Les Cahiers du GERAD ISSN: 0711–2440

An adaptive large neighborhood search heuristic to optimize mineral supply chain operations under metal and material type uncertainties

A. Lamghari
R. Dimitrakopoulos

G–2015–93

September 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

An adaptive large neighborhood search heuristic to optimize mineral supply chain operations under metal and material type uncertainties

Amina Lamghari

Roussos Dimitrakopoulos

GERAD & COSMO – Stochastic Mine Planning Laboratory, Department of Mining and Materials Engineering, McGill University, Montreal (Quebec) Canada, H3A 2A7

amina.lamghari@mcgill.ca
roussos.dimitrakopoulos@mcgill.ca

September 2015

Les Cahiers du GERAD

G–2015–93

Abstract: This paper addresses the optimization of mineral supply chain operations under metal and material type uncertainties. A mathematical model to simultaneously optimize the mining decisions, the destination decisions, and consequently the cut-off grade, is proposed. A fix-and-optimize scheme that exploits the structure of the problem and uses relaxation and decomposition techniques is introduced to obtain an initial solution, and an adaptive large neighborhood search heuristic is developed to improve this solution. The proposed solution approach is tested on a copper-gold deposit. The results of these experiments show the ability of the proposed solution approach to efficiently address large instances of realistic size and provide schedules where the most valuable material is mined and processed early in the life of the mine and where the risk of not meeting production targets is reduced.

Key Words: Scheduling, heuristics, adaptive large neighborhood search, open-pit mining, metal uncertainty, material type uncertainty, multiple processors.

1 Introduction

Scheduling operations of open-pit mines is an important and critical issue in surface mine planning as it determines the raw materials to be produced yearly over the life of the mine and has a definitive impact on the economic value of the mine. The intrinsic uncertainty of the corresponding optimization problem, its combinatorial nature, and its large scale make it challenging to solve, and thus it has attracted significant research in recent years. Typically, an open-pit mine is represented as a three-dimensional array of blocks, each of which represents a volume of material that can be extracted from the ground and possibly processed to produce final sellable products. To convert blocks into final sellable products by separating valuable minerals from rock, a sequence of operations must be performed. Each operation is treated as a separate stage in the sequence, and material passes through each stage in succession. Moreover, at each stage, there are different facilities of different capacities that might vary from period to period. The first stage facilities are fed with blocks that come from the mine, while the facilities associated with the other stages are fed from the facilities of the previous stages. The last stage facilities provide the final products.

Not all blocks are available at the beginning of the planning horizon. They become available as they are extracted. An available block cannot be sent to just any first stage facility. It can only go to a predetermined subset of facilities depending on the block's material type. In what follows, we will say that a block is admissible for a facility if it can be sent to this facility. When and how much material will be sent from one facility to the next depends on the type of facility. For each facility, there are three possible scenarios for how material can pass through the stages in the processing chain by which the facilities can be categorized.

1. Category 1: In any period, all the material added to the facility is reclaimed from it in the same period; inventory is not carried over to the next period (concentrators, for instance).
2. Category 2: In any period, material can be added to the facility but is never reclaimed from it (dump sites, for instance).
3. Category 3: In any period, material can be added to a facility as well as reclaimed from it. Inventory is carried from the previous periods, and any amount less than or equal to the amount available at the facility can be sent to a facility in the next stage (stockpiles, for instance).

Finally, a facility of a given stage can send material to any of a pre-determined set of facilities of the next stage. Such is the case except for facilities of Category 2 (dump sites), which do not send material anywhere. Sending blocks to the first stage facilities and from one facility to another incurs transportation and/or processing costs. The refined mineral processed through the final stage is sold and generates revenues. The problem is to decide which blocks to mine and when to mine them, where to send each mined block, and how much material to send from each facility to another at each period, so as to maximize the total discounted profit, while meeting the capacities and the specifications at each facility. Decisions are also shaped by the goal of meeting the physical and technical requirements for extracting the blocks. These requirements, referred to as the slope constraints or the precedence constraints, prevent blocks from being mined before others on top of them (their predecessors) to prevent the walls of the pit from collapsing.

The metal content of the blocks and their material type are not known with certainty at the time decisions are made but are interpolated using information obtained from exploration drilling. Over the last decades, different studies have consistently shown the importance and benefits of integrating this uncertainty in the optimization process when devising mine long-term production schedules (Ravenscroft (1992); Dowd (1994, 1997); Dimitrakopoulos et al. (2002); Godoy and Dimitrakopoulos (2004); Menabde et al. (2005); Whittle and Bozorgebrahimi (2005); Kent et al. (2007); Boland et al. (2008); Albor and Dimitrakopoulos (2010); Ramazan and Dimitrakopoulos (2013); Goodfellow and Dimitrakopoulos (2013)). The authors have shown that the stochastic approach can provide major improvements in NPV, in the order of 10% to 30%, compared to the solution obtained by solving a deterministic problem based on expected values. They also demonstrated that the stochastic approach substantially reduces risk in meeting production forecasts and finds larger pit limits, contributing to the sustainable utilization of mineral resources.

The problem addressed in this paper, which we will denote as SMSC, accounts for metal and material type uncertainties. It considers a particular case of the mineral supply chain described above that has the

two following specificities: First, the first stage facilities are the last stage facilities; that is, the mine feeds a set of processing facilities and these facilities produce the final products to be sold (the refined minerals). Second, all the facilities are of category 1 or 2; that is, the facilities to which the blocks are sent, referred to as destinations in the rest of the paper, do not include stockpiles. SMSC is similar to the problem studied in Goodfellow and Dimitrakopoulos (2013), except that the authors consider dynamic recoveries in the different processing facilities that depend on the average grade of the incoming material at these facilities. Instead, in this paper, the recoveries are considered to be fixed. There are also some similarities between SMSC and some stochastic variants of the open-pit mine production scheduling problem (MPSP) studied in the literature. The main differences between SMSC and the variant of the MPSP addressed in Lamghari and Dimitrakopoulos (2013) is that, in addition to the metal content, the material type of the blocks is also considered uncertain in this paper. Moreover, while in Lamghari and Dimitrakopoulos (2013) a block is processed in the facility that gives the highest economic value; that is, the destination decisions are fixed a priori, in this paper they are not. In this sense, SMSC can be seen as a generalization of the problem addressed in Lamghari and Dimitrakopoulos (2013) through the consideration of material type uncertainty and destination decisions. SMSC can also be seen as a generalization of the problem addressed in Goodfellow and Dimitrakopoulos (2012). Indeed, SMSC reduces the problem studied in Goodfellow and Dimitrakopoulos (2012) by omitting the extraction component. Clearly, SMSC is more complex than the two problems addressed in Goodfellow and Dimitrakopoulos (2012) and Lamghari and Dimitrakopoulos (2013) since considering the mining and destination decisions simultaneously greatly increases the size of the stochastic problem to be solved and represents substantial challenges from a computational point of view.

The main contributions of this paper are threefold. First, a two-stage stochastic formulation is provided, which is an extension of the formulation in Lamghari and Dimitrakopoulos (2012). This formulation is different from the one proposed in Goodfellow and Dimitrakopoulos (2013). Indeed, in Goodfellow and Dimitrakopoulos (2013) the destination decisions, referred to as destination policies, are considered first-stage decisions; that is, they are taken before the uncertainty is revealed and thus are independent of the random data, whereas they are considered second-stage decisions in this paper. In other words, it is assumed that they are taken after the uncertainty is revealed. Second, a solution approach based on the adaptive large neighborhood search framework (ALNS) is developed. The structure of the problem is exploited and relaxation and decomposition techniques are used to obtain the initial solution to be improved with ALNS. The proposed solution approach is also different from the one proposed by Goodfellow and Dimitrakopoulos (2013), which used Gemcom Whittle (Whittle, 2009) to generate an initial solution and a hybrid of simulated annealing and particle swarm methods to improve it. The proposed approach is tested on a copper-gold mining complex with six material types and six destinations.

The remainder of the paper is organized as follows: In Section 2, the approach used to deal with metal and material type uncertainties is outlined, and a mathematical formulation of the SMSC studied in this paper is introduced. The following sections present the heuristics used in the initialization phase and the improvement phase, respectively. Computational results are reported and discussed in Section 5. Finally, conclusions are drawn in Section 6.

2 Mathematical formulation

This section describes the mathematical model used to define the problem studied in this paper. As mentioned previously, the uncertain parameters are related to the metal content and material type of the blocks (the supply). The material type has an impact on the tonnage of the blocks, the mining costs, and the admissibility of the blocks to the different processing destinations (W, C and A, respectively). The metal content has an impact on the economic value generated after processing the extracted blocks (V). The joint distribution of the stochastic parameters is assumed to be known. Let $\xi = (W, C, A, V)$ be the random data vector and let $\xi(s)$ denote one particular realization of ξ (a scenario). It is assumed that there is a finite number of scenarios and that the scenarios are equiprobable. The objective is to maximize the expected revenue from the refined minerals sold minus the mining, processing, transportation and selling costs. The problem can be formulated as a two-stage stochastic program with recourse (Birge and Louveaux, 2011). The following notation is used in the formulation:

Sets:

$\mathcal{N}$: Set of blocks considered for scheduling, $\mathcal{N} = \{1, \ldots, N\}$.

$\mathcal{P}(i)$: Set of immediate predecessors of block i; i.e., blocks that directly precede block i and have to be extracted to have access to i.

$\mathcal{T}$: Set of time periods over which blocks are being scheduled, $\mathcal{T} = \{1, \ldots, T\}$.

$\mathcal{D}$: Set of possible destinations for the blocks once extracted, $\mathcal{D} = \{1, \ldots, D\}$.

$\mathcal{S}$: Set of scenarios, $\mathcal{S} = \{1, \ldots, S\}$.

Indices and superscripts:

i, j: Block index, $i, j \in \mathcal{N}$.

t, τ: Period index, $t, \tau \in \mathcal{T}$.

d: Destination index, $d \in \mathcal{D}$.

s: Scenario index, $s \in \mathcal{S}$.

Parameters:

w_{is}: Tonnage of block i in scenario s.

a_{ids}: 1 if block i is admissible for destination d in scenario s (i.e., if it can be processed in this destination given its material type in scenario s), 0 otherwise.

$\underline{E^t}$: Minimum amount that should be extracted in period t (lower bound on mining).

$\overline{E^t}$: Maximum amount that should be extracted in period t (upper bound on mining or mining capacity).

$\underline{F_d^t}$: Minimum amount of material (flow) that should be sent to destination d in period t (demand at d).

$\overline{F_d^t}$: Maximum amount of material (flow) that should be sent to destination d in period t (processing capacity at d).

c_{is}: Cost of mining block i in scenario s.

$E[c_i]$: Expected cost of mining block i. Recall that the scenarios are equiprobable. Hence, $E[c_i] = \frac{1}{S} \sum_{s \in \mathcal{S}} c_{is}$.

v_{ids}: Economic value to be generated if block i is processed at destination d in scenario s. This value is calculated as the return from selling the recovered metal minus the processing, the transportation, and any related costs.

p^-: Per-unit cost incurred for failing to meet the lower bound on mining.

p^+: Per-unit cost incurred for not satisfying the mining capacity.

$q_d{}^-$: Per-unit cost incurred for failing to meet the demand at destination d.

$q_d{}^+$: Per-unit cost incurred for exceeding the capacity of destination d.

δ_1: The discount rate.

δ_2: The geological rate discount.

Variables:

x_i^t: 1 if block i is extracted in period t, 0 otherwise.

e_s^{t-}: Amount in shortage of mined material in scenario s and period t.

e_s^{t+}: Amount of extra material mined in scenario s and period t; i.e., amount above the upper bound $\overline{E^t}$.

y_{ids}^t: 1 if block i is sent to destination d in scenario s and period t, 0 otherwise.

f_{ds}^{t-}: Amount in shortage at destination d in scenario s and period t; i.e., amount of unsatisfied demand.

f_{ds}^{t+}: Amount of extra material sent to destination d in scenario s and period t; i.e., amount above the capacity $\overline{F_d^t}$.

The stochastic SMSC addressed in this paper can be formulated as follows:

$$\max f(x) = -\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} \frac{E[c_i]}{(1+\delta_1)^t} x_i^t + E[Q(x,\xi)] \tag{1}$$

$$\text{s.t.} \sum_{t\in\mathcal{T}} x_i^t \leq 1 \qquad \forall\, i\in\mathcal{N} \tag{2}$$

$$x_i^t - \sum_{\tau=1}^{t} x_i^\tau \leq 0 \qquad \forall\, i\in\mathcal{N}, j\in\mathcal{P}(i), t\in\mathcal{T} \tag{3}$$

$$x_i^t \in \{0,1\} \qquad \forall\, i\in\mathcal{N}, t\in\mathcal{T} \tag{4}$$

where $E[Q(x,\xi)] = \frac{1}{S}\sum_{s\in\mathcal{S}} Q(x,\xi(s))$, and $Q(x,\xi(s))$ is the optimal value of the following problem (second-stage problem):

$$\max \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}}\sum_{d\in\mathcal{D}} \frac{v_{ids}}{(1+\delta_1)^t} y_{ids}^t \tag{5a}$$

$$-\sum_{t\in\mathcal{T}} \frac{p^-}{(1+\delta_2)^t} e_s^{t-} \tag{5b}$$

$$-\sum_{t\in\mathcal{T}} \frac{p^+}{(1+\delta_2)^t} e_s^{t+} \tag{5c}$$

$$-\sum_{t\in\mathcal{T}}\sum_{d\in\mathcal{D}} \frac{q_d^-}{(1+\delta_2)^t} f_{ds}^{t-} \tag{5d}$$

$$-\sum_{t\in\mathcal{T}}\sum_{d\in\mathcal{D}} \frac{q_d^+}{(1+\delta_2)^t} f_{ds}^{t+} \tag{5e}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} w_{is} x_i^t + e_s^{t-} \geq \underline{E^t} \qquad \forall\, t\in\mathcal{T} \tag{6}$$

$$\sum_{i\in\mathcal{N}} w_{is} x_i^t - e_s^{t+} \leq \overline{E^t} \qquad \forall\, t\in\mathcal{T} \tag{7}$$

$$x_i^t = \sum_{d\in\mathcal{D}} y_{ids}^t \qquad \forall\, i\in\mathcal{N}, t\in\mathcal{T} \tag{8}$$

$$\sum_{d\in\mathcal{D}} (1-a_{ids})\, y_{ids}^t = 0 \qquad \forall\, i\in\mathcal{N}, t\in\mathcal{T} \tag{9}$$

$$\sum_{i\in\mathcal{N}} w_{is} y_{ids}^t + f_{ds}^{t-} \geq \underline{F_d^t} \qquad \forall\, d\in\mathcal{D}, t\in\mathcal{T} \tag{10}$$

$$\sum_{i\in\mathcal{N}} w_{is} y_{ids}^t - f_{ds}^{t+} \leq \overline{F_d^t} \qquad \forall\, d\in\mathcal{D}, t\in\mathcal{T} \tag{11}$$

$$y_{ids}^t \in \{0,1\} \qquad \forall\, i\in\mathcal{N}, d\in\mathcal{D}, t\in\mathcal{T} \tag{12}$$

$$e_s^{t-}, e_s^{t+}, f_{ds}^{t-}, f_{ds}^{t+} \geq 0 \qquad \forall\, d\in\mathcal{D}, t\in\mathcal{T} \tag{13}$$

The first-stage decisions are the mining (extraction) decisions. They are major long-term decisions and are independent of the random data. Destination decisions are made when the uncertainty is revealed (once the blocks are extracted) and are second-stage decisions. The optimal value $Q(x,\xi(s))$ of the second-stage problem (5)–(13) is function of the first-stage decision variable x (the mining sequence) and a realization $\xi(s)$ of the uncertain parameters (the random data vector $\xi = (W, C, A, V)$).

The objective function (1) minimizes the expected first-stage costs and maximizes the expected second-stage profits. The expected first-stage costs are the expected discounted mining costs. The objective function (5) of the second-stage problem consists of five parts that represent the discounted economic value generated from processing the extracted blocks (5a), the penalties for mining less material than the minimum required (5b), the penalties for exceeding the mining capacity (5c), the penalties for the unsatisfied demand at the

different destinations (5d), and the penalties for lost material occurred because of the insufficient capacity at the different destinations (5e). Note that the parameter δ_2 used to calculate the penalty costs is the so-called geological risk discount introduced by Dimitrakopoulos and Ramazan (2004) to define the importance given to satisfying the production targets over time; i.e., to encourage deferring the shortage/surplus to the last periods of the life of the mine in order to reduce the risk of not meeting the production targets in the first periods. Note also that, since a finite number of equiprobable scenarios are used to model the uncertainty, the objective function (1) can be expressed in an extended form as follows:

$$\begin{aligned}\max f(x) = &- \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} \frac{E[c_i]}{(1+\delta_1)^t} x_i^t \\ &+ \frac{1}{S}\sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}}\left\{ \sum_{i\in\mathcal{N}}\sum_{d\in\mathcal{D}} \frac{v_{ids}}{(1+\delta_1)^t} y_{ids}^t - \frac{p^-}{(1+\delta_2)^t} e_s^{t-} \right. \\ &\left. - \frac{p^+}{(1+\delta_2)^t} e_s^{t+} - \sum_{d\in\mathcal{D}} \frac{q_d^-}{(1+\delta_2)^t} f_{ds}^{t-} - \sum_{d\in\mathcal{D}} \frac{q_d^{t+}}{(1+\delta_2)^t} f_{ds}^{t+} \right\}. \quad (14)\end{aligned}$$

Constraints (2) and (3) define the feasible set of the first-stage problem (reserve and slope constraints, respectively). They ensure that a block is mined at most once (constraints (2)) and prevent a block from being mined before its predecessors (constraints (3)). Constraints (6) and (7) are related to the minimal and maximal mining levels at each period (mining constraints). Shortage, as well as surplus, is allowed, but incurs penalty costs. Constraints (8) and (9) link the mining decisions to the destination decisions. More specifically, they allow a block to be sent to one and only one destination if the block is mined (Constraints (8)) and if it is admissible to this destination (Constraints (9)). Constraints (10) and (11) are related to the amount processed in each destination at each period (processing constraints). If a shortfall occurs in a given destination at a given period, a penalty cost is applied. Similarly, if there is a surplus, a penalty cost is applied.

The problem contains $NT + NTDS$ binary variables. Even a small open-pit mine of 10,000 blocks with 4 destinations, to be scheduled over 5 years accounting for 10 scenarios represents more than one million binary variables (1,050,000) and millions of constraints, which is a size beyond the reach of exact methods. For this reason, a heuristic approach is proposed in this paper to obtain a good quality solution in a reasonable amount of time. First, an initial solution is generated in the first phase of the solution procedure (initialization phase), then this solution is improved in the second phase (improvement phase) using an adaptive large neighborhood heuristic (ALNS). The methods used in the two phases of the solution procedure are described in the following sections.

3 Initialization phase

The approach used to generate the initial solution takes advantage of the problem structure and uses a fix-and-optimize scheme to reasonably quickly provide a feasible solution to be improved with the ALNS heuristic in the second phase of the solution procedure. The basic idea is to divide the problem into a series of smaller, easier to-solve-sub-problems. Each sub-problem deals with a subset of variables, and when it is solved, these variables are fixed according to the solution obtained, and another sub-problem is considered to optimize another subset of variables. This is done as follows.

Using the general time-decomposition approach described in Lamghari et al. (2014), the periods $t \in \mathcal{T}$ are considered one at time in increasing order. In this paper, unlike in Lamghari et al. (2014), each sub-problem associated with a period t includes two sub-problems: a mining problem and a destination problem that are interrelated through constraints (8). These two problems are solved sequentially, which means that the set of blocks to be mined in period t is first determined (a mining plan is constructed in the first step), then each of those blocks determined in the first step is assigned to one admissible destination in each scenario (a destination plan is designed for each scenario). This strategy not only prevents blocks not mined in period t from being processed in period t, ensuring that constraints (8) are satisfied, but it also allows us to decompose

the problem further to speed up the solution process. Indeed, because the scenarios are independent, the destination problem can be divided into S independent sub-problems, each associated with a scenario. This can be exploited to reduce the time required to solve the destination problem by solving the sub-problems associated with the scenarios in parallel. To summarize, starting with $t = 1$, the mining problem handles the mining variables x_i^t and the deviation variables e_s^{t-} and e_s^{t+}. Afterwards, these variables are fixed according to the solution obtained, and the destination problem is solved to determine the destination variables y_{ids}^t and the deviation variables f_{ds}^{t-} and f_{ds}^{t+}, considering one scenario at time. This process is repeated until all periods are considered.

The approach described above is appealing since the problems to solve at each iteration involve many fewer binary variables than the original problem and thus can be solved efficiently. However, it gives rise to the following question: how can one optimize mining decisions without explicitly modeling the destination decisions? One way to get around this difficulty is to relax the problem to consider a priori destinations for the blocks, as explained in the next section.

3.1 Solving the mining problem (MP^t)

In what follows, t is fixed, and the mining problem is denoted by MP^t. Recall that this problem is solved to determine the set of blocks to be mined in period t, which we will denote by $\mathcal{B}^t$ (initially $\mathcal{B}^t = \emptyset$). If there are no restrictions on the amount of material processed at each destination; that is, if constraints (10) and (11) are dropped, then in an optimal solution, each extracted block will be sent to the most profitable destination. Hence, following this assumption, the variables y_{ids}^t as well as the variables f_{ds}^{t-} and f_{ds}^{t+} can be eliminated from the formulation, and the sub-problem associated with period t reduces to:

$$\max \sum_{i \in \mathcal{R}^t} \frac{E\left[v_i^*\right] - E[c_i]}{(1+\delta_1)^t} x_i^t - \frac{1}{S} \sum_{s \in \mathcal{S}} \frac{p^-}{(1+\delta_2)^t} e_s^{t-} - \frac{1}{S} \sum_{s \in \mathcal{S}} \frac{p^+}{(1+\delta_2)^t} e_s^{t+} \tag{15}$$

$$\text{s.t.} \quad x_i^t - x_j^t \leq 0 \qquad \forall\, i \in \mathcal{R}^t, j \in \mathcal{P}(i) \cap \mathcal{R}^t \tag{16}$$

$$\sum_{i \in \mathcal{R}^t} w_{is} x_i^t + e_s^{t-} \geq \underline{E^t} \qquad \forall\, s \in \mathcal{S} \tag{17}$$

$$\sum_{i \in \mathcal{R}^t} w_{is} x_i^t - e_s^{t+} \leq \overline{E^t} \qquad \forall\, s \in \mathcal{S} \tag{18}$$

$$x_i^t \in \{0,1\} \qquad \forall\, i \in \mathcal{R}^t \tag{19}$$

$$e_s^{t-}, e_s^{t+} \geq 0 \qquad \forall\, s \in \mathcal{S}. \tag{20}$$

where, $\mathcal{R}^t$ is the set of blocks not mined at the beginning of period t ($\mathcal{R}^1 = \mathcal{N}$ and $\mathcal{R}^t = \mathcal{R}^{t-1} \backslash \mathcal{B}^{t-1}$ if $t = 2, \ldots, T$), and $E\left[v_i^*\right] = \frac{1}{S} \sum_{s \in \mathcal{S}} v_{id^*(i,s)s}$ represents the expected economic value to be generated if in each scenario s, block i is processed in the most profitable destination $d^*(i.s) = argmax_{d \in \mathcal{A}(i,s)} v_{ids}$, $\mathcal{A}(i,s)$ being the set of destinations for which i is admissible in scenario s.

The objective function (15) maximizes the expected net present value, assuming that under each scenario each extracted block is processed in the most profitable destination. It also minimizes the expected penalty costs incurred whenever the amount mined in period t does not fall within the specified limits $[\underline{E^t}, \overline{E^t}]$. Constraints (16) ensure that the slope constraints are satisfied, while constraints (17) and (18) are related to the mining levels. The reserve constraints are implicitly satisfied since the set $\mathcal{R}^t$ is updated as one goes along from one period to another ($\mathcal{R}^t = \mathcal{R}^{t-1} \backslash \mathcal{B}^{t-1}$). The same applies to the admissibility constraints because only one admissible destination is accounted for when calculating the $E\left[v_i^*\right]$'s.

Note that the proposed relaxation obtained by dropping constraints (10) and (11) does not provide a tight upper bound on the optimal value of the sub-problem associated with period t, but it allows us to account for the profit to be generated from processing the blocks without explicitly modelling the destination decisions, which considerably reduces the size of the problem to be solved. Indeed, the mixed-integer stochastic problem MP^t ((15)–(20)) involves $|\mathcal{R}^t|$ binary variables and $2S$ continuous variables as opposed to $|\mathcal{R}^t| + |\mathcal{R}^t| DS$

binary variables and $2S + 2DS$ continuous variables if one has to consider the destination decisions as well. This size is relatively small and thus MP^t can be solved using a mixed-integer programming solver.

Because the mining decisions will be modified in the improvement stage (i.e., when applying the ALNS heuristic), MP^t does not need to be solved to optimality. In the numerical results presented in Section 5, an optimality tolerance of 1% is used; that is, the solution process stops if the solver can guarantee that the current best solution is within 1% of the global optimum.

MP^t can also be solved using a heuristic, which will require less computational time. In this paper, a random heuristic, henceforth denoted by RP, is used. RP is an extension of the heuristic proposed in Lamghari and Dimitrakopoulos (2012) to account for the fact that the blocks' tonnages are also uncertain. At each iteration, a block having no predecessors or having all its predecessors already mined is randomly selected. The process continues until the total expected weight of blocks mined at t (i.e., $\sum_{i\in\mathcal{B}^t} E[w_i] = \frac{1}{S}\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{B}^t} w_{is}$) reaches $\frac{\underline{E^t}+\overline{E^t}}{2}$. Note that this heuristic does not account for the first term of the objective function (15). Its main purpose is to quickly identify a set $\mathcal{B}^t$ (a mining plan) that satisfies the reserve and slope constraints. The stopping criterion of the heuristic induces some lookahead features to leave some blocks unmined for the next periods so as to satisfy the lower bounds on mining in these periods. The other reason why we use RP to solve MP^t is to evaluate the ability of ALNS to improve a bad quality solution.

3.2 Solving the destination problem (DP^t)

Once the MP^t is solved using one of the methods described in the previous section, the next step is to determine the destinations in which the blocks determined at the previous step are processed under each scenario. The way these destinations are determined is described in this section.

Recall that t is fixed. If the mining variables x_i^t are fixed according to the solution obtained when solving the MP^t, the original sub-problem associated with period t reduces to the following problem:

$$\max \frac{1}{S}\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{B}^t}\sum_{d\in\mathcal{D}}\frac{v_{ids}}{(1+\delta_1)^t}y_{ids}^t - \frac{1}{S}\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}}\frac{q_d^-}{(1+\delta_2)^t}f_{ds}^{t-} - \frac{1}{S}\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}}\frac{q_d^+}{(1+\delta_2)^t}f_{ds}^{t+} \tag{21}$$

$$\text{s.t.} \quad \sum_{d\in\mathcal{D}} y_{ids}^t = 1 \qquad \forall\ i\in\mathcal{B}^t, s\in\mathcal{S} \tag{22}$$

$$\sum_{d\in\mathcal{D}}(1-a_{ids})y_{ids}^t = 0 \qquad \forall\ i\in\mathcal{B}^t, s\in\mathcal{S} \tag{23}$$

$$\sum_{i\in\mathcal{B}^t} w_{is}y_{ids}^t + f_{ds}^{t-} \geq \underline{F_d^t} \qquad \forall\ d\in\mathcal{D}, s\in\mathcal{S} \tag{24}$$

$$\sum_{i\in\mathcal{B}^t} w_{is}y_{ids}^t - f_{ds}^{t+} \leq \overline{F_d^t} \qquad \forall\ d\in\mathcal{D}, s\in\mathcal{S} \tag{25}$$

$$y_{ids}^t \in \{0,1\} \qquad \forall\ i\in\mathcal{B}^t, d\in\mathcal{D}, s\in\mathcal{S} \tag{26}$$

$$f_{ds}^{t-}, f_{ds}^{t+} \geq 0 \qquad \forall\ d\in\mathcal{D}, s\in\mathcal{S}. \tag{27}$$

The objective function (21) maximizes the total expected economic value to be generated from the blocks in $\mathcal{B}^t$; that is, those determined by solving the MP^t. It also minimizes the expected penalty costs incurred whenever the amounts processed in the different destinations are below the demand $\underline{F_d^t}$ or exceed the capacity, $\overline{F_d^t}$. Constraints (22) and (23) allow a block to be processed at only one destination and only if the block is admissible to this destination. Constraints (24) and (25) guarantee that the total amounts sent to each destination under each scenario are within the limits $[\underline{F_d^t}, \overline{F_d^t}]$; otherwise, penalty costs are incurred.

Since the scenarios are independent, the formulation (21)–(27) is scenario-separable, which means that the destinations can be determined independently for each scenario.

In what follows, we denote by DP_s^t the sub-problem associated with scenario s. The objective function of DP_s^t has the following form:

$$\max \sum_{i\in\mathcal{B}^t}\sum_{d\in\mathcal{D}} \frac{v_{ids}}{(1+\delta_1)^t} y_{ids}^t - \sum_{d\in\mathcal{D}} \frac{q_d^-}{(1+\delta_2)^t} f_{ds}^{t-} - \sum_{d\in\mathcal{D}} \frac{q_d^+}{(1+\delta_2)^t} f_{ds}^{t+}. \tag{28}$$

Three methods are proposed to solve a given scenario sub-problem DP_s^t to determine a destination plan under scenario s:

1. MIP: Since the DP_s^t is of relatively small size, it can be solved using a mixed-integer programming solver.
2. MCFP: This method first solves the linear relaxation of DP_s^t obtained by replacing the integrality constraints (26) by $y_{ids}^t \in [0,1]$, thus allowing fractions of blocks to be sent to destinations. It then modifies the obtained solution to generate a feasible destination plan.
   The linear relaxation of DP_s^t can be solved efficiently as a minimum cost flow problem (MCFP). The MCFP is defined on a directed graph $G = (V, A)$. The vertex set $V = \mathcal{B}^t \cup \mathcal{D} \cup \mathcal{D}' \cup \{L, F\}$ has four types of vertices:
   - $\mathcal{B}^t$, block vertices representing the blocks mined in period t (i.e., those determined by solving the MP^t)
   - $\mathcal{D}$, destination vertices representing the different destinations to which the blocks can be sent once mined
   - $\mathcal{D}'$, dummy destination vertices, copies of the destination vertices used to absorb the amount in excess at each destination
   - L, a dummy supplier vertex used to provide the unsatisfied demands
   - F, a sink vertex.

   The graph contains six types of arcs:
   - Arcs that connect a block vertex $i \in \mathcal{B}^t$ to a destination vertex $d \in \mathcal{D}$. The arc (i, d) is included in the set of arcs A only if i is admissible to d. The cost of this arc is $-\frac{v_{ids}}{(1+\delta_1)^t}$, the lower bound is 0, and the upper bound is w_{is}, the tonnage of block i in scenario s.
   - Arcs that connect the dummy supplier vertex L to a destination vertex $d \in \mathcal{D}$. The flow on any such arc corresponds to the amount in shortage at destination d (the unsatisfied demand). Therefore, these arcs are uncapacited and their per-unit cost flow is equal to $\frac{q_d^-}{(1+\delta_2)^t}$.
   - Arcs that connect a destination vertex $d \in \mathcal{D}$ to its copy $d \in \mathcal{D}'$. The flow on the arcs (d, d') denotes the total tonnage of material sent to destination d. The arcs (d, d') are also uncapacited. They have zero costs and positive lower bounds equal to the demand of destination d in period t, $\underline{F_d^t}$.
   - Arcs that connect the dummy destination vertices $d' \in \mathcal{D}'$ to the sink F. Two arcs connect each vertex d' to F. The lower and upper bounds on the first arc are equal to $\underline{F_d^t}$ and $\overline{F_d^t}$, respectively, and the cost is equal to 0. The flow on the second arc denotes the excess in destination d and thus any such arc is uncapacited and has a cost of $\frac{q_d^+}{(1+\delta_2)^t}$.
   - Finally, there is an arc that connects the dummy supplier vertex L to the sink vertex F. This arc is uncapacited and its cost is set equal to 0.

   Figure 1 illustrates the graph in a situation with one block i and one destination d and where i is admissible to d. The cost and the bounds of each arc are displayed above and below the arc, respectively.
   In an optimal solution of the MCFP described above, some blocks might be sent to more than one destination, thus violating constraints (26). To obtain a feasible destination plan, we use the following heuristic rule. Let z_{id}^* denote the flow on arc (i, d) in the optimal solution of the MCFP. If there exists a destination d such that $z_{id}^* = w_{is}$, then block i is assigned to this destination (i.e., y_{ids}^t is set equal

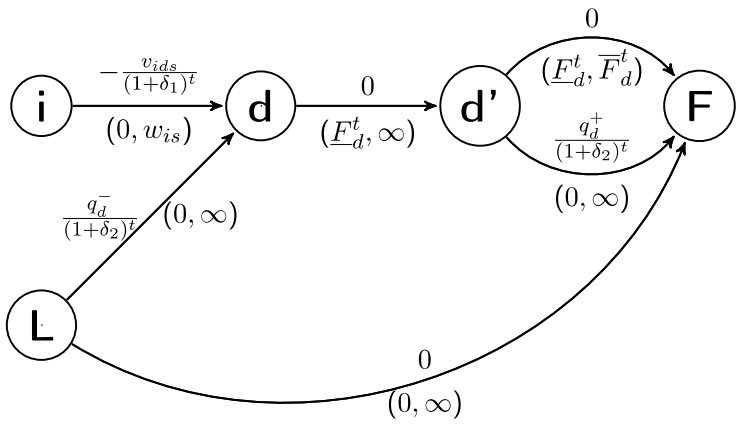

Figure 1: Graph in a situation with one block and one destination.

to 1). Otherwise, i is assigned to destination $\hat{d} = argmax_{d:(i,d)\in A} z_{id}$ (i.e., y^t_{ids} is set equal to 1). In both cases, the way the graph is constructed (arcs (i, d) exist only if i is admissible to d) guarantees that constraints (23) are satisfied and thus the destination plan is feasible.

3. GD: This heuristic generates the destination plan following a greedy approach. The blocks are first ordered in increasing order of the number of destinations for which they are admissible (i.e., $\sum_{d\in\mathcal{D}} a_{ids}$). The blocks are selected in that order starting with the block with the fewest possibilities. For each block, the heuristic selects an admissible destination to maximize the objective function (28) given the blocks already assigned to this destination.
4. RD: Unlike the methods described above (MIP, MCFP, and GD), this method does not consider the objective function value when assigning blocks to destinations. It is an iterative process used to quickly obtain a feasible destination plan. This is done as follows. At each iteration, a block i is randomly selected. Among all the destinations to which i is admissible, one is selected randomly, and i is assigned to this destination. The process continues until all blocks in $\mathcal{B}^t$ are considered.

4 Improvement phase

This phase applies an adaptive large neighborhood search heuristic (ALNS) to improve the initial solution generated in the previous phase. Proposed by Ropke and Pisinger (2006), ALNS extends the large neighborhood search heuristic (LNS) introduced by Shaw (1997). In what follows, the main features of LNS and ALNS are first described, then a step-by-step description of the proposed adaptation of ALNS is provided.

4.1 ALNS framework

4.1.1 General outline

Because ALNS is an extension of LNS, a brief description of LNS is first provided. Starting with an initial solution, LNS progressively improves it by using two methods: a destroy method and a repair method. More specifically, at each iteration, the destroy method removes a certain number of variables from the current solution x, returning an infeasible solution x^-, and the repair method rebuilds the partial solution x^- and returns a feasible solution, x'. This feasible solution x' is either accepted as a new current solution or rejected according to some pre-specified rule. This can be, for example, the Metropolis acceptance rule of the simulated annealing method (Metropolis et al., 1953), by which if x' is better than x, then x' is accepted as the new current solution (i.e., the search resumes from x'); otherwise, x' is accepted with some probability. LNS alternates between destroying and repairing the current solution until some stopping criterion is met. LNS can be seen as a neighborhood search method where the neighborhood of the current solution is defined by the destroy and repair methods. Clearly, the larger the degree of destruction is (i.e., the destroy method

removes a large number of variables), the larger the neighborhood is (a large number of variables are modified at each iteration). To explore the large neighborhood defined by the destroy and repair methods, LNS does not generate it entirely. It rather samples it (Pisinger and Ropke, 2010).

ALNS is based on similar ideas as LNS, except that its uses multiple destroy and repair methods within the search rather than using the same destroy/repair method at each iteration. More precisely, a set of destroy and repair methods is considered. At a given iteration, the destroy/repair method to be used is selected according to an adaptive probabilistic mechanism, which can be seen as a learning mechanism. ALNS associates with each destroy/repair method a weight, also referred to as a score. Each time a method is used, its performance is recorded. The weights are updated periodically based on these recorded performances. The weights thus measure how well each method has performed recently, the highest weights indicating methods that have recently been found successful for the instance being solved. A roulette-wheel mechanism is then used to bias selection towards these methods. ALNS has been shown to be very efficient for a large variety of vehicle routing, scheduling, and production problems; see for instance, Pisinger and Ropke (2007); Cordeau et al. (2010); Demir et al. (2012); Kovacs et al. (2012); Muller et al. (2012); Ribeiro and Laporte (2012); Masson et al. (2013); Adulyasak et al. (2014).

4.1.2 Components of the proposed adaptation of ALNS

The main components of our adaptation of ALNS are described below. The description follows the framework proposed by Ropke and Pisinger (2006) and Pisinger and Ropke (2007) outlined in the previous section and summarized in Algorithm 1.

1. *Large neighborhood*: Let x be the current solution. At each iteration, β blocks are removed from x and are rescheduled in different periods and/or sent to different destinations. Selecting the blocks to remove is done using one of the destroy methods described in Section 4.2. Selecting the new periods and/or destinations for these blocks is done using one of the repair methods described in Section 4.3. The number of blocks β involved in the modifications applied to the current solution is a parameter of the ALNS heuristic, and this parameter has a significant impact on the efficiency of ALNS (Pisinger and Ropke, 2010). Clearly, a small value of β might not allow a thorough exploration of the search space, as the effect of a large neighborhood is lost. On the other hand, a large value of β reduces ALNS to independent reoptimization, in addition to being time-consuming. In this paper, the value of β varies during the search in the interval $[\beta_{min}, \beta_{max}]$, and it is increased or decreased according to the quality of the solutions recently obtained. More specifically, β is initially set to β_{min}. Every 20 iterations, it is updated as follows:
$$\beta = \min\left(\max\left(2^{[(\kappa/10)-1]}\beta, \beta_{min}\right), \beta_{max}\right)$$
where, κ is the number of improving solutions found during the last 20 iterations. Thus, if all 20 previous solutions are non-improving, β is multiplied by 2; if they are all improving, β is divided by 2; intermediate cases lead to smaller changes in the value of β; and in all cases, at least β_{min} blocks but no more that β_{max} blocks are removed from the current solution. By doing so, when improving solutions are found, fewer blocks are removed from the current solution (compared to the previous iteration), and thus few changes are made to the solution to intensify the search in the region of these improving solutions. When ALNS fails to improve the solution, larger changes are made to leave the current region and diversify the search. The values of the parameters β_{min} and β_{max} used in the numerical experiments are given in Section 5.
2. *Adaptive search engine*: As in Ropke and Pisinger (2006) and many other implementations of ALNS, the selection of the destroy and repair methods to be applied at a given iteration is controlled by a roulette-wheel mechanism. The two methods are selected independently. Let ρ_d^- be the weight of destroy method d. d is selected with a probability $\frac{\rho_d^-}{\sum_{m\in\Omega^-}\rho_m^-}$. The probabilities for selecting the repair methods are calculated in a similar manner.
3. *Adaptive weight adjustment*: Without loss of generality, consider the case of the destroy methods $d \in \Omega^-$ (adjusting the weights associated with the repair methods is done in a similar manner). The values of

the weights are initially set to 1 and are updated every 5 iterations using the following formula, which is similar to that used by Adulyasak et al. (2014):

$$\rho_d^- = \frac{10^\zeta}{T_{d/\underline{d}}}$$

where

- $\zeta = 1 + \lambda \frac{\nu_d}{\mu_d}$, μ_d and ν_d representing, respectively, the number of times that the method d has been used and the number of times this method has been able to improve the current solution. Clearly, the more a method d has been successful in improving the solution, the higher the value of $\frac{\nu_d}{\mu_d}$ is. λ is a parameter defining the importance given to the methods that can improve the solution. With a high value of λ, the algorithm will tend to select methods that improve the solution, favouring intensification and reducing diversification. In this paper, the value of λ was set to 3, as in Adulyasak et al. (2014).
- $T_{d/\underline{d}} = \frac{T_d}{T_{\underline{d}}}$, T_d being the average computational time per iteration for method d and $\underline{d} = argmin_{d \in \Omega^-} T_d$ is the method that requires the least amount of average computational time among all the methods in Ω^-.

Thus, the weights are computed accounting not only for the efficiency of the methods to improve the solution but also for the time efficiency of these methods. Methods able to improve the solution in short computational times will be assigned higher weights and are thus more likely to be selected.

4. *Acceptance criterion*: At each iteration, x', the solution resulting from applying the selected destroy and repair methods, is accepted or rejected according to the simulated annealing (SA) criterion. Let $\Delta f = f(x') - f(x)$ be the difference between the value of the new solution x' and the value of the current solution x. x' replaces x as the current solution if it is better than x (i.e., if $\Delta f > 0$). If $\Delta f \leq 0$, x' replaces x with probability $e^{\frac{\Delta f}{T_f}}$. The temperature factor T_f is initially set to a value T_f^0 and is multiplied every iteration by the cooling factor $0 < c < 1$ to decrease its value. The values of the parameters T_f^0 and c used in the numerical experiments are given in Section 5.
5. *Stopping criterion*: The stopping criterion is specified in terms of a maximum number of consecutive iterations $maxIter$ without an improvement of the objective function value. The value of this parameter used in the numerical experiments is also given in Section 5.

4.2 Destroy methods

This section describes 14 destroy methods used to select the β blocks to be removed from the current solution, x. Unless otherwise specified, all destroy methods follow the general scheme summarized in Algorithm 2 and select blocks based on priority rules. More precisely, an index p_i, henceforth referred to as a priority value, is first calculated for each block i using information derived from the current solution and/or from the history of the search. Blocks are then ranked in ascending or descending order of p_i, and the β first blocks are selected. The way the priority values p_i are computed differs from one method to another. Before describing how each method compute the p_i values and how it ranks the blocks, some extra notation is introduced.

Let $\alpha_{is} = \sum_{d \in \mathcal{D}} a_{ids}$ be the number of destinations to which block i can be sent without violating the admissibility constraints (recall that $a_{ids} = 1$ if block i is admissible for destination d under scenario s and 0 otherwise). Given the current solution x, denote by:

- $t_i(x)$: the period in which block i is extracted in solution x.
- $E_i(x) = \max_{j \in \mathcal{P}(i)} t_j(x)$: the earliest period in which block i can be extracted without violating the slope constraints (recall that $\mathcal{P}(i)$ denotes the set of immediate predecessors of block i).
- $L_i(x) = \min_{j \in \mathcal{S}(i)} t_j(x)$: the latest period in which block i can be extracted without violating the slope constraints ($\mathcal{S}(i)$ denotes the set of immediate successors of block i).
- $\mathcal{M}_s^t(x) = \sum_{i \in \mathcal{N}} w_{is} x_i^t$: the total tonnage, under scenario s, of blocks extracted in period t in solution x.

Algorithm 1 Adaptive Large Neighborhood Search framework

Initialization

x^0, an initial feasible solution
$x := x^0$, the current solution
$xbest := x^0$, the best solution found so far
Ω^-, the set of destroy methods that will be used in the search
Ω^+, the set of repair methods that will be used in the search
$\rho^- \in \mathbb{R}^{|\Omega^-|} = \{1, \ldots, 1\}$, the weights associated with the destroy methods
$\rho^+ \in \mathbb{R}^{|\Omega^+|} = \{1, \ldots, 1\}$, the weights associated with the repair methods

Algorithm

repeat

Adaptive Search Engine

Select a destroy method $d \in \Omega^-$ and a repair method $r \in \Omega^+$ using roulette-wheel selection based on previously obtained weights ρ^- and ρ^+, respectively.

Apply d to x, and then r to the resulting solution, x^-. Let x' be the so-obtained solution (i.e., $x' = r(d(x))$).

Acceptance Criterion

if accept(x', x) **then**
$x := x'$
end if

Update the incumbent

if x' is better than $xbest$ (i.e., if $f(x') > f(xbest)$) **then**
$xbest := x'$
end if

Adaptive weight adjustment

Update ρ^- and ρ^+

until the stopping criterion is met

return $xbest$.

Algorithm 2 to select the blocks to be removed from the current solution

Initialization

x, the current solution
$\mathcal{L} := \emptyset$, the list of selected blocks
$d \in \Omega^-$, the method to be used to select the blocks

Selecting the blocks

for each block i **do**
Compute the corresponding priority value p_i using the formula associated with method d
end for
Rank the blocks in ascending or descending order of the p_i values
Select the β first blocks
Add these blocks to the list $\mathcal{L}$
return $\mathcal{L}$.

- $\mathcal{P}^t_{ds}(x) = \sum_{i \in \mathcal{N}} w_{is} y^t_{ids}$: the total tonnage, under scenario s, of blocks processed at destination d during period t in solution x.

In what follows, in order to simplify the notation, we will omit the dependence on x whenever there is no risk of ambiguity (that is, we will use t_i instead of $t_i(x)$ and so on).

4.2.1 Random picker (D1)

The main purpose of this method is to diversify the search. It simply selects at random β blocks from the current solution x to alleviate the risk of choosing the same blocks many times. This can be seen as associating with each block a random integer value p_i chosen between 1 and N and ranking the blocks in ascending order of p_i (recall that N denotes the number of blocks being scheduled).

4.2.2 Historical frequency (D2)

This method uses historical information to select the blocks. It relies on a frequency array $\mathcal{F} = (\mathcal{F}_i)$ where each entry $\mathcal{F}_i$ is associated with a block i. This frequency array keeps track of the number of times that each block i has been involved in destroying the solution since the beginning of the search process; that is, the $\mathcal{F}_i$ values are initially set to 0, and whenever the block i is selected by a destroy method, then the value of the entry $\mathcal{F}_i$ is incremented. The priority values of the blocks are calculated as $p_i = \mathcal{F}_i$, and the blocks are ranked in ascending order of p_i. Thus, this method selects the blocks less frequently chosen so far in order to diversify the search.

4.2.3 Historical best (D3)

This method is also based on historical information, but it additionally uses the value of the current solution, $f(x)$. It aims to select the blocks that seem to be sent to the wrong destinations in the current solution with regard to the best known solutions. More precisely, let $\mathcal{X}_{it}$ be the set of solutions found so far in which block i is extracted in period t. We define an $N \times T$ matrix $\mathcal{Z}$. The value $\mathcal{Z}_{it}$ of entry (i,t) in this matrix corresponds to the value of the best solution in the set $\mathcal{X}_{it}$ (i.e., $\mathcal{Z}_{it} = \max_{sol \in \mathcal{X}_{it}} f(sol)$). All $\mathcal{Z}_{it}$ values are initially set to a large negative value, and they are updated each time a new solution is found. Recall that t_i denotes the period in which block i is extracted in the current solution. The priority value of block i is calculated as $p_i = \mathcal{Z}_{it_i} - f(x)$. Hence, a positive value of p_i means that in one of the solutions found in the past (sol), block i is extracted in the same period as it is in the current solution (x); however, the value of sol is better than the value of x. This might be because in the current solution i is sent to the wrong destinations, and thus removing it from these destinations might result in an improvement. The blocks are ranked in descending order of p_i to favour the blocks that present the largest deviations $\mathcal{Z}_{it_i} - f(x)$.

4.2.4 Greedy picker (D4)

This method selects the most costly blocks in the current solution in an attempt to extract them in other periods where they will generate more profit and/or send them to better destinations. Identifying these blocks reduces to identifying blocks whose removal increases the value of the objective function the most. Let $f(x - \{i\})$ denote the value of the current solution x if block i is removed from the schedule. The priority value of i is calculated as the difference between this value and the value of the current solution; i.e., $p_i = f(x - \{i\}) - f(x)$. The blocks are ranked in decreasing order of p_i.

4.2.5 Period mobility (D5)

This method selects the blocks that can be extracted in other periods without violating the slope constraints, for it is easier to modify the period in which these blocks are extracted and thus create new feasible solutions different from the current one. Moreover, when selecting such blocks, we take care to favour blocks whose removal does not increase the mining shortage cost (third term of the objective function (14)). Let ϵ be a

small value ($\epsilon = 0.0001$ in the numerical results presented in Section 5). The priority value of block i is calculated as follows:

$$p_i = \frac{L_i - E_i}{\sum_s \max\left(\epsilon, \underline{E_{t_i}} - \left(\mathcal{M}_s^{t_i} - w_{is}\right)\right)}.$$

Note that the numerator $(L_i - E_i)$ represents the number of periods in which block i can be extracted in the current solution without violating the slope constraints, while the denominator $\left(\sum_s \max(\epsilon, \underline{E_{t_i}} - (\mathcal{M}_s^{t_i} - w_{is}))\right)$ is equal to a small value if removing i from its current period t_i does not incur a mining shortage in t_i under any scenario, and it is equal to the total shortage amount considering all scenarios otherwise. Hence, a block with a high value of p_i has more feasible reinsertion possibilities (in terms of periods of extraction) and is less likely to incur a mining shortage if removed from its current period. For this reason, the blocks are ranked in descending order of p_i.

4.2.6 Destination mobility (D6)

This method has the same objective as the previous one; that is, to select blocks that can lead to a new feasible solution different from the current one. However, the blocks are ranked by the number of destinations to which they can be sent rather than the number of periods in which they can be extracted. Recall that $\alpha_{is} = \sum_{d \in \mathcal{D}} a_{ids}$ represents the number of admissible destinations for block i in scenario s. The priority values are calculated using the formula below, and the blocks are ranked in descending order of these values:

$$p_i = \begin{cases} \sum\limits_{s \in \mathcal{S}} \alpha_{is} & \text{if } i \text{ is extracted in the current solution } \left(\text{i.e., if } \sum\limits_{t \in \mathcal{T}} x_{it} = 1\right), \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, an unmined block cannot be sent to any destination, and thus if selected, one cannot modify its destination and get a new solution different from the current one. This is why unmined blocks are given less priority (the corresponding p_i values are set to 0 to avoid selecting them).

4.2.7 Combined mobility (D7)

This method combines the two previous ones (Period mobility and Destination mobility). It accounts not only for the periods in which each block can be extracted, but also for the destinations to which the block can be sent. More precisely, the priority values are computed using the following formula:

$$p_i = \begin{cases} \sum\limits_{s \in \mathcal{S}} (\alpha_{is} - 1) + \sum\limits_{t=E_i, t \neq t_i}^{L_i} \sum\limits_{s \in \mathcal{S}} \alpha_{is} & \text{if } \sum\limits_{t \in \mathcal{T}} x_{it} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The term $(\sum_{s \in \mathcal{S}} (\alpha_{is} - 1))$ accounts for the number of destinations to which block i can be sent if its period of extraction is not modified (its current destination under each scenario does not contribute to this term), while the term $(\sum_{t=E_i, t \neq t_i}^{L_i} \sum_{s \in \mathcal{S}} \alpha_{is})$ considers the other periods to which the block can be moved while satisfying the slope constraints. The blocks are ranked in descending order of p_i to select those with many feasible reinsertion possibilities.

4.2.8 Predecessor relatedness (D8)

This method has the same objective as the last three ones: create new feasible solutions different from the current one. However, it does not rely on the number of feasible reinsertion possibilities of each block nor does it follow the general scheme outlined in Algorithm 2. It selects blocks along with their predecessors mined in the same period, as this should allow modifying their period of extraction and create a new feasible solution (more specifically, advance their extraction together which will ensure the satisfaction of the slope constraints). This is done in three steps. In the first step, a random integer value τ is chosen between 1 and T (recall that T is the number of periods over which the blocks are being scheduled). Then, τ periods are

selected at random. The priority values are not calculated for all blocks but only for blocks extracted in one of these τ periods. Let t be such a period and i be a block extracted in t. Denote by γ_i the number of blocks in the inverted cone formed by i and its predecessors mined in t. Recall that β blocks should be selected. We set $p_i = \left|\gamma_i - \left\lceil \frac{\beta}{\tau} \right\rceil\right|$ and among the blocks currently extracted in t, the block with the smallest value of p_i is selected, as well as its predecessors mined in t. Ties are broken up randomly. This process is repeated for each of the τ periods.

4.2.9 Successor relatedness (D9)

This method is similar to the previous one except that it selects blocks along with their successors. Apart from the fact that γ_i represents the number of blocks in the cone formed by i and its successors mined in the same period, the procedure to select the blocks is identical to the procedure described in the previous section.

4.2.10 Mining reduction (D10)

The objective of this method is to select blocks whose removal can reduce the mining surplus (i.e., decrease the value of the fourth term of the objective function) or reduce the tightness of the mining capacity constraints to make room for new blocks. The priority values are computed as follows:

$$p_i = \begin{cases} \min(1, L_i - E_i) \max_{s \in \mathcal{S}} \frac{\mathcal{M}_s^{t_i}}{\overline{E^{t_i}}} & \text{if } \sum_{t \in \mathcal{T}} x_{it} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The blocks are ranked in descending order of p_i. Note that if an extracted block i cannot be moved to another period ($t \neq t_i$) without violating the slope constraints (i.e., if $\sum_{t \in \mathcal{T}} x_{it} = 1$ and $E_i = L_i = t_i$), then the corresponding priority value p_i is equal to 0 to avoid selecting it. This is done to make it easy for the repair method to create new feasible solutions. The priority value of a block that is not extracted in the current solution (i.e., such that $\sum_{t \in \mathcal{T}} x_{it} = 0$) is also set to $p_i = 0$ to avoid selecting it, as such blocks are not mined and thus have no influence on the mining capacity constraints (changing their periods does not affect the fourth term of the objective function (14)).

4.2.11 Processing reduction (D11)

This method is based on similar ideas as the previous one and aims to reduce the amount of surplus at the different destinations and/or the tightness of the processing capacity constraints. To be more precise, recall that $\mathcal{P}_{ds}^t$ denotes the total tonnage of blocks processed at destination d during period t under scenario s in the current solution, and that $\overline{F_d^t}$ is the processing capacity at d during t. Denote by $\sigma_{is} = \min(1, (\alpha_{is} - 1) + \alpha_{is}(L_i - E_i))$ the number of feasible reinsertion possibilities for block i under scenario s. Thus, if block i cannot be moved to another period (i.e., if $E_i = L_i = t_i$), and if it can be sent to only one destination under scenario s (i.e., and if $\alpha_{is} = 1$), then $\sigma_{is} = 0$. Otherwise, $\sigma_{is} \geq 1$. The priority values p_i are calculated using the formula below, and the blocks are ranked in descending order of p_i:

$$p_i = \begin{cases} \max_{s \in \mathcal{S}} \left\{ \sigma_{is} \frac{\mathcal{P}_{ds}^{t_i}}{F_d^{t_i}} \right\} & \text{if } \sum_{t \in \mathcal{T}} x_{it} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the way the p_i's are defined implies that the blocks that are not extracted in the current solution, as well as the blocks that are extracted but can neither be mined in a different period nor sent to another destination for all scenarios, have the lowest priority values (0) to avoid selecting them. As for the previous method, the idea is to prevent getting a solution similar to the current one when applying the repair method.

4.2.12 Shortage cautious (D12)

The purpose of this method is to find blocks that can be sent to destinations more profitable than their current destinations without incurring a shortage in their current destination. Denote by c the destination

to which block i is sent under scenario s in the current solution. Let d^* be the best destination to which block i can be sent under scenario s; i.e., $d^* = argmax_{d\in\mathcal{D}} v_{ids}$ (recall that v_{ids} represents the economic value to be generated if block i is processed at destination d in scenario s. This value is calculated as the return from selling the recovered metal minus the processing, transportation, and selling costs. v_{ids} is set to a large negative value if i is not admissible for destination d under scenario s). Again, let ϵ be a small positive value, and let C be a large positive value. Recall that δ_1 is the discount rate. The priority values are computed using the following formula, and the blocks are ranked in descending order of the priority values:

$$p_i = \begin{cases} \frac{1}{(1+\delta_1)^{t_i}} \sum\limits_{s\in\mathcal{S}} \frac{v_{id^*s} - v_{ics}}{\max\left(\epsilon, \underline{F_c^{t_i}} - \left(\mathcal{P}_{cs}^{t_i} - w_{is}\right)\right)} & \text{if } \sum\limits_{t\in\mathcal{T}} x_{it} = 1, \\ -C & \text{otherwise.} \end{cases}$$

In this formula, $\frac{1}{(1+\delta_1)^{t_i}}$ is used to account for the discount factor. The numerator $(v_{id^*s} - v_{ics})$ is used to favour blocks that can improve the second term of the objective function (14) the most. The denominator $\left(\max(\epsilon, \underline{F_c^{t_i}} - (\mathcal{P}_{cs}^{t_i} - w_{is}))\right)$ is used to favour blocks that will not incur a shortage if removed from their current destination (will not increase the fifth term of the objective function (14)). Finally, the priority values of blocks that are not extracted in the current solution are set to a large negative value to avoid selecting them, as these blocks do not contribute to any term of the objective function (14).

4.2.13 Empty one period (D13)

This method does not compute the priority values and does not necessarily select β blocks. It randomly selects one period and adds all the blocks extracted in that period to the list $\mathcal{L}$ (list of selected blocks). The motivation is to allow all destinations in a given period to be empty and completely remake the destination decisions (reconstruct the destination plans) with the repair method. By doing so, more opportunities for new block combinations in the different destinations are created.

4.2.14 Empty waste dump (D14)

In a given period and under a given scenario, some blocks might be sent to the waste dump while they can be sent to profitable destinations where they can be processed and generate revenue. This method aims to select such blocks to improve the solution. Let π_{is} be a parameter equal to 1 if block i is sent to the waste dump under scenario s in the current solution, and 0 otherwise. Clearly, $\pi_{is} = 0 \forall\ \ s$ if i is not extracted in the current solution. The priority values are computed using the formula below, and the blocks are ranked in descending order of these values:

$$p_i = \sum_{s\in\mathcal{S}} \pi_{is} \left[(\alpha_{is} - 1)(L_i - E_i + 1)\right]$$

The term $(\alpha_{is} - 1)$ accounts for the number of the destinations to which block i can be sent under scenario s, excluding its current destination, while $(L_i - E_i + 1)$ accounts for the periods in which i can be extracted without violating the slope constraints, including its current period of extraction. The factor π_{is} is used to avoid selecting blocks that are currently not in the waste dump under any scenario.

4.3 Repair methods

The blocks in the list $\mathcal{L}$, identified by the destroy method, are removed from the current solution x, resulting in an infeasible solution x^-. This means that all the variables associated with the blocks that are not in $\mathcal{L}$ are fixed, the remaining variables are "free". One of the seven methods described in this section is used to reinsert each block in $\mathcal{L}$ in other feasible periods and/or destinations to obtain a new feasible solution x'; that is, to optimize the "free" variables. The seven methods can be seen as variants of the methods used to generate the initial solution described in Section 3.

The following notation is used in the rest of this section. Given the solution to repair x^-, $\mathcal{B}^t = \{i \in \mathcal{N} : x_i^t = 1\}$ denotes the set of blocks mined in period t. The set of blocks processed in destination d during period t under scenario s is denoted by $\Lambda_{ds}^t = \{i \in \mathcal{N} : y_{ids}^t = 1\}$.

4.3.1 Random repair (R1)

This method considers one block $i \in \mathcal{L}$ at a time and sequentially chooses the period and the destinations in which i will be scheduled. The block is selected randomly, and the criteria to choose the period and the destinations are similar to those used in the RP and RD heuristics described in Sections 3.1 and 3.2, respectively. More specifically, the method starts by identifying the set of feasible periods $\mathcal{FP}(i)$ in which i can be extracted without violating the slope constraints. In doing so, the predecessors and the successors of i that are in the list $\mathcal{L}$ are not accounted for. Then, one of the periods in $\mathcal{FP}(i)$ is selected randomly, and i is scheduled to be mined in that period. The next step is to decide in which destination i will be processed under each scenario, and again this is done randomly; i.e., under each scenario, i can be processed in any destination as long as it is an admissible destination. When all scenarios are considered, i is removed from $\mathcal{L}$, another block is chosen, and the process is repeated until the list $\mathcal{L}$ is empty.

4.3.2 Greedy repair (R2)

This method is similar to the previous one in the sense that it considers blocks $i \in \mathcal{L}$ one at a time and sequentially determines the period and the destinations for the selected block before considering another block, but, to this end, it uses selection criteria different from those used by R1 as explained below. To simplify the presentation, we denote by

$$g\left(\mathcal{B}^t\right) = \sum_{i \in \mathcal{B}^t} \frac{E[c_i]}{\left(1+\delta_1\right)^t} + \frac{1}{S} \sum_{s \in \mathcal{S}} \left[\frac{p^-}{\left(1+\delta_2\right)^t} \max\Big(\underline{E^t} - \sum_{i \in \mathcal{B}^t} w_{is}, 0\Big) + \frac{p^+}{\left(1+\delta_2\right)^t} \max\Big(\sum_{i \in \mathcal{B}^t} w_{is} - \overline{E^t}, 0\Big) \right] \quad (29)$$

$$h\left(\Lambda_{ds}^t\right) = \sum_{i \in \Lambda_{ds}^t} \frac{v_{ids}}{\left(1+\delta_1\right)^t} - \frac{q_d^-}{\left(1+\delta_2\right)^t} \max\Big(\underline{F_d^t} - \sum_{i \in \Lambda_{ds}^t} w_{is}, 0\Big) - \frac{q_d^+}{\left(1+\delta_2\right)^t} \max\Big(\sum_{i \in \Lambda_{ds}^t} w_{is} - \overline{F_d^t}, 0\Big). \quad (30)$$

$g(\mathcal{B}^t)$ is used to evaluate the cost of mining a block i in period t accounting for all blocks that are already mined in this period, whereas $h(\Lambda_{ds}^t)$ is used to measure how profitable it is in scenario s and period t to process an additional block in destination accounting for blocks that are already processed in d.

Again, let $\mathcal{FP}(i)$ denote the set of feasible periods in which i can be extracted, considering only its predecessors and successors that are not in $\mathcal{L}$. For each period $t \in \mathcal{FP}(i)$, the repair method R2 first uses function (29) to compute the cost of mining i in period t: $\triangle_1(i,t) = g(\mathcal{B}^t \cup \{i\}) - g(\mathcal{B}^t)$. Then, it considers the scenarios sequentially and for each scenario, it finds, following a greedy approach, the destination in which i can be processed. For that, function (30) is used and the following is computed to measure how feasible and profitable it is to process i in d: $\triangle_2(i,d,s,t) = h(\Lambda_{ds}^t \cup \{i\}) - h(\Lambda_{ds}^t)$ if i is admissible to d under scenario s, and $\triangle_2(i,d,s,t)$ is set equal to a large negative value otherwise. Let $d^*\left(i,s,t\right) = \underset{d \in \mathcal{D}}{arg\max} \triangle_2(i,d,s,t)$. Clearly, the maximum profit that one can expect if block i is mined in period t is $\triangle\left(i,t\right) = -\triangle_1\left(i,t\right) + \frac{1}{S}\sum_{s \in \mathcal{S}} \triangle_2(i,d^*\left(i,s,t\right),s,t)$. Once all periods in $\mathcal{FP}(i)$ are considered, the period $t^* = argmax_{t \in \mathcal{FP}(i)} \triangle\left(i,t\right)$ is identified and block i is scheduled to be mined in t^* (i.e., it is included in the set $\mathcal{B}^t$). Finally, for each scenario s, i is sent to destination $d^*\left(i,s,t\right)$ (i.e., it is included in the set $\Lambda_{d^*(i,s,t)s}^t$).

4.3.3 Capacity cautious repair (R3)

This method is very similar to the previous one except for the following differences:

1. $g\left(\mathcal{B}^t\right)$ and $h(\Lambda_{ds}^t)$ are here defined by equations (31) and (32) below rather than (29) and (30) in order to select the periods/destinations that will leave the most residual capacity for forthcoming blocks. This is done to introduce some lookahead perspective when reinserting the blocks.

$$g\left(\mathcal{B}^t\right) = \frac{\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{B}^t} w_{is}}{\overline{E^t}} \quad (31)$$

$$h\left(\Lambda_{ds}^t\right) = \frac{\sum_{i \in \Lambda_{ds}^t} w_{is}}{\overline{F_d^t}}. \quad (32)$$

2. Accordingly, $d^*(i,s,t) = arg\min_{d\in\mathcal{D}} \triangle_2(i,d,s,t)$ instead of $d^*(i,s,t) = arg\max_{d\in\mathcal{D}} \triangle_2(i,d,s,t)$ (i.e., the best destination for a block in a given scenario and period is the one having the smallest capacity utilization).
3. $\triangle(i,t)$ is set equal to $\triangle_1(i,t) + \frac{1}{S}\sum_{s\in\mathcal{S}} \triangle_2(i,d^*(i,s,t),s,t)$ and $t^* = argmin_{t\in\mathcal{FP}(i)} \triangle(i,t)$.

Although some care is taken to include some look ahead features in R3, the sequential approach used in R1, R2, R3 is myopic and might lead to poor solutions. A better approach would be a global approach that considers multiple blocks when deciding the destinations. That is what the two methods described in the next two sections try to do.

4.3.4 MCFP repair (R4 and R5)

To repair the solution, this method combines the random repair heuristic (R1) and the MCFP heuristic described in Sections 4.3.1 and 4.3.2, respectively. More specifically, it starts by assigning feasible periods to blocks in $\mathcal{L}$ as in R1; that is, for each block $i \in \mathcal{L}$, $\mathcal{FP}(i)$ is first identified, then i is included in $\mathcal{B}^t$ where t is chosen randomly in $\mathcal{FP}(i)$. Once this step is completed, the destination plans for each period t that have been affected at the previous step are determined by solving the DP^t described in 3.2. This is done by applying the MCFP heuristic on each scenario separately. When applying the MCFP heuristic, the decisions associated with the blocks that were not in $\mathcal{L}$ (i.e., blocks that were not selected by the destroy method) are fixed to their current values. Another alternative, which is more flexible and might lead to better quality solutions but at the expense of longer computational times, is to reconstruct the destination plan from scratch (i.e., none of the parts of the plan is fixed and all the decisions are to optimize). In this paper, we examine the two alternatives, which leads to two variants of the MCFP repair method. The variant that solves a partial destination problem (first alternative) is denoted by R4, while the variant that solves the full destination problem (second alternative) is denoted by R5.

4.3.5 MIP repair (R6 and R7)

This method is very similar to the previous one. All the mining decisions are made first before designing the destination plans. Again, the latter are determined by considering only periods that have been affected when making the mining decisions, considering the scenarios separately. However, rather than using MCFP, a mixed-integer programming solver is used to find the optimal values of the variables y^t_{ids}, f^{t-}_{ds}, and f^{t+}_{ds} that maximize the net present economic value to be generated from processing the blocks mined in period t minus the total penalty costs of not satisfying the demands or exceeding the capacities of the different destinations during this period. Again, one can fix the binary variables y^t_{ids} corresponding to the blocks i that were not selected by the destroy method to their current values, which results in a variant of the MIP repair method that we will denote by R6 as one can "free" all variables, which gives yield to another variant of the MIP repair method denoted by R7.

5 Numerical results

The solution approach proposed in this paper is tested on the same copper-gold deposit used in the study by Goodfellow and Dimitrakopoulos (2013) where 175,598 blocks are considered for scheduling over 22 years. A set of 40 equiprobable scenarios is used to model the uncertainty in copper, gold, tonnages, and material types. There are three main material groups: sulphides, transition, and oxides, and each group is separated into two groups, for a total of six material types. There are also six destinations: a sulphide mill (SM), a sulphide heap leach (SHL), a sulphide waste dump (SWD), a transition heap leach (THL), an oxide heap leach (OHL), and an oxide waste (OW). Figure 2 (Goodfellow and Dimitrakopoulos, 2013) depicts the different material types and destinations at the copper-gold mine.

As mentioned in Goodfellow and Dimitrakopoulos (2013), the sulphide mill only accepts sulphide materials and produces both copper and gold. The sulphide heap leach produces only copper, but it accepts both sulphide and transition materials. Moreover, for chemistry reasons, it can only process the materials above 0.2% copper, hence the creation of distinct material types around this grade. Like the sulphide heap leach,

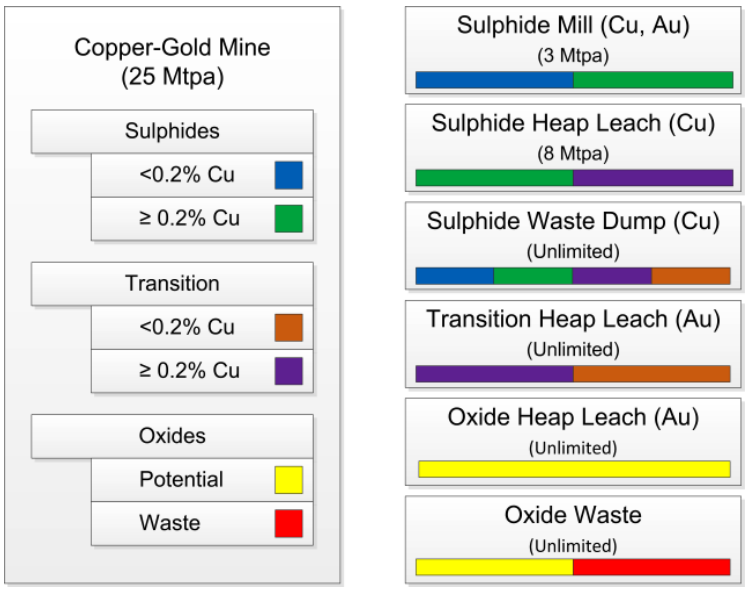

Figure 2: Material types and destinations at the copper-gold mine.

the sulphide waste dump can also extract only copper and accepts both sulphide and transition materials. The difference between the sulphide heap leach and the sulphide waste dump is that the latter is essentially a waste dump where excess sulphide and transition materials go for leaching, regardless of whether or not it is profitable to treat the material. The transition and oxide heap leaches accept only transition or oxide materials, respectively, and both extract only gold. The oxide waste dump accepts both oxide materials, but it does not treat any of the material and hence produces neither copper nor gold.

Table 1 shows the parameters used to define the right hand side of the constraints (lower and upper bounds), while Table 2 summarizes the economic parameters used to compute the coefficients of the objective function (14). For confidentiality purposes, the mining and processing costs are expressed relative to a base cost "k" to give an idea of the order of magnitude of costs for the various processes. Note that the parameters in Tables 1 and 2 are similar to those used in Goodfellow and Dimitrakopoulos (2013), except for the recoveries. Indeed, as previously noted in Section 1, Goodfellow and Dimitrakopoulos (2013) consider dynamic recoveries in the different processing facilities that depend on the average grade of the incoming material at these facilities, while in this paper, the recoveries are considered to be fixed at the values given in Table 2.

As noted previously, in this case study, 175,598 blocks are considered for scheduling over 22 years. A 45-degree slope angle is considered to define the precedence constraints, and 40 simulations are used to model the uncertainty in copper, gold, tonnages, and material types. The corresponding two-stage stochastic model ((1)–(13)) contains more than 931 million binary variables and millions of constraints, which is a size beyond the reach of exact methods. The results obtained with the heuristic-based solution approach described in this paper are presented below. Figures 3 and 4 show the risk profiles (P10, P50, and P90) for the tonnages processed at each destination and the cumulative NPV, respectively. Figure 5 shows sample cross-sections of the physical schedule obtained.

The following observations can be made from the graphs in Figure 3. The sulphide mill, which is limited to processing three million tonnes per year, is used at full capacity during the first 10 years. The amount processed in this destination drops after that, and a very small amount of material is treated towards the end of the horizon, when the risk is higher. The differences between the P10, P50 and P90 curves are negligible for the first 10 periods, indicating that, given the simulations used, there is a very small risk of not providing enough material to fill the sulphide mill capacity. The same observations apply to the sulphide heap leach. This processor has an eight million tonnes per year capacity, which is fully utilized, as the SHL receives

Table 1: Parameters used to define the right hand side of the constraints.

Parameter	**Value**
Lower bound on mining ($\underline{E^t}$)	0
Upper bound on mining ($\overline{E^t}$)	25 million tonnes
Lower bound on processing or demand at destination $d\left(\underline{F_d^t}\right)$	
Sulphide Mill (SM) for the first 10 years	2.9 million tonnes
Sulphide Mill (SM) for years 11 to 22	0
Sulphide Heap Leach (SHL) for the first 10 years	7.8 million tonnes
Sulphide Mill (SM) for years 11 to 22	0
Sulphide Dump Leach (SDL)	0
Transition Heap Leach (THL)	0
Oxide Heap Leach (OHL)	0
Oxide Waste (OW)	0
Upper bound on processing or capacity of destination $d\left(\overline{F_d^t}\right)$	
Sulphide Mill (SM)	3 million tonnes
Sulphide Heap Leach (SHL)	8 million tonnes
Sulphide Dump Leach (SDL)	Unlimited
Transition Heap Leach (THL)	Unlimited
Oxide Heap Leach (OHL)	Unlimited
Oxide Waste (OW)	Unlimited

Table 2: Economic parameters used to compute the objective function coefficients.

Parameter	**Value**
Mining cost	\$1*k/t
Sulphide Mill (SM)	
Processing cost	\$11.30*k/t
Recovery Cu	0.93
Recovery Au	0.59
Sulphide Heap Leach (SHL)	
Processing cost	\$2.98*k/t
Recovery Cu	0.7
Recovery Au	0
Sulphide Dump Leach (SDL)	
Processing cost	\$1.87*k/t
Recovery Cu	0.4
Recovery Au	0
Transition Heap Leach (THL)	
Processing cost	\$2.15*k/t
Recovery Cu	0
Recovery Au	0.5
Oxide Heap Leach (OHL)	
Processing cost	\$2.06*k/t
Recovery Cu	0
Recovery Au	0.55
Oxide Waste (OW)	
Processing cost	0
Recovery Cu	0
Recovery Au	0
Copper price (including selling and G&A costs)	\$2.88/lb Cu recovered
Gold price (including selling and G&A costs)	\$1480/oz Au recovered
Undiscounted cost for failing to meet the lower bound on mining (p^-)	10\$/t
Undiscounted cost for not satisfying the mining capacity (p^+)	10\$/t
Undiscounted cost for failing to meet the demand at destination d (q_d^-)	25\$/t if $d = SM$, 10\$/t otherwise
Undiscounted cost for exceeding the capacity of destination d (q_d^+)	25\$/t if $d = SM$, 10\$/t otherwise
Discount rate (δ_1)	10%
Risk discount rate (δ_2)	7%

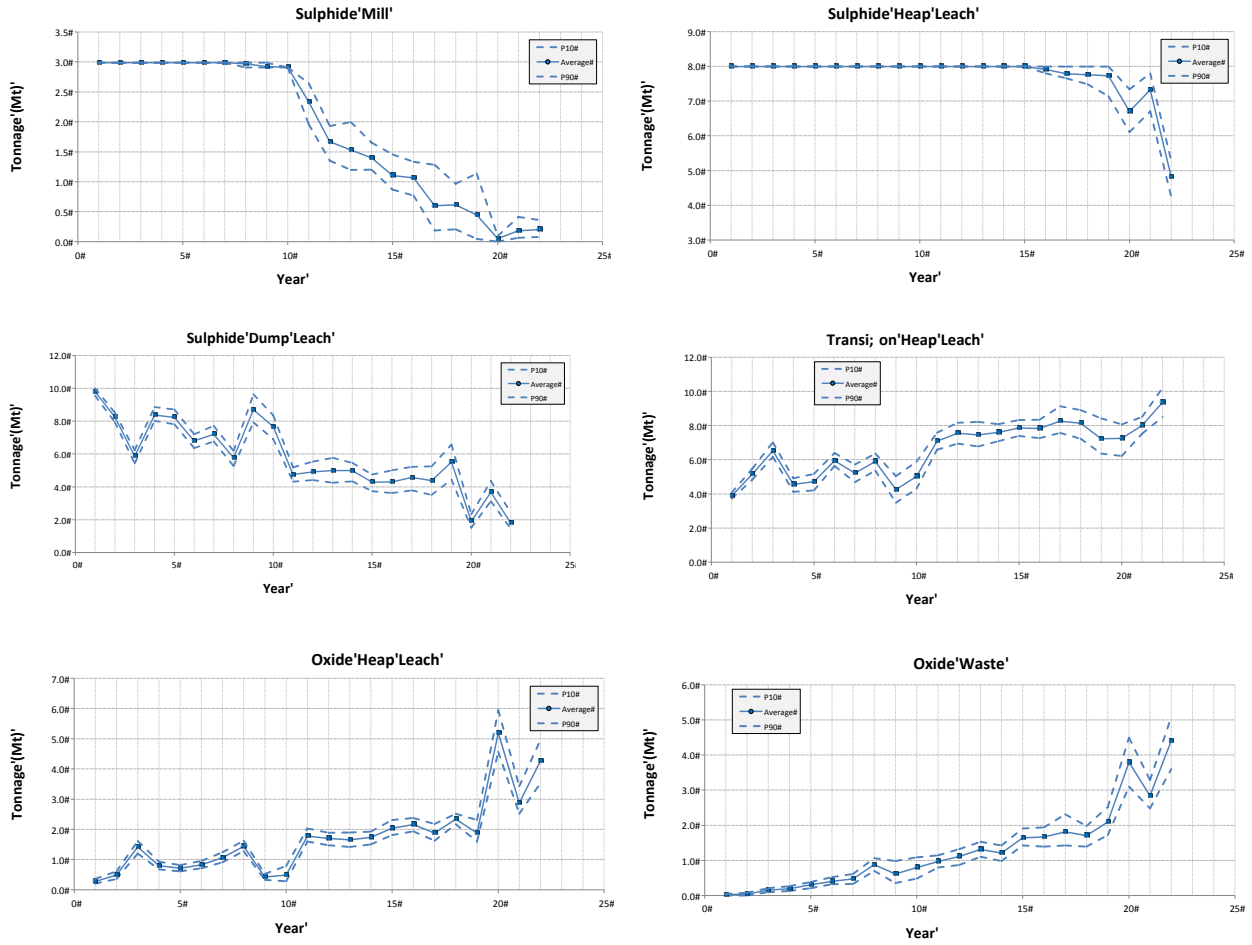

Figure 3: Risk analysis for tonnages processed at each destination.

consistently this amount except in the last three periods. The risk of not meeting the production targets is higher towards the end of the life of the mine. Regarding the sulphide dump leach, one can observe that there are more fluctuations in this processor compared to the two previous ones. This is due to the fact that this processor has an unlimited capacity and accepts sulphide and transition materials. As noted earlier in this section, the sulphide leach dump is essentially a waste dump where excess sulphide and transition materials go for leaching. Hence, surplus low grade material is treated in this processor. The transition heap leach has also an unlimited capacity. The amounts processed in this destination range between 4 and 9 million tonnes and the risk increases towards the end of the life of the mine, as can be seen from the more pronounced differences between the P10, P50 and P90 curves. The risk profiles for the oxide waste show that the tonnage of waste is higher in the last periods than it is in the first periods, as the extraction of non-profitable blocks is delayed. Less than one million tonnes of waste is mined during the first 10 periods. Finally, Figure 4 shows the risk profiles for the cumulative NPV. It appears from the graph that the first six periods account for 80% of the total NPV. This is due to the fact that the most valuable material is extracted and processed early in the life of the mine, as can be seen from the graphs in Figure 3.

6 Conclusions

This paper deals with the development of new models and solution approaches to address the large and complex problems faced by the mining industry when optimizing mineral supply chain operations under supply uncertainty. A two-stage stochastic model that integrates metal and material type uncertainty and

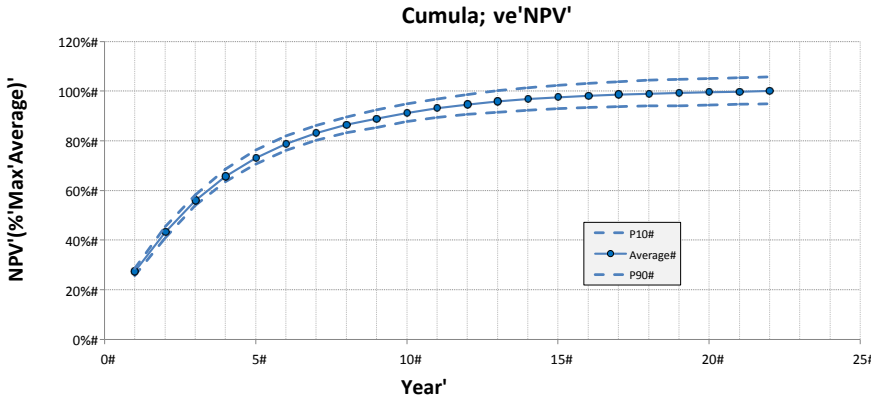

Figure 4: Risk analysis for cumulative discounted cash flows.

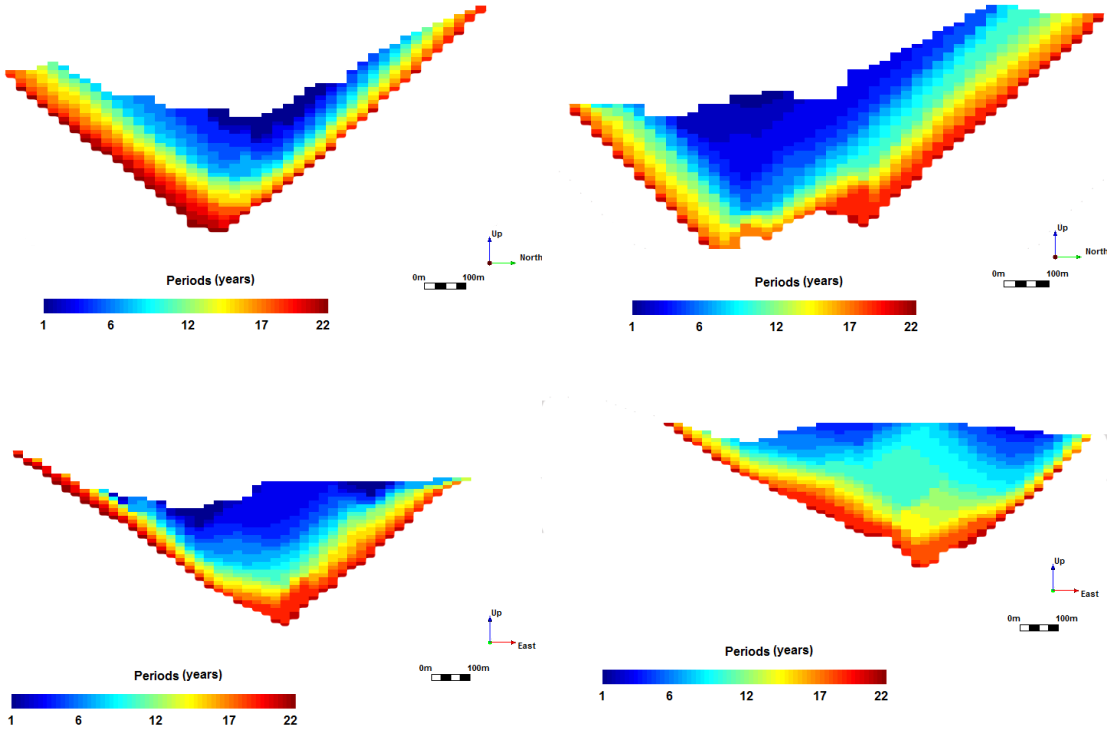

Figure 5: Cross-sections of the physical schedule.

simultaneously optimizes mining and destination decisions has been proposed. A heuristic approach based on the adaptive large neighborhood search framework (ALNS) has been developed to efficiently solve the proposed formulation. The structure of the problem has been exploited and relaxation and decomposition techniques have been used to obtain the initial solution to be improved with ALNS. ALNS uses 14 destroy methods and seven repair methods that are suited for the problem addressed in the paper and ensure both intensification and diversification.

The proposed solution approach has been tested on a copper-gold deposit with six material types and six destinations. The results of the numerical experiments indicate the ability of the proposed approach to efficiently address large instances with almost one billion binary variables and provide schedules where the most valuable material is mined and processed early in the life of the mine and where the risk of not meeting production targets is reduced.

Although the proposed mathematical model and solution approach consider a single mine, they can be easily extended to address the case of mineral supply chains comprised of multiple mines. Both the model and the solution approach can also be adapted to address the more general mineral supply chain where the first destinations are not the last destinations. Future work will follow these directions.

References

Adulyasak, Y., Cordeau, J.-F., Jans, R. (2014) Optimization-based adaptive large neighborhood search for the production routing problem. Transportation Science 48(1):20–45.

Albor, F., Dimitrakopoulos, R. (2010) Algorithmic approach to pushback design based on stochastic programming: Method, application and comparisons. IMM Transactions, Mining Technology 119(2):88–101.

Birge, J., Louveaux, F. (2011) Introduction to Stochastic Programming, Second Edition. Springer.

Boland, N., Dumitrescu, I., Froyland, G. (2008) A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology. Optimization Online, http://www.optimization-online.org/DB_FILE/2008/10/2123.pdf.

Cordeau, J.-F., Laporte, G., Pasin, F., Ropke, S. (2010) Scheduling technicians and tasks in a telecommunications company. Journal of Scheduling 13(4):393–409.

Demir, E., Bektas, T., Laporte, G. (2012) An adaptive large neighborhood search heuristic for the pollution-routing problem. European Journal of Operational Research 223(2):346–359.

Dimitrakopoulos, R., Farrelly, C., Godoy, M. (2002) Moving forward from traditional optimization: Grade uncertainty and risk effects in open pit mine design. IMM Transactions 111:A82–A88.

Dimitrakopoulos, R., Ramazan, S. (2004) Uncertainty based production scheduling in open pit mining. SME Transactions 316:106–112.

Dowd, P. (1994) Risk assessment in reserve estimation and open-pit planning. Transactions of the Institution of Mining and Metallurgy 103:A148–A154.

Dowd, P. (1997) Risk in minerals projects: Analysis, perception and management. Transactions of the Institution of Mining and Metallurgy 106:A9–A18.

Godoy, M., Dimitrakopoulos, R. (2004) Managing risk and waste mining in long-term production scheduling of open-pit mines. SME Transactions 316:43–50.

Goodfellow, R., Dimitrakopoulos, R. (2012) Mining supply chain optimization under geological uncertainty. COSMO research report 1(6):1–43.

Goodfellow, R., Dimitrakopoulos, R. (2013) Global asset optimization of open-pit mining complexes under uncertainty. COSMO research report 1 (7):1–44.

Kent, M., Peattie, R., Chamberlain, V. (2007) Incorporating grade uncertainty in the decision to expand the main pit at the navachab gold mine, namibia, through the use of stochastic simulation. The Australasian Institute of Mining and Metallurgy, Spectrum Series 14:207–218.

Kovacs, A.A., Parragh, S.N., Doerner, K.F., Hartl, R.F. (2012) Adaptive large neighborhood search for service technician routing and scheduling problems. Journal of Scheduling 15(5):579–600.

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. (2013) A network-flow based algorithm for scheduling production in multi-processor open-pit mines accounting for metal uncertainty. Les Cahiers du GERAD, G–2013–63, HEC Montréal.

Lamghari, A., Dimitrakopoulos, R., Ferland, J. (2014) A variable neighbourhood descent algorithm for the open-pit mine production scheduling problem with metal uncertainty. Journal of the Operational Research Society 65:1305–1314.

Masson, R., Lehuede, F., Peton, O. (2013) An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science 47(3):344–355.

Menabde, M., Froyland, G., Stone, P., Yeates, G. (2005) Mining schedule optimisation for conditionally simulated orebodies. In: Orebody modelling and strategic mine planning: uncertainty and risk management models: 353–357.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. (1953) Equation of state calculations by fast computing machines. Journal of Chemical Physics 21:1087–1092.

Muller, L.F., Spoorendonk, S., Pisinger, D. (2012) A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. European Journal of Operational Research 218(3):614–623.

Osanloo, M., Gholamnejad, J., Karimi, B. (2008) Long-term open pit mine production planning: A review of models and algorithms. International Journal of Mining, Reclamation and Environment 22:3–53.

Pisinger, D., Ropke, S. (2007) A general heuristic for vehicle routing problems. Computers & Operations Research 34(8):2403–2435.

Pisinger, D., Ropke, S. (2010) Large neighborhood search. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics. Springer, Boston: 399–419.

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

Ravenscroft, P. (1992) Risk analysis for mine scheduling by conditional simulation. Transactions of the Institution of Mining and Metallurgy, Section A: Mining Technology: A104– A108.

Ribeiro, G.M., Laporte, G. (2012) An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research 39(3):728–735.

Ropke, S., Pisinger, D. (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40:455–472.

Shaw, P. (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. Technical Report, University of Strathclyde, Glasgow.

Whittle, D., Bozorgebrahimi, A. (2005) Hybrid pits – linking conditional simulation and lerchs-grossmann through set theory. In: Orebody modelling and strategic mine planning: uncertainty and risk management models: 37–42.

Whittle, J. (2009) The global optimizer works – What next? In: Advances in orebody modelling and strategic mine planning: old and new advances in a changing world, AusIMM Spectrum Series 17:3–5.