# Open-Pit Mine Production Scheduling and Crusher Location-Relocation Plan under Semi-Mobile IPCC Systems

by

Dingbang Liu

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

Master of Science

in

Mining Engineering

Department of Civil and Environmental Engineering
University of Alberta

# ABSTRACT

In most open-pit mines, material haulage costs can comprise over 50% of the total mining costs. This portion can even increase as the future mining conditions will become more challenging. Although the conventional truck-and-shovel system is adopted by over 95% of existing surface mines, in-pit crushing and conveying (IPCC) systems receive more attention to the modern mining industry due to their lower operating costs and carbon footprint than conventional truck-and-shovel systems. However, the IPCC system's implementation can introduce substantial upfront investment and reduce mining flexibility; thus, careful mine planning and design are required before the systems' application.

This study considers a situation that the conveyor is fixed in one pit side throughout the mine life, and it can be extended to deeper levels when the mining operation goes deeper. This configuration introduces additional mining direction requirements: mining starts from the conveyor side. It then expands to the other side of the level, and the mining activities below the conveyor line should be avoided. A set of candidate high angle conveyor layouts is generated along the final pit wall, and the situation for a conventional low-angle conveyor with ramp slots is also considered. Each conveyor scheme is considered a scenario for later calculation.

The study aims to develop, implement, and verify a theoretical optimization framework to maximize the economic return measured by NPV while considering the total transportation costs under the application of semi-mobile IPCC systems. In this sense, three mathematical models are proposed to solve the problem from different perspectives. The first model is a two-step linear programming (LP) model for the conventional conveyor installed in the ramp slot. Its first step is a MILP formulation that aims to maximize the NPV; the generated scheduling results

are fed to the second step, a facility location problem, to minimize the total transportation costs. The second model is a two-step model similar to the first one in which instead of the conventional conveyor and ramp slot, a high angle conveyor (HAC) is considered. The third model is a binary-integer linear programming (BILP) formulation for the HAC case, which can make the production scheduling and crusher location-relocation decisions simultaneously. All the presented mathematical models are run at a cluster level to reduce the computation time. A hierarchical clustering approach is applied to aggregate blocks into larger mining units, and their precedence relationships are determined.

The main scientific contributions of this research on the body of knowledge are: (i) introducing a new production scheduling optimization strategy under semi-mobile IPCC systems by incorporating the mining direction requirements from the conveyor's perspective, (ii) developing a conveyor location optimization framework by generating various conveyor lines along the final pit wall and comparing the NPV under each location scenario, (iii) proposing mathematical models for making the production scheduling and crusher location-relocation decision to maximize the NPV while considering the transportation costs, with respect to operational and technical constraints.

# PREFACE

This thesis is an original work by Dingbang Liu. Part of this work is published as Liu D, Pourrahimian Y. A Framework for Open-Pit Mine Production Scheduling under Semi-Mobile In-Pit Crushing and Conveying Systems with the High-Angle Conveyor. *Mining*. 2021; 1(1):59-79. Dingbang Liu has been responsible for model development, computer programming, writing, and editing of this paper. Dr. Yashar Pourrahimian is the supervisory author and was involved with the guidance of concept formation and manuscript composition

This Thesis is Proudly Dedicated to:

*My parents:*

*Shaojun* and *Guanghua,*

*Who always give me unconditional support*

*My partner:*

*Yunlong,*

*Who was happier than me when I finished this work*

# ACKNOWLEDGMENT

First of all, I would like to sincerely thank my supervisor, Dr. Yashar Pourrahimian, for providing academic support, funding, thoughtful ideas and encouragement throughout my graduate program and this research. I admire your conscientious attitude, extensive knowledge, and your ability to illustrate a complex concept in an accessible way. I am very grateful for your careful revision and valuable suggestion during my writing process of this thesis.

I would like to thank all the professors I met and attended their lectures at the University of Alberta, especially Dr. Hooman Askari-Nasab, Dr. Wei Victor Liu and Dr. Derek Apel. I could not finish the thesis without your wonderful lessons and the knowledge I have learned from you. I am grateful for all my examination committee numbers, including Dr. Juliana Leung, for your time to review my thesis and your valuable advice.

I would like to express my appreciation to my friends at the Mining Optimization Lab at the University of Alberta: Magreth, Dismas, Roberto, Nasib, Alireza, Agar, and Soroush. Spacial thanks to Roberto for your great help in my research. I always want to thank my good friends during this program here: Daniel U., Tony T., Kelly R., Arman, Youwei, Jiaxin, Yuran, Duowei, Papinder, Xinfang, Di Y, Jun W, Shaosen M. I would not have enjoyed my time here without you.

Finally but most importantly, I would like to thank my family: my parents Shaojun and Guanghua, my uncle Guanquan, my grandma Xiurong, for their understanding, encouragement, and love. My partner and soulmate Yunlong D. has been my source of power throughout my program.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| BEV | Block Economic Value |
| BILP | Binary Integer Linear Programming |
| BIP | Binary Integer Programming |
| CLEV | Cluster Economic Value |
| DCF | Discounted Cash Flow |
| EPGAP | Relative MILP Gap Tolerance |
| FLP | Facility Location Problem |
| HAC | High Angle Conveyor |
| ILP | Integer Linear Programming |
| IPCC | In-pit crushing and conveying |
| LP | Linear Programming |
| LTOPP | Long-Term Open Pit Production Plan |
| MILP | Mixed-Integer Linear Programming |
| NPV | Net Present Value |
| RT | Rock Type |
| SCW | Straight Conveyor Wall |
| SMIPCC | Semi-mobile IPCC |
| TS | Truck and Shovel |
| UPL | Ultimate Pit Limit |

# LIST OF NOMENCLATURE

**Sets**

| | |
|---|---|
| $\mathbb{N}$ | Set of clusters in the model |
| $\mathbb{B}_i^w$ | Set of waste blocks cluster *i* |
| $\mathbb{B}_i^o$ | Set of ore blocks cluster *i* |
| $\mathbb{P}_i$ | For each cluster *i*, there is a set of immediate predecessors that must be extracted before extraction of cluster *i* |
| $\mathbb{L}_j$ | Set of all clusters in level *j* |

**Indices**

| | |
|---|---|
| $i \in \{1,...,I\}$ | Index for clusters |
| $j \in \{1,...,J\}$ | Index for pit levels |
| $t \in \{1,...,T\}$ | Index for scheduling periods |

**Parameters**

| | |
|---|---|
| $I$ | Total number of clusters |
| $J$ | Total number of levels |
| $T$ | Number of scheduling periods |
| $r$ | Discount rate |
| $CLEV_i$ | The undiscounted economic value of cluster *i* |
| $Ton_b$ | The tonnage of block *b* |
| $Ton_i^w$ | The total tonnage of waste in cluster *i* |
| $Ton_i^o$ | The total tonnage of ore material in cluster *i* |
| $Ton_j^s$ | The extra tonnage of extracting ramp slot in level *j* (as waste) |
| $g_b$ | Grade of ore block *b* |
| $g_i^o$ | The average grade of ore material in cluster *i* |

| | |
|---|---|
| $Pc_t$ | Price per unit of product sold in period *t* |
| $s$ | Selling cost per unit of product |
| $m^w$ | Cost of mining a tonne of waste |
| $m^o$ | Cost of mining a tonne of ore |
| $m^s$ | Cost of mining a tonne of waste in the ramp slot (including transportation costs) |
| $c^o$ | Cost of processing a tonne of ore |
| R | The recovery rate for ore material |
| $f_{ijt}$ | Discounted transportation cost for cluster i sent to crusher *j* in period *t* |
| $F_{ij}$ | Undiscounted transportation cost for a unit weight of material in cluster *i* sent to crusher *j* in period *t* |
| $fc_{jt}$ | Discounted transportation cost for all materials sent to crusher *j* in period *t* |
| $X_{i,t}$ | Vector denotes the portion of cluster *i* mined in period *t* (the result of decision variable $x_{i,t}$ generated in scheduling MILP model) |
| $S_{j,t}$ | Vector denotes if the slot slice in level *j* is mined in period *t* (the result of decision variable $s_{i,t}$ generated in scheduling MILP model) |
| $c_t$ | Discounted crusher relocation cost at period *t* |
| $n$ | The minimum period interval for crusher relocation |
| $DH_i$ | The horizontal distance from the centroid of cluster *i* to the crusher station *j* |
| $DV_{ij}^T$ | The vertical distance from cluster *i* to the crusher station *j* (truck hauling) |
| $DV_j^C$ | The vertical distance from crusher station *j* to the pit exit (conveying) |
| $CT_H$ | Unit truck horizontal hauling cost per tonne per meter |
| $CT_V$ | Unit truck vertical hauling cost per tonne per meter |
| $CC$ | Unit conveyor vertical lifting cost per tonne per meter |
| $\overline{M}^t$ | Upper bound of mining capacity in period *t* |
| $\underline{M}^t$ | Lower bound of mining capacity in period *t* |
| $\overline{P}^t$ | Upper bound of processing capacity in period *t* |

| | |
|---|---|
| $\underline{P}^t$ | Lower bound of processing capacity in period $t$ |
| $\overline{G}^t$ | Upper bound on allowable average grade of processed ore in period $t$ |
| $\underline{G}^t$ | Lower bound on allowable average grade of processed ore in period $t$ |

**Decision Variables**

| | |
|---|---|
| $x_{i,t} \in \{0,1\}$ | Binary variable denoting the if cluster $i$ mined in period $t$ |
| $x_{i,t}^c \in [0,1]$ | Continuous variable denoting the portion of cluster $i$ mined in period $t$ |
| $x'_{i,t} \in \{0,1\}$ | Binary variables equal to 1 if the precedent clusters are all cleared for cluster $i$; 0 otherwise |
| $x''_{ijt} \in \{0,1\}$ | Binary variables denoting if cluster $i$ is crushed at level $j$ in period $t$ |
| $y_{j,t} \in \{0,1\}$ | Binary variables equal to 1 if the crusher is in level $j$ in period $t$; 0 otherwise |
| $z_{j,t} \in \{0,1\}$ | Binary variables equal to 1 if the crusher is relocated in level $j$ in period $t$; 0 otherwise |

# CHAPTER 1

# INTRODUCTION

## 1.1 Background

Open-pit mining is the most common surface mining technique for extracting minerals from shallow reserves by excavating a massive hole (Hustrulid et al. 2013). The pit wall typically expands and deepens until either the mineral resource is exhausted or the ratio of waste to ore (stripping ratio) becomes too high to make a profit. The pit shell's shape is similar to a converted cone to maintain a safe slope angle and prevent rock falls. Soil and rock overlying the mineral deposit, known as overburden, must be removed to expose the orebody. This stripping process contributes to massive extraction materials. Some large-scale copper mines can transport up to one million tons of overburden and minerals each day (Mero 2017). As a result, material handling is the most expensive part of mine development, accounting for over 50% of capital and operating costs. Different transportation systems can make the difference between a sustainably profitable mine in a competitive market environment and one that is struggling to meet its marginal costs.

Today's mining industry is becoming more competitive due to resource depletion, climate change, and global low commodity prices. As large-scale, high-grade, and sallow deposits have been exhausting, the future mining conditions are challenging. Open-pit mines will become deeper and broader, with a higher stripping ratio and lower mineral grade. Excavated materials should overcome longer distances and more elevation differences to exit the pit than current operation; correspondingly, more trucks are required in the hauling fleet, contributing to higher fuel consumption and operating costs. The number of operation and maintenance staff also increases during the operation. Moreover, other challenges, including low commodity prices, greenhouse gas emissions, mining companies face pressure to develop more efficient and less energy-consuming operations in the mineral extraction processes.

Mining trucks are widely used in all types of surface mines at a considerable economic scale. The truck-and-shovel (TS) system is adopted by over 95% of existing surface mines (Czaplicki 2008). With geological and market price uncertainty, the TS system provides significant flexibility for deposit exploitation by simultaneously switching mining zones. TS system can also adjust the production rate efficiently due to its discrete nature. When the pit scale is small at the early stage of mine life, the TS system is highly efficient. However, when the pit becomes deeper and broader, the truck-and-shovel system's costs rise exponentially. The pit expansion can result in a higher stripping ratio and waste level growth. Also, the hauling distance and the elevation difference from the loading point to the destination increase sharply. The additional costs include more trucks needed in the fleet, more fuel consumption, tires and labour expenses, and more extended hauling road construction and maintenance (Abbaspour and Maghaminik 2016, Demirel and Gölbaşı 2016). Furthermore, truck hauling is also inefficient in terms of cycle time. About a half of truck travelling time is on the way of returning, wasting both operating time and fuel to move the hundreds of tons of the empty truck.

The in-pit crushing and conveying (IPCC) system offers another material handling operation than the pure-truck system, and it has attracted more attention in the recent mining industry. The IPCC system is a continuous transportation method composed of a series of feeding, crushing, conveying, and discharging modules (Osanloo and Paricheh 2019). According to the crusher station's movability, the IPCC system can be categorized into fully mobile, semi-mobile, and fixed systems (Darling 2011). The truck haulage is partially or fully replaced by the belt conveying based on the system's category. As conveying has much lower operating costs and more energy-efficient than trucking, especially in deep and large-scale open-pit mines (Czaplicki 2008), the IPCC systems can provide significant cost savings.

## 1.2 Statement of Problem

The proposed research mainly focuses on the long-term open-pit production scheduling problem (LTOPP) with a semi-mobile IPCC system. LTOPP is a large-scale optimization problem that aims to optimize the block extraction sequence that produces the maximum possible net present value (NPV), subject to a set of technical and economic constraints (Osanloo et al. 2008). For the purpose of scheduling extraction sequence in open-pit mines, the mine deposit is divided into numerous small and equal cubes called blocks, each of which is

assigned estimated attributes such as rock type, mineral grades and density (Chicoisne et al. 2012). The set of all blocks that comprises a mineral deposit is known as a block model. The extraction profit for each block in the model is calculated based on the block's attributes. In the NPV estimation, this profit is reduced by the discount factor of its corresponding extraction period, reflecting the time value of money. Under the optimal production schedule, the accessible high-grade ore is extracted as soon as possible, maximizing the cash flow in the early mine life to avoid a higher discount factor.

Most current mine planning research assumes using the TS system as the transportation method. The trucking fleet provides considerable flexibility for selective mining, as each truck is discrete and can be dispatched efficiently. The active mining zones can switch between different levels in a short time. Compared to the TS transportation method, the IPCC system introduces additional mining sequence and pit expansion constraints. The active mining faces should be close to the crushing station that cannot be relocated frequently to reduce the trucking portion. Also, the main in-pit conveyor belts are typically accommodated in stationary structures without subsequent transfer (Dryzhenko et al. 2016, Dryzhenko et al. 2017); therefore, the open-pit mine should be developed toward the opposite edge of the conveyor wall, and the mining activities below the conveyor system are prohibited. Moreover, the capacity of the IPCC system is generally constant once the conveyor is installed. This variant of LTOPP under IPCC system is more complex than basic LTOPP. This is mainly due to the additional extraction sequencing and pit expansion restriction that are not required in traditional TS systems.

Based on the equipment mobility, IPCC systems can be categorized into three types: fully-mobile, semi-mobile, and fixed systems (Darling 2011). The semi-mobile IPCC systems are the most popular category, as they can be easily transited from the truck-and-shovel system. The crusher's relocation nature and the relatively lower initial investment make these types of IPCC systems more appealing to be adopted in modern mining activities (Ritter 2016). Under the semi-mobile IPCC system, the material is initially transported by a fleet of trucks from the loading point to the crushing station located some point inside the pit, and the crushed material is then transferred through a conveying system out of the pit. The relocation frequency of the crushing station is typically 2-5 years, and the conveyor belts are usually anchored to the pit wall and mounted on concrete structures or in a pit slot where relocation of this system is rare (Ritter 2016).

In this sense, two research problems arise based on semi-mobile IPCC systems' features: (i) the production scheduling plan that gives the maximum NPV with additional mining sequence and pit expansion restrictions, and (ii) the crusher location-relocation plan that minimizes the material handling and crusher relocation costs. This study focuses on the production scheduling and crusher location optimization under semi-mobile IPCC systems. Moreover, because the different conveyor layouts can result in various possible extraction schedules, a series of candidate conveyor locations are investigated and compared.

In this research, the LTOPP and crusher location-relocation problem is studied based on the following assumptions:

- The conveyor system, either the high-angle conveyor (HAC) or conventional conveyor with a slot, is anchored along one side of the pit wall throughout the mine life. However, the conveyor can be extended to a deeper level as the mining operation move forward by installing another conveyor flight.
- Mining starts from the conveyor side and then expands to the other side of the pit. Material within the pit limit and close to the conveyor line should be mined first.
- The HAC conveyor can be implemented directly along the final pit wall, while the conventional low-angle conveyor requires a dedicated ramp slot on the pit wall to flatten the slope.
- The crushing station is installed along the conveyor line. It can be relocated to different levels while still along the conveyor line.
- All the materials, including ore and waste, are transported through the semi-mobile IPCC system. It can be realized through parallel conveyor belts that transfer different types of material respectively.

## 1.3 Summary of Literature Review

Many studies in the IPCC systems optimization have been focused on crusher location and relocation plans. Several techniques, including simulations, trial and error, facility location problem, mixed-integer programming model, transportation problem and heuristics, have been

used to solve the crusher location problem in previous works. The main object lies in minimizing the total transportation costs.

Some earlier works adopted the discrete event simulation (DES) method to minimize the truck cycle time or the truck's capacity with respect to the crusher location (Sturgul 1987) (Peng and Zhang 1988). More recently, Konak et al. (2007) established a trial-and-error process to enumerated different possible crusher locations on a level basis. The level gives the minimum average haulage distance as the optimum location. Some recent works applied facility location problems (FLPs) and their variant to find the optimum crusher locations in the long-term plan horizon. Rahmanpour et al. (2013) considered each of the possible crusher locations as a hub node. They solved the problem by an integer-programming model to minimize the truck haulage cost and crusher relocation cost. Paricheh et al. (2017) incorporated the time factor and solved the problem using a dynamic facility location approach. Paricheh et al. (2018) developed a heuristic framework based on the dynamic location problem to solve the transition time from a pure-truck system to an IPCC system. They combined the two integer linear programming models based on a heuristic approach to obtain the IPCC application's optimum timing and corresponding crusher locations. The problem optimizes NPV in consideration of transportation costs. Abbaspour et al. (2018) solve the crusher relocation plan by the transportation problem. They defined each mining unit as a source and each pit level where crushers can be located as a destination. Then they investigated different crusher relocation intervals and considered the case with the lowest operating and relocation costs as the optimum plan.

There has been an increasing amount of literature on mine planning and design under IPCC systems in recent years. Some studies considered IPCC systems optimization as a production scheduling problem. Nehring et al. (2018) compared the NPV of different mining sequences of the pure truck, semi-mobile, and fully-mobile IPCC systems. They found that the IPCC system is more applicable for large-scale mines with large horizontal extensions and stable mining plans. Builes (2017) used a mixed-integer goal programming model to maximize the NPV, and instead of a fixed production rate, a set of goal deviational variables and penalties were set. Paricheh and Osanloo (2019) proposed an integrated mixed-integer linear programming model to solve semi-mobile IPCC system planning problems and equipment costs synchronously. Although they specify a set of initial candidate conveyor locations, they did not consider the conveyor wall location. Samavati et al. (2020) solved the mine production scheduling problem

under fully-mobile IPCC systems, with additional sequence constraints. Moreover, the effect of IPCC systems' implementation on the ultimate pit limit (UPL) was investigated (Hay et al. 2019). The authors determined the conveyor wall's optimum orientation, which gives maximum present values. The UPL is generated by the network flow method with the additional conveyor wall requirements.

Although many studies have evaluated IPCC systems' operating and capital costs, fewer have analyzed the production scheduling problem under these systems. Furthermore, most recent studies focus on crusher location and relocation plans based on a set of predetermined candidate sites (e.g., the centroid of each level). However, finding the proper candidate locations is a critical challenge in real cases (Paricheh and Osanloo 2019), and a greater number of these locations can significantly increase the complexity of the model. Additionally, the crusher location plan can be infeasible for conveyor implementation. This study considers the scheduling problem from the conveyor location's perspective, aiming to propose a new mathematical framework for optimizing the conveyor and crusher locations under semi-mobile IPCC systems that maximize the NPV, while considering the material handling and crushing station relocation costs.

## 1.4 Objectives of the Study

The study's main objective is to develop, apply, and validate a theoretical optimization framework for long-term open-pit production scheduling in the presence of a semi-mobile IPCC system. The goal of the proposed mathematical models is to maximize the NPV of the mining operation while incorporating the material handling and crushing station relocation costs. A series of in-pit conveyor layouts and the corresponding mining sequences are investigated, with the maximum economic return measured by NPV. Furthermore, a crusher location-relocation plan is considered based on the facility location problem to minimize transportation costs under the generated mining schedule and conveyor layout.

To achieve these objectives, this work includes the development and testing of the conceptual multi-step framework that focuses on:

- Developing a mathematical model to generate a strategic production scheduling plan to maximum the NPV under semi-mobile IPCC systems while respecting the additional constraints introduced by the system location;
- Generating a crusher location and relocation plan with the minimum total costs of material handling and crusher relocation based on the conveyor layout;
- Evaluating the mining direction and precedence among mining units with a fixed conveyor located in one pit side;
- Identifying the optimum conveyor layout by comparing the different NPVs obtained by a series of conveyor locations around the UPL;
- Developing a methodology to cluster blocks with size control and determine the cluster extraction precedence while considering the additional mining direction constraints.
- Designing the dedicated ramp slot for conventional conveyor belts whose inclination angle is flatter than the stable pit wall slope while comparing the NPV with high angle conveyor.

## 1.5 Scope and Limitations of the Study

The study addresses the long-term production scheduling and crusher location-relocation problem for open-pit mines with semi-mobile IPCC systems as a material handling method. Some assumptions and simplifications limit the optimization framework compared to the real-world case, the geological, operating, and marketing requirements.

- This research considers the optimization of a semi-mobile IPCC system's situation. Thus, there is a small fleet of trucks hauling material from the loading points to the crushing station, from where the conveyor transfers the crushed material to exit the pit. The in-pit crusher is fixed for typically 2-5 years before relocation.
- Input data in this framework, including the block grade, commodity price, and all types of operating costs, have deterministic values. The geological and economic uncertainties are not considered and out of the scope of the study.

- The UPL of the block model is determined before scheduling. Because the conveyor line is installed along the final pit wall, the slope angle of the UPL should be stable at different benches.
- The main conveyor line in this IPCC model is fixed through the mine life. Therefore, the proposed methodology is not applicable for mobile conveyors as the primary lifting system. However, the conveyor line should be extensible to a lower level as the mining operation advances.
- This thesis focuses on a long-term open-pit production scheduling plan. Short-range operational mine plan, haul road layout, and truck-and-shovel dispatch are not considered.
- The model is solved at the cluster level; therefore, the proposed models cannot obtain the production scheduling plan within a specific cluster. The scheduling resolution generated from the model is based on cluster size.

## 1.6 Research Methodology

The primary motivation for this research's development is to optimize the production scheduling under semi-mobile IPCC systems by investigating different conveyor locations and their impact on the mining precedence into mathematical models to maximize the NPV while minimizing the total transportation costs. The scheduling optimization models work at the cluster level. Figure 1-1 illustrates a summary of the research methodology.

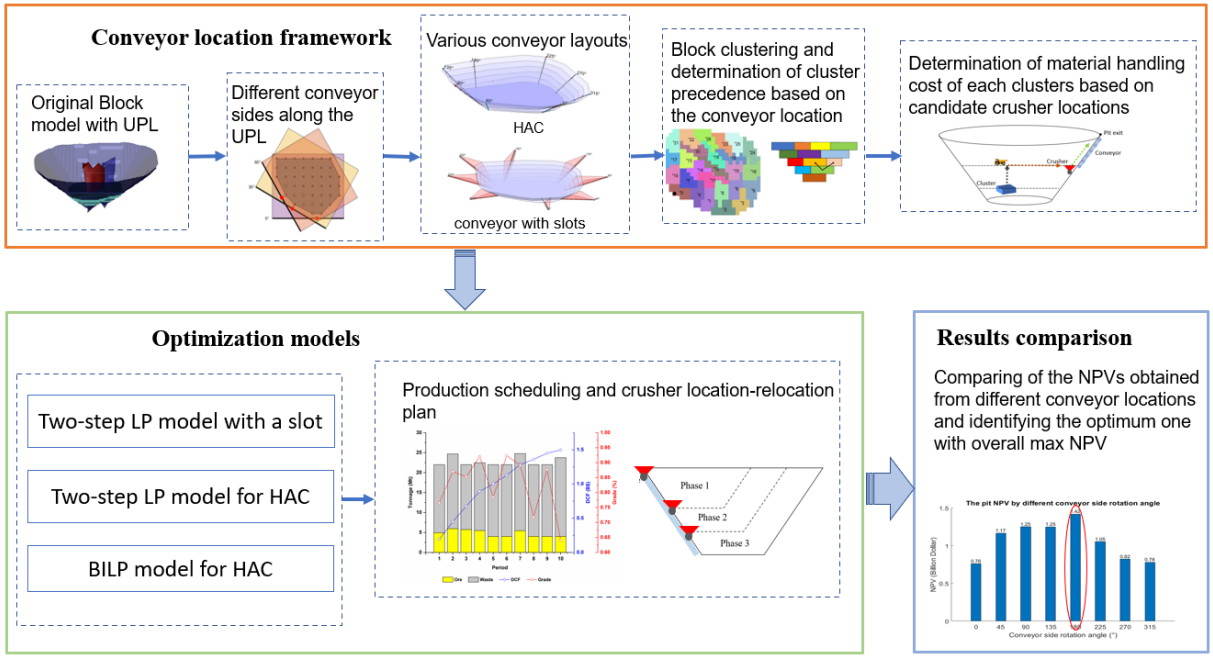



Figure 1-1. Summary of research methodology

The following summarizes the research tasks that are completed to achieve the study's objectives:

- Establish a theoretical framework using coordinate rotation and the least square method to generate a series of candidate conveyor locations along the final pit wall.
- Propose a modified hierarchical clustering algorithm to generate clusters with concentrated size while considering the mining direction. A decision-making process determines the precedence relationship among clusters.
- Evaluate the material handling costs based on the locations of the loading point and crushing station.
- Develop two-step LP models to determine the optimum production scheduling plan and crusher location-relocation plan successively. The first step is a MILP model that aims to maximize the NPV under a specified mining direction; the generated scheduling results are fed to the second step, a facility location problem, to minimize the total transportation costs.

- Propose a conveyor ramp slot situation with additional waste extraction on one side of the UPL to accommodate the conventional conveyor. Compare the optimization results with HAC.
- Develop a BILP model to maximize the NPV while making production scheduling and crusher location-relocation decisions simultaneously.
- Implement the formulations in a MATLAB platform using IBM CPLEX as the optimization solver.
- Test the methodology on a case study to assess the results in terms of the mining plan's practical feasibility and validity.

## 1.7 Scientific Contributions and Industrial Significance of the Research

Due to the high initial investment and the reduction of operating flexibility, IPCC systems require accurate planning before their application. In most cases, decision-makers still rely on qualitative analysis through broad professional experiences, knowledge, and engineers' judgment. It can take weeks to generate a mine plan that can be far from optimal (Samavati et al. 2020). Therefore, this research has developed mathematical models for open-pit production scheduling and crusher location-relocation planning under semi-mobile IPCC systems implementation. The resulting techniques and formulations improve the current literature on open-pit production planning by:

- Introducing a new production scheduling optimization strategy under semi-mobile IPCC systems by incorporating the mining direction requirements from the conveyor's perspective.
- Developing a conveyor location optimization framework by generating various conveyor lines along the final pit wall. The conveyor location that gives the overall highest NPV is considered as the optimum conveyor location.
- Proposing mathematical models for making the production scheduling and crusher location-relocation decision to maximize the NPV of the mine and minimize the transportation costs, with respect to operational and technical constraints.

- Introducing a modified block model clustering algorithm that generates the mining units with stable size; developing a decision-making procedure to determine the extraction precedence relationship between the mining units.

The presented framework's main application is in the feasibility or early development stage of the open-pit mine projects, where an economic and operational valuation is required for a deposit with a high degree of confidence. The production scheduling and crusher location-relocation decision generated by the proposed mathematical model can provide a reference for the future mine plan and the implementation of IPCC systems.

## 1.8 Organization of the Thesis

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

Chapter 2 reviews the literature related to this project. The review starts with an overview of the IPCC systems and conveyor exit scheme. It reviews the relevant IPCC system optimization studies in the crusher location problems and production plan. It also includes the background in open-pit mine planning and design and the approximate strategies used in solving large-scale production scheduling models.

Chapter 3 contains the theoretical framework of the different conveyor location scenarios. This chapter includes the determination of conveyor lines around the final pit wall based on a rotation process. A modified clustering algorithm and a decision-making process for determining cluster precedence relationships are proposed. An approach for material handling costs estimation is presented. Moreover, a ramp slot on the UPL for the conventional conveyor’s case is established.

Chapter 4 provides the three mathematical models for open-pit mine production scheduling under a fixed conveyor wall. It contains a two-step LP model for the conventional conveyor case, a two-step LP model for the HAC case, and a BILP model for the HAC case. The BILP model can solve the production scheduling and crusher location-relocation problem simultaneously, where the two LP models solve the two problems successively. The

conventional conveyor's situation with a ramp slot and extra waste extraction is considered in the first LP model.

Chapter 5 provides the implementation of the conveyor location framework and the mathematical models. A small test dataset is used in this chapter to formulate the models and study the effect of different conveyor locations on the NPV.

Finally, the summary, contribution of the research, and suggestions for future work are discussed in chapter 5.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 In-pit Crushing and Conveying System

### 2.1.1 General Overview

The IPCC system is a continuous haulage system that comprises feeding, crushing, conveying and discharging processes (Osanloo and Paricheh 2019). The concept of the IPCC system was first proposed in Germany in 1956. The soft and wet floor conditions restricted the trucking operation of a limestone quarry (Koehler 2003). The in-pit crusher was mounted on a fully mobile crawler and connected to an overland conveyor. However, Since then, the IPCC systems had not drawn much attention until the last two decades. According to Ritter (2016), at least 447 in-pit crusher stations have been installed globally by 2014.

Some early literature classifies the IPCC systems based on the crushing station's mobility into mobile, semi-mobile, movable, modular, semi-fixed, and fixed (Ritter 2016). Nowadays, the mining industry simplifies the classification into fixed, semi-mobile and fully-mobile IPCC systems (Darling 2011, Ritter 2016, Builes 2017):

- The full-mobile IPCC system (FMIPCC) is an integrated transportation method. Trucks are not required, and the materials are fed by shovels or dozers directly. The crusher station, usually track mounted or wheel mounted, should be flexible enough to follow the working face with shovels simultaneously and continuously.
- In a fixed IPCC system, the crusher is stationary in one site for over ten years. The crusher station is typically mounted on a concrete structure where the mining operation cannot be affected throughout the mine life. The crusher is typically located in the pit rim (rim-mounted crusher), or on the other side of the pushback direction.

- In a semi-mobile IPCC system (SMIPCC), They are typically located near the operational level; rather than hauling or conveying material all the way, the material delivered by trucks is first dumped into in-pit crushers and thereby conveyed by conveyor to move out of the pit. When pushbacks go further, the crushers should be relocated to keep the location near the centroid of the working face. The relocation frequency can be 1 to 10 years, depending on different situations.

Much research on the economic comparison of the IPCC systems and the conventional truck and shovel operation has been done (Tutton and Streck 2009, Oberrauner and Turnbull 2013, Dzakpata et al. 2016, Nehring et al. 2018, Bernardi et al. 2020). The main concern lies in the area of operating cost and capital expenditure. Generally, an IPCC system requires considerably more capital investment than a pure-truck system, but the IPCC systems typically have lower operating costs in the long run. The savings by IPCC systems varies depending upon the mining operation size (Osanloo and Paricheh 2019). For a large-scale mine that requires long strip mining, as the equipment is more flexible to move along a straight bench conveyor within a level, the operating cost for material handling is massive, which can justify the IPCC usage. For a pit of 200m deep, the cost can be about the same for both systems (Hartman et al. 1992). Also, the IPCC systems can offer more savings for longer mine life. McCarthy and Eng (2011) note the mine life is the most critical factor for IPCC's application, which should be greater than ten years. This requirement is due to IPCC systems' high initial investment and needing a long-term operation to justify its low operating costs. Moreover, the capital expense of the IPCC system can be lower over the mine life. That is because the associated equipment of IPCC systems such as are usually replaced every 20–25 years, while the economic replacement age for trucks is around 7-10 years (Dean et al. 2015)

Since its appearance, IPCC systems have been mainly used for ore material handling. However, IPCC systems are also gaining attention for waste as the striping ratio increase. In the last decade, a rising proportion of IPCC systems are installed for waste handling. From 2010 to 2014, approximately one-third of newly applied IPCC systems have been used for waste transportation (Ritter 2016, Osanloo and Paricheh 2019).

### 2.1.2 Conveyor Exit Schemes

For the conventional conveyor belt, the maximum inclination angle is determined by the repose angle of loose materials, vary from 15˚ to 22˚ with respective angles of repose from 29˚ to 44˚(Dos Santos 1986). This inclination angle is considerably lower than the pit slope (around 40˚), which means the conventional conveyor cannot be directly implemented along with the pit limit. In the previous practice, four methods have been proposed to install the conveyor with the inclination requirement (Paricheh and Osanloo 2019):

- Implement the conveyor along an existing haul road;
- Construct a dedicated (generally steep) conveyor slot;
- Excavate an inclined tunnel;
- High-angle conveyors.

The first three methods involve the conventional conveyor. The maximum ramp gradient is between 8% and 10% (4.57° to 5.41°) for the existing haul road to maintain off-road trucks' performance. Spiral ramps and switchbacks are applied to mitigate the ramp gradient (Yarmuch et al. 2019). However, the typical conveyor gradient angle is 18˚, much higher than the haul road ramp. Implementing the conveyor on the existing haul road can increase the minimum conveyor length by at least 2.5 times and consume more power (Oberrauner and Turnbull 2013). Also, the ramp's width must be expanded to accommodate both the conveyor and truck systems.

The concept conveyor ramp slot has been practiced in the past at Chuquicamata and Carmeaux mines, and the crushing station can be installed on the slot (Tutton and Streck 2009). The plan view sketch of an open-pit mine with a slot is shown in Figure 2-1. Although the slot can flatten the pit wall's slope, it results in massive waste extraction and impacts the pushback development. The construction of a tunnel is subject to geological and topographical conditions, and it is technically the last option to be used in most cases (Turnbull and Cooper 2010).

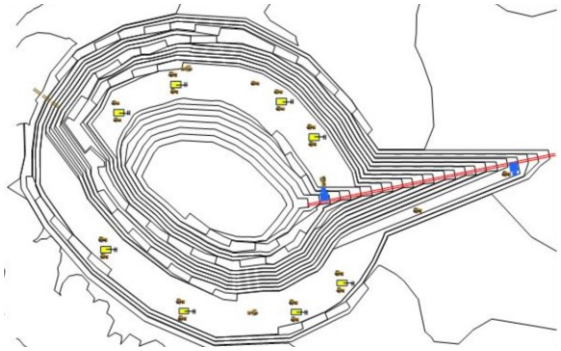


Figure 2-1. A plan view of conveyor ramp slot (Tutton and Streck 2009)

On the other hand, applying a high-angle conveyor (HAC) can avoid many obstacles rising by the conventional counterpart. Because HAC can transport material at a deep angle along the pit wall, the conveyor belt length is minimized, and no extra waste material needs to be mined. The slope angle of HAC can be up to 90˚ (Dos Santos 2017), so it can be installed along the pit wall without extra mining. HACs are designed in various forms, including cleated belt conveyor, pocket belt conveyor, and bucket elevator, while the sandwich belt conveyor is the most used in the mining industry (Santos and Frizzell 1983). The concept of the sandwich conveyor is employing two conventional belts that are parallel to each other, moving together and firmly contacting the material between them. This setting increases the friction angle between material-to-belt and materials-to-material interface and avoids material backsliding, making steep incline runs achievable (Dos Santos and Stanisic 1986). The HAC structure is anchored to the mine slope and is mounted on concrete footings (Ritter 2016). Steel structures, such as truss spans and bents, are required to reinforce the conveyor in a high slope angle and curvature (Dos Santos 2013). Because the construction of these structures contributes a significant part to capital costs, the HAC is not intended to relocate for a long time. However, some HAC systems are adaptable to multi-module sections using self-contained units, and can be extended or shortened by connecting another conveyor flight.

## 2.2 IPCC System Optimization

### 2.2.1 In-Pit Crusher Location Problem

The majority of research on IPCC system optimization has been concerned with the in-pit crusher location. Since the emergence of IPCC systems aims to reduce trucking costs in deep and large open-pit mines, the relevant research has focused on minimizing transportation costs

using approaches such as the facility location problem (FLP). FLP is a branch of operations research that focuses on the optimal positioning of facilities that gives the minimum transportation costs while subject to various constraints. In this problem, the in-pit crusher station is considered as the facility site where material from the working face is transported to the crusher station. The optimization goal is to minimize the total transportation and relevant cost from a given set of candidate crusher sites.

The basic form of an FLP model can be formulated as:

$$\textit{min} \qquad \sum_i \sum_j C_{ij} x_{ij} + \sum_t R_t y_t$$

$$\textit{subject to} \qquad x_{ij} = 0 \; or \; 1, \;\; \forall i, \; j$$
$$y_{ij} = 0 \; or \; 1, \;\; \forall i$$

Where $C_{ij}$ is the transportation cost from block i to crusher station j, $R_t$ is relocation cost in period t, the $x_{ij}$ is equal to 1 if the material of block i is sent to crusher j and equal to 0 otherwise; $y_t$ is equal to 1 if the crusher is relocated in period t and equal to 0 if it is static. The objective function is to minimize the combination of transportation and crusher relocation costs.

Rahmanpour et al. (2013) apply a single hub location problem, an extension of classical FLP, to find the optimum crusher location. In this model, each possible crusher location is a single hub node that aims to minimize the total truck haulage and crusher relocation costs. The problem is solved by an integer-programming formulation. The authors use a decision-making approach called the analytical hierarchical process to reduce the number of candidate crusher locations to four. This research, however, is static and based on the short-range plan without considering the crusher relocation and the conveyor location.

The subsequent studies incorporate the time factor to investigate the time-dependent crusher relocation plan and the IPCC system implementation timing. Paricheh et al. (2017) consider the crusher location-relocation plan as a dynamic location problem. They examined all IPCC introduction times from the first capital payback period to the end of mine life, each as different scenarios. Each scenario was solved independently as the total haulage costs minimization problem in a deterministic mining schedule. The author found the optimum transition period from TS to IPCC system and its corresponding crusher relocation schedule.

The same authors did a further study to integrate the transition time from a pure-truck system to an IPCC system using a dynamic location model and heuristics (Paricheh et al. 2018). They divided the problem into optimum crusher location (OL) and optimum IPCC application time (OT). For the first part, as in their previous paper, they solved the OL problem by using dynamic facilities location problems, obtained the optimum crusher locations for each year after the IPCC system introduced; the second part determined when to transfer the shovel-truck system into the IPCC system based on the result of the first part. The second part aims to maximize the NPV with respect to the transfer time from the pure trucking to IPCC systems and the corresponding discounted cash flow. The authors combined the two mathematic models into an iterative loop until the NPV converge to the optimum. However, they assumed the gravity centre of each level as the candidate crusher location, which could be infeasible when the operating face and the crusher are in different levels. For example, to support the crusher station, the materials of lower levels cannot be fully extracted; if the crusher station is located lower than the operating level, then the mining sequence should be changed. Also, they used a predetermined discounted cash flow to calculate the NPV and did not consider the influence of applying IPCC system on production scheduling.

Another operations research approach to solve the crusher location problem is done by Abbaspour et al. (2018). The authors use the transportation problem to solve the crusher location and relocation plan of semi-mobile IPCC systems. They defined each mining unit as a source and each bench where crushers can be located as a destination. The author then investigated different crusher relocation intervals and found the optimum relocation plan with the lowest operating and relocation costs. Although this study was a long-term plan, the production schedule is predefined without considering the systems relocation plan. Furthermore, the author did not mention the location of the system within a specific level, which can lead to an inaccurate operating cost.

Other approaches such as simulation, enumeration, transportation, and heuristic algorithms have also been used to solve the crusher location problem. Sturgul (1987) adopted the discrete event simulation (DES) method to simulate the truck cycle time for three possible crusher locations. Based on the means and deviations of the trucks' discrete events (load, haul, dump and return), the author applied GPSS (general-purpose simulation system) to simulate the truck and shovel cycle time within a certain period for each candidate. Peng and Zhang (1988) did

similar work. They considered the capacity of in-pit loading-haulage and crushing-conveying systems. They simulated the production rate of different conveyor crusher locations as well as equipment size. However, this model is restricted to an existing mine dataset, and no optimal solution was proposed.

Konak et al. (2007) established an enumeration approach to solve the long-term crusher location plan for a limestone quarry. They applied a trial-and-error process to enumerate all possible crusher locations level by level. The mine deposit is divided into 5 m thickness slices, where the crusher is on the edge of one slice. The slice level gives the minimum overall haulage distance is the optimum crusher location. For haulage between different levels, upward haulage coefficients are introduced to adjust truck travel distances. They also considered the crusher relocation as the crusher level change and solved it by a heuristic approach. However, they did not consider the crusher location within a certain level. The crusher relocation cost was not mentioned either. Similar to Konak's study, Taheri and Irannajad (2009) considered crusher install and replacement costs and conveyor capital costs. They considered three crusher location alternatives and calculated the total operating costs for each case. Because the number of possible crusher location alternatives is more than thousands, the proposed several candidate locations may be far from optimal. Even though trial-and-error method is intuitional and safe to find a solution, it may not find the optimum solution or even all solutions, and the computation could be huge and infeasible to test every single alternative.

Roumpos et al. (2014) studied the belt conveyor distribution point, where materials are sorted and transferred to different dumpsites or processing plants. The problem was treated as a p-median problem. The objective is to minimize the mining face's total transportation cost via the distribution points and finally to different destinations. They applied an iterative methodological model to evaluate the distribution points under a spatial analysis perspective. In the case study, eight scenarios with varied waste dump site placement were simulated. The results suggest the total transportation cost and optimum distribution point location highly depend on the external dump location. Instead of a fixed production rate, goal deviational variables and penalties were set.

In the previous research, a number of candidate crusher locations are predefined. The size of the crusher location problem can be largely controlled by well-defined candidate conveyor

locations. In this sense, Paricheh and Osanloo (2019) developed a search algorithm for in-pit crusher candidate locations. They aggregated blocks within the same azimuth ranges, pushbacks, and benches into clusters, where each cluster denotes a candidate location. The algorithm searches all candidate locations and removes those locations defined as infeasible through a series of rules such as processing plant location, depth restriction, pushback restriction. In the case study, the total number of candidate points is reduced from over 3000 to only 23, making the IPCC planning problem tractable.

Some recent studies apply a heuristic approach to the crusher location determination. Gu et al. (2020) establish a heuristic algorithm to solve the crusher location problem on a large scale. They develop a two-stage fusion particle swarm algorithm (TSF-PSO) to optimize the open-pit mine crushing station layout. The objective is to minimize the total transport work, which is a combination of each block to the crushing station and its tonnage. Even though the authors have considered the dynamic nature of loading points and the truck hauling path as the working face moving forward, they do not consider the crusher relocation plan throughout the mine life. This algorithm can solve the crushing location problem with 301,875 blocks in 125 seconds. Yarmuch et al. (2017) adopt Markov chain model to determine the crusher location with the minimum capital and operating costs while considering equipment failure probability. Two alternative location configurations are evaluated using the stationary probabilities of a Markov chain model, and the results are validated with a discrete-event simulation model. The Markov model generates insights into the relationships between the variables that a discrete-event simulation cannot provide, and does so without the latter's greater costs and complexities of modelling, solving and calibration.

### 2.2.2 Mine Planning under IPCC Systems

Compared to the traditional TS system, the IPCC system introduces additional mining sequence and pit expansion constraints. While most open-pit production scheduling studies are based on the TS system, there are almost no studies for optimizing the operations under IPCC systems' application until recent years. Nehring et al. (2018) compared the different mining sequences of the pure truck, semi-mobile, and fully mobile IPCC systems in a 2-dimension block model. The production scheduling plan was defined for each case scenario based on the ore grade distribution and crusher relocation plan. They found that IPCC haulage scenarios,

especially FMIPCC, have more overall resource recovery and longer mine life than pure truck haulage scenarios. The reason is that IPCC systems have lower operating expenses and can maintain positive economic block value for lower grade blocks. The authors concluded FMIPCC system is more suitable for large-scale mines with large horizontal extension and stable mining plans. The case study shows a 58.9% NPV increase for the FMIPCC scenario compared with the truck scenario.

E. Hay et al. (Hay et al. 2019) investigated the influence of semi-mobile IPCC systems on the ultimate pit limit (UPL). Firstly, they determined the optimum orientation and the depth of the straight conveyor ramp. The conveyor ramp is along one side of UPL and can give maximum pit discounted value. Next, based on the optimum conveyor wall, a UPL was generated by the network flow method with the additional mining dependencies. This study considered various conveyor line locations at the different edges of the UPL; however, due to the configuration, the conveyor cannot reach a deeper level, which restricts the model to a horizontally developed pit only. Moreover, the block extraction sequence is predetermined by the block's particular location.

Jimenez Builes (2017) considered the IPCC location plan and mine plan scheduling through a dynamic uncapacitated facility problem model. The author incorporated more complex economic factors and equipment operating constraints, such as the truck number and cost, conveyor length and operating cost, IPCC system investment and installation/reinstallation, and stockpile in a MILP model. Instead of a fixed production rate, goal deviational variables and penalty were set. The objective is to maximize the NPV with consideration of production deviation, IPCC operating cost. The author tested the proposed model on a layered deposit with low dips, and applied a k-means clustering algorithm to reduce the model size. Two predetermined conveyor layouts were investigated. The candidate crusher locations are on the conveyor segments. The result shows IPCC transportation cases have about 5% higher NPV than the TS case. However, the candidate crusher locations are limited and cannot guarantee an optimal solution. Also, because the mine in the case study is horizontally developed, the levels of crusher locations were not considered.

Paricheh and Osanloo (2019) proposed an integrated MIP model to solve semi-mobile IPCC system planning problems concurrently. The model comprises three parts: open-pit mine

production scheduling (OPMPS), crusher relocation planning, and truck fleet sizing/replacement planning. However, they specified a set of initial candidate conveyor locations and did not consider the conveyor wall location.

Samavati et al. (2020) establish a programming formulation and a heuristic to solve the mine production scheduling problem under fully mobile IPCC systems. The authors consider a situation that a conveyor network is constructed inside the deposit and the flow of the extracted material relies on the conveyor layout. This configuration introduces additional mining constraints and makes the mathematical model nonlinear. Then they propose a heuristic to solve the LP relaxed problem with the concurrent optimization approach.

## 2.3 Long-Term Open-Pit Mine Planning and Design

Long-term open-pit planning is a large-scale optimization problem made for several years to the whole mine life. This production scheduling problem aims to find the optimum block extraction sequence on a large scale that maximum the net present value (NPV) while obeying a series of technical and economic constraints. This problem is usually solved in the block model, consisting of discretized cubic blocks of the physical mineral deposit. Each of the blocks is assigned an estimated tonnage and estimated mineral grades.

Due to the complexity and the limitation of the computing power, many earlier works solved the long-term scheduling problem by decomposing it into (i) determination of the ultimate pit limit, (ii) nested pit shell (pushback) design, and finally, (iii) temporary production scheduling (Askari-Nasab et al. 2007). The three steps can be solved independently, and the whole problem is decomposed into a series of solvable parts.

At the first stage, the deposit's pit extraction limit, including the orebody and the corresponding overburden, is determined. The optimum ultimate pit limit (UPL) gives the maximum undiscounted pit value that satisfies the slope constraints. The UPL problem can be solved using heuristic algorithms such as the floating cone (Elahi et al. 2012) and the Lerchs-Grossmann algorithm (Lerchs 1965). The floating cone method is a simple heuristic that involves a moving inverted cone with its side corresponding to the pit slope. The apex of the cone is repeatedly applied to each block with positive EBV. The economic value of all the blocks inside the cone is examined to find the maximum positive pit value. Although this method is fast to compute,

it sometimes yields a non-optimal solution. The Lerchs-Grossmann algorithm is based on graph theory. Each block is defined as a node with the EBV, and arcs are generated between two nodes if they have direct precedence relations under the slope constraint. The objective is to find the maximum value closure of the graph. The UPL determination is essential in the planning for surface facilities such as processing plan, waste dump, tailing ponds and other equipment during the mining operation (Johnson 1968). However, these algorithms do not provide a mathematical guarantee of an optimal solution, and they do not consider the mining schedule. Also, an optimal production schedule or extraction sequence throughout each bench subject to geometric and resource constraints may not necessarily fall within the ultimate pit limits(Bjørndal et al. 2012).

After the UPL is determined, pushbacks or phases are designed as a series of incremental nested pits. The pushback selection for each time range is the sub-problem of UPL determination, and every pushback is used as the unit for the subsequent production scheduling (Jélvez et al. 2020).

However, the UPL and pushback design are the determination of pit shells by using undiscounted values. Mining and processing capacities, grade blending constraints and discount rate are not considered in those two steps. However, to generate a time-dependent mine plan, many factors should be taken into account to obtain the maximum NPV. Production scheduling over a certain stretch of time is known as the scheduling horizon, which typically contains three ranges: long-term, medium-term and short-term, corresponding to 20-30 years, 1-5 years and 1-6 month planning horizon, respectively (Osanloo et al. 2008).

The early mine planning method is the trial-and-error, hand-calculated, cross-section approach until the work done by Johnson (1968). This pioneering work applies a linear programming (LP) model to determine a feasible extraction schedule that maximizes the profits over the multiple planning periods. The master model is divided into sub-problems by one period, and each of them is solved independently. Although this method incorporates the time value of money, rock types, and a dynamic economic cut-off to generates optimum results. It does not solve the whole problem. Moreover, because all the variables are continuous in the formula, it can result in the fractional extraction of precedent blocks. The solution may be infeasible with the overlying blocks suspending in the air.

Johnson's LP model is subsequently modified by Gershon (1983). The author adds a set of binary decision variables in the original LP model to denote the status of the precedent blocks. Thus, the partial blocks can be mined only if all precedent blocks have been completely removed. The mixed-integer linear programming (MILP) model provides a more practical extraction sequence in mine scheduling. The model can handle multiple ore processing options and multiple grades. However, the large numbers of binary variables make the model intractable for the large-scale problem. Caccetta and Hill (2003) propose a MIP formulation of the mine production scheduling problem and solve it by a branch-and-cut method. The model contains multiple constraints, including extraction precedence, mining and processing capacities, blending grades, stockpiles and various operational requirements such as minimum pit-bottom width and maximum vertical depth.

## 2.4 Solving Large-Scale Mathematical Models

It is well known that mixed-integer programming is NP-hard (Bixby et al. 2004). The computation time required to solve them increases much faster than the size of the problem. In the real case, the open-pit mine block models usually include several thousand to millions of blocks, making it impossible to find an exact solution for the open pit production scheduling problem in most cases. From the practical point of view, obtaining a feasible solution in a practical time horizon is equally vital as the model accuracy for large-scale problems (Meagher et al. 2014). As a result, approximate techniques should be applied to obtain the near-optimal solution within a reasonable time. The strategies to solve large-scale MILP model including (Liu and Kozan 2016):

- Reduce the whole model size by aggregating the blocks and periods.
- Relax the model constraints.
- Decompose the master model into a number of subproblems
- Heuristics and metaheuristics

### 2.4.1 Blocks Clustering and Aggregation

Block aggregation is the most straightforward way to reduce the model's size. However, there is no universal block aggregation method that is compliable with all block models and

should be tailored to particular instances. In general, blocks can be aggregated from three approaches: reblocking block size, same-level block aggregation and slope-based block aggregation (Mai 2017).

#### 2.4.1.1 Reblocking

Reblocking is the simplest method to reduce the total number of blocks and can be realized in most mine database software; however, the block model's resolution is compromised. Jélvez et al. (2016) improve this method to maintain the granularity of the original block model. They present a heuristic algorithm to aggregate and disaggregate blocks in a large-scale deposit. The algorithm has a forward stage and a backward stage. In the forward stage, the authors divide the block scheduling problem into different time-period and solve each period in sequence. Next, blocks were aggregated into larger units, where each unit is treated as a new block. The simplified problem is solved on the unit level. The backward stage is to restore the resolution of the new block model: for these neighbouring blocks extracted at different periods, they are disaggregated into the original scale and solved through the scheduling problem again. The final solution is the combination of the two stages. Although this algorithm can not guarantee a feasible solution, the authors apply it to the instances in the MineLib library and manage to reduce the problem size (number of variables and constraints) by 80%.

#### 2.4.1.2 Level-based block aggregation

This type of clustering method aims to aggregate blocks at the same level based on several criteria. A significant work developed by Tabesh and Askari-Nasab (2011) is a hierarchical algorithm to aggregate blocks into mining units. The clustering process is based on a pair-wise similarity matrix calculated by rock types, ore grades and plane distances between every two blocks. Blocks are merged in each iteration based on their similarity indices until predefined conditions are met. Because of each attribute's different importance, a set of weights is defined for each type of attribute, respectively. This algorithm is applied on a level basis, so all blocks within a specific cluster are from the same level. The authors apply the Tabu search procedure as the post process to reduce the clusters' precedence arcs between two adjacent levels. The generated mining is preferable for selective mining, as the blocks from the same cluster have similar characteristics. The clusters can be further merged to mining panels or bench-phases, defined as the intersections of pushbacks and mining. Each penal is a massive unit for material

extraction containing a set of mining-cuts and controls the mine production operation sequencing. Tabesh's algorithm has been applied to several studies proved to be an effective block aggregation method in both surface mining (Badiozamani and Askari-Nasab 2016) and underground mining (Nezhadshahmohammad and Pourrahimian 2018).

Other same-level block aggregation attempts are also established by various studies. Klingman and Phillips (1988) calculated the production planning of a phosphate mine based on horizontal layers. The authors built a mathematical model with binary variables to determine whether a layer will be mined and processed. However, the method of generating these layers is not mentioned. Weintraub et al. (2008) apply the k-means algorithms to reduce the size of mixed-integer programming models for a block-caving mine. This partitional clustering method divides each level into several clusters while the total distances (dissimilarity measures) of all the in-cluster blocks to the mean (cluster centroid) is minimal. Another partitional clustering method, fuzzy c-means, is used by Askari-Nasab and Awuah-Offei (2010) and Ren and Topal (2014) to improve the k-means algorithm to create more compact mining units.

The level-based block aggregation methods are more applicable for flat-lying or shallow deposit such as oil sand mines (Badiozamani and Askari-Nasab 2016). A massive unit for material extraction can be defined as a mining panel and significantly reduces problem size. However, the precedence relationship cannot be directly identified by these methods, and the slope constraints maybe not respected between mining units. Additional steps are required to determine the precedent arcs among clusters before fed into mathematical models.

#### 2.4.1.3 Slope-based block aggregation

This type of aggregation strategy incorporates the block precedence and pit slope into the clustering process. Ramazan et al. (2005) developed a so-called fundamental tree algorithm to aggregate blocks in the mixed-integer linear programming problem. Each fundamental tree is treated as a minimum mining unit that contains a group of blocks. All blocks within a tree can be mined under the pit slope constraints, and the summation of the block economic is positive. Each block within the UPL belongs to a specific tree. The authors create sub-graphs of the block model and applied an iterative network flow algorithm to aggregate blocks into trees. Thus, the number of binary variables and constraints in the MIP model is decreased, and the pit slope angle is strictly respected. In a case study, the author aggregate 12350 blocks into 1640 trees

and generate a schedule with a 7% higher NPV than those given by traditional software packages. However, in some cases, a large number of ore blocks are the single FTs that only contain one block, so the mining units cannot be effectively reduced. Also, the generation of FTs and the mining sequence depend on a set of predetermined pushbacks, which impacts the optimality of the results.

Another noticeable work of slope-based block aggregation is proposed by Mai et al. (2018). Recognizing the potentials and drawbacks of existing aggregation techniques, the authors improve the FT method by another similar block aggregation technique called the TopCone algorithm (TCA). This algorithm is capable of controlling the minimum number of blocks in each cluster (top cone) with the pit slope constraint, so the large-scale block model is tractable. Firstly, the block model's network flow is generated based on Lerchs-Grossmann algorithm. A linear programming formulation is set to generate top cones with minimum size and a positive economic value in the clustering step. The iteration is repeated from top to bottom level, and the top cones with a smaller size than the lower bound are fed into the next iteration. The TCA can combine mining precedence requirements with the size and number of generated TCs, making the downstream mathematical scheduling model tractable.

### 2.4.2 MILP Relaxation and Decomposition

The main idea of relaxation is to remove some constraints of the original MILP model. The constraints can be removed from two directions: linear programming (LP) based relaxation and Lagrangian relaxation. The former one aims to relax the integrality of the variables without modifying the objective function; the latter one is based on removing some constraints and transfer them to the objective function by a set of weights (Lagrangian multipliers).

LP-based relaxation transforms an integer problem into an associated linear problem that that can be solved in polynomial time (Boland et al. 2009). The optimal value generated from LP relaxation can be treat as the upper bound in some algorithms, such as branch and bound. The latter is applied to find the exact optimal or near-optimal solution while respect the integrality restriction. Boland et al. (2009) apply linear Programming (LP) relaxation to an iterative block disaggregation procedure in a MILP model. This algorithm can substantially reduce the LP relaxation gap and the computation time required to find the optimal integer

solution in almost all cases. Bienstock and Zuckerberg (2010) develop an iterative algorithm for solving the LP relaxation of production scheduling problem. The algorithm can solve a relaxed MILP model with 87,831 blocks and 12 periods within one hour.

The concept of Lagrangian relaxation is to relax the problem by removing some constraints that are hard to be satisfied and transfer them into the objective function, assigned with weights (the Lagrangian multiplier). The initial application in open-pit mine scheduling is done by Dagdelen (1987). The author solves an integer programming model by decomposing the original problem by dividing the constraints into (i) constraints for block precedence, and (ii) constraints associated with production capacities. The second part is then moved to the objective function and multiplied by various Lagrangian multipliers. This additional part can impose penalties on the objective function value if any violation from the soft constraints. Since then, various mine planning studies have applied the Lagrangian relaxation due to its simplicity and reliability. More recently, Chatterjee and Dimitrakopoulos (2020) integrate Lagrangian relaxation and branch-and-cut algorithm to schedule an open pit mine under geological uncertainty. The algorithm imposes Lagrangian relaxed sub-problems sequentially and then uses the branch-and-cut algorithm to generate an efficient production scheduling model respecting the Lagrangian relaxed solution's feasibility.

### 2.4.3 Heuristic and Metaheuristics

The complexity of the large-scale open-pit mine scheduling problem and its variants led to the development of numerous heuristic/metaheuristic algorithms. These algorithms are approximation approaches that can obtain near-optimal solutions within reasonable computational time.

A heuristic method applied in many previous studies is the sliding time window heuristic (STWH), proposed by Cullenbine et al. (2011). This approach combines the Lagrangian relaxation technique and time decomposition heuristic. The full scheduling periods are divided into three sequential subsets: a beginning part with fixed variables value, a middle period (time window) applied by the original programming model, and a relaxed Lagrangian part for the subsequent periods. The algorithm uses a fix-and-optimize scheme and iteratively solving relaxed sub-problems of a time window with fewer binary variables. The authors solve a

problem instance of 25,000 blocks and 15 periods in a few hours, although this algorithm may not find a feasible solution for some cases.

Moreover, several metaheuristic approaches, including genetic algorithm, simulated annealing, tabu search, ant colony optimization, and particle swarm optimization are also adopted to mine planning in recent years (Shishvan and Sattarvand 2015).

## 2.5 Summary and remarks

IPCC systems have been known in the mining industry for many decades. They gain more attention as material handling options due to their low operating costs and social and environmental benefits. However, due to the high upfront costs, reduced mining flexibility, and conveyor exit requirements, IPCC systems' application requires careful and detailed planning at the early stages of mine life to provide financial savings.

As mentioned in the literature, most current IPCC system optimization studies focus on crusher location and relocation plans based on various predetermined candidate locations. The objective of these studies lies in the total transportation costs minimization. On the other hand, although the mine design and planning under IPCC systems are unique, few studies have been done on the production scheduling problem.

Mathematical programming has been developed as a powerful tool to solve the open-pit mine production scheduling problem and maximize the NPV while considering a number of operational and economic constraints. Some applications of mathematical programming models in mine planning are documented. However, these kinds of problems are NP-hard and can become intractable as the problem size grows. Therefore, various approximate approaches have been proposed to reduce the computation time and obtain near-optimal solutions.

# CHAPTER 3

# THEORETICAL FRAMEWORK

## 3.1 Introduction

The long-term mine production scheduling is a large-scale optimization problem to determine the mining sequence, optimizing the company's strategic objective while respecting the operational and geotechnical restrictions over the mine life. The target is typically maximizing the net present value (NPV), with the optimum production schedule that preferably extracts high-grade materials as soon as possible, and low-grade materials and waste are postponed to the late period of the mine life.

However, the implementation of the IPCC system can reduce mining flexibility and introduce additional mining sequence requirements. For system stability, the conveyor belt is usually anchored to the pit wall and mounted on concrete structures, which means the relocation of this system is rare (Ritter 2016). The conveyor line should be located in such a way so that it cannot affect the future pit expansion.

This framework considers a situation where the conveyor system is fixed along one side of the final pit wall throughout the mine life. The excavated material should be trucked towards this side via a crusher station located on the conveyor line and then transferred by the conveyor system to exit the pit. This implementation introduces additional mining direction requirements: mining starts from the pit side where the conveyor is installed and then expands to the opposite side. The extraction of blocks closer to the conveyor has precedence over others, as Figure 3-1 shows. As a result, the location of the conveyor can impact the mining sequence and the NPV.

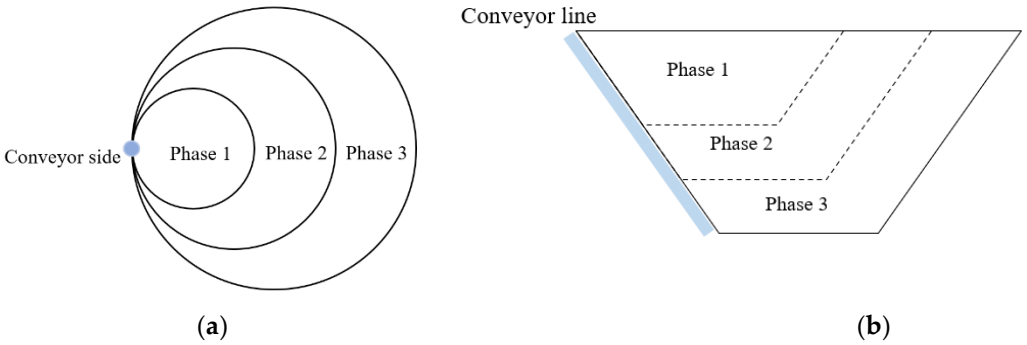



Figure 3-1. Schematic view of the mining method with the conveyor system along one side of the UPL: (a) plan view; (b) vertical view

In this framework, A set of conveyor locations are generated around the final pit wall, and each conveyor location is solved independently by a mathematic model in Chapter 4. Also, the blocks in this model are aggregated into larger mining units to reduce the problem size. A modified clustering method with cluster size control is applied to reduce the number of blocks in each level, and the clusters' precedence relationships are determined based on pit slope and mining direction. The material handling costs and implementation of conventional conveyors are also discussed in this chapter.

## 3.2 Determination of the HAC and Ramp Slot Locations

As mentioned before, the HAC system is usually anchored to the pit wall and mounted on concrete structures in the deep open-pit mine for its stability. The HAC system is also a continuous transportation method, which means the shutdown of any conveyor segment can halt the whole system. As a result, the conveyor is usually fixed in the mining operation to avoid significant downtime during the conveyor system's relocation. In this research, it is assumed the HAC line is fixed along one side of the final pit wall throughout the mine life, where this location cannot affect the future pit expansion. In this section, a series of procedures are proposed to determine a set of HAC schemes in different locations.

### 3.2.1 Conveyor Side Rotation around the UPL

In this thesis a pit rotation approach is proposed to investigate the possible locations along the UPL. In this approach, each scenario's conveyor scheme is based on a series of equal interval rotation angles, making this investigation evenly distributed. This method is initially used by

Hay et al. (2019) to determine the UPL with a straight conveyor wall. This concept is used here as a tool to investigate the possible conveyor azimuth along the UPL.

Based on the existing UPL, a group of convex hulls is created for each level. Each convex hull is the minimum convex polygon that circumscribes all the centroids of blocks to be mined in that level, and all the interior angles are equal or less to 180˚. Then the minimum bounding box is generated from the convex hull. The bounding box is extended an additional 0.5 unit of the block width on each side to include all parts of blocks.

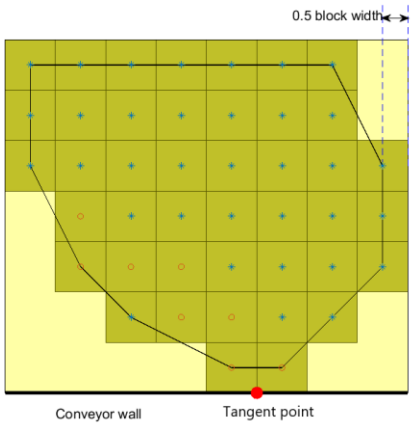



Figure 3-2. Creation of convex hull and bounding box for a specific level

The bounding box's specified edge is considered the conveyor wall side (CWS), as the bold line shows in Figure 3-2. The point of tangency between the UPL and the CWS is identified as the conveyor feeding spot. If there is more than one point of tangency, the midpoint is considered, shown as the red point in Figure 3-2. The bounding boxes are determined again at each rotation angle, and the tangent points also move around the UPL accordingly. Figure 3-3 shows the tangent points for a specific level when the bounding box rotates 0˚, 30˚, and 60˚ clockwise, respectively.

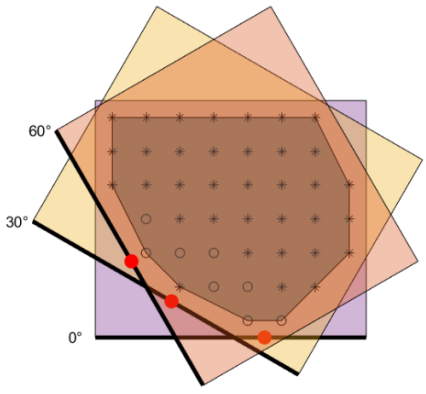


Figure 3-3. The plane revolution of the bounding box around the convex hull

The bounding box's rotation is equivalent to the pit's anticlockwise rotation. The latter method is simpler to implement by multiplying a rotation matrix to the block coordinates and ensures all levels have a uniform rotation angle. Equation (3.1) shows the rotation transformation with a 3×3 matrix (Deutsch). The rotated coordinates $(x', y', z)$ are obtained by rotating the X and Y axes in a clockwise angle α with respect to the Z-axis, and the coordinate in the z-axis remains unchanged. Figure 3-4 shows the X and Y axes rotation effect with respect to a specific pit level. The bounding box and the CWS are updated under the new coordinate system so that the CWS can rotate around the level's convex hull at α angle. The tangent point is also determined based on the CWS.

$$\begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.1}$$

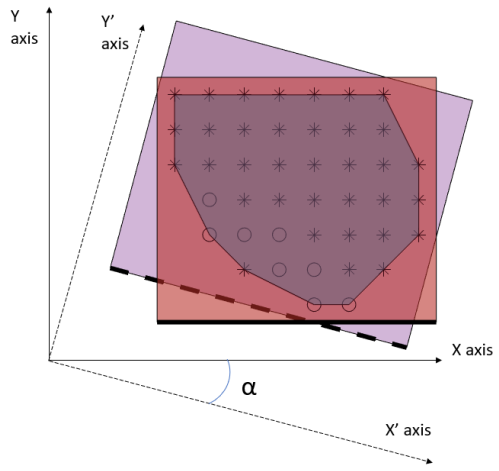



Figure 3-4. Pit level and the new bounding box with a 2D rotation

Next, this rotated coordinate system is applied to all levels within the UPL. It should be noted the rotation transformation will change the original X and Y coordinates of each block; however, since the study focuses on the relative location between the conveyor and the pit, the absolute coordinates during the rotation are not a concern. Then, the coordinate system rotates clockwise by a step angle, and the associated CWS and tangent points are updated each time until it returns to the original position. Each rotation angle is a scenario for later calculation. The step angle should have a decent resolution to cover all scenarios accurately, but not too small as

the computation time will increase. Besides, the pit walls' geotechnical condition plays a significant role in selecting the step angle.

### 3.2.2 Generation of the Straight Conveyor Line for HAC

After calculating all the tangent points at all levels with the same rotation angle, the three-dimension least square regression algorithm is applied. This method can generate a spatial straight line from the pit bottom to the rim, where the sum of squared distances from each tangent point to the generated line is minimum, as showing in Figure 3-5. This line is considered as the layout of HAC, and it is generally on the pit boundary with an inclination equal to the pit slope. Moreover, two sets of HACs should be installed on the layout line for transporting ore and waste, respectively. This setting can also increase the throughput of the conveying system as the capacity of HAC is generally lower than the conventional conveyor (Dos Santos 2016). The capacity of HAC for ore and waste may be different depending on the overall stripping ratio.

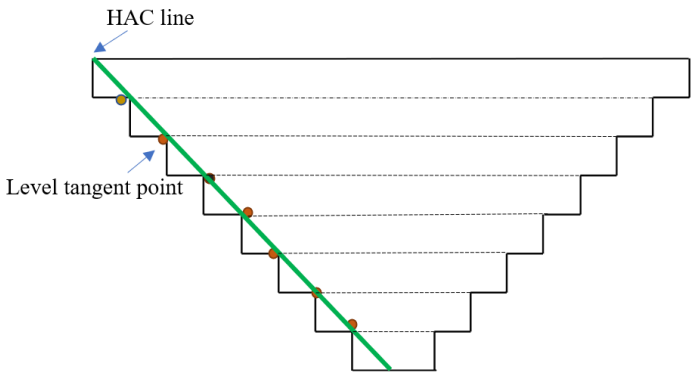



Figure 3-5. The schematic diagram of the pit with the regression line

In order to calculate their linear regression, first, the coordinates of the tangent points are transformed into vectors and a matrix. Assume that the block model has j levels, and each level has a tangent point based on the conveyor side rotation. The coordinates of each tangent point are denoted by $x$, $y$ and $z$, among which $z$ is a free-of-error variable to predict the value of x and y, respectively. Equation (3.2) shows the variables in vectors and matrix form. $\mathbf{X}$, $\mathbf{Y}$ are two $j \times 1$ vectors that represent the coordinates of all tangent points in the x-axis and y-axis,

respectively; $\mathbf{Z}$ is a $j \times 2$ matrix where the elements of the first column are all 1, and the secon11d column is the z-axis coordinates of all tangent points.

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \end{bmatrix} \qquad \mathbf{Z} = \begin{bmatrix} 1 & z_1 \\ 1 & z_2 \\ \vdots & \vdots \\ 1 & z_j \end{bmatrix} \tag{3.2}$$

The concept of weighted linear regression is used to increase the model's accuracy and adaptability. The weight of each observation (tangent point) is a positive number measured by the total undiscounted block economic value (EBV) of the corresponding level, as Equation (3.4) shows. The summation of all weights is equal to 1. Level with relatively low value is given lower weights to its tangent point, and verse visa. By incorporating the knowledge of the value, the generated straight line can be closer to the tangent points of high-value levels than the standard linear regression algorithm. Therefore, the original pit design and the boundary can be respected in the high-value levels. $\mathbf{w}$ is a $j \times j$ diagonal matrix with all the weight values in its diagonal (see Equation (3.4)).

$$w_i = \frac{block\ conomic\ value\ in\ level\ i}{total\ undiscounted\ \ pit\ value} \tag{3.3}$$

$$\mathbf{w} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_j \end{bmatrix} \tag{3.4}$$

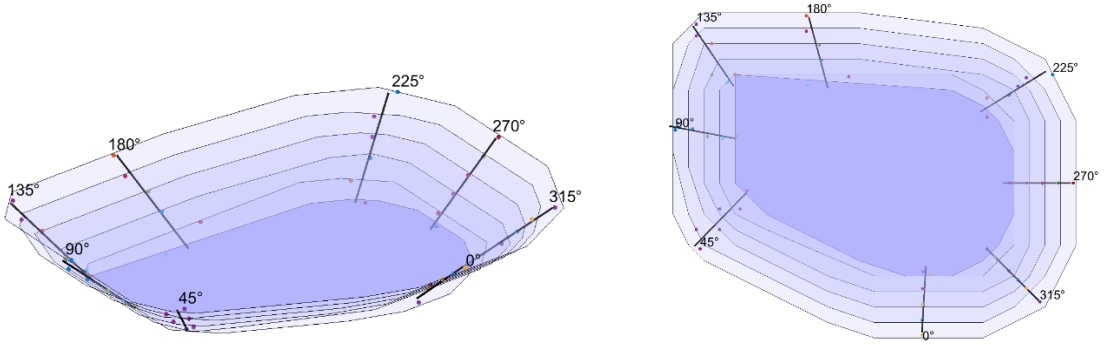
The fitting line is three-dimensional that contains an independent variable z and two response variables x and y. The linear least square method is applied to each of the response variables with respect to z; therefore, the sum of horizontal squared distance from the tangent points to the spatial fitting line is minimum. The spatial fitting line projection in the XZ-plane is the linear regression for variable x and z, and its projection YZ-plane is the linear regression for variable x and z. Equation (3.5) calculates the projection of the weighted linear regression in XZ-plane and YZ-plan, respectively. The predicted values in matrix form are given by Equation (3.6). $\alpha_0$ is the X-intercept to the YZ plane projection, and $\alpha_1$ is the slope to the YZ

plane; similarly, $\beta_0$ and $\beta_1$ are the Y-intercept and slope to the XZ plane, respectively. The z-axis coordinate of each level remains unchanged after the linear regression. The conveyor line is determined based on the fitting line, from the bottom level to the surface.

$$\begin{cases} \begin{bmatrix} \alpha_0 & \alpha_1 \end{bmatrix} = \left( \mathbf{Z}^{\mathrm{T}} \mathbf{w} \mathbf{Z} \right)^{-1} \mathbf{Z}^{\mathrm{T}} \mathbf{w} \mathbf{X} \\ \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix} = \left( \mathbf{Z}^{\mathrm{T}} \mathbf{w} \mathbf{Z} \right)^{-1} \mathbf{Z}^{\mathrm{T}} \mathbf{w} \mathbf{Y} \end{cases} \tag{3.5}$$

$$\begin{cases} \mathbf{X}' = \alpha_0 + \alpha_1 \mathbf{Z} \\ \mathbf{Y}' = \beta_0 + \beta_1 \mathbf{Z} \\ \mathbf{Z}' = \mathbf{Z} \end{cases} \tag{3.6}$$

The conveyor spot for a specific level is defined as the conveyor line's intersection with that level, whose coordinates can be calculated by Equation (3.6) with the associated z-axis value. Because the crusher is located and relocated along the conveyor line, each conveyor spot is considered as the candidate crusher location. Moreover, the weighted least square method is repeated to generate the straight conveyor lines around the pit limit for the tangent points under different conveyor rotation angles. Figure 3-6 shows the eight candidate HAC lines with a step of 45˚.




(3) (b)

Figure 3-6. The outline of eight candidate HAC lines (black straight line) based on tangent points and rotation angles: (a) perspective view; (b) plan view

In this research, the conveyor system is used to transport both ore and waste. As a result, two sets of HACs should be installed on the layout line for the two types of material, respectively. This setting can also increase the throughput of the conveying system as the

capacity of HAC is generally lower than the conventional conveyor (Dos Santos 2013). The capacity of each HAC may be different depending on the overall stripping ratio.

### 3.2.3 Generation of the Ramp Slot for Conventional Conveyor

In this section, a situation that the conventional conveyor is accommodated in a ramp slot is considered. The generated conveyor line's slope is equal to the pit slope that usually is greater than the conventional conveyors with a slope at most 22°. As mentioned in the literature review, one solution to the inclination problem is constructing a dedicated ramp slot to flatten the slope [23]. This method enables the straight and short conveyor route to exit the pit, and it has been practiced in open-pit mines in the past (Tutton and Streck 2009).

Based on the conveyor line generated for HAC, the ramp slot is corrected in the same vertical plane of the conveyor line considering allowable inclination angle, as Figure Figure 3-7 shows. Both HAC and the conventional conveyor lines start from the same point $(x_0, y_0, z_0)$ at the bottom level, and their horizontal projections are in a line. The transformation between these two lines is shown in Equation (3.7) and (3.8).

$$k = \frac{\tan\alpha}{\tan\theta} \tag{3.7}$$

$$\begin{aligned} \mathbf{x}' &= k(\mathbf{x} - x_0) + x_0 \\ \mathbf{y}' &= k(\mathbf{y} - y_0) + y_0 \\ \mathbf{z}' &= \mathbf{z} \end{aligned} \tag{3.8}$$

Where $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and $(\mathbf{x}', \mathbf{y}', \mathbf{z}')$ are the set of coordinates of the points in the HAC and corrected conveyor line, respectively; $(x_0, y_0, z_0)$ is the start point at the bottom level for both conveyor lines; k is a adjust factor in x and y-axis direction; $\alpha$ is the inclination angle for HAC and its slope is approximate to the pit slope; $\theta$ is the corrected conveyor line's inclination angles to the horizontal plane, with a slope typically between 12° to 22°. Based on the HAC, the conveyor spots of the corrected conveyor line can be calculated by Equation (3.8).

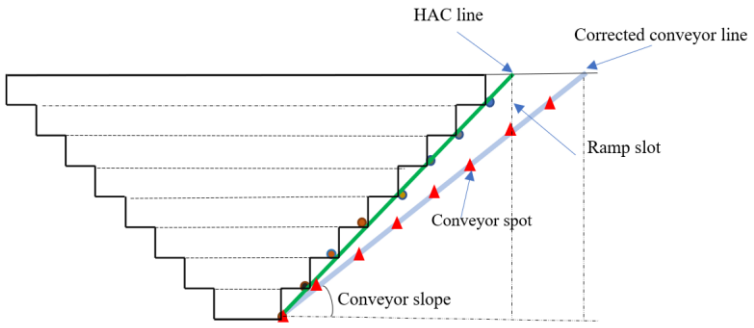
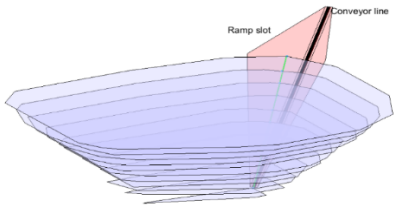



Figure 3-7. The 2D sketch of the pit with ramp slot and conveyor line inside

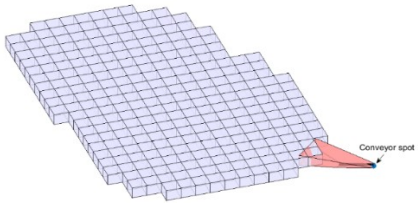
The conveyor spots on the corrected conveyor line are defined at each level as the candidate crusher location, as the triangles are shown in Figure 3-7. Similar to the configuration of HAC, the conventional system transports all materials inside the UPL; therefore, two sets of the conveyor should be installed in the ramp slot to transfer ore and waste, respectively, and the ramp slot width should be wide enough to accommodate two parallel conveyor belts. This conveyor exit strategy introduces additional waste excavation. The 3D sketch of the ramp slot is shown in

Figure 3-8 (a). In this study, the entire slot is divided level by level, and the portion of the slot in each level is called a slot slice, as

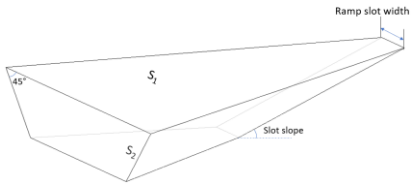
Figure 3-8(b)(c) illustrate. The slot is deepened to the corresponding levels with the mining development. Before the crusher is relocated to a lower level, all slot slices should be extracted at that level.




(a)




(b)




(c)

Figure 3-8. A 3D sketch of ramp slot: (a) the overview for the whole pit; (b) the ramp slot in a level; (c) the geometry of the ramp slot slice in a level

The top and bottom surfaces of the slot slice are horizontal. Among its four side faces, the one separate from UPL is the conveyor face with an inclination of the maximum conventional conveyor slope angle; the other three faces have the same inclination as the pit safety slope, which is 45˚ to the horizontal plane. Each slot slice's shape is not a standard frustum, so its volume is not straightforward to be solved analytically. Alternatively, because its vertices coordinates can be deduced from the conveyor lines, the volume of each slot slice can be calculated by Delaunay triangulation in MATLAB (MATLAB 2018). Delaunay triangulation is an algorithm of joining a set of vertices to make a triangular mesh, and the corresponding volume enclosed can be calculated from these triangulated surfaces (Kudowor and Taylor 1998). Another study that calculates the slot volume analytically can be seen in Paricheh and Osanloo (2019).

The amount of extra waste excavation is mainly determined by the ramp slope and its depth. Figure 3-9 shows the slot volume based on different slot depth and slope angles, with a slot width of 10 m. The slot volume is directly proportional to slot depth's cubic power, so the additional waste extraction tonnage increases dramatically as the pit deepen. The slot slope also has a significant impact on the slot volume. For a depth of 300 m, about 8 million cubic meters of waste removal is required at a 20° ramp slope, whereas this volume is over 8 million cubic meters for a slope of 15°. In addition to the extra waste excavation, the conveyor length must be identified based on each depth and ramp slope.

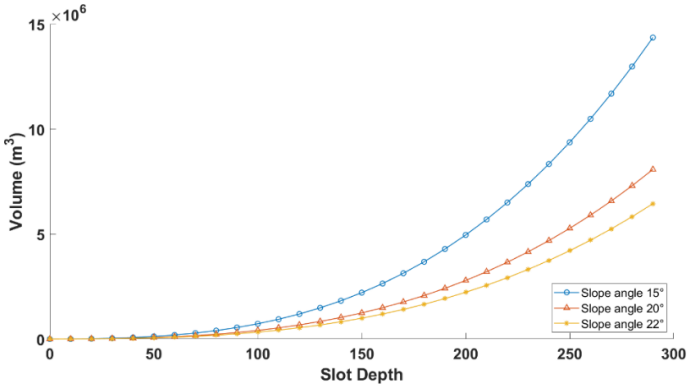



Figure 3-9. The volume of ramp slot by its depth and inclination slope

Similar to the HAC, this process is repeated under each conveyor rotation angle. The eight candidate ramp slots are illustrated in Figure 3-10.

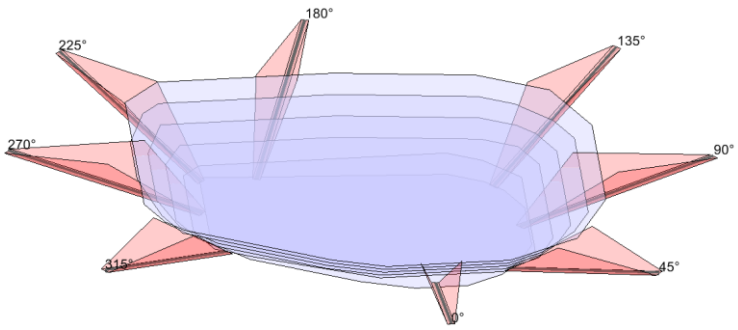



Figure 3-10. The outline of candidate conveyor lines and the corresponding ramp slots under different rotation

### 3.2.4 Pit Wall Stability Considerations for Conveyor Wall Locations

In open-pit mining, a major geotechnical challenge is the excavation of the steepest possible slope angle while maintaining the lowest stripping ratio. As a result, the waste rock removal tonnage and the total mining-related costs are minimized. However, steep pit slopes may induce failures that can overweigh the economic benefits that are initially aimed at {Obregon, 2019 #296}. For the conveyor systems that are anchored on the pit wall, the failure can result in loss of life, damage to equipment and the environment. On the other hand, the conveyor systems require a higher standard for pit wall stability than the traditional pit design, as the implementation of the conveyor belts can introduce additional pressure and vibration. Therefore, the geotechnical and geological conditions of the final pit wall and its slope must be considered before the generation of candidate conveyor locations.

Generally, the geotechnical and geological conditions are different around the final pit wall. In this case, the conveyor should only be installed along the stable pit wall, as an example shows in Figure 3-11. The UPL is divided into two sectors: stable pit wall zone and unstable pit wall zone. Five candidate conveyor lines are generated inside the stable zone. This research only focuses on the conveyor's location from the mining sequence and NPV's perspectives, and it is assumed the pit is stable for the conveyor around the UPL under a specific slope angle. However, it should be noted in terms of the conveyor's location determination, the pit wall stability has a higher priority than the economic comparison proposed in this research.

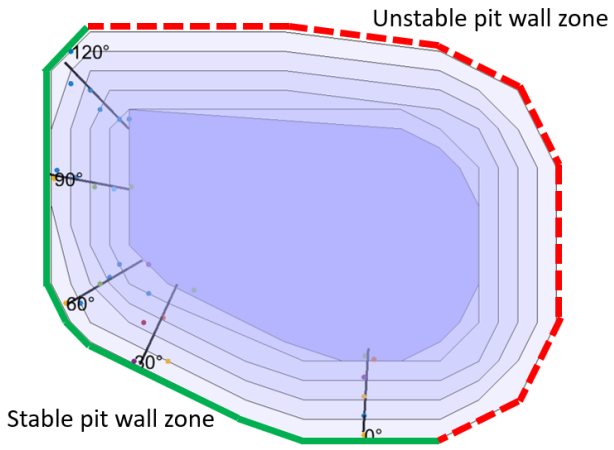



Figure 3-11. The considerations for pit wall stability for conveyor locations

## 3.3 Hierarchical Agglomerative Clustering

It is well known that mixed-integer programming is NP-hard that cannot be solved in polynomial time. The proposed mathematic model contains binary (0-1) integer variables, which can be intractable for the real case open-pit mine block models with thousands to millions of blocks. As mentioned in chapter 2, various techniques such as block clustering, decomposition, relaxation, and heuristic are used in the previous literature. In this research, a block aggregation approach is adopted to group blocks with similar attributions and locations into the same cluster. This approach can significantly reduce the number of mining units and binary integer variables at the cluster level; thus, the problem can be solved in a reasonable time. Moreover, it can generate practical mining schedules that follow a selective mining unit without scattered scheduling solutions and an overestimated NPV (Eivazy and Askari-Nasab 2012).

On the other hand, aggregation techniques are highly dependent on the structure of the problem and, in general, are tailored specifically for a class of problems or even for a specific instance of a problem. In this case, clusters are generated on a level basis, with a relatively stable size that gives both acceptable resolutions and running time. The hierarchical agglomerative clustering algorithm is initially presented by Tabesh and Askari-Nasab (Tabesh and Askari-Nasab 2011). In this framework, the original algorithm is modified to reduce the number of small-size clusters under a specific threshold during the clustering process. The mining direction that adapts to the IPCC system is also incorporated in the modified algorithm.

### 3.3.1 Similarity Matrix Calculation

This cluster algorithm depends on a measure of similarity between the blocks. The similarity values form an $n \times n$ pairwise similarity matrix and merging similar blocks in an iterative procedure. The similarity value between a pair of blocks (with an index of i and j) is measured based on rock type, grade, Euclidean distance, mining direction, and block adjacency Boolean. Adjacent blocks with more similar attributes have higher similarity values; thus, they are grouped into one cluster. The similarity between block i and j are calculated by Equation (3.9):

$$S_{ij} = \frac{RT^{w_{RT}}}{Dis_{ij}^{\ w_{dis}} \times Gr_{ij}^{\ w_{gr}} \times Dir_{ij}^{\ w_{dir}}} \times Adj_{ij} \tag{3.9}$$

In which $Dis_{ij}$, $Gr_{ij}$, $Dir_{ij}$, and $RT_{ij}$ are the normalized value of Euclidean distance, grade difference, mining direction difference, and rock type penalty between blocks i and j, respectively; $Adj_{ij}$ is a Boolean (0-1) value that denotes whether i and j are adjacent blocks; $w_{dis}$, $w_{gr}$, $w_{dir}$, and $w_{RT}$, in the power position, are a set of positive numbers denoting the weights of corresponding parameters. Setting a higher weight for a specific parameter can promote the clustering results to follow that characteristic. For example, increasing $w_{dis}$ can create rounder clusters while increasing $w_{gr}$ makes clusters more compliant with the grade distribution. The value of $Dis_{ij}$, $Gr_{ij}$, $Dir_{ij}$, and $RT_{ij}$ are measured as follows:

#### 3.3.1.1 Normalized Euclidean Distance

The normalized distance difference is the Euclidean distance between two blocks centre divided by the maximum distance of two blocks in that level. This normalization ensures all the distance difference is equal or less than one regardless of the unit's real block dimension. Let $\mathcal{B}$ denote the set of blocks in the target level. The normalized distance between block i and j can be calculated by Equation (3.10):

$$Dis_{ij} = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\max\{\sqrt{(x_m - x_n)^2 + (y_m - y_n)^2} \mid \forall m, n \in \mathcal{B}\}} \tag{3.10}$$

The denominator is the maximum distance of two blocks. The value of x and y can be either the indices or coordinate of the corresponding block centre, only if the unit is consistent in the formula.

#### 3.3.1.2 Normalized Grade Difference

The calculation of the normalized grade difference is similar to normalized Euclidean distance. Let $gr_i$ and $gr_j$ denote the grade of block i and j, and the normalized grade difference can be given by Equation (3.11).

$$Gr_{ij} = \frac{\sqrt{(gr_i - gr_j)^2}}{\max\{\sqrt{(gr_m - gr_n)^2} \mid \forall m, n \in \mathcal{B}\}} \tag{3.11}$$

The numerator is the absolute value of the grade difference between block i and j, and the denominator is the maximum grade difference in that level. Moreover, if $gr_i$ and $gr_j$ have the same value under the dataset's precision, $Gr_{ij}$ will be assigned to a sufficiently small positive number Ɛ to avoid getting a zero value.

#### 3.3.1.3 Rock Type Penalty

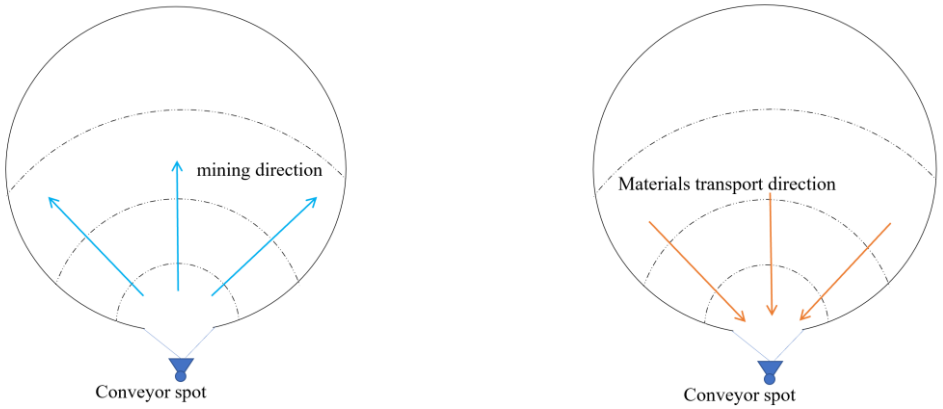
The rock type penalty is a qualitative value indicating whether block i and j belong to the same rock type. If the rock type of cluster i and j are the same, the rock type similarity will be 1. Otherwise, the similarity is a penalty number less than 1, as Equation (3.12) shows.

$$\mathrm{RT} = \begin{cases} 1, & \text{if block i and j belong to the same rock type} \\ penalty, & \text{otherwise} \end{cases} \tag{3.12}$$

The less the penalty value is, the less similarity will be for clusters with different rock-type. The penalty is set to zero means only the block in one rock type can be aggregated into the same cluster. Because the rock type per se is a qualitative value, as a result, the penalty could be varied by different groups of rock types. For example, if the ore blocks are affiliated to some specific rock types, then the penalties within these rock types should have less impact compared with a waste rock type.

3.3.1.4 Mining Direction Difference

In the presented methodology, mining operations should be started from the conveyor spot and then expand to the other side of the level (Figure 3-12a). Blocks closer to the conveyor should be mined before other blocks, leaving space for the latter to be transported toward the conveyor. The materials transport direction is opposite to the mining direction (Figure 3-12b), from the working face to the conveyor side.




(a) Mining direction (b) Materials transport direction

Figure 3-12. Illustration of mining and materials transport direction for a certain level

Blocks with the same distance to the conveyor spot are more likely to be mined in the same period; thus, they should be assigned more similarity in mining direction. In other words, the long sides of clusters should face the conveyor location.

Figure 3-13 illustrates the effect of mining direction on the clustering, where the clusters show rectangular. Clusters with darker colors are mined earlier; for those with the same color, they are more likely to be extracted in one period because they have the approximate distances to the conveyor. Therefore, the mining direction difference between them should be small.

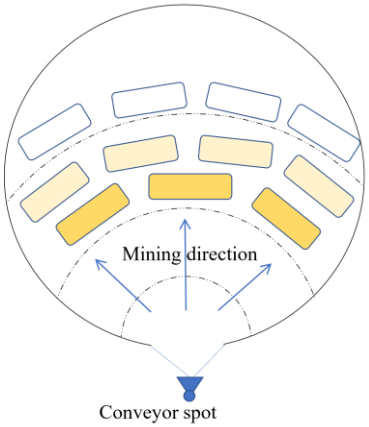



Figure 3-13. Diagram of clusters given mining direction

The mining direction difference measures the difference between two blocks' distance to the conveyor spot. The Equation (3.13) shows the calculation of the normalized mining direction difference $Dir_{ij}$ between block i and j at the same level. Where $(x_o, y_o)$ is the coordinate of the conveyor spot at that level; $(x_i, y_i)$ and $(x_j, y_j)$ are the coordinate of block i and j, respectively. The denominator is the maximum distance between the cluster centroid and conveyor spot at that level.

$$Dir_{ij} = \frac{|\sqrt{(x_i - x_o)^2 + (y_i - y_o)^2} - \sqrt{(x_j - x_o)^2 + (y_j - y_o)^2}|}{\max\{\sqrt{(x_m - x_o)^2 + (y_m - y_o)^2} \mid \forall m \in \mathcal{B}\}} \tag{3.13}$$

3.3.1.5 Block Adjacency Boolean

Each cluster is a mining unit for schedule planning; therefore, clusters should be continuous and closed to be mined without frequent relocating shovels. To avoid generating fragmented clusters, blocks can only be merged to adjacent blocks with a common side. Figure 3-14 shows the four adjacent blocks (light-shaded) of the target one (dark-shaded). The Boolean value $Adj_{ij}$ is set to identify the adjacency between block i and j, based on the horizontal distance between the two block centres (Equation (3.14)). For two adjacent blocks in the same level, their centres' distance is equal to the block width, and the adjacency Boolean between the two blocks is 1. If two blocks are not adjacent, their centre's distance is greater than the block width, and the corresponding adjacency Boolean value is equal to 0. In this case, the similarity value between the two nonadjacent blocks is 0 as well, so these two blocks will not be clustered.

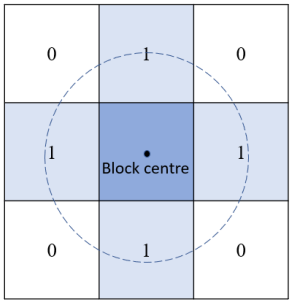



Figure 3-14. Blocks adjacency relationship

$$Adj_{ij} = \begin{cases} 1, \text{ if } \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le block\ width \\ 0, \text{ if } \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} > block\ width \end{cases} \tag{3.14}$$

### 3.3.2 Block Clustering

The similarity value of each pair of blocks is stored in a similarity matrix. Because the block's selection order is not a factor for the similarity value, and a specific block cannot merge with itself, the similarity matrix is symmetric with zero diagonal values. In the beginning, every block is considered as an individual cluster, then the two blocks with the maximum similarity are merged into a cluster. After the merging, the similarity values are updated for each of the merged blocks to decide the cluster's similarity values to each of the other clusters.

An example of the clustering process is shown in Figure 3-15. the similarity matrix is a 7×7 pairwise matrix with seven blocks. First, the maximum similarity value is identified, which is 48 between blocks 1 and 3. After merging, each associated row and column's similarity value is updated, so these two blocks have the same similarity values to the other blocks. In this example, the new cluster's similarity values are replaced by the smaller values between blocks 1 and 3. This value update approach is called complete linkage, in which the similarity between clusters is measured by the most dissimilar objects in the two clusters (Hartigan 1985). The complete linkage can generate compact clusters that are evenly distributed and with similar diameters. The maximum similarity value is set to zero, so these two blocks cannot be merged again. That is the first clustering iteration. In the next iteration, the next maximum similarity is identified (between blocks 4 and 7), and the clustering process repeats until certain criteria are met.

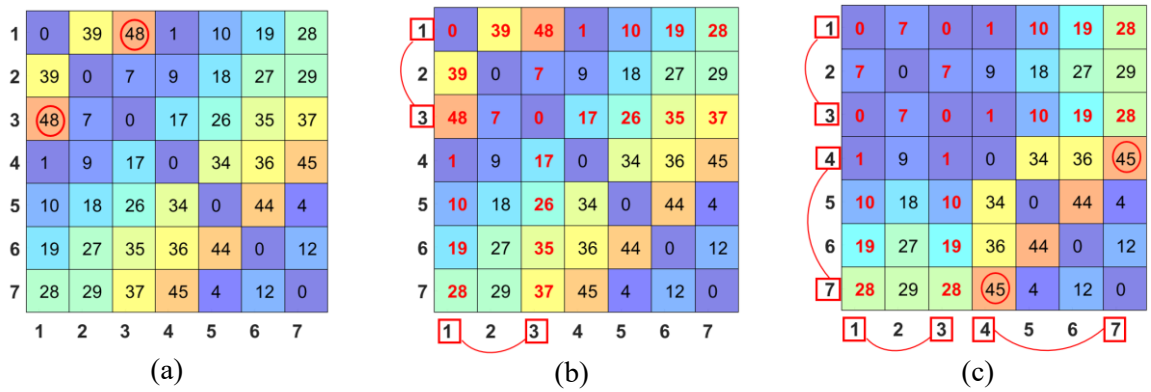



Figure 3-15. The first clustering iteration in the similarity matrix: (a) identify the maximum similarity value; (b) merge the corresponding blocks; (c) update the similarity values and identify the next maximum value.

The maximum cluster size can be specified in the hierarchical clustering algorithm: once a cluster size increases up to the maximum size, its similarity values are set to 0 and no longer merge to the other clusters, which guarantees all clusters' size is not greater than the upper bound. However, because this algorithm follows a bottom-up aggregation pattern, it cannot control the clusters' minimum size. In fact, some clusters may only contain one or a few blocks while their adjacent clusters already reach the maximum size. Additionally, the hierarchical clustering algorithm cannot undo any previous steps, and the results are sensitive to noise and outliers, which makes the results highly unpredictable, and no apriori information can be incorporated in the clustering process (Weintraub et al. 2008).

On the other hand, cluster size is a crucial factor for mathematical framework at the cluster level. Each cluster is a mining unit that is considered equally in the decision variables of the mathematical models. These small-sized clusters can disturb the precedence arcs between adjacent clusters, making the mathematical models inaccurate or even infeasible. On the contrary, a concentrated cluster size distribution can secure the production rate, generate robust cluster precedence relationships, and maintain a stable slope angle.

Therefore, a modified algorithm is proposed to generate more concentrated-sized clusters by applying a multiplier during the clustering and a merging post process. In the modified algorithm, for those clusters smaller than a threshold in every clustering iteration, their similarity values are magnified by a multiplier greater than one; thus, the possibility of merging for these

clustering small-sized clusters is increased. During the beginning iterations, because no clusters exceed the size threshold, the multiplier is applied to all similarity values. Since the similarity values are the qualitative comparison between different clusters and the multiplier is applied to all similarity values within a specific cluster, it will not change the other clusters' similarity rank to this cluster. At the end of the clustering, any cluster that is smaller than a lower bound will be merged into one of its adjacent clusters with the minimum size. This post-processing step can further eliminate small-sized outliers. The flowchart of this modified clustering algorithm is shown in Figure 3-16.

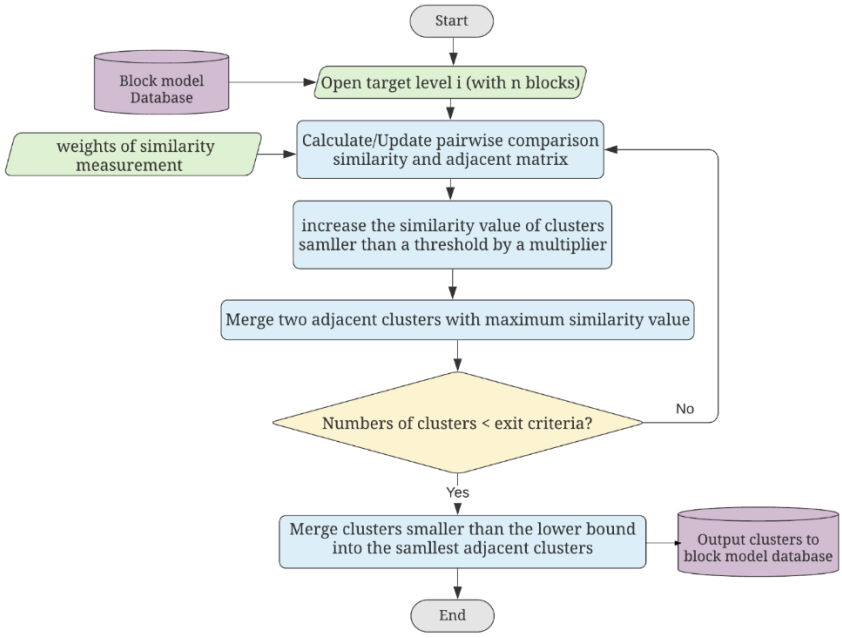



Figure 3-16. The flowchart of the modified clustering algorithm for cluster size control

A trial-and-error approach is developed for tuning the value of the multiplier based on cluster size. The objective is to find a multiplier's value that gives the minimum variance of the cluster size. The attempt value starts from 1 but also not too large to make the similarity value infinite over the iterations. In practice, the attempt values are between 1 and 2 with a reasonable resolution. Each attempt is a repetition of the modified clustering process. Figure 3-17 shows a multiplier tuning practice, and the attempt values are from 1 to 2 with a step of 0.005. It is evident that the size variance is highly irregular with respect to the multiplier's value, and the lowest variance at 4.37 is obtained when the multiplier is equal to 1.865. are tailored specifically for a class of problems or even for a specific instance of a problem.

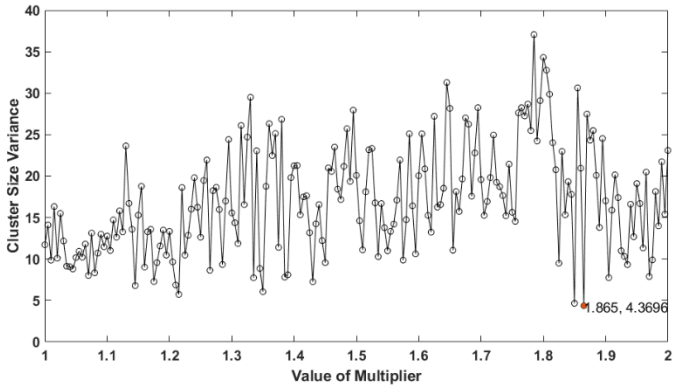
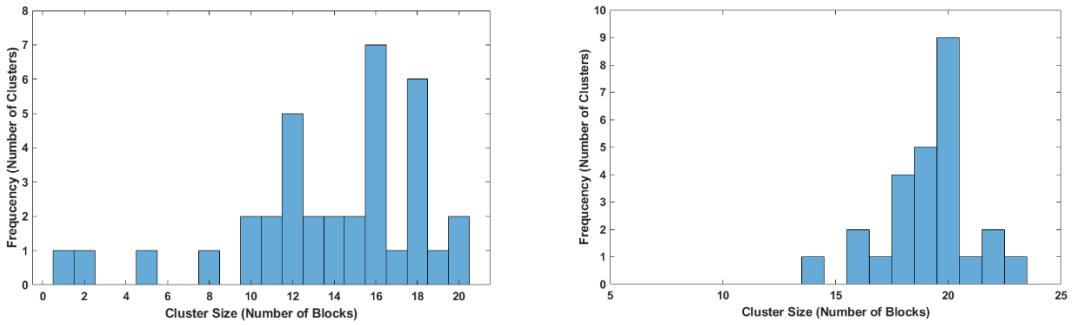



Figure 3-17. The variance of cluster size by different multiplier's values

The two histograms of cluster size for the original and the modified algorithm practice are shown in Figure 3-18, respectively. In the original clustering algorithm, the maximum cluster size is the only parameter for the size control. Figure 1.14(a) shows that the cluster size is dispersive, with an average size of 15.8 blocks, and the variance is 20.54. Whereas the modified algorithm can generate clusters whose size distribution is concentrated in a small range, as Figure 1.14(b) shows. In this case, the average cluster size is 19.5, and the variance is 4.37.




(a) (b)

Figure 3-18. Histogram of cluster size: (a) the original clustering algorithm; (b) the modified clustering algorithm

Figure 3-19 displays the results generated by the modified clustering algorithm. The target cluster size is 20 blocks. Each cluster's centroid is labeled; ore blocks are marked with white circles, and the big black dot shows the level's conveyor feeding spot. In general, the results reveal that the modified clustering approach can generate clusters with more concentrated sizes while retaining the similarity features within each cluster.

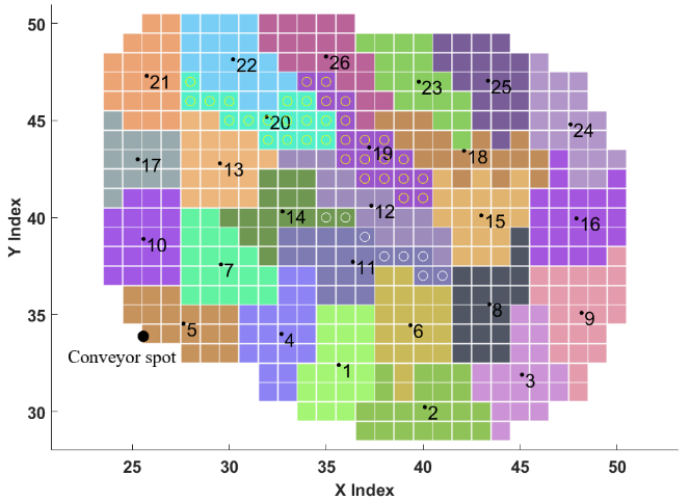



Figure 3-19. Clustering result for a level by applying the modified clustering algorithm

## 3.4 Clusters Precedence Relationships

The precedence relation between blocks is straightforward, as all the blocks are arranged in the orthogonal XYZ grid system and typically have a uniform cubic shape. However, because the mathematical models are solved at the cluster level, any precedence must be defined between clusters. Because the clusters do not have regular shapes and are not in a regular grid, their extraction precedence is not explicit. Moreover, the specific mining direction required for the IPCC system introduces additional precedence with a level. In this section, the precedence relationships are divided into two types (i) horizontal precedence at the same level and (ii) vertical precedence between two levels.

### 3.4.1 Horizontal precedence

The horizontal precedence is a result of the mining direction. In this research, it is assumed that the conveyor spot is fixed on one side of the level throughout the mine life. The materials mined from a certain level are sent toward this side. At a specific level, the precedent clusters must be extracted in advance to make the target cluster available for mining, and the mining operation is expanded away from the conveyor spot. Additionally, the drilling and blasting progression, equipment mobility and shovel access should be considered to limit unnecessary equipment moves and make a smooth planning transition. As a result, continuous mining from

one cluster to its neighboring clusters is preferred. The horizontal precedence clusters should be:

- adjacent to the target cluster;
- their centroid point should be closer than the centre point of the target cluster to the conveyor spot.

The coordinate centroid of a specific cluster is calculated from the average x- and y-coordinate of the blocks inside the cluster. Figure 3-20 illustrates the horizontal precedence among clusters based on the mining direction. The target cluster numbered 13 has two direct horizontal precedent clusters numbered 12 and 14, where both clusters' centres are closer to the conveyor spot than the target one. These two clusters should be mined first to make cluster 13 accessible by the mining equipment.

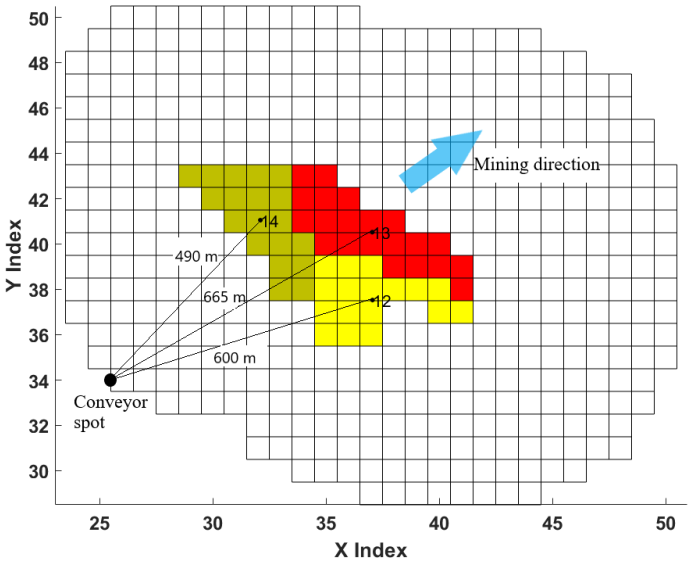



Figure 3-20. Schematic view of the direct horizontal precedence

### 3.4.2 Vertical precedence

The vertical precedence is challenging to determine at the cluster level. To be specific, the shapes of the generated clusters are irregular. The clustering process is independent level by level, so the relative location of clusters between two adjacent levels is unspecified. Badiozamani and Askari-Nasab ( define the cluster precedence relation from the corresponding block precedence inside each cluster. In this method, one cluster is precedent to another if there exists a block precedence relation between their two sets of blocks. However, this method can

overestimate the number of precedence relations since each cluster contains a number of blocks. Those unnecessary precedence constraints can result in a significantly lower pit slope angle than the requirement and increase the mathematical models' solution time. In this section, the cluster vertical precedence relation is determined by multiple adjustable criteria based on the block precedence, which is adaptable to different pit slope requirements and decreases the number of precedence arcs. The method is demonstrated in two steps as follows:

***Step 1:*** precedent blocks of each block within the target cluster are defined based on the classic precedence rule. The predecessors can be found from the upper-level blocks based on their coordinates or indices (see Figure 3-21).

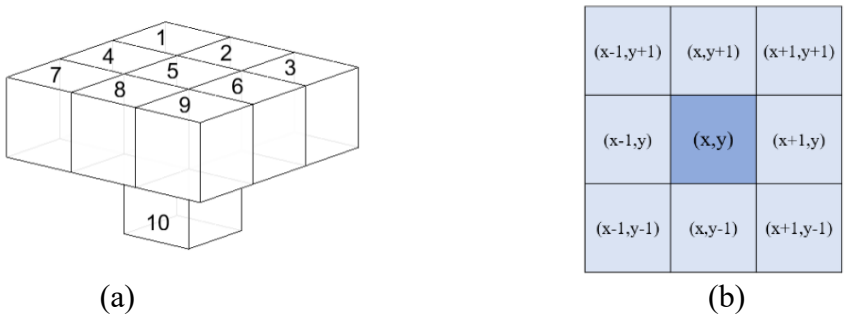



Figure 3-21. Nine predecessors' pattern to control pit slope: (a) to mine the block numbered 10, its nine predecessors at the upper level should be mined first; (b) the relative indices of nine predecessors in the 2D plan.

***Step 2:*** along the blocks' border, a boundary is created to envelop all the target cluster's precedent blocks. Figure 11 shows the target cluster's example outlined by a bold black line, and clusters 11, 12, 14, 16, and 17 are located at the upper level. The red line is the precedent blocks' boundary in the upper level. The materials from the upper level and inside the red boundary line should be extracted before the target cluster. Based on the red boundary line, each precedent cluster can be identified if any of the following criteria are satisfied:

a) Cluster's centroid is inside the red boundary (clusters 14, 16, and 17)
b) Cluster is partly inside the boundary, and the conveyor is closer to the cluster's centroid than to the centroid of the target cluster (clusters 11, 12, and 17)
c) The portion of the cluster inside the red boundary is greater than 40%.

All these five clusters (11, 12, 14, 16, and 17) are the target cluster's vertical predecessors, and they are one level upward. The flowchart of the determination process is shown in Figure 3-23.

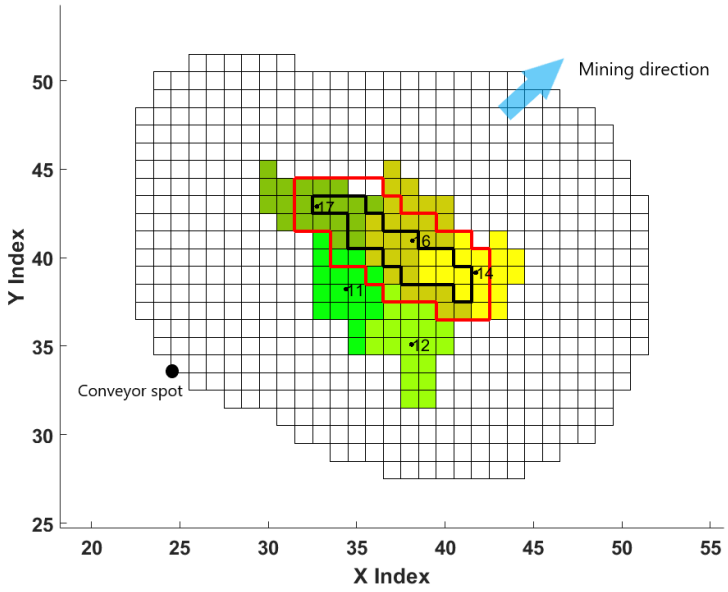



Figure 3-22. Vertical precedent clusters (clusters 11,12,14,16 and 17) for the target cluster at the lower level (in bold black outline)

In practice, the criteria of determining the precedent clusters are adjustable based on the pit safety slope and the cluster size. For example, in criterion 3, the minimum portion of a precedent cluster inside the boundary can be adjusted as needed; also, because most precedent clusters generated from criterion 1 and 3 are overlapped, one criterion can be deleted to increase the processing time.

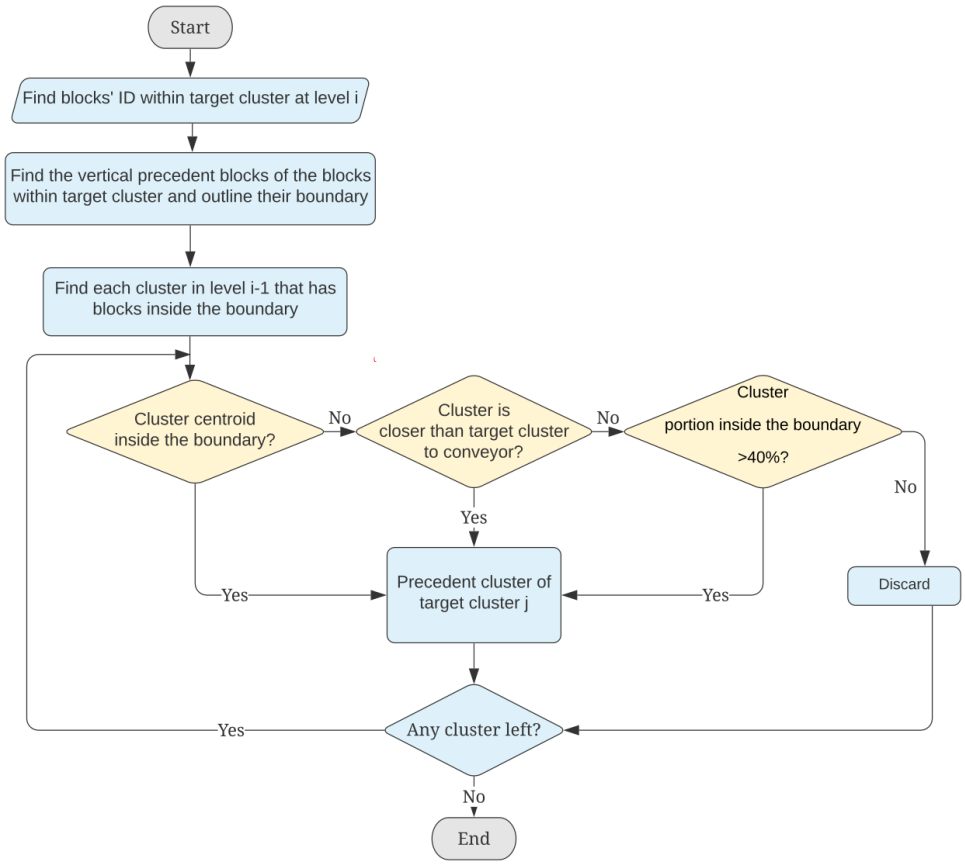



Figure 3-23. Flowchart for determining the precedent clusters

## 3.5 Material handling cost

After drilling and blasting, the loose material is excavated by shovel and then loaded to the trucks. Trucks are adaptable to transport run-of-mine rock from the working face to the crusher station on a different bench, whereas the conveyor belt can only convey crushed material. The crusher station plays as a transfer point that connects the two transportation methods: in the first segment, a small truck fleet is hauling material from the loading point to the crusher, and from where the conveyor belt system sends the crushed material as the second segment to the pit exit. If the crusher station is closer to the working face, the trucking distance will be shorter, and correspondingly, the conveying portion will increase. On the other hand, because conveying is generally cheaper than trucking, it is favourable to increase the portion of conveying by locating the crusher close to the working face to save the total material handling costs.

By aggregating the entire block model within the UPL, the centroid's coordinates and tonnage of each cluster have been identified. The material handling costs are calculated on a cluster basis; therefore, the materials within a specific cluster have the same unit handling cost.

For material handling cost estimation, the whole in-pit transportation process is divided into three parts: vertical trucking, horizontal trucking and conveying. A schematic diagram is illustrated in Figure 3-24. The transportation distance of each part is measured first, then the associated handling costs are estimated based on the distances of each part, respectively.

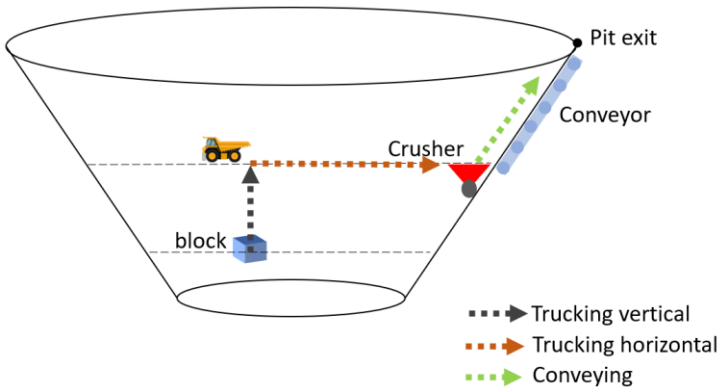



Figure 3-24. Diagram of three parts of material handling cost (showing in dash line arrows: trucking vertical, trucking horizontal, and conveying)

Although the truck hauling road is a continuous ramp with curves and switchbacks, the trucking part is divided into horizontal and vertical components for the cost calculation. The division is because trucks consume considerably more energy in hauling material to a different elevation than the same distance of horizontal hauling. Generally, the truck fleet should travel significantly longer to reach a different level than hauling within the same level. That is because the maximum haul road gradient is limited to 10% or a slope of 5.7˚ to the horizontal plane. For example, if the trucks travel across a bench with a height of 15 m, the minimize ramp distance should be 151 m. Meanwhile, the truck should overcome the grade resistance in the uphill travel and the frequent braking during downhill travel. Moreover, due to the other design factors such as stopping distance, curves and switchbacks, the actual ramp length can be even longer. As a result, the truck hauling costs are sensitive to the elevation difference, and the vertical and horizontal directions should be considered separately for the cost estimation. The horizontal trucking distance for a specific cluster is the Euclidean distance from the cluster's centroid to

the crusher in the horizontal projection plane; the vertical trucking distance is measured in the difference in elevation between a specific cluster and the crusher level.

Conveyors are the main components for the material elevating in the IPCC system. Once the material is comminuted in the in-pit crusher station, it is transferred through the conveyor belt to exit the pit. The conveyor system usually comprises several conveyor flights, and each flight may have a different slope angle and curvature; therefore, the vertical lifting height from the crusher level to the surface is measured to simplify the conveying distance.

Equation (3.16) calculates the material handling cost per ton (denoted by $F_{ij}$) for a specific cluster *i* with respect to the crusher location *j*. It contains the cost of horizontal trucking, vertical trucking, and vertical conveying. Each part of the cost is the product of the corresponding unit cost and its transportation distance. Where $DH_{ij}$ is the horizontal trucking distance from the cluster *i* to the crusher location *j*; $CT_H$ is the unit horizontal trucking cost; similarly, $DV_i^T$ and $DV_i^C$ are the vertical trucking distance from cluster *i* to the crusher location *j* and vertical conveying distance from crusher location *j* to the surface, respectively; $CT$ and $CC$ are their corresponding unit vertical lifting cost.

$$F_{ij} = \underbrace{DH_{ij} \times CT_H}_{\text{horizontal trucking}} + \underbrace{DV_{ij}^T \times CT_V}_{\text{vertical trucking}} + \underbrace{DV_j^C \times CC}_{\text{vertical conveying}} \tag{3.15}$$

Since the material handling cost happens every period along with the cluster extraction, each year's discount factor is applied to this cost. In this research, it is assumed the material handling costs for ore and waste are the same. The discounted cost for cluster *i* in period t is calculated by Equation (3.16). Where $Ton_i^o$ and $Ton_i^w$ is the ore and waste tonnage of cluster *i*, respectively; r is the yearly discount rate.

$$f_{ijt} = F_{ij} \times (Ton_i^o + Ton_i^w) \times \frac{1}{(1+r)^t} \tag{3.16}$$

It should be mentioned that in mixed-integer programming, a cluster may be mined during several different periods. In this case, an associated decision variable, which denotes the extraction portion for that cluster in a certain period, is applied to calculate the partial material handling cost.

## 3.6 Summary and Conclusion

This chapter focuses on a framework that generates a series of candidate high-angle conveyor lines along the final pit wall. The conveyor system is fixed along one side of the final pit wall throughout the mine life. First, a rotation method is applied to identify the conveyor side wall for each level in different orientations. Next, the least square algorithm is developed to find straight conveyor lines under various rotations around the UPL. The crusher is located on the conveyor line and can be relocated to different levels.

Because the mining operation cannot be developed under the conveyor line, the fixed conveyor system can reduce mining flexibility and introduce additional mining direction. A clustering method integrated with mining direction and cluster size control is applied, which aims to reduce the block model's size. The mining precedence between clusters is determined based on a proposed decision-making step.

An approach for estimating the material handling costs is proposed in this chapter. The excavated material should be trucked to a crusher station located on the conveyor line and then transferred by the conveyor system to exit the pit, where the crusher connects the material flow between trucks and conveyors. Because of the different operating costs between these two methods, each method's handling distance should be clarified based on the locations of a specific cluster and the crusher.

Moreover, the installment for the conventional conveyor is discussed. Because the pit slope, typically 40˚, is steeper than the maximum conventional conveying angle of 20˚, dedicated ramp slots are designed to accommodate these types of conveyors. However, the ramp slot requires additional waste excavation, and the conveyor route is more extended than HAC.

# CHAPTER 4

# MATHEMATICAL MODELS

## 4.1 Introduction

Long-term planning is a large-scale optimization problem made for several years to the whole mine life. This production scheduling problem aims to find the optimum block extraction sequence on a large scale that maximum the net present value (NPV) while obeying a series of technical and economic constraints. However, other objectives such as minimizing the operating costs are also considered.

In this chapter, a two-step linear programming model (LP) for conventional conveyors, a two-step LP model for high-angle conveyors (HAC), and a binary-integer linear programming (BILP) model are proposed respectively. An overview framework of the proposed mathematical models is shown in Figure 4-1.

The first and the second models consider the conventional conveyor and HAC situations, respectively. Both models contain a two-step framework, which separately solves the production scheduling and the crusher location-relocation plans. The production scheduling step is solved by mixed-integer linear programming formulations, which contain continuous variables so that a specific cluster can be partially mined within a period. The second step is the facility location solved based on the production scheduling results obtained from the previous step. In contrast to the HAC situation of the second model, the slot's extra stripping tonnage is considered in the first model for the conventional conveyor.

The BILP model aims to maximize the NPV while minimizing the transportation and crusher relocation costs of the mining operation, which is a combination of production scheduling and facility location problems. Therefore, this model can determine the extraction sequence and the crusher station location-relocation plan simultaneously. All the decision variables in the BILP model are 0-1 binary. These variables control the production scheduling

decision, mining precedence, material handling decision, the crushing unit location and relocation decision. The objective function and all the constraints are linear equations.

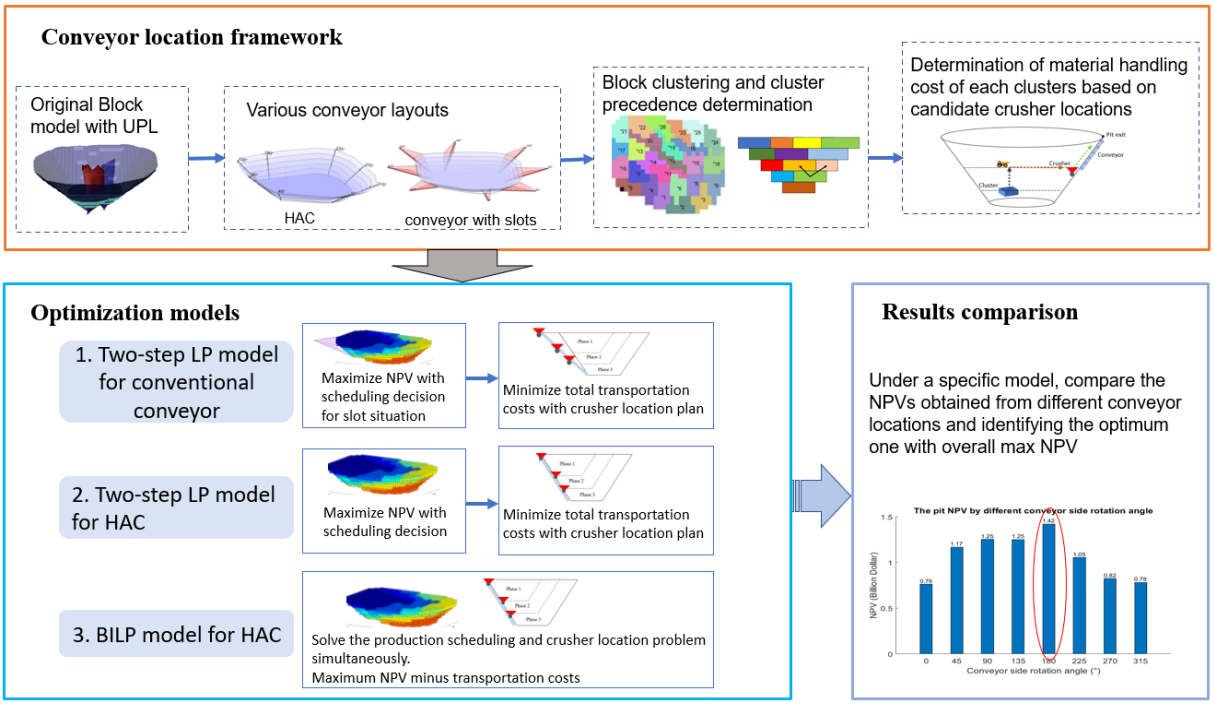



Figure 4-1. The overview framework of the proposed mathematical models

All three models are solved by a commercial optimization solver IBM ILOG CPLEX (IBM 2011). For the mathematical model in this research, MATLAB (MATLAB 2018) is used as the numerical modelling platform and CPLEX toolbox as the optimization solver. The formulations in these models are transformed to the matrix structures that the CPLEX engine can solve. The formulation structure of the BILP model, as a complete model, is explained in this chapter.

## 4.2 Models Configuration

### 4.2.1 Models Assumptions

Due to the considerable complication of real-world problems in their entirety, every mathematical model is subject to assumptions and simplifications. These assumptions should be clarified to avoid obtaining a result that is misleading and difficult to assess. The major assumptions of the mathematical models include:

1. The deposit's geostatistical information is known and stored in a block model, where each block is a 3D cube with its unique coordinate and indices. Each block is assigned to a set of deterministic attributes, including element grade, rock type, density. Other economic and technical parameters are also predetermined as inputs model.
2. This framework is proposed based on the semi-mobile IPCC systems' situation. The economic and technical comparison among semi-mobile IPCC systems, conventional truck-and-shovel system and other types of IPCC systems are not considered in this research. The in-pit transportation method should be determined before the application of this framework.
3. This research assumes the final pit wall is stable under a specific acceptable slope angle; thus, the conveyor wall can be rotated to any angles along the pit limit. However, based on the geological and geotechnical conditions in the real cases, the candidate conveyor locations are only restricted to stable pit zone.
4. The UPL of the block model is determined before scheduling. Because the conveyor line is installed along the final pit wall, the slope angle of the UPL should be stable at different benches.
5. The models are solved at the cluster level. Although in the MILP models, the clusters can be partially removed in one period since the model contains continuous decision variables, these models cannot obtain the specific cluster's production scheduling plan. The scheduling resolution generated from the model is based on cluster size. Additionally, this model does not guide short-term planning.
6. The conveyor belts are not intended for relocation once conveyor flights are installed. However, a new conveyor flight can be installed to connect the crusher station that is relocated lower level. The entire conveyor system is still in a straight line after its extension.
7. The conveyor systems are used to transfer both ore and waste. Therefore, parallel conveyor belts should be installed along the same conveyor line to transfer different material types and avoid the mixture of ore and waste. The crusher station should be configured in such a way so that it can handle all types of material without ore dilution.
8. The crusher station is installed on the conveyor line, and it can only be relocated downward with the general mining development. The crusher station relocation costs

are massive, and the conveyor system should also be reinstalled, resulting in considerable operational downtime. It is unreasonable to relocate the crusher opposite to the general mining direction.

9. The upfront investment, salvage, and maintenance costs are not explicitly counted in the mathematical models. However, they can be directly implemented in the model by deducting the relevant cost from the objective function.
10. For the conventional conveyor implementation, a ramp slot is extracted at the final pit wall to reduce the conveyor line's slope angle. The slot is divided into slices based on the pit level. The slot slice of a specific level should be extracted before mining material from that level, and the crusher station is located inside the slot.

### 4.2.2 Notation

All the notations used to formulate the proposed models are classified into sets, indices, parameters, and decision variables, as Table 4-1 presents.

Table 4-1. Overview of the notations used in the three mathamtical models in this study

| **Sets** | |
|---|---|
| $\mathbb{N}$ | Set of clusters in the model |
| $\mathbb{B}_i^w$ | Set of waste blocks cluster $i$ |
| $\mathbb{B}_i^o$ | Set of ore blocks cluster $i$ |
| $\mathbb{P}_i$ | For each cluster $i$, there is a set of immediate predecessors that must be extracted before extraction of cluster $i$ |
| $\mathbb{L}_j$ | Set of all clusters in level $j$ |
| **Indices** | |
| $i \in \{1,...,I\}$ | Index for clusters |
| $j \in \{1,...,J\}$ | Index for pit levels |
| $t \in \{1,...,T\}$ | Index for scheduling periods |
| **Parameters** | |
| $I$ | Total number of clusters |
| $J$ | Total number of levels |

| | |
|---|---|
| $T$ | Number of scheduling periods |
| $r$ | Discount rate |
| $CLEV_i$ | The undiscounted economic value of cluster *i* |
| $Ton_b$ | The tonnage of block *b* |
| $Ton_i^w$ | The total tonnage of waste in cluster *i* |
| $Ton_i^o$ | The total tonnage of ore material in cluster *i* |
| $Ton_j^s$ | The extra tonnage of extracting ramp slot in level *j* (as waste) |
| $g_b$ | Grade of ore block *b* |
| $g_i^o$ | The average grade of ore material in cluster *i* |
| $Pc_t$ | Price per unit of product sold in period *t* |
| $s$ | Selling cost per unit of product |
| $m^w$ | Cost of mining a tonne of waste |
| $m^o$ | Cost of mining a tonne of ore |
| $m^s$ | Cost of mining a tonne of waste in the ramp slot (including transportation costs) |
| $c^o$ | Cost of processing a tonne of ore |
| R | The recovery rate for ore material |
| $f_{ijt}$ | Discounted transportation cost for cluster i sent to crusher *j* in period *t* |
| $F_{ij}$ | Undiscounted transportation cost for a unit weight of material in cluster *i* sent to crusher *j* in period *t* |
| $fc_{jt}$ | Discounted transportation cost for all materials sent to crusher *j* in period *t* |
| $X_{i,t}$ | Vector denotes the portion of cluster *i* mined in period *t* (the result of decision variable $x_{i,t}$ generated in scheduling MILP model) |
| $S_{j,t}$ | Vector denotes if the slot slice in level *j* is mined in period *t* (the result of decision variable $s_{i,t}$ generated in scheduling MILP model) |
| $c_t$ | Discounted crusher relocation cost at period *t* |
| $n$ | The minimum period interval for crusher relocation |

| | |
|---|---|
| $DH_i$ | The horizontal distance from the centroid of cluster *i* to the crusher station *j* |
| $DV_{ij}^{T}$ | The vertical distance from cluster *i* to the crusher station *j* (truck hauling) |
| $DV_{j}^{C}$ | The vertical distance from crusher station *j* to the pit exit (conveying) |
| $CT_H$ | Unit truck horizontal hauling cost per tonne per meter |
| $CT_V$ | Unit truck vertical hauling cost per tonne per meter |
| $CC$ | Unit conveyor vertical lifting cost per tonne per meter |
| $\overline{M}^t$ | Upper bound of mining capacity in period *t* |
| $\underline{M}^t$ | Lower bound of mining capacity in period *t* |
| $\overline{P}^t$ | Upper bound of processing capacity in period *t* |
| $\underline{P}^t$ | Lower bound of processing capacity in period *t* |
| $\overline{G}^t$ | Upper bound on allowable average grade of processed ore in period *t* |
| $\underline{G}^t$ | Lower bound on allowable average grade of processed ore in period *t* |
| **Decision Variables** | |
| $x_{i,t} \in \{0,1\}$ | Binary variable denoting the if cluster *i* mined in period *t* |
| $x_{i,t}^{c} \in [0,1]$ | Continuous variable denoting the portion of cluster *i* mined in period *t* |
| $x'_{i,t} \in \{0,1\}$ | Binary variables equal to 1 if the precedent clusters are all cleared for cluster *i*; 0 otherwise |
| $x''_{ijt} \in \{0,1\}$ | Binary variables denoting if cluster *i* is crushed at level *j* in period *t* |
| $y_{j,t} \in \{0,1\}$ | Binary variables equal to 1 if the crusher is in level *j* in period *t*; 0 otherwise |
| $z_{j,t} \in \{0,1\}$ | Binary variables equal to 1 if the crusher is relocated in level *j* in period *t*; 0 otherwise |

## 4.3 Two-Step LP model for Conventional Conveyor

This LP model considers a situation that the conventional conveyor belts are accommodated in a ramp slot. This conveyor ramp slot requires additional stripping at each level's conveyor spot. Extra mining tonnage and slot extraction decisions should be considered.

The slot is divided into slices based on the pit level, and each slice can be extracted in different periods. The slot design and its tonnage calculation are explained in Chapter 3.

In this model, the production scheduling and crusher location-relocation decisions are solved using two linear programming formulations, respectively. The first step is a MILP formulation that aims to maximize the NPV and obtain associated the cluster extraction sequence. The production scheduling decision variables are continuous numbers between 0 to 1, which means the clusters can be partially extracted in a period. The second step is a binary integer programming formulation deciding on the crusher location-relocation plan that gives the minimum total transportation costs. The second step is based on the production scheduling results generated from the first step: all the materials mined in a specific period *t* are sent to the crushing station at a specific level *j* during that period, and the crushing station's location is subject to the active mining face.

### 4.3.1 Production scheduling MILP formulation

In the MILP formulation, the material transportation costs are not considered. Production scheduling plan is generated based on the CLEV of each cluster and a set of constraints, including mining precedence, capacities and slot extraction. At each level, the slot slice is mined first before the extraction of that level, and the additional slot tonnage should be considered in the mining capacity. The production scheduling obtained by this formulation is considered the second step's input parameter.

***Objective function***

The objective function (Equation (4.1)) contains two items. The first one is the summation of the discounted CLEV of each period, and the second term is the additional mining costs for the ramp slot extraction. Equation (4.3) calculates the undiscounted CLEV.

$$\text{Maximize} \quad \sum_{t=1}^{T}\sum_{i=1}^{I}\left\lfloor \frac{CLEV_i}{(1+r)^t} \right\rfloor \times x_{i,t}^{c} - \sum_{t=1}^{T}\frac{1}{(1+r)^t} \times Ton_j^s \times m^s \times s_{j,t} \tag{4.1}$$

$$CLEV_i = \underbrace{\left(Ton_i^o \times g_i^o \times R\right) \times \left(Pc_t - s\right)}_{\text{revenue}} - \underbrace{\left(Ton_i^o \times m^o + Ton_i^w \times m^w\right)}_{\text{mining cost}} - \underbrace{Ton_i^o \times c^o}_{\text{processing cost}} \tag{4.2}$$

$$g_i^o = \frac{\sum_{b \in \mathbb{B}_i^o} Ton_b \times g_b}{\sum_{b \in \mathbb{B}_i^o} Ton_b} \tag{4.3}$$

Equations (4.2) and (4.3) show the concept of cluster disaggregation. Although the mathematical model is built at the cluster level, the blocks defined as ore or waste within a specific cluster are considered separately in the formulations. Since ore should be processed and may have a different mining cost than waste, the cost differences, process capacity and grade blending constraints are specialized for ore. This disaggregation process can increase the model's resolution at the cluster level without compromising the model's computation time. Each cluster's average grade is based on the ore blocks in each cluster, calculated by Equation (4.3).

***Constraints***

- *Mining capacity*

The mining capacity constraint is the bottleneck capacity among the truck-shovel fleet, crusher, and conveying system. It is measured by the total tonnage of material they can move in each period, including the slot's additional excavation tonnage. This limit can vary from period to period. Equation (4.4) ensures that the total tonnage of material extracted from active clusters and the ramp slot in each period is within an acceptable range that allows flexibility for potential operational variations.

$$\underline{M}^t \leq \sum_{i=1}^{I} Ton_i^r \times x_{i,t}^c + \sum_{j=1}^{J} Ton_j^s \times s_{j,t} \leq \overline{M}^t \qquad \forall t \in \{1,...,T\} \tag{4.4}$$

- *Processing capacity*

The structure of processing capacity constraint is similar to mining capacity constraint. This capacity is mainly determined by the processing plant's throughput, and only ore tonnage is considered. Equation (4.5) certifies that the amount of ore mined in each period is within the processing plant's acceptable range.

$$\underline{P}^t \leq \sum_{i=1}^{I} Ton_i^o \times x_{i,t}^c \leq \overline{P}^t \qquad \forall t \in \{1,...,T\} \tag{4.5}$$

- *Blending grade*

Grade blending is an important constraint on the processing plant. It controls the grade of material that the processing plant for recovery efficiency can process. Equation (4.6) and (4.7) force the mining system to achieve the desired grade between the lower bound and upper bound, respectively.

$$\sum_{i=1}^{I}\left(Ton_i^o \times \left(\underline{G}_t - g_i\right)\right) \times x_{i,t}^c \le 0 \qquad \forall t \in \{1,...,T\} \tag{4.6}$$

$$\sum_{i=1}^{I}\left(Ton_i^o \times \left(g_i - \overline{G}_t\right)\right) \times x_{i,t}^c \le 0 \qquad \forall t \in \{1,...,T\} \tag{4.7}$$

- *Reserves*

Equation (4.8) is the reserve constraint, which ensures each cluster within the UPL can be at most mined once. The predetermined UPL is based on the traditional truck-and-shovel system. It also gives the model freedom to decide whether a cluster is mined or not, as some lower value block may not contribute to the NPV.

$$\sum_{t=1}^{T} x_{i,t}^c \le 1 \qquad \forall i \in \{1,...,I\} \tag{4.8}$$

- *Precedence*

Equations (4.9) and (4.10) are the precedence constraints, which contain both horizontal and vertical precedent clusters. These constraints reflect the pit slope angle and the mining direction. Equation (4.10) ensures cluster can only be partially or wholly extracted if all its precedent clusters are entirely removed.

$$\left|\mathbb{P}_i\right| \times x'_{i,t} - \sum_{i \in \mathbb{P}_i} \sum_{t'=1}^{t} x_{i,t'}^c \le 0 \qquad \forall i \in \{1,...,I\},\ t \in \{1,...,T\} \tag{4.9}$$

$$x_{i,t}^c - x'_{i,t} \le 0 \qquad \forall i \in \{1,...,I\},\ t \in \{1,...,T\} \tag{4.10}$$

- *Ramp Slot*

For a specific level, the slice of the ramp slot should be mined before extending the conveyor to that level. Equation (4.11) guarantees the ramp slot slice of level j must be extracted

before mining any clusters in that level. Equation (4.12) enforces the ramp slot slice for each level can only be extracted once.

$$\sum_{i\in\mathbb{L}_j} x_{i,t}^{c} - \sum_{t=1}^{T} \left|\mathbb{L}_j\right| \times s_{j,t} \le 0 \qquad \forall j \in \{1,...,J\},\ t \in \{1,...,T\} \tag{4.11}$$

$$\sum_{t=1}^{T} s_{j,t} = 1 \qquad \forall j \in \{1,...,J\},\ t \in \{1,...,T\} \tag{4.12}$$

In order to input the MILP model into the CPLEX solver, all the formulations in this section should be transformed to variable vectors, several coefficient matrices, and bound vectors. The constraint matrix of MILP is similar to the BILP model, which will be presented in Subsection 4.5.2. The results from this step will be used to minimize the material handling and crusher relocation costs of the project in the next formulation.

### 4.3.2 Crusher Location-Relocation Formulation

This step is a facility location problem that aims to optimize the crusher location based on the results from the first step. The formulation contains two sets of conjoint binary variables that denote the crusher location and relocation, respectively. The values of production scheduling variables $x_{i,t}^{c}$ and slot extraction decision variables $s_{j,t}$ determined in the first step are used as the input parameters, denoted by $X_{i,t}^{c}$ and $S_{j,t}$, respectively.

***Objective function***

The objective function (4.30) is defined as minimizing the total transportation costs. The first part of the function is the material handling costs, and the second part is the relocation costs of the IPCC. The coefficient $fc_{jt}$ represents the total discounted material handling costs of all the material extracted in period *t* send to the crusher location at level *j*, where $X_{i,t}^{c}$ is production scheduling results obtained by the formulation presented in Subsection 4.4.1. The value of $fc_{jt}$ is calculated by Equation (4.14).

***Objective function***

$$\text{Minimize} \quad \sum_{j}^{J}\sum_{t}^{T} fc_{jt} \times y_{jt} + \sum_{j}^{J} c_t \times z_{jt} \tag{4.13}$$

Where:

$$fc_{jt} = \sum_{i=1}^{I} F_{ij} \times X_{i,t}^{c} \times (Ton_i^o + Ton_i^w) \times \frac{1}{(1+r)^t} \tag{4.14}$$

***Constraints***

$$\sum_{j=1}^{J} y_{j,t} = 1 \qquad \forall\, t \in \{1,...,T\} \tag{4.15}$$

$$\sum_{i}^{J} y_{i,t-1} \geq \sum_{i}^{J} y_{i,t} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \tag{4.16}$$

$$z_{j,t} \geq y_{j,t} - y_{j,(t-1)} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \tag{4.17}$$

$$z_{j,t} \leq y_{j,t} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \tag{4.18}$$

$$z_{j,t} = y_{j,t} \qquad \forall j \in \{1,...,J\}, t = 1 \tag{4.19}$$

$$\sum_{t=1}^{T} y_{i,t} - n \times \sum_{t=1}^{T} z_{i,t} \geq 0 \qquad \forall j \in \{1,...,J\} \tag{4.20}$$

$$y_{j,t} \leq \sum_{t=1}^{T} S_{j,t} \qquad \forall j \in \{1,...,J\}, t \in \{1,...,T\} \tag{4.21}$$

In those constraints, Equation (4.15) certifies that exactly one crusher station is available in each period. Equation (4.16) ensures that the crusher station can only be relocated to lower levels or remain static in any period. Equations (4.17) - (4.19) set the crusher station's relocation conditions; the relocation variable $z_{jt}$ is equal to 1 only if the crusher station moved to level j from another level in period t. Equation (4.20) controls the minimum frequency of the crusher station's relocation, where the cluster station should stay at a specific level for at least n period

before relocation. Equation (4.21) forces that the ramp slot slice of a specific level should be mined before available for the crusher's relocation to that level.

## 4.4 Two-Step LP Model for HAC

This LP model considers a situation that the HAC is installed directly along the final pit wall. Similar to the model in Section 4.3, the production scheduling and crusher location-relocation decisions are solved by two linear programming formulations, respectively. The first step is the NPV maximization while considering the extra mining cost for the slot; then, based on the obtained production scheduling result, the second step is deciding on the crusher location-relocation plan.

### 4.4.1 Production Scheduling Formulation

This MILP formulation is similar to the one for conventional conveyor in 4.3.1. However, because the HAC can be directly installed along the final pit wall, no ramp slot and its associated constraints are considered in this model.

***Objective function***

The objective function in Equation (4.22) sums the discounted cluster economic value multiply their corresponding scheduling decision variables. The undiscounted cluster economic values (CLEV) in this formulation are the same as the first model calculated by Equation (4.2).

$$\text{Maximize} \quad \sum_{t=1}^{T}\sum_{i=1}^{I}\left\lfloor \frac{CLEV_i}{(1+r)^t} \right\rfloor \times x_{i,t}^{c} \tag{4.22}$$

***Constraints***

$$\underline{M}^t \le \sum_{i=1}^{I} Ton_i^r \times x_{i,t}^{c} \le \overline{M}^t \qquad \forall t \in \{1,...,T\} \tag{4.23}$$

$$\underline{P}^t \le \sum_{i=1}^{I} Ton_i^o \times x_{i,t}^{c} \le \overline{P}^t \qquad \forall t \in \{1,...,T\} \tag{4.24}$$

$$\sum_{i=1}^{I}\left(Ton_i^o \times \left(\underline{G}_t - g_i\right)\right) \times x_{i,t}^{c} \le 0 \qquad \forall t \in \{1,...,T\} \tag{4.25}$$

$$\sum_{i=1}^{I}\left(Ton_i^o \times \left(g_i - \overline{G}_t\right)\right) \times x_{i,t}^c \le 0 \qquad \forall t \in \{1,...,T\} \tag{4.26}$$

$$\sum_{t=1}^{T} x_{i,t}^c \le 1 \qquad \forall i \in \{1,...,I\} \tag{4.27}$$

$$\left|\mathbb{P}_i\right| \times x'_{i,t} - \sum_{i \in \mathbb{P}_i}\sum_{t'=1}^{t} x_{i,t'}^c \le 0 \qquad \forall i \in \{1,...,I\},\ t \in \{1,...,T\} \tag{4.28}$$

$$x_{i,t}^c - x'_{i,t} \le 0 \qquad \forall i \in \{1,...,I\},\ t \in \{1,...,T\} \tag{4.29}$$

Equation (4.23) is the mining capacity constraint which ensures that the total tonnage of material extracted from active clusters is within an acceptable range that allows flexibility for potential operational variations. Equations (4.24) to (4.29) are the same as the corresponding constraints in the previous MILP formulation for conventional conveyors.

The results obtained from this step will be used to minimize the material handling and crusher relocation costs of the project in the next formulation.

### 4.4.2 Crusher Location-Relocation Formulation

Same as the first model, this formulation contains two sets of conjoint binary variables that denote the crusher location and relocation, respectively. The values of production scheduling variables $x_{i,t}^c$ determined in the first step are used as the input parameters, denoted by $X_{i,t}^c$, to calculate the material handling costs.

***Objective function***

$$\text{Minimize} \qquad \sum_{j}^{J}\sum_{t}^{T} fc_{jt} \times y_{jt} + \sum_{j}^{J} c_t \times z_{jt} \tag{4.30}$$

The objective function (4.30) is defined as minimizing the total transportation costs. The first part of the function is the material handling costs, and the second part is the relocation costs of the IPCC. The value of $fc_{jt}$ is calculated by Equation (4.14).

***Constraints***

$$\sum_{j=1}^{J} y_{j,t} = 1 \qquad \forall\, t \in \{1,...,T\} \tag{4.31}$$

$$\sum_{i}^{J} y_{i,t-1} \geq \sum_{i}^{J} y_{i,t} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \tag{4.32}$$

$$z_{j,t} \geq y_{j,t} - y_{j,(t-1)} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \tag{4.33}$$

$$z_{j,t} \leq y_{j,t} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \tag{4.34}$$

$$z_{j,t} = y_{j,t} \qquad \forall j \in \{1,...,J\}, t = 1 \tag{4.35}$$

$$\sum_{t=1}^{T} y_{i,t} - n \times \sum_{t=1}^{T} z_{i,t} \geq 0 \qquad \forall j \in \{1,...,J\} \tag{4.36}$$

In those constraints, Equation (4.31) certifies that exactly one crusher station is available in each period. Equation (4.32) ensures that the crusher station can only be relocated to lower levels or remain static in any period. Equations (4.33) - (4.35) set the crusher station's relocation conditions; the relocation variable $z_{jt}$ is equal to 1 only if the crusher station moved to level j from another level in period t. Equation (4.36) controls the minimum frequency of the crusher station's relocation, where the cluster station should stay at a specific level for at least n period before relocation.

## 4.5 BILP Model for HAC

The BILP model is a combination of production scheduling and facility location problems. It is designed to make the cluster extraction scheduling and the crusher station location-relocation decisions. It aims to maximize the NPV while considering the total transportation costs, including material handling and crusher relocation costs. To maintain this model's linearity, all the decision variables in the formulation are 0-1 binary.

### 4.5.1 BILP Formulation

***Objective function***

The BILP model is designed to simultaneously determine the mining units' extraction period and the crusher station location-relocation plan. The objective function in Equation (4.37) comprises three items: the discounted summation of clusters' economic value (CLEV), material transportation costs, and crusher station allocation and relocation costs.

The first item is the summation of each block's economic value within the same cluster, and the value of each CLEV is calculated by Equation (4.2). The second item calculates the time-dependent transportation costs based on the production scheduling variable $x_{i,t}$ and crusher location variable $y_{j,t}$. These two sets of variables are connected by another set of decision variables $x''_{ijt}$, which are binary variables denoting the material flow from cluster *i* via crusher at level *j* in period *t*. The coefficient $f_{ijt}$, calculated by Equation (4.38), is the discounted transportation costs of the whole cluster *i* via the crusher at level *j* in period *t*. The costs include three parts: the cost of horizontal trucking toward the conveyor side, the additional cost of trucking between different levels, and the conveyor lifting cost. Each part is the product of a specific distance and its corresponding unit cost. The third item calculates the crusher relocation costs.

$$Maximize \quad \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{I}\left\lfloor \frac{CLEV_i}{(1+r)^t} \right\rfloor \times x_{i,t}}_{\text{discounted cluster values}} - \underbrace{\sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{i=1}^{I} x''_{ijt} \times f_{ijt}}_{\text{material handling cost}} - \underbrace{\sum_{t=1}^{T}\sum_{j=1}^{J} c_t \times z_{j,t}}_{\text{relocation costs}} \tag{4.37}$$

Where:

$$f_{ijt} = F_{ij} \times (Ton_i^o + Ton_i^w) \times \frac{1}{(1+r)^t} \tag{4.38}$$

***Constraints***

$$\underline{M}^t \le \sum_{i=1}^{I} Ton_i^r \times x_{i,t} \le \overline{M}^t \qquad \forall t \in \{1,...,T\} \tag{4.39}$$

$$\underline{P}^t \le \sum_{i=1}^{I} Ton_i^o \times x_{i,t} \le \overline{P}^t \qquad \forall t \in \{1,...,T\} \tag{4.40}$$

$$\sum_{i=1}^{I}\left(Ton_i^o \times \left(\underline{G}_t - g_i\right)\right) \times x_{i,t}^c \le 0 \qquad \forall t \in \{1,...,T\} \qquad (4.41)$$

$$\sum_{i=1}^{I}\left(Ton_i^o \times \left(g_i - \overline{G}_t\right)\right) \times x_{i,t}^c \le 0 \qquad \forall t \in \{1,...,T\} \qquad (4.42)$$

$$\sum_{t=1}^{T} x_{i,t} \le 1 \qquad \forall i \in \{1,...,I\} \qquad (4.43)$$

$$\left|\mathbb{P}_i\right| \times x'_{i,t} - \sum_{i \in \mathbb{P}_i}\sum_{t'=1}^{t} x_{i,t'} \le 0 \qquad \forall i \in \{1,...,I\},\ t \in \{1,...,T\} \qquad (4.44)$$

$$x_{i,t} - x'_{i,t} \le 0 \qquad \forall i \in \{1,...,I\},\ t \in \{1,...,T\} \qquad (4.45)$$

$$\sum_{1}^{J} y_{j,t} = 1 \qquad \forall\, t \in \{1,...,T\} \qquad (4.46)$$

$$\sum_{i}^{J} y_{i,t-1} \ge \sum_{i}^{J} y_{i,t} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \qquad (4.47)$$

$$z_{j,t} \ge y_{j,t} - y_{j,(t-1)} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \qquad (4.48)$$

$$z_{j,t} \le y_{j,t} \qquad \forall j \in \{1,...,J\}, t \in \{2,...,T\} \qquad (4.49)$$

$$z_{j,t} = y_{j,t} \qquad \forall j \in \{1,...,J\}, t = 1 \qquad (4.50)$$

$$\sum_{t}^{T} y_{i,t} - n \times \sum_{t}^{T} z_{i,t} \ge 0 \qquad \forall j \in \{1,...,J\} \qquad (4.51)$$

$$x''_{ijt} \ \le\ 0.5\,(x_{i,t} + y_{j,t}) \qquad \forall i \in \{1,...,I\},\ j \in \{1,...,J\}, t \in \{1,...,T\} \qquad (4.52)$$

$$x_{i,t} + y_{j,t} - 1.5 \ \le\ x''_{ijt} \qquad \forall i \in \{1,...,I\},\ j \in \{1,...,J\}, t \in \{1,...,T\} \qquad (4.53)$$

Equation (4.39) to (4.45) are the production scheduling constraints, which is similar to Equations (4.23) to (4.29); however, the decision variables $x_{i,t}$ are binary integers. Equation (4.39) is the mining capacity constraint that ensures that the total tonnage of material extracted in each period is within an acceptable range. Equation (4.40) controls the quantity of ore mined in each period is within the processing plant's capacity. Equations (4.41) and (4.42) force the mining system to achieve the desired grade. Equation (4.43) is the reserve constraint, which

ensures each cluster can be at most mined once. Equations (4.44) and (4.45) are precedence constraints ensuring clusters can only be extracted if all its precedent clusters are entirely removed.

Equation (4.46) guarantees that only one crusher station is available in each period. Equation (4.47) certifies that the crusher station can only be relocated to lower levels or remain static in any period. Equations (4.48) - (4.50) set the relocation conditions of the crusher station; the relocation variable $z_{jt}$ is equal to 1 only if the crusher station moved to level j from another level in period t. Equation (4.51) controls the minimum frequency of the crusher station's relocation, where the cluster station should stay at a specific level for at least N period before relocation.

Equations (4.52) and (4.53) are the key parts for the simultaneously optimization model. They add the material flow decision variable $x''_{ijt}$ to the model. $x''_{ijt}$ is equal to 1 if cluster *i* is mined in period t (denoted by the production scheduling variable $x_{i,t}$) and it is crushed at the level j in the same period (denoted by the crusher location variable $y_{j,t}$ ).

### 4.5.2 BILP Model Structure

The BILP model represents the problem using linear equations with binary decision variables. The objective function and all the constraints are linear, and all the decision variables are binary (0-1). The standard form of a BILP problem to be solved by CPLEX solver is displayed in Equation (4.54):

$$\begin{aligned} &\min \quad \mathbf{c}_f{}^T \mathbf{x} \\ &s.t. \\ &\mathbf{Ax} \le \mathbf{b} \\ &x_k \in \{0,1\} \quad \forall x_k \in \mathbf{x} \end{aligned} \tag{4.54}$$

Where:

$\mathbf{c}_f$ is the linear coefficient vector of the formulation in the objective function: a vector of $n \times 1$, and *n* is the length of decision variable vector;

**x** is the decision variable of the formulation: a vector of $n \times 1$;

**A** denote coefficients of inequality and equality constraints in the model: a matrix of $m \times n$, and $m$ is the number of all the constraints;

In order to solve the formulation in CPLEX, the objective function and all the constraints should be transferred into the standard form by the decision variable vector, coefficient matrices and the bound vectors. As a result, all decision variables are concatenated into a single vector to be solved into the CPLEX optimizer. The objective functions, constraints are also organized into several coefficient matrices or vectors. In this section, the structures of the variables and coefficient matrix are demonstrated.

#### 4.5.2.1 Structures of decision variables

The first type of variables $x_{i,t}$ is a set of binary variables denote whether the cluster $i$ is extracted in period $t$. In this configuration, clusters can only be mined or not mined at all. There are $I \times T$ variables in this type where $I$ is the total number of mining units (clusters), and T is the number of planning periods. All variables in this set are either 0 and 1. The structure of the first variable type is defined in Table 4-2.

Table 4-2. The structure of mining scheduling variables

| | | | | | | |
|---|---|---|---|---|---|---|
| $x_{1,1}$ | ... | $x_{I,1}$ | ... | $x_{1,T}$ | ... | $x_{I,T}$ |
| $t=1$ | | | ... | $t=T$ | | |

The second type of variable $x'_{i,t}$ is a set of binary variables that control the mining precedence of cluster i in period t. The value of $x'_{i,t}$ is equal to 1 only if all the precedent clusters for cluster i are completely extracted in period t. The structure of this variable type (See Table 4-3) is the same as the first type of variables.

Table 4-3. The structure of mining precedence variables

| | | | | | | |
|---|---|---|---|---|---|---|
| $x'_{1,1}$ | ... | $x'_{I,1}$ | ... | $x'_{1,T}$ | ... | $x'_{I,T}$ |
| $t=1$ | | | ... | $t=T$ | | |

The third type of variable $x''_{ijt}$ represents the material flow decision variable $x''_{ijt}$ in this model. $x''_{ijt}$ It is equal to 1 if cluster *i* is mined in period t and transferred via crushed at the level *j* in the same period. (See Table 4-5). There are $I \times J \times T$ variables in this type $J$ is the total number of pit levels. All variables in this set are either 0 and 1.

Table 4-4. The structure of material flow variable

| $x''_{1,1,1}$ | ... | $x''_{I,1,1}$ | ... | $x''_{1,J,1}$ | ... | $x''_{I,J,1}$ | ... | $x''_{1,1,T}$ | ... | $x''_{I,1,T}$ | ... | $x''_{1,J,T}$ | ... | $x''_{I,J,T}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $j=1$ | | | ... | $j=J$ | | | ... | $j=1$ | | | ... | $j=J$ | | |
| $t=1$ | | | | | | | ... | $t=T$ | | | | | | |

The fourth type of variable $y_{j,t}$ is a set of binary variables that denote the crusher location. The value of $y_{j,t}$ is equal to 1 only if the crusher is located at level j in period t. There are J×T variables in this set. The structure of this variable type is shown in Table 4-5.

Table 4-5. The structure of crusher location variables

| $y_{1,1}$ | . . . | $y_{J,1}$ | . . . | $y_{1,T}$ | . . . | $y_{J,T}$ |
|---|---|---|---|---|---|---|
| $t=1$ | | | . . . | $t=T$ | | |

The crusher relocation variables are affiliated to crusher location variables $y_{j,t}$ , and both have the same structure (see Table 4-6). The binary variable $z_{j,t}$ is equal to 1 only if the crusher is relocated to level *j* in period *t*, that is $y_{j,t}=1$ and $y_{j,t-1}=0$.

Table 4-6. The structure of crusher relocation variables

| $z_{1,1}$ | . . . | $z_{J,1}$ | . . . | $z_{1,T}$ | . . . | $z_{J,T}$ |
|---|---|---|---|---|---|---|
| $t=1$ | | | . . . | $t=T$ | | |

4.5.2.2 The BILP Models Objective Function

The objective function as defined by the Equation (4.37) consists of three terms summed over mine life. The first term is the total revenue generated from the portion of cluster economic

value. The second term is the material handling costs, and the last term is the crusher relocation costs. The objective is to maximize the NPV minus the transportation and crusher relocation costs of the mining operation. The general form of the BILP model in the CPLEX solver is to minimize the objective function. Therefore, the negative sign is applied to all the coefficient arrays in the objective function, so the maximization problem can be changed to minimize its minus value. The structures of the decision variable vector and the coefficient vector are presented in Table 4-7.

Table 4-7. The decomposition of the decision variable and coefficient vectors

| | Decision variable types | | | | |
|---|---|---|---|---|---|
| | Scheduling | Precedence | Material flow | Crusher location | Crusher relocation |
| **x** elements | $x_{i,t}$ | $x'_{i,t}$ | $x''_{ijt}$ | $y_{j,t}$ | $z_{j,t}$ |
| **c** corresponding values | Discounted $CLEV_i$ | 0 | Material handling cost $f_{ijt}$ | 0 | Relocation cost $c_t$ |
| Size | $I \times T$ | $I \times T$ | $I \times J \times T$ | $J \times T$ | $J \times T$ |

4.5.2.3 The BILP Models constraints

The constraints are linear inequalities of the continuous and integer variables. Each constraint is transformed to a number of rows of the coefficient matrix multiplied by the decision variables. The coefficient matrix can be considered as a linear combination of the decision variables. These combinations represent the constraint equations, which should satisfy the lower and upper bounds. The general structure of the inequality constraints in matrix form is shown in Figure 4-2. The equality constraints have the same form, whose corresponding lower bound is equal to its high bound. The number of rows for each constraint and its associated decision variables is shown in Table 4-8.

Table 4-8. Number of rows in constraint's coefficient matrix

| Constraint | Number of rows | Associated variables |
|---|---|---|
| Mining capacity | T | $x_{i,t}$ |
| Processing capacity | T | $x_{i,t}$ |
| Blending grade | T | $x_{i,t}$ |
| Reserves | I | $x_{i,t}$ |

| | | |
|---|---|---|
| Precedence | 2×(T×I) | $x_{i,t}, x'_{i,t}$ |
| Material flow control | 2×(T×J ×I) | $x_{i,t}, x''_{i,t}, y_{j,t}$ |
| Crusher location and relocation | T+3J×(T-1)+2J | $y_{j,t}, z_{j,t}$ |

The constraints' coefficient matrix has the same number of columns as the number of decision variables, and the matrix can be further divided into different areas according to the decision variables. The number of these areas is defined based on the number of the decision variables type. If a variable type is not used in a specific constraint, the coefficient matrix elements in its associated area and the constraint's rows are set to zero.

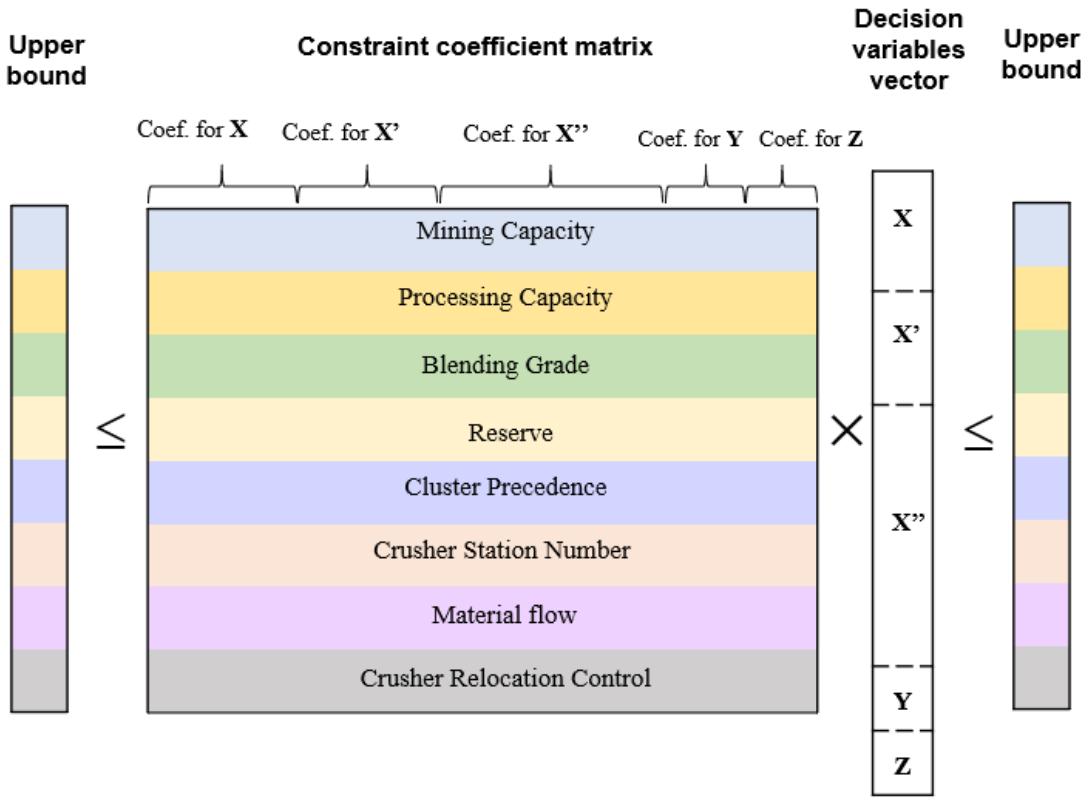



Figure 4-2. Coefficient matrix general structure for the constraints. Adapted from Pourrahimian (2013)

## 4.6 Summary and Conclusion

In this chapter, three mathematical models are proposed to solve the SMIPCC optimization problem.

The first model considered a situation in which the conventional conveyor system is installed in a ramp slot, and the dedicated ramp slot is required to accommodate the lower-angle belts; Therefore, additional stripping tonnage from the slot is considered. This model is a two-step framework to make the production scheduling and crusher location-relocation decisions successively. The first step is a MILP formulation for making production scheduling decisions while maximizing the NPV. The scheduling results obtained from the first step are used as the second step's input parameter to decide the crusher location-relocation plan. The second model is similar to the first one but considered a situation in which the conventional conveyor system is installed in a ramp slot, and the dedicated ramp slot is required to accommodate the lower-angle belts.

The third model is a binary integer linear programming (BILP) model built to simultaneously determine the mining units' extraction period and the crusher station location-relocation plan. This model aims to maximize the NPV while considering the material handling and crusher relocation costs during the mining operation. The material handling costs are determined by both production scheduling variables and the crusher location variables; therefore, another set of decision variables are created to denote the material flow while maintaining the formulation's linearity.

Because the mathematical model is implemented in a MATLAB environment using CPLEX as the optimization engine, the formulations are structured into vectors and matrices form as the input parameters. The coefficient matrices and the structure of the decision variables in the first model are demonstrated.

# CHAPTER 5

# CASE STUDY AND DISCUSSION OF RESULTS

## 5.1 Introduction

This chapter presents the application of the developed model in a case study. The mineral deposit presented for testing purposes is a copper deposit from a standard test dataset called Marvin (Espinoza et al. 2013). A small dataset as a part of the main dataset is considered for modelling purposes.

First, the block model of the dataset was aggregated into clusters, and different conveyor locations for HAC and conventional conveyors were defined. Then, three proposed mathematical models: (i) two-step LP model for conventional conveyors and (ii) two-step LP model for HAC, and (iii) BILP model for HAC were applied. Each model was implemented to the cluster level under different conveyor locations, respectively. Then production scheduling and crusher location-relocation planning results were obtained.

## 5.2 Modelling Dataset

This model was applied to a small-scale copper mine dataset with 2006 blocks and six levels, where each block is 50 m×50 m in width, 40 m in height. The UPL was predetermined by Geovia's Whittle software. All the blocks' attributes, including rock type, copper grade and rock density, were predetermined by geostatistical approaches. The summary of block rock type information is shown in Table 5-1. Figure 5-2 presents the selected block model dataset that is used in this case study, and the copper grade for ore blocks is showing in different colours. A clustering procedure presented in Chapter 3 is implemented to the block model, and cluster size is set at 20 blocks. The clustering results for this dataset are shown in Figure 5-1.

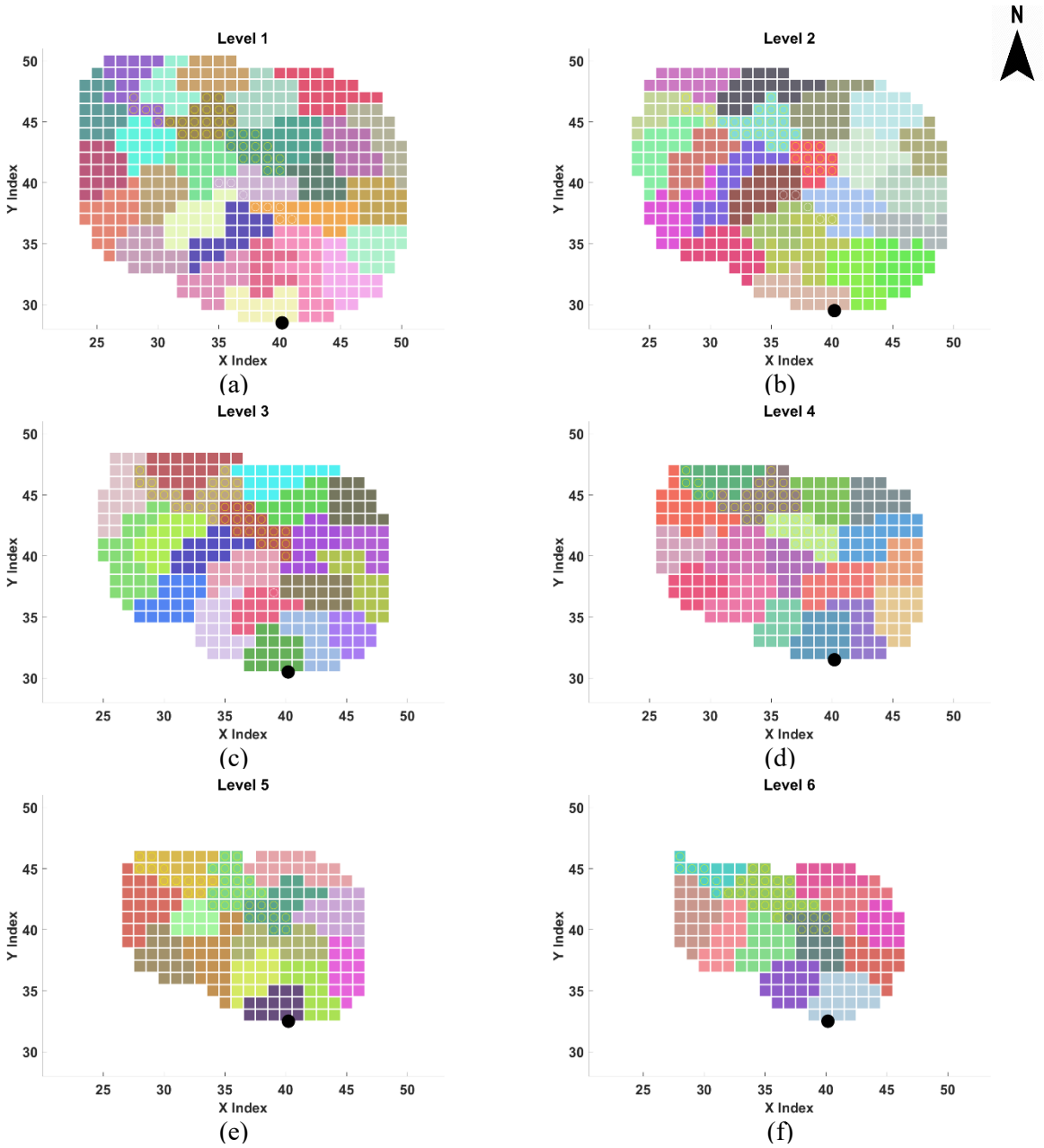



Figure 5-1. Clustering results for conveyor rotation of 0˚ from the top (a) to the bottom level (f)

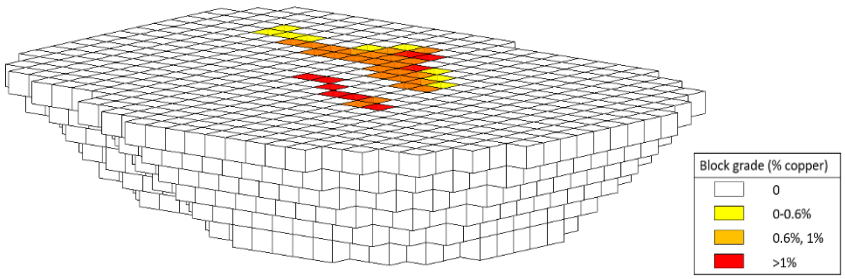



Figure 5-2. The selected block model dataset with the UPL and mineralized zones

Table 5-1. Summary of rock type information in the block model

| Rock type | Density (tonne/m$^3$) | Average grade (%) | Tonnage (Mt) | Number of Blocks |
|---|---|---|---|---|
| MZ1 (ore) | 2.1 | 1.836 | 2.94 | 14 |
| MZ2 (ore) | 2.1 | 0.822 | 44.73 | 213 |
| W (waste) | 1.8 | 0 | 320.22 | 1779 |

Because the mining direction is incorporated in the clustering process, the clustering results can be varied by different conveyor side rotations. Table 5-2 summarizes the levels' information after clustering at conveyor side rotation of 0˚. In this case, the total number of clusters within the UPL of the block model was 105.

Table 5-2. Summary of the levels' information

| | Number of Blocks | Number of Clusters | Total Tonnage (Mt) | Ore Tonnage (Mt) | Average Ore Grade (%) |
|---|---|---|---|---|---|
| Level 1 | 498 | 25 | 91.36 | 9.28 | 0.81 |
| Level 2 | 426 | 22 | 78.24 | 8.79 | 0.90 |
| Level 3 | 356 | 20 | 65.56 | 8.45 | 0.94 |
| Level 4 | 294 | 15 | 54.56 | 8.34 | 0.86 |
| Level 5 | 238 | 13 | 44.36 | 8.03 | 0.78 |
| Level 6 | 194 | 10 | 36.31 | 7.51 | 0.76 |

Various technical and economic parameters were set as the input of the mathematical models. Their values are shown in Table 5-3. Part of the data obtained from (de Werk et al. 2017).

Table 5-3. Production parameters of the mathematical models

| Category | Parameter | Quantity |
|---|---|---|
| Economic Factors | Reference mining cost* ($/t) | 1.5 |
| | Ore processing cost ($/t) | 3.06 |
| | Recovery (%) | 90 |
| | Cu price ($/t) | 7,936 |
| | Selling cost ($/t) | 0 |
| | Discount rate per year (%) | 8 |

| | | |
|---|---|---|
| Transportation Costs | Horizontal trucking cost ($/km·t) | 0.2 |
| | Conveyor vertical lifting cost ($/level·t) | 0.3 |
| | Vertical trucking cost ($/level·t) | 1.2[a] |
| | Crusher relocation cost ($/time) | 1,000,000[b] |
| Production | Upper mining capacity (Mt/yr) | 30 |
| | Lower mining capacity (Mt/yr) | 25 |
| | Upper processing capacity (Mt/ yr) | 6 |
| | Lower processing capacity (Mt/ yr) | 4 |
| | Upper ore blending grade (%) | 1.1 |
| | Lower ore blending grade (%) | 0.5 |
| | Mine life (yrs) | 10 |
| | Minimum crusher relocation interval (yr) | 2 |

*Note.* *This reference mining cost excludes transporting cost and crushing cost. Data are from de Werk et al. (2017)[a] , and Abbaspour et al. (2018)[b]

In this model, eight scenarios with different conveyor side rotation were calculated, from 0˚ to 315˚ with a step size of 45˚. For each scenario, the clustering was done at all levels, and then the problem was solved for that scenario. Figure 5-3 shows these eight scenarios, and the HAC locations are displayed in straight lines.

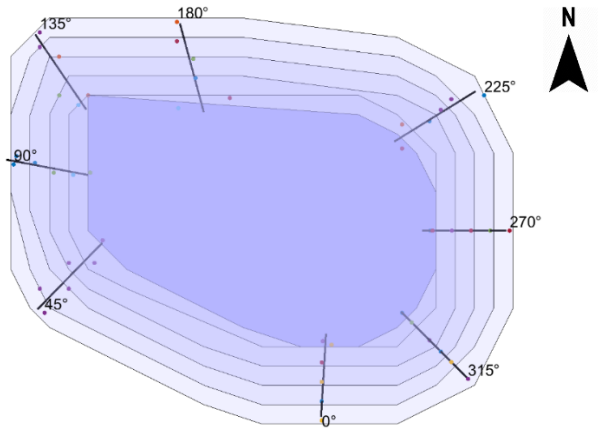



Figure 5-3. The outline of various candidate HAC lines (black straight line) based on tangent points and rotation angles

## 5.3 Application of Two-Step LP Model for Conventional Conveyor

This proposed two-step LP model considers a situation that the conventional conveyor is located in a ramp slot. This ramp slot introduces extra mining tonnage and crusher location constraints. The performance of different conveyor layouts was analyzed based on the NPV and the corresponding production scheduling generated in the first step (MILP formulation). The crusher location-relocation plan is then determined based on the optimum scenario obtained from the first step. The mathematical framework was developed in MATLAB (MATLAB 2018) and solved in the IBM ILOG CPLEX (IBM 2011) environment. CPLEX uses a branch-and-bound scheme with an integer linear programming solver to solve the convex BILP model, ensuring an optimal solution if the algorithm is run to completion. A gap (EPGAP) was used as an optimization termination criterion. Its value is a relative tolerance on the gap between the best integer objective and the best objective value among the remained nodes (IBM 2011).

### 5.3.1 Production Scheduling Results

This step considers all the conveyor location scenarios and maximizes the NPV of each scenario respectively. Due to ramp slot implementation's the extra stripping tonnage, the mining capacity was changed and was set between 30 to 35 Mt/year. The slope angle of the conveyor line is set to 20˚. The additional extraction tonnage of the ramp slot in each level is summarized in Table 5-4. The material handling costs are incorporated into the CLEV calculation. Other parameters are the same as Table 5-3 defines.

Table 5-4. Additional extraction tonnage of ramp slot in each level

| | Total Tonnage in UPL (Mt) | Ore Tonnage (Mt) | Ramp Slot Tonnage (Mt) |
|---|---|---|---|
| Level 1 (Top) | 91.36 | 9.28 | 9.69 |
| Level 2 | 78.24 | 8.79 | 7.24 |
| Level 3 | 65.56 | 8.45 | 4.89 |
| Level 4 | 54.56 | 8.34 | 2.81 |
| Level 5 | 44.36 | 8.03 | 1.20 |
| Level 6 | 36.31 | 7.51 | 0.23 |

An Intel four-core CPU at 3.2GHz with 6GB RAM was used to conduct the computation. Since the dataset is relatively small, the relative gap tolerance was set to 0. The average CPU time for each scenario was about 50 seconds. Figure 5-8 shows a summary of the obtained NPVs for eight rotation angles. It can be seen the maximum NPV was obtained at a 180˚ rotation angle. The highest NPV was 40.6% higher than the worst scenario.

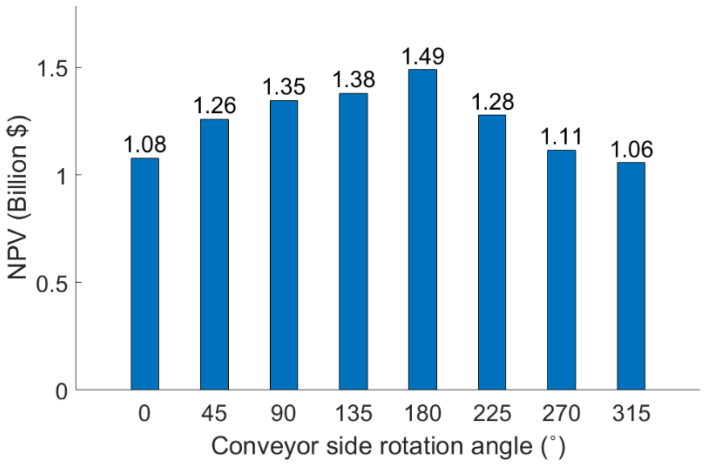



Figure 5-4. Comparison of NPV obtained by the MILP formulation for different scenarios

Table 5-5 summarizes the results of different scenarios in detail. Each scenario's total ore extraction tonnages were almost equal, while the waste tonnage was different. Because the mining capacity was increased, more ore tonnage can be recovered from the pit. However, the generates NPVs were lower than the MILP formulation for HAC in Subsection 5.4.1. The reason could be more waste tonnage, longer trucking and less production flexibility due to the slot construction. The optimum scenario has the lowest total extraction tonnage and stripping ratio, which means less waste was extracted, and the mining cost was reduced.

Table 5-5. Summary of results for eight conveyor side rotation scenarios

| Rotation angle (˚) | Total tonnage (Mt) | Ore tonnage (Mt) | Total stripping ratio | NPV (B$) |
|---|---|---|---|---|
| 0 | 350.00 | 50.17 | 5.98 | 1.078 |
| 45 | 332.86 | 50.17 | 5.63 | 1.259 |
| 90 | 311.52 | 50.17 | 5.21 | 1.346 |

| | | | | |
|---|---|---|---|---|
| 135 | 317.25 | 50.17 | 5.32 | 1.380 |
| 180 | 299.35 | 50.17 | 4.97 | 1.488 |
| 225 | 308.23 | 45.30 | 5.80 | 1.277 |
| 270 | 350.00 | 50.17 | 5.98 | 1.114 |
| 315 | 350.00 | 50.17 | 5.98 | 1.056 |

The production scheduling results for the optimum scenario are displayed in Figure 5-5. The blue line shows the commutative discounted cash flow (DCF); the yellow bars and the grey bars represent the extracted tonnage of ore and waste in each period, respectively; the red line is the average ore grade. The tonnage of extracted material in each period is between 30 Mt to 35Mt, which are the lower and upper bound of the mining capacity, and the ore tonnage per period is within the processing capacity limits. The DCFs of the first half periods contribute the over 65% of NPV, as the early production periods have relatively low discounted factors. The average ore grade has some fluctuation in different periods due to the ore grade distribution, and no noticeable trends can be seen from the results. Because this model was solved at the cluster level, and the ore grade was averaged within each cluster, the results could not reflect the selective mining for high-grade blocks.

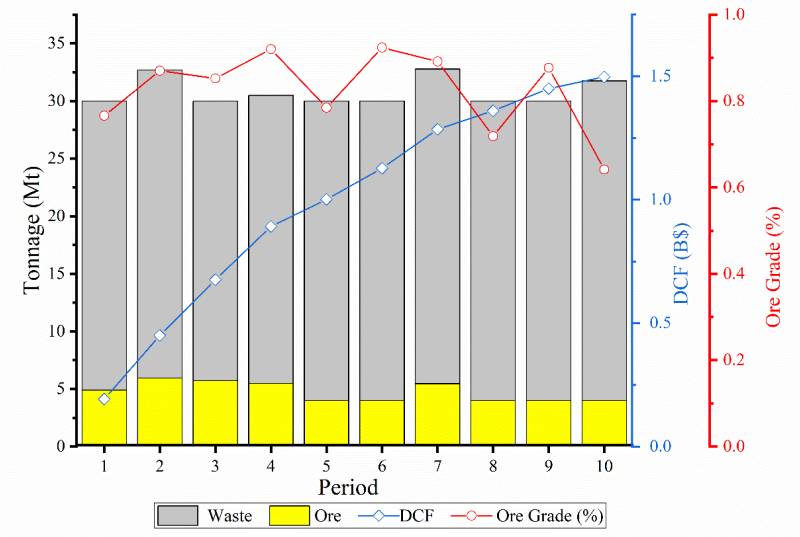



Figure 5-5. Production scheduling for the scenario with 180˚ rotation angle

The mining sequence of the optimal scenario generated by the first step MILP formulation is illustrated in Figure 5-6, and Figure 5-7 shows the plan view of each level separately. Each

colored block corresponds to a specified period that it is completely extracted. Unscheduled blocks are uncolored. Clusters closer to conveyor side or higher levels are mined earlier, while some blocks from the opposite side of the conveyor wall remain intact. These blocks have the least precedence and would be mined during the latter period of mine life, which is depreciated by a higher discount factor. The total tonnage of the extracted material was 299.35 Mt, including 16.37 Mt from the ramp slot. Total tonnage in the UPL was 367.89 Mt; thus, only 76.9% of materials inside the UPL were mined. In a test calculation, although the upper bound of the mining capacity was removed, these far side blocks were still not scheduled in this model. That indicates the conventional UPL based on the track-and-shovel system should be updated according to the conveyor side, as materials in the opposite of conveyor have less contribution to the NPV.

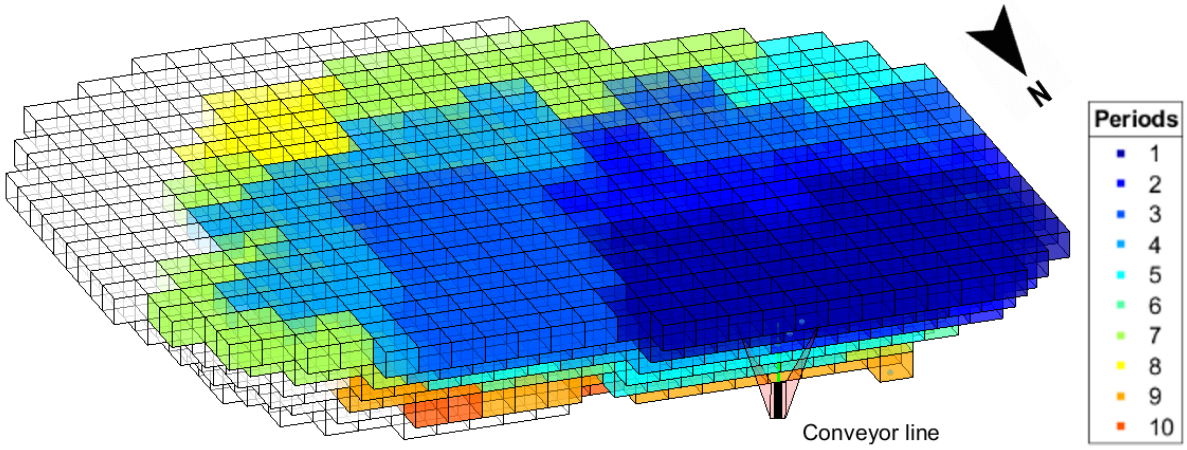
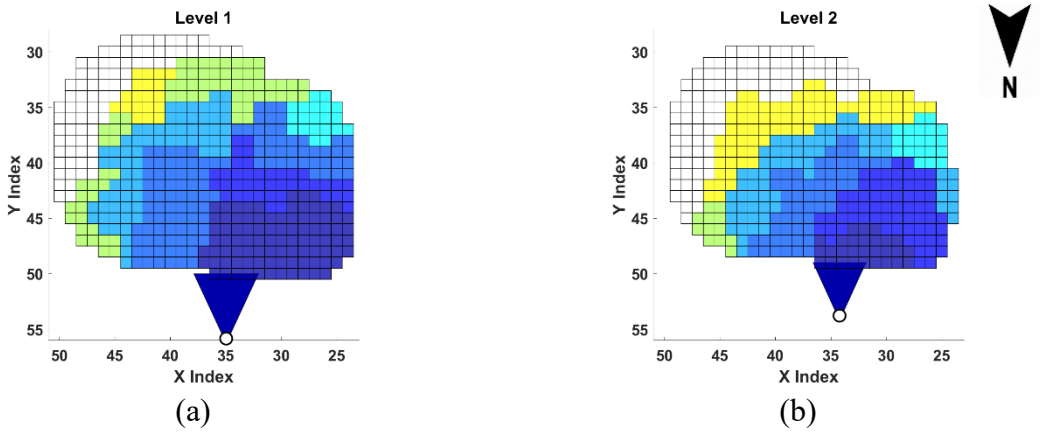



Figure 5-6. Production scheduling results for the optimum scenario with the ramp slot in 3D view



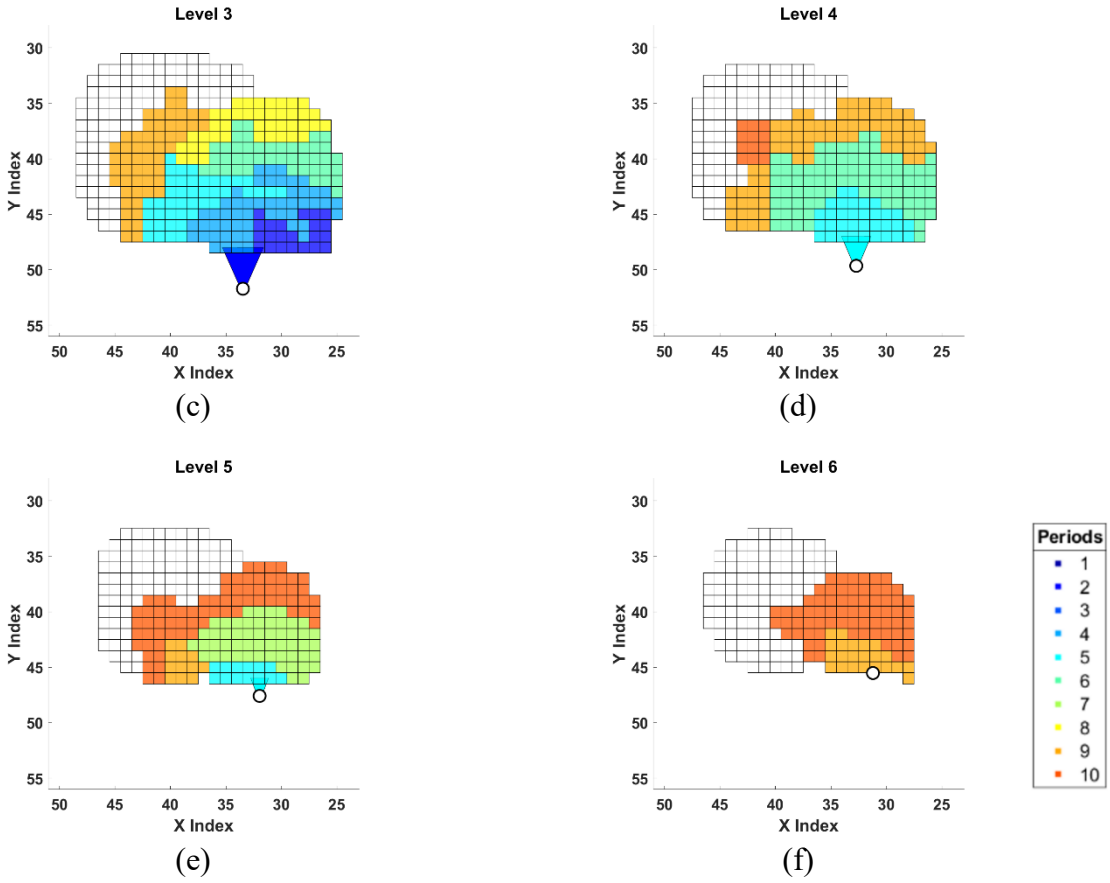



Figure 5-7. Plan view of production scheduling results from the top level (a) to the bottom level (f) with ramp slot and conveyor spot

### 5.3.2 Crusher Location-Relocation Results

The second step was a facility location problem, which decides the crusher location-relocation plan based on the production schedule and slot excavating plan generated from the first step. In this study, only the optimum conveyor location scenario was considered at the 180˚ rotation obtained by the MILP formulation.

The formulation in this step contained 120 binary variables and could be solved in less than one second. The crusher location result is shown in Table 5-6. In this model, the minimum crusher relocation interval was set at two periods. The crusher was relocated twice in P3 and P6 from its initial location as the mining level goes downward. The minimum total transportation costs were $ 365.14 million. After deducting the total transportation costs, the optimum NPV was $ 1122.9 million.

Table 5-6. Results of the crusher location-relocation plan

| Variable values | | Period | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
| Level | L1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | L2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | L3 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | L4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | L5 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | L6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 5.4 Application of Two-step LP Model for HAC

This proposed two-step LP model considers a situation that the high-angle conveyor is installed directly along the final pit wall. Similar to the first model, the production scheduling and crusher location-relocation plans are generated from two separate steps. However, ramp slot is not considered in this model.

### 5.4.1 MILP formulation results

This step investigated all the conveyor location scenarios and maximizes the NPV of each scenario respectively. The parameters are the same as Table 5-3 defines.

An Intel four-core CPU at 3.2GHz with 6GB RAM was used to conduct the computation. Since the dataset is relatively small, the relative gap tolerance was set to 0. The average CPU time for each scenario was about 40 seconds. Figure 5-8 shows a summary of the obtained NPVs for eight rotation angles. It can be seen the maximum NPV was obtained at a 180˚ rotation angle. The highest NPV was 42.6% higher than the worst scenario.

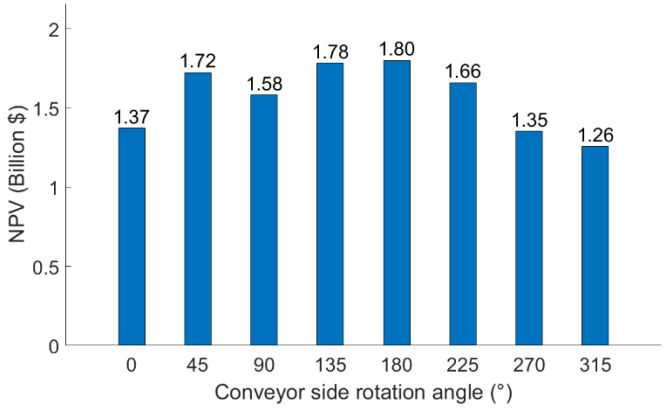


Figure 5-8 Comparison of NPV obtained by the MILP formulation for different scenarios

The cumulative block economic value (BEV) was generated to evaluate the relationship between the block model value distribution and the optimum conveyor location (Figure 5-9). The cumulative BEV summarizes each column BEV within the UPL in this model, and linear interpolation is applied to smooth the value distribution. As expected, the optimum conveyor location is close to the block model's high-value area (the yellow and red area in Figure 5-9). Because the mining operation starts from the conveyor location, the high-value area can be extracted sooner under the optimum conveyor location than the other locations. This time difference can impact discounted cash flow during the early stage of mine life and result in different NPVs.

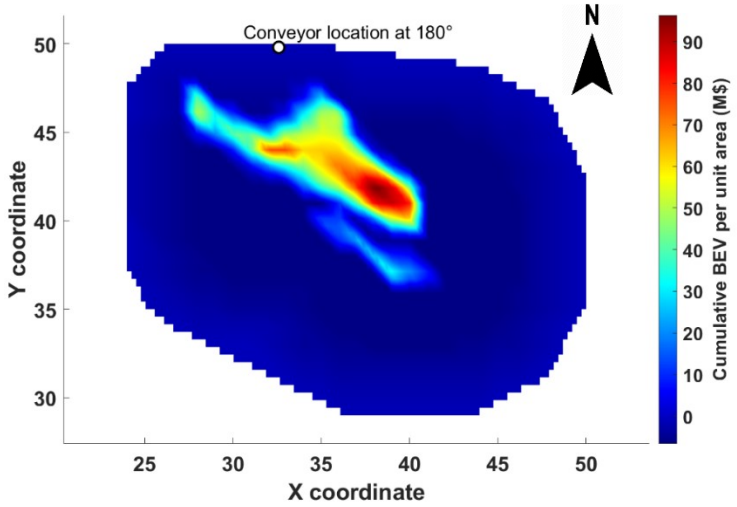



Figure 5-9. The plan view of the cumulative BEV of the block model

Table 5-7 shows the results of different scenarios in detail. While each scenario's total tonnages were between 273 to 300 Mt, almost equal, the ore tonnages show a more significant fluctuation. The optimum scenario had the lowest total stripping ratio, which means less waste was extracted, and the mining cost was reduced.

Table 5-7. Summary of MILP results for eight conveyor side rotation scenarios

| Rotation Angle (˚) | Total Tonnage (Mt) | Ore Tonnage (Mt) | Total Stripping Ratio | NPV (B$) |
|---|---|---|---|---|
| 0 | 299.66 | 37.58 | 6.97 | 1.372 |

| | | | | |
|---|---|---|---|---|
| 45 | 284.73 | 50.17 | 4.68 | 1.723 |
| 90 | 297.71 | 45.53 | 5.54 | 1.581 |
| 135 | 281.24 | 49.76 | 4.65 | 1.782 |
| 180 | 272.72 | 50.17 | 4.44 | 1.798 |
| 225 | 292.08 | 43.32 | 5.74 | 1.658 |
| 270 | 300.00 | 38.10 | 6.87 | 1.352 |
| 315 | 296.65 | 36.91 | 7.04 | 1.258 |

Figure 5-10 shows the production tonnage of waste and ore for the scenario with a 180˚ rotation angle. The commutative discounted cash flow (CDCF) increases stably during the 10-period of mine life.

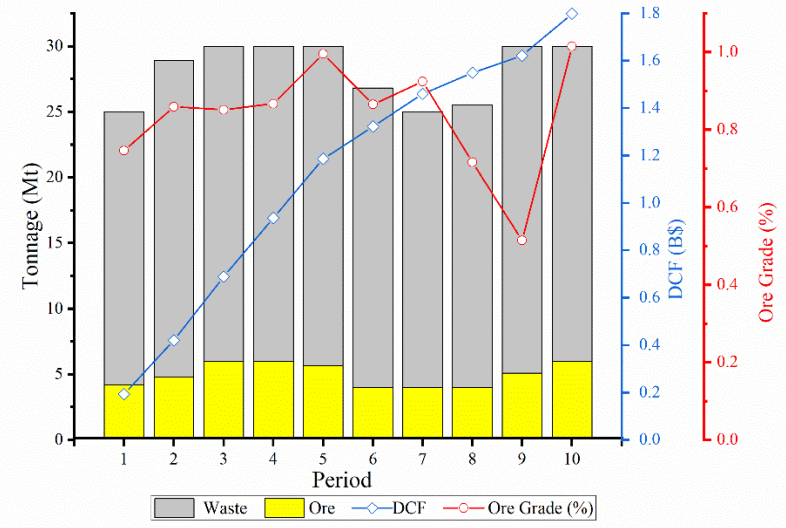



Figure 5-10. Production scheduling for the scenario with 180˚ rotation angle

The mining sequence of the optimal scenario generated by the MILP formulation is illustrated in Figure 5-11, and Figure 5-12 shows the plan view of each level separately. Each colored block corresponds to a specified period that it is completely extracted. Blocks from clusters that are leftover by the end of mine life are uncolored. In this result, The total tonnage of the extracted material was 272.29 Mt, which means only 74.0% of materials inside the UPL were mined, considering the total tonnage in UPL was 367.89 Mt.

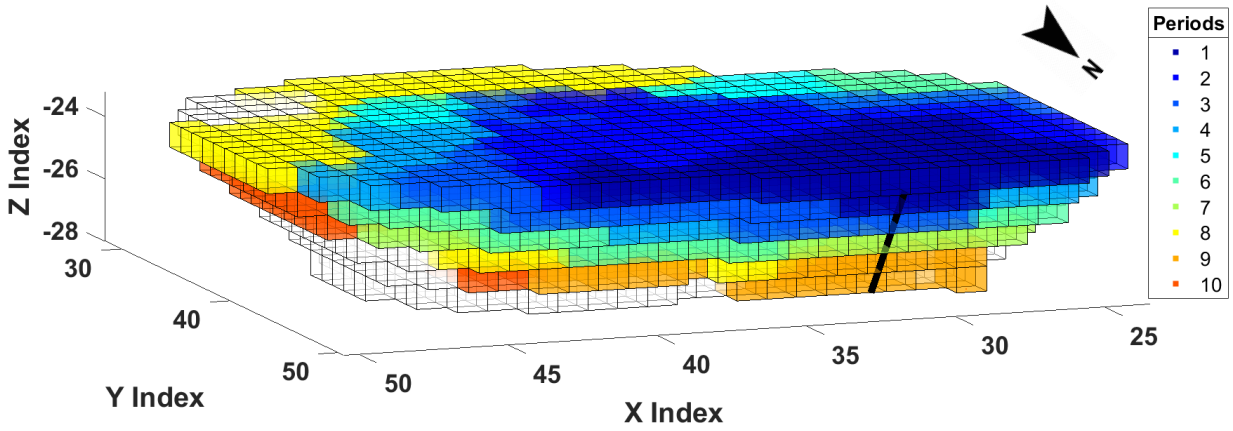
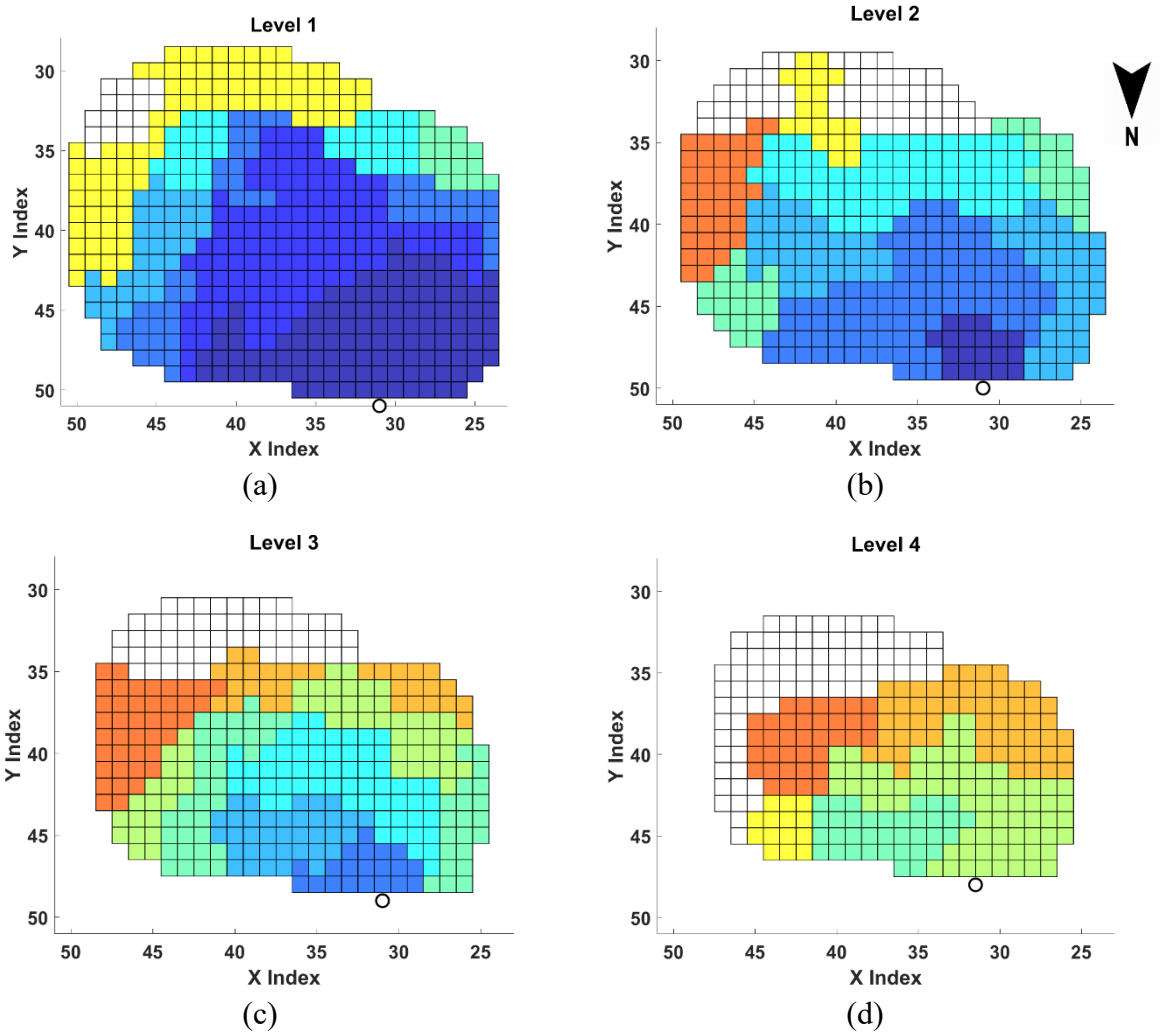



Figure 5-11. Production scheduling results for the optimum scenario in 3D view




(a) (b)

(c) (d)

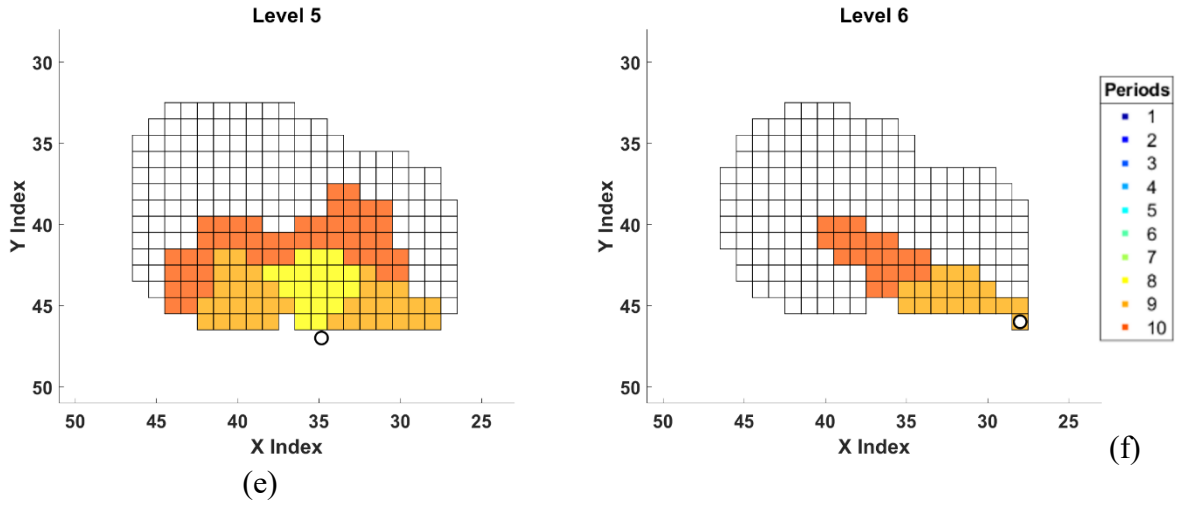



Figure 5-12. Plan view of production scheduling results of LP model for HAC

### 5.4.2 Crusher Location-Relocation Results

The BIP formulation decides the crusher location-relocation plan based on the production scheduling generated from the first step. In this section, only the optimum conveyor location scenario obtained from the previous step was considered.

The crusher location result is shown in Table 5-8. The crusher was relocated three times in P3, P6 and P9 from its initial location as the mining operation went downward. The minimum duration was an input parameter to the model. The minimum total transportation costs, including material handling and crusher relocation, were $326.96 million.

Table 5-8. Results of the crusher location-relocation plan

| Variable values | | Period | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
| Level | L1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | L2 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | L3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| | L4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | L5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | L6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 5.5 Application of BILP Model for HAC

The proposed BILP model was applied to each conveyor location scenario to make production scheduling and crusher station location-relocation decisions simultaneously. An Intel four-core CPU at 3.2GHz with 6GB RAM was used to conduct the computation. The relative gap tolerance was set to 0. The average CPU times were varied under different scenarios, with an average of 450 seconds. The goal was to maximize the NPV considering the material handling costs and crushing station relocation costs. Figure 5-13 summarizes the obtained NPVs for eight rotation angles: the maximum NPV was $ 1.47 billion obtained at a 180˚ rotation angle. This rotation angle is the same as the previous two models. This value was 58.1% higher than the NPV in the worst scenario.

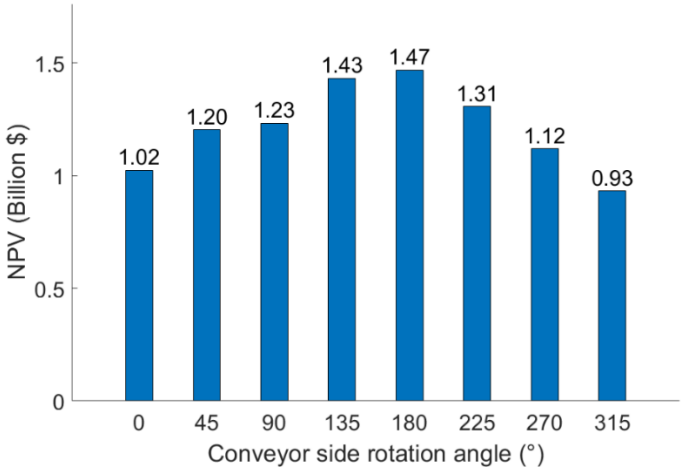



Figure 5-13. Comparison of NPV obtained by the BILP model for different scenarios

Table 5-9 shows the results of different scenarios in detail. The optimum scenario has the highest ore tonnage and lowest stripping ratio. In this case, the conveyor layout is closer to the ore body, and less waste should be extracted before the ore body's exposure.

Table 5-9. Summary of BILP results for eight conveyor side rotation scenarios

| Rotation angle (˚) | Total tonnage (Mt) | Ore tonnage (Mt) | Total stripping ratio | NPV (B$) |
|---|---|---|---|---|
| 0 | 287.02 | 34.48 | 7.33 | 1.023 |
| 45 | 258.05 | 41.33 | 5.24 | 1.204 |
| 90 | 258.79 | 42.43 | 5.10 | 1.232 |

| | | | | |
|---|---|---|---|---|
| 135 | 286.17 | 46.17 | 5.20 | 1.432 |
| 180 | 277.23 | 45.75 | 5.06 | 1.470 |
| 225 | 292.08 | 43.32 | 5.74 | 1.306 |
| 270 | 278.11 | 36.91 | 6.54 | 1.118 |
| 315 | 282.10 | 35.14 | 7.03 | 0.932 |

### 5.5.1 Results of Production Scheduling

Figure 5-14 shows the production tonnage of waste and ore for the scenario with a 180˚ rotation angle. The commutative discounted cash flow (DCF) increases stably during the 10-year mine life. The DCF in this model includes the transportation costs of the associated year.

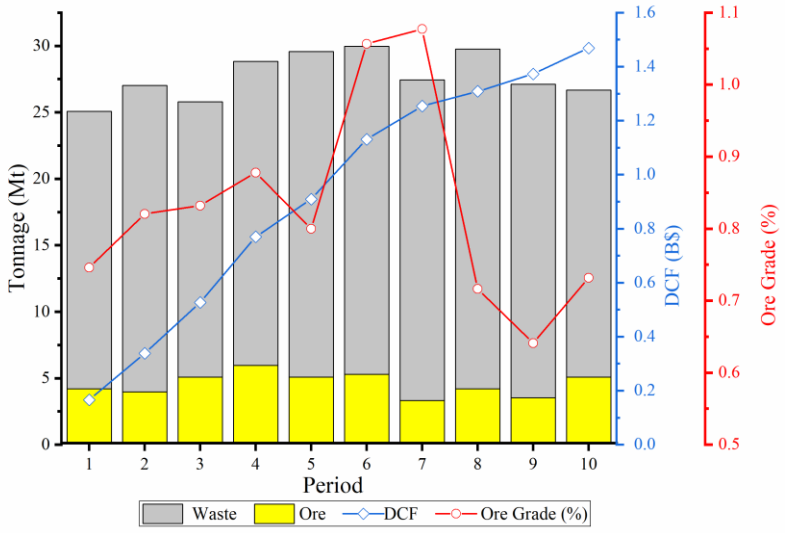



Figure 5-14. Production scheduling for the scenario with 180˚ rotation angle

The mining sequence of the optimal scenario generated by the BILP model is illustrated in Figure 5-15, and Figure 5-16 shows the plan view of each level separately. Each colored block corresponds to a specified period that it is completely extracted, and blocks from unmined clusters by the end of mine life are uncolored. The total tonnage of the extracted material was 277.23 Mt, which means only 75.4% of materials inside the UPL were mined

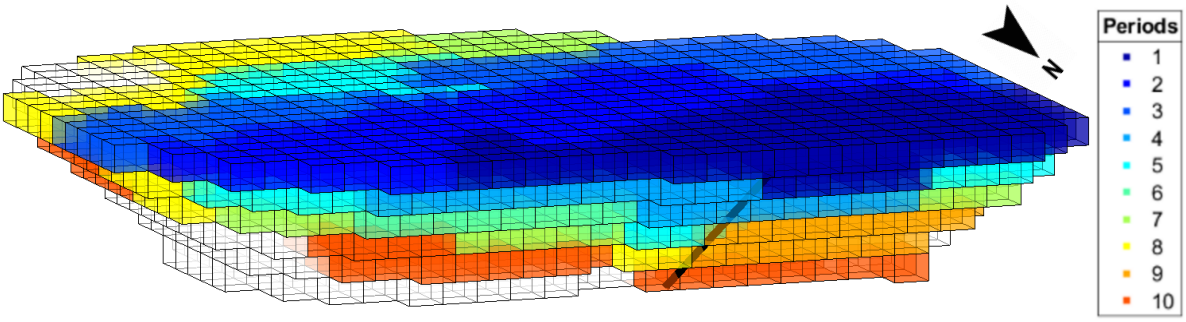



Figure 5-15. Production scheduling results for the optimum scenario in 3D view

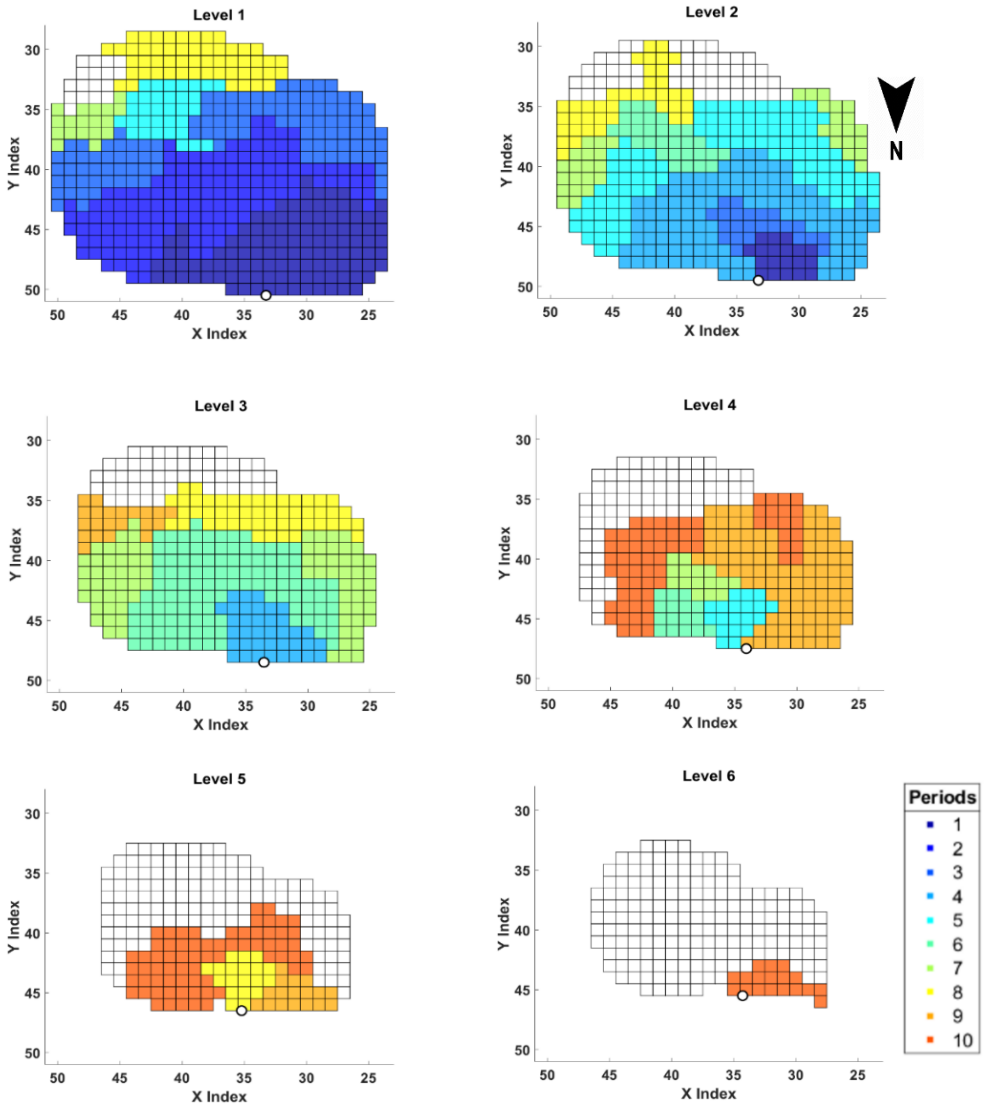



Figure 5-16. Plan view of production scheduling results from the top level (a) to the bottom level (f) with the conveyor spot (hollow circle)

### 5.5.2 Crusher Location-Relocation Results

The crusher location-relocation plan was determined by variables $y_{j,t}$. This set of variables was restructured into a matrix and displayed in Table 5-2. The value 1 in each cell represents the crusher is located in the associated level and period along the conveyor line. The minimum duration for the crusher staying at a certain level is set to 2 periods. The crusher was initially installed in the first level; then, it was relocated three times as the mining level goes downward. In this model, the minimum number of crusher location periods was set at 2 to avoid frequent relocation.

Table 5-10. Results of the crusher location-relocation plan

| Variable values | | Period | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **P7** | **P8** | **P9** | **P10** |
| Level | **L1** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **L2** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | **L3** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| | **L4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | **L5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **L6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The total discounted material handling cost generated from the model was $ 206.3 million, and the net present crushing station relocation cost was $ 2.01 million. These parts of costs were combined into the NPV value of the results.

When the crushing station was located at a specific level, all extracted materials were hauled to that level by trucks. As the mining operation moved downward, the crushing station is also relocated to a lower level, and the materials extracted during this relocation period were sent to the crusher's new location. Figure 5-17 shows the crusher destination of each cluster hauled by trucks. Clusters in a specific color were sent to the corresponding crusher level as the legend displays. The numbers in each cluster represent the level difference between the crusher destination and that cluster. Clusters with positive numbers indicate the inside materials are trucked to the crushing station at a higher level; in contrast, negative numbers mean the associated clusters are trucked downward to the crusher, and 0 implies the associated clusters are trucked horizontally to the crushing station at the same level. Each crushing station's

location is shown in downward-pointing triangles with a specific color corresponding to the legend.

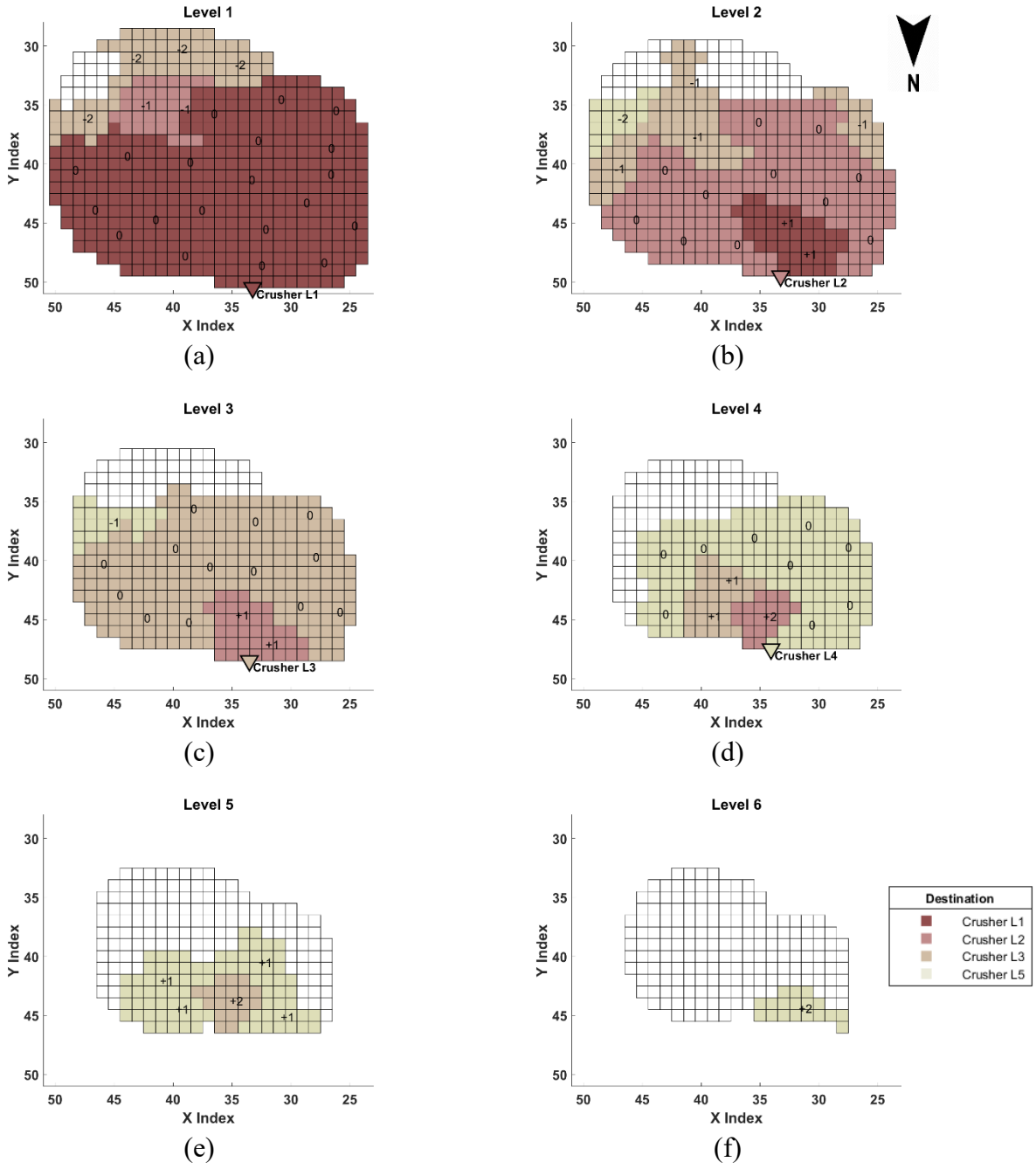



Figure 5-17. Plan view of crusher locations and their feeding clusters of each level

### 5.5.3 Sensitivity Analysis

In this section, the impact of changes in constraints on the NPV and CPU time was investigated for different MIP tolerance gaps and cluster resolution. Also, the changes in transportation costs to the NPV and the changes in the relocation costs to the crusher relocation plan were investigated.

#### 5.5.3.1 Relative MIP gap tolerance (EPGAP)

EPGAP a relative tolerance on the gap between the best integer objective and the objective of the best node remaining. When this difference falls below the value of this parameter, the mixed-integer optimization is stopped. A larger relative MIP gap can result in early termination within a reasonable amount of computation time; however, the model accuracy is compromised. Four different EPGAPs are set in the BILP model, and the summary of CPU time and NPV are presented in Table 5-11. This result is based on the conveyor side rotation angle at 180˚. As the gap declines, the generated NPV shows a slight improvement, while the computation time increases accordingly.

Table 5-11. Summary of different EPGAP values and the solution results

| Relative MIP Gap | CPU Time (s) | NPV (Billion $) |
|---|---|---|
| 10% | 75 | 1.4503 |
| 5% | 145 | 1.4608 |
| 0 | 197 | 1.4700 |

#### 5.5.3.2 Cluster size

The purpose of the clustering step is to reduce the number of mining units in the block model; thus, the number of binary variables and the constraints are also declined, resulting in smaller problem size and shorter running time. The clustering algorithm follows a hierarchical approach, starting from individual blocks to larger clusters; therefore, it takes a slightly longer processing time to create larger clusters. However, due to the complexity of the BILP model, the reduction in the running time with larger clusters is significant. Table 5-1 summarizes the problem size and the optimization framework's running time in three different cluster sizes. All three cases are based on the same optimum conveyor location (rotation of 180˚), and the EPGAP

was set to 5% in the CPLEX solver. The BILP model's CPU time increases dramatically as the number of clusters rises.

On the other hand, for large cluster sizes, the BILP model's resolution is compromised, or even the model becomes infeasible. From Table 5-12, it can be seen that the NPV decreases as the cluster size from 10 to 20. That is because the mining production scheduling is less flexible and accurate under a lower resolution. However, when the cluster size was set to 30, the mathematical model becomes unfeasible. In this case, the cluster size was too large to generate a feasible production scheduling solution while satisfying the production constraints.

Table 5-12. BILP run summary for different cluster sizes

| Cluster Size (Blocks) | No. of Clusters | No. of Binary Variables | NPV (Billion $) | Clustering CPU time (s) | BILP model CPU time (s) |
|---|---|---|---|---|---|
| 10 | 211 | 17000 | 1.543 | 47 | 6548 |
| 20 | 105 | 8520 | 1.461 | 56 | 145 |
| 30 | 87 | 7080 | Infeasible | 64 | * |

#### 5.5.3.3 Material handling costs

Due to the changing energy prices, equipment depreciation, and maintenance costs, the trucking and conveying economic parameters may deviate throughout the mining operation. The sensitivity analysis can identify the critical cost parameters and their influence on costs. In this sense, the original trucking costs (both vertical and horizontal part) and conveying costs were respectively changed by a set of deviations and their impact on the NPV investigated, as Figure 5-18 shows. In this specific case, the changes in conveying costs have a slightly more impact on the NPV than the same percentage of trucking. For example, if the conveying costs increase by 10%, the total NPV will be increased by $ 15.1 million, which is about a 1.0% decline of the NPV; while the NPV will be decreased by 12.3 if the trucking costs increase by 10% trucking costs. However, it should be noted this sensitivity analysis is also dependent on the original material handling costs. If the original trucking costs become higher, the NPV will be more dependent on the trucking costs in the sensitivity analysis.

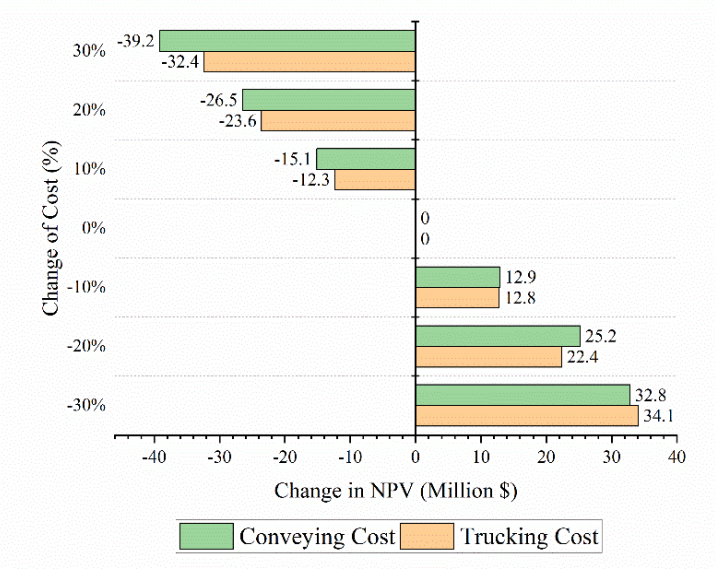



Figure 5-18. Sensitivity analysis for material handling costs

#### 5.5.3.4 Crusher relocation cost

The crusher relocation cost can directly affect the NPV as well as the relocation decision. Further numerical tests were done by modifying the crusher relocation cost to see how its value impacts the crusher relocation plan. No change in the optimum crusher locations occurred by increasing the value of this cost up to $25 million. When the relocation costs exceed $25 million, the crusher location-relocation plan is presented in Table 5-13. In this case, the crusher is relocated twice during the mine life.

Table 5-13. Crusher location-relocation plan scenario by increasing the reloction costs

| Variable values | | Period | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
| Level | **L1** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **L2** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | **L3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **L4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | **L5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **L6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 5.6 Summary and Conclusion

This chapter covered the case study and verification of three mathematical models. All the models were run at the cluster level to maximize the NPV in different conveyor locations, considering the total transportation costs.

The first and the second models were both two-step LP models built for the conventional conveyor and HAC systems, respectively. The HAC can be directly installed along the final pit wall, while a conventional conveyor requires a flatter inclination slop than the final pit wall. Thus, a ramp slot and the additional extraction tonnage and constraints were considered for the conventional conveyor's situation. Both models contain a two-step framework: the first step was to maximize and decide the production schedule by a MILP formulation; then, based on production scheduling results, the second step aimed to minimize the total transportation costs and made the crusher location-relocation decisions.

The third model (BILP model) combined production scheduling and facility location problems. It can make extraction sequencing and crusher location-relocation decisions simultaneously. It was configurated for a HAC situation. The crusher station relocated three times from its initial location during the ten years of mine life. Sensitivity analyses of the CPLEX solver's gap tolerance, material handling costs, and crusher relocation costs were conducted to estimate their impact on the solutions.

All the three models was run at a small-scale block model with 2006 blocks and ten scheduling periods. The results showed that all the models give the optimum conveyor location at the rotation of 180˚. This optimum conveyor location was close to the high-value ore body, where the mining operation could extract the orebody early with fewer waste precedence clusters, generating a higher profit in the early stage of the mine life. Table 5-14 summarizes the obtained results of each model. It can be seen that the first model generated the best NPV value. Because the scheduling decision variables were continuous, this model gives the freedom to partially mine clusters within a period that yield a more flexible production schedule. However, as the total transportation costs were minimized separately in the first and second models, these costs obtained are significantly higher than the BILP model. On the other hand, the CPU times for the two-step LP models are shorter than the BILP model. The first model, a

conventional conveyor's situation with a ramp slot, had the lowest NPV and highest total transportation costs.

Table 5-14. Summary of result for the three mathematical models

| | Model | | |
|---|---|---|---|
| | LP for convl. conveyor | LP for HAC | BILP for HAC |
| NPV include trans. costs (M$) | 1123 | 1471 | 1470 |
| Trans. costs (M$) | 365 | 327 | 208 |
| Avg. CPU time (s) | 50 | 40 | 450 |
| Total tonnage (Mt) | 299.35 | 272.29 | 277.23 |
| Ore tonnage (Mt) | 50.17 | 50.17 | 45.75 |
| Total stripping ratio | 4.97 | 4.44 | 5.06 |
| Material mined within UPL (%) | 76.9% | 74.0.% | 75.4% |

From the production scheduling results of all three models, materials inside the ULP were not wholly mined: some blocks in the pit side opposite the conveyor wall are unscheduled. It indicates the UPL should be updated under the IPCC systems. Figure 5-19 presents the original blocks within the UPL and the scheduled blocks obtained by each of the three models, viewing from the opposite side of the conveyor wall.

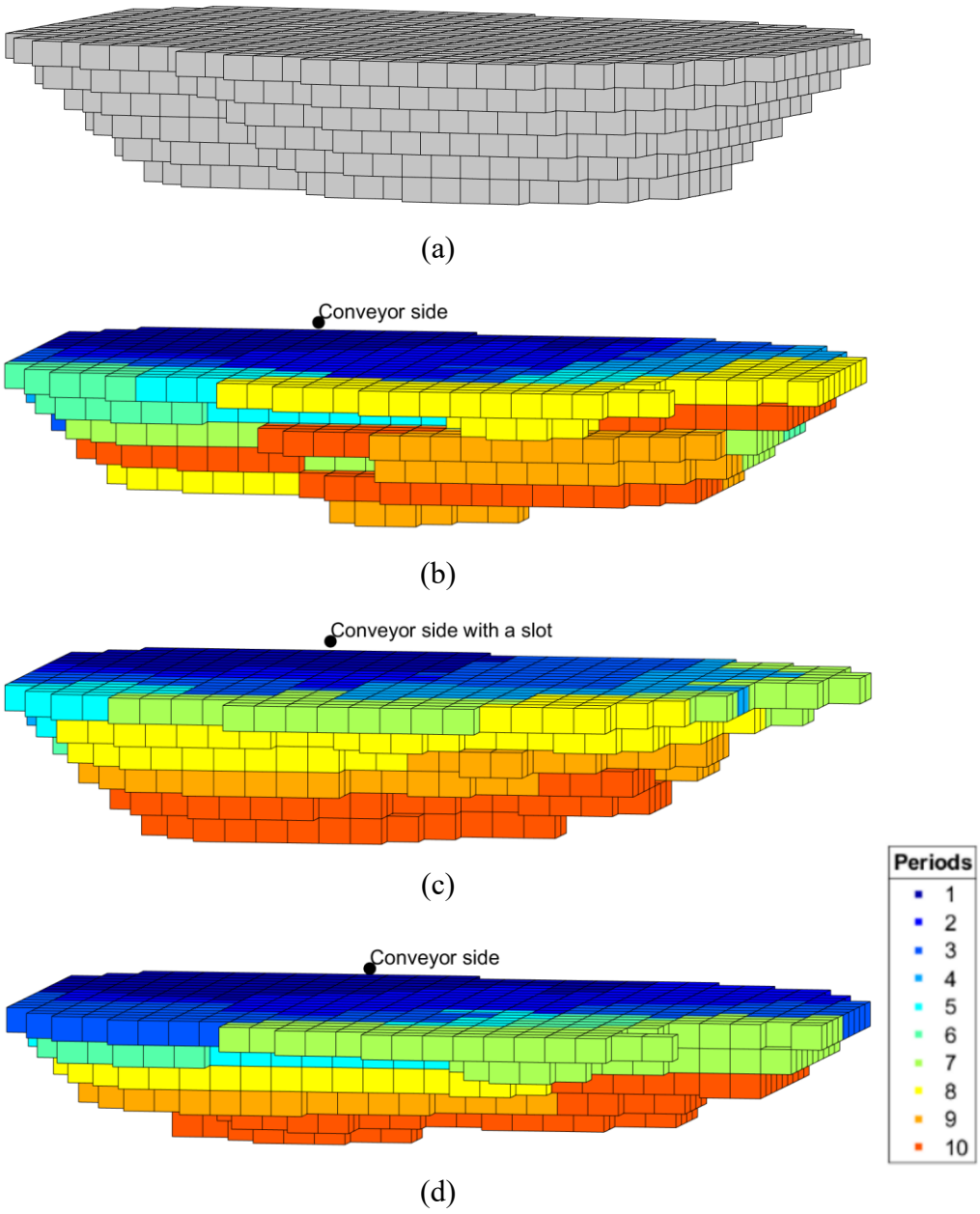



Figure 5-19. The comparison of the scheduled mining areas (a) original UPL, (b) LP model for HAC, (c) LP model for conventional conveyor, and (d) BILP model for HAC

# CHAPTER 6

# SUMMARY, CONCLUSION AND RECOMMENDATIONS

## 6.1 Summary of Research

In-pit crushing and conveying (IPCC) systems have drawn attention to the modern mining industry due to the numerous benefits over the conventional truck-and-shovel system. However, the implementation of the IPCC system can reduce mining flexibility and introduce additional mining sequence requirements. Therefore, under IPCC systems, the strategy mining plans are different from the traditional production scheduling plans based on the truck-and-shovel system.

This research aimed to solve the long-term production scheduling and the crusher location-relocation problem of open-pit mines using a semi-mobile IPCC system. It assumed the conveyor system, either a conventional conveyor or a high-angle conveyor (HAC), was located on one side of the final pit wall throughout the mine life, and the conveyor line could be extended to lower levels as the mining operations go deep. Additional mining sequence constraints were considered from the conveyor line's perspective. A series of candidate conveyor locations were generated along the UPL, and the candidate crusher locations were on the conveyor line of each level.

For each conveyor location, a hierarchical clustering algorithm was applied to aggregate the blocks into larger mining units while considering the mining direction based on the conveyor location. A proposed decision-making step determines the mining precedence among clusters. The unit material handling costs from the loading point to the pit exit were estimated for each cluster based on their coordinates and the candidate crusher location. Moreover, this research investigated a situation for the conventional low-angle conveyor. A series of candidate ramp slots were generated to accommodate the conventional conveyor, while additional slot excavation was considered.

Afterwards, three mathematical models were formulated to solve the LTOPP problems: (i) a two-step LP model for conventional conveyor, (ii) a two-step LP model for HAC, and (iii) a BILP model for HAC. The first two models solved scheduling and location problems in different steps. Each conveyor location was solved independently by a specific mathematical model, aiming to maximize the net present value (NPV) while considering the material handling costs and crushing station relocation costs. The BILP model could simultaneously make production scheduling and crushing station location-relocation decisions.

The conveyor location scenario with the overall maximum NPV was considered the optimum or near-optimum results. The obtained production scheduling and crusher location-relocation decision can provide a reference for the future mine plan and IPCC implementation. The main optimization framework in this thesis is presented in Figure 6-1.

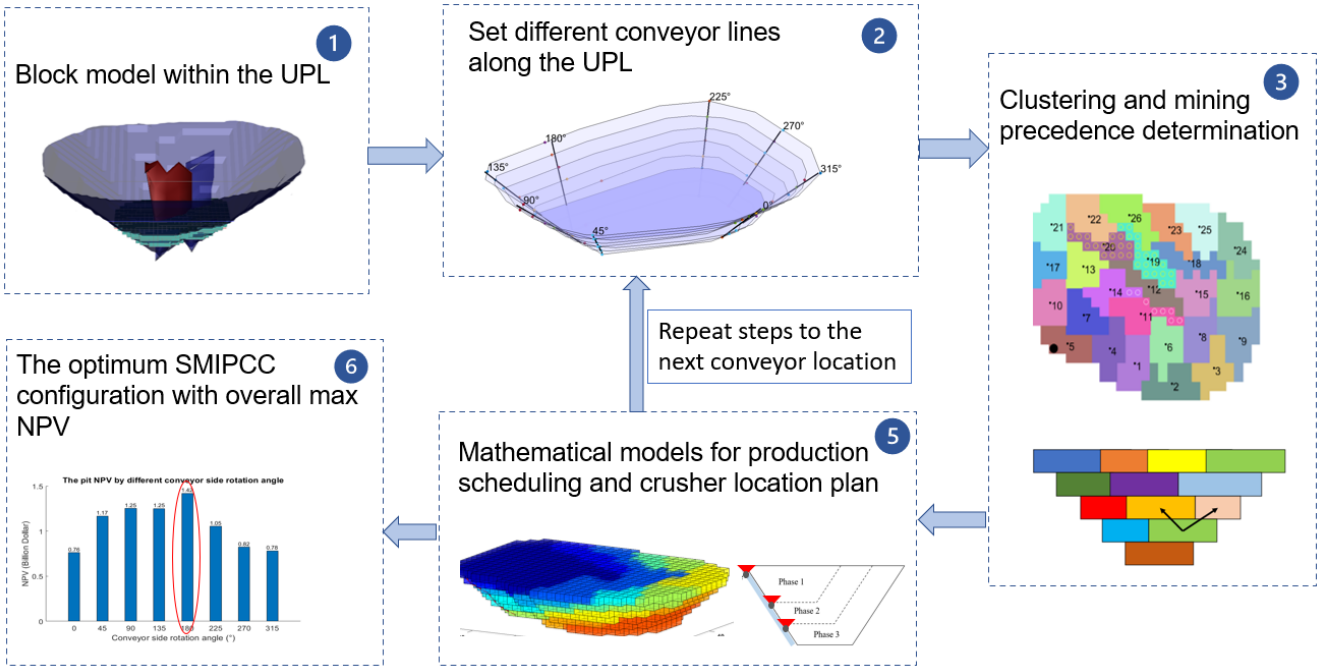



Figure 6-1. A summary of the main optimization framework

## 6.2 Conclusions

One of the most important gaps in the current IPCC systems optimization literature is the LTOPP under the application of IPCC systems. This thesis proposed a new IPCC systems optimization framework to consider the production scheduling problem from the conveyor

location perspective. Various conveyor layouts along the final pit wall were generated as the inputs to the mathematical models, and their NPVs were compared.

It should be noted that although additional mining constraints in the IPCC system can result in a lower NPV than the traditional truck-and-shovel system, the IPCC's operating costs is generally much lower than the trucking fleet. As a result, the overall profit can still favourable to the IPCC transportation method.

The main conclusions of this research are summarized as follows:

- The mathematical models generate production scheduling results at the cluster level. The results are the strategic, yearly production schedule to maximize the NPV while considering the material handling costs and crusher relocation costs;
- Different conveyor locations around the UPL can lead to varied mining direction requirements and generate different production schedules, which results in different NPV. In the case study, the NPV in the best scenario can be more than 50% higher than the value in the worst scenario. Therefore, the conveyor's layout should be designed carefully before implementing the IPCC system, especially for the conveyor line fixed through the mine life.
- The two-step LP model for conventional conveyor was based on the second model, while considering the additional slot extraction and its constraints. The maximum NPV for this model was $1123 million. Due to the extra waste tonnage for the ramp slot and operational constraints, the NPV obtained from the conventional conveyor case was the lowest.
- In the two-step LP model for HAC, the first step was a MILP formulation that aimed to maximize the NPV under a specified mining direction; the generated scheduling results were fed to the second step, a facility location problem, to minimize the total transportation costs. The obtained NPVs under the best scenario were $ 1471 million.
- The BILP model can make the production scheduling and crusher location-relocation decisions simultaneously. In the case study, it was applied to the HAC case. The maximum NPV obtained from the best conveyor location scenario was $ 1470 million, and transportation cost was the lowest among all the models.

- The computation time required for the BILP model was several times longer than the two-step LP models. However, this model can make various decisions simultaneously and generate the lowest transportation costs in the case study.
- From the production scheduling results, not all material in the UPL was mined even though the upper mining capacity limit was removed. The unmined part indicated that the UPL should be updated based on the IPCC system's location.

## 6.3 Recommendations for Future Research

Although the framework developed in this thesis provides new methods and formulations for production scheduling in semi-mobile IPCC systems, further improvement and investigation are required for the mathematical models. The following suggestions for future research address the limitations that can be improved in the future.

- Incorporating geological uncertainties. The reality in the mining industry is much more complicated than this simple assumption may suggest. In practice, mine planners cannot fully understand the quantity and quality of deposits in the ground. On the other hand, because the IPCC systems can reduce the mining flexibility, its application is sensitive to the ore body configuration and grade distribution of the deposit.
- Investigating different orebody configurations. This framework is suitable for dipping ore bodies with flat terrain on the surface. In this case, various conveyor layouts around the UPL can result in different distances to the ore bodies. However, for vertical orebodies or flat-lying seam deposits, the impact of different conveyor locations on the NPV may not be that significant.
- Incorporating other approximate approaches such as heuristics or relaxations to solve the large-scale problems at the block level. This research solved the models based on clusters; therefore, the scheduling resolution is compromised, and the slope angle between clusters cannot be guaranteed.
- Transferring the BILP model to a mixed-integer programming model. That is, using continuous decision variables to denote the production schedule so that each mining unit can be partially mine within a period. The mixed-integer programming model

can generate a higher NPV than the binary model, as the former model gives more freedom in making scheduling decisions. However, this transformation can make the mathematical model nonlinear and cannot be solved by the CPLEX solver. Future studies will need to find a way to solve the mixed-integer programming model while simultaneously optimizing the production scheduling and crusher location-relocation plan.

- Incorporating deviations from production targets. The strict production target constraints in the mathematical model can make the problem infeasible in specific scenarios. By removing this type of constraints and introducing deviation penalty costs can increase the robustness of the proposed models.

# BIBLIOGRAPHY

Abbaspour, H., Drebenstedt, C., Paricheh, M. and Ritter, R. (2018). "Optimum location and relocation plan of semi-mobile in-pit crushing and conveying systems in open-pit mines by transportation problem." International Journal of Mining, Reclamation and Environment **33**(5): 297-317.

Abbaspour, H., Drebenstedt, C., Paricheh, M. and Ritter, R. (2018). "Optimum location and relocation plan of semi-mobile in-pit crushing and conveying systems in open-pit mines by transportation problem." 1-21.

Abbaspour, H. and Maghaminik, A. J. S. R. o. R. (2016). "Equipment replacement decision in mine based on decision tree algorithm." (1): 187-194.

Askari-Nasab, H. and Awuah-Offei, K. (2010). A clustering algorithm for open pit mine production scheduling. Proceedings of Society for Mining, Metallurgy & Exploration (SME) Annual Meeting Preprint, Phoenix, Arizona, USA.

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

Badiozamani, M. and Askari-Nasab, H. (2016). "Integrated mine and tailings planning: a mixed integer linear programming model." International Journal of Mining, Reclamation and Environment **30**(4): 319-346.

Badiozamani, M. M. and Askari-Nasab, H. "An Integrated Model for Oil Sands Long-Term Mine Planning, Tailings and Reclamation Plans."

Bernardi, L., Kumral, M. and Renaud, M. (2020). "Comparison of fixed and mobile in-pit crushing and conveying and truck-shovel systems used in mineral industries through discrete-event simulation." Simulation Modelling Practice and Theory: 102100.

Bienstock, D. and Zuckerberg, M. (2010). Solving LP relaxations of large-scale precedence constrained problems. International Conference on Integer Programming and Combinatorial Optimization, Springer.

Bixby, R. E., Fenelon, M., Gu, Z., Rothberg, E. and Wunderling, R. (2004). Mixed-integer programming: A progress report. The sharpest cut: the impact of Manfred Padberg and his work, SIAM**:** 309-325.

Bjørndal, T., Herrero, I., Newman, A., Romero, C. and Weintraub, A. (2012). "Operations research in the natural resource industry." International Transactions in Operational Research **19**(1-2): 39-62.

Bliek1ú, C., Bonami, P. and Lodi, A. (2014). Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report. Proceedings of the twenty-sixth RAMP symposium.

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

Builes, C. A. J. (2017). A Mixed-Integer Programming Model for an In-Pit Crusher Conveyor Location Problem, Ecole Polytechnique, Montreal (Canada).

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

Chatterjee, S. and Dimitrakopoulos, R. (2020). "Production scheduling under uncertainty of an open-pit mine using Lagrangian relaxation and branch-and-cut algorithm." International Journal of Mining Reclamation and Environment **34**(5): 343-361.

Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E. and Rubio, E. (2012). "A New Algorithm for the Open-Pit Mine Production Scheduling Problem." Operations Research **60**(3): 517-528.

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

Czaplicki, J. M. (2008). Shovel-Truck Systems: Modelling, Analysis and Calculations, CRC Press.

Dagdelen, K. (1987). "Optimum multi period open pit mine production scheduling."

Darling, P. (2011). SME mining engineering handbook, SME.

de Werk, M., Ozdemir, B., Ragoub, B., Dunbrack, T. and Kumral, M. (2017). "Cost analysis of material handling systems in open pit mining: Case study on an iron ore prefeasibility study." The Engineering Economist **62**(4): 369-386.

Dean, M., Knights, P., Kizil, M. and Nehring, M. J. F. M., AusIMM, The University of New South Wales, Australia (2015). "Selection and planning of fully mobile in-pit crusher and conveyor systems for deep open pit metalliferous applications." **146**.

Demirel, N. and Gölbaşı, O. (2016). "Preventive replacement decisions for dragline components using reliability analysis." Minerals **6**(2): 51.

Deutsch, M. (2015). "The Angle Specification for GSLIB Software." In J. L. Deutsch (Ed.), Geostatistics Lessons. Retrieved February 05, 2021, from https://geostatisticslessons.com/lessons/anglespecification.

Dos Santos, J. (2013). "The Cost and Value of High Angle Conveying–A Comparison of Economics for different Conveying Paths." Bulk Solids Handling **33**(1): 18.

Dos Santos, J. (2017). "High Angle Conveying: The Vital (missing) Link to IPCC Systems–2017." Bulk Solids Handling **37**(1): 16-26.

Dos Santos, J. A. (1986). High angle conveyor, Google Patents.

Dos Santos, J. A. (2016). Sandwich Belt High Angle Conveyors Coal Mine to Prep Plant and Beyond–2016. XVIII International Coal Preparation Congress, Springer.

Dos Santos, J. A. and Stanisic, Z. (1986). In-pit crushing and high angle conveying in a Yugoslavian copper mine. Mining Latin America/Minería Latinoamericana, Springer**:** 101-113.

Dryzhenko, A., Shustov, A. and Moldabayev, S. (2017). "Justification of parameters of building inclined trenches using belt conveyors." International Multidisciplinary Scientific GeoConference: SGEM **17**(1.3): 471-478.

Dryzhenko, A., Shustov, O. and Adamchuk, A. (2016). "Prospects for future mining of steep iron-ore deposits in the context of Kryvbas." Metallurgical and mining industry(10): 46-52.

Dzakpata, I., Knights, P., Kizil, M. S., Nehring, M. and Aminossadati, S. M. (2016). "Truck and shovel versus in-pit conveyor systems: a comparison of the valuable operating time."

Eivazy, H. and Askari-Nasab, H. (2012). "A mixed integer linear programming model for short-term open pit mine production scheduling." Mining Technology **121**(2): 97-108.

Espinoza, D., Goycoolea, M., Moreno, E. and Newman, A. (2013). "MineLib: a library of open pit mining problems." Annals of Operations Research **206**(1): 93-114.

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

Gu, Q., Li, X., Chen, L. and Lu, C. (2020). "Layout optimization of crushing station in open-pit mine based on two-stage fusion particle swarm algorithm." Engineering Optimization: 1-24.

Hartigan, J. A. (1985). "Statistical theory in clustering." Journal of classification **2**(1): 63-76.

Hartman, H. L., Britton, S. G., Mutmansky, J. M., Gentry, D. W., Schlitt, W. J., Karmis, M. and Singh, M. M. (1992). SME mining engineering handbook, Society for Mining, Metallurgy, and Exploration Denver.

Hay, E., Nehring, M., Knights, P. and Kizil, M. S. (2019). "Ultimate pit limit determination for semi mobile in-pit crushing and conveying system: a case study." International Journal of Mining, Reclamation and Environment: 1-21.

Hustrulid, W. A., Kuchta, M. and Martin, R. K. (2013). Open pit mine planning and design, two volume set & CD-ROM pack, CRC Press.

IBM (2011). " IBM ILOG CPLEX optimization studio." CPLEX user's manual 12. IBM Corporation: pp 1–148.

Jélvez, E., Morales, N. and Askari-Nasab, H. (2020). "A new model for automated pushback selection." Computers & Operations Research **115**: 104456.

Jélvez, E., Morales, N., Nancel-Penard, P., Peypouquet, J. and Reyes, P. (2016). "Aggregation heuristic for the open-pit block scheduling problem." European Journal of Operational Research **249**(3): 1169-1177.

Jimenez Builes, C. A. (2017). A Mixed-Integer Programming Model for an In-Pit Crusher Conveyor Location Problem Masters thesis, École Polytechnique de Montréal.

Johnson, T. B. (1968). Optimum open pit mine production scheduling. Berkeley, Calif., Berkeley Operations Research center,.

Klingman, D. and Phillips, N. (1988). "Integer programming for optimal phosphate-mining strategies." Journal of the Operational Research Society **39**(9): 805-810.

Koehler, F. (2003). In-Pit Crushing System the Future Mining Option. Twelfth International Symposium on Mine Planning and Equipment Selection, Apr 23 - 25 2003, Kalgoorlie, WA, Australia, Australasian Institute of Mining and Metallurgy.

Konak, G., Onur, A. and Karakus, D. (2007). "Selection of the optimum in-pit crusher location for an aggregate producer." Journal of the Southern African Institute of Mining and Metallurgy **107**(3): 161-166.

Kudowor, A. Y. and Taylor, G. (1998). "Triangulation based volume calculation." University of Newcastle.

Lerchs, H. (1965). "Optimum design of open-pit mines." Trans CIM **68**: 17-24.

Liu, S. Q. and Kozan, E. (2016). "New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems." Expert Systems with Applications **43**: 59-65.

Mai, N., Topalt, E. and Ertent, O. (2018). "A new open-pit mine planning optimization method using block aggregation and integer programming." Journal of the Southern African Institute of Mining and Metallurgy **118**(7): 705-714.

Mai, N. L. (2017). A new Mine Planning Methodology using Topcone Algorithm and Mathematical Programming, Curtin University.

MATLAB (2018). "9.7.0.1190202 (R2019b)." The MathWorks Inc. Natick, Massachusetts.

McCarthy, R. and Eng, P. (2011). "In-pit crushing and conveying: fitting a square peg in a round open pit." Proceedings CIM Montreal 2011.

Meagher, C., Dimitrakopoulos, R. and Avis, D. (2014). "Optimized open pit mine design, pushbacks and the gap problem—a review." Journal of Mining Science **50**(3): 508-526.

Mero, G. B. C. (2017). "Mining." Encyclopædia Britannica.

Nehring, M., Knights, P. F., Kizil, M. S. and Hay, E. (2018). "A comparison of strategic mine planning approaches for in-pit crushing and conveying, and truck/shovel systems." International Journal of Mining Science and Technology **28**(2): 205-214.

Nezhadshahmohammad, F. and Pourrahimian, Y. (2018). "A Clustering Algorithm for Block-Cave Production Scheduling." Global Journal of Earth Science and Engineering **5**: 45-53.

Oberrauner, A. and Turnbull, D. (2013). Essentials on in-pit crushing and conveying (IPCC). Beltcon 17. International Materials Handling Conference.

Osanloo, M., Gholamnejad, J. and Karimi, B. (2008). "Long-term open pit mine production planning: a review of models and algorithms." International Journal of Mining, Reclamation and Environment **22**(1): 3-35.

Osanloo, M. and Paricheh, M. (2019). "In-pit crushing and conveying technology in open-pit mining operations: a literature review and research agenda." International Journal of Mining, Reclamation and Environment **34**(6): 430-457.

Paricheh, M. and Osanloo, M. (2019). "Concurrent open-pit mine production and in-pit crushing–conveying system planning." Engineering Optimization: 1-16.

Paricheh, M. and Osanloo, M. (2019). How to Exit Conveyor from an Open-Pit Mine: A Theoretical Approach. Proceedings of the 27th International Symposium on Mine Planning and Equipment Selection-MPES 2018, Springer.

Paricheh, M. and Osanloo, M. (2019). A New Search Algorithm for Finding Candidate Crusher Locations Inside Open Pit Mines. International Symposium on Mine Planning & Equipment Selection, Springer.

Paricheh, M., Osanloo, M. and Rahmanpour, M. (2017). "In-pit crusher location as a dynamic location problem." Journal of the Southern African Institute of Mining and Metallurgy **117**(6): 599-607.

Paricheh, M., Osanloo, M. and Rahmanpour, M. (2018). "A heuristic approach for in-pit crusher and conveyor systems time and location problem in large open-pit mining." International Journal of Mining, Reclamation and Environment **32**(1): 35-55.

Peng, S. and Zhang, D. (1988). "Computer simulation of a semi-continuous open-pit mine haulage system." International Journal of Mining and Geological Engineering **6**(3): 267-271.

Pourrahimian, Y. (2013). Mathematical programming for sequence optimization in block cave mining, University of Alberta.

Rahmanpour, M., Osanloo, M. and Adibee, N. (2013). An approach to determine the location of an in pit crusher in open pit mines. 23rd International Mining Congress and Exhibition of Turkey, IMCET 2013, April 16, 2013 - April 19, 2013, Antalya, Turkey, Chamber of Mining Engineers of Turkey.

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

Ren, H. and Topal, E. (2014). "Using Clustering Methods for Open Pit Mine Production Scheduling." Mining education Australia-Research Projects Review **3**: 45-49.

Ritter, R. (2016). Contribution to the capacity determination of semi-mobile in-pit crushing and conveying systems Doctoral dissertation, Technische Universität Bergakademie Freiberg, Freiberg, Germany.

Roumpos, C., Partsinevelos, P., Agioutantis, Z., Makantasis, K. and Vlachou, A. (2014). "The optimal location of the distribution point of the belt conveyor system in continuous surface mining operations." Simulation Modelling Practice and Theory **47**: 19-27.

Samavati, M., Essam, D., Nehring, M. and Sarker, R. (2020). "Production planning and scheduling in mining scenarios under IPCC mining systems." Computers & Operations Research **115**: 104714.

Santos, J. and Frizzell, E. (1983). "Evolution of sandwich belt high-angle conveyors." CIM Bull.;(Canada) **76**(855).

Shishvan, M. S. and Sattarvand, J. (2015). "Long term production planning of open pit mines by ant colony optimization." European Journal of Operational Research **240**(3): 825-836.

Sturgul (1987). "How to determine the optimum location of in-pit movable crushers." **5**(2): 143-148.

Tabesh, M. and Askari-Nasab, H. (2011). "Two-stage clustering algorithm for block aggregation in open pit mines." Mining Technology **120**(3): 158-169.

Taheri, M. and Irannajad, M. (2009). "An approach to determine the locations of in-pit crushers in deep open-pit mines." International Multidisciplinary Scientific GeoConference: SGEM: Surveying Geology & mining Ecology Management **1**: 341.

Turnbull, D. and Cooper, A. (2010). "In-pit crushing and conveying (IPCC)-a tried and tested alternative to trucks: Part 1." AusIMM Bulletin(5): 60-64.

Tutton, D. and Streck, W. (2009). The application of mobile in-pit crushing and conveying in large, hard rock open pit mines. Mining Magazine Congress, Canada.

Weintraub, A., Pereira, M. and Schultz, X. (2008). "A priori and a posteriori aggregation procedures to reduce model size in MIP mine planning models." Electronic Notes in Discrete Mathematics **30**: 297-302.

Yarmuch, J., Epstein, R., Cancino, R., Peña, J. C. J. I. J. o. M., Reclamation and Environment (2017). "Evaluating crusher system location in an open pit mine using Markov chains." **31**(1): 24-37.

Yarmuch, J. L., Brazil, M., Rubinstein, H. and Thomas, D. A. (2019). "Optimum Ramp Design in Open Pit Mines." Computers & Operations Research.

# APPENDIX
## MATLAB Codes

This appendix includes the MATLAB codes developed for the implementation of the optimization framework presented in this research. The codes presented here include the data reading and preparation, clustering and determination of cluster's precedence, binary programming mathematical model and the scripts for plotting and result analysis. First, the block model's data is read from a .txt file containing the block model's attributes, including each block's coordinates, rock type and copper grade. Second, the block model is aggregated into clusters, and the mining precedence among those clusters is determined. Next, the clusters in the block model are fed into the mathematical model; the coefficient matrices are also generated in this step, and the problem is solved by the IBM ILOG CPLEX optimization toolbox. The CPLEX optimization toolbox for MATLAB must be added to the MATLAB path before run the binary linear programming model. To reduce the redundancy, only the code for the third model (BILP for HAC) is presented here. Plotting and result analysis scripts are developed for personal use on the display of the results. The code is presented in a series of scripts and functions, which are later run sequentially on the main script for the whole calculation.

The input parameters are read from a Microsoft EXCEL file, however, it can also be directly modified from the MATLAB code. To display the value of the parameters in this appendix, all the parameters' values are assigned in the functions and scripts.

Functions 1 to 7, functions 10 to 13 must be run in the presented order, script 2 is used when the problem is solved by CPLEX. Scripts S3 to S9 are used for plotting purposes and can be run after running function 13. Figure A1 presents the flowchart of the optimization model and how the functions are connected

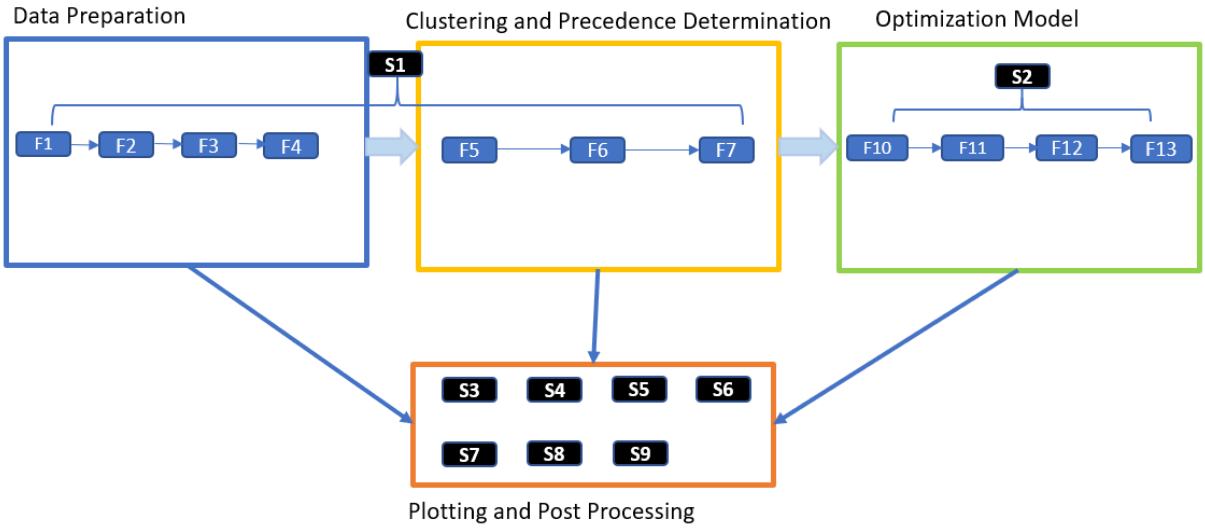



Figure A1. Flowchart of the optimization framework (MATLAB functions)

## S1. Preparing block model at the cluster level

```
% A script used to create and plot the clusters and leave the results in
% the workspace so that they can be saved
% Date: October, 2020
%--------------------------------------------------------------------------
clear
clc
tic
fclose('all');
load(['Param_sz',num2str(30)]);
% Size = 30;
ConveyorSetting = 'HAC';

for ang = 0:45:315 %315
    LevelTop = 30; LevelBottom = 35;
        ConvSpot = [];
    for LevelLoop = LevelTop:LevelBottom
        %% Open block model file
        % Divide the pit by its levels
        [LevelData] = f_OpenPitData();
        %% Rotate the conveyor side and find the nearest conveyor spot
        [BlockData,ConvSpot] = f_Rotate(LevelData,ang,LevelLoop,ConvSpot);
    end
        %% ramp slot for conventional conveyor
        if ~strcmp(ConveyorSetting,'HAC')
            slope = 20;
            [ConvSpot_slope] = ...
            f_ConvWall_slope(slope,ConvSpot,LevelTop,LevelBottom);
         % Calculate extra excavation tonnage for slot level by level
             ConvWidth = 0.2;  % 0.2*50(block width, m) = 10 m
            [SlotTon] = ...
            f_SlotTon(ConvSpot,ConvSpot_slope,ConvWidth,LevelTop,LevelBottom);
            SlotTon_rot.(['ang',num2str(ang)]) = SlotTon;
         % Update conveyor slot
         ConvSpot = ConvSpot_slope;
        else
            SlotTon_rot = [];
        end
    for LevelLoop = LevelTop:LevelBottom
```

```
        %% Agglomerative Hierarchical Clustering
            % open blocks of the target level
            [BlockData,~] = f_Rotate(LevelData,ang,LevelLoop,ConvSpot);

            StartStep = Param.(strcat('ang',num2str(ang)))(LevelLoop,1);
            ExpanFactor = Param.(strcat('ang',num2str(ang)))(LevelLoop,2);

            [AdjBlockMat,AdjClusterMat,Clusters,SmAjMatrix] = f_Clustering...
                (BlockData,ConvSpot,LevelLoop,Size,ExpanFactor,StartStep);
        %% Merge small cluster
        [Clusters,Clusters_data,BlockCoor,BasisBlockIndex] = f_MergeSmall...
     (Clusters,AdjClusterMat,Size,SmAjMatrix,BlockData);

        %% Create Clusters data structure                  %  AdjClusterList,...
        RefOPEX = 1.5;
        [ClustersData] = f_DataStructure(Clusters_data,...
                                         BlockCoor,...
                                         BlockData,...
                                         LevelLoop,...
                                         BasisBlockIndex,...
                                         RefOPEX, ...
                                         ConvSpot);

 %% Horizontal Precedence
        [ClustersData] = f_HorizontalPrecedence...
        (ClustersData,ConvSpot,AdjBlockMat,LevelLoop,Size);


        eval(['PitData.Level',num2str(LevelLoop),' = ClustersData',';']);
        clearvars -except Size PitRot ang ClustersData PitData ...
        Conv* Level* Param SlotTon_rot
    end

%% Verticl Precedence
% Give the top level null VP value
PitData.(strcat('Level',num2str(LevelTop)))(1).VP = [];

% Top-Down
for LevelLoop = LevelTop+1:LevelBottom  % Start from the level below to the top
    [PitData] = f_VerticalPrecedence(PitData,LevelLoop,ConvSpot,Size);
end
%%
ConvSpot_rot.(['ang',num2str(ang)]) =  ConvSpot;
disp(['Calculating conveyor side rotation angle at ',num2str(ang),' degree']);
toc
  eval(['PitRot.ang',num2str(ang),' = PitData',';']);
end
 save(['PitRot_sz',num2str(Size),'.mat'],'PitRot','SlotTon_rot','ConvSpot_rot')
%% plots
ang = 180;
level = LevelTop;
ConvSpot = ConvSpot_rot.(['ang',num2str(ang)]);

[figure1 figure2] = plot_Clusters(ang,level,PitRot,ConvSpot);
clearvars level
```

## F1. f_OpenPitData

```
% Purpose: Open the block model's data from a txt file, and store the blocks'
% data bench by bench into a matlab structure
% Dingbang Liu, September 2020
%--------------------------------------------------------------------------
% Inputs
%--------------------------------------------------------------------------
%   The file "finalpit.txt" with 6 columns:
```

```
%   Z(index),Y(index),X(index),RockCode,Grade%
%--------------------------------------------------------------------------
% Outputs
%--------------------------------------------------------------------------
%   LevelData = A structure with the blocks' data for each bench.
%--------------------------------------------------------------------------

%% Open file and data
% Divide the pit by levels
function [LevelData] = f_OpenPitData()
fid = fopen('finalpit.txt');
 block=cell2mat(textscan(fid,'%f %f %f %f %f','Delimiter',','));
%  % block: Z,Y,X,RockCode,Grade%
  % divide block model by its level

  LevelMin = min(block(:,1));
  LevelMax = max(block(:,1));
  LevelLoop = LevelMin;
   level_data=[];k=0;

   % create z_level
   for BlockLoop = 1:length(block)

        if block(BlockLoop,1) == LevelLoop  % The LevelLoop-th level
            k=k+1;
            level_data(k,:)=block(BlockLoop,2:5); % Y X RockCode Grade%

        else
            eval(['LevelData.Level',num2str(LevelLoop),'=','[level_data]',';']);
            level_data = [];
            LevelLoop = LevelLoop + 1;
            k = 0;
        end
   end
% the last level
eval(['LevelData.Level',num2str(LevelLoop),'=','[level_data]',';']);
clearvars BlockLoop k LevelLoop level_data
end
```

## F2. f_Rotate

```
% Purpose: generate the coordination of ramp slot for each level
%
% Date: Aug 2020
% Dingbang Liu
%--------------------------------------------------------------------------
% Inputs
%--------------------------------------------------------------------------
% LevelData = A structure with the blocks' data for each bench
% ang = Conveyor wall rotation angle
% LevelLoop = Level index from the top level to the bottom level
% ConvSpot = The conveyor spots' coordinates for all levels under a rotation angle
%--------------------------------------------------------------------------
% outputs
%--------------------------------------------------------------------------
% BlockData = The block data with rotated coordinates
% ConvSpot = The conveyor spots' coordinates for all levels under a rotation angle

function [BlockData,ConvSpot] = f_Rotate(LevelData,ang,LevelLoop,ConvSpot)

    rad = ang*pi/180; % Counter clockwise
    rot = [cos(rad),-sin(rad);sin(rad),cos(rad)];
    BlockData = LevelData.(['Level',num2str(LevelLoop)]);
    BlockData(:,[1 2]) = BlockData(:,[2 1]); % exchenge the first column(x,y)
    Bench_rotate = [BlockData(:,[1,2])*(rot'),BlockData(:,[3:4])];

    % find the ConvSpot of tangency of conveyor wall
    % if point of tangancy are more than one, find the middle one
```

```
    ymin = min(Bench_rotate(:,2)) + 1; %  id is the index  k(id), index 1 is the relaxation
    % find the index of all points of tangency
    % use round to ignore the accurate of operation
    id = find(round(Bench_rotate(:,2),2) <= round(ymin,2));
    ConvSpot(LevelLoop,:) = [mean(BlockData(id,1))-.5*sind(ang),min(BlockData(id,2))-
.5*cosd(ang),-LevelLoop * 0.8]; % X Y coordinate

    clearvars id ymin_p rad rot
end
```

## F3. f_LeastSqLine

```
% Purpose: generate the fitting line based on the tangent points of all levels
% Date: April 2020
% Dingbang Liu
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% OriginalXYZ = A n*3 array that denote the coordinate of the tangent points
%-------------------------------------------------------------------------
% outputs
%-------------------------------------------------------------------------
% x1,y1,z = the coordination of conveyor line at each level

function [x1,y1,z] = f_LeastSqLine(OriginXYZ)
L=length(OriginXYZ(:,1));
x=OriginXYZ(:,1);
y=OriginXYZ(:,2);
z=OriginXYZ(:,3);
F=[ones(1,L);z'];
   % no weights
M=F*F';
N=F*x;
O=F*y;

%    % apply weights
%  M=F*w*F';
%  N=F*w*x;
%  O=F*w*y;
%   % w is a L*L diagnal matrix
A=(M\N)';
B=(M\O)';
x1=(A(2)*z'+A(1))';
y1=(B(2)*z'+B(1))';
```

## F4. f_ConvWall_slope

```
% Purpose: generate the coordination of ramp slot for each level
% Date: Aug 03, 2020
% Dingbang Liu
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% slope = the slope of conveyor
% Coneveyor location of all levels under a specific rotation
% LevelTop = The index of the top level
% LevelBottom = The index of the bottom level
%-------------------------------------------------------------------------
% outputs
%-------------------------------------------------------------------------
% ConvSpot_slope = the coordination of ramp slot for each level
```

```
function [ConvSpot_slope] = f_ConvWall_slope(slope,ConvSpot,LevelTop,LevelBottom)


%% Give each point a weight
% lambda = NPV_rot(:,3);
% lambda(end)= []; % delete the grand total row
% lambda = lambda(lambda>=0); % delete the negative NPV
% lambda = lambda/max(lambda); % normalization
% data([length(data(:,1));-length(lambda)+1:length(data(:,1));],4) = lambda; % store in matrix
% lambda = data(:,4);
% w= diag(lambda); % weights matrix

%% least squre multivarible(arg X Y)
ConvSpot_data = ConvSpot(LevelTop:LevelBottom,:);
L=length(ConvSpot_data(:,1));
x=ConvSpot_data(:,1);
y=ConvSpot_data(:,2);
z=ConvSpot_data(:,3);
F=[z';ones(1,L)];
   % no weights
M=F*F';
N=F*x;
O=F*y;

%    % apply weights
%  M=F*w*F';
%  N=F*w*x;
%  O=F*w*y;

A=(M\N)';
B=(M\O)';
x1=(A(1)*z'+A(2))';
y1=(B(1)*z'+B(2))';

% plot3(x1,y1,z,'r','LineWidth',2)

% Calculate the original fitting line gradient
vector1 = [x1(1)-x1(2),y1(1)-y1(2),z(1)-z(2)];
% sin angle
angle = vector1(3)/norm(vector1);
angle = rad2deg(asin(angle));

% to adjust the gradient of line to 'grad' degree
% the x,y coordinate with multiplied by a extension coefficient k
k = sqrt((1/(sind(slope))^2-1)/(1/(sind(angle))^2-1));
% the origin point is the lowest point(bottom)
x2= k.*(x1-x1(end))+x1(end);
y2= k.*(y1-y1(end))+y1(end);

ConvSpot_slope(LevelTop:LevelBottom,:) = [x2,y2,z];

% hold on
% plot3(x2,y2,z,'Color',[1 .6 .6],'LineWidth',4)


% verify if the gradient is 'grad'
vector2 = [x2(1)-x2(2),y2(1)-y2(2),z(1)-z(2)];
angle2 = vector2(3)/norm(vector2);
angle2 = rad2deg(asin(angle2));

%% Plot
% title('Straight Conveyor Wall Fitting Line ');
% view(2)
% view(-20,10)

%% Slowering the slope based on conveyor spot in other level
% i = length(data(:,1)); %the bottom level
% x2= k.*(x1-x1(i))+x1(i);
% y2= k.*(y1-y1(i))+y1(i);
% % plot3(x2,y2,z,'y','LineWidth',2)
```

```
end
```

## F5. f_clusterting

```
% Purpose: Clustering blocks into larger mining units level by level using
% Hierarchical Agglomorative Clustering (HAC) algorithm
% Dingbang Liu, September 2020
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% BlockData = A structure with the blocks' data for a specific bench
% ConvSpot = The conveyor spots' coordinates for all levels under a rotation angle
% LevelLoop = Level index from the top level to the bottom level
% Size = The target size of the cluster (# of blocks)
% ExpanFactor = Input parameter for cluster size control, intial value is 1
% StartStep = Input parameter for cluster size control, intial value is 1
%-------------------------------------------------------------------------
% Outputs
%-------------------------------------------------------------------------
% AdjBlockMat = A pairwise matrix that contains blocks' adjacent relations
% AdjClusterMat = A pairwise matrix that contains the blocks adjacent relations
% Clusters = Cells contains the index of blocks that in the same cluster
% SmAjMatrix = A pairwise matrix that contains blocks' similarity value,
%               while considering blocks' adjacency
%-------------------------------------------------------------------------

function [AdjBlockMat,AdjClusterMat,Clusters,SmAjMatrix] = f_Clustering...
    (BlockData,ConvSpot,LevelLoop,Size,ExpanFactor,StartStep)


    Num = length(BlockData); % # of blocks for this bench %

     % Bottom-up fashion, from the lowest level to surface %
    count = 0; % Count the times of clustering
    epsilon = 0.000001;    % Avoid the divisor to be 0 %
    MinSize = Size*.8;

     %% set the weight of each parameter, original weights are all 1
     [W_RT,W_Dir,W_Dist, W_Gr] = deal(1,10,48,1);

     % Same RT: 1; Different RT: penalty
     % Rock type penalty, the less the more powerful
     RTPenalty = 0;

    MaxNumCluster = ceil(Num/Size); % The maximum # of clusters

    %% Clustering
    % Each block is a cluster from beginning %
    % Store the information of each cluster into a cell %

    Clusters = num2cell(1:Num); % Initial # cluster = # blocks
    Xindex = BlockData(:,1);
    Yindex = BlockData(:,2);
    % deplicate vector to squre matrix
    Xmatrix = Xindex(:,ones(1,Num));
    Ymatrix = Yindex(:,ones(1,Num));
    % Distance pairwise matrix
    DistMatrix = sqrt((Xmatrix - Xmatrix') .^ 2 + ...
    (Ymatrix - Ymatrix') .^ 2);
    MaxDist = max(DistMatrix(:)); % Find the largest distance %
    clearvars Xmatrix Ymatrix
    % Adjacent pairwise matrix.(Adjacent:1; Otherwise:0) %
    AdjBlockMat = DistMatrix;
    % Ajacent: 1; Otherwise: 0 %
```

```
AdjBlockMat(AdjBlockMat > 1) = 0;
% diagonal set to 0
AdjBlockMat = AdjBlockMat .* (1-eye(length(AdjBlockMat)));
% Original cluster adjacent matrix is equal to block's
AdjClusterMat = AdjBlockMat;

%%  Mining direction pairwise matrix
%  (Distance to the optimum conveyor location(OCL))
DistOPL_X = Xindex - ConvSpot(LevelLoop,1);
DistOPL_Y = Yindex - ConvSpot(LevelLoop,2);
DistOPL = sqrt(DistOPL_X .^ 2 + DistOPL_Y .^ 2);
DirMatrix = DistOPL(:,ones(1,Num));
DirMatrix = abs(DirMatrix - DirMatrix');
MaxDir = max(DirMatrix(:));
clearvars DistOPL_X DistOPL_Y XMat YMat

%% Grade %
GrVector = BlockData(:,4);
GrMat = GrVector(:,ones(1,Num));
GrMatrix = (GrMat-GrMat').^2;
MaxGr = max(GrMatrix(:));
clearvars GrVector GrMat

%% Rock type %
RTVector = BlockData(:,3);
RTMat = RTVector(:,ones(1,Num));
RTMatrix = double(RTMat ~= RTMat');  % Same RT: 0; Otherwise: 1 %
  % Same RT: 1; Otherwise: Penalty %
RTMatrix = (RTPenalty - 1) * RTMatrix + 1;
MaxRT = max(RTMatrix(:));

%% Similarity matrix %
DistMatrix(DistMatrix < epsilon) = epsilon;
DirMatrix(DirMatrix < epsilon) = epsilon;
GrMatrix(GrMatrix < epsilon) = epsilon;

% Normalisation/Weighting %
DistMatrix = (DistMatrix / MaxDist) .^ W_Dist; % Distance
DirMatrix = (DirMatrix / MaxDir ) .^ W_Dir;    % Direction
GrMatrix = (GrMatrix / MaxGr) .^ W_Gr;       % Grade
RTMatrix = (RTMatrix / MaxRT) .^ W_RT;
SMatrix = RTMatrix ./ (DistMatrix .* DirMatrix .* GrMatrix);
SMatrix = SMatrix .*(1-eye(Num)); % Not cluster the same block %
SmAjMatrix = SMatrix .* AdjClusterMat; % Similarity and Adjacent %
% clearvars RTMatrix DistMatrix DirMatrix GrMatrix

%% Create Clusters %
NumCluster = Num; % Number of clusters = Number of blocks %

if Size >1

    while NumCluster > MaxNumCluster
        % Find the pair of most similar ajacent clusters %
        [MaxSM MaxIndex] = max(SmAjMatrix(:));
        if MaxSM == 0 % when all elements in similarity matrix are 0
            break;
        end
        % Column number, cluster 2
        Cluster_J = ceil(MaxIndex/Num);
        %mod(MaxIndex,Num);  % Row number, cluster 1
        Cluster_I = MaxIndex - (Cluster_J - 1) * Num;
        % Quit clustering when the merging size is no less than the max size %
        if length(Clusters{Cluster_I}) + length(Clusters{Cluster_J}) > Size
            % Not merge the two cluster together anymore %
            SmAjMatrix(Cluster_I,Cluster_J) = 0;
            SmAjMatrix(Cluster_J,Cluster_I) = 0;
        else
            % Merge the two clusters %
            % Update the similarity matrix using min value(Complete Link) %
            SMatrix(Cluster_I,:) = ...
            min(SMatrix(Cluster_I,:),SMatrix(Cluster_J,:));
```

```
                SMatrix(:,Cluster_I) = SMatrix(Cluster_I,:)';
                SMatrix(Cluster_I,Cluster_I) = 0;% Not merge the same cluster %

                % Update the similarity matrix using max value %
                AdjClusterMat(Cluster_I,:) =...
                max(AdjClusterMat(Cluster_I,:),AdjClusterMat(Cluster_J,:));
                AdjClusterMat(:,Cluster_I) = AdjClusterMat(Cluster_I,:)';
                SmAjMatrix(Cluster_I,:) = ...
                SMatrix(Cluster_I,:) .* AdjClusterMat(Cluster_I,:);
                SmAjMatrix(:,Cluster_I) = 0;

               % The other cluster is not considered anymore
                SMatrix(Cluster_J,:) = 0;
                SMatrix(:,Cluster_J) = 0;
                AdjClusterMat(Cluster_J,:) = 0;
                AdjClusterMat(:,Cluster_J) = 0;
                SmAjMatrix(Cluster_J,:) = 0;
                SmAjMatrix(:,Cluster_J) = 0;

               % Merge the two cluster
                Clusters{Cluster_I} = [Clusters{Cluster_I} Clusters{Cluster_J}];
                Clusters{Cluster_J} = [];
                NumCluster = NumCluster - 1;
                count = count + 1;
            end

                if count >= StartStep
                    ClustersSize = cellfun('size',Clusters,2);
                    smlCluster = find(ClustersSize < MinSize & ClustersSize > 0);
                    SMatrix(smlCluster,:) =  SMatrix(smlCluster,:) .* ExpanFactor;

                    SMatrix(:,smlCluster) = SMatrix(smlCluster,:)';
                end
        end
    end

    AdjCluster = AdjClusterMat .* (1-eye(length(AdjClusterMat)));

        clearvars Cindex W_Dist W_Dir W_Gr W_RT MaxMumCluster Num

end
```

## F6. f_MergeSmall

```
% Purpose: Merge clusters smaller than a certain threshold to the adjacent clusters
% Dingbang Liu, September 2020
%--------------------------------------------------------------------------
% Inputs
%--------------------------------------------------------------------------

% Clusters = Cells contains the index of blocks that in the same cluster
% AdjMatrix = A pairwise matrix that contains the clusters' adjacent relations
% Size = The target size of the cluster (# of blocks)
% SmAjMatrix = A pairwise matrix that contains blocks' similarity value,
%               while considering blocks' adjacency
% BlockData = A structure with the blocks' data for a specific bench
%--------------------------------------------------------------------------
% Outputs
%--------------------------------------------------------------------------

% Clusters = Cells contains the index of blocks that in the same cluster
% Clusters_data = Blocks' index within a cluster
% BlockCoor = An array of block cluster within a cluster
% BasisBlockIndex =Tthe first block that are merged in a cluster
%--------------------------------------------------------------------------
% Cluster_J is he small cluster need to be merged
% Cluster_I is the smallest cluster among all adjacent clusters of Cluster_J
% merge Cluster_J and Cluster_J=I
```

```
%%
function [Clusters,Clusters_data,BlockCoor,BasisBlockIndex] = f_MergeSmall...
     (Clusters,AdjMatrix,Size,SmAjMatrix,BlockData)

 MinSize = Size * .8;

% Find small cluster index
ClustersSize = cellfun('size',Clusters,2);
index = find(ClustersSize > 0 & ClustersSize < MinSize);

for ClusterLoop = 1:length(index)
    % Blocks index within this small cluster
    SmallCluster = Clusters{index(ClusterLoop)};
    SmAjMatrix(SmallCluster,:) = 1;
    SmAjMatrix(:,SmallCluster) = SmAjMatrix(SmallCluster,:)';

    %% find the smallest adjacent cluster
    SmAjMatrix = AdjMatrix .* SmAjMatrix;
    % Adjacent Cluster Index
    AdjClusterIndex = find(max(SmAjMatrix(SmallCluster,:),[],1) == 1);
    % Remove the cluster itself
    AdjClusterIndex = AdjClusterIndex(AdjClusterIndex ~= index(ClusterLoop));
    % Find the adjacent cluster with smallest size
    [~,SmallestIdx] = min(ClustersSize(AdjClusterIndex));

    Cluster_I = AdjClusterIndex(SmallestIdx);  % the smallest adjacent cluster
    Cluster_J = index(ClusterLoop);
    % Cluster_J = double(ClustersSize(SmallCluster) ~=0) * SmallCluster';
    % Small cluster index
        if isempty(Cluster_I)% when all elements in similarity matrix are 0
            continue
        end

    % Merge two cluster
    if length(Clusters{Cluster_I})+length(Clusters{Cluster_J}) <= ...
    Size*1.25||length(Clusters{Cluster_J}) < 5

        Clusters{Cluster_I} = [Clusters{Cluster_I} Clusters{Cluster_J}];
        Clusters{Cluster_J} = [];
        AdjMatrix(Cluster_J,:) = 0;
        AdjMatrix(:,Cluster_J) = 0;
 %       NumCluster = NumCluster - 1;
    end
end

    % Delete null cells
    % Save the blocks data into Clusters_data
    % Clusters_data: index for the level, X index, Y index
    Cindex = cellfun(@isempty, Clusters) == 0; % find the index of null cell
    Clusters_data = Clusters(Cindex)';  % block index for the level

    for ClusterLoop = 1:length(Clusters_data)
        % blocks index with cluster(ClusterLoop)
        idx = cell2mat(Clusters_data(ClusterLoop));
        BlockCoor(ClusterLoop,1) = {BlockData(idx,1)}; % Block X index
        BlockCoor(ClusterLoop,2) = {BlockData(idx,2)}; % Block Y index
    end
    BasisBlockIndex = find(Cindex==1);

 end
```

## F7. f_DataStructure

```
% Purpose: Calculate Cluster Economical Value and create clusters' data strcture
```

```
% Dingbang Liu, September 2020
%--------------------------------------------------------------------------
% Inputs
%--------------------------------------------------------------------------
% Clusters_data = cells that store block's index within in each cluster
% BlockCoor = Blocks' coordinate
% BlockData = Array of blocks' attributes (Coordinates, rock type, grade)
% LevelLoop = Level index from the top level to the bottom level
% BasisBlockIndex = The first block that are merged in a cluster
% RefMiningC = Mining reference cost, exclude trucking and conveying costs
% ConvSpot = Coneveyor location of all levels under a specific rotation
%--------------------------------------------------------------------------
% Outputs
%--------------------------------------------------------------------------
% ClustersData = Clusters' attributes stored in a MATLAB structure
%--------------------------------------------------------------------------


function [ClustersData] = f_DataStructure(Clusters_data,...
                                          BlockCoor,...
                                          BlockData,...
                                          LevelLoop,...
                                          BasisBlockIndex,...
                                          RefMiningC,...
                                          ConvSpot)

    ClustersData = struct('BlockIdx',       Clusters_data,...
                          'BlockCoor',      0,...
                          'XCentroid',      0,...
                          'YCentroid',      0,...
                          'Level',          0,...
                          'BasisBlock',     0,...
                          'Destination',    0,...
                          'MiningC',           0,...
                          'CEV',            0,...
                          'AvgGr',          0,...
                          'RockCode',       0,...
                          'Tonnage',        0,...
                          'TonnageOre',     0);

    ClusterIndex = 1;
   for ClusterLoop = 1 : length(Clusters_data)

    % Index_Block: the index of blocks within a cluster
    Index_Block = cell2mat(Clusters_data(ClusterLoop));

    % Distinguish the ore and wasste block by its rock type
    Logic_Ore = BlockData(Index_Block,3) == 55|BlockData(Index_Block,3) == 50;
    Logic_Waste = BlockData(Index_Block,3) ~= 55 & BlockData(Index_Block,3) ~= 50;

    ClustersData(ClusterLoop).BlockCoor = cell2mat(BlockCoor(ClusterLoop,:));
    ClustersData(ClusterLoop).XCentroid = mean(BlockData(Index_Block,1));
    ClustersData(ClusterLoop).YCentroid = mean(BlockData(Index_Block,2));
    ClustersData(ClusterLoop).Level = LevelLoop;
   % ClustersData(ClusterLoop).ClusterCentroid.ZI = LevelLoop;

    ClustersData(ClusterLoop).BasisBlock = BasisBlockIndex(ClusterLoop);
    ClustersData(ClusterLoop).Destination = ConvSpot(LevelLoop,:);
    ClustersData(ClusterLoop).RockCode = BlockData(Index_Block,3);

    % Formula: ReferenceMiningC + 10% increase for each lower level ...
    % + Horizontal distance to ConvSpot
    % Horizontal Distance (HD) from cluster centroid to ConvSpot
    Xc = ClustersData(ClusterLoop).XCentroid;
    Yc = ClustersData(ClusterLoop).YCentroid;
    HD = sqrt((Xc-ConvSpot(LevelLoop,1))^2 + (Yc-ConvSpot(LevelLoop,2))^2).*40;
    Cost_HD = HD*0.001;
    ClustersData(ClusterLoop).HD2Spot = HD;
    ClustersData(ClusterLoop).MiningC = RefMiningC *+ Cost_HD;
    MiningC = ClustersData(ClusterLoop).MiningC;
    ClustersData(ClusterLoop).AvgGr = mean(BlockData(Index_Block(Logic_Ore),4));
```

```
    ClustersData(ClusterLoop).AvgGr(isnan(ClustersData(ClusterLoop).AvgGr))=0;
    ClustersData(ClusterLoop).Tonnage = ...
    sum(1.0e+5 * (2.21 * Logic_Ore + 1.8 * Logic_Waste));
    Tonnage = ClustersData(ClusterLoop).Tonnage;
    ClustersData(ClusterLoop).TonnageOre = sum(1.0e+5 * (2.21 * Logic_Ore ));
    % Economic cluster value
    % sigma(TonOre*Price*grade - TonOre*CostP ...
    % -(TonW+TonO)*CostM) - Operation cost(mining,transporting)
    ClustersData(ClusterLoop).CEV = ...
    (15000174.*BlockData(Index_Block,4)-642600)'* Logic_Ore - MiningC .* Tonnage;

  end
clearvars Index_Block Logic_Ore Logic_Waste

end
```

## F8. f_HorizontalPrecedence

```
% Purpose: Determine the cluster's precedence relations in the same level
% Dingbang Liu, September 2020
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% ClustersData = Clusters' attributes stored in a MATLAB structure
% ConvSpot = Coneveyor location of all levels under a specific rotation
% AdjBlockMat = Pairwise matrix denote whether two blocks are adjacent
% LevelLoop = Level index from the top level to the bottom level
% Size = The target size of the cluster (# of blocks)

%-------------------------------------------------------------------------
% Outputs
%-------------------------------------------------------------------------
% ClustersData = Clusters' attributes stored in a MATLAB structure
%   with an added field of horizontal precedence clusters' indeces
%-------------------------------------------------------------------------

function [ClustersData] =
f_HorizontalPrecedence(ClustersData,ConvSpot,AdjBlockMat,LevelLoop,Size)
%% Find the set of adjacent clusters for each cluster

if Size > 1
    % update the cluster adjacent relations based on block's
      AdjCluster1 = zeros(length(ClustersData),length(AdjBlockMat));
      AdjClusterMat = zeros(length(ClustersData)); % preallocate matrix
    % Adjacent matrix merge in row

    for ClusterLoop = 1:length(ClustersData)
        BlocksIndex = ClustersData(ClusterLoop).BlockIdx;    % The index of blocks within a
cluster
        % BasisBlocks = cat(2,ClustersData.BasisBlock);       % Read the column(.BasisBlock)
of the struct
        AdjCluster1(ClusterLoop,:) = max(AdjBlockMat(BlocksIndex,:));
    end

    % Adjacent matrix merge in column
    for ClusterLoop = 1:length(ClustersData)
      BlocksIndex = ClustersData(ClusterLoop).BlockIdx;
      AdjClusterMat(:,ClusterLoop) = max(AdjCluster1(:,BlocksIndex),[],2);

    end

else
    AdjClusterMat = AdjBlockMat;     % Each block itself is a cluster
end
```

```
    clearvars AdjCluster1 BlocksIndex ClusterLoop
    % Exclude each cluster itself from the adjecant matrix
    AdjClusterMat = AdjClusterMat - eye(length(AdjClusterMat));


  % Save the adjacent clusters number into ClustersData
  for ClusterLoop = 1:length(ClustersData)
     ClustersData(ClusterLoop).AdjacentCluster = find(AdjClusterMat(:,ClusterLoop) == 1);
  end

    %% Calculate the distance from each cluster centroid to the conveyor ConvSpot

    Xdist = cat(1,ClustersData.XCentroid)-ConvSpot(LevelLoop,1);
    Ydist = cat(1,ClustersData.YCentroid)-ConvSpot(LevelLoop,2);
    Dist = sqrt(Xdist.^ 2 + Ydist.^ 2);

%% Find the set of clusters which are closer to the conveyor

    for ClusterLoop = 1:length(ClustersData)
        AdjIdx = ClustersData(ClusterLoop).AdjacentCluster;
        ClustersData(ClusterLoop).HP =
ClustersData(ClusterLoop).AdjacentCluster(Dist(AdjIdx)<Dist(ClusterLoop));
    end
end
```

## F9. f_VerticalPrecedence

```
% Purpose: Determine the cluster's precedence relations from one upper level
% Dingbang Liu, September 2020
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% PitData = Clusters' attributes for all levels stored in a MATLAB structure
% ConvSpot = Coneveyor location of all levels under a specific rotation
% LevelLoop = Level index from the top level to the bottom level
% Size = The target size of the cluster (# of blocks)

%-------------------------------------------------------------------------
% Outputs
%-------------------------------------------------------------------------
% PitData = Clusters' attributes for all levels stored in a MATLAB structure
%   with an added field of horizontal precedence clusters' indeces
%-------------------------------------------------------------------------

function [PitData] = f_VerticalPrecedence(PitData,LevelLoop,ConvSpot,Size)


    %% Cluster Loop
    % Cluster numbers for that level
     ClNum = length(PitData.(strcat('Level',num2str(LevelLoop))));
    % Set of vertical precedence store in cell array VP
    VP = cell(ClNum,3);

for ClusterLoop = 1:ClNum
        % Get the block coordinate inside a certain block
        % BlockCoor = PitData.Level**(##).BlockCoor
        BlockCoor = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).BlockCoor;
        % BlockCoor = ClustersData(1).BlockCoor;
        XI = BlockCoor(:,1);
        YI = BlockCoor(:,2);     % Get the block coordinate in this cluster

        % Get the centroid coordination of the this cluster (Target cluster)
        Xc = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).XCentroid;
        Yc = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).YCentroid;

        % Find the boundry of a cluster
        Bd1 = boundary(XI,YI,1);
```

```
        if isempty(Bd1)
            Bd1 = 1:length(XI);
        else
             Bd1(end,:) = [];  % Delete the last index (duplicate with the first)
        end

        X1 = XI(Bd1,1);
        Y1 = YI(Bd1,1);

        % Find the precedence blocks of the boundry block for the upper level
        % Block precedence relationship using pattern 1:9
        % Expand the matrix to the end
        id_upper = [X1-1,Y1-1;
                    X1-1,Y1  ;
                    X1-1,Y1+1;
                    X1,  Y1-1;
                    X1,  Y1  ;
                    X1,  Y1+1;
                    X1+1,Y1-1;
                    X1+1,Y1  ;
                    X1+1,Y1+1];

        % Find the boundry of precedence blocks for the upper level
        Bd2 = boundary(id_upper(:,1),id_upper(:,2),1);
        BdX_upper = id_upper(Bd2,1);
        BdY_upper = id_upper(Bd2,2);


        % Open the upper level clusters
        CenX_upper = cat(1,PitData.(['Level',num2str(LevelLoop-1)]).XCentroid);
        CenY_upper = cat(1,PitData.(['Level',num2str(LevelLoop-1)]).YCentroid);
        %% 1. Find which cluster's centroid is inside/on the boundry
        IN1 = inpolygon(CenX_upper,CenY_upper,BdX_upper,BdY_upper);
        VP(ClusterLoop,1) = {find(IN1 ==1)};

        clearvars IN CenX_upper CenY_upper

    if Size>1

        % Find which blocks from the upper level are inside the boundry
        % load X,Y coordinate of the upper blocks

        UpperBlocks = cat(1,PitData.(['Level',num2str(LevelLoop-1)]).BlockCoor);
        IN2 = inpolygon(UpperBlocks(:,1),UpperBlocks(:,2),BdX_upper,BdY_upper);

        % Cluster numbers for the upper level
        ClNum_Up = length(PitData.(['Level',num2str(LevelLoop-1)]));
         LwB = 1;UpB =0; % block index within a cluster


        % Cluster Loop for the upper level: ClusterLoop1
        for ClusterLoop1 = 1:ClNum_Up
          % Nmubers of blocks for Cluster(ClusterLoop1)
          Clustersize = ...
          length(PitData.(['Level',num2str(LevelLoop-1)])(ClusterLoop1).BlockIdx);
          UpB = LwB + Clustersize - 1; % block upper index within a clutser
          % Nmuber of blocks from that cluster inside the boundary
          % sum of the binary varible inside a cluster
          NumInBd = sum(int8(IN2([LwB:UpB])));
          if NumInBd ~=0  % Cluster from the upper level has precedence blocks
          %% 2. Distance from target cluster centroid to the conveyor

              % Check if this cluster is on back side
              % Calculate the diatance from the cluster centroid to the conveyor
              % Back side if closer than target cluster centroid to the conveyor

              % load the centroid of the cluster(ClusterLoop1)
             CenX = PitData.(['Level',num2str(LevelLoop-1)])(ClusterLoop1).XCentroid;
             CenY = PitData.(['Level',num2str(LevelLoop-1)])(ClusterLoop1).YCentroid;
             Dist_sqr = (CenX-ConvSpot(LevelLoop,1))^2 + (CenY-ConvSpot(LevelLoop,2))^2;
```

```
            DistRe_sqr = (Xc-ConvSpot(LevelLoop,1))^2 + (Yc-ConvSpot(LevelLoop,2))^2;

            if Dist_sqr < DistRe_sqr        % Closer to ConvSpot than targrt cluster
               VP(ClusterLoop,2) = {[cell2mat(VP(ClusterLoop,2));ClusterLoop1]};
            end

            %% 3. Ratio of blocks inside the boundary to the tatal
            %     block number for a cluster
             Ratio = NumInBd/Clustersize;
            if Ratio>=0.4
              VP(ClusterLoop,3) = {[cell2mat(VP(ClusterLoop,3));ClusterLoop1]};
            end
          end
          LwB = UpB+1;
          VP(ClusterLoop,4)= ...
          {union(cell2mat(VP(ClusterLoop,2)),cell2mat(VP(ClusterLoop,3)))};

       end
     end

    % Write set of vertical precedence to pit data
     for ClusterLoop = 1:ClNum
       if Size > 1
            PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).VP = ...
                cell2mat(VP(ClusterLoop,4));
       else  % Cluster Size = 1     classic 9 precedence case
            PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).VP = ...
                cell2mat(VP(ClusterLoop,1));
     %       ClustersData(ClusterLoop).VP = cell2mat(VP(ClusterLoop,4))
       end
     end
end
end
```

## S2. Mathematical model

```
% A script used to create and plot the clusters and leave the results in
% the workspace so that they can be saved
% Date: October, 2020
%-------------------------------------------------------------------------
%% cluster size 20
% load('PitRot8.9.mat');

clear;
Size = 20;
Gap = 0.01;
load(['PitRot_sz',num2str(Size),'.mat']);

%% Conveyor setting
% HAC: high angle conveyor
% Slot: designated ramp slot (with exta waste)
ConveyorSetting = 'HAC';

for ang = 0:45:315  % Pit rotation loop, from 0 to 315, step size = 45
      ConvSpot = ConvSpot_rot.(['ang',num2str(ang)]);
    Periods = 10;
    %% MIP formula input parameters
    % transportation costs vector
     [f_c,LvlNum,Num,Periods,VarLen,LevelSize] = ...
     f_TranportCostMatix(Periods,ang,PitRot,ConvSpot);

     % Cluster Economic Value
     [f_v, Cluster_Ton, Cluster_Ore, Cluster_Gr] = ...
     f_CLEV(PitRot,ConveyorSetting, SlotTon_rot, ang,Num,Periods,VarLen);
```

```
    % CPLEX input1
     [Aineq1,bineq1,Aeq1,beq1,sostype,sosind,soswt,lb,ub,ctype,x0] = ...
     f_MIQP_input1_value(PitRot,ConveyorSetting,SlotTon_rot, ang,Num,LvlNum,Periods,VarLen);

    % CPLEX input2
     [Aineq2,bineq2,Aeq2,beq2] = f_MILP_input2_FLP(LvlNum,Num,Periods,VarLen,LevelSize,ang);

      Aineq = [Aineq1;Aineq2];
      bineq = [bineq1;bineq2];
     Aeq = [Aeq1;Aeq2];
     beq = [beq1;beq2];
     f = f_v + f_c; % Coefficient of Objective Function-Linear Item

     %% milp Caculation
     tic

     options = cplexoptimset('cplex');
     options.mip.tolerances.mipgap = Gap;

    [x,fval,exitflag,output] = ...
    cplexmilp(f, Aineq, bineq, Aeq, beq, sostype, sosind, soswt, lb, ub, ctype, x0, options);


disp(['Conveyor slot rotation angle at ', num2str(ang),' degree ',...
    ' The maximum NPV for this scenario is ',num2str(-fval),' M$']);
       %% Output the result
      x_Mat = reshape(x([1:Num * Periods]),Num,Periods); % Cluster extraction plan
      x2p_Mat = reshape(x([2 * Num * Periods+1: 2 * Num * Periods + LvlNum * Num *
Periods]),Periods,LvlNum*Num);
      y_Mat = reshape(x(VarLen-2*LvlNum*Periods + 1 : VarLen-LvlNum*Periods),LvlNum,Periods);
% Crusher reloction
      z_Mat = reshape(x(VarLen-LvlNum*Periods + 1 : VarLen),LvlNum,Periods); % Slot slice
extraction plan
       % Cluster economic value with discount
       fx_Mat = reshape(f([1:Num * Periods]),Num,Periods);
       fc_Mat = reshape(f_c([2 * Num * Periods+1: 2 * Num * Periods + LvlNum * Num *
Periods]),Periods,LvlNum*Num);
       % Cluster economic value with discount
       fz_Mat = reshape(f_v(VarLen-LvlNum*Periods + 1 : VarLen),LvlNum,Periods);

      CostT =  f_c' * x;  % Transportation cost matrix: LvlNum*Periods
      CostT_Vector = sum(fc_Mat' .* x2p_Mat');
      % eval(['result.ang',num2str(ang),'.x = x_Mat',';']);
      result.(['ang',num2str(ang)]).x = x_Mat;
      result.(['ang',num2str(ang)]).y = y_Mat;
      result.(['ang',num2str(ang)]).z = z_Mat;
      result.(['ang',num2str(ang)]).NPV = - fval;
      result.(['ang',num2str(ang)]).CostT = sum(CostT);
      disp(['Total material handling cost is ', num2str(sum(CostT))]);
      % {DCF = Discounted CLEV + Transportation cost + Relocation cost}in period t
      result.(['ang',num2str(ang)]).DCF = - sum(x_Mat.* fx_Mat)- CostT_Vector -
sum(z_Mat.*fz_Mat);
      result.(['ang',num2str(ang)]).runtime = output.time;
 toc


 eval(['result_sz',num2str(Size), '.ang',num2str(ang),'= result.ang',num2str(ang),';']);

 clearvars -except Gap Size ConveyorSetting SlotTon_rot result* ConvSpot* PitRot NPV Level*
Ub* Lb* Disc Periods f Cluster_Ton output Cluster_CEV

end
 save('result_sz.mat',strcat('result_sz',num2str(Size)),'-append')

% save('resultNew_sz.mat',strcat('result_sz',num2str(Size)),'-append')
```

## F10. f_CLEV

```
% Purpose: Generate coefficient vector for cluster economic values (CLEV)
% Dingbang Liu, October 2020
%--------------------------------------------------------------------------
% Inputs
%--------------------------------------------------------------------------
% PitRot = block model at the cluster level under a specific conveyor rotation
% ConveyorSetting = the type of conveyor:'HAC'-high angle conveyor, or'Slot'- conventional
conveyor
% SlotTon_rot = An array for conveyor slot tonnage of all levels under a specific rotation
% ang = The conveyor rotation angle
% Num = Number of total mining units (clusters) within the UPL
% Periods = Mine life
% VarLen = The total length of the decision variables
%--------------------------------------------------------------------------
% Outputs
%--------------------------------------------------------------------------
% f_value = A vector of cluster economic value
% Cluster_Ton = A vector of cluster total tonnage
% Cluster_Ore = A vector of cluster ore tonnage
% Cluster_Gr = A vector of cluster average ore blending grade
%--------------------------------------------------------------------------

function [f_value, Cluster_Ton, Cluster_Ore, Cluster_Gr] = ...
    f_CLEV(PitRot,ConveyorSetting, SlotTon_rot, ang,Num,Periods, VarLen)

     LevelTop = 30;
     LevelBottom = 35;

     Disc = 0.08;   % Discount rate
     CostReloc = 1; % relocation cost 1M$

LevelSize = [];
     PitData = PitRot.(strcat('ang',num2str(ang)));
     for LevelLoop = LevelTop:LevelBottom
         % the number of clusters for each level
         Size = length(PitData.(strcat('Level',num2str(LevelLoop))));
         LevelSize(LevelLoop,1) = Size;
         % Cumulative size
         LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
     end
     %Precedence constraint matrix
     PrecedenceMatrix = zeros(Num,Num);

     i=1;
     hp=[];vp =[];

      for LevelLoop = LevelTop:LevelBottom
          for ClusterLoop = 1:LevelSize(LevelLoop)
                % Tranfer from level index to pit index
                hp = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).HP;
                hp = sum(LevelSize(19:LevelLoop-1))+ hp;
                hp=hp(:);
            if LevelLoop ~= LevelTop
                vp = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).VP;
                vp = sum(LevelSize(19:LevelLoop-2))+ vp;
                vp=vp(:);  % change all to column vector
            end

            PrecedenceMatrix(i,[hp;vp]) = 1;
            NumPre(i,1) = length([hp;vp]);
            % PrecedenceMatrix(i,i) = -length([hp;vp]);
            i=i+1; % the number of the cluster
              % sum(LevelSize(19:LevelLoop-1)) + ClusterLoop;
          end
      end

     clearvars Bench hp i LevelLoop Size vp
    %% Cluster Economic Value (coefficient of objective function)
```

```
    Cluster_CEV = [];
    Cluster_Ton = [];
    Cluster_Ore = [];
    Cluster_Gr = [];

    for LevelLoop = LevelTop:LevelBottom
        % the number of clusters for each level
        eval(['temp1 = cat(1,PitData.Level',num2str(LevelLoop),'.CEV)',';']);
        eval(['temp2 = cat(1,PitData.Level',num2str(LevelLoop),'.Tonnage)',';']);
        eval(['temp3 = cat(1,PitData.Level',num2str(LevelLoop),'.TonnageOre)',';']);
        eval(['temp4 = cat(1,PitData.Level',num2str(LevelLoop),'.AvgGr)',';']);
        Cluster_CEV = [Cluster_CEV;temp1];
        Cluster_Ton = [Cluster_Ton;temp2];
        Cluster_Ore = [Cluster_Ore;temp3];
        Cluster_Gr = [Cluster_Gr;temp4];
    end
    %% Discounted Cluster Economic Value vector
      LvlNum = LevelBottom - LevelTop + 1; % the total numbers of level
      Cluster_Ton = Cluster_Ton / 1e+6;   % ton -> million ton
      Cluster_Ore =  Cluster_Ore / 1e+6;
      f_value = repmat(Cluster_CEV,Periods,1) / 1e+6; % CEV repetitive vector. ton -> million
ton
       PeriodMat = fix([0:Periods * Num-1]'/Num); % Period repatitive vector,
       Discount = ones(Periods * Num,1)./((1+Disc) .^ PeriodMat);

       % Discounted cost for slot excavation & crusher relocation
       PeriodMat1 = fix([0:Periods * LvlNum-1]'/LvlNum); % Period repatitive vector
       Discount1 = 1./((1+Disc) .^ PeriodMat1);
       % relocation cost dicounted vector
       CostReloc_v = CostReloc .* Discount1;
       % slot cost discounted vector
       if ConveyorSetting == 'HAC'
          f_value = [-f_value .* Discount; zeros(VarLen - Periods*Num -
Periods*LvlNum,1);CostReloc_v];
       else
          SlotCost = SlotTon_rot.(['ang',num2str(ang)])(LevelTop:LevelBottom,2)/ 1e+6;
          f_value = [-f_value .* Discount; zeros(Periods * Num,1); repmat(SlotCost,Periods,1)
.* Discount1];
       end
       clearvars Bench hp i LevelLoop Size vp temp* Discount PeriodMat*
end
```

## F11. f_TransportCostMatrix

```
% Purpose: Generate coefficient matrix for material handling costs
% Dingbang Liu, October 2020
%--------------------------------------------------------------------------
% Inputs
%--------------------------------------------------------------------------
% Periods = Mine life
% ang = The conveyor rotation angle
% PitRot = block model at the cluster level under a specific conveyor rotation
% ConvSpot = Coneveyor location of all levels under a specific rotation
%--------------------------------------------------------------------------
% Outputs
%--------------------------------------------------------------------------
% LvlNum = Numbers of levels
% Num = Number of total mining units (clusters) within the UPL
% Periods = Mine life
% VarLen = The total length of the decision variables
% LevelSize = the number and cumulative number of clusters at each level
%--------------------------------------------------------------------------

function [f_c,LvlNum,Num,Periods, VarLen, LevelSize] =
f_TranportCostMatix(Periods,ang,PitRot,ConvSpot)
LevelTop = 30;
LevelBottom = 35;
Disc = 0.08;
```

```
Cost_TruckHori = 0.2 ; % horizontal hauling cost $/Km
Cost_TruckV_up = 0.05 ;
Cost_TruckV_down = 0.02 ;
Cost_ConvyVert = 0.01 ; % conveying cost $/m

PitData = PitRot.(strcat('ang',num2str(ang)));
%% LevelNumber
PitData_cat = [];
LevelSize = [];
     PitData = PitRot.(strcat('ang',num2str(ang)));
     for LevelLoop = LevelTop:LevelBottom
         % the number of clusters for each level
         Size = length(PitData.(strcat('Level',num2str(LevelLoop))));
         LevelSize(LevelLoop,1) = Size;
         % Cumulative size
         LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
         % concatenation all clusters
         PitData_cat = [PitData_cat;PitData.(['Level',num2str(LevelLoop)])];
     end
     % total number of clusters
     Num = sum(LevelSize(:,1));
     LvlNum = LevelBottom - LevelTop + 1; % the total numbers of level
     VarLen = 2 * Periods * Num + Num * Periods * LvlNum + 2 * Periods * LvlNum;

f_Cost = zeros(Num*LvlNum*Periods,1);
%% Distances with respect to CCL(candidate crusher locations)
    % Truck- HoriDistance(cluster centroid to CCL),VertDistance(cluster centroid to CCL)
    % Conveyor- VertDistance(CCL centroid to pit exit)
    V_Level = [PitData_cat.Level]';  % Level vector of all clusters
    V_Ton = [PitData_cat.Tonnage]';
    % initial variables
    V_TruckHoriD = zeros(Num,LvlNum); % Hori(x,y) distance from cluster centroid to each
level's CCL
    V_TruckVertD = zeros(Num,LvlNum); % Vert(z)   distance from cluster centroid to each
level's CCL
    V_ConvyVertD = zeros(Num,LvlNum); % Vert(z)   distance from each level's CCL to pit exit
    k = 0;
     % unit transpot cost for each cluster with crusher in level LevelLoop
     for LevelLoop = LevelTop:LevelBottom % Crusher location level
         k = k + 1;
         CrusherLoc = ConvSpot(LevelLoop,:);
         V_CoorDif = [[PitData_cat.XCentroid]' - CrusherLoc(1),[PitData_cat.YCentroid]' -
CrusherLoc(2)];
         V_TruckHoriD(:,k) = sqrt(sum(V_CoorDif.^2,2)).*50;
         V_TruckVertD(:,k) = (V_Level - LevelLoop) * 40;
         V_ConvyVertD(:,k) = (LevelLoop - LevelTop)  * 40;
     end

     %% cost
     TotalCostUnit = Cost_TruckHori * V_TruckHoriD/1000 + Cost_TruckV_up * V_TruckVertD
.*(V_TruckVertD>0)...
         + Cost_TruckV_down * -V_TruckVertD .*(V_TruckVertD<0)+ Cost_ConvyVert * V_ConvyVertD;
     CostP1 = repmat(V_Ton,1,LvlNum).*TotalCostUnit ./1e+6; % transportation cost(M$) for
subsequent periods ;
     CostP = reshape(CostP1,[],1); % transform the matrix to vector

     for t = 1 : Periods
         CostP = CostP ./ (1+Disc);
         f_Cost((Num*LvlNum*(t-1)+1:Num*LvlNum*t),1) = CostP;
     end

     f_c = [zeros(2*Num*Periods,1); f_Cost; zeros(2*LvlNum*Periods,1)];
end
```

## F12. f_ BILP_input1_value

```
% Purpose: Generate coefficient matrices and right-hand side vectors for
% the constraints group 1
% Dingbang Liu, October 2020
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% PitRot = block model at the cluster level under a specific conveyor rotation
% BlockData = A structure with the blocks' data for a specific bench
% ConveyorSetting = the type of conveyor:'HAC'-high angle conveyor, or'Slot'- conventional
conveyor
% SlotTon_rot = An array for conveyor slot tonnage of all levels under a specific rotation
% ang = The conveyor rotation angle
% Num = Number of total mining units (clusters) within the UPL
% LvlNum = Numbers of levels
% Periods = Mine life
% VarLen = The total length of the decision variables
%-------------------------------------------------------------------------
% Outputs
%-------------------------------------------------------------------------
% Aineq1 = Double matrix for linear inequality constraints
% bineq1 = Double column vector for linear inequality constraints
% Aeq1 = Double matrix for linear equality constraints
% beq1 = Double column vector for linear equality constraints
% sostype,sosind,soswt = [];
% ctype = String with possible char values 'B': Binary variable(0-1 variable)
% x0 = [];  Double column vector of initial point of x

%-------------------------------------------------------------------------

function [Aineq1,bineq1,Aeq1,beq1,sostype,sosind,soswt,lb,ub,ctype,x0] = ...
f_BILP_input1_value(PitRot,ConveyorSetting, SlotTon_rot, ang,Num,LvlNum,Periods,VarLen)

%     LevelTop = 30;
%     LevelBottom = 35;
%     LbMc = 25;      % yearly mining capacity in Mt
%     UbMc = 30;
%     LbPc = 3;       % yearly processing capacity in Mt
%     UbPc = 6;
%     LbGr = 0.5;     % Minimum processing grade(%)
%     UbGr = 1.1;
%     Disc = 0.08;    % Discount rate
%     CostReloc = 1; % relocation cost 1M$

LevelSize = [];
     PitData = PitRot.(strcat('ang',num2str(ang)));
     for LevelLoop = LevelTop:LevelBottom
         Size = length(PitData.(strcat('Level',num2str(LevelLoop))));
         LevelSize(LevelLoop,1) = Size;
         % Cumulative size
         LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
     end

     %Precedence constraint matrix
     PrecedenceMatrix = zeros(Num,Num);
     i=1;
     hp=[];vp =[];

      for LevelLoop = LevelTop:LevelBottom
          for ClusterLoop = 1:LevelSize(LevelLoop)
                % Tranfer from level index to pit index
                hp = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).HP;
                hp = sum(LevelSize(19:LevelLoop-1))+ hp;
                hp=hp(:);
            if LevelLoop ~= LevelTop
                vp = PitData.(['Level',num2str(LevelLoop)])(ClusterLoop).VP;
                vp = sum(LevelSize(19:LevelLoop-2))+ vp;
                vp=vp(:);  % change all to column vector
            end

            PrecedenceMatrix(i,[hp;vp]) = 1;
            NumPre(i,1) = length([hp;vp]);
```

```
            % PrecedenceMatrix(i,i) = -length([hp;vp]);
            i=i+1; % the number of the cluster
              % sum(LevelSize(19:LevelLoop-1)) + ClusterLoop;
          end
      end
     clearvars Bench hp i LevelLoop Size vp
    %% Cluster Economic Value (coefficient of objective function)
     Cluster_CEV = [];
     Cluster_Ton = [];
     Cluster_Ore = [];
     Cluster_Gr = [];

     for LevelLoop = LevelTop:LevelBottom
         % the number of clusters for each level
         eval(['temp1 = cat(1,PitData.Level',num2str(LevelLoop),'.CEV)',';']);
         eval(['temp2 = cat(1,PitData.Level',num2str(LevelLoop),'.Tonnage)',';']);
         eval(['temp3 = cat(1,PitData.Level',num2str(LevelLoop),'.TonnageOre)',';']);
         eval(['temp4 = cat(1,PitData.Level',num2str(LevelLoop),'.AvgGr)',';']);
         Cluster_CEV = [Cluster_CEV;temp1];
         Cluster_Ton = [Cluster_Ton;temp2];
         Cluster_Ore = [Cluster_Ore;temp3];
         Cluster_Gr = [Cluster_Gr;temp4];

     end
     %% Discounted Cluster Economic Value vector
       LvlNum = LevelBottom - LevelTop + 1; % the total numbers of level
       Cluster_Ton = Cluster_Ton / 1e+6;   % ton -> million ton
       Cluster_Ore =  Cluster_Ore / 1e+6;
       f_value = repmat(Cluster_CEV,Periods,1) / 1e+6; % CEV repetitive vector. ton -> million
ton
       PeriodMat = fix([0:Periods * Num-1]'/Num); % Period repatitive vector,
       Discount = ones(Periods * Num,1)./((1+Disc) .^ PeriodMat);

       % Discounted cost for slot excavation & crusher relocation
       PeriodMat1 = fix([0:Periods * LvlNum-1]'/LvlNum); % Period repatitive vector
       Discount1 = 1./((1+Disc) .^ PeriodMat1);
       % relocation cost dicounted vector
       CostReloc_v = CostReloc .* Discount1;

       clearvars Bench hp i LevelLoop Size vp temp* Discount PeriodMat*
    %% decision variables:
    % X = [x,y,z]
    % x: continuous variables, portion of cluster mined,size(Periods * Num,1)
    % y: binary variables,1 if precedent clusters are cleared,size(Periods * Num,1)
    % z: binary variables,1 if that level starts to be mined, size(Levels,1)

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %% Define Constraints Matrix
      StMC = zeros(Periods,VarLen);
      StPC = zeros(Periods,VarLen);
      StGrLb = zeros(Periods,VarLen);
      StGrUb = zeros(Periods,VarLen);

      StPre1_x = zeros(Periods * Num,Periods * Num);
      StPre1_xb = StPre1_x;

      StSlot1 = zeros(Periods * LvlNum, VarLen);
      StSlot2 = zeros(Periods, VarLen);


      % Slot Constraints (x part matrix)
      StSlot1_temp1 = zeros(LvlNum, Num);
        k1=1; % Initial cluster index within a level
        for LevelLoop = LevelTop:LevelBottom
           k2 = k1 + LevelSize(LevelLoop,1) -1;
           StSlot1_temp1(LevelLoop-LevelTop+1,k1:k2) = 1;
           k1 = k2;
        end
      clearvars k1 k2
```

```
      for t = 1:Periods

          s1 = (t-1)* Num + 1; % Initial index of Num loop in this period
          s2 = t * Num;        % Final index of Num loop in this period

          L1 = (t-1)* LvlNum + 1; % Initial index of LvlNum loop in this period
          L2 = t* LvlNum ;        % Final index of LvlNum loop in this period

          % Mining Capacity
          StMC(t,s1:s2) = Cluster_Ton;
          if strcmp(ConveyorSetting,'Slot') % ConveyorSetting == 'Slot'
            SlotTon = SlotTon_rot.(['ang',num2str(ang)])(LevelTop:LevelBottom,1);
          elseif strcmp(ConveyorSetting,'HAC') % ConveyorSetting == 'HAC'
            SlotTon = zeros(1,LvlNum);
          end
          StMC(t,Periods * Num * 2 + L1 : Periods * Num * 2 + L2) = SlotTon'; % slot
excavation tonnage

          % Processing Capacity
          StPC(t,s1:s2) = Cluster_Ore;

          % Grade Constraint
          StGrLb(t,s1:s2) = Cluster_Ore.* (LbGr-Cluster_Gr);
          StGrUb(t,s1:s2) = Cluster_Ore.* (Cluster_Gr - UbGr);


          % Precedence Constraints (x part matrix)
          StPre1_x(s1:s2,1:s2) = -repmat(PrecedenceMatrix,1,t);
          StPre2_x(s1:s2,1:s2) = repmat(diag(ones(Num,1)),1,t);

          %% Slot Excavation Constraints
          if strcmp(ConveyorSetting,'Slot') % ConveyorSetting == 'Slot'
              StSlot1(L1:L2,s1:s2) = StSlot1_temp1;
              StSlot1(L1:L2,[1:L2] + 2 * Num * Periods) = -
repmat(LevelSize(LevelTop:LevelBottom,1) .* eye(LvlNum),1,t);
              StSlot2 = sparse([zeros(LvlNum, 2 * Num *
Periods),repmat(diag(ones(LvlNum,1)),1,Periods)]);
              StSlot2_RHS = ones(LvlNum,1);
          elseif strcmp(ConveyorSetting,'HAC') % ConveyorSetting == 'HAC'
              StSlot1 = zeros(LvlNum * Periods, VarLen);
              StSlot2 = zeros(LvlNum, VarLen);
              StSlot2_RHS = zeros(LvlNum,1);
          end
      end



      % Precedence Constraints (y part matrix)
      % set diagonal element
       StPre1_xb(logical(eye(Num*Periods))) = repmat(NumPre,Periods,1);
       StPre2_xb = -diag(ones(Num*Periods,1));

      % change to sparse matrix
     StMC_s = sparse(StMC);
     StPC_s = sparse(StPC);
     StGr_s = sparse([StGrLb;StGrUb]);


     %% Reserve Constraint (x:size(Num,Num*Periods))
     StRev_s = sparse([repmat(diag(ones(Num,1)),1,Periods),zeros(Num,VarLen - Num*Periods)]);

     % StPre1 and StPre2 are both precedence constraints
     % (dicision variables [x,y,z]
     StPre1_s = [StPre1_x,StPre1_xb,zeros(Periods*Num, VarLen - 2*Num*Periods)];
     % x - y <= 0
     StPre2_s = [StPre2_x,StPre2_xb,zeros(Periods*Num,VarLen - 2* Num*Periods)];

     %% x'' constraints
     % x''<=0.5(x + y)
     Len_x2 = Periods * LvlNum * Num;
```

```
     StMH_x = zeros(Len_x2,Periods*Num);
     StMH_y = zeros(Len_x2,Periods*LvlNum);

     StMH_x2 = eye(Len_x2,Len_x2);

     for t = 1:Periods

         StMH_x((t-1)*Num*LvlNum+1 : t*Num*LvlNum, (t-1)*Num+1 : t*Num) =
repmat(eye(Num,Num),LvlNum,1);

     end

     for k = 1 : LvlNum*Periods
        StMH_y((k-1)*Num+1 : k*Num, k) = 1;
     end

     StMH1 = [-0.5*StMH_x, zeros(Len_x2,t*Num), StMH_x2, -0.5*StMH_y, zeros(Len_x2,t*LvlNum)];
     StMH2 = [ StMH_x,    zeros(Len_x2,t*Num), -StMH_x2,      StMH_y, zeros(Len_x2,t*LvlNum)];

     StMH1_s = sparse(StMH1);
     StMH2_s = sparse(StMH2);
     clearvars StMH1_x StMH1_x2 StMH1_y StMH1 StMH2
     %% Slot Excavation Constraints
     StSlot1_s = sparse(StSlot1);
     StSlot2_s = sparse(StSlot2);
     clearvars StPre StMC StPC StGr StSlot1 StSlot2 s1 s2 L1 L2 StPre_temp*

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% set right-hand side (bineq)%%%%%%%%%%%%
     LbMc_RHS = ones(Periods,1).* LbMc;
     UbMc_RHS = ones(Periods,1).* UbMc;
     LbPc_RHS = ones(Periods,1).* LbPc;
     UbPc_RHS = ones(Periods,1).* UbPc;
     % processing capacity is lower during the first two years
     UbPc_RHS(1) = UbPc_RHS(1).*0.7;
     UbPc_RHS(2) = UbPc_RHS(2).*0.8;

     StGr_RHS = zeros(Periods * 2,1);
     StPre1_RHS = zeros(Num * Periods,1);
     % MI_RHS = [0.99 .* repmat(NumPre,Periods,1);zeros(Num * Periods,1)];
     StPre2_RHS = zeros(Num * Periods,1);

     StMH_RHS = [zeros(Len_x2,1);ones(Len_x2,1)*1.5];
     StSlot1_RHS = zeros(LvlNum * Periods,1);

     %% MLP value assignment
     Aineq1 = [-StMC_s; StMC_s;     -StPC_s;  StPC_s;    StGr_s;    StRev_s;     StPre1_s;
StPre2_s;   StMH1_s; StMH2_s;      StSlot1_s];
     bineq1 = [-LbMc_RHS; UbMc_RHS; -LbPc_RHS; UbPc_RHS; StGr_RHS; ones(Num,1); StPre1_RHS;
StPre2_RHS;StMH_RHS;   StSlot1_RHS];

     Aeq1 = StSlot2_s;
     beq1 = StSlot2_RHS;

     lb = zeros(VarLen,1);
     ub = ones(VarLen,1);
     ctype = [repmat('B',1,VarLen)];

     sostype = [];
     sosind = [];
     soswt = [];
     x0 = [];
end
```

## F13. f_BILP_input2_FLP

```
% Purpose: Generate coefficient matrices and right-hand side vectors for
% the constraints group 2 for crusher location(Facility Location Problem, FLP)
```

```
% Dingbang Liu, October 2020
%-------------------------------------------------------------------------
% Inputs
%-------------------------------------------------------------------------
% LvlNum = Numbers of levels
% Num = Number of total mining units (clusters) within the UPL
% Periods = Mine life
% VarLen = The total length of the decision variables
% LevelSize = the number and cumulative number of clusters at each level
% ang = The conveyor rotation angle
%-------------------------------------------------------------------------
% Outputs
%-------------------------------------------------------------------------
% Aineq2 = Double matrix for linear inequality constraints
% bineq2 = Double column vector for linear inequality constraints
% Aeq2 = Double matrix for linear equality constraints
% beq2 = Double column vector for linear equality constraints
%-------------------------------------------------------------------------
function [Aineq2,bineq2,Aeq2,beq2] =
f_BILP_input2_FLP(LvlNum,Num,Periods,VarLen,LevelSize,ang)
% clearvars -except PitRot result* x_Mat ConvSpot_slope
LevelTop = 30;
LevelBottom = 35;

N = 2; % Consective periods for one location

    %% Set coefficient matrix (Left-hand side)
    % relocation constraints
    % y(i) = 1 if crusher locates in level i
    % x(i,t) = 1 if crusher locates in level i at period t

    % Create super diagnal matrix without period 1
    diag_super_t2 = diag(ones(LvlNum*(Periods-1),1),LvlNum);
    Const1_y = -eye(LvlNum*Periods) + diag_super_t2;

    Const1 = [zeros(LvlNum*Periods,VarLen - 2*LvlNum*Periods),Const1_y, -diag_super_t2];
    Const1(end-LvlNum+1:end,:) = [];

    Const2 = [zeros(LvlNum*Periods,VarLen - 2*LvlNum*Periods),-diag_super_t2, diag_super_t2];
    % y_t1 = z_t1
    Const_t1_y = [eye(LvlNum),zeros(LvlNum,LvlNum*(Periods-1))];
    Const_t1 = [zeros(LvlNum,VarLen - 2*LvlNum*Periods),-Const_t1_y,  Const_t1_y];

     clearvars Const1_y  Const_t1_y


    % 3. crusher number(1) constraint
    k= VarLen - 2 * Periods * LvlNum;
    Const3 = zeros(Periods,VarLen);
    for t = 1:Periods
        Const3(t,[(k+1):(k+LvlNum)]) = 1;
        k = k + LvlNum;
    end

%     % 4. Precendence constraint (Slot first, crusher later)
%     Const4_x = zeros(LvlNum*Periods,Num*Periods);
%         j = 1;
%         k = 1;
%     for LevelLoop = LevelTop:LevelBottom
%         Const4_x(j,k:LevelSize(LevelLoop,2)) = 1;
%         j = j + 1;
%         k = k + LevelSize(LevelLoop,1);
%     end
%     Const4 =
[Const4_x,zeros(LvlNum*Periods,Num*Periods),eye(LvlNum*Periods),zeros(LvlNum*Periods,LvlNum*Pe
riods)];
%     clearvars Const4_x j k

    % 5. Crusher can only relocate to lower levels
    Const5_y = zeros((Periods-1) * LvlNum,Periods* LvlNum);
    temp1 = ones(LvlNum);
```

```
    temp2 = tril(temp1); % Lower triangular matrix
    for t = 1:Periods-1
        Const5_y([(t-1)*LvlNum+1:t*LvlNum],[(t-1)*LvlNum+1:t*LvlNum]) = -temp2;
        Const5_y([(t-1)*LvlNum+1:t*LvlNum],[t*LvlNum+1:(t+1)*LvlNum]) = temp2;
    end
    Const5 = zeros((Periods-1)*LvlNum,VarLen);
    Const5(:,[VarLen-2*Periods*LvlNum+1 : VarLen-Periods*LvlNum]) = Const5_y;
    clearvars Const5_y temp*

    % 6. crusher stays at one level for at least N consecutive periods
    Const6_y = zeros(LvlNum,LvlNum*Periods);

    for j = 1:LvlNum
        temp = zeros(1,LvlNum);
        temp(j)= 1;
        Const6_y(j,:) = repmat(temp,1,Periods);
    end

    Const6 = [zeros(LvlNum, VarLen-2*Periods*LvlNum),-Const6_y, N.*Const6_y];

    %% Set right-hand side
    % (bineq)
    Const1_RHS = zeros(LvlNum*(Periods-1),1);
    Const2_RHS = zeros(LvlNum*Periods,1);
    Const4_RHS = zeros(LvlNum*Periods,1);
    Const5_RHS = zeros((Periods-1)*LvlNum,1);
    Const6_RHS = zeros(LvlNum,1);
    % (beq)
    Const_t1_RHS = zeros(LvlNum,1);
    Const3_RHS = ones(Periods,1);


    Aineq2 = [Const1; Const2; Const5; Const6]; % ;
    bineq2 = [Const1_RHS; Const2_RHS; Const5_RHS; Const6_RHS]; % ; Const6_RHS

    Aeq2 = [ Const_t1; Const3];
    beq2 = [ Const_t1_RHS; Const3_RHS];

end
```

## F14. plotcube

```
% Purpose: Plot cube(block) with a specific color

function plotcube(varargin)
% PLOTCUBE - Display a 3D-cube in the current axes
%
%   PLOTCUBE(EDGES,ORIGIN,ALPHA,COLOR) displays a 3D-cube in the current axes
%   with the following properties:
%   * EDGES : 3-elements vector that defines the length of cube edges
%   * ORIGIN: 3-elements vector that defines the start point of the cube
%   * ALPHA : scalar that defines the transparency of the cube faces (from 0
%             to 1)
%   * COLOR : 3-elements vector that defines the faces color of the cube
%
% Example:
%   >> plotcube([5 5 5],[ 2  2  2],.8,[1 0 0]);
%   >> plotcube([5 5 5],[10 10 10],.8,[0 1 0]);
%   >> plotcube([5 5 5],[20 20 20],.8,[0 0 1]);
% Default input arguments
inArgs = { ...
  [10 56 100] , ... % Default edge sizes (x,y and z)
  [10 10  10] , ... % Default coordinates of the origin point of the cube
  .7          , ... % Default alpha value for the cube's faces
```

```
  [1 0 0]       ... % Default Color for the cube
  };
% Replace default input arguments by input values
inArgs(1:nargin) = varargin;
% Create all variables
[edges,origin,alpha,clr] = deal(inArgs{:});
XYZ = { ...
  [0 0 0 0]  [0 0 1 1]  [0 1 1 0] ; ...
  [1 1 1 1]  [0 0 1 1]  [0 1 1 0] ; ...
  [0 1 1 0]  [0 0 0 0]  [0 0 1 1] ; ...
  [0 1 1 0]  [1 1 1 1]  [0 0 1 1] ; ...
  [0 1 1 0]  [0 0 1 1]  [0 0 0 0] ; ...
  [0 1 1 0]  [0 0 1 1]  [1 1 1 1]   ...
  };
XYZ = mat2cell(...
  cellfun( @(x,y,z) x*y+z , ...
    XYZ , ...
    repmat(mat2cell(edges,1,[1 1 1]),6,1) , ...
    repmat(mat2cell(origin,1,[1 1 1]),6,1) , ...
    'UniformOutput',false), ...
  6,[1 1 1]);
cellfun(@patch,XYZ(Bliek1ú et al.),XYZ{2},XYZ{3},...
  repmat({clr},6,1),...
  repmat({'FaceAlpha'},6,1),...
  repmat({alpha},6,1)...
  );
view(3);
```

## S3. Plot conveyor line

```
% A script used to plot the pit levels and conveyor lines (HAC)
% level by level for a spacific scenario (conveyor rotation angle)
% Dingbang Liu
% Date: April, 2020
figure;
[LevelData] = f_OpenPitData();

LevelTop = 30;
LevelBttom = 35;
for ang = 0:45:315
     t=0;
    for LevelLoop = LevelTop:LevelBttom % from upper level to lower level
            BlockC =  LevelData.(['Level',num2str(LevelLoop)]);
         % angel with maximum NPV
        % rotate all orientation with step=30 degree

             rad = - ang*pi/180;
             rot = [cos(rad),-sin(rad);sin(rad),cos(rad)];

                BlockC_rot = [BlockC(:,[1,2])*(rot'),BlockC(:,[3 4])]; % rotate y,x(first 2
column)
             [k,av] = convhull(BlockC_rot(:,[1,2]));
              pgon = polyshape(BlockC_rot(k,2),BlockC_rot(k,1)); % X Y

              fill3(BlockC(k,2),BlockC(k,1), ...
                   ones(length(k))*(11-LevelLoop).*0.8,[1 .5
.5],'FaceAlpha',.2,'FaceColor',[.7 .7 1]);

               BlockC_O=[];BlockC_W=[];

             hold on

            % bounding box
             [xlim,ylim] = boundingbox(pgon);
                xmax=max(xlim)+0.5;xmin=min(xlim)-0.5;
                ymax=max(ylim)+0.5;ymin=min(ylim)-0.5;
```

```
                xbox=[xmin,xmin,xmax,xmax];
                ybox=[ymin,ymax,ymax,ymin];
                % bbox=polyshape(xbox,ybox);
                % plot(bbox)
                ymin_p = min(BlockC_rot(k,1))+1; %  id is the index  k(id)
                id = find(round(BlockC_rot(k,1),2) <= round(ymin_p,2));
                point = [mean(BlockC_rot(k(id),2));mean(BlockC_rot(k(id),1))];

                            rad = rad;
            rot_R = [cos(rad),-sin(rad);sin(rad),cos(rad)]; % rotation for x1,y1
            xy1 = rot_R*[xbox;ybox];
            xbox=xy1(1,:);ybox=xy1(2,:);
             t=t+1;
            ConvSpot(t,:) = [rot*point;(11-LevelLoop)*0.8]';
            scatter3( ConvSpot(:,1),ConvSpot(:,2),ConvSpot(:,3),20,'filled')

        %    plot3( tan_P(:,1),tan_P(:,2),tan_P(:,3),'-o','Color','c','LineWidth',2)
% % plot bounding box
%             fill3(xbox,ybox,ones(4,1)*(11-level).*0.8 ...
%                   ,[0.5,1,.5],'FaceAlpha',.2);
%           % straight conveyor wall
%         scw = plot3([xbox(1) xbox(4)],[ybox(1),ybox(4)], ...
%                     [11-level,11-level].*0.8,'Color','k','LineWidth',2);
            %% disconting power
            % it is a function of distance to staright wall and depth
            % distance to conveyor wall: abs(y-ymin)
            % depth: (level-11)*40

    end
%% least squre multivarible(arg X Y)
[x1,y1,z] = f_LeastSqLine(ConvSpot);

plot3(x1,y1,z,'Color','k','LineWidth',2)

%% Slot

slope = 20;
[ConvSpot_slope] = f_ConvWall_slope(slope,ConvSpot);
% if ang==0
    text(ConvSpot(1,1),ConvSpot(1,2),ConvSpot(1,3)+0.5,[num2str(ang),'°'],'FontSize',16)
end

   view(3)
   axis equal
   axis tight
   axis off
```

## S4. Plot NPV comparison for different scenarios

```
% A script used to plot the NPV generate from different conveyor rotation
% scenario
% Dingbang Liu
% Date: October, 2020

LevelTop = 30;
LevelBottom = 35;
Periods = 10;
Size = 20;
load('result_sz.mat');
load(['PitRot_sz',num2str(Size)]);
eval(['result = result_sz',num2str(Size),';']);
```

```
for ang = 0:45:315
          LevelSize = [];
%           load(['PitRot_sz',num2str(Size)]);
%           load('result_sz')
     PitData = PitRot.(strcat('ang',num2str(ang)));
     for LevelLoop = LevelTop:LevelBottom
         % the number of clusters for each level
         ClusterNum = length(PitData.(strcat('Level',num2str(LevelLoop))));
         LevelSize(LevelLoop,1) = ClusterNum;
         LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
     end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% reset  period field
 for level = LevelTop:LevelBottom
    for n =1 : length(PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)]))
    PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).Period = [];
    end
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Write the extraction period for each cluster

 for t = 1:Periods
     x_Mat = result.(['ang',num2str(ang)]).x;
     Cl(:,1) = find(x_Mat(:,t)>0.99);    % x_Mat(:,t)==1, which clusters are mined in t
     level = LevelTop;
     for ClusterLoop = 1:length(Cl)  % clusters mined in period t
        % find cluster from which level
         while Cl(ClusterLoop,1)-LevelSize(level,2)> 0
            level = level+1;
         end
         Cl(ClusterLoop,2) = level;
         n = Cl(ClusterLoop,1) - LevelSize(level-1,2);  % cluster number within level
         % Write the period infomation into PitData
         PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).Period = t;
     end
   clearvars level PitData ClusterLoop LevelLoop ClsterNum Cl n s1 s2

 end
%  save('PitRot.mat','PitRot','result');
 clearvars -except Size Level* result ConvSpot PitRot NPV count x_Mat Top Bottom UbMc UbPc
Disc Periods f Cluster_Ton Cluster_CEV
end

%%% Plot NPV tonnage for each scenario
%% NPV: ang, NPV, Tonnage, Ore
NPV = [];
PitData_cat = [];
k=0;
for ang = 0:45:315
    k = k + 1;
    PitData_cat = [];
    PitData = PitRot.(strcat('ang',num2str(ang)));
    for level = LevelTop:LevelBottom
        PitData_cat = [PitData_cat;PitData.(['Level',num2str(level)])];
    end

    NPV(k,1)=ang;
      % Total tonnage (Mt)
    NPV(k,2)= sum(result.(['ang',num2str(ang)]).x,2)'* cat(1,PitData_cat.Tonnage)/1e+6;
      % ore tonnage  (Mt)
    NPV(k,3)= sum(result.(['ang',num2str(ang)]).x,2)'* cat(1,PitData_cat.TonnageOre)/1e+6;

      % Stripping ratio
    NPV(k,4) = NPV(k,2) ./ NPV(k,3);
      % npv
    NPV(k,5)= result.(['ang',num2str(ang)]).NPV/1000; % M$ -> B$

end
%% The total extraction tonnage by different conveyor side rotation angle
figure;
 hold on
```

```
 bar(NPV([1:k],1),NPV(:,2)/1e+6,0.4,'b');  %
%  bar(NPV([1:ii],1),NPV(:,4)/1e+6,0.4,'y');  %
 TotalOre = sum(cat(1,PitData_cat.TonnageOre))/1e+6;
 line([-10 325],[TotalOre TotalOre],'Color','red','LineStyle','--');

 xticks(NPV([1:k],1));
 xticklabels(string(NPV([1:k],1)));
 ylim([0,max(NPV([1:k],2)*1.2/1e+6)]);

 title('The total extraction tonnage by different conveyor side rotation angle')
 xlabel('Conveyor side rotation angle ()')
 ylabel('Tonnage(Mt)')

 set(gca,'fontsize',16,'FontWeight','normal');
 barlabel = string(roundn(NPV(:,2)/1e+6,-2)); % the second decimal place
 text(NPV(:,1),NPV(:,2)/1e+6,barlabel,'HorizontalAlignment','center',...
     'VerticalAlignment','bottom','fontsize',12,'FontWeight','normal');
% legend({'Waste','Ore','Total in-pit Ore'},'Location','southoutside',...
%      'fontsize',14,'FontWeight','normal','Orientation','horizontal');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% NPV comparison by scenarios(different rotation angle)
 figure;
 hold on
 bar(NPV([1:k],1),NPV(:,5),0.4);
 ylim([0,max(NPV([1:k],5)*1.2)]);
 xticks(NPV([1:k],1));
 xticklabels(string(NPV([1:k],1)));
% title('The pit NPV by different conveyor side rotation angle')
 xlabel('Conveyor side rotation angle ()')
 ylabel('NPV (Billion $)')
 set(gca,'fontsize',16,'FontWeight','normal');
 barlabel = sprintfc('%0.2f',NPV(:,5));
 text(NPV(:,1),NPV(:,5),barlabel,'HorizontalAlignment','center',...
     'VerticalAlignment','bottom','fontsize',16,'FontWeight','normal');
```

## S5. Plot production scheduling bar

```
% A script used to plot Production scheduling graph for a spacific scenario
% Dingbang Liu
% Date: October, 2020

ang = 180;
Disc = 0.08;
LevelTop = 30;
LevelBottom = 35;
Periods = 10;
Size = 20;
ConveyorSetting = 'HAC';
load(['PitRot_sz',num2str(Size)]);
load(['result_sz']);
eval(['result = result_sz',num2str(Size),';']);

PitData_cat = [];
PitData = PitRot.(['ang',num2str(ang)]);
% concatenation all clusters
    for level = LevelTop:LevelBottom
        PitData_cat = [PitData_cat;PitData.(['Level',num2str(level)])];
    end

% x_Mat = result.(strcat('ang',num2str(ang))).x;

Cluster_CEV = [PitData_cat.CEV]';
Cluster_Ton = [PitData_cat.Tonnage]';
Cluster_Ore = [PitData_cat.TonnageOre]';
Cluster_Gr = [PitData_cat.AvgGr]';

% Num = length(Cluster_CEV);
%  f = repmat(Cluster_CEV,Periods,1);
```

```
%  PeriodMat = fix([1:Periods * Num]'/Num);
%  Discount = ones(Periods * Num,1).*(1-Disc) .^ PeriodMat;
%  f = f .* Discount;
%  f_Mat = reshape(f,Num,Periods);
 %% calculate discounted cash flow,tonnage,NPV, ore tonnage for each period

 x_Mat = result.(['ang',num2str(ang)]).x;
 z_Mat = result.(['ang',num2str(ang)]).z;

 if ConveyorSetting ~= 'HAC'
    SlotTon = SlotTon_rot.(['ang',num2str(ang)])(LevelTop:LevelBottom,2);
 end

 result_DCF = result.(strcat('ang',num2str(ang))).DCF;
 result_ton = sum(x_Mat .* Cluster_Ton,1)'./1e+6; % + sum(z_Mat .* SlotTon,1)'./1e+6/2;
 result_cDCF = cumsum(result_DCF)'/1000; % cumulative DCF
cat('NPV.sz',num2str(Size)) = sum(result_DCF);
 result_Gr = ((Cluster_Ore.*Cluster_Gr)'*x_Mat) ./ (Cluster_Ore'*x_Mat);
 result_Gr =  result_Gr';
% calculate the tonnage of ore in block level
result_ore = (Cluster_Ore'*x_Mat)/1e+6;
result_ore = result_ore';
result_waste = result_ton-result_ore;
OriginPlot = [[1:Periods]',result_ton,result_ore,result_cDCF,result_Gr];

% export to excel
% xls_SchedulingData = [[1:10]';result_ton-result_ore,result_ore,result_cDCF/1000,result_Gr];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %% plot scheduling graph

 fig = figure;
 left_color = [0,0,0];
 right_color = [0,0,0];
 set(fig,'defaultAxesColorOrder',[left_color;right_color]);
 % title('The tonnage and profit graph by different periods')
 xlabel('Period')

 yyaxis left
 bar(result_ton,'b'); % tonnage of all mining materials
 hold on
 bar(result_ore,'y') % tonnage of ore blocks
 ylabel('Tonnage(Mt)')
 ylim([0 ceil(max(result_ton/10))*10]);

 yyaxis right
 % plot([1:10],result_profit,'-*g','LineWidth',2); % discounted cash flow
%  plot([1:Periods],result_cDCF,'-ok','LineWidth',2) % cumulative DCF
 ylabel('Value(B$)')
 ylim([0 1.500]);
 set(gca,'fontsize',16,'FontWeight','bold','LooseInset',get(gca,'TightInset'));

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %% Create triple yy-axis to show grade

 x3 = [1:Periods];
 y3 = result_Gr;
 cfig = get(gcf,'color');
pos = [0.1  0.1  0.7  0.8];
offset = pos(3)/10;
%Reduce width of the two axes generated by plotyy
pos(3) = pos(3) - offset/2;
%%

[ax,hlines(1),hlines(2)] = plotyy(x3,result_ton,x3,result_cDCF);
set(ax(2),'YTickLabel',[]);
%%
set(ax,'position',pos);
%Determine the position of the third axes
pos3=[pos(1) pos(2) pos(3)+offset pos(4)];
%Determine the proper x-limits for the third axes
limx1=get(ax(1),'xlim');
```

```
limx3=[limx1(1)   limx1(1) + 1.1*(limx1(2)-limx1(1))];
%Bug fix 14 Nov-2001: the 1.2 scale factor in the line above
%was contributed by Mariano Garcia (BorgWarner Morse TEC Inc)
limy3 = [0,1.2];
ax(3)=axes('Position',pos3,'box','off',...
   'Color','none','XColor','k','YColor','r',...
   'xtick',[],'xlim',limx3,'ylim',limy3,'yaxislocation','right');
hlines(2) = line(x3,result_cDCF,'Marker','o','Parent',ax(2),'LineWidth',2);
hlines(3) = line(x3,y3,'Color','r','Parent',ax(3),'LineWidth',2);
set(get(ax(3),'ylabel'),'string','Ore Blending Grade (%)') % set label
% line([limx1(2),limx3(2)],[limy3(1),limy3(1)],'parent',ax(3),'color','w','LineWidth',1.5);

set(gca,'fontsize',16,'FontWeight','bold','LooseInset',get(gca,'TightInset');
```

## S6. Plot block sequencing for the whole pit

```
% A script used to plot block sequencing for a spacific scenario
% Dingbang Liu
% Date: October, 2020

ang = 180; % The optimum conveyor rotation angle
LevelTop = 30;
LevelBottom = 35;
Periods = 10;
PitData = PitRot.(['ang',num2str(ang)]);

% %% reset period
%
%  for level = Top:Bottom
%     for n =1 : length(PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)]))
%     PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).Period = [];
%     end
% end

%  LevelSize = [];
%
%      for LevelLoop = LevelTop:LevelBottom
%          % the number of clusters for each level
%          Size = length(PitData.(strcat('Level',num2str(LevelLoop))));
%          LevelSize(LevelLoop,1) = Size;
%          LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
%      end

     x_Mat = result.(['ang',num2str(ang)]).x;
     % clusters mined completely in period t

     for ClusterLoop = 1:length(x_Mat)
         level = LevelTop;
        % find the complete extraction period t of each cluster
         temp = find(x_Mat(ClusterLoop,:)>0.9); % period of complete extraction and afterwords

         if ~isempty(temp)
            Cl_Period(ClusterLoop,1) = temp(1);
         else
            Cl_Period(ClusterLoop,1) = NaN;
         end


        % find cluster from which level
         while ClusterLoop-LevelSize(level,2)>0
            level = level+1;
         end

         n = ClusterLoop - LevelSize(level-1,2);  % cluster number within the level
         % Write the period infomation into PitData
```

```
        PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).Period =
Cl_Period(ClusterLoop,1);
    end
        clearvars level ClusterLoop LevelLoop Size Cl n s1 s2 temp
 PitData = PitRot.(strcat('ang',num2str(ang)));
  %% overall number ->> level number
  % save the periods to database(PitData)
    LevelSize = [];

    for LevelLoop = LevelTop:LevelBottom
        % the number of clusters for each level
        Size = length(PitData.(strcat('Level',num2str(LevelLoop))));
        LevelSize(LevelLoop,1) = Size;
        LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
    end
%% load conveyor spot for that angle
ConvSpot = [];
[LevelData] = f_OpenPitData();
for LevelLoop = LevelTop : LevelBottom
[~,ConvSpot] = f_Rotate(LevelData,ang,LevelLoop,ConvSpot);
end

tic
figure;
hold on
set(gca,'fontsize',18,'FontWeight','bold');
% title(['Production scheduling in with conveyor rotation of ',num2str(ang),'']);
xlabel('X Index');
ylabel('Y Index');
zlabel('Z Index');
 PitData_cat = [];

 for level = LevelTop:LevelBottom
     % write field 'level' to every cluster
    for i = 1:length(PitData.(['Level',num2str(level)]))
        PitData.(strcat('Level',num2str(level)))(i).Level = level;
    end
    % transfer level structure into cluster structure
    PitData_cat = [PitData_cat;PitData.(['Level',num2str(level)])];
%     % plot conveyor spot location
%     scatter3(ConvSpot(level,1),ConvSpot(level,2),-level*.8,100,'k')
 end
% plot conveyor line
   [x1,y1,z1] = f_LeastSqLine(ConvSpot([LevelTop:LevelBottom],[1:3]));
   plot3(x1,y1,z1,'linewidth',5,'Color','k')
 colorjet = jet(Periods);
 for i = 1:length(PitData_cat)
 BlockCoor = PitData_cat(i).BlockCoor;
    % Open each cluster centroid
    % CentroidX = PitData_cat(i).XCentroid;
    % CentroidY = PitData_cat(i).YCentroid;
    % Plot cluster's centroid and number

    % Plot blocks inside the cluster
        for j = 1:length(BlockCoor)
            x = BlockCoor(:,1);
            y = BlockCoor(:,2);
            z = PitData_cat(i).Level *(-.8);  % x,y, z coordinate
            t = PitData_cat(i).Period;
            if isnan(t)  % this cluster will never be mined
                c = [1,1,1];
            elseif isempty(t)
                c = [1,1,1];
            else
                c = colorjet(t,:);
            end
            plotcube([1 1 .8],[x(j)-0.5 y(j)-0.5 z],.5,c);
        end
 end
%  legend(t,{'1','2','3','4','5','6','7','8','9','10'})
axis equal
```

```
axis tight
        %% create legend
 legend off
 for t = 1:Periods
    sc(t)= scatter3(ConvSpot(level,1),ConvSpot(level,2),-
level*.8,1,colorjet(t,:),'s','filled');
 end
    lgd = legend(sc,{'1','2','3','4','5','6','7','8','9','10'},...
        'fontsize',14,'FontWeight','normal','Location','northeast');
    title(lgd,'Periods')
      legend('boxon')

      view(160,10)
```

## S7. Plot block sequencing for each level

```
% A script used to plot block sequencing level by level for a spacific
scenario
% Dingbang Liu
% Date: October, 2020

clearvars -except PitRot result* x_Mat ConvSpot_slope ConveyorSetting

ang = 180;
LevelTop = 30;
LevelBottom = 35;
Periods = 10;
PitData = PitRot.(strcat('ang',num2str(ang)));
slope = 20;
ConveyorSetting = 'HAC';
% x_Mat = result.(['ang',num2str(ang)]).x;

%% reset period
 for level = LevelTop:LevelBottom
    for n =1 : length(PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)]))
    PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).Period = [];
    end
end
  %% overall number ->> level number
  % save the periods to database(PitData)
     LevelSize = [];
     for LevelLoop = LevelTop:LevelBottom
         % the number of clusters for each level
         ClustersInLevel = length(PitData.(strcat('Level',num2str(LevelLoop))));
         LevelSize(LevelLoop,1) = ClustersInLevel;
         % cummulative size
         LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
     end
     x_Mat = result.(['ang',num2str(ang)]).x;
     x_Cum = cumsum(x_Mat,2); % cumulative sum of each row of x_Mat
     % clusters mined completely in period t
     for ClusterLoop = 1:length(x_Mat)
         level = LevelTop;
        % find the complete extraction period t of each cluster
         temp = find(x_Cum(ClusterLoop,:)>0.9); % period of complete extraction and afterwords
         if ~isempty(temp)
            Cl_Period(ClusterLoop,1) = temp(1);
         else
            Cl_Period(ClusterLoop,1) = NaN;
         end

        % find cluster from which level
         while ClusterLoop-LevelSize(level,2)>0
            level = level+1;
```

```
        end
        n = ClusterLoop - LevelSize(level-1,2);  % cluster number within the level
        % Write the period infomation into PitData
        PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).Period =
Cl_Period(ClusterLoop,1);
    end
        clearvars level ClusterLoop LevelLoop Size Cl n s1 s2 temp

%% load conveyor spot for that angle

[LevelData] = f_OpenPitData();
ConvSpot = [];
for LevelLoop = LevelTop : LevelBottom
[~,ConvSpot] = f_Rotate(LevelData,ang,LevelLoop,ConvSpot);
end

tic

 PitData = PitRot.(['ang',num2str(ang)]);


 for level = LevelTop:LevelBottom

     % write field 'level' to every cluster
    for ClusterLoop = 1:length(PitData.(['Level',num2str(level)]))
        PitData.(strcat('Level',num2str(level)))(ClusterLoop).Level = level;
    end
 end

 colorjet = jet(Periods);

 %% plot a series of figures level-by-level
 % gragh axis range ( the pit range of the top level)
 % initialize x,y bound
 xmin = 0;
 ymin = 0;
 xmax = 0;
 ymax = 0;
 TopCoor = [];

 for LevelLoop = LevelTop:LevelBottom
     PitData_level = PitData.(['Level',num2str(LevelLoop)]);
     t_slot = min([PitData_level.Period]); % the period of extacting slot slice
     fig = figure('Renderer', 'painters', 'Position', [10 10 900 600]);
     hold on
    for ClusterLoop = 1:length(PitData_level)
     BlockCoor = PitData_level(ClusterLoop).BlockCoor;
        % Open each cluster centroid
        % CentroidX = PitData_cat(i).XCentroid;
        % CentroidY = PitData_cat(i).YCentroid;
        % Plot cluster's centroid and number

        % Plot blocks inside the cluster
            for BlockLoop = 1:length(BlockCoor)
                x = BlockCoor(:,1);
                y = BlockCoor(:,2);
                z = PitData_level(ClusterLoop).Level *(-.8);  % x,y, z coordinate
                t = PitData_level(ClusterLoop).Period;

                if isnan(t)  % this cluster will never be mined
                   c = [1,1,1]; % set color to pure write
                elseif isempty(t)
                   c = [1,1,1]; % set color to pure write
                else
                   c = colorjet(t,:); % set color to corresponding period
                end
                plotcube([1 1 .8],[x(BlockLoop)-0.5 y(BlockLoop)-0.5 z],.5,c);
            end

           % find the pit bound for the top level
            if LevelLoop == LevelTop
```

```
                TopCoor = [TopCoor;BlockCoor];
            end

 %   scatter3(CentroidX,CentroidY,-28.4,20, 'filled','k')
 %   text(PitData.Level29(i).XCentroid,PitData.Level29(i).YCentroid,-
28.4,num2str(i),'FontSize',15)
    end

   %% plot slot & conveyor spot (if not HAC)
%    if strcmp(ConveyorSetting,'Slot')
%     ColorSlot = colorjet(t_slot,:);
%     plot2_ConveyorSlot_Slice(ColorSlot,ConvSpot_slope,LevelLoop,LevelTop,LevelBottom)
%     % plot the conveyor spot
%     [ConvSpot_slope] = f_ConvWall_slope(slope,ConvSpot,LevelTop,LevelBottom);
%     scatter3(ConvSpot_slope(LevelLoop,1),ConvSpot_slope(LevelLoop,2),-
LevelLoop*0.8+1,200,...
%         'w','filled','MarkerEdgeColor','k','LineWidth',2);
%    elseif strcmp(ConveyorSetting,'HAC') && LevelLoop ~= LevelBottom
%            scatter3(ConvSpot(LevelLoop,1),ConvSpot(LevelLoop,2)-cosd(ang),-
LevelLoop*0.8+1,150,...
%         'w','filled','MarkerEdgeColor','k','LineWidth',2);
%    else
%            scatter3(ConvSpot(LevelLoop,1),ConvSpot(LevelLoop,2),-LevelLoop*0.8+1,150,...
%        'w','filled','MarkerEdgeColor','k','LineWidth',2);
%
%    end
%
   %% Figure setting

    set(gca,'fontsize',18,'FontWeight','bold');
    title(['Level ',num2str(LevelLoop-LevelTop+1)]);
    xlabel('X Index');
    ylabel('Y Index');
    zlabel('Z Index');
    % Specify Axis Limits
    xmin = min(TopCoor(:,1));
    xmax = max(TopCoor(:,1));
    ymin = min(TopCoor(:,2));
    ymax = max(TopCoor(:,2));
   ax = gca();
   ax.YDir = 'reverse';
   ax.XDir = 'reverse';

   axis equal
   view(2)
   xlim([xmin-1 xmax+1]);
   ylim([ymin-1 ymax+1]);
  %  saveas(fig,['Slice_L',num2str(LevelLoop)],'tiffn')
 end
```

## S8. Plot block material flow

```
% A script used to plot crusher location and block material flow
% level by level for a spacific scenario (conveyor rotation angle)
% Dingbang Liu
% Date: October, 2020
clearvars -except PitRot result* x_Mat ConvSpot_slope ConveyorSetting
% load('PitRot8.9.mat')
ang = 180;
LevelTop = 30;
LevelBottom = 35;
Periods = 10;
PitData = PitRot.(strcat('ang',num2str(ang)));
slope = 20;
LvlNum = LevelBottom - LevelTop +1;
ConveyorSetting = 'HAC';
% x_Mat = result.(['ang',num2str(ang)]).x;
```

```
%% reset period

 for level = LevelTop:LevelBottom
    for n =1 : length(PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)]))
    PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).CrusherLoc = [];
    end
end

  %% overall number ->> level number
  % save the periods to database(PitData)

     LevelSize = [];
     for LevelLoop = LevelTop:LevelBottom
         % the number of clusters for each level
         ClustersInLevel = length(PitData.(strcat('Level',num2str(LevelLoop))));
         LevelSize(LevelLoop,1) = ClustersInLevel;
         % cummulative size
         LevelSize(LevelLoop,2) = sum(LevelSize(LevelTop:LevelLoop));
     end
        x_Mat = result_sz20.(['ang',num2str(ang)]).x;
        y_Mat = result_sz20.(['ang',num2str(ang)]).y;
        z_Mat = result_sz20.(['ang',num2str(ang)]).z;
        LocNum = sum(sum(z_Mat));
        Cl_Crusher = x_Mat*y_Mat';
     % cumulative sum of each row of x_Mat
     % clusters mined completely in period t

     for ClusterLoop = 1:length(Cl_Crusher)
         level = LevelTop;
        % find the complete extraction period t of each cluster
         temp = find(Cl_Crusher(ClusterLoop,:)>0.5); % period of complete extraction and
afterwords

         if ~isempty(temp)
            Cl_CrusherLoc(ClusterLoop,1) = temp(1);
         else
            Cl_CrusherLoc(ClusterLoop,1) = NaN;
         end

        % find cluster from which level
         while ClusterLoop-LevelSize(level,2)>0
            level = level+1;
         end

         n = ClusterLoop - LevelSize(level-1,2);  % cluster number within the level
         % Write the period infomation into PitData
         PitRot.(['ang',num2str(ang)]).(['Level',num2str(level)])(n).CrusherLoc =
Cl_CrusherLoc(ClusterLoop,1);
     end
         clearvars level ClusterLoop LevelLoop Size Cl n s1 s2 temp

%% load conveyor spot for that angle

[LevelData] = f_OpenPitData();
ConvSpot = [];
for LevelLoop = LevelTop : LevelBottom
[~,ConvSpot] = f_Rotate(LevelData,ang,LevelLoop,ConvSpot);
end

tic

 PitData = PitRot.(['ang',num2str(ang)]);

 for level = LevelTop:LevelBottom

     % write field 'level' to every cluster
    for ClusterLoop = 1:length(PitData.(['Level',num2str(level)]))
        PitData.(strcat('Level',num2str(level)))(ClusterLoop).Level = level;
    end
    % transfer level structure into cluster structure
    % plot conveyor spot location
```

```
 end

 colormap = pink(LvlNum)/1.5+0.3;

 %% plot a series of figures level-by-level

 % gragh axis range ( the pit range of the top level)
 % initialize x,y bound
 xmin = 0;
 ymin = 0;
 xmax = 0;
 ymax = 0;
 TopCoor = [];

 for LevelLoop = LevelTop:LevelBottom
     LvlIdx = LevelLoop - LevelTop + 1;
     PitData_level = PitData.(['Level',num2str(LevelLoop)]);
     fig = figure('Renderer', 'painters', 'Position', [10 10 900 600]);
     hold on
    for ClusterLoop = 1:length(PitData_level)
     BlockCoor = PitData_level(ClusterLoop).BlockCoor;
     XCentroid = PitData_level(ClusterLoop).XCentroid;
     YCentroid = PitData_level(ClusterLoop).YCentroid;
        % Open each cluster centroid
        % CentroidX = PitData_cat(i).XCentroid;
        % CentroidY = PitData_cat(i).YCentroid;
        % Plot cluster's centroid and number

        % Plot blocks inside the cluster
            for BlockLoop = 1:length(BlockCoor)
                x = BlockCoor(:,1);
                y = BlockCoor(:,2);
                z = PitData_level(ClusterLoop).Level *(-.8);  % x,y, z coordinate
                CrusherLoc = PitData_level(ClusterLoop).CrusherLoc;

                if isnan(CrusherLoc)  % this cluster will never be mined
                   c = [1,1,1]; % set color to pure write
                elseif isempty(CrusherLoc)
                   c = [1,1,1]; % set color to pure write
                else
                   c = colormap(CrusherLoc,:); % set color to corresponding period
                end
                plotcube([1 1 .8],[x(BlockLoop)-0.5 y(BlockLoop)-0.5 z],1,c);
            end

            % Mark the level difference between cluster and crusher
            LvlDiff = LvlIdx - CrusherLoc;
            if  LvlDiff>0
                text(XCentroid,YCentroid,z+1,['+',num2str(LvlDiff)],'FontSize',16);
            elseif  ~isnan(LvlDiff)
                text(XCentroid,YCentroid,z+1,num2str(LvlDiff),'FontSize',16);
            end
           % find the pit bound for the top level
            if LevelLoop == LevelTop
                 TopCoor = [TopCoor;BlockCoor];
            end

 %   scatter3(CentroidX,CentroidY,-28.4,20, 'filled','k')
 %   text(PitData.Level29(i).XCentroid,PitData.Level29(i).YCentroid,-
28.4,num2str(i),'FontSize',15)
    end

        LocList = unique(Cl_CrusherLoc);
        LocList = LocList(~isnan(LocList));
  if ismember(LvlIdx,LocList) %that level locate crusher
   %% plot slot & conveyor spot (if not HAC)
   if strcmp(ConveyorSetting,'Slot')
    ColorSlot = colormap(t_slot,:);
    plot2_ConveyorSlot_Slice(ColorSlot,ConvSpot_slope,LevelLoop,LevelTop,LevelBottom)
    % plot the conveyor spot
    [ConvSpot_slope] = f_ConvWall_slope(slope,ConvSpot,LevelTop,LevelBottom);
```

```
    scatter3(ConvSpot_slope(LevelLoop,1),ConvSpot_slope(LevelLoop,2),-LevelLoop*0.8+1,300,...
        colormap(LvlIdx,:),'v','filled','MarkerEdgeColor','k','LineWidth',2);
   elseif strcmp(ConveyorSetting,'HAC') && LevelLoop ~= LevelBottom
            scatter3(ConvSpot(LevelLoop,1),ConvSpot(LevelLoop,2)-cosd(ang),-
LevelLoop*0.8+1,400,...
        colormap(LvlIdx,:),'v','filled','MarkerEdgeColor','k','LineWidth',2);
   else
            scatter3(ConvSpot(LevelLoop,1),ConvSpot(LevelLoop,2),-LevelLoop*0.8+1,300,...
        colormap(LvlIdx,:),'v','filled','MarkerEdgeColor','k','LineWidth',2);
   end
   text(ConvSpot(LevelLoop,1)-.3,ConvSpot(LevelLoop,2)+1.5,-LevelLoop*0.8+1,...
        ['Crusher L',num2str(LevelLoop-LevelTop+1)],'FontSize',16,'FontWeight','bold'); % Mark
the crusher location

  end

   %% Figure setting
    set(gca,'fontsize',18,'FontWeight','bold');
    title(['Level ',num2str(LevelLoop-LevelTop+1)]);
    xlabel('X Index');
    ylabel('Y Index');
    zlabel('Z Index');
    % Specify Axis Limits
    xmin = min(TopCoor(:,1));
    xmax = max(TopCoor(:,1));
    ymin = min(TopCoor(:,2));
    ymax = max(TopCoor(:,2));

   ax = gca();
   ax.YDir = 'reverse';
   ax.XDir = 'reverse';

   axis equal

   view(2)
    xlim([xmin-1 xmax+1]);
    ylim([ymin-1 ymax+1]);

    saveas(fig,['Slice_L',num2str(LevelLoop)],'tiffn')
 end

 legend off

 for i = 1:length(LocList)
     t = LocList(i);
    sc(i)= scatter3(ConvSpot(level,1),ConvSpot(level,2),-
level*.8,100,colormap(t,:),'s','filled');
    label(i) = cellstr(['Crusher L',num2str(LocList(i))]);
 end
    lgd = legend(sc,label,...
        'fontsize' ,6,'FontWeight','normal','Location','northeast');
    title(lgd,'Destination')
      legend('boxon')
```

## S9. Plot the updated UPL

```
% A script used to plot the updated UPL
% level by level for a spacific scenario (conveyor rotation angle)
% Dingbang Liu
% Date: October, 2020

LevelTop = 30;
LevelBottom = 35;
Block_cat = [];

fid = fopen('finalpit.txt');
block=cell2mat(textscan(fid,'%f %f %f %f %f','Delimiter',','));
```

```
blockIDX = find(block(:,1)>=30 & block(:,1)<=35);
    block = block(blockIDX,:);
    % Open each cluster centroid
    % CentroidX = PitData_cat(i).XCentroid;
    % CentroidY = PitData_cat(i).YCentroid;
    % Plot cluster's centroid and number

    % Plot blocks inside the cluster
        for j = 1:length(block)
            x = block(:,3);
            y = block(:,2);
            z = block(:,1);  % x,y, z coordinate
            g = block(:,5);

              alpha = 0.8 ; % transparency
            if g(j) == 0  % this cluster will never be mined
%                 continue
                c =  [1,1,1];
                alpha = 1;
            elseif g(j) > 0 && g(j)<0.6
                c = [1,1,0];
            elseif g(j) > 0.6 && g(j) <1
                c = [1,0.5,0];% colorjet(t,:);

            else
                c = [1,0,0];% colorjet(t,:);
            end
            c = [.5,.5,.5];
            plotcube([1 1 .8],[x(j)-0.5 y(j)-0.5 -z(j)*.8],alpha,c);
        end

%         Cluster_CEV(i) = PitData_cat(i).CEV;
%         Cluster_Ton(i) = PitData_cat(i).Tonnage;
%         Cluster_Ore(i) = PitData_cat(i).TonnageOre;
%   scatter3(CentroidX,CentroidY,-28.4,20, 'filled','k')
%   text(PitData.Level29(i).XCentroid,PitData.Level29(i).YCentroid,-
28.4,num2str(i),'FontSize',15)

%
%   legend off
%   colormap = [[1,1,1];[1,1,0];[1,0.5,0];[1,0,0]];
%
%  for t = 1:4
%     sc(t)= scatter3(30,30,30,1,colormap(t,:),'s','filled');
%  end
%     lgd = legend(sc,{'1','2','3','4'},...
%         'fontsize',14,'FontWeight','normal','Location','northeast');
%     title(lgd,'Periods')
%       legend('boxon')
%
%  for i = 1:length(LocList)
%      t = LocList(i);
%     sc(i)= scatter3(ConvSpot(level,1),ConvSpot(level,2),-
level*.8,100,colormap(t,:),'s','filled');
%     label(i) = cellstr(['Crusher L',num2str(LocList(i))]);
%  end
%     lgd = legend(sc,label,...
%         'fontsize' ,6,'FontWeight','normal','Location','northeast');
%     title(lgd,'Destination')
%       legend('boxon')


%  legend(t,{'1','2','3','4','5','6','7','8','9','10'})
axis equal
axis tight
axis off
view(40,10)
% view(160,10)
% view(270,0)
      %% plot conveyor side
```

```
% scatter3(ConvSpot(Top,1),ConvSpot(Top,2),-Top*0.8+1,500,'w','filled',...
%     'MarkerEdgeColor','k','LineWidth',2);
        %% create legend
%  legend off
%  for t = 1:Periods
%     sc(t)= scatter3(ConvSpot(level,1),ConvSpot(level,2),-
level*.8,1,colorjet(t,:),'s','filled');
%  end
%     lgd = legend(sc,{'1','2','3','4','5','6','7','8','9','10'},...
%         'fontsize',14,'FontWeight','normal','Location','northeast');
%     title(lgd,'Periods')
%       legend('boxon')
```