Université de Montréal

Algorithme de recherche à voisinage adaptatif pour l'optimisation stochastique des complexes miniers

An adaptive neighborhood search algorithm for optimizing stochastic mining complexes

par Sean Grogan

Département d'Informatique et Recherche Opérationnelle
Faculté de Arts et Sciences

Mémoire présenté
en vue de l'obtention du grade de Maîtrise
en Informatique

8 août 2016

Résumé

Les métaheuristiques sont très utilisées dans le domaine de l'optimisation discrète. Elles permettent d'obtenir une solution de bonne qualité en un temps raisonnable, pour des problèmes qui sont de grande taille, complexes, et difficiles à résoudre. Souvent, les métaheuristiques ont beaucoup de paramètres que l'utilisateur doit ajuster manuellement pour un problème donné. L'objectif d'une métaheuristique adaptative est de permettre l'ajustement automatique de certains paramètres par la méthode, en se basant sur l'instance à résoudre. La métaheuristique adaptative, en utilisant les connaissances préalables dans la compréhension du problème, des notions de l'apprentissage machine et des domaines associés, crée une méthode plus générale et automatique pour résoudre des problèmes.

L'optimisation globale des complexes miniers vise à établir les mouvements des matériaux dans les mines et les flux de traitement afin de maximiser la valeur économique du système. Souvent, en raison du grand nombre de variables entières dans le modèle, de la présence de contraintes complexes et de contraintes non-linéaires, il devient prohibitif de résoudre ces modèles en utilisant les optimiseurs disponibles dans l'industrie. Par conséquent, les métaheuristiques sont souvent utilisées pour l'optimisation de complexes miniers. Ce mémoire améliore un procédé de recuit simulé développé par Goodfellow & Dimitrakopoulos (2016) pour l'optimisation stochastique des complexes miniers stochastiques. La méthode développée par les auteurs nécessite beaucoup de paramètres pour fonctionner. Un de ceux-ci est de savoir comment la méthode de recuit simulé cherche dans le voisinage local de solutions. Ce mémoire implémente une méthode adaptative de recherche dans le voisinage pour améliorer la qualité d'une solution. Les résultats numériques montrent une augmentation jusqu'à 10% de la valeur de la fonction économique.

Mots-clés : recuit simulé, métaheuristiques adaptatifs, optimisation des mines, incertitude métallique

Abstract

Metaheuristics are a useful tool within the field of discrete optimization that allow for large, complex, and difficult optimization problems to achieve a solution with a good quality in a reasonable amount of time. Often metaheuristics have many parameters that require a user to manually define and tune for a given problem. An adaptive metaheuristic aims to remove some parameters from being tuned or defined by the end user by allowing the method to specify and/or adapt a parameter or set of parameters based on the problem. The adaptive metaheuristic, using advancements in understanding of the problem being solved, machine learning, and related fields, aims to provide this more generalized and automatic toolkit for solving problems.

Global optimization of mining complexes aims to schedule material movement in mines and processing streams to maximize the economic value of the system. Often due to the large number of integer variables within the model, complicated constraints, and non-linear constraints, it becomes prohibitive to solve these models using commercially available optimizers. Therefore, metaheuristics are often employed in solving mining complexes. This thesis builds upon a simulated annealing method developed by Goodfellow & Dimitrakopoulos (2016) to optimize the stochastic global mining complex. The method outlined by the authors requires many parameters to be defined to operate. One of these is how the simulated annealing algorithm searches the local neighborhood of solutions. This thesis illustrates and implements an adaptive way of searching the neighborhood for increasing the quality of a solution. Numerical results show up to a 10% increase in objective function value.

Keywords: Simulated Annealing, Adaptive Metaheuristics, Mine Optimization, Metal Uncertainty

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

ANS: Adaptive Neighborhood Solution

APF: Acceptance Probability Function

Ag: Silver

Au: Gold

CF: Cash Flow

Cu: Copper

DE: Differential Evolution

GI: Global Iteration

IP: Integer Program

LOM: Life of Mine

LP: Linear Program

NPV: Net Present Value

ObjFn: Objective Function

PSO: Particle Swarm Optimization

GD: Goodfellow & Dimitrakopoulos (2016)

RS: Random Start

SA: Simulated Annealing

SIP: Stochastic Integer Program

All spiritual growth comes from reading and reflection. By reading we learn what we did not know; by reflection we retain what we have learned. The conscientious reader will be more concerned to carry out what he has read than merely to acquire knowledge of it. In reading we aim at knowing, but we must put into practice what we have learned in our course of study.
Isidore de Seville

Remerciements

I would first like to thank Prof. Jacques Ferland for the opportunity to study here at the Université de Montréal and for his guidance in the creation of this thesis. Next, I would like to thank Amina Lamghari for her help also in writing and creating the new method established in this thesis. Furthermore, thanks to Professor Roussos Dimitrakopoulos for opening up this field to me and presenting me with the chance in my Undergrad to be exposed to more advanced ideas in the mining industry.

In addition, I would like to thank a few of my colleagues and mentors from over the years: Ryan Goodfellow, Luis Montiel Petro, Maria Fernanda Del Castillo, Chotipong Somrit, Alessandro Navarra, Hani Mitri, John Mossop, Ryan Lechner, Linda Muratore, Michael O'Boyle, and Mercedes Brand. You guys served as great role models and teachers. You all were generous in sharing your expertise over the years.

Most importantly, I wish to thank my family members: My Mom, Dad, sister Maura, grandmothers Alice and Synnie, and grandpa William. Without your support over the last twenty-six years I would have never been able to get to where I am today. Thank you!

Finally, I would finally like to thank my friends – my extended family in COSMO and Challenge; you have kept me grounded, sane, on track, and fed through the past two years. Many of you have been excellent confidants and mentors, done little things to improve my quality of life, and even directly aided me with this thesis. Thanks to all of you!

1 Introduction

Mining and related industries is one of the largest and riskiest sectors of the Canadian economy. In 2015, mining and related industries was the third largest industry in Canada after real estate and manufacturing, representing about 8% of the economy, and accounting for 28% of all goods producing industries. In the province of Québec, metallic mineral production represents 26% of the nation's mineral production (by dollars) and is second only to Ontario (Natural Resources Canada, 2016; Énergie et Ressources Naturelles Québec, 2016; Statistics Canada, 2016). Proper planning procedures and interpretations can mitigate the risk associated with developing and operating mines and mining complexes around the world. In the latter half of the 20th century, new ways of modelling mining complexes and interpreting what is in the ground have been developed. Furthermore, mathematical models, such as integer programs (IP), have been developed and implemented to schedule the production in mining complexes. As mathematical formulations grew to be more detailed representations of mining complexes, metaheuristics have been developed to efficiently solve them. One such metaheuristic is simulated annealing (SA). SA is used by Goodfellow & Dimitrakopoulos (2016) to optimize their updated model formulation for mining complexes. Their model specifically introduces the ability to account for the "inherent non-linearity related to the blending and stockpiling of materials" (Goodfellow & Dimitrakopoulos, 2016). Their work is based on and updates work of several authors such as Ramazan & Dimitrakopoulos (2013), Jewbali (2006), and Benndorf & Dimitrakopoulos (2004). The SA based optimization method outlined by the authors takes static search parameters for neighbor solution selection. While the authors are able to find a good solution in a reasonable amount of time, this thesis takes progress made in adaptive metaheuristics and related fields, such as from Pisinger & Ropke (2007) and Lamghari & Dimitrakopoulos (2015), to improve the optimization method used by Goodfellow & Dimitrakopoulos (2016). This update to the solver should produce better solutions.

1.1 Overview of Open Pit Mining

Most often, it is companies (not, for example, the government) that partake in mining operations around the world. A company is said to be in the *mining industry* if they specialize

in extracting naturally occurring and nonrenewable resources. This can be, but is not limited to: gold, copper, potash, uranium, or the oil sands. In addition, mining includes operations that occur after extraction. Examples of these *downstream operations* include refineries, smelters, and transportation, which transform the extracted material to something *more useful* (Government of Canada, 2016). We use "more useful" here colloquially. Material is deemed to have been "transformed into something more useful" if the material is transformed, typically called *processed*, into another material which is closer to something that can be used by a consumer or another industry. Examples are oil into gasoline for a car or insitu copper ore into a copper block to be used by a pipe company. Mining companies operate in areas where they have identified a *deposit* of material. For this thesis, a *deposit* is where the companies have deemed there exists material which is economically feasible to be extracted and sold. That is, they can operate a *mining complex* at a profit. A *mining complex* is the system by which material is moved from the earth, transformed if possible, and then sold (Blechynden, Gardener, & Mossop, 2012; Government of Canada, 2016; Newman, et al., 2010).

Before a mining complex is established, the deposit must be located. Locating and gaining information about a deposit is known as *exploration*. There are many different techniques to locate a deposit. Once a company has found a potential area for a mining complex, they must begin a technique known as *core hole drilling*. Core hole drilling is when machines drill long vertical holes into the earth and retrieve long solid pieces of rock. These long pieces of rock are known as *core holes*. In the region the company desires to extract material, hundreds to thousands of these core holes will be extracted in a regular grid. These core holes can be viewed as conditioning data for geologists to interpret what is within the earth (Blechynden, Gardener, & Mossop, 2012; Buro, 2013; Newman, et al., 2010). Geologist interpretations are commonly known as *orebody models* which are ultimately used in planning mine complexes.

In the mine planning, scheduling, and optimizing process, the orebody models are typically discretized into regular sized *blocks*. This modified model is often referred to as a *block model*. For each block in a block model, a geologist will assign an attribute which represents the amount of a specific material as a proportion by weight. This attribute is known as the *grade* of the block. Often, the *grade* is a percentage. For example, a block that weighs 1000 tonnes and has 15% copper grade will have 150 tonnes of copper. However, some metals,

such as gold or platinum, have small amounts of the metal present in a given block. These metals are typically recorded in grams per tonne. Detailed block models and multi-element deposits will have many attributes in each block. For example, due to molecular similarities, gold deposits will often have economical amounts of silver or copper in addition to gold. Deleterious, or waste, elements are also present. Often copper or gold deposits will have amounts of sulfides, a common deleterious element, present in each block (Buro, 2013; Newman, et al., 2010).

Once a geologist provides a block model interpretation, mine planners must decide how to extract the valuable material at a profit for the company. The first decision a mine planner must make is what kind of mine to establish. There are two basic types of mines: *open pit* and *underground* mines. Open pit mines are developed in places where valuable material is close to the surface of the earth. They proceed with extracting material at the surface of the earth and continue to work their way deeper until all the valuable material has been extracted. With open pit mining, large amounts of waste material must be removed to gain access to valuable material. In underground mining, on the other hand, is where valuable materials typically extracted through tunnels or shafts. This allows access to valuable material deeper in the earth and reduces the amount of waste material required to be removed if open pit mining techniques were used (Blechynden, Gardener, & Mossop, 2012; Newman, et al., 2010). This thesis will focus on open pit mining. Open pit mining accounts for the largest number of mines in the world and the industry partners we are working with have offered their deposit data, all of which is set up as an open pit mining operation.

The next decision the mine planner must make is how to move the material through the mining complex. In open pit mining, a mining complex typically has four main locations. The first main location is the physical *mine* itself. The mine is where the material (blocks) is being extracted from the earth. Once extracted, the blocks are transported to one of three destination locations: a *waste dump*, a *processor*, or a *stockpile*. The decision on where to send each block is based largely on the grade of the block. Blocks sent to the waste dump typically lack sufficient quantities of valuable material to be processed and sold at any kind of advantage for the company. Once it arrives at the waste dump, block typically will never be rehandled, or moved, again. The second destination is a processor. This destination processes, or transforms, the

material to something more useful. The output of a processor typically has the ability to be sold at a market or to a contracted firm. In a copper or gold mine, there are two types of processors. The first processor type is a *plant*. A plant will attempt to grind ore material into a fine powder and use a separation technique, such as flotation, to separate waste material from the valuable material. The flotation happens in a modified water solution where the properties of the water are chemically adjusted in such a way where valuable material will stick to air bubbles and float to the top of a flotation cell while waste will settle to the bottom. The second processor type is a *leach pad* where acid is used to "leach" away valuable material from waste material. The third possible destination for extracted material is a stockpile. Stockpiles are areas that companies set aside to store material until there is an opportunity to process it. Stockpiling material is due to limitations in the amount of material that can be processed or a desire to *blend* material at a processor to increase the advantage to a company. Blending material is processing two units of different material at the same time to get the average of the material's properties. An outline of the basic mining complex and the main destinations can be seen in Figure 1 below (Blechynden, Gardener, & Mossop, 2012).

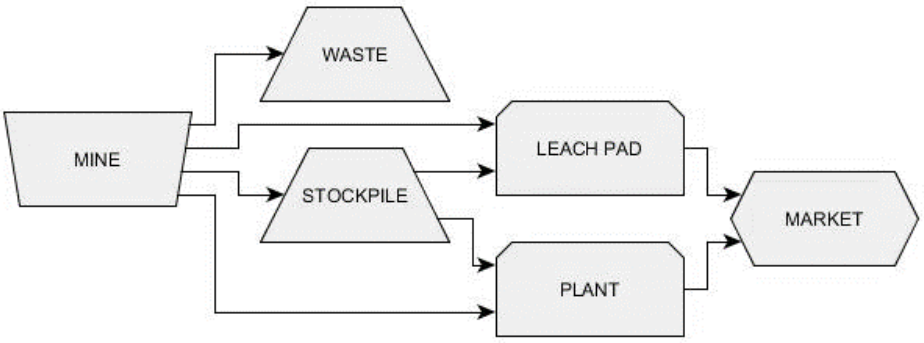

Figure 1: An example of a mining complex

As mentioned above, a processor attempts to separate the waste material from the valuable material. This separation is not perfect. For each processor, there is an associated value called the *recovery*, which represents a percentage that is the amount of valuable material that is saved, or recovered, from each block. For example, if a plant has a 90% recovery, a block with 150 tonnes of copper will yield 135 tonnes of the copper out of the plant. The other 15 tonnes will be discarded. Compare this with a leach pad that has a 50% recovery; that same block will yield 75 tonnes of copper if processed at the leach pad. A processor's recovery is

typically a function of the amount of material and/or the grade of the material sent to the processor. Therefore, the recovery of a processor is often non-linear and can vary on a variety of factors (Blechynden, Gardener, & Mossop, 2012).

There are two main constraints in an open pit mining complex. The first are the *capacity constraints*. Equipment, safety, and other limitations exist which restrict the amount of material that can be moved or processed at each part of the complex in a given *period*. A period is a unit of time, typically a year, that a mine complex is operated. Capacity constraints take a few forms. *Mining capacity* is the amount of material that can be extracted from the mine in a period. *Processing capacity* is the amount of material that can be sent to a specific processor in a period. *Stockpiling capacity* is the amount of material that can be stored in a stockpile. While mining and processing capacities are calculated on a per period basis, the stockpile capacity is the upper limit on the amount of material that can be stored there at any given time. The sum of all the material is carried over period to period until it is *rehandled*, or moved again – in this case from the stockpile to the processor (Newman, et al., 2010).

In addition to capacity constraints, *precedence constraints* exist in open pit mining complexes, sometimes called *slope constraints* in the literature. In open pit mining when scheduling with a block model, for a given block, the block directly above and blocks adjacent to the directly above block must be extracted before the given block can be. An example of this can be seen in Figure 2 below.

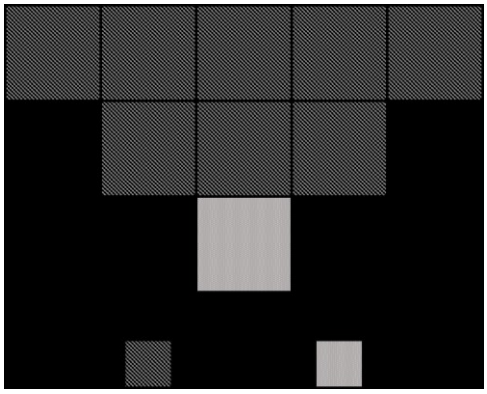

Figure 2: A 2D schematic of blocks that must be removed to access block b

As can be seen in Figure 2, the blocks identified as the *Overlying blocks* set $\mathbb{O}_b$ (blocks highlighted in dark grey) must be extracted in the same period or in an earlier period before

Block b (the block highlighted in light grey) can be extracted. Note that we will use slopes of 45° in the examples in this thesis. However, slopes may take a variety of angles for geotechnical or other safety reasons. It is these precedence constraints which often make optimizing mining complexes quite long and complicated (Newman, et al., 2010).

1.2 Mining Complex Economic Evaluation

After core holes have been extracted and before a company decides to establish a mining complex, the company will conduct a *feasibility study* to determine the economic value of the deposit and complex. One part of the feasibility study is to schedule material movement in the mining complex, which is what the formulation and the solution method in this thesis can be applied to. Planners will attempt to maximize this economic value over the course of the *life of mine* (LOM). The LOM is the number of periods from the beginning of development of the project until there remains no more material in the deposit or stockpiles that can be processed at an advantage for the company. Note that the LOM and economic evaluation of a mine will typically not include the exploration and core hole drilling costs and procedures. However, the economic evaluation will often include other capital expenditures such as the construction of plants and the purchase of equipment. The exception to including capital expenditures is if a new mine is being developed in the vicinity of an already established mining complex where equipment can simply be moved or be used in multiple complexes (Albach, 1967; Blechynden, Gardener, & Mossop, 2012; Gentry, 1988).

Most mining projects calculate the economic advantage with a criterion known as *net present value* (NPV). Broadly speaking, the NPV is the value of a project. An NPV with a positive value indicates that the projected earnings generated by a project or an investment exceed the anticipated costs when taking the value of money over time into consideration. The NPV of a mining project is calculated as follows:

1) Calculate the *cash flow* (CF) of each period for the entire LOM. The CF of a period is typically the value of the material sold, minus the costs incurred to process and transport the material.

2) Determine the *discount rate* of the project. This discount rate is a percentage that is associated with a project which encompasses the risk of a project and the rate of return from other possible investments. It is typically determined from other similar projects and the state of the economy.

3) For each period, determine the *present value* (PV) of the CF for each period. The PV is calculated by dividing the CF of the period by one plus the discount rate raised to the number of periods from the first period. I.e. $\mathrm{PV_t} = \mathrm{CF_t}/(1 + \text{discount rate})^{\mathrm{t}}$. This can be viewed as the CF at time t has a value of $\mathrm{PV_t}$ today.

4) Sum all the PV's for each period to get the NPV of the project: i.e. $\mathrm{NPV} = \sum_{t \in \mathbb{T}} \mathrm{PV}_t$

Other evaluation methods can be used, but NPV is the most common and widely understood in the industry (Gitman & Joehnk, 1999; Investopedia, 2016; Whittle, 2014).

The factors that go into determining the value of a CF in a period are the costs associated with extracting, moving, and processing material from a mine to the three destinations and the value of the material sold on the market. Recall we are operating an open pit mine and there are precedence constrains associated with extracting a block. Looking back to Figure 2 on page 5, imagine if block b was the only block that had valuable material in it and all the overlying blocks in the set $\mathbb{O}_b$ were waste material. We must extract and incur the cost associated with removing the 8 blocks in the overlying block $\mathbb{O}_b$ set in addition to the cost of extracting, processing, and subsequently selling block b. We only see revenue in the period after block b is processed and then sold.

Mining projects are quite a high risk investment. These projects typically have a few unique attributes compared to other investment opportunities. They are:

- Capital intensive projects – they have a high initial starting cost.
- Non-renewable resource – Once the material is extracted, it is gone from the Earth. Any infrastructure built is typically abandoned or sold at a significant loss if the mine is in a remote location.

- Long pre-production periods – it may take several periods for unwanted material to be extracted to get to valuable material. In addition, infrastructure, such as roads and plants, may need to be built to handle the material.
- The indestructibility of the material – gold mined in Quebec will be essentially the same as gold from Nevada or Ghana. Therefore, one can seek either cheaper deposits elsewhere or look in low risk locations.

The combination of these factors makes mining a high risk environment for investors. Therefore, proper planning and evaluation is critical for investors and companies to make informed decisions about mining projects (Gentry, 1988; The Northern Miner, 1990; Whittle, 2014).

1.3 Optimization of Open Pit Mining Complexes

Traditionally, mining complexes were scheduled both locally and iteratively. We say locally because each location and element in the mining complex was independently scheduled. This is akin to a greedy heuristic and can lead to a sub-optimal global solution. Recall we are attempting to maximize NPV. An example of the local scheduling technique can be to maximize recovery of a processor. Maximizing the recovery of a processor ensures the most amount of valuable material is sent to the market (least amount of valuable material is wasted). However, this greedy local decision to maximize recovery may lead to a sub-optimal decision. Typically, a higher recovery reduces the rate at which material can be processed in the plant (i.e. the processor operator must lower processing capacity), but increasing the processing capacity of the processor and lowering recovery, may increase the overall NPV. A mining complex is considered to be scheduled iteratively because a small change in one part of the mining complex can affect the value of the complex as a whole. Using an example as an illustration, let there be a plant engineer who decided to plan for the processing capacity to be PC_1. If we change the processing capacity from PC_1 to a lower value PC_2, the engineers who manage extraction scheduling must adjust the amount of material they send to the processor; by sending some material as waste, by opening a stockpile to store excess material, by reducing the mining capacity, or by changing which blocks are mined in each period. Once the schedule is updated,

planners may then decide to make another small adjustment to the mining complex parameters and then the process repeats (Gentry, 1988; The Northern Miner, 1990; Whittle, 2014).

Because of this local and iterative process for determining a schedule to extract material over the life of mine, we desire a global way to optimize these complexes. That is, we desire a way to simultaneously optimize production scheduling. This led to the development of linear and integer programs to represent the production schedule. These mathematical formulations have the ability to choose the best extraction scheduling decisions. These methods were first explored in the 1960's (Albach, 1967; Gershon, 1983; Gholamnejad & Osanloo, 2007; Bienstock & Zuckerberg, 2010; Busnach, Mehrez, & Sinuany-Stern, 1985). Due to the wide range of mines in the world and how basic concepts are applicable across many deposits and complexes, mine complex optimization is a well-studied field (Newman, et al., 2010). Some of the most modern methods do incorporate what is known as *metal uncertainty*.

1.4 Metal Uncertainty and the Simulation of Orebody Models

Because mining complexes are developed based on the information obtained from core hole drilling, the interpretation of what is in the deposit can have a significant impact on the valuation of a mining complex. Traditionally, mining complexes were optimized using a singular orebody model. This model was developed using an interpolation method, such as kriging (Krige, 1951), between the known data points, the core holes. These traditional methods smooth the transition of grades between the core holes. This incorrect estimation will lead to an inaccurate and high-risk evaluation of a deposit (Ravenscroft, 1992; Godoy M. , 2003; Dimitrakopoulos, 2015; Consuegra & Dimitrakopoulos, 2009; Dimitrakopoulos, Farrelly, & Godoy, 2002; Osterholt, 2005).

Geostatistical or *stochastic conditional simulation* is an estimation tool which generate models of a deposit based on the same core hole data used in traditional methods. When one generates multiple simulated orebody models, they take on two properties (Dimitrakopoulos, 2015):

1. Simulations reproduce the available information of the core holes. That is, the simulations will reproduce similar information that is already represented by the core holes.
2. Each simulation is an equiprobable representations of the deposit.

Examples of simulation methods that are used are direct block simulation, Gaussian simulation methods, and high order simulation methods (Benndorf & Dimitrakopoulos, 2004; Godoy & Dimitrakopoulos, 2004). Authors who optimize using simulated orebody models typically find a higher NPV and a lower risk schedule when compared to using a single orebody model. For example, Goodfellow & Dimitrakopoulos (2016) use a stochastic integer program (SIP) to represent the mining complex. This model was able to achieve an NPV 6.6% higher than the deterministic model. In addition, less risk is associated with the amount of material sent to various destinations. Another example is by Dimitrakopoulos, Farrelly, and Godoy (2002) who perform a risk analysis on a mine. Their analysis shows that when using simulated orebodies, the deterministically scheduled mine has only 15% chance of reaching the original NPV. Additionally, the authors conclude that the expected NPV of the schedule is 25% below what is originally projected using the deterministic model. A third case study is where Godoy (2003) completes a risk analysis for a mine in Australia. Results of using simulated orebodies yields a 28% increase in NPV compared to the deterministic solution and a 3% chance of the stochastic schedule failing to meet yearly production targets, as opposed to the 13% for the deterministic schedule. The author notes that the increase in NPV is due to the optimizer's ability to extract more valuable material earlier in the life of mine.

This thesis will utilize simulated orebody models in the optimization process.

1.5 Objectives of the Research

In this thesis, we refer to a recent and more general mathematical formulation representing a mining complex – the act of moving raw material from the earth to selling refined material on the market – presented by Goodfellow & Dimitrakopoulos (2016). In order to solve this problem, we also refer to their simulated annealing approach using several neighborhoods to determine a schedule of events through the life of the project. In their implementation, the

neighborhood used at each iteration is selected among a set of different neighborhoods according to a distribution specified a priori. The main contribution in this thesis is to improve this solution approach by using the adaptive principles introduced in Pisinger & Ropke (2007) and in Lamghari & Dimitrakopoulos (2015) to select the neighborhood. The motivation for this contribution is twofold. Indeed, it allows for a different mining complex to be resolved without having a user to determine a priori the distribution for selecting the neighborhood, and also for a mine planner who may not be familiar with metaheuristic principles, to use this method to develop a schedule for their mining complex. The numerical results show an average increase of 1 to 2% of the objective function value for a single element deposits. For a larger copper-gold deposit, we observe an average increase of 10% for the objective function value and a reduction of about 40% of the solution time.

The remainder of this thesis is organized as follows. The simulated annealing approach and the principles of adaptive metaheuristics are summarized in Chapter 2. Chapter 3 includes the general mathematical formulation of the model introduced in Goodfellow & Dimitrakopoulos (2016). It also includes their implementation of the simulated annealing and the details of the adaptive selection of the neighborhood. The numerical results are summarized in Chapter 4. Two single element deposit problems including copper and gold, respectively, and a larger problem including a copper-gold deposit are solved in order to illustrate the advantage of using the adaptive approach. Chapter 5 includes conclusions.

2 Review of Metaheuristics and Adaptive Metaheuristics

"Metaheuristics are solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space" (Gendreau & Potvin, 2010). That is, metaheuristics are a set of algorithms which allow for broad search of solutions, even solutions that are non-improving, to discover high quality solutions. Metaheuristics are a useful tool within the field of discrete optimization that allow for large, complex, and difficult optimization problems, such as the one addressed in this thesis, to be solved in a reasonable amount of time. Solving discrete optimization problems in an exact way may take orders of magnitude longer to solve than using a metaheuristic to reach a good or acceptable solution to the problem. Adaptive metaheuristics aim to increase the generalization of a metaheuristic method for a given problem. This may allow for a user who is untrained in the implementation of a metaheuristic to use the method to find a solution to the problem.

This section outlines the metaheuristics used by Goodfellow & Dimitrakopoulos (2016), specifically simulated annealing (SA). In the solution approach of Goodfellow & Dimitrakopoulos (2016), they use a strategy to optimize downstream (processor) variables after SA. This strategy relies on the population based procedures differential evolution (DE) and particle swarm optimization (PSO). Since we are not modifying these strategies in this thesis, we will not describe their implementation further except for a brief comment on their use in the solving method outlined by Goodfellow & Dimitrakopoulos (2016) in section 3.2.

2.1 Simulated Annealing

Simulated annealing (SA) is a local neighborhood search metaheuristic allowing to modify the current solution even with one deteriorating the objective the objective function value in order to move away from a local optimal solution. Kirkpatrick, et al. (1994) and Cerny (1985) were the first to propose solving combinatorial problems with this approach used in thermodynamics to search for an equilibrium. To ease this presentation, suppose that we are solving the following problem of maximizing a function $f(x)$ over a feasible domain $X \in \mathbb{R}^n$. At each iteration, a new solution x' is randomly selected in the neighborhood of the current

solution x. A neighbor solution x' is typically similar to the current solution, with a few simple modifications to the current solution x. The SA algorithm then allows to compare the quality of the selected solution against the quality of the current solution. If the selected solution is better (i.e. $f(x') > f(x)$), then it becomes the current solution. Otherwise (if the selected solution x' is worse), x' can replace the current solution x even if $\Delta f = f(x') - f(x) \leq 0$ according to a function which calculates the probability as a function of Δf and the number of iterations already completed; i.e. x' replaces x with the acceptance probability function $e^{\Delta f/\tau}$ where τ (the temperature parameter) decreases with the number of iterations completed.

In this variant, we complete several iterations n^{iter} with the same temperature τ. Note that a special case is to modify the temperature at each iteration (i.e. $n^{iter} \leftarrow 1$). The temperature τ is modified with the parameter ε (i.e. at $\tau \leftarrow \tau \cdot \varepsilon$), where $0 < \varepsilon < 1$. Two stopping criteria are used. The first one is specified in terms of the number of iterations the SA is ran ($iter_{Max}$). The second is one is specified in by counting the number of global best updates (i_{limit}^{gbu}), that is, the number of times a new global best solution is found. A variant of the procedure can be summarized in Algorithm 1.

Algorithm 1: Simulated Annealing

```
Initialization:
      Select an initial solution x^0 ∈ X and an initial temperature τ^0
      Let iter ← 0; τ ← τ^0
      Let x ← x^0; x* ← x^0
While stopping criteria is not met
      i^gbu ← 0
      iter ← iter + 1
      Repeat n^iter times with the same temperature τ
            Generate randomly x' ∈ N(x)
            Δf = f(x') − f(x)
            If Δf > 0
                  x ← x'
            Else generate a random number r ∈ [0,1]
                  If r < e^(Δf/τ)
                        x ← x'
            If f(x') > f(x*)
                  x* ← x' ; i^gbu ← i^gbu + 1
      τ ← ε · τ
      If iter > iter_Max or i^gbu > i_limit^gbu
            Return x*
```

In order to improve the quality of the solution generated with any local neighborhood search procedure, it should be combined with a diversification strategy to search more extensively the feasible domain of the problem. Many such strategies exist, and they are most of the time specific to the problem.

2.2 Adaptive Neighborhood Search Techniques

One of the difficulties in using metaheuristics in optimization is that often the methods require parameter tuning by the user to increase the quality of the final solution. One promising area of research is utilizing adaptive neighborhood search (ANS) to help guide the search of the solution space. ANS is especially useful when using a metaheuristic which has a local search framework, such as in the case of SA.

In this section, we analyze the step "generate randomly $x' \in N(x)$" in the SA procedure in Algorithm 1. Moreover, consider the case where $N(x)$ is specified using a set of neighborhoods $\{n_1, n_2, \dots, n_{|\mathbb{N}|}\}$. Note that in the SA implementation of Goodfellow & Dimitrakopoulos (2016), the number of neighborhoods $|\mathbb{N}|$ is equal to three. Before generating x', we first select randomly the neighborhood to be used. In Goodfellow & Dimitrakopoulos (2016), this section is made by a probability distribution specified a priori and manually tuned by the authors. In the proposed contribution in this thesis, this probability distribution is made adaptive. The probability of selecting neighborhood n_i is proportional to its efficiency in the solving process. This approach follows the adaptive large neighborhood search (ALNS) approach outline by Pisinger & Ropke (2007).

The selection process is summarized as follows: At each iteration to generate a neighbor solution x', first a neighborhood n_i must be selected. n_i is selected by an associated probability p_i for all the $i \in \mathbb{N}$. The same values of the probabilities $p_i, \forall i \in \mathbb{N}$ should be used for the same number of $(ScoreUpdate)_{Skip}$ iterations in the local search method. At each $(ScoreUpdate)_{Skip}$ iteration, the probabilities p_i should be updated based on a score parameter s_i for each neighborhood n_i. The scores should be proportional to the efficiency of the

neighborhood. Therefore, larger scores will represent neighborhoods that have a better impact on the quality of the solution.

To update the scores after $(ScoreUpdate)_{Skip}$ iterations, there is a scalar κ_i indicating the number of times neighborhood n_i is selected. In addition, the value of $\pi(n_i)$ represents the efficiency of neighborhood n_i. The values are updated each time neighborhood n_i is selected as follows:

$$\kappa_i \leftarrow \kappa_i + 1 \tag{1}$$

$$\pi(n_i) \leftarrow \pi(n_i) + \sigma \tag{2}$$

where σ represents the value of the efficiency of n_i. We will calculate the efficiency σ as a function of the change in the objective function value. After completing $(ScoreUpdate)_{Skip}$ iterations, the scores s_i are updated as follows:

$$s_i \leftarrow \begin{cases} (1-\alpha)s_i + \alpha\left(\frac{\pi(n_i)}{\kappa_i}\right) & \text{If } \kappa_i > 0 \\ s_i & \text{Otherwise} \end{cases} \tag{3}$$

and the probabilities p_i become

$$p_i \leftarrow \frac{s_i}{\sum_{k\in\mathbb{N}} s_k} \; \forall i \in \mathbb{N} \tag{4}$$

Take note that in (3), if a neighborhood is not called, the score remains the same for the next $(ScoreUpdate)_{Skip}$ iterations. There is also the introduction of a parameter $\alpha \in [0,1]$ which controls the emphasis on historical scores versus new scores. That is, if the parameter α is set close to 1, more emphasis is placed on newer information versus an α closer to 0 which places emphasis on historical information.

In the following chapter, we introduce the model proposed in Goodfellow & Dimitrakopoulos (2016) for an open pit mining complex and their specific implementation of SA the authors use to solve it. Then, this thesis introduces a more sophisticated implementation of the adaptive approach for selecting the neighborhood at each iteration based on the notation in Lamghari & Dimitrakopoulos (2016) to specify the value of σ.

An outline of the method described above can be seen here in Algorithm 2.

Algorithm 2: Basic adaptive framework as posed by Pisinger & Ropke (2007)

```
GENERATE x  # Initial Solution x
SET x* ← x  # Best Solution x*
i ← 0
WHILE Stopping criteria is not met
      i ← 1 + 1
      If i mod (ScoreUpdate)_skip = 0
            Update the probabilities p_i based on scores s_i
            Set s_i ← 0 ∀i ∈ ℕ
      Choose a neighborhood n_i probabilities p_i
      GENERATE x′ from x using the neighborhood n_i
      IF x′ is accepted
            x ← x′
            UPDATE π(n_i) based on success
      ELSE
            UPDATE π(n_i) based on failure
      IF x is a better solution than x*
            x* ← x
RETURN x*
```

3 Implementation of Adaptive Neighborhood Choice in Simulated Annealing to Optimize the Stochastic Mining Complex

This section introduces an overview of the model (section 3.1) and the solver (section 3.2) presented by Goodfellow & Dimitrakopoulos (2016). The third part of the section goes over the contribution to include an adaptive neighborhood search in the solving method (section 3.3). This adaptive neighborhood search used in this contribution is based on the work of Lamghari & Dimitrakopoulos (2015).

Recall that a naïve way of scheduling a mining complex is to discretize a deposit into a collection of blocks and to assign a dollar value to each block; this value is calculated by the grade of the block, the recovery value of the processor, costs incurred in processing, and the market value of the metal. This approach to valuing a complex is inaccurate when applying it to a mine in practice. For example, recall the basic mining complex from section 1.1. Each processor has a different recovery and this difference in recovery will result in a different value of the block being mined. Therefore, we must use a model to analyze the value of a mining complex referring to its outputs rather than to each block value.

3.1 Stochastic Integer Model of an Open Pit Mining Complex

Goodfellow & Dimitrakopoulos (2016) utilize a two-stage stochastic optimization model. The formulation, replicated here, is written to be more holistic than models that appear elsewhere in the literature, such as those explored in section 1.3. That is, they aim the model to be able to be applied to a wide variety of deposits with more production constraints. In addition, the model is better at valuing the output of the processor outputs each period rather than the value of each block sent through a processor.

Goodfellow & Dimitrakopoulos (2016) view the mining complex as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$ to keep track of the flow of material through the mining complex. The nodes $\mathcal{N}$ are classified into three sub-groups:

1. $\mathcal{C}$: Clusters of mined material, i.e. blocks, that have similar attributes.

2. $\mathcal{S}$: Nodes associated with stockpiles.
3. $\mathcal{P}$: Nodes associated with processors.

Note that using this notation, a waste dump can also be viewed as a processor that has a recovery of zero, i.e. no value is gained from a "waste dump processor." Moreover, a cluster $\mathcal{C}$ is used to group together similar blocks of material in the mine. The authors group material into clusters using a k means++ clustering algorithm. K means++ was used because it generally produces stable clusters relative to regular k means clustering algorithm and a much more diverse cluster sets due to the weighting in the algorithm (Arthur & Vassilvitskii, 2007). The authors operate under the assumption that if two distinct blocks in separate parts of the mine have similar attributes (such as grade of the material or the amount of deleterious elements in a block), the two distinct blocks will have the same destination in $\mathcal{S}$ or $\mathcal{P}$. For example, if two blocks have a grade of zero then both blocks will potentially be sent to the same destination – the waste dump. Therefore, we will be making the decision for extracting a block referring to the blocks and its destination is made by referring to its cluster.

In the following notation, material will flow from node $i \in \mathcal{N}$ to $j \in \mathcal{N}$ (material flows from node i to node j). $\mathcal{O}(i)$ represents the set of nodes that can receive materials from node i. $\mathcal{I}(j)$ is the set of nodes that can send material to node j.

Indices and sets of the model are:

- $m \in \mathbb{M}$ is a set of mines within a complex.
- $b \in \mathbb{B}_m$ is the set of blocks within a mine $m \in \mathbb{M}$.
- $t \in \mathbb{T}$ is a set of time periods, typically years, where $|\mathbb{T}|$ represens the life of mine of the complex.
- $u \in \mathbb{O}_b$ is the set of blocks overlaying block $b \in \mathbb{B}_m$.
- $s \in \mathbb{S}$ is a set of scenarios that represent a realization (simulation) of all sources of uncertainty. Specifically, for this model it is the uncertainty of the metal grade in a block that is accounted for (metal uncertainty). When using simulated orebody models, each scenario is equiprobable (Dimitrakopoulos, 2015).

- $p \in \mathbb{P}$ represent primary attributes, fundamental variables of interest sent through the model (such as metal content, tonnage). These attributes are typically linked directly with the attributes of the simulation and are always linearly transferred between parts of the mining complex. The value of primary attribute p from i at time t under period s is denoted as $v_{p,i,t,s}$. These attributes often originate at mines $m \in \mathbb{M}$ and may flow through the mining complex to the final products. The value of the attribute recovered after treatment is denoted by $r_{p,i,t,s}$.
- $h \in \mathbb{H}$ represents hereditary attributes. These attributes may be described as linear and non-linear functions of primary attributes, $f_{h,i}(v_{p,i,t,s})$, of the primary attributes. The value of the hereditary attribute h at location i at time t under scenario s is denoted as $v_{h,i,t,s}$.

The parameters of the model are defined as follows:

- $\varphi_{h,i,t}$ $\forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}$ represents the discounted revenue or expense associated with hereditary attribute h at a given node i in time t. Typically, with a given economic discount rate d_e, $\varphi_{h,i,t} = \frac{\varphi_{h,i,1}}{(1+d_e)^t}$.
- $U_{h,i,t}$ and $L_{h,i,t}$ $\forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}$ represent the upper and lower limits, or target, of attribute h at destination i in period t. For example, this could be a processor target.
- $c^+_{h,i,t}$ and $c^-_{h,i,t}$ $\forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}$ represent the cost (penalty) of deviation from a target (above and below the upper and lower targets respectively) for hereditary attribute h, at destination i, in period t. Here, the authors use a separate discount rate, called the risk discount rate d_r, to calculate the value of $c^+_{h,i,t}$ and $c^-_{h,i,t}$. That is, $c^+_{h,i,t} = \frac{c^+_{h,i,1}}{(1+d_r)^t}$ and $c^-_{h,i,t} = \frac{c^-_{h,i,1}}{(1+d_r)^t}$.
- $\beta_{p,b,s}$ $\forall p \in \mathbb{P}, s \in \mathbb{S}, b \in \mathbb{B}_m$ represents the amount of primary attribute p is in block b under scenario s.
- $\theta_{b,c,s}$ $\forall b \in \mathbb{B}_m, m \in \mathbb{M}, c \in \mathcal{C}, s \in \mathbb{S}$ is a pre-processed parameter to place a block into a cluster. For a given cluster c, if simulation s of block b is

determined to be a member of cluster c, $\theta_{b,c,s} = 1$, otherwise $\theta_{b,c,s} = 0$. It is understood that $\sum_{c \in \mathcal{C}} \theta_{b,c,s} = 1 \; \forall b \in \mathbb{B}_m, m \in \mathbb{M}, s \in \mathbb{S}$

In addition to the a priori parameters defined above, there is also a transformation function:

- $f_{h,i}(*) \; \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P}$ is a function for the hereditary attribute. This function is defined a priori. What this function does is take a value of a primary attribute $v_{p,i,t,s}$ and convert it into a hereditary attribute $v_{h,i,t,s}$. An example is recovery of metal from a processor.

Goodfellow & Dimitrakopoulos (2016) define three main decision variables in their solution vector. The solution vector is $\Phi = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$ where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ represent the decision variables in the stochastic integer program. The variables are defined as follows:

1. Extraction sequence decision variables ($\boldsymbol{x} \in \Phi$): $x_{b,t}$ is the extraction sequence decision variable where 1 represents mining block b in period t, 0 otherwise.
2. Destination policy decision variables ($\boldsymbol{z} \in \Phi$): $z_{c,j,t}$ is a binary variable where blocks in cluster c are sent to destination j in period t
3. Processing stream decision variables ($\boldsymbol{y} \in \Phi$): $y_{i,j,t,s}$ is a continuous variable between 0 and 1 indicating the proportion of material sent from node i to destination node j in period t under scenario (realization) s

Goodfellow & Dimitrakopoulos (2016) also define the following as variables whose values depend on both the realization of the metal content and the values of the three variables above.

- $v_{p,i,t,s} \; \forall p \in \mathbb{P}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}, s \in \mathbb{S}$ represent the value of the primary attribute p at a given node i in time t under scenario s.
- $v_{h,i,t,s} \; \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S}$ represents the value of the hereditary attribute h at a given node i in time t under scenario s.
- $\varphi_{p,c,t,s} \forall p \in \mathbb{P}, c \in \mathcal{C}, t \in \mathbb{T}, s \in \mathbb{S}$ is the quantity of the value attribute p in cluster c at time t under scenario s.

- $d^{+}_{h,i,t,s}$ and $d^{-}_{h,i,t,s}$ $\forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S}$ represent the value of deviation from a target (above and below the upper and lower targets respectively) for hereditary attribute h, at destination i, in period t, when scenario s occurs.
- $r_{p,i,t,s}$ $\forall p \in \mathbb{P}, i \in \mathcal{S}, t \in \mathbb{T}, s \in \mathbb{S}$ is a variable to mass balance primary attributes in the mining sequence. This can be viewed as the percent of recovery.

The model is defined as follows:

$$g(\Phi) = \max\left\{\frac{1}{|\mathbb{S}|}\sum_{i\in\mathcal{S}\cup\mathcal{P}\cup\mathbb{M}}\sum_{t\in\mathbb{T}}\sum_{h\in\mathbb{H}}\sum_{s\in\mathbb{S}}\varphi_{h,i,t}\cdot v_{h,i,t,s} - \frac{1}{|\mathbb{S}|}\sum_{i\in\mathcal{S}\cup\mathcal{P}\cup\mathbb{M}}\sum_{t\in\mathbb{T}}\sum_{h\in\mathbb{H}}\sum_{s\in\mathbb{S}}\left(c^{+}_{h,i,t}\cdot d^{+}_{h,i,t,s} + c^{-}_{h,i,t}\cdot d^{-}_{h,i,t,s}\right)\right\} \tag{5}$$

$$\sum_{t\in\mathbb{T}} x_{b,t} \le 1 \quad \forall b \in \mathbb{B} \tag{6}$$

$$x_{b,t} \le \sum_{t'=1}^{t} x_{u,t'} \quad \forall b \in \mathbb{B}_m, u \in \mathbb{O}_b, t \in \mathbb{T} \tag{7}$$

$$v_{p,m,t,s} = \sum_{b\in\mathbb{B}_m} \beta_{p,b,s}\cdot x_{b,t} \quad \forall m \in \mathbb{M}, p \in \mathbb{P}, t \in \mathbb{T}, s \in \mathbb{S} \tag{8}$$

$$\gamma_{p,c,t,s} = \sum_{b\in\mathbb{B}_m} \theta_{b,c,s}\cdot\beta_{p,b,s}\cdot x_{b,t} \quad \forall m \in \mathbb{M}, p \in \mathbb{P}, c \in \mathcal{C}, s \in \mathbb{S} \tag{9}$$

$$\sum_{j\in\mathcal{O}(c)} z_{c,j,t} = 1 \quad \forall c \in \mathcal{C}, t \in \mathbb{T} \tag{10}$$

$$r_{p,i,t,s} = 1 \quad \forall p \in \mathbb{P}, i \in \mathcal{S}, t \in \mathbb{T}, s \in \mathbb{S} \tag{11}$$

$$r_{p,i,t,s} = f_{h,i}\left(v_{p,i,t,s}\right) \quad \forall p \in \mathbb{P}, i \in \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S} \tag{12}$$

$$\sum_{j \in \mathcal{O}(i)} y_{i,j,t,s} \leq 1 \quad \forall i \in \mathcal{S}, t \in \mathbb{T}, s \in \mathbb{S} \tag{13}$$

$$\sum_{j \in \mathcal{O}(i)} y_{i,j,t,s} = 1 \quad \forall i \in \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S} \tag{14}$$

$$v_{p,j,(t+1),s} = \sum_{i \in (\mathcal{J}(j) \setminus \mathcal{C})} r_{p,i,t,s} \cdot v_{p,i,t,s} \cdot y_{i,j,t,s} + \sum_{i \in (\mathcal{J}(j) \cap \mathcal{C})} \varphi_{p,c,(t+1),s} \cdot z_{c,j,(t+1)} + \left(v_{p,j,t,s} \cdot \left(1 - \sum_{k \in \mathcal{O}(j)} y_{j,k,t,s} \right) \right) \quad \forall p \in \mathbb{P}, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S} \tag{15}$$

$$v_{h,i,t,s} = v_{p,i,t,s} \cdot \left(1 - \sum_{j \in \mathcal{O}(i)} y_{i,j,t,s} \right) \quad \forall i \in \mathcal{S}, t \in \mathbb{T}, s \in \mathbb{S} \tag{16}$$

$$v_{h,i,t,s} = f_{h,i}\left(v_{p,i,t,s}\right) \quad \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}, s \in \mathbb{S} \tag{17}$$

$$v_{h,i,t,s} - d^{+}_{h,i,t,s} \leq U_{h,i,t} \quad \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}, s \in \mathbb{S} \tag{18}$$

$$v_{h,i,t,s} + d^{-}_{h,i,t,s} \geq L_{h,i,t} \quad \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}, s \in \mathbb{S} \tag{19}$$

$$x_{b,t} \in \{0,1\} \quad \forall b \in \mathbb{B}_m, m \in \mathbb{M}, t \in \mathbb{T} \tag{20}$$

$$z_{c,j,t} \in \{0,1\} \quad \forall c \in \mathcal{C}, j \in \mathcal{O}(c), t \in \mathbb{T} \tag{21}$$

$$y_{i,j,t,s} \in [0,1] \quad \forall i \in \mathcal{S} \cup \mathcal{P}, j \in \mathcal{O}(i), t \in \mathbb{T}, s \in \mathbb{S} \tag{22}$$

$$\varphi_{p,c,t,s} \geq 0 \quad \forall p \in \mathbb{P}, c \in \mathcal{C}, t \in \mathbb{T}, s \in \mathbb{S} \tag{23}$$

$$r_{p,i,t,s} \in [0,1] \quad \forall p \in \mathbb{B}_m, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S} \tag{24}$$

$$v_{p,i,t,s} \geq 0 \quad \forall p \in \mathbb{P}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}, s \in \mathbb{S} \tag{25}$$

$$v_{h,i,t,s} \in \mathbb{R} \quad \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P} \cup \mathbb{M}, t \in \mathbb{T}, s \in \mathbb{S} \tag{26}$$

$$d^{+}_{h,i,t,s}, d^{-}_{h,i,t,s} \geq 0 \quad \forall h \in \mathbb{H}, i \in \mathcal{S} \cup \mathcal{P}, t \in \mathbb{T}, s \in \mathbb{S} \tag{27}$$

The first function is the objective function of the model, defined in (5). The first part of the objective function represents the discounted revenues and costs associated with the mining complex operation. The second part of the objective function represent the risk discounted penalties for deviations from production targets. Recall that the scenarios all have an equal probability of occurring.

The following three constraints are the mine extraction constraints. Constraints (6) represent the reserve constraint; i.e., a single block b can only be mined in one period t. Constraints (7) are called the block access constraints. Recall from Figure 2 in section 1.1 that the blocks in the overlying set $\mathbb{O}_b$ must be removed before or including the period t which we desire to extract block b. Constraints (8) convert the values of the primary attribute of blocks extracted in period t into the variable $v_{p,m,t,s}$ using $\beta_{p,b,s}$.

Constraints (9) and (10) are destination policy constraints. Constraints (9) are similar to the mine extraction constraints (8) as they determine the quantity of the material in a specific cluster and scenario. Constraints (10) ensures that a given cluster is sent to only one destination in a given period.

Constrains (11) to (17) are processing flow stream constraints. Constraints (11) represent the recovery of material at stockpile nodes in the mining complex. We typically assume that the recovery from a stockpile is always 100%. Constraints (12) represents the recovery of material from a processor node i. Recall from Section 1.1 that a processor's grade-recovery curve $f_{h,i}(*)$ can be non-linear. Constraints (13) and (14) are similar to the mining reserve constraint. Constraints (13) ensure that the proportion material sent from stockpile nodes are appropriately balanced. Constraints (14) ensure that the proportion of material sent from processing nodes are appropriately balanced. Constraints (15) represent the mass-balancing of material from mines to stockpiles and/or processors. Constraints (16) are used to calculate and represent the amount of material left in the stockpiles at end of the year. Finally, constraints (17) represent the amount of a primary attribute after applying some kind of transformation, such as those at a processor.

Constraints (18) and (19) represent the capacity constraints of the material at a given node i. Constraints (18) represent the upper bound of the equipment's ability to handle

material h at location i in period t and allow the deviation $d^{+}_{h,i,t,s}$ if the amount is exceeded. Conversely, constraint (19) represents a lower bound of the value (amount) of the attribute h the equipment is to handle at location i in period t and allow the deviation $d^{-}_{h,i,t,s}$ if the amount is less than the limit.

Finally, (20) to (27) represent variable definitions of the model. Constraints (20) and (21) are the binary decisions of the mining decision and destination decision, respectively. Constraint (22) is the continuous decision of the processing stream decision.

3.2 Solution Method

Optimizing the open pit mining complexes with metal uncertainty can be challenging to solve using exact methods. Often metaheuristics are used to optimize these mining complexes. Viewing a more simplified model, Lamghari & Dimitrakopoulos (2012) note the open pit mining problems can be seen as Precedence-Constrained Knapsack Problem (PCKP). The authors note the model is NP-Hard. Often, as mentioned by Lamghari & Dimitrakopoulos (2012) and in Goodfellow & Dimitrakopoulos (2016), metaheuristic methods employed to attain good solutions in a reasonable amount of time. Goodfellow & Dimitrakopoulos (2016) selected simulated annealing (SA) as the base method to optimize their new formulation. This method is selected because of previous success using SA to optimize extraction sequences of mining complexes. Referring to the SA specified in Algorithm 1, the neighborhood $N(\Phi)$, where the neighbor solution is selected, is partitioned into three neighborhoods n_x, n_y, or n_z is obtained by modifying a variable $x_{b,t} \in \boldsymbol{x}$, $y_{i,j,t,s} \in \boldsymbol{y}$, or $z_{c,j,t} \in \boldsymbol{z}$, respectively. At each iteration, one of the neighborhoods is selected randomly according to a probability distribution specified a priori by the user. The neighbor solution is obtained by modifying the current solution using a perturbation specific to the neighborhood. The perturbations are formally defined as follows:

1. Extraction sequence perturbations ($\boldsymbol{x} \in \Phi$): a block $b \in \mathbb{B}_m$ is randomly selected. A different period of extraction is then selected randomly for extracting b. There is a probability of changing the period to "not mining" block b. Moreover, some predecessor or successor blocks' periods, if block b is moved to an earlier or later period, respectively, may be adjusted to satisfy the slope constraint, constraints (7).

For example, if a block is moved from period 3 to period 2 and all the predecessor blocks are mined in period 1, the predecessor blocks will not change. However, if all the predecessor blocks are in period 3, then all the predecessor blocks would have to be adjusted to maintain slope constraints. Also bear in mind this will subsequently effect destination and processing stream decisions.

2. Destination policy perturbations ($\boldsymbol{z} \in \Phi$): a cluster's destination decision variable is randomly selected and sent to a different destination, if possible. A random variable $z_{c,j,t}$ is selected from the sub-vector $\boldsymbol{z} \in \Phi$ and then a new $j \in \mathcal{O}(c)$ is selected.

3. Processing stream perturbations ($\boldsymbol{y} \in \Phi$): a processing stream variable $y_{i,j,t,s}$ is randomly selected and its value is modified using a random normal number; i.e. $y_{i,j,t,s} \leftarrow N\left(y_{i,j,t,s}, 0.1\right) + y_{i,j,t,s}$. The authors note that the variance of the normal distribution is sufficiently small to allow both local and global exploration. After the selection and modification of the selected $y_{i,j,t,s}$ variable, the associated $y_{i,j',t,s}$ $\forall j' \in \mathcal{O}(j)$ are normalized based on equation (13) or (14).

Once the neighborhood is selected and the current solution Φ is modified to Φ' and the probability of accepting a neighbor solution Φ' is defined as follows:

$$P(g(\Phi), g(\Phi'), \delta_i) = \begin{cases} 1 & \text{If } \Phi' \text{is an improving solution} \\ e^{\frac{g(\Phi')-g(\Phi)}{\delta_i}} & \text{Otherwise} \end{cases} \quad (28)$$

Where $g(\Phi)$ and $g(\Phi')$ are the objective function values before and after the perturbation, respectively, and δ_i is the annealing temperature for a neighborhood i. In Goodfellow & Dimitrakopoulos (2016), rather than use a single value τ in the SA method (where the singular temperature for all neighborhoods and is cooled over time), the method will have three temperature values, one for each neighborhood. Some neighborhoods will have a much larger impact on the objective function value when selected. Therefore, having a constant temperature for all the neighborhoods could cause those neighborhoods with a smaller impact to be almost always accepted while the greater impact neighborhoods will only accept neighbor solutions which improve the current solution. So, the authors introduce a starting acceptance probability ρ instead of a starting temperature. The value ρ can be thought of as a "target probability of

acceptance" for a given set of iterations. The value ρ is identical for all neighborhoods and ρ (where $0 < \rho \leq 1$) is cooled by a parameter ε (where $0 < \varepsilon < 1$) every n^{iter} iterations. The temperature δ_i is calibrated using the reduction in objective function over the past n^{iter} iterations (that is, we only consider worsening solutions). The temperature δ_i is updated for each neighborhood is updated such as in (29).

$$\delta_i \leftarrow \frac{\overline{|\Delta g|}}{\ln(\rho)} \tag{29}$$

where $\overline{|\Delta g|}$ is the average reduction in objective function over the past iterations and $\ln(\rho)$ is the natural logarithm for ρ. The authors note that this better reflects the current search space rather than the search space when the SA algorithm began.

The stopping criteria for the SA is either when the global best update counter, k^{gbu}, reaches a specified count or the number of iterations of the SA, k, reaches a specified number, whichever comes first. After the simulated annealing is complete, the method checks to see if the method found a new global best solution. If no new best solution was found, the method terminates and returns the global best solution. However, if a new global best solution is found, then the SA method is reset and executed again with Φ^g as an initial solution. Each time a SA is executed to diversify the solution, we call this a global iteration (GI).

In Goodfellow & Dimitrakopoulos (2016), the authors develop three variations of their solver to optimize their model. The first method is the basic SA that was outlined in this section. It uses SA to optimize over all three variable sets. The other two variations to the solver use SA before applying a second metaheuristic to optimize the values of both the $\boldsymbol{y}$ and $\boldsymbol{z}$ variables, known collectively as downstream variables. These variations incorporate either differential evolution (DE) or particle swarm optimization (PSO) after each SA is executed. Recall that DE and PSO are better suited for continuous variables and also recall that $\boldsymbol{y}$ is a continuous variable. Note that in these two variations, the DE and PSO do not modify the extraction sequence variables, i.e. the variables in x.

Most population based metaheuristics have been problematic in mining problems because of decisions which have to be made surrounding repair operators in the precedence

constraints. Goodfellow & Dimitrakopoulos (2016) use DE and PSO only for downstream variables as to avoid such problems. The authors also note that as such, PSO and DE are sensitive to the initial sequences and destination policies generated for the population.

When Goodfellow & Dimitrakopoulos utilize DE (Storn & Price, 1997) in their algorithm, they see an approximate increase of 2.57% in the NPV of the resulting solution. The authors also note that while we gain 2.57% on the NPV, it takes approximately 2.9 times as long to complete the algorithm using the same criteria then just executing SA alone.

When Goodfellow & Dimitrakopoulos (2016) utilize particle swarm optimization (PSO) in their optimization process. PSO is another population based metaheuristic outlined in Khan & Niemann-Delius (2014). While PSO does achieve an increase in objective function value (1.91% when compared to SA alone), it does take on average 2.4 times as long to achieve the same stopping criteria.

The diversification method used in Goodfellow & Dimitrakopoulos (2016) is to re-run the selected variation (SA Only, SA+DE, or SA+PSO) of the method beginning from the global best solution found in the previous iteration. That is, the method takes the global best result x^* and re-initializes SA (with or without either PSO or DE) from the initial parameters, resetting the temperature and stopping criteria, using the previous found x^* as an initial solution in Algorithm 1. Here, we refer to each time the SA is reset and run again as a global iteration GI. An example of this can be seen in Algorithm 3.

Algorithm 3: Iterating Simulated Annealing many times

```
Initialization:
      Select an initial solution x^0 ∈ X
      Let x ← x^0; x* ← x^0
While stopping criteria is not met
      x ← x*
      NewGBS ← False
      While stopping criteria is not met from Algorithm 1
            Execute SA Similar to Algorithm 1
            If a new global best solution (GBS) is found in Algorithm 1
                  NewGBS ←True
      If not (NewGBS)
            Return x*
```

This method of diversification allows for the stopping criteria being the number of global best updates to be an acceptable choice because the individual SAs start with the global best solution. if further intensification is possible then it can be picked up on the next global iteration. otherwise, it allows for the method to work away from a local maximum every i^{gbu} iterations.

A general algorithm can be seen in Algorithm 4:

Algorithm 4: Simulated annealing as developed by Goodfellow & Dimitrakopoulos (2016)

```
Build Φ ← {x, y, z} ∀ i ∈ N, generate an initial solution
Set Φ^g ← Φ where Φ^g is the global best solution
While True
      Φ ← Φ^g
      NewGBS ← False, keeps track of a new global best solution (GBS)
      k, k^gbu ← 0, keeps track of iterations i and number of times there is
                  A new GBS i^gbu
      Print "Beginning SA", we begin a global iteration here (GI)
      While Stopping Criteria is not Met
            Select a neighborhood x, y, or z with a fixed probability
            Select a variable in the selected neighborhood to modify
            Store the modified solution as Φ'
            If g(Φ') ≥ g(Φ)
                  Φ ← Φ'
                  If g(Φ') ≥ g(Φ^g)
                        Φ^g ← Φ', NewGBS ← True, k^gbu ← k^gbu + 1
            Else If P(g(Φ), g(Φ'), δ_i) ≥ U{0,1}, Accepted by APF
                  Φ ← Φ'
            If needed, cool the temperature ρ and δ_i
            k ← k + 1
      If UseDE
            Execute DE on downstream variables
      Else If UsePSO
            Execute PSO on downstream variables
      If Not(NewGBS)
            Return Φ^g
```

3.3 Implementing an Adaptive Neighborhood Selection Procedure into the Mine Complex Optimization Procedure

This section will introduce an adaptive procedure for selecting neighborhoods. This section uses the method outlined by Lamghari & Dimitrakopoulos (2015) and is an expansion on the method introduced in section 2.2. More explicitly, we will expand on the assignment of

the value of σ from Algorithm 2 on page 16 The value assigned to the measure is a function of the change of the objective function value ($\Delta g = g(\Phi') - g(\Phi)$). We will modify the scores differently depending on the result of the SA; accept outright, conditionally accept, or reject. Using these scores, we will apply a roulette-style selection method for selecting the neighborhood to search.

Lamghari & Dimitrakopoulos (2015) use a method to adaptivey select heurstics in their method for solving mining complexes. In addition, they draw from methods outlined by Burke, et al. (2013) and Drake, et al. (2012). They update the scores of the neighborhoods as follows. As initially discussed in section 2.2, the neighborhoods are grouped together in such a way where a small perturbation, or called a low-level heuristic here, is applied to a current solution to get a neighbor solution. The authors denote the low-level heuristic by h_j. The notation $\Delta g(h_j)$ is the difference in the value of the current solution and the neighbor generated using h_j. The authors also use the time as part of their measures, where $T(h_j)$ as the time (in seconds) it takes for a low-level heuristic to be applied. Note that it may take a while for some low-level heuristics to apply and repair a solution. The authors also introduce two unique measures to keep track of the low-level heuristics, $\pi_1(h_j)$ and $\pi_2(h_j)$. Both measures are set initially to zero. The authors update the measures based on whether the neighbor solution is an improving solution or not. Therefore, they have two cases, which are as follows: Suppose that if a heuristic creates a neighbor solution with an improving objective function, we then increase $\pi_1(h_j)$ by $\Delta g(h_j)/T(h_j)$. Conversely, suppose that if a heuristic creates a neighbor solution with a non-improving objective function, we then increase $\pi_2(h_j)$ by $1/|\Delta g(h_j)|T(h_j)$.

$$\text{Measure incriment cases} \begin{cases} \pi_1(h_j) \leftarrow \pi_1(h_j) + \dfrac{\Delta g(h_j)}{T(h_j)} & \text{If } \Delta g(h_j) \geq 0 \\ \pi_2(h_j) \leftarrow \pi_2(h_j) + \dfrac{1}{|\Delta g(h_j)|T(h_j)} & \text{Otherwise} \end{cases} \tag{30}$$

If we analyze both cases in (30) for incrementing the measure, the first one (i.e. if $\Delta g(h_j) > 0$) is straightforward. The better the increase in objective function value and the less time it takes to find an update, the greater the measure. The second case (i.e. "otherwise") emphasizes

minimal deterioration to the objective function value. That is, the smaller the reduction in the objective function, and the shorter the time to find a solution, the greater the measure.

In this thesis, we modify (30) to better match the cases in SA and remove the time value from the measures. The decision to remove this was based on the fact the neighbor solution creation time was inconsistent even within a neighborhood. The following notation also redefines the "low-level heuristic h_j" as neighborhood n_i. Recall that in SA, there are three cases that can occur when deciding to accept the neighbor solution: improving, worse solution but accepting, and rejecting the neighbor solution. Therefore, we have added a third case into (30) to reflect the three outcomes of SA. That is, the measure increment cases are defined as follows:

$$\text{Measure incriment cases for ANS in SA} \begin{cases} \pi_1(n_i) \leftarrow \pi_1(n_i) + \Delta g(n_i) & \text{if } \Delta g(n_i) \geq 0 \\ \pi_1(n_i) \leftarrow \pi_1(n_i) + \dfrac{1}{|\Delta g(n_i)|} & \text{if } \Delta g(n_i) < 0 \text{ and accepted} \\ \pi_2(n_i) \leftarrow \pi_2(n_i) + \dfrac{1}{|\Delta g(n_i)|} & \text{Otherwise} \end{cases} \quad (31)$$

In both Lamghari & Dimitrakopoulos (2015) and this thesis, the values of the score modification $\pi_1(n_i)$ and $\pi_2(n_i)$ (associated with the success and failure of using n_i, respectively) are specified as follows: at the beginning of the period of $(ScoreUpdate)_{skip}$ iterations, the value of $\pi_1(n_i)$ and $\pi_2(n_i)$ are initialized to zero. In this thesis specifically, each time a neighbor solution in neighborhood n_i is selected, in addition to incrementing $\kappa(n_i)$, one of three cases will happen:

1. Suppose that using neighborhood n_i leads to an improvement of the current solution. That is, we outright accept the neighbor solution. Then $\pi_1(n_i)$ is increased by $|\Delta g|$.
2. Suppose that using neighborhood n_i leads to a non-improving solution, $\Delta g < 0$, but the SA method accepts the neighbor solution to be the current solution based on the APF, then $\pi_1(n_i)$ is increased by $1/|\Delta g|$.

3. Finally suppose that using neighborhood n_i leads to a non-improving solution, $\Delta g < 0$, and keeps the current solution (rejects the neighbor solution due to the APF), then $\pi_2(n_i)$ is increased by $1/|\Delta g|$.

At each $(ScoreUpdate)_{skip}$ iterations, the modification of score s_i associated with each neighborhood $n_i, \forall i \in \mathbb{N}$ (where $\mathbb{N}$ is the set of neighborhoods allowed) is similar to the process implemented in Lamghari & Dimitrakopoulos (2015). To be more explicit, assume that the scores are updated after each period of $(ScoreUpdate)_{skip}$ iterations of SA. Let $\kappa(n_i)$ the number of times that n_i is selected during the $(ScoreUpdate)_{skip}$ iterations. The term to modify the score s_i is specified in equation (32) here.

$$s_i \leftarrow \begin{cases} (1-\alpha)s_i + \alpha\left(\dfrac{\beta\,\pi_1(n_i) + (1-\beta)\pi_2(n_i)}{\kappa(n_i)}\right) & \text{If } \kappa(n_i) > 0 \\ s_i & \text{Otherwise} \end{cases} \tag{32}$$

In equation (32), $\alpha \in [0,1]$ is a static parameter that the user can define to specify how much emphasis to place on newer information (that is α being closer to 1) versus on historical information (that is α being closer to 0). The variable $\kappa(n_i)$ is the number of times a neighborhood n_i is called. If $\kappa(n_i) = 0$, then $s_i \leftarrow s_i$. After $(ScoreUpdate)_{skip}$ iterations, the score s_i of neighborhood n_i is updated according to equation (32). The self-adjusted parameter $\beta \in [0,1]$ is the impact of the successful versus unsuccessful neighborhoods. The α remains static in the method and β changes based on the last update to the local best solution. If:

1. A new local best is found in the last $(ScoreUpdate)_{skip}$ iterations, β is set to 1.
2. No new local best solution is found in the $(ScoreUpdate)_{skip}$ iterations, β is reduced by 0.1 until it reaches zero. That is $\beta \leftarrow \max[\beta - 0.1, 0.0]$.

The count is reset to zero each time the scores s_i are updated. Keeping in mind that $\pi_1(n_i)$ represents a heuristic's "success" score and $\pi_2(n_i)$ represents a heuristic's "failure" score, let us look at this sub-part of the equation (32) above:

$$\beta\,\pi_1(n_i) + (1-\beta)\pi_2(n_i) \tag{33}$$

We can see in (33) above that when β is closer to 1, more weight is placed on the successful heuristics. That is, we can view this as more emphasis is placed on intensifying the search of neighborhoods using recently successful heuristics. As β is reduced and ultimately reaches zero, more weight is placed on searching neighborhoods of the unsuccessful heuristics. However, these heuristics are the ones which would have not reduced the overall objective function value by a large amount if the neighbor solutions were accepted (that is, unsuccessful heuristic closer to zero). This emphasis placed on better quality but failing heuristics can be viewed as a diversification method.

After the value of β has been updated, the method then updates the scores for each neighborhood by equation (32). If the count $\kappa(n_i)$ is equal to zero, i.e. n_i was not called in the last $(ScoreUpdate)_{skip}$ iterations, then the score remains unchanged. Once the scores have been updated, the method updates the probabilities p_j for the next $(ScoreUpdate)_{skip}$ iterations using a roulette-style method, such as in (34).

$$p_j \leftarrow \frac{s_j}{\sum_{k \in \mathbb{N}} s_k} \; \forall j \in \mathbb{N} \tag{34}$$

The ANS in SA then proceeds as follows: at the beginning of each global iteration, the initial probabilities in scores are set to be equally probable. Here, this means the initial probability for all the neighborhoods is set to 33% each. After each perturbation in the SA, we will modify either $\pi_1(n_i)$ or $\pi_2(n_i)$ for the selected neighborhood as depicted above. After $(ScoreUpdate)_{skip}$ iterations, the score will be updated as equation (32) depicts and the probabilities will be updated similarly to (34). Continue solving the SA until the stopping criteria is met; this stopping criteria remains identical to the previous method. After SA is finished, DE or PSO can be executed if desired. As with Goodfellow & Dimitrakopoulos (2016), we will continue to diversify until no further global best solution can be found. While a detailed pseudocode is outlined in the Appendix B, placing the ANS method into Algorithm 4 gives us Algorithm 5. Lines in bold were added to Algorithm 4 to highlight the differences.

Algorithm 5: Simulated annealing with an adaptive neighborhood search method for optimizing stochastic mining complexes

Build $\Phi \leftarrow \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\} \forall\, i \in N$, generate an initial solution
Set $\Phi^g \leftarrow \Phi$ where Φ^g is the global best solution
While True
    $NewGBS \leftarrow$ False, keeps track of a new global best solution (GBS)
    $k, k^{gbu} \leftarrow 0$, keeps track of iterations i and number of times there is
        A new GBS i^{gbu}
    $\boldsymbol{ResetBeta} \leftarrow$ **False, keeps track if we need to set beta to 1.0**
    $\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3} \leftarrow \left(\frac{1}{3}\right), \boldsymbol{\beta} \leftarrow \mathbf{0.5}, \boldsymbol{\alpha} \leftarrow \mathbf{0.7}$
    Print "Beginning SA", we begin a global iteration here (GI)
    While Stopping Criteria is not Met
        If $\boldsymbol{i}$ **mod** $(\boldsymbol{ScoreUpdate})_{\boldsymbol{skip}}$ **= 0**
            If $\boldsymbol{ResetBeta}$
                $\boldsymbol{\beta} \leftarrow \mathbf{1.0}$, $\boldsymbol{ResetBeta} \leftarrow$ **False**
            Else
                $\boldsymbol{\beta} \leftarrow \max[\mathbf{0.0}, \boldsymbol{\beta} - \mathbf{0.1}]$
            Update the scores of all the neighborhoods by Eq (32)
            $\boldsymbol{\pi_1}(\boldsymbol{n_i}), \boldsymbol{\pi_2}(\boldsymbol{n_i}), \boldsymbol{\kappa}(\boldsymbol{n_i}) \leftarrow \mathbf{0}$ **for all the neighborhoods**
            Update probabilities $\boldsymbol{p_i}$ **for the neighborhoods by Eq (34)**
        Select a neighborhood $\boldsymbol{x}, \boldsymbol{y},$ **or** $\boldsymbol{z}$ with **adapted probabilities** $\boldsymbol{p_i}$
        Select a variable in the selected neighborhood n_i to modify
        Store the modified solution as Φ'
        If $g(\Phi') \geq g(\Phi)$
            $\Phi \leftarrow \Phi'$
            $\boldsymbol{\pi_1}(\boldsymbol{n_i}) \leftarrow \boldsymbol{\pi_1}(\boldsymbol{n_i}) + \boldsymbol{\Delta g}$
            $\boldsymbol{ResetBeta} \leftarrow$ True
            If $g(\Phi') \geq g(\Phi^g)$
                $\Phi^g \leftarrow \Phi'$, $NewGBS$, $k^{gbu} \leftarrow k^{gbu} + 1$
        Else If $P(g(\Phi), g(\Phi'), \delta_i) \geq U\{0,1\}$, Accepted by APF
            $\Phi \leftarrow \Phi'$
            $\boldsymbol{\pi_1}(\boldsymbol{n_i}) \leftarrow \boldsymbol{\pi_1}(\boldsymbol{n_i}) + \frac{\mathbf{1}}{|\boldsymbol{\Delta g}|}$
        Else
            $\boldsymbol{\pi_2}(\boldsymbol{n_i}) \leftarrow \boldsymbol{\pi_2}(\boldsymbol{n_i}) + \frac{\mathbf{1}}{|\boldsymbol{\Delta g}|}$
        If needed, cool the temperatures ρ and δ_i
        $k \leftarrow k + 1$
    If $UseDE$
        Execute DE on downstream variables
    Else If $UsePSO$
        Execute PSO on downstream variables
    If Not($NewGBS$)
        Return $\Phi^g \leftarrow \Phi$

4 Numerical Results

To test the method outlined in this thesis, we ran the method on three deposits. There were two single element deposits, one copper and one gold, and one multi-element deposit, a copper-gold deposit. The copper-gold deposit is the same deposit used by Goodfellow & Dimitrakopoulos (2016) to test their model and method.

4.1 Copper-Gold Deposit

The following figures and tables represent the copper-gold deposit. Figure 3 summarizes the mine complex materials and the processing options. In short, there are two elements in the deposit, copper (Cu) and gold (Au). There are four processing options (one mill and three leach pads), two waste dump options, and a stockpile to feed only one processor. The mine contains three material groups: sulfides, transition, and oxides. They are separated into these groups because of the chemistry constraints on the materials in the mine. In order to respect the chemistry requirements at the sulfide heap leach (processor), the sulfide and transition material groups are both separated into two different material types based on being above or below 0.2% copper. The oxide materials are classified as ore or waste based on chemistry.

With the exception of the waste dumps, all processors have variable grade-recovery curves that are based on the average grade of the incoming material in a period. In general, the higher the grade of the material into the processor, the higher the recovery of the processor.

All cost-related parameters in Table 1 and Table 2 are expressed relative to the mining cost for confidentiality purposes. Table 2 summarizes the constraints and penalty costs used in the models. A risk discount rate of 10% is used to penalize the deviations from the production capacities, and ensures that riskier material is deferred to later periods when more information is available (such as economic, geological, processing, etc.). The mine model contains 34,057 blocks that may be scheduled over 22 years. Also, 25 simulated orebody models (scenarios) were used in the SIP, similar to what has been done in Goodfellow & Dimitrakopoulos (2016). The simulations were provided by the industry partners and were generated using a sequential conditional simulation method. Note that 25 realizations are sufficient to capture metal uncertainty as previous studies, such as that of Consuegra & Dimitrakopoulos (2009), show that

after about 15 realizations, stochastic schedules converge to a stable final physical schedule and stable production forecasts. Finally, a slope angle of 45◦ is used.

Unless otherwise noted, the copper-gold deposit uses a "starter schedule." It was developed using a deterministic scheduler. The "starter schedule" is an initial extraction sequence. That is, it is a file which denotes a period each block is extracted thus giving an initial solution to the $x_{b,t}$ variables.

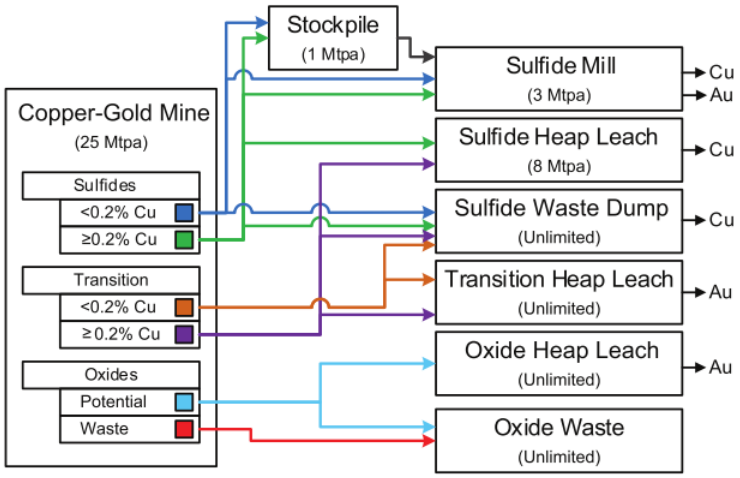

Figure 3: Definition of material types at the copper-gold mine, along with the various destinations (Goodfellow & Dimitrakopoulos, 2016)

Table 1: Economic Parameters of the Model

Economic Parameters	Value
Mining Cost*	$1.00/t
Sulfide Mill*	$11.30/t
Sulfide Heap Leach*	$2.98/t
Transition heap leach*	$2.15/t
Oxide heap leach*	$2.06/t
Gold Price	$1480/oz.
Copper Price	$2.88/lb.

* For confidentiality, this parameter is normalized to the mining cost

Economic discount rate	7%
Risk discount rate	10%

Table 2: Lower and upper bound on constraints (18) and (19) and associated penalties

Constraint	$L_{h,i,t}$ ($\times 10^6$)	$U_{h,i,t}$ ($\times 10^6$)	$c^-_{h,i,1}$ (\$/tonne)	$c^+_{h,i,1}$ (\$/tonne)
Mine Capacity		25.0		10
Stockpile Capacity		1.0		20
Sulfide Mill Capacity	2.8	3.0	50	50
Sulfide Heap Leach Capacity	7.8	8.0	10	25

4.2 Single Element Deposits

This section outlines the single element copper and single element gold deposit used in this thesis. Both deposits are very similar to each other, varying only in the financial parameters and the grade of the material.

These deposits have a very simple chemistry as compared to the copper-gold deposit. Figure 4 summarizes the mine complex's processing options. Both deposits will have the same processing stream decisions and parameters.

There is one mine, two processor options each with a stockpile, and a waste dump. Each stockpile sends material to a unique processor. In both complex problems, the processors have a fixed recovery with the Plant at 90% and the Leach Pad at 55%.

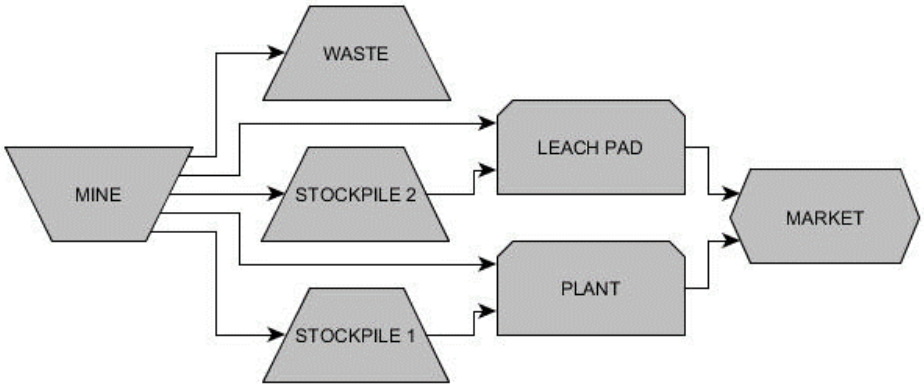

Figure 4: Illustration of the complexes associated with the single element deposits

All cost-related parameters in Table 3 (Copper) and Table 5 (Gold) are expressed relative to their mining cost for confidentiality purposes. The elements of Table 4 (Copper) and Table 6 (Gold) summarize the constraints and penalty costs used in the models. The copper mine model contains 28,154 blocks, the LOM is 16 years, and a slope angle of 45° is used. The gold mine model contains 48,821 blocks, the LOM is 14 years, and a slope angle of 45° is used.

For both deposits, there are 20 orebody simulations (scenarios) which were used in the solution method. The full 20 realizations were utilized as they are what was provided by the industry partners. There was no initial extraction schedule used for these deposits. That is, the extraction sequence was decided completely in the method by setting the initial $\boldsymbol{x}$ variables to "not mined."

Table 3: Economic Parameters of the Model (Copper)

Economic Parameters	Value (Copper)
Mining Cost†	$1.00/t
Leach Pad†	$3.21/t
Plant†	$12.86/t
Mine to Processor (any) †	$0.43/t
Mine to Stockpile (any) †	$0.43/t
Stockpile to Processor†	$0.64/t
Processor to Market†	$944.83/t
Metal Price	$2.00/lb.
Economic discount rate	10%
Risk discount rate	10%

† For confidentiality, this parameter is normalized to the mining cost

Table 4: Lower and upper bound on constraints (18) and (19) and penalties (Copper)

Constraint	$L_{h,i,t}$ ($\times 10^6$)	$U_{h,i,t}$ ($\times 10^6$)	$c^-_{h,i,1}$ (\$/tonne)	$c^+_{h,i,1}$ (\$/tonne)
Mine Capacity		20.9		10
Stockpile 1 Capacity		1,139.1		10
Stockpile 2 Capacity		629.1		10
Leach Pad Capacity		2.0		10
Plant Capacity		3.6		10

Table 5: Economic Parameters of the Model (Gold)

Economic Parameters	Value (Gold)
Mining Cost‡	\$1.00/t
Leach Pad†	\$8.57/t
Plant†	\$21.43/t
Mine to Processor (any) †	\$0.43/t
Mine to Stockpile (any) †	\$0.36/t
Stockpile to Processor†	\$0.64/t
Processor to Market†	\$0.29/g
Metal Price	\$42.86/g
Economic discount rate	10%
Risk discount rate	10%

‡ For confidentiality, this parameter is normalized to the mining cost

Table 6: Lower and upper bound on constraints (18) and (19) and penalties (Gold)

Constraint	$L_{h,i,t}$ ($\times 10^6$)	$U_{h,i,t}$ ($\times 10^6$)	$c^-_{h,i,1}$ (\$/tonne)	$c^+_{h,i,1}$ (\$/tonne)
Mine Capacity		21.6		10
Stockpile 1 Capacity		69.8		10
Stockpile 2 Capacity		936.3		10
Leach Pad Capacity		0.2		10
Plant Capacity		3.3		10

4.3 Implementation and Parameters

All the methods were developed in C++ using Visual Studio 2015 Community edition. The computer used was a commercially available Dell Inspiron 24 7000 Series All-in-One. The specifications are as follows:

Table 7: Computer Used

Processor	Intel® Core™ i7-4710MQ CPU @ 2.50GHz
Processor Cores	4 Cores
RAM	12.0 GB
Operating System	Windows 8.1 Enterprise (x64)

The parameters for the metaheuristics are as follows:

Table 8: Simulated Annealing Parameters

Initial annealing acceptance probability (ρ)	0.40[§]; 0.30
Cooling Factor (ε)	0.99
Cooling iterations (n^{iter})	600
Perturbation probability - extraction sequence ($prob^{seq}$)[**]	0.30[§]; 0.50
Perturbation probability - destination policy ($prob^{dest}$)[††]	0.60[§]; 0.40
Annealing global best updates before diversification (i^{gbu})	2,000;
Total annealing iterations before diversification (i^{total})	500,000

In the implementation from Goodfellow & Dimitrakopoulos (2016), the SA algorithm is executed multiple times to diversify the solution. Recall that after each SA that is executed, if a new global best solution is uncovered, the SA program is reset and then run again. The initial acceptance probability and the probabilities to select the neighborhood in the first execution of SA differs from the ones used in the subsequent executions. The probabilities of the first run of SA are marked with § in Table 9. It is clear that the authors recognize the use of a single set of parameters for their method is improved by having multiple sets of parameters for different global iterations. I.e., they are manually adapting their parameters to improve their method. Of course, this thesis takes the adaptive concept further by adjusting the parameters "on the fly," that is, altering the parameters during execution.

Table 9: Adaptive Neighborhood Search Parameters

New information Smoothing (α)	0.70
Initial intensification/diversification decision (β)	0.5
Iterations between score updates $(ScoreUpdate)_{skip}$	See sensitivity analysis

[§] Goodfellow & Dimitrakopoulos (2016) use a different set of parameters for the first run of SA to compensate for the large number of blocks to be moved in the extraction sequence.

[**] Does not exist in ANS versions

[††] Does not exist in ANS versions

The values from Table 9 were taking from the work of Lamghari & Dimitrakopoulos (2015). Using these values, specifically the value of α, produced better results than the standard version outlined by Goodfellow & Dimitrakopoulos (2016). The value of β is less sensitive to its initial value because it is a self-adjusted parameter. Furthermore, the method usually found a new global best solution in the first $(ScoreUpdate)_{Skip}$ iterations of the SA. Therefore, one can view the de facto initial value of β to be 1. Recall that β is set to 1 if there is a new global best solution found in the last $(ScoreUpdate)_{Skip}$ iterations.

4.4 Results

The following section is broken down into three tests; Basic Implementation – Simulated Annealing Only, Using Differential Evolution, and Using a Random Initial Probability. All the tests use the copper-gold deposit outlined in section 4.1. The single element gold and single element copper deposit were only tested in the Basic Implementation – Simulated Annealing Only tests.

4.4.1 Basic Implementation – Simulated Annealing Only

The basic implementation tests were simply executing the SA method posed by Goodfellow & Dimitrakopoulos (2016) (see Algorithm 4 in section 3.2) and then the SA method with ANS posed in this thesis (see Algorithm 5 in section 3.3). In the following tables uses several *ANS* trials with a sensitivity over different values of $(ScoreUpdate)_{skip}$. The values used were 10, 33, 100, 333, and 1000. The ANS begins with each neighborhood having an equiprobable start. In this case, each neighborhood has a 1/3 chance of being selected. The first row uses the notation *GD SA* to indicate the standard SA method outlined by Goodfellow & Dimitrakopoulos (2016) with the static neighborhood probability parameters from Table 8. Static refers to the fact the method does not update the probabilities in a single SA. *GI* stands for global iteration, or the number of times simulated annealing is reset to diversify the global best solution. *ObjFn* refers to the objective function value, with the column marked *Avg* is the numerical average and *Max* is the maximum value found over the trials. For each method, 25 trials were run. The best values in each column are in bold, which is the maximum value for

ObjFn (Avg) and *ObjFn (Max)* columns and the minimum value for *GI* and *Time (s)* columns. All the tables are normalized to the *GD SA* method for confidentiality purposes.

Table 10: Copper-Gold deposit results

Run Type	*ObjFn (Avg)*	*ObjFn (Max)*	*GI*	*Time (s)*
GD SA	1.0000	1.0181	**2.20**	4961
ANS 10	1.0209	1.0699	2.72	**364**
ANS 33	1.0808	1.1074	2.64	564
ANS 100	1.0956	**1.1269**	2.60	868
ANS 333	**1.1010**	1.1205	2.60	955
ANS 1000	1.0987	1.1179	2.76	938

Table 11: Copper deposit results

Run Type	*ObjFn (Avg)*	*ObjFn (Max)*	*GI*	*Time (s)*
GD SA	1.0000	1.0276	2.58	**71**
ANS 10	0.8911	1.0564	2.36	144
ANS 33	0.7831	1.0480	**2.32**	207
ANS 100	0.8830	1.0682	2.44	156
ANS 333	0.9762	1.0643	2.34	95
ANS 1000	**1.0149**	**1.0743**	2.40	93

Table 12: Gold deposit results

Run Type	*ObjFn (Avg)*	*ObjFn (Max)*	*GI*	*Time (s)*
GD SA	1.0000	1.0087	2.40	**1504**
ANS 10	0.6635	**1.0193**	**2.00**	1700
ANS 33	0.7630	1.0180	2.16	2163
ANS 100	0.8870	1.0165	2.20	2486
ANS 333	**1.0122**	1.0182	2.12	2328
ANS 1000	1.0071	1.0171	2.20	2092

Table 13: Copper-Gold deposit results with no starter schedule

Run Type	*ObjFn (Avg)*	*ObjFn (Max)*	*GI*	*Time (s)*
GD SA	0.8643	0.8996	**2.08**	1166
ANS 10	0.0036	0.0640	2.32	**340**
ANS 33	0.3636	0.9008	2.40	713
ANS 100	0.8148	0.9214	2.32	1963
ANS 333	**0.8713**	**0.9246**	2.24	1722
ANS 1000	0.8547	0.8913	2.12	1302

Starting with the copper-gold deposit (Table 10), irrespective of the $(ScoreUpdate)_{skip}$ value used by the ANS, there is an increase in the objective function value. We can also see, on average, about a 10% increase in objective function value when a $(ScoreUpdate)_{skip}$ value of 333 is used. The average execution time was 955 seconds (about 16 minutes) versus 4,961 seconds (about 1.38 hours) for a static search. Recall that the stopping criteria for a single SA iteration (a single GI) is the number of global best updates found, Therefore, it can take a while for the SA to finish in the *GD SA*. With a reduction of 80% in the time to execute the method, we can run about five *SA* with *ANS 333* before a single *GD SA* is run. With executing these five, we can then choose the schedule with the maximum value. This thought process leads us to show the maximum objective function value of each method. We continue to see about a 10% increase in objective function value when comparing to the maximum value of the static SA, over a 12% increase when comparing to the average value of the static SA.

To test the replicability of the method, the single element copper and the single element gold deposits were tested (Table 11 and Table 12, respectively). In the copper deposit (Table 11), we see about a 1% increase, on average, of the objective function using $(ScoreUpdate)_{skip}$ value of 1000. In those runs, we see an increase in time to solve the method (about a 28% increase, on average). If we look at the maximum objective function results, we see a 3% to 5% increase in objective function value across all values of the $(ScoreUpdate)_{skip}$ with the best coming at a $(ScoreUpdate)_{skip}$ value of 1000. In the gold deposit (Table 12), we again see about a 1% increase, on average, of the objective function using a $(ScoreUpdate)_{skip}$ value of

333. In those runs, we see an increase in time to solve the method (about a 55% increase, on average). If we look at the maximum objective function results, we see a 1% increase in objective function value across all values of the $(ScoreUpdate)_{skip}$ with the best coming at a $(ScoreUpdate)_{skip}$ value of 10.

Looking at the results from the copper-gold deposit and the single element deposits, we see a 10% increase with the former, however, only about a 1% increase with the latter. One of the differences between the deposits is the use of a starter schedule, with the other differences being size and the number of downstream decisions. To attempt to account for this, we re-ran the copper-gold deposit without a starter schedule and posted the results in Table 13. Here we see under a 1% increase in objective function value with an increase in solving time. If we look at the schedules with the maximum value, we see an increase of about 2% over the maximum static schedule. Note that in this test, when $(ScoreUpdate)_{skip}$ is set to a value of 10 and 1000, there is a reduction in quality with the value of 10 producing a very poor quality schedule.

The source of discrepancy between the use of the starter schedule and not having to use the starter schedule can be associated with the solver being hampered with having to establish an initial extraction sequence. That is, decisions around processing streams have little impact until an initial extraction sequence is discovered. If the method were to apply a change to a downstream variable, there will be no change in the value of the complex if the associated block has not been extracted. In addition to the use of a starter schedule, an additional source of the discrepancy in objective function value improvements between simple single element deposits and the complicated copper-gold deposit can be associated with how the simple mine will have significantly less decision variables to manage. Therefore, the methods will begin to converge on similar solutions

4.4.2 Using Differential Evolution

Due to the success seen in Goodfellow & Dimitrakopoulos (2016) using differential evolution (DE) to assist in both diversification and solving the downstream variables ($\boldsymbol{y}$ and $\boldsymbol{x}$), we also implemented differential evolution into the ANS solution method. Recall, after executing a SA algorithm to its completion, DE is then executed. Each method was executed 10 times. *GD SA /w DE* is the SA method outlined by Goodfellow & Dimitrakopoulos (2016)

with DE. *ANS 100 /w DE* is the SA method outline in this thesis with ANS and a $(ScoreUpdate)_{skip}$ value of 100. After each SA with ANS, a DE is then executed. The copper-gold deposit used a starter schedule for both trials. The results in Table 14 are normalized to the objective function value from Table 10's GD Default SA's average. This can give us a quick way to inspect and compare methods.

Table 14: Copper-Gold Deposit with Differential Evolution

Run Type	*ObjFn (Avg)*	*ObjFn (Max)*	*GI*	*Time (h)*
GD SA /w DE	1.0614	1.0781	18.90	26.23
ANS 100 /w DE	**1.1541**	**1.1651**	**18.10**	**24.39**

When using DE, ANS on average yields about an 8% increase in objective function value over the GD SA method and has a reduction of 7% in solving time. When looking back to the non-DE methods, take note that the increase in objective function value does come at a high cost in the solving time. As Goodfellow & Dimitrakopoulos (2016) stated, implementing the population-based DE to solve the mine scheduling problem does increase the computational time. DE does provide a better method for solving continuous variables, such as the processing stream (***y***) variables in the model. This ability to solve the downstream variables to, quite often, a better solution enables the solution method to find a new global best solution in each global iteration. Because of finding this new solution there are many more global iterations (GI) that are executed and as a result, a longer time is required.

Recall the results from Table 10; Here we can see about a 5% increase in objective function value from the method without DE to using DE, but it takes much longer to find this increase. Recall that the copper-gold mine is stated to be a real-world mine. The objective function has a value in the order of 10^{10}, or a deposit with a valuation of a billion dollars. From a practical, real world perspective, this increase is well "worth it." That is, the extra time in whole numbers (about a day to find a solution) is very tolerable for this increase in objective function. Even if we operate with the procedure of executing several methods and pick the greatest, we can spend a few weeks to uncover the best solution the method can find.

4.4.3 Using a Random Initial Probability

To test the ability of ANS to work out of poor starting situations, tests were executed which had a random initial probability, noted as *RS* (Random Start). That is, instead of each neighborhood beginning with an equiprobable chance of being selected, for the first $(ScoreUpdate)_{skip}$ iterations, each neighborhood was given a random probability. This is noted as *ANS 100 /w RS* and *ANS 1000 /w RS* in Table 15. To compare, 25 trials were executed and averaged for *ANS 100 /w RS* and *ANS 1000 /w RS*. To give it a basis for comparison, *ANS 100* and *ANS 1000* were copied from Table 10. Once again, the results in Table 15 are normalized to the objective function value from Table 10's *GD SA*'s average.

Table 15: Copper-Gold Deposit with Random Start

Run Type	*ObjFn (Avg)*	*ObjFn (Max)*	*GI*	*Time (s)*
ANS 100	**1.0956**	**1.1269**	**2.60**	**868**
ANS 100 /w RS	1.0638	1.1019	**2.60**	870
ANS 1000	**1.0987**	**1.1179**	2.76	**938**
ANS 1000 /w RS	0.9952	1.0725	**2.24**	1642

We see a much better result using equiprobable initial neighborhoods versus random initial neighborhoods. However, this was anticipated as we would expect the solver to take a few score updates of the probability to stabilize and find the best combination of scores and probabilities. Please take note, we still see an increase in the in the objective function over the base method outlined by Goodfellow & Dimitrakopoulos (2016) when using the random start for *ANS 100 /w RS* and an increase for the maximum for both random start methods. Therefore, even though the quality of random start solutions is not as good as the equiprobable start, the power of the ANS is able to work through many negative starts.

5 Conclusion

This thesis aimed to implement an adaptive neighborhood search based on the work of Lamghari & Dimitrakopoulos (2015) into a simulated annealing optimization method developed and implemented by Goodfellow & Dimitrakopoulos (2016). The implementation was successful in that we found better objective function values often irrespective of the value of the number of iterations between each score update – $(ScoreUpdate)_{skip}$. That is, very frequently we found an improved solution over the static parameters and method outlined by Goodfellow & Dimitrakopoulos (2016). It was found that the higher values of the $(ScoreUpdate)_{skip}$ produced the best results in the tests (when $(ScoreUpdate)_{skip}$ was 333 or 1000). This indicates that the method prefers to have significantly more information before updating the set of probabilities. We also illustrated that the use of differential evolution in the optimization of the downstream variables (in addition to SA optimizing all the variables) yields a better result over both using the static parameters with differential evolution and using simulated annealing with adaptive neighborhood search without differential evolution. To test the robustness of the solver, there was the exploration of initializing the test with random neighborhood probabilities, that is, an unequally probable start. Although, in general the objective function value did not achieve the same value result versus the adaptive neighborhood search with an equiprobable start, the adaptive neighborhood search with random initial neighborhood probabilities outperforms the static methods.

Future reach should look deeper into adapting the stopping criteria in the simulated annealing method as a function of the size of the problem, rather than manually adjusting the criteria for the given problem. In addition, implementing, even a rudimentary heuristic, to establish an initial mining schedule could prove a valuable addition to the solver.

6 Bibliography

Albach, H. (1967). Long Range Planning in Open-Pit Mining. *Management Science, 13*(10), B-549-B-568.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1027-1035). New Orleans: SIAM.

Benndorf, J., & Dimitrakopoulos, R. (2004). New efficient methods for conditional simulation of large orebodies. *Orebody Modelling and Strategic Mine Planning -- Uncertainty and Risk Management International Symposium 2004* (pp. 103-109). Perth: The Australasian Institute of Mining and Metallurgy.

Bienstock, D., & Zuckerberg, M. (2010). Solving LP Relaxations of large scale precedence constrained problems. In F. Eisenbrand, & F. B. Shepherd, *Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne* (pp. 1-14). Berlin: Springer Berlin Heidelberg.

Blechynden, D., Gardener, N., & Mossop, J. (2012). Surface Mining. *MIME 419 - Surface Mining.* Montreal: Department of Mining And Materials Enginnering, McGill University.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operations Research Society, 64*(12), 1695-1724.

Buro, Y. A. (2013). Mining Geology. *Course Notes from MIME 524 - Mining Geology.* Montreal: Department of Mining and Materials Engineering, McGill University.

Busnach, E., Mehrez, A., & Sinuany-Stern. (1985). A production problem in phosphate mining. *Journal of the Operations Research Society, 36*(4), 285-288.

Cerny, V. (1985). Thermodynamical Approach to the Traveling Salesman Problem : an Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications, 45*(1985), 41-51.

Consuegra, F. R., & Dimitrakopoulos, R. (2009). Stochastic mine design optimization based on simulated annealing: pit limits, production schedules, multiple orebody scenarios and sensitivity analysis. *Mining Technology, 118*(2), 79-90.

Dimitrakopoulos, R. (2015). *Ore Reserve Risk and Mine Planning Optimization: Stochastic Models and Optimization with Applications.* Montreal: McGill COSMO Stochastic Mine Planning Laboratory.

Dimitrakopoulos, R., Farrelly, C. T., & Godoy, M. (2002). Moving forward from traditional optimization : grade uncertainty and risk effects in open-pit design. *Mining Technology*(111), A82-A88.

Drake, J., Ozcan, E., & Burke, E. (2012). An improved choice function heuristic selection for cross domain heuristic search. *Lecture Notes in Computer Science 7492*, 307–316.

Énergie et Ressources Naturelles Québec. (2016, April 18). *Statistiques Minières*. Retrieved from Énergie et Ressources Naturelles Québec: https://www.mern.gouv.qc.ca/mines/statistiques/index.jsp

Gendreau, M., & Potvin, J.-Y. (Eds.). (2010). *Handbook of Metaheuristics* (Vol. 146). New York: Pisinger.

Gentry, D. (1988). Minerals project evaluation - an overview. *Conference on Applied Rock Engineering* (pp. A25-A35). Tyne, England: University of Newcastle.

Gershon, M. E. (1983). Optimal mine production scheduling: evaluation of large scale mathematical programming approaches. *International Journal of Mining Engineering, 1*(4), 315-329.

Gholamnejad, J., & Osanloo, M. (2007). Using chance constrained binary integer programming in optimizing long term production scheduling for open pit mine design. *Transactions of the Institution of Mining and Metallurgy, Section A: Mining Technology, 116*(2), 58-66.

Gitman, L. J., & Joehnk, M. D. (1999). *Fundamentals of Investing* (7th ed.). Reading, MA: Addison-Wesley.

Godoy, M. (2003). *The effective management of geological risk in long-term production scheduling of open pit mines.* Unpublished Thesis: The University of Queensland.

Godoy, M., & Dimitrakopoulos, R. (2004). New Efficient Methods for Conditional Simulation of Large Orebodies. *Transactions, 316*, 43-50.

Goodfellow, R. C., & Dimitrakopoulos, R. (2016, March). Global Optimization of Open Pit Mining Complexes with Uncertainty. *Applied Soft Computing*(40), 292-304.

Goodfellow, R., Consuegra, F., Dimitrakopoulos, R., & Lloyd, T. (2012). Quantifying multi-element and volumetric uncertainty, Coleman McCreedy deposit, Ontario, Canada. *Computers & Geosciences, 42*, 71–78.

Government of Canada. (2016, April 18). *Mining, Quarrying, and Oil and Gas Extraction (NAICS 21): Definition - Canadian Industry Statistics - Industries and Business - Industry Canada*. Retrieved from Innovation, Science and Economic Development Canada: https://www.ic.gc.ca/app/scr/sbms/sbb/cis/definition.html?code=21

Investopedia. (2016, April 18). *Investopedia Dictionary*. Retrieved from Investopedia: http://www.investopedia.com/dictionary/

Jewbali, A. (2006). *Modelling geological uncertanity for stochastic short term production scheduling in open pit metal mines.* University of Queensland. Brisbane: PhD Thesis.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *IEEE International Conference on Neural Networks* (pp. 1942–1948). IEEE Press.

Khan, A., & Niemann-Delius, C. (2014). Production Scheduling of Open Pit Mines Using Particle Swarm Optimization Algorithm. *Advances in Operations Research, 2014*, 1-9.

Kirkpatrick, S., Gelatt Jr, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science, 220*(1983), 671-680.

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Chemecal, Metallurgy, and Mining Society of South Africa , 52*(6), 119–139.

Lamghari, A., & Dimitrakopoulos, R. (2012). A diversified Tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. *European Journal of Operational Research, 222*, 642-652.

Lamghari, A., & Dimitrakopoulos, R. (2015). Hyper-heuristic approaches for solving stochastic optimization formulations of mineral value chains. *Elsevier (Pre-print submitted)*, 40.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., & . (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics, 21*(6), 1087-1092.

Natural Resources Canada. (2016, April 18). *Preliminary estimate of the mineral production of Canada, by province, 2015*. Retrieved from Natural Resources Canada: http://sead.nrcan.gc.ca/prod-prod/ann-data-en.aspx?FileT=2015&Lang=en

Newman, A. M., Rubio, E., Caro, R., Wienraub, A., Eurek, K., & . (2010). A Review of Operations Research in Mine Planning. *Interfaces, 40*(3), 222-245.

Nikolaev, A. G., & Jacobson, S. H. (2010). Simulated Annealing. In M. Gendreau, & J.-Y. Potvin, *Handbook of Metaheurstics, 2nd Edition* (pp. 1-40). New York: Springer.

Osterholt, V. (2005). A Multistage Stochastic Programming Approach to Open Pit Mine Production Scheduling with Uncertain Geology. *Thesis, The University of Queensland*, 126.

Pisinger, D., & Ropke, S. (2007, August). A general heuristic for vehicle routing problems. *Computers and Operations research, 34*(8), 2403-2435. doi:10.1016/j.cor.2005.09.012

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle Swarm Optimization. *Swarm Intell, 1*(1), 33-57.

Ramazan, S., & Dimitrakopoulos, R. (2013). Production scheduling with uncertain supply: a new solution to the open pit mining problem. *Optimization Engineering, 14*(2), 361-380.

Ravenscroft, P. (1992). Risk Analysis for Mine Scheduling by Conditional Simulations. *Mining Technology*, 104-108.

Statistics Canada. (2016, April 18). *Gross domestic product (GDP) at basic prices, by North American Industry Classification System (NAICS)*. Retrieved from Statistics Canada: http://www5.statcan.gc.ca/cansim/a26

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization, 4*(11), 341-359.

The Northern Miner. (1990). *Mining Explained: A guide to prospecting and mining.* (P. Whiteway, Ed.) Toronto, Ontario: John S. Cooke.

Whittle, G. (2014). Money Mining & Sustainability. *Money Mining & Sustainability.* Toronto: Whittle Consulting.

Appendix A: Updated Pseudocode with Adaptive Search Procedures

Algorithm 6: Initialization and Generation

```
GENERATE
      Build X ← {n_1, n_2, ..., n_i} ∀ i ∈ N # i.e. generate an initial solution
      N is a set of neighborhoods within a solution that can be perturbed
      z = f(X) # We will maximize z

INITIALIZE
      SET X*, X^GBS ← X # Where X is the current solution, X* is a local best
      solution, X' is a perturbed solution, and X^GBS is a global best
      solution
      SET α ← 0.7; β ← 0.5; # Linear combination factors[g]
      NewBest ← FALSE  # tracker for updated solution
      GET (iter)_max, (iter)_skip, USE_DE

EXECUTE ANS_SGOPM(X, X*, X^GBS, α, β, (iter)_max, (ScoreUpdate)_skip, USE_DE)
```

Algorithm 7: Simulated Annealing with Adaptive Neighborhood Search to solve the two-stage stochastic open pit mining problem

```
FUNCTION ANS_SGOPM(X, X*, X^GBS, α, β, (iter)_max, (iter)_skip, USE_DE)
      WHILE f(X^GBS) ≤ f(X*) # Executing a global iteration
            X^GBS ← X*
            EXECUTE SAIteration()
            IF USE_DE THEN EXECUTE DE() # See execution in Annex C.
```

[g] Values taken from Lamghari & Dimitrakopoulos (2015)

Algorithm 8: A singular execution of a simulated annealing metaheuristic

```
FUNCTION SAIteration()
    SET iter ← 0
    EXECUTE SetInitialScores()
    WHILE iter < (iter)_max
        iter ← iter + 1
        IF iter mod (ScoreUpdate)_skip = 0
            EXECUTE ComputeProbabilities()
        EXECUTE SingleIteration()
```

Algorithm 9: Setting the initial scores and probabilities for the search neighborhood

```
FUNCTION SetInitialScores()
    FOR EACH i, ∀ i ∈ N
        SET π1(n_i), π2(n_i), κ(n_i) ← 0
        SET s_i ← 1 # Score of neighborhood i
        SET p_i = (1/|N|) # Probability of neighborhood i[h]
```

Algorithm 10: Computing the probabilities using scores gained from the simulated annealing iterations

```
FUNCTION ComputeProbabilities()
    FOR EACH s_i, ∀ i ∈ N
        IF κ(n_i) > 0
            s_i = (1 − α)s_i + α((β π1(n_i) + (1−β)π2(n_i)) / κ(n_i))
    FOR EACH i, ∀ i ∈ N
        p_i = s_i / (Σ_{j∈N} s_j)
        SET π1(n_i), π2(n_i), κ(n_i) ← 0
    IF NewBest
        SET β ← 1.0
    ELSE
        SET β ← max(β − 0.1, 0.0)
    SET NewBest ← FALSE
```

[h] In "Random Start" the p_i will take a random value where $\sum_{\forall i \in N} p_i = 1$

Algorithm 11: Executing a single iteration of the simulated annealing algorithm

```
FUNCTION SingleIteration()
    EXECUTE GeneratePertubation(N, X, X′) to get neighborhood i and
        perturbed solution X′
    κ(n_i) ← κ(n_i) + 1
    # recall here we are maximizing the objective function
    IF f(X′) ≥ f(X) # accepted outright
        X ← X′
        SET NewBest ← TRUE
        SET π_1(n_i) ← π_1(n_i) + |Δf|
    ELSE IF X′ accepted under other criteria # e.g. temperature
        X ← X′
        SET π_1(n_i) ← π_1(n_i) + 1/|Δ f|
    ELSE # Rejection of solution
        SET π_2(n_i) ← π_2(n_i) + 1/|Δf|
```

Algorithm 12: An algorithm that chooses a neighborhood which to perturb the solution and yield a neighborhood solution.

```
FUNCTION GeneratePertubation(N, X, X′)
    SET r ← U{0,1} # r is a uniform random number
    Using r and p_i ∀i ∈ N choose a neighborhood n_i to perturb the solution
    GENERATE a perturbation P from n_i
    SET X′ ← X ⊕ P
```

Appendix B: Pseudocode from Goodfellow & Dimitrakopoulos (2016)

Algorithm 13: Global optimization of mining complexes

```
Require:
      Φ = {x, y, z}
      usePSO
      useDE
FUNCTION GlobalOptimization(Φ, usePSO, useDE)
      Φ^g ← Φ
      i^gopt ← 0
      WHILE true DO
            i^gopt ← i^gopt + 1
            Φ ← Φ^g
            Φ^g ←SimulatedAnnealing(Φ^g)
            if usePSO = true or useDE = true
                  Φ^g ←DownstreamOptimization(Φ^g)
            if g(Φ^g) = g(Φ) then
                  break
```

Algorithm 14: Simulated Annealing for open pit mining complexes

```
Require:
        ρ, k, n^iter
        prob^seq, prob^dest
        i^gbu
        i^total
        cdf_seq, cdf_dest, cdf_proc
FUNCTION SimulatedAnnealing(Φ^g):
        Φ, Φ' ← Φ^g
        i, i^u ← 0
        δ ← 0
        WHILE true DO
                i ← i + 1
                if i mod n^iter = 0 then
                        ρ ← ρ · k
                Φ', δ ←PerturbSolution(Φ, ρ)
                r ← U[0,1]
                if P(g(Φ), g(Φ'), δ) ≥ r then
                        Φ ← Φ'
                if g(Φ) > g(Φ') then
                        Update cdf_seq, cdf_dest or cdf_proc with |g(Φ) − g(Φ')|
                if g(Φ) ≤ g(Φ')
                        Φ^g ← Φ'
                        i^u ← i^u + 1
                if i = i^total or i^u = i^gbu then
                        break
        RETURN Φ^g
```

Algorithm 15: Solution perturbation

```
FUNCTION PerturbSolution(Φ, ρ)
      r ← U[0,1]
      δ ← 0
      If r < prob^seq
            Randomly select an x_{b,t} from x ∈ Φ
            Find the set of blocks x_b(t′)that must be extracted in t′ to
                  satisfy Eq. (7)
            Φ′ ← [x⊕x_b(t′), z, y]
            δ ← cdf_seq^{-1}(ρ)
      Else if r < (prob^seq + prob^dest) then
            Randomly select z_{c,t} an encoded variable from z ∈ Φ
            z′_{c,t} ← U[0, |O(c)|]
            Φ′ ← [x, z⊕z′_{c,t}, y]
            δ ← cdf_dest^{-1}(ρ)
      Else
            Randomly select a y_{i,j,t,s} from y ∈ Φ
            y′_{i,j,t,s} ← y_{i,j,t,s} + N(y_{i,j,t,s}, 0.1)
            Φ′ ← [x, z, y⊕y′_{i,j,t,s}]
            Normalize y ∈ Φ_a to obey Eq. (13) and Eq. (14)
            δ ← cdf_proc^{-1}(ρ)
Return Φ′, δ
```

Algorithm 16: downstream optimization using PSO or DE

```
REQUIRE:
      NP
      NP^local
      c_1, c_2, c_3
      CR, F
      pcc
      i^total
FUNCTION DownstreamOptimization(Φ^g, usePSO, useDE):
      For all q ∈ {1, … NP} do
            Randomize Φ_q (PSO, DE) and velocity V_q (PSO)
            (x ∈ Φ_q) ← (x ∈ Φ^g)
            Normalize y ∈ Φ_q to obey Eq. (13) and Eq. (14)
            Φ_q^Best ← Φ_q
      Φ_NP^best ← Φ^g
      WHILE true DO
            i ← i + 1
            For all q ∈ {1, … NP} DO
                  IF usePSO DO
```

```
            Get Φ_lbest^best, the member within q ± NP^local with the
                  best objective function
            Φ_q ←PSOUpdate(V_q, Φ_q, Φ_q^best, Φ_lbest^best)
      ELSE IF useDE DO
            Randomly select a, b, and c where a, b, c, ∉ q
            Φ_q ←DECrossover(Φ_q^best, Φ_a^best, Φ_b^best, Φ_c^best)
            Correct z ∈ Φ_q to obey eq. (10)
            Correct y ∈ Φ_q to obey Eq. (13) and (14)
      IF g(Φ_q) ≥ g(Φ_q^best) THEN
            Φ^g ← Φ_q
      avg ← (1/NP) Σ_{q=i}^{NP} g(Φ_q^bets)
      IF i = i^total or (g(Φ_q^best) − avg)/avg < pcc ∀q = {1, … NP} THEN
            Break
RETURN Φ^g
```

Algorithm 17: PSO update for particle q

```
FUNCTION PSOUpdate(V_q, Φ_q, Φ_q^best, Φ_lbest^best)
      Let V_q^z, V_q^y ∈ V_q represent the velocities of the downstream variables
      Let Φ_q^z, Φ_q^y ∈ Φ_q represent the values of the downstream variables
      r_1, r_2 ← U[0,1]
      r_3, r_4 ← U[0,1]
      V_q^z ← c_1 · V_q^z + c_2 · r_1 · (z_q^best − z_q) + c_3 · r_2 · (z_q^best − z_q)
      V_q^y ← c_1 · V_q^y + c_2 · r_3 · (y_q^best − y_q) + c_3 · r_4 · (y_q^best − y_q)
      z_q ← z_q + V_q^z
      y_q ← y_q + V_q^y
      RETURN Φ_q
```

Algorithm 18: DE for agent q

```
FUNCTION DECrossover(Φ_q^best, Φ_a^best, Φ_b^best, Φ_c^best)
      Φ_q ← Φ_q^best
      For all z_{ϵ,t}^q ∈ z_q do
            r ← U[0,1]
            If r ≤ CR Then
                  z_{ϵ,t}^q ← z_{ϵ,t}^{a,best} + F · (z_{ϵ,t}^{b,best} − z_{ϵ,t}^{c,best})
      For all y_{i,j,t,s}^q ∈ y_q do
            r ← U[0,1]
            If r ≤ CR then
```

$$y_{i,j,t,s}^{q} \leftarrow y_{i,j,t,s}^{a,best} + F \cdot \left(y_{i,j,t,s}^{b,best} - y_{i,j,t,s}^{c,best}\right)$$

RETURN Φ_q

Appendix C: Hyper-heuristic from Lamghari & Dimitrakopoulos (2015)

Algorithm 19: Hyper-heuristic outlined Lamghari & Dimitrakopoulos (2015)

```
INITIALIZE
      Generate initial solution X
      SET X* ← X
      SET α ← 0.7, β ← 0.5
STAGE I: Generate initial scores
      Add all heuristics to a list H
      WHILE length(H) > 0
            Choose a heuristic h_j, ∀j ∈ H at random
            GENERATE a new solution X' from X using h_j
            IF X' is better than X* then
                  SET X* ← X'
                  SET newIncumbent ← TRUE
            END IF
            Calculate score of h_j
            SET X ← X'
            REMOVE h_j from H
      END WHILE
Stage II: Selecting heuristics based on their score and tabu status
      iter ← 1
      FOR each heuristic h_j do
            SET π1(h_j) ← 0, π1(h_j) ← 0, and η(h_j) ← 0
      END FOR
      WHILE stopping criterion not met DO
      IF all heuristics are tabu THEN
            Revoke the tabu status of all heuristics
      END IF
      Choose, among the heuristics that are not tabu, a heuristic h_j
            using roulette-wheel selection based on scores
      Set η(h_j) ← η(h_j) + 1
      Generate a new solution X' from X using h_j
      IF X' is better than X* then
            SET X* ← X'
            SET newIncumbent ← TRUE
      END IF
      IF X' is better than X then
            Update π1(h_j)
      ELSE
            Update π2(h_j)
            Generate a random number ω in [Ω_min ; Ω_max ]
```

```
            Make h_j tabu for ω iterations
      END IF
      IF iter < κ then
            SET iter ← iter + 1
      ELSE
            IF newIncumbent = TRUE then
                  Set β ← 1
            ELSE
                  Set β ← max(β − 0.1 , 0)
            END IF
            Update the score of all heuristics using
            Revoke the tabu status of all heuristics
            FOR each heuristic h_j do
                  SET π_1(h_j) ← 0, π_1(h_j) ← 0, and η(h_j) ← 0
            END FOR
            Set iter ← 1
            Set newIncumbent = FALSE
      END IF
      SET X ← X'
END WHILE
RETURN X*
```