Variable Information

$x_{1,1}^{1}$	$\cdots$	$x_{P,1}^{1}$	$\cdots$	$x_{1,CS}^{1}$	$\cdots$	$x_{P,CS}^{1}$	...	$x_{1,1}^{T}$	$\cdots$	$x_{P,1}^{T}$	$\cdots$	$x_{1,CS}^{T}$	$\cdots$	$x_{P,CS}^{T}$
$d=1$			$\cdots$	$d=CS$			$\cdots$	$d=1$			$\cdots$	$d=CS$		
$t=1$							$\cdots$	$t=T$						

9. The second group of variables is y_{m}^{t} which is the fraction of mining extracted from the mine in each period. Therefore, there are $M \times T$ variables representing this group where M is the number of mining units, and T is the number of planning periods. Consequently, columns $P \times CS \times T + 1$ to $P \times CS \times T + M \times T$ in the A

matrix correspond to y_m^t variables. The variables in this group are defined as in Table 7.11.

Table 7.11. Variable Information

y_1^1	$\cdots$	y_M^1	...	y_1^T	$\cdots$	y_M^T
$t=1$			$\cdots$	$t=T$		

10. The third group of variables is b_m^t which is the binary decision variable indicating if all the predecessors of mining unit m are completely extracted by or in period t. Therefore, there are $M \times T$ variables representing this group where M is the number of mining units, and T is the number of planning periods. Consequently, columns $P \times CS \times T + M \times T + 1$ to $P \times CS \times T + M \times T \times 2$ in the A matrix correspond to b_m^t variables. The variables in this group are defined as in Table 7.12.

Table 7.12. Variable Information

b_1^1	$\cdots$	b_M^1	...	b_1^T	$\cdots$	b_M^T
$t=1$			$\cdots$	$t=T$		

11. The forth group of variables is $f_{s,c}^t$ which is the tonnage of material extracted from stockpile s and sent to processing destination c in period t. Therefore, there are $S \times C \times T$ variables representing this group where S is the number of stockpiles, C is the number of processing destinations, and T is the number of planning periods. Consequently, columns $P \times CS \times T + M \times T \times 2 + 1$ to $P \times CS \times T + M \times T \times 2 + S \times C \times T$ in the A matrix correspond to $f_{s,c}^t$ variables. The variables in this group are defined as in Table 7.13.

Table 7.13. Variable Information

$f_{1,1}^{1}$	$\dots$	$f_{S,1}^{1}$	$\dots$	$f_{1,C}^{1}$	$\dots$	$f_{S,C}^{1}$	...	$f_{1,1}^{T}$	$\dots$	$f_{S,1}^{T}$	$\dots$	$f_{1,C}^{T}$	$\dots$	$f_{S,C}^{T}$
$c=1$			$\dots$	$c=C$			$\dots$	$c=1$			$\dots$	$c=C$		
$t=1$							$\dots$	$t=T$						

Next, we will explain the order of constraint definitions for the A matrix. There are nine groups of constraints explained in the order they are developed in the code. We create a function for each group of constraints to make it easier to track and debug.

12. Mining tonnage constraints: the first group of constraints are the mining capacity constraints which are controlling the total tonnage extracted from the mine in each period. Therefore, there are T rows in matrix A corresponding to the mining tonnage constraints.

13. Processing tonnage constraints: the second group of constraints are controlling the tonnage of ore sent to each processing destination or stockpile pile. Therefore, there are $(C+S)\times T$ rows in matrix A corresponding to the processing tonnage constraints.

14. Grade blending constraints: the next group of constraints control the average grade of material sent to processing plants and stockpiles. Since the structures of constraints are different for processing destinations and stockpiles they are created separately. However, both sets of constraints require rearranging to avoid nonlinearity and the lower and upper bound constraints have to be created separately. The first $C\times E\times T$ constraints are the lower bounds for average grade values for the processing destinations. The second group are $S\times E\times T$ rows corresponding to lower bound constraints for average element grades to stockpiles.

The same order is followed for upper bound constraints for processing destinations and stockpiles respectively.

15. Mining and processing tonnage control: the next group of constraints control the relationship between the tonnage mined from panels and tonnage processed from cuts. Therefore, there are $M \times T$ constraints corresponding to every panel in each period.

16. Precedence constraints: the precedence constraint function creates three sets of constraints. The first group consists of $M \times T$ rows controlling the relationship between the binary variables b_m^t and continuous variables y_m^t for each panel in each period. The second group consists of $M \times T$ rows controlling the relationship between the binary variable for each panel and the summation of fractions extracted from its predecessor panels. The third group consists of $M \times T$ rows controlling the relationship between b_m^t and b_m^{t+1}. Note that the last M rows in this group are not given any values but are kept to have consistent dimensions for the three groups.

17. Reserve constraints: this group of constraints are corresponding to the summation of y_m^t values for each panel. The constraints can be defined in two ways: extract everything within the ultimate pit or let the optimizer decide what to extract. The difference between the two is in the lower bound of the constraints. In both cases, there are M constraints in this group.

18. Destinations constraints: this group of constraints are corresponding to the summation of $x_{p,d}^t$ values for each cut. These constraints make sure that the

summation of cut fractions sent to different destinations does not exceed 1. There are P constraints in this group.

19. Stockpile tonnage constraints: this group of constraints control the relationship between the tonnage of material sent to the stockpile and the tonnage reclaimed from the stockpile and sent to different processing destinations. As mentioned earlier, we tried to avoid defining variables for stockpile inventory and these constraints are based on cumulative tonnage sent to the stockpile from period 1 to period $t-1$ and the cumulative tonnage reclaimed from stockpile from period 1 to period t. There are $S \times T$ constraints in this group.
20. Stockpile content constraints: as mentioned earlier, there is an error regarding estimation of stockpile grade with a fixed number. Therefore, there is a chance that the solution is not feasible as the average grade of material in the stockpile is always less than the assumed reclamation grade. In this situation, the element content reclaimed from the stockpile will be more than the element content sent to the stockpile. This group of constraints prevent the model from offering a solution that the element content reclaimed from the stockpile is more than the element content sent to the stockpile for each element. Therefore, there are $S \times E \times T$ constraints in this group.

7.5.5 *Preprocessing*

After creating the matrices we call a preprocessing procedure before calling the solver with the generated matrices. In this step, we remove some of the variables and constraints from the matrices. The first step is to update cut possible destinations. In this step, if a cut is made of a rock type that cannot be sent to a destination (for example, if stockpiles are limited to specific rock types) the corresponding $x_{p,d}^{t}$ variables are marked for removal. Next, the predecessor and successor cones are determined for each panel and the possible periods for each panel are updated. Consequently, corresponding y_{m}^{t} variables based on the two cones are marked for removal. Similarly, if we are using an initial solution to limit the extraction periods and reduce problem size, the y_{m}^{t} variables outside the determined range are marked for removal. Afterwards, $x_{p,d}^{t}$ variables related to those y_{m}^{t} variables (cuts within those panels) will be marked. Finally, we set the upper bound on the variable to zero and the optimizer will remove the variable at the beginning of the process.

7.5.6 *Cplex Optimizer*

After preparing the matrices and preprocessing we call the Tomlab/Cplex solver to solve the model. The most significant options we use for Tomlab/Cplex solver are the EPGAP, EPMRK, branchprio and knapsack heuristic. EPGAP is the optimality gap that is set by the user and lets the optimizer stop when the optimality gap falls below this threshold. EPMRK is the amount of difference from integer that a value for integer variables is allowed to get as a feasible solution. branchprio is the priority of branch and bound procedure for integer variables. For this setting we set the priority based on periods i.e. variables corresponding to earlier periods have higher priority. Finally, we ask Cplex to apply its knapsack heuristic since our model can be considered as an extension to knapsack problem and have similar structure.

7.5.7 *Interpreting Results*

After solving the MILP model with Tomlab/Cplex, we have to interpret the results. Since the output of the solver is a vector of values, the first step is to break up the solution vector into $x_{p,d}^{t}$, y_{m}^{t}, b_{m}^{t} and $f_{s,c}^{t}$ variables based on the order defined earlier. However, the important point to consider is that some variables are removed from the model before sending to the solver. Afterwards, we assign extraction and destination schedules to panels and cuts based on uniform extraction assumption. Next, we calculate the production, destination and stockpile schedules. Table 7.14 summarizes the variables calculated and saved in this stage.

Table 7.14. Variable Information

File Name	Variable Name	Data Type	Description
MILPResults.mat	scheduledProcessingCutsKDT	double $(P \times D \times T)$	Fraction of cuts sent to each destination in each period ($x_{p,d}^{t}$)
	scheduledMiningPanelsBT	Double $(M \times T)$	Fraction of panels extracted in each period (y_{m}^{t})
	scheduledRehandlingStockpilesSPT	double $(S \times C \times T)$	Total tonnage of material reclaimed from each stockpile and sent to each processing destination in each period ($f_{s,c}^{t}$)
	RTOreTonnageKDT	double $(P \times D \times T)$	Tonnage of ore sent from each cut to each destination in each period
	RTWasteTonnageKDT	double $(P \times D \times T)$	Tonnage of waste sent from each cut to each destination in each period
	TotalDestTonnagesKDT	double $(P \times D \times T)$	Tonnage of material sent from each cut to each destination in each period
	elemGradeEDT	double $(E \times D \times T)$	Average grade of each element sent to each destination in each period (excluding material reclaimed from stockpile)
	cutOffsEDT	double $(E \times D \times T)$	Lowest grade of each element sent to each destination in each period (excluding material reclaimed from stockpile)
	SPRealGrade	double $(E \times S \times T)$	Calculated average grade of each element in each stockpile in each period
	SPBlendingGrade	double $(E \times S \times T)$	Calculated average grade of each element sent to each processing destination in each period (including material reclaimed

			from stockpile)
	scheduleDestTonnageDT	double $(D \times T)$	Tonnage of material sent to each destination in each period
	scheduleMiningTonnageT	double $(T \times 1)$	Tonnage of material mined in each period
	scheduledStockpilesInventoryST	double $(S \times T)$	Total tonnage of material in each stockpile in each period
	Panels	Struct $(M \times 1)$	Panels structure updated with mining schedules
	Cuts	Struct $(P \times 1)$	Cuts structure updated with mining and destination schedules
	RevenuesDT	double $(D \times T)$	Revenues made in each destination in each period
	CashFlowsT	double $(T \times 1)$	Total cash flow of the operation in each period
	discountedCashFlowsT	double $(T \times 1)$	Total discounted cash flow of the operation in each period

8. APPENDIX II – GRAPHICAL USER INTERFACE

8.1. Step 1: Prepare “.dat” Block Model

In order to start working with the clustering application, we need a text file containing the block information. This file is a tab-separated text file with 3 sections: the header, column headers and the block data. It can be exported from any mine planning software and formatted to adhere to the following features. It has to be saved with a “.dat” extension. You can find a sample block model file in “2014_MILP/InputData/Test/Blocks.dat”. Open the file in a text editor to have an example of formatting the block model file.

8.1.1 *The Header*

The header consists of 17 lines as follows (“-…” at the end of each line is a comment and is not imported into Matlab):

Table 8.1. Header Lines

Line #	Description	Example	Notes
1	Number of Destinations	4{tab}-Number of Destinations	
2	Destination Names	MILL{tab}Process{tab}W005{tab}-np-{tab}-Destination Names	The destination names should match the number of destinations in line 1 (e.g. –np- refers to not processed material and defined as a waste destination)
3	Number of Processes	2{tab}-Number of Processes	The first 2 destinations are assumed to be processes
4	Number of Waste Dumps	2{tab}-Number of Waste Dumps	The last 2 destinations are assumed to be waste dumps
5	Number of Stockpiles	0{tab}-Number of Stockpiles	Considered for later extensions
6	Number of Rock Types	7{tab}-Number of Rock Types	Number of rock types in the block model
7	Rock Types	2{tab}3{tab}5{tab}8{tab}101{tab}201{tab}301{tab}-Rock Types	The rock codes should match the number of rock types in line 6
8	Number of Element	3{tab}-Number of Elements	Number of elements in the block model
9	Elements	P{tab}S{tab}MWT{tab}-Elements	The elements should match the number of elements in line 8
10	Number of Elements Processed	1{tab}-Number of Elements Processed	Number of elements processed (not contaminants)
11	Number of Rock Types Processed	3{tab}-Number of Rock Types Processed	Number of rock types processed (mineralized rock types)
12	The origin of Indices	41{tab}68{tab}22{tab}-XI,YI,ZI Origin	Only saved for later references and not used for clustering operations (can be left 0,0,0)
13	The block dimensions	25{tab}25{tab}15{tab}-X,Y,Z Dimension	
14	The origin of coordinates	97525{tab}600200{tab}1440{tab}-X,Y,Z Origin	
15	Number of Blocks	19561{tab}-Number of Blocks	The number of blocks in the block model/The number of unique rows
16	Number of Rows	42117{tab}-Number of Rows	The number of rows in the file excluding the header rows
17	Number of Production Periods	20{tab}-Number of Periods	The number of production periods currently determined using mine planning tools

8.1.2 *The Column Headers*

Line number 18 in the file contains the column headers. The number of columns and their headers changes based on the block model properties such as number of elements. The columns should be formatted in the following order (Table 8.2).

Table 8.2. Block Data Columns

Column #	Header	Type	Description
1	BlockID	Integer	Unique block incremental ID number
2	IX	Integer	Block column index
3	IY	Integer	Block row index
4	IZ	Integer	Block elevation index
5	X	Float	Block X coordinate
6	Y	Float	Block Y coordinate
7	Z	Float	Block Z coordinate
8	PitID	Integer	Pit number (in case there are multiple pits in one model)
9	PhaseID	Integer	Push back number
10	BenchPhaseID	Integer	Bench-phase/panel number
11	numRockTypesInBlock	Integer	Number of rock types in the block
12	RockType	String	The rock type e.g. 101, HYPO
13	RockCode	Integer	The rock type order (defined in the header e.g. 5 for 101)
14	NumDestination	Integer	Number of possible destinations for the block
15	Destination	String	Block destination name e.g. MILL
16	DestinationCode	Integer	Block destination order e.g. 1 for MILL
17	BlockTonnage	Float	Total block tonnage
18	BlockValue	Float	Total block value if sent to this destination
19	Revenue	Float	Total revenue if sent to this destination
20	ProcessingCosts	Float	Total processing cost at this destination
21	MiningCostAndHaulage	Float	Total cost of mining and hauling to this destination
22	BlockReferenceMiningCost	Float	Total cost of mining and hauling to this destination
23	MineralizedTonnage	Float	Total extractable mineralized tonnage from this rock type (considering dilution e.g. 0.95*block tonnage)
24	OreTonnage	Float	Total Ore tonnage (mineralized tonnage above cut-off) from this rock type
25	WasteTonnage	Float	Total waste rock in the block
26	TotalWasteTonnage	Float	Total waste tonnage (waste rock + undefined waste + dilution)
27	ProcessThroughPut	Float	Process throughput at this destination for this block
28	BlockFraction	Float	Fraction of block extracted in this period
29	Period	Integer	Period number

30	{element}_Quantity	Float	Element quantity in the block (tonnage)
31	{element}_Grade	Float	Element grade in the block (%mass)
32	{element}_Recovery	Float	Element recovery at this destination (0-1)
33	{element}_Revenue	Float	Element revenue (in addition to block processing revenue)
34	{element}_ProcessingCost	Float	Element processing cost (in addition to block processing cost)
…	Repeat 30 to 34 for each element		
	DestinationFraction	Float	Fraction of block sent to this destination in this period

8.1.3 *The Block Model*

The block data is listed under the header according the aforementioned columns. Bear in mind that no field is allowed to be left empty. Therefore, you have to use NA and 0 for unavailable string and number fields respectively. The block data is allowed to have multiple rows per block where a block consists of multiple rock types or it can be sent to multiple destinations. It is also allowed to have multiple rows per block when a block is divided between two periods or pushbacks. In this case, the import function will add up tonnages from the two rows. However, make sure that multiple rows describing a single block have to have a common block ID.

The clustering algorithm works based on a single element in most cases and it is easier to have one element imported into Matlab. However, it is possible to define multiple elements, import them into Matlab and try different combinations when using the clustering procedure. Many of the columns introduced earlier are defined to account for MILP formulations and are not needed for clustering. Therefore, they can be left as NA or 0 based on their data type.

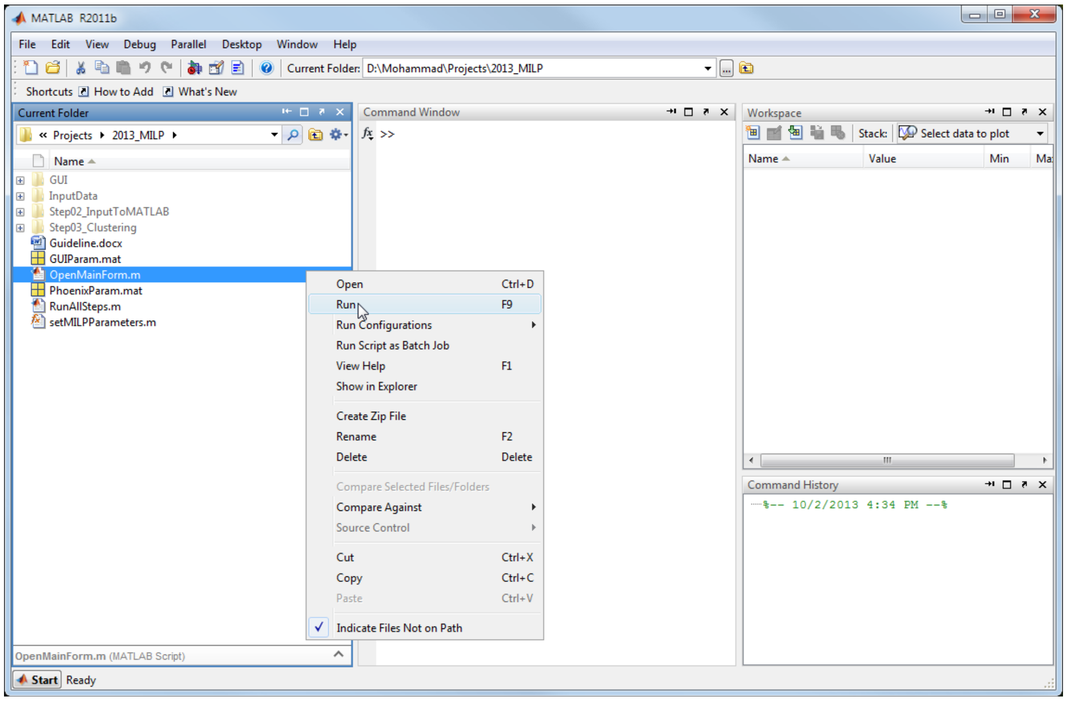
8.2. Step 2: Import to Matlab

The next step is to import the block model into Matlab with the specified format.

1. Open Matlab

2. Change the active directory to "2014_MILP"

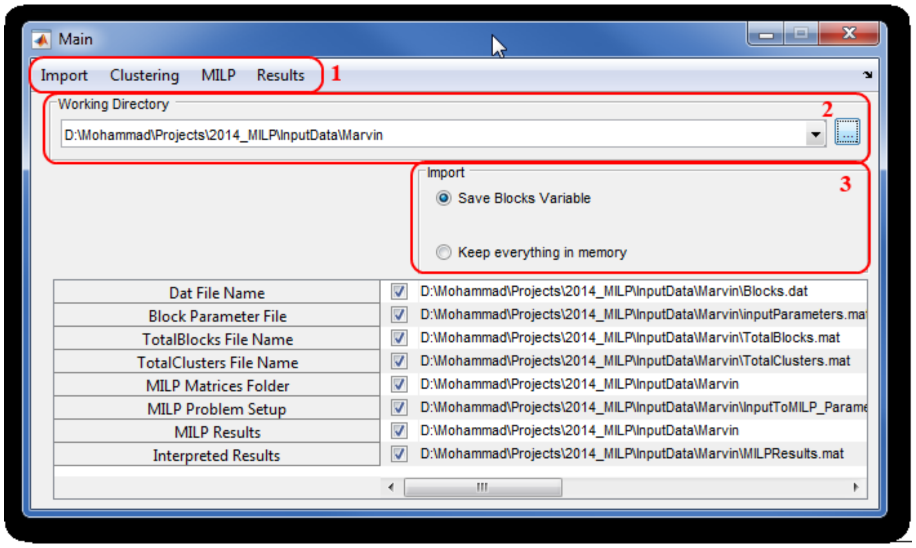
3. Right-click on the script file "OpenMainForm.m" and click on "Run" as in Figure 8.1. It will open the GUI developed for running the clustering algorithm.



```
%-- 10/2/2013 4:34 PM --%
```

Figure 8.1. Opening GUI

The main GUI has 3 sections to be explained separately: the menus, the active directory and memory usage option. The menus are used to navigate through different parts of the application. However, using the application starts by setting the working directory. Results of every step are loaded from and saved into the active directory. There are two options for memory usage: "Save Blocks Variable" and "Keep Everything in memory". The first option takes more time since it loads blocks variable, performs the operation, saves the results and clears memory every time. This helps the application consume less memory. In contrast, the second option keeps the blocks variable in the memory and the user has to save it if needed. There is also a read-only table on the main GUI that checks which files exist in the active directory.



Working Directory: D:\Mohammad\Projects\2014_MILP\InputData\Marvin

Import: [✓ Save Blocks Variable] Keep everything in memory

Dat File Name	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin\Blocks.dat
Block Parameter File	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin\inputParameters.ma
TotalBlocks File Name	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin\TotalBlocks.mat
TotalClusters File Name	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin\TotalClusters.mat
MILP Matrices Folder	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin
MILP Problem Setup	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin\InputToMILP_Parame
MILP Results	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin
Interpreted Results	✓	D:\Mohammad\Projects\2014_MILP\InputData\Marvin\MILPResults.mat

Figure 8.2. Main GUI

The first step is to create a directory and set it as the active directory.

1. Click on the "…" button in section 2 of the GUI

2. Browse to the desired directory (e.g. 2014_MILP/InputData)

3. Press the "New Folder" button and give it a name

Afterwards, you have to import the block file into Matlab. Choose the "Import DAT File" command from the "Import" menu and browse to the tab-separated block model file. It will create a copy of the file in the active directory and import the block data into Matlab structured variable. It will then save the TotalBlocks variable to hard disk or keep it in memory based on the option chosen. The "Load Existing TotalBlocks" and "Save TotalBlocks" menus can be used to manually load and save the blocks variable when the keep in memory option is chosen.

8.3. Step 3: Clustering

Two clustering methods are currently available in the software: horizontal hierarchical and k-means. They can be called from the clustering menu on the upper left corner of the main GUI.

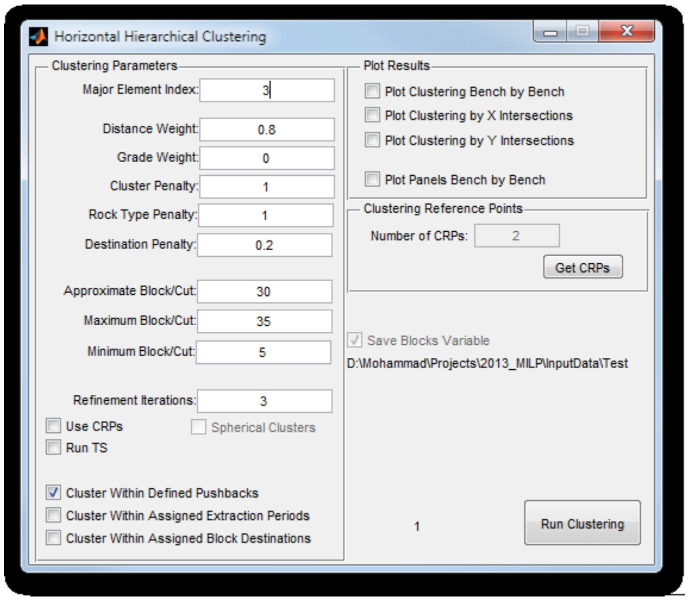
- If the “Save Blocks Variable” option is chosen:

The clustering algorithms automatically load the TotalBlocks variable. However, if the program is not able to find the TotalBlocks.mat file in the active directory the user has to use the “Load Existing TotalBlocks” menu to locate the TotalBlocks variable which will be copied to the active directory then and loaded when the clustering algorithm starts.

- If the “Keep in Memory” option is chosen:

The user has to load the TotalBlocks variable if it is not already loaded in the memory by using “Load Existing TotalBlocks” menu and browsing to the TotalBlocks.mat file.

The two algorithms and their control parameters are explained in the following sections. The clustering GUI (Figure 8.3) contains 5 textboxes to enter the weight and penalty parameters as explained in Appendix I.



Horizontal Hierarchical Clustering

Clustering Parameters

Major Element Index:	3
Distance Weight:	0.8
Grade Weight:	0
Cluster Penalty:	1
Rock Type Penalty:	1
Destination Penalty:	0.2
Approximate Block/Cut:	30
Maximum Block/Cut:	35
Minimum Block/Cut:	5
Refinement Iterations:	3

- [ ] Use CRPs
- [ ] Spherical Clusters
- [ ] Run TS
- [✓] Cluster Within Defined Pushbacks
- [ ] Cluster Within Assigned Extraction Periods
- [ ] Cluster Within Assigned Block Destinations

Plot Results

- [ ] Plot Clustering Bench by Bench
- [ ] Plot Clustering by X Intersections
- [ ] Plot Clustering by Y Intersections
- [ ] Plot Panels Bench by Bench

Clustering Reference Points

Number of CRPs: 2

Get CRPs

- [✓] Save Blocks Variable

D:\Mohammad\Projects\2013_MILP\InputData\Test

1

Run Clustering

Figure 8.3. Hierarchical Clustering GUI

- Major Element Index: Since the data structure is able to handle multiple elements you need to indicate which element grade to use for calculating the grade difference between the blocks. This textbox is considered to input the order index of the element of interest in the input file (3 represents MWT in our example).

- Distance Weight: The weight put on distance and direction measure in calculating the similarities (W_D in equation (3.3), Section 3.3.1)

- Grade Weight: The weight put on the major element grade difference measure in calculating the similarities (W_G in equation (3.3), Section 3.3.1)

- Cluster Penalty: The penalty value for the blocks located above different clusters in calculating the similarities (C_{ij} in equation (3.3), Section 3.3.1)

- Rock Type Penalty: The penalty value for the blocks from different rock types in calculating the similarities (R_{ij} in equation (3.3), Section 3.3.1)

- Destination Penalty: The penalty value for the blocks determined to be sent to different destinations in calculating the similarities (N_{ij} in equation (3.3), Section 3.3.1)

- There are also 3 textboxes considered for average, maximum and minimum blocks per cut (cluster) that control the cluster sizes as explained in Appendix I.

- Refinement Iterations: The number of shape refinement iterations to be performed after the clustering

- Use CRPs: If checked the user has to provide CRPs for directional clustering, otherwise the M_{ij} will be set to 1 (equations (3.14) and (3.17) in Section 3.3.4).

- Spherical Clusters: This checkbox will be enabled if “Use CRPs” is checked. If this is checked, the mining direction factor will be calculated based on equation (3.17) instead of equation (3.14).

- Run TS: The Tabu search (TS) procedure will be called, if this checkbox is checked, in order to reduce the number of arcs between formed clusters in each bench and the ones in the bench below. This helps planning stage get to the bottom of the pit faster. However, experiments of various datasets have shown that reducing the precedence arcs by manipulating the cluster shapes decreases the homogeneity of the clusters and the practicality of the production plans.

- There are 3 checkboxes in the lower left side of the GUI for clustering within boundaries. If the phase (pushback) IDs, production periods or destinations are assigned for the blocks in the input file they can be used to apply strict boundaries on the clustering. This means blocks from different regions cannot be merged to form a cluster.

- There are 4 checkboxes in the “Plot Results” panel that can be used to plot the clustering results after the clustering is performed. They will produce plots of clustering bench by bench, X intersections and Y intersection. However, for a real-size dataset there will be many figures generated and it is not recommended to plot them at the same time in this GUI. Another GUI is designed for plotting the results in plan views and cross sections. The last checkbox “Plot Panels Bench by Bench” plots the panels/bench phases to be able to compare against the results when "Clustering within Defined Pushbacks" is desired.

- The number of CRP points is currently fixed to 2 but the textbox is considered for future expansions on the algorithm. The GUI has a button to add the CRPs to each bench. It starts from the lowest bench and continues to the higher benches. The red squares represent blocks on each bench and the user can use the cross hair to specify the start and end point of the mining direction vector. The same way, the user can specify 2 points for what we call the spherical clustering (check Appendix I for stratified deposit).

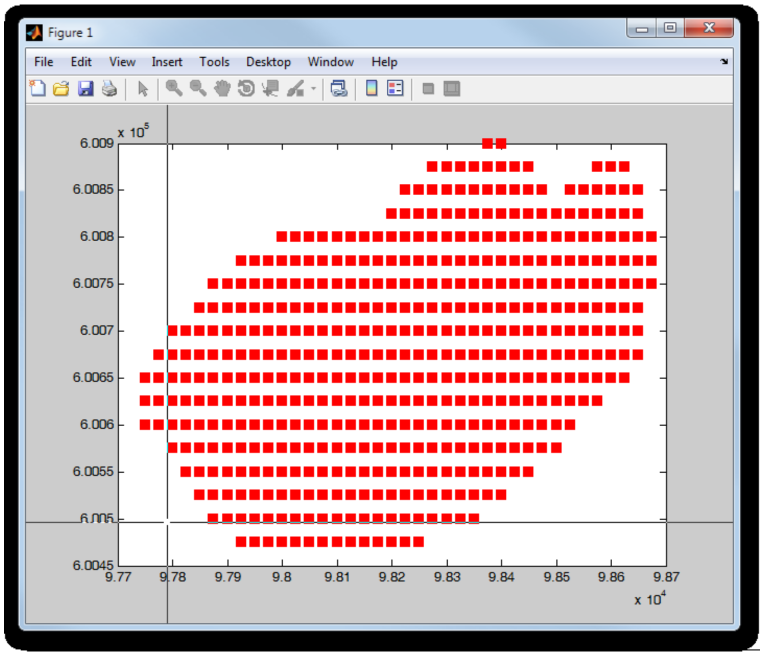
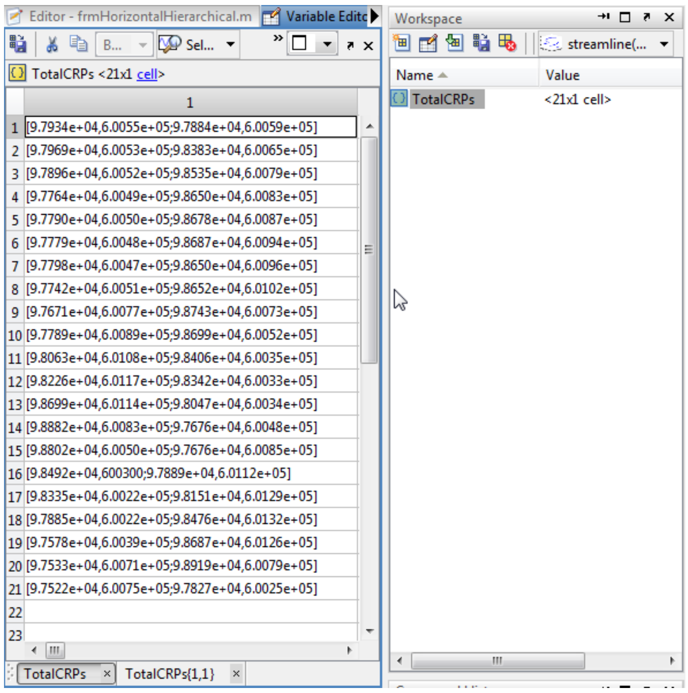


Figure 8.4. Get CRPs Plot

There are instances that the user prefers to enter exact coordinates for CRPs; for example, ramp entrance coordinates can be used as CRPs for spherical clusters. The CRPs are saved in a variable called "TotalCRPs.mat" in the active directory. If the user needs to change the values or input them manually, the file can be opened in Matlab. The TotalCRPs variable is a cell array with a cell per bench starting from the lowest bench. Each cell contains a 2×2 matrix with the coordinates of the 2 reference points as in Figure 8.5 and Figure 8.6.

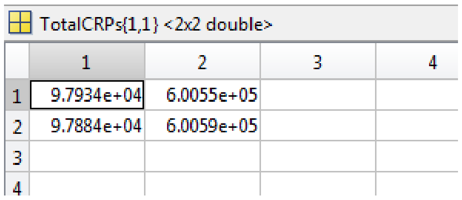


TotalCRPs <21x1 cell>

	1
1	[9.7934e+04,6.0055e+05;9.7884e+04,6.0059e+05]
2	[9.7969e+04,6.0053e+05;9.8383e+04,6.0065e+05]
3	[9.7896e+04,6.0052e+05;9.8535e+04,6.0079e+05]
4	[9.7764e+04,6.0049e+05;9.8650e+04,6.0083e+05]
5	[9.7790e+04,6.0050e+05;9.8678e+04,6.0087e+05]
6	[9.7779e+04,6.0048e+05;9.8687e+04,6.0094e+05]
7	[9.7798e+04,6.0047e+05;9.8650e+04,6.0096e+05]
8	[9.7742e+04,6.0051e+05;9.8652e+04,6.0102e+05]
9	[9.7671e+04,6.0077e+05;9.8743e+04,6.0073e+05]
10	[9.7789e+04,6.0089e+05;9.8699e+04,6.0052e+05]
11	[9.8063e+04,6.0108e+05;9.8406e+04,6.0035e+05]
12	[9.8226e+04,6.0117e+05;9.8342e+04,6.0033e+05]
13	[9.8699e+04,6.0114e+05;9.8047e+04,6.0034e+05]
14	[9.8882e+04,6.0083e+05;9.7676e+04,6.0048e+05]
15	[9.8802e+04,6.0050e+05;9.7676e+04,6.0085e+05]
16	[9.8492e+04,600300;9.7889e+04,6.0112e+05]
17	[9.8335e+04,6.0022e+05;9.8151e+04,6.0129e+05]
18	[9.7885e+04,6.0022e+05;9.8476e+04,6.0132e+05]
19	[9.7578e+04,6.0039e+05;9.8687e+04,6.0126e+05]
20	[9.7533e+04,6.0071e+05;9.8919e+04,6.0079e+05]
21	[9.7522e+04,6.0075e+05;9.7827e+04,6.0025e+05]
22	
23	

Name	Value
TotalCRPs	<21x1 cell>

Figure 8.5. TotalCRPs Variable with 21 Benches



TotalCRPs{1,1} <2x2 double>

	1	2	3	4
1	9.7934e+04	6.0055e+05		
2	9.7884e+04	6.0059e+05		
3				
4				

Figure 8.6. CRP Coordinates for the First Bench (X,Y)

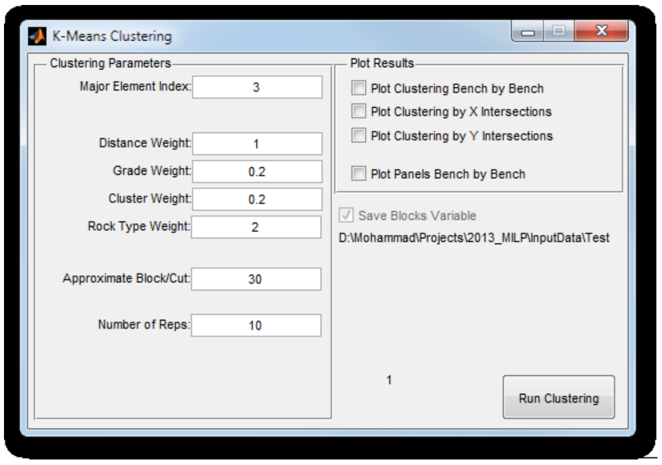
After running the clustering algorithm the user may want to “Save the Clustering Results” if working with “Keep in Memory” option and then use the “Plot and Export” GUI to plot the results or export them into a text file.

8.4. K-Means Clustering

A famous example of partitional algorithms is the k-means, which attempts to find cluster means and assign data points to the closest mean. K-means is a partitioning technique which tries to find a good partitioning scheme by iteratively modifying the partitions. One extension to k-means clustering which is relevant to this project is the kernel k-means which is developed to be able to partition data points which are not linearly separable by mapping them into a kernel space. The clustering technique used is an implementation of the kernel k-means algorithm based on gradient descent search. In this approach, K initial cluster centers are randomly selected at each replication. Then the objects are assigned to the nearest center. Afterwards, based on the gradient descent search technique, the centers are manipulated in such a way that the summation of distances between the objects and the means is locally minimized. Another replication is then started with a new random set of means and the process continues for a limited number of replications.

The first step for this algorithm is to form the feature matrix, which holds all the important properties of all objects. To be consistent with the hierarchical clustering technique, the same parameters are used with the same weighting approach. Then the matrix has to be kernelized in order to get better results. When objects are not linearly separable in their initial space, kernel functions are used to map data points from the initial space to the kernelized space and do the clustering in there. Then the same map is used in returning to the initial space with all the objects labeled as belonging to various clusters. Having tested various kernel functions and parameters, a polynomial kernel function with $d = 1$ is used in this implementation. Afterwards, K initial cluster centers are randomly selected in the kernelized space and objects are assigned to the closest mean. Then the objective function, which is a summation of Euclidean distances between all objects and cluster means, is calculated. Cluster means are then manipulated in an iterative manner based on gradient descent until a local minimum is found. This is stored as a solution to the clustering problem and a new replication starts with another random definition of cluster means. Finally, all of the replications are compared, and the one with the lowest objective function is selected as the solution to the clustering problem on that bench [3]. The problem with the k-means approach is that there is no size control. In addition, it is possible to generate fragmented clusters that are problematic if used in later planning stages.

The k-means algorithm GUI (Figure 8.7) can be called from the GUI, clustering menu. Instead of penalty values, there are only weights in k-means GUI. The X, Y, grade and rock type are put in a matrix and each are powered to their weights. User can also input the average blocks per cut that determines the number of center points in the k-means algorithm. The last parameter is the number of replications. More replications usually results in better results but by talking more time. The rest of the k-means GUI works the same as the horizontal hierarchical GUI.


K-Means Clustering

Clustering Parameters
Major Element Index: 3
Distance Weight: 1
Grade Weight: 0.2
Cluster Weight: 0.2
Rock Type Weight: 2
Approximate Block/Cut: 30
Number of Reps: 10

Plot Results
☐ Plot Clustering Bench by Bench
☐ Plot Clustering by X Intersections
☐ Plot Clustering by Y Intersections
☐ Plot Panels Bench by Bench

☑ Save Blocks Variable
D:\Mohammad\Projects\2013_MILP\InputData\Test

1

Figure 8.7. K-Means Clustering GUI

8.5. Plot and Export

After performing the clustering with hierarchical or k-means algorithms, the user may need to evaluate the clustering results by looking at plan views or cross sections, or export the results to a text file to import into other applications. A GUI, which can be called from the clustering menu, is developed for this purpose (Figure 8.8). The GUI consists of 3 main panels: Clustering Stats, Plot and Export.

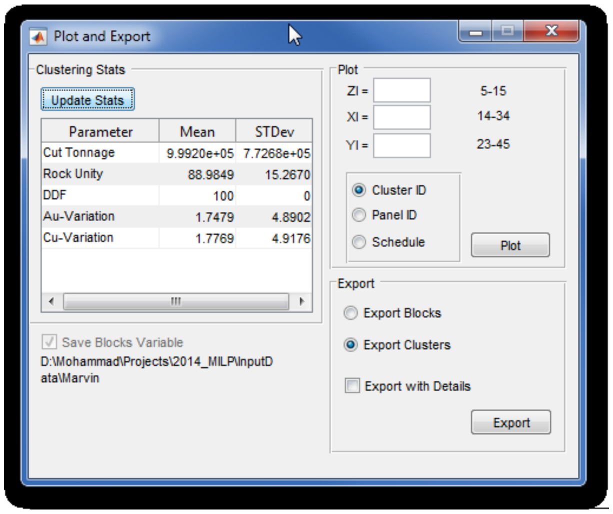
The first panel is designed to provide a summary of the clustering measures to be able to quickly evaluate clustering scheme and change the parameters to get better results. The clustering statistics table is updated by pressing the “Update Stat” button on the upper left corner. Once updated, it will provide the user with the following information:

- Cut Tonnage: Average and the standard deviation of the tonnage of material in the cuts is the first evaluation criterion in the table. This can be used to check if the differences in the tonnage of material in the generated cuts are reasonable.

- Rock Unity: Based on the structure of the problem and existing criteria for categorical variables, a new index is defined as the percentage of rocks in a mining-cut belonging to the most dominant rock type in that mining-cut. This is called the rock unity and is depicted in second row of the table.

- DDF: Destination Dilution Factor is defined in the same way as the rock unity but by considering the predetermined block destination as the homogeneity factor.

- Element Variation: The coefficient of variation (CV) is defined as the standard deviation of a variable divided by its mean. The average and standard deviation of the CV values for each element is presented in the following rows. Average CV can represent the variations in the grade values among the blocks grouped together. This can be helpful when creating mining polygons for estimating the head grade to the processing plant.



Parameter	Mean	STDev
Cut Tonnage	9.9920e+05	7.7268e+05
Rock Unity	88.9849	15.2670
DDF	100	0
Au-Variation	1.7479	4.8902
Cu-Variation	1.7769	4.9176

[✓ Save Blocks Variable]
D:\Mohammad\Projects\2014_MILP\InputData\Marvin

ZI = 5-15
XI = 14-34
YI = 23-45

[✓ Cluster ID] Panel ID Schedule

Export Blocks [✓ Export Clusters]

Export with Details

Figure 8.8. Plot and Export GUI

The “Plot” panel can be used to plot the clustering results in plan views, cross sections and in 3D. The minimum and maximum index values for the blocks are presented in front of each textbox. The user can choose to filter the blocks to be plotted by entering the index value in each textbox. The application will create a 3D plot of the dataset if all textboxes are left empty. Otherwise, a plan view (if ZI is specified) or cross section (if XI or YI is specified) will be created. Note that the user cannot enter a range in the filter textboxes but only integer numbers. There are three options to plot clusters, panels and the schedule from the input file.

The “Export” panel is developed to export the clustering results into tab-delimited text files to be used as input to other software. The “Export Blocks” option creates a file with a row for each block with indices, coordinates, cluster IDs etc. The columns in the text file are presented in Table 8.3. The “Export Clusters” option provides a file with the same format but with one row for each cluster. The indices and coordinates presented for clusters are the average values of the blocks in that cluster.

Table 8.3. Blocks Brief Output File

Column Number	1	2	3	4	5	6	7	8	9	10	11
Value	XI	YI	ZI	X	Y	Z	Panel ID	Cluster ID	Cluster ID in Bench	Period	Destination

There is also a checkbox labeled "Export with Details" that adds extra columns to the original format and can be mostly useful for cluster exports. This option can be used to get detailed information about the quality of the generated clusters such as their rock unity, destination dilution factor and element grade variations. It will also provide the precedence arcs between the generated clusters in different benches. This can be especially useful when evaluating the Tabu Search results. The extra columns are presented in Table 8.4. After column 20, there are 2 columns for each element in the dataset: mean and variance of grades.

Table 8.4. Block Detailed Output File

Column Number	1	…	13	14	15	16
Value	ID	…	Phase ID	Rock Unity	Destination Dilution Factor	Total Tonnage
Column Number	17	18	19	20	…	
Value	Downward Relation Count	Upward Relation Count	ID in Bench	Bench ID	Average Grade	Grade Variance

8.6. Prepare Input to MILP

As explained in section 7.5.1 of this appendix, we have to prepare the matrices for the MILP formulation based on four different resolutions: cluster-panel, cluster-cluster (CC), block-cluster (BC) and block-block (BB). This can be done through the menu items under MILP menu as shown in Figure 8.9.

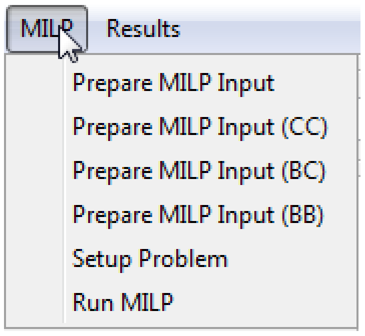


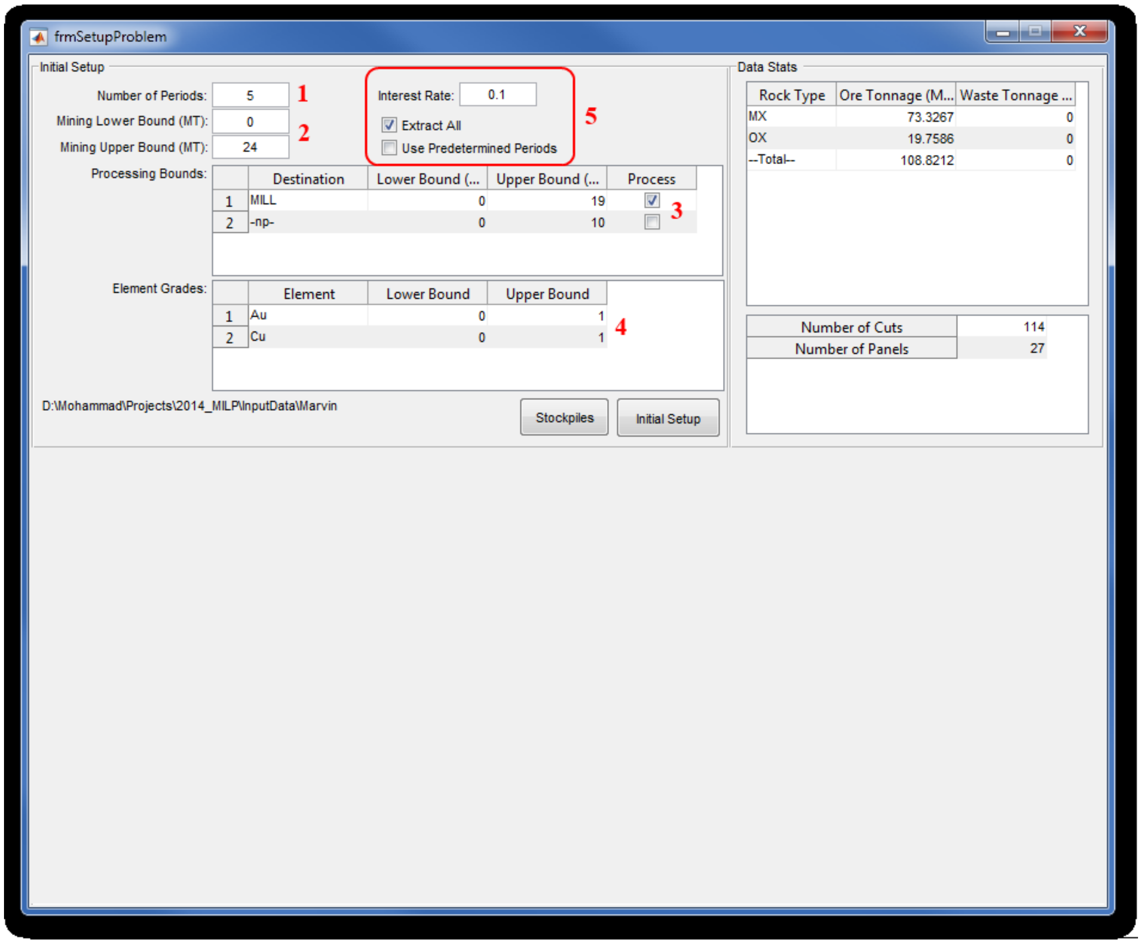
Figure 8.9. MILP Menu

8.7. Setup MILP Parameters

We have developed a GUI for setting up the MILP by changing the parameters in two user-friendly forms. The first form is called the setup form and includes any parameters related to the MILP except than the stockpiles. The stockpile settings are in a separate form that can be opened from the main setup form. The setup form can be opened by the "Setup Problem" item under the MILP menu.

8.7.1 *Initial Setup*

The first step for setting up the MILP parameters is to define the number of periods and the default values for mining and processing constraints. The initial setup panel on the top-left corner of the form (Figure 8.10) is where the initial settings have to be defined. We start by defining the number of periods in the textbox marked with number 1 in Figure 8.10. Two textboxes, market with number 2 in Figure 8.10, are designed to input the default values for lower and upper limits on the mining capacity. The table marked with number 3 in Figure 8.10 presents the list of destinations defined in the input file. We can check the Process checkbox for processing destinations (the ones that have a limited capacity) and assign lower and upper bounds for them. Note that the lower and upper bound values are in millions. The table marked with number 4 in Figure 8.10 presents the elements in the block model and provides the option to assign default lower and upper values for average grade constraints for each element. Capacities and average grade bounds can be edited for each period in the next step. The interest rate is a fixed number for the mine life and can be provided in the textbox market with number 5. If the checkbox “Extract All” is checked the model will be forced to extract everything within the final pit instead of determining what to mine and what to leave in ground based on NPV. The checkbox “Use Predetermined Periods” will limit the extraction variables to predetermined periods for each panel and cut in cases where a multi-step solution is required. This will be explained in more details later. The “Data Stats” panel on the top-right corner is the summary of the block model to be used for determining the proper capacities.



Number of Periods: 5
Mining Lower Bound (MT): 0
Mining Upper Bound (MT): 24
Interest Rate: 0.1
[✓] Extract All
[ ] Use Predetermined Periods

Processing Bounds:

	Destination	Lower Bound (...	Upper Bound (...	Process
1	MILL	0	19	✓
2	-np-	0	10	

Element Grades:

	Element	Lower Bound	Upper Bound
1	Au	0	1
2	Cu	0	1

D:\Mohammad\Projects\2014_MILP\InputData\Marvin

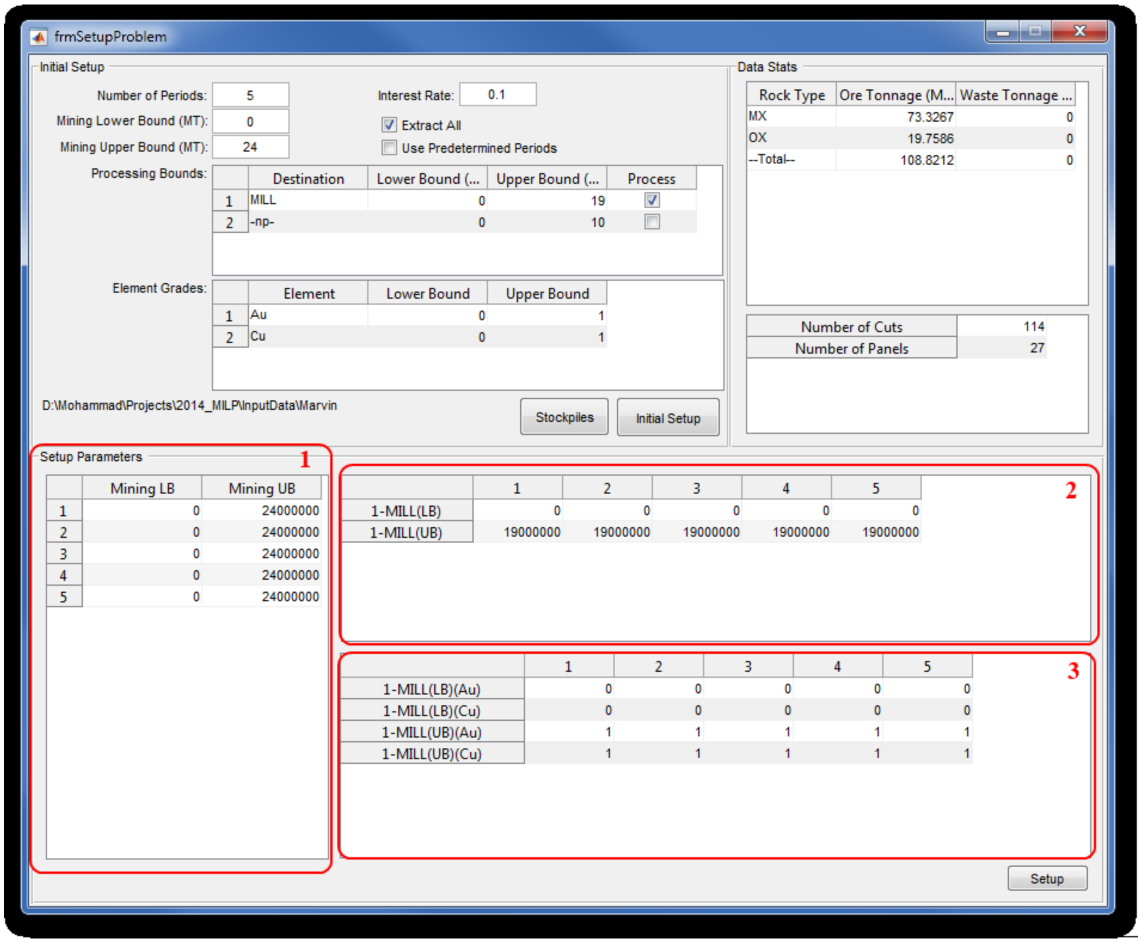
Rock Type	Ore Tonnage (M...	Waste Tonnage ...
MX	73.3267	0
OX	19.7586	0
--Total--	108.8212	0

Number of Cuts	114
Number of Panels	27

Figure 8.10. Initial Setup Panel

8.7.2 *Period by Period Setup*

Pressing the “Initial Setup” button will replicate the default values for lower and upper limits for the number of periods and make the corresponding tables visible. Now we can change the constraints for different periods in the three tables marked with 1 to 3 in Figure 8.11. The numbers in these three tables are not in millions in contrast to the default values.



Number of Periods: 5
Mining Lower Bound (MT): 0
Mining Upper Bound (MT): 24
Interest Rate: 0.1
[✓] Extract All
[ ] Use Predetermined Periods

Processing Bounds:

	Destination	Lower Bound (...	Upper Bound (...	Process
1	MILL	0	19	✓
2	-np-	0	10	☐

Element Grades:

	Element	Lower Bound	Upper Bound
1	Au	0	1
2	Cu	0	1

D:\Mohammad\Projects\2014_MILP\InputData\Marvin

Rock Type	Ore Tonnage (M...	Waste Tonnage ...
MX	73.3267	0
OX	19.7586	0
--Total--	108.8212	0

Number of Cuts	114
Number of Panels	27

	Mining LB	Mining UB
1	0	24000000
2	0	24000000
3	0	24000000
4	0	24000000
5	0	24000000

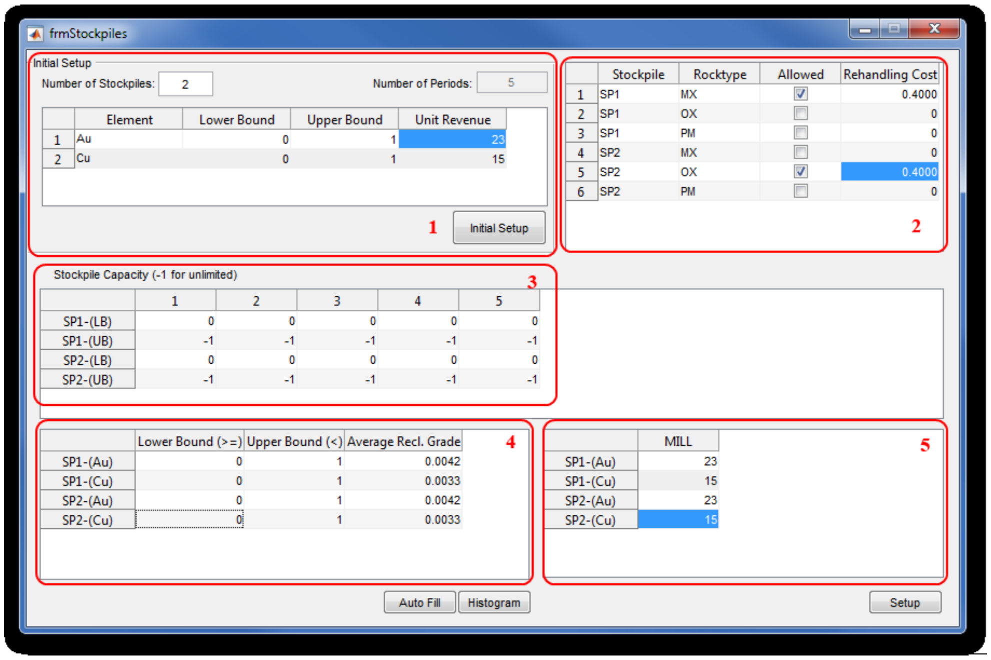
	1	2	3	4	5
1-MILL(LB)	0	0	0	0	0
1-MILL(UB)	19000000	19000000	19000000	19000000	19000000

	1	2	3	4	5
1-MILL(LB)(Au)	0	0	0	0	0
1-MILL(LB)(Cu)	0	0	0	0	0
1-MILL(UB)(Au)	1	1	1	1	1
1-MILL(UB)(Cu)	1	1	1	1	1

Figure 8.11. Period by Period Setup

8.7.3 *Stockpile Setup*

Following the same design as the main setup form, the stockpile setup requires an initial setup with the number of stockpiles and default values for lower and upper bounds on average grades as well as the reclamation revenues (Figure 8.11). After setting the number of stockpiles and pressing the initial setup, we can limit stockpiles to rock types and assign rehandling cost per ton of material reclaimed in the top-right table market with number 2. Moreover, we can limit the tonnage sent to the stockpile in each period through the table marked with number 3. The lower and upper bounds on the average grade of material sent to the stockpile and the reclamation grade for each stockpile and element can be set in table 4. The revenue made from processing elements in the stockpile can be set in table 5.



Initial Setup

Number of Stockpiles: 2

Number of Periods: 5

	Element	Lower Bound	Upper Bound	Unit Revenue
1	Au	0	1	23
2	Cu	0	1	15

	Stockpile	Rocktype	Allowed	Rehandling Cost
1	SP1	MX	✓	0.4000
2	SP1	OX	☐	0
3	SP1	PM	☐	0
4	SP2	MX	☐	0
5	SP2	OX	✓	0.4000
6	SP2	PM	☐	0

Stockpile Capacity (-1 for unlimited)

	1	2	3	4	5
SP1-(LB)	0	0	0	0	0
SP1-(UB)	-1	-1	-1	-1	-1
SP2-(LB)	0	0	0	0	0
SP2-(UB)	-1	-1	-1	-1	-1

	Lower Bound (>=)	Upper Bound (<)	Average Recl. Grade
SP1-(Au)	0	1	0.0042
SP1-(Cu)	0	1	0.0033
SP2-(Au)	0	1	0.0042
SP2-(Cu)	0	1	0.0033

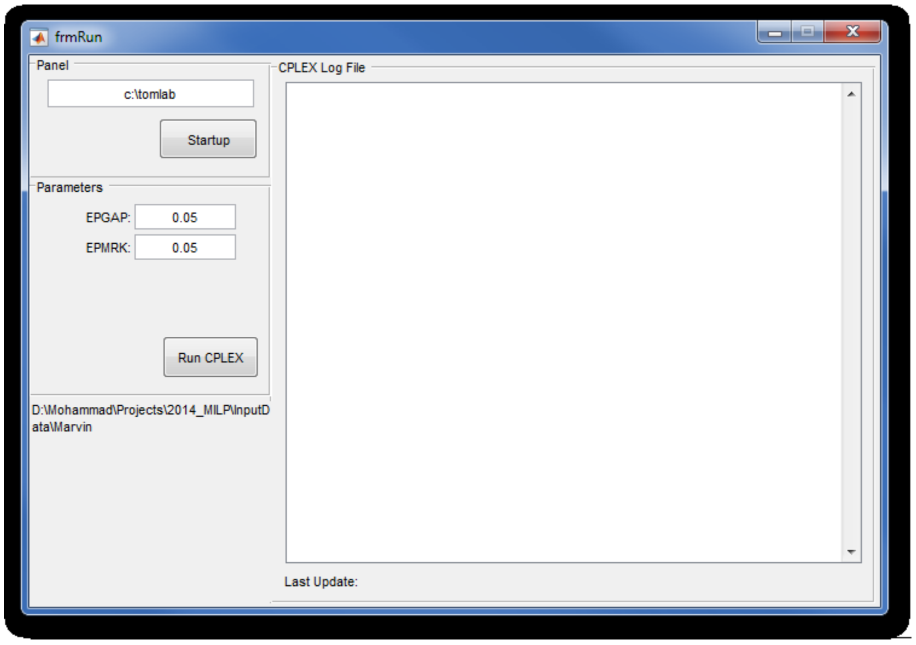
	MILL
SP1-(Au)	23
SP1-(Cu)	15
SP2-(Au)	23
SP2-(Cu)	15

Figure 8.12. Stockpile Initial Setup

We have added two buttons to help find out the proper reclamation grades for stockpiles. The first button is the “Auto Fill” that will calculate the average grade of element by filtering clusters based on the lower and upper bounds provided. The “Histogram” button plots the histograms of the grades based on the provided lower and upper bounds.

8.8. Run MILP Solver

After setting up the MILP parameters, we run the solver to the determined gaps. We developed a GUI for starting up Tomlab, setting the gaps and calling the solver. The GUI is presented in Figure 8.13.



c:\tomlab

EPGAP: 0.05

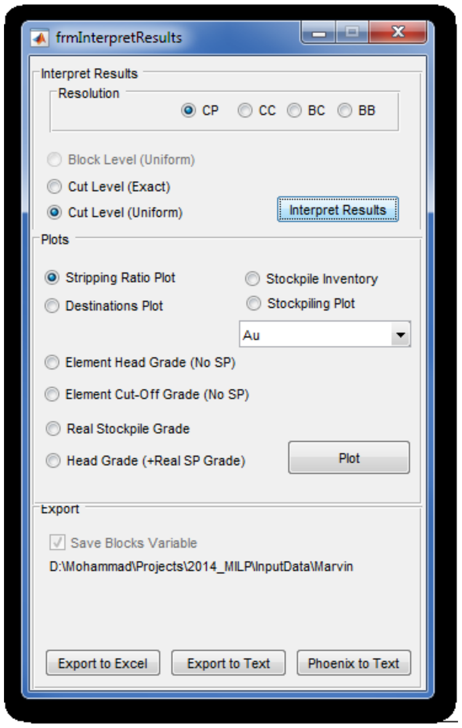
EPMRK: 0.05

D:\Mohammad\Projects\2014_MILP\InputData\Marvin

Figure 8.13. Run MILP

8.9. Interpret Results

The next step, after solving the MILP, is to interpret the results as explained in section 7.5.7. For this step, we have developed a GUI that provides various options for the user to interpret, plot and export the results in different formats. There are four options for the different resolutions that were used in creating the MILP matrices as well as two options for updating cluster schedules based on panel schedules. Next, we have the plot functions available for initial evaluation of the output. We usually use the saved variables from Table 7.14 to create the plots with better looks in Excel. The first four plots are stripping ratio, destination (production), stockpiling (rehandling) and stockpile inventory. For the rest of the plots, the user has to choose an element from the drop-down list before calling the plot function to plot the head grade, cut-off grade (lowest grade of each element sent to process in each period), real stockpile grade and head grade with stockpile.



Resolution: [✓ CP] CC BC BB

Block Level (Uniform)
Cut Level (Exact)
[✓ Cut Level (Uniform)]

[✓ Stripping Ratio Plot]
Destinations Plot
Stockpile Inventory
Stockpiling Plot
Au
Element Head Grade (No SP)
Element Cut-Off Grade (No SP)
Real Stockpile Grade
Head Grade (+Real SP Grade)

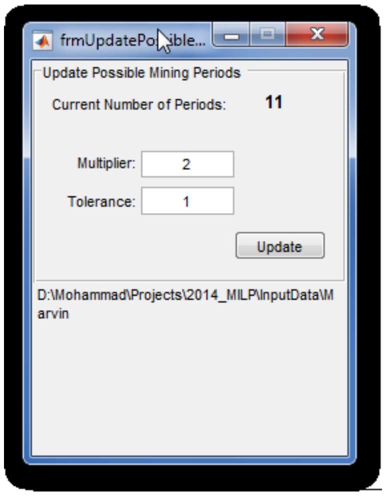
[✓ Save Blocks Variable]
D:\Mohammad\Projects\2014_MILP\InputData\Marvin

Figure 8.14. Interpret Result GUI

The export to Excel button exports the matrices into an Excel file where the Export to Text feature creates a text file from all the blocks with their assigned extraction period and destination. The output file is in the same format as in Table 8.4.

8.10. Update Possible Periods

As mentioned we sometimes solve the MILP model and use the solution to limit the decision variables and resolve the model. For example, if the mine life is 15 years, we can initially solve the model for 5 periods of three years. Then, by using a multiplier of 3 and a tolerance of 1, a panel that is initially scheduled to be extracted in period 2 will have extraction variables for years 3 to 7. We call these the possible periods of that panel and save it as an attribute for the each panel. We have developed a simple GUI to update the possible periods as shown in Figure 8.15. After updating the possible periods, the user has to go back to the MILP setup, set the number of periods to 15 and check the "Use Possible Periods" checkbox to use this information to solve the model in shorter time.



Current Number of Periods: 11

Multiplier: 2

Tolerance: 1

D:\Mohammad\Projects\2014_MILP\InputData\Marvin

Figure 8.15. Update Possible Periods