International Journal of Mining, Reclamation and Environment

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/nsme20

An adaptive large neighborhood search heuristic to optimize mineral value chains under metal and material type uncertainty

Amina Lamghari & Roussos Dimitrakopoulos

To cite this article: Amina Lamghari & Roussos Dimitrakopoulos (2022) An adaptive large neighborhood search heuristic to optimize mineral value chains under metal and material type uncertainty, International Journal of Mining, Reclamation and Environment, 36:1, 1-25, DOI: 10.1080/17480930.2021.1949858

To link to this article: https://doi.org/10.1080/17480930.2021.1949858

INTERNATIONAL JOURNAL OF MINING, RECLAMATION AND ENVIRONMENT
2022, VOL. 36, NO. 1, 1–25
https://doi.org/10.1080/17480930.2021.1949858

OPEN ACCESS

An adaptive large neighborhood search heuristic to optimize mineral value chains under metal and material type uncertainty

Amina Lamghari[a,b] and Roussos Dimitrakopoulos

[a]COSMO - Stochastic Mine Planning Laboratory, Department of Mining and Materials Engineering, McGill University, Montreal, Quebec, Canada; [b]École de gestion, département de management, Université du Québec À Trois-Rivières, Trois-Rivières, Quebec, Canada

ABSTRACT
This paper addresses the optimisation of mineral value chains under metal and material type uncertainty. A mathematical model to simultaneously optimise the extraction decisions and the destination decisions is proposed. A fix-and-optimise 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 neighbourhood search heuristic is developed to improve this solution. The proposed solution approach is tested on a real copper-gold mining complex. 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 successfully managed.

ARTICLE HISTORY
Received 3 July 2020
Accepted 6 June 2021

KEYWORDS
Heuristics; adaptive large neighbourhood search; mineral value chain; mining complex; metal uncertainty; material type uncertainty; risk management

1. Introduction

A mineral value chain includes and involves all aspects of a mining operation from extraction to ore processing and transportation, leading to refined minerals to be sold in the market. Simultaneous optimisation of all aspects of the mineral value chain overcomes the drawbacks of viewing the series of processes as occurring in a set of silos by providing a holistic view that accounts for the dynamic nature of relationships between and among the various parts of the chain. However, it entails solving a large and complex combinatorial optimisation problem. Accounting for the geological uncertainty inherent to mining operations makes the problem even more difficult to solve and thus presents substantial computational challenges, requiring approaches that, while not necessarily guaranteeing optimal solutions, can obtain good quality solutions in reasonable computing times.

In the last fifteen years, there has been a sustained development of such approaches and, concurrently, modelling frameworks, that incorporate aspects of mining operations with more details [1,2]. Early research focused on developing modelling frameworks that explicitly account for geological uncertainty in open-pit mine production scheduling, often considering a single-element deposit and one processor. Ramazan and Dimitrakopoulos [3] were the first to propose a two-stage stochastic framework that uses multiple equiprobable scenarios to account for geological uncertainty. Small instances with up to 200 blocks were solved using the general-purpose solver CPLEX. The approach was later extended to handle multi-element deposits [4,5], multiple processors [6], and to optimise cut-off grade [7]. Because large instances of practical interest are beyond the scope

CONTACT Amina Lamghari amina.lamghari@uqtr.ca COSMO - Stochastic Mine Planning Laboratory, Department of Mining and Materials Engineering, McGill University, Montreal, Quebec, Canada

of general-purpose solvers, several studies have employed metaheuristics, such as simulated annealing [8–11], tabu search [12,13], variable neighbourhood search [14], ant colony optimisation [15], and genetic algorithms [16]. In recent research, it is increasingly common to consider hybrid methods that combine different heuristics or elements of different metaheuristics with other optimisation techniques to increase their efficiency. A hybrid method combining multi-neighbourhood simulated annealing with particle swarm optimisation was introduced in [17] and further studied in [18,19]. Methods that combine network flow techniques with (meta)heuristics were considered in [20–23], while hyper-heuristic approaches that combine elements from reinforcement learning and tabu search have recently been proposed in [24]. There is also some work where simulation is combined with different analysis tools to manage geological uncertainty and related risks [25,26].

While considerable progress has been made at the modelling and solving levels, there is still a need to devise enhanced algorithms that overcome the challenges of scale, complexity, and uncertainty in the optimisation of mineral value chains. Such an algorithm is what we propose in this paper. A novel solution approach based on the adaptive large neighbourhood search (ALNS) framework is introduced. This framework was initially proposed by Ropke and Pisinger [27] for vehicle routing problems. The approach consists of destroying a part of the current solution and reconstructing it in a different way in an attempt to improve it. To do so, several destroy and several repair methods are used. The heuristic keeps track of the performance of each method and adapts to the instance being solved by favouring methods that have performed well up to that point. The algorithm proposed in this paper integrates this classical framework with new destroy and repair methods specifically designed for the mining problem under study. Another new method is introduced to generate the initial solution to be improved by ALNS. This method, unlike the ones proposed in [13,17–24], exploits the structure of the problem and uses relaxation and decomposition techniques to efficiently generate a good quality solution. We show through a real case study, namely a copper-gold mining complex, that the proposed solution approach provides a powerful algorithmic framework to meet the inherent challenges of complexity, scale, and uncertainty in optimising mineral value chains.

The remainder of the paper is organised as follows: In Section 2, the mineral value chain studied in this paper is formally described, the approach used to deal with metal and material type uncertainty is outlined, and a mathematical formulation of the optimisation problem is introduced. The hybrid method used to generate the initial solution and the ALNS algorithm used to improve this solution are described in Sections 3 and 4, respectively. Computational results are reported and discussed in Section 5, followed by conclusions in Section 6.

2. Formal problem description and mathematical formulation

The mineral value chain considered in this paper consists of a single open-pit mine that feeds a set of processing facilities, henceforth referred to as destinations, that produce the final products to be sold (the refined minerals). An extracted block cannot be sent to just any destination. It can only go to a predetermined subset of destinations depending on the block's material type (e.g. sulphides, oxides), which determines its admissibility for a given destination. Extracting blocks incurs mining costs, sending blocks to the destinations incurs transportation and/or processing costs, while the refined mineral processed through the different facilities is sold and generates revenues. The problem is to decide which blocks to extract and when to extract them, and where to send each extracted block so as to maximise the total discounted profit, while meeting the capacities and the specifications at each destination. Decisions are also shaped by the goal of meeting the physical and technical requirements for extracting the blocks (slope constraints).

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. The material type has an impact on the tonnage of the blocks (W), the mining costs

(C), and the admissibility of the blocks to the different processing destinations (A). The metal content has an impact on the economic value (V) generated after processing the extracted blocks. 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 realisation of ξ (i.e. a scenario). It is assumed that there is a finite number of scenarios and that the scenarios are equiprobable.

The problem can be formulated as a two-stage stochastic program with recourse [28]. The following notation is used in the formulation:

(1) **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, $\mathcal{S} = \{1, \ldots, S\}$.

(2) **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, $s \in \mathcal{S}$.

(3) **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).
- $\overline{E}$: 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 $E[c_i] = \frac{1}{S} \sum\limits_{s \in \mathcal{S}} c_{is}$.
- 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: Economic discount rate.
- δ_2: Geological discount rate.

(4) **Variables**

- x_i^t: 1 if block i is extracted in period t, 0 otherwise.
- e_s^{t-}: Amount in shortage of extracted material in scenario s and period t.
- e_s^{t+}: Amount of extra material extracted 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 $\overline{F_d^t}$.

Using the notation introduced above, the problem can be modelled as follows:

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

s.t.

$$\sum_{t\in\mathcal{T}} x_i^t \le 1 \qquad i\in\mathcal{N} \tag{2}$$

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

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

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

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

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

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

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

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

s.t.

$$\sum_{i\in\mathcal{N}} w_{is}x_i^t + e_s^{t-} \ge \underline{E^t} \qquad t\in\mathcal{T} \tag{6}$$

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

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

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

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

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

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

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

The objective function (1) minimises the expected first-stage costs and maximises 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 extracting less material than the minimum required (5b), the penalties for exceeding the mining capacity (5 c), 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 in [29] to define the importance given to satisfy 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:

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

Constraints (2) and (3) define the feasible set of the first-stage problem (reserve and slope constraints, respectively). They ensure that a block is extracted at most once (constraints (2)) and prevent a block from being extracted before its predecessors (constraints (3)). Constraints (6) and (7) are related to the minimal and maximal extraction levels at each period (mining constraints). Shortage, as well as surplus, is allowed, but incurs penalty costs. Constraints (8) and (9) link the extraction decisions to the destination decisions. More specifically, they allow a block to be sent to one and only one destination if the block is extracted (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 mining complex with 10,000 blocks and 4 destinations, to be scheduled over 5 years accounting for 10 metal and material type scenarios represents more than one million binary variables (1,050,000) and millions of constraints. A problem of this size is beyond the reach of exact methods and general-purpose solvers, so 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 (initialisation phase), then this solution is improved in the second phase (improvement phase) using an adaptive large neighbourhood heuristic (ALNS). The methods used in the two phases of the solution procedure are described in the following sections.

3. Initialisation phase

The approach used to generate the initial solution takes advantage of the problem structure and uses a fix-and-optimise 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 optimise another subset of variables. This is done as follows.

Using the general time-decomposition approach described in [14], the periods $t \in \mathcal{T}$ are considered one at time in increasing order. In this paper, unlike in [14], each sub-problem associated with a period t includes two sub-problems: an extraction 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 extracted in period t is first determined, then each of those blocks determined in the first step is assigned to one admissible destination in each scenario. This strategy not only prevents blocks not extracted 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 summarise, starting with $t = 1$, the extraction problem handles the extraction 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 optimise extraction decisions without explicitly modelling the destination decisions? One way to get around this difficulty is to relax the problem as explained in the next section.

3.1. Solving the extraction problem (EP^t)

In what follows, t is fixed, and the extraction problem is denoted by EP^t. Recall that this problem is solved to determine the set of blocks to be extracted 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:

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

s.t.

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

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

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

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

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

where, $\mathcal{R}^t$ is the set of blocks not extracted 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) maximises the expected net present value, assuming that under each scenario each extracted block is processed in the most profitable destination. It also minimises the expected penalty costs incurred whenever the amount extracted in period t does not fall within the specified limits $\left[\underline{E}^t, \overline{E}^t\right]$. 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 EP^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 EP^t can be solved using a mixed-integer programming solver.

Because the extraction decisions will be modified in the improvement stage (i.e. when applying the ALNS heuristic), EP^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.

3.2. Solving the destination problem (DP^t)

Once EP^t is solved using the method 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 extraction variables x_i^t are fixed according to the solution obtained when solving EP^t, the original sub-problem associated with period t reduces to the following problem:

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

s.t.

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

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

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

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

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

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

The objective function (21) maximises the total expected economic value to be generated from the blocks in $\mathcal{B}^t$; that is, those determined by solving EP^t. It also minimises 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 $\left[\underline{F_d^t}, \overline{F_d^t}\right]$; 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:

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

The method proposed to solve a given scenario sub-problem DP_s^t consists of solving the linear relaxation of DP_s^t, obtained by replacing the integrality constraints (26) by $y_{ids}^t \in [0,1]$, and then applying a repair heuristic to modify the so-obtained solution and 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 extracted in period t (i.e. those determined by solving EP^t).

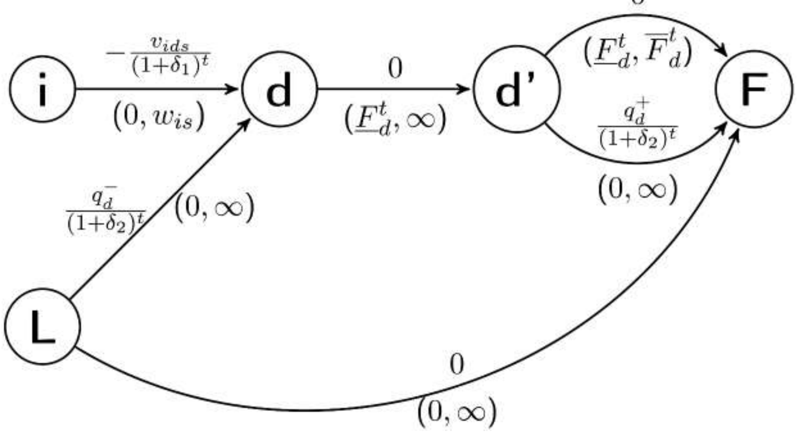
- $\mathcal{D}$: destination vertices representing the different destinations to which the blocks can be sent once extracted.
- $\mathcal{D}'$: dummy destination vertices, which are copies of the destination vertices. These dummy vertices are used to absorb the amount in excess at each destination.
- 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 set to $-\frac{v_{ids}}{(1+\delta_1)^t}$, the lower bound is set to 0, and the upper bound is set to 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 uncapacitated 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 uncapacitated. 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 uncapacitated 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 uncapacitated and its cost is set equal to 0.

Figure 1 illustrates the graph in a situation with one block i and one destination d to which i is admissible. The cost and the bounds of each arc are displayed above and below the arc, respectively.

Reformulating the problem as an MCFP allows us to quickly solve the linear relaxation of DP_s^t. The so-obtained fractional solution is rounded using 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 to 1). Otherwise, i is



i
d
d'
F
L
$-\frac{v_{ids}}{(1+\delta_1)^t}$
$(0, w_{is})$
0
$(\underline{F}_d^t, \infty)$
0
$(\underline{F}_d^t, \overline{F}_d^t)$
$\frac{q_d^+}{(1+\delta_2)^t}$
$(0, \infty)$
$\frac{q_d^-}{(1+\delta_2)^t}$
$(0, \infty)$
0
$(0, \infty)$

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

assigned to destination $\hat{d} = argmax_{d:(i,d)\in A} z_{id}$ (i.e. $y^t_{i\hat{d}s}$ 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.

4. Improvement phase

This phase applies an adaptive large neighbourhood search heuristic (ALNS) to improve the initial solution generated using the method described in the previous section. In what follows, the main features of ALNS are first described in a general context, then a step-by-step description of the proposed adaptation of ALNS is provided. For a discussion of the pros and cons of ALNS, the reader is referred to [27,30].

4.1. General ALNS framework

Because ALNS is an extension of the large neighbourhood search heuristic (LNS) proposed by Shaw [31], 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 [32], 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 neighbourhood search method where the neighbourhood 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 neighbourhood is (a large number of variables are modified at each iteration). To explore the large neighbourhood defined by the destroy and repair methods, LNS does not generate it entirely. It rather samples it.

ALNS is based on similar ideas as LNS, except that its uses several destroy and several repair methods within the search rather than using the same destroy/repair method at each iteration [27]. 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. 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 difficult combinatorial problems such as vehicle routing, scheduling, and production problems [30,33–36], but, to the best of our knowledge, it has not been applied to solve mine planning problems.

4.2. Components of the proposed adaptation of ALNS

The main components of our adaptation of ALNS are described below. The description follows the framework outlined in the previous section and summarised in Algorithm 1.

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
  $\Omega^-$, the set of destroy methods that will be used in the search
  $\Omega^+$, 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 $\rho^-$ and $\rho^+$, 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 $\rho^-$ and $\rho^+$
  until the stopping criterion is met
  return $xbest$.
```

4.2.1. Destroy and repair methods

Fourteen destroy methods and seven repair methods, which are described in detail in appendices A and B, were developed for use in the ALNS framework. The destroy methods select a set of blocks to be removed from the current solution, whereas the repair methods select new periods and/or new destinations for these blocks. The destroy methods serve various purposes and are appropriate for different situations. Some are fit for diversification and choose the blocks randomly, for example, while others are fit for intensification and choose critical blocks. These may be blocks that are currently extracted in the period with the highest penalty cost or blocks that are sent to the waste dump instead of to a processor, where they can generate revenue. Other methods choose the blocks based on historical information either to drive the search towards unexplored regions of the feasible space or to avoid returning to already visited bad solutions or based on some relatedness measure to facilitate repairing the solution. The repair methods, on the other hand, include greedy heuristics, exact methods that reconstruct the destination plan for a given period from scratch, and variants of the MCFP method described in Section 3.2. Random fast methods are also used to avoid performing the same modifications to the solution repeatedly and thereby stagnating the search.

4.2.2. Large neighbourhood

As mentioned in the previous section, at each iteration, a set of blocks, selected using a destroy method, are removed from the current solution and are then rescheduled in different periods and/or sent to different destinations using a repair method. Let β be the number of blocks involved in these modifications. β is a parameter of the ALNS heuristic that has a significant impact on its efficiency. Clearly, a small value of β might not allow a thorough exploration of the search space, as the effect of a large neighbourhood is lost. On the other hand, a large value of β reduces ALNS to independent re-optimisation, 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:

$$\beta = \min\left(\max\left(2^{\left[\left(\kappa/10\right)-1\right]}\beta, \beta_{min}\right), \beta_{max}\right)$$

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. In this paper, the values of the parameters β_{min} and β_{max} are set to $0.001N$ and $0.3N$, respectively (recall that N denotes the number of blocks being scheduled).

4.2.3. Adaptive search engine

As in [27] 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.

4.2.4. 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 in [36]:

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

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.

- $T_{d/\underline{d}} = \frac{T_d}{T_{\underline{d}}}$, 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 that can improve the solution in short computational times will be assigned higher weights and are thus more likely to be selected.

4.2.5. 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 set to 0.3 and 0.995, respectively.

4.2.6. 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. In the numerical results presented in Section 5, the value $maxIter = 100$ was used.

5. Numerical results

The solution approach proposed in this paper is tested on a real copper-gold mining complex 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. These scenarios were generated from the available drilling data using the geostatistical sequential simulation framework [37–39] and the direct block simulation method for multiple correlated variables [40].

There are three main material groups: sulphides, transition, and oxides, and, to respect the chemistry requirements, each group is further separated into two sub-groups, for a total of six

Table 1. Parameters used to define the right-hand side of constraints (6), (7), (10), and (11).

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* ($\underline{F}_d^t$)	2.9 million tonnes
Sulphide Mill (SM) for the first 10 years	0
Sulphide Mill (SM) for years 11 to 22	7.8 million tonnes
Sulphide Heap Leach (SHL) for the first 10 years	0
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)	
Upper bound on processing or capacity of destination d($\overline{F}_d^t$)	3 million tonnes
Sulphide Mill (SM)	8 million tonnes
Sulphide Heap Leach (SHL)	Unlimited
Sulphide Dump Leach (SDL)	Unlimited
Transition Heap Leach (THL)	Unlimited
Oxide Heap Leach (OHL)	Unlimited
Oxide Waste (OW)	

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

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

material types, with the sulphide and transition material groups separated into two different material types based on being above or below 0.2% copper, and the oxide materials classified as ore or waste depending on chemistry. There are also six destinations: a sulphide mill (SM), a sulphide heap leach (SHL), a sulphide dump leach (SDL), a transition heap leach (THL), an oxide heap leach (OHL), and an oxide waste (OW). 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, 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, the sulphide dump leach can also extract only copper and accepts both sulphide and transition materials. The difference between the sulphide heap leach and the sulphide dump leach 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 produce only gold. The oxide waste 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 constraints (6), (7), (10), and (11), while Table 2 summarises 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. To set the penalty costs, a preliminary analysis, similar to that discussed in [4,17], was conducted.

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

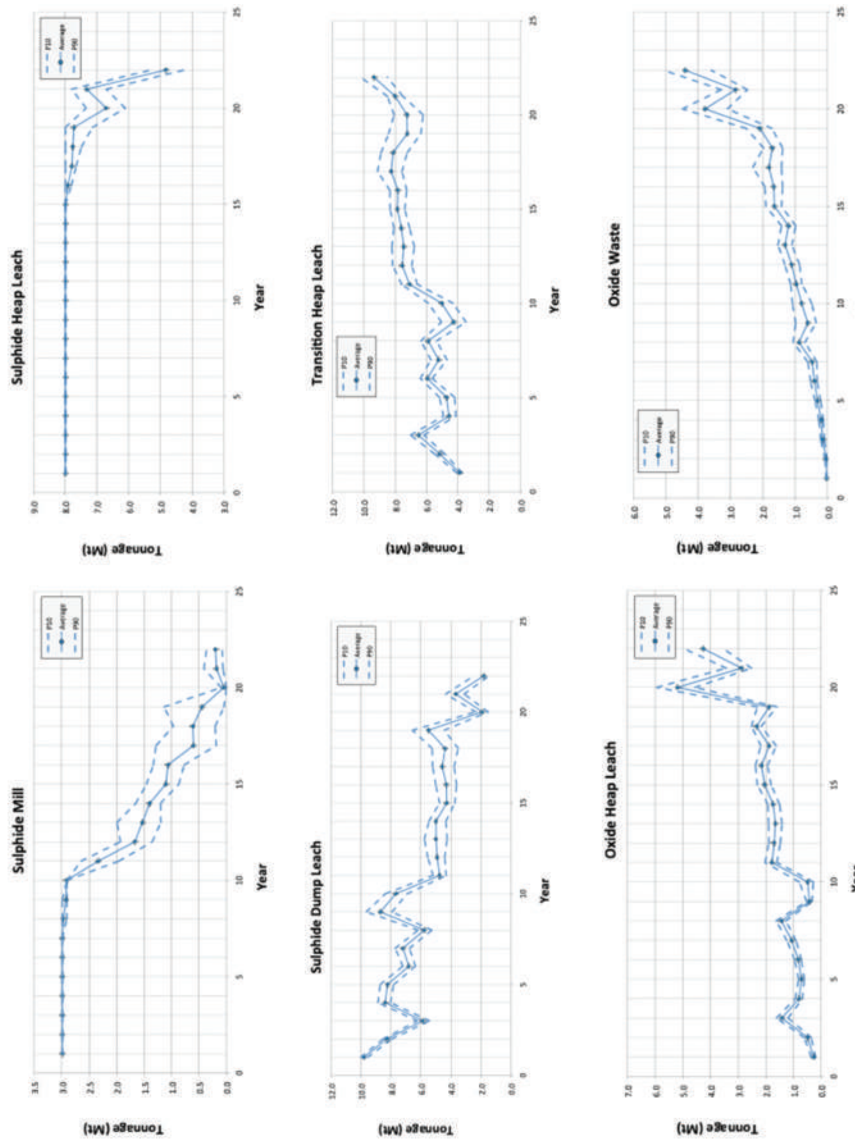


Figure 2. Risk analysis for tonnages processed at each destination. The dotted lines represent P10 and P90, while the solid lines represent P50.

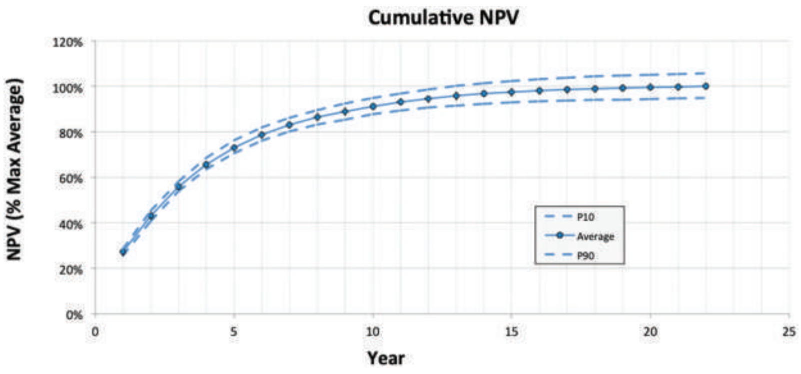


Figure 3. Risk analysis for cumulative discounted cash flows. The dotted lines represent P10 and P90, while the solid line represents P50.

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. CPLEX was not able to solve even the linear relaxation of the problem within four weeks. The heuristic-based solution approach described in this paper required significantly less time, namely 3678.03 minutes. The results obtained are presented below. Figure 2 shows the risk profiles (10th, 50th, and 90th percentiles denoted by P10, P50, and P90, respectively) for the yearly tonnages processed at each destination, while Figures 3 ,4 and show the risk profiles for the cumulative NPV and sample cross-sections of the physical schedule obtained, respectively.

The following observations can be made from the graphs in Figure 2. 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 life of the mine, 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 (SHL) has an eight million tonnes per year capacity, which is fully utilised, as the SHL consistently receives 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 dump leach 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. Figure 3, which shows the risk profiles for the cumulative NPV, indicates 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 2.

In sum, it is apparent from the results discussed above that the proposed solution approach can not only address large instances of realistic size and handle the complexity of simultaneously optimising extraction and destination decisions while accounting for both metal and material type uncertainty, but it can also provide very good quality solutions; that is, schedules where the

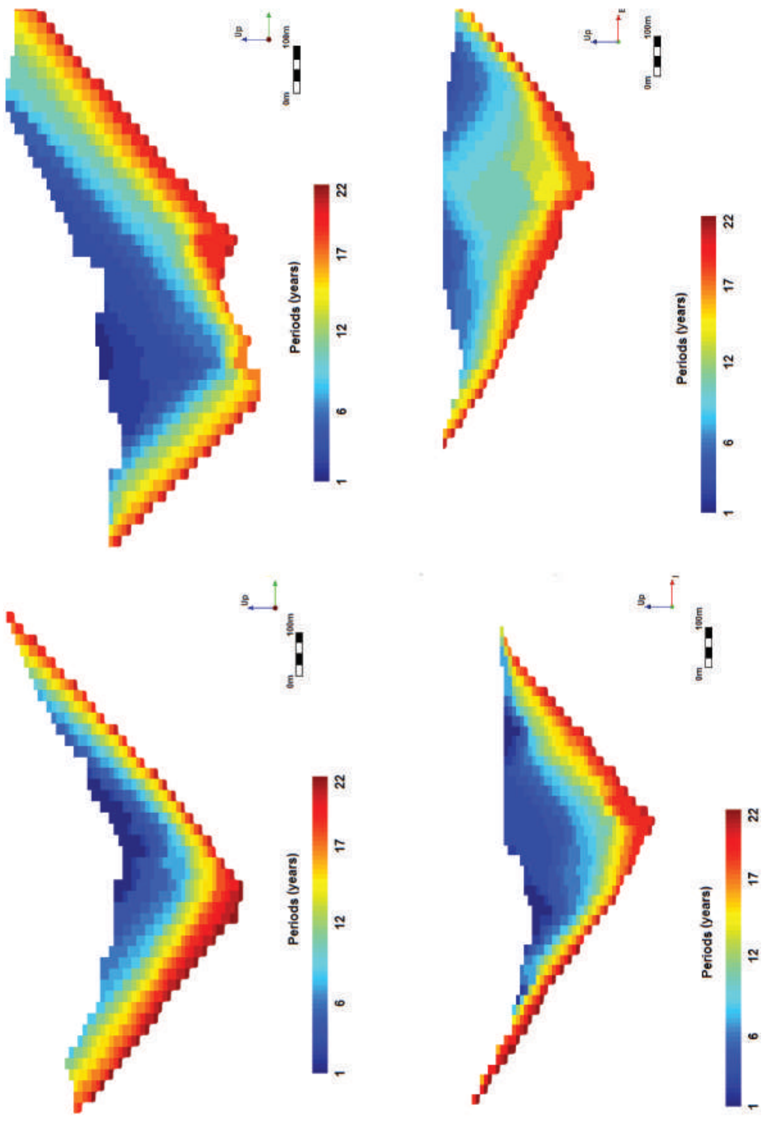


Figure 4. Cross-sections of the physical schedule obtained with the proposed solution approach.

most valuable material is extracted and processed early in the life of the mine, where processing capacities are fully utilised in most periods, and more importantly, where the tonnage profiles are quite consistant across the scenarios, indicating that the risk of not meeting production targets is managed.

8. Conclusions

This paper introduces 1) a new mathematical model for optimising mineral value chains under uncertainty; 2) a new method to generate an initial solution; and 3) a new method to improve the initial solution. The model is a two-stage stochastic model that integrates metal and material type uncertainty and simultaneously optimises extraction and destination decisions. The first new method exploits the structure of the problem, using relaxation and decomposition techniques. The second new method is based on the adaptive large neighbourhood search framework (ALNS). It uses fourteen destroy methods and seven repair methods that are suited for the problem addressed in the paper to ensure both intensification and diversification.

The proposed solution approach was tested on a real copper-gold mining complex 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 extracted and processed early in the life of the mine and where the risk of not meeting production targets is successfully managed.

Although the proposed mathematical model and solution approach consider a single mine, they can be easily extended to address the case of mineral value chains comprised of multiple mines. Both the model and the solution approach can also be adapted to address the more general mineral value chain, where the first destinations are not the last destinations. Future work will follow these directions. It would also be interesting to examine if the performance of ALNS is affected by the quality of the initial solution. To this end, the method used in the initialisation phase must be replaced by an alternative method to generate several different initial solutions. Finally, another important research direction could be to develop alternative improvement methods for the problem addressed in the paper and compare them to the ALNS method. Such methods could include a greedy randomised adaptive search procedure (GRASP) or other metaheuristics from the literature.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The work in this paper was funded by NSERC CRDPJ 411270-10, NSERC Discovery Grant 239019-06, and the industry members of the COSMO Stochastic Mine Planning Laboratory: AngloGold Ashanti, Barrick, BHP, De Beers, Kinross, Newmont, and Vale. This support is gratefully acknowledged.

ORCID

Amina Lamghari http://orcid.org/0000-0002-5632-2629
Roussos Dimitrakopoulos http://orcid.org/0000-0001-7329-2209

References

[1] R. Dimitrakopoulos Editor, *Advances in Applied Strategic Mine Planning*. Springer Nature Heidelberg (2018), 10.1007/978-3-319-69320-0

[2] A. Lamghari, *Mine planning and oil field development: A survey and research potentials*, Mathematical Geosciences. 49, 3 (2017), pp. 395–437. doi:10.1007/s11004-017-9676-z

[3] S. Ramazan and R. Dimitrakopoulos, *Stochastic Optimization Of Long-term Production Scheduling For Open Pit Mines With A New Integer Programming Formulation, in Orebody modelling and strategic mine planning, Spectrum series*, Cham: Springer International Publishing, 2005, pp. 353–360.
[4] J. Benndorf and R. Dimitrakopoulos, *Stochastic long-term production scheduling of iron ore deposits: Integrating joint multi-element geological uncertainty*, . Journal of Mining Science 49 (1) (2013), pp. 68–81. doi:10.1134/S1062739149010097.
[5] N.L. Mai, E. Topal, O. Erten, and B. Sommerville, *A new risk-based optimisation method for the iron ore production scheduling using stochastic integer programming*, Resour. Policy 62 (2019), pp. 571–579. doi:10.1016/j.resourpol.2018.11.004
[6] S. Ramazan and R. Dimitrakopoulos, *Production scheduling with uncertain supply: A new solution to the open pit mining problem*, Optim. Eng. 14 (2) (2013), pp. 361–380. doi:10.1007/s11081-012-9186-2.
[7] A. Khan and M.W.A. Asad, *A method for optimal cut-off grade policy in open pit mining operations under uncertain supply*, Resources Policy 60 (2019), pp. 178–184. doi:10.1016/j.resourpol.2018.12.003
[8] A. Leite and R. Dimitrakopoulos, *Stochastic optimisation model for open pit mine planning: Application and risk analysis at copper deposit*, . Mining Technology 116 (3) (2007), pp. 109–118. doi:10.1179/174328607X228848.
[9] F. Albor and R. Dimitrakopoulos, *Stochastic mine design optimisation based on simulated annealing: Pit limits, production schedules, multiple orebody scenarios and sensitivity analysis*, Min. Technol. 118 (2009), pp. 80–91.
[10] R. Goodfellow and R. Dimitrakopoulos, *Algorithmic integration of geological uncertainty in pushback designs for complex multiprocess open pit mines*, Mining Technology 122 (2) (2013), pp. 67–77. doi:10.1179/147490013X13639459465736.
[11] L. Montiel and R. Dimitrakopoulos,*Optimizing mining complexes with multiple processing and transportation alternatives: An uncertainty-based approach*, European Journal of Operational Research. 247, 1 (2015), pp. 166–178. doi:10.1016/j.ejor.2015.05.002
[12] A. Lamghari and R. Dimitrakopoulos, *A diversified Tabu search approach for the open-pit mine production scheduling problem with metal uncertainty*, European Journal of Operational Research. 222, 3 (2012), pp. 642–652. doi:10.1016/j.ejor.2012.05.029
[13] R. Senécal and R. Dimitrakopoulos, *Long-term mine production scheduling with multiple processing destinations under mineral supply uncertainty, based on multi-neighbourhood Tabu search*, Int. J. Min. Reclam. Environ (2019), pp. 1–14. doi:10.1080/17480930.2019.1595902.
[14] A. Lamghari, R. Dimitrakopoulos, and J.A. Ferland, *A variable neighbourhood descent algorithm for the open-pit mine production scheduling problem with metal uncertainty*, Journal of the Operational Research Society. 65, 9 (2014), pp. 1305–1314. doi:10.1057/jors.2013.81
[15] S.-O. Gilani and J. Sattarvand, *Integrating geological uncertainty in long-term open pit mine production planning by ant colony optimization*, Computers & Geosciences 87 (2016), pp. 31–40. doi:10.1016/j.cageo.2015.11.008
[16] M.E. Villalba Matamoros and M. Kumral, *Underground mine planning: Stope layout optimisation under grade uncertainty using genetic algorithms*, International Journal of Mining, Reclamation and Environment. 33, 5 (2019), pp. 353–370. doi:10.1080/17480930.2018.1486692
[17] R. Goodfellow and R. Dimitrakopoulos, *Global optimization of open pit mining complexes with uncertainty*, Applied Soft Computing 40 (2016), pp. 292–304. doi:10.1016/j.asoc.2015.11.038
[18] Z. Levinson and R. Dimitrakopoulos, *Simultaneous stochastic optimization of an open-pit gold mining complex with waste management*, Int. J. Min. Reclam. Environ (2019), pp. 1–15. doi:10.1080/17480930.2019.1621441.
[19] Z. Saliba and R. Dimitrakopoulos, *An application of simultaneous stochastic optimization of an open-pit mining complex with tailings management*, Int. J. Min. Reclam. Environ (2019), pp. 1–14. doi:10.1080/17480930.2019.1688954.
[20] M. de Freitas Silva, R. Dimitrakopoulos, and A. Lamghari, *Solving a large SIP model for production scheduling at a gold mine with multiple processing streams and uncertain geology*, Mining Technology. 124, 1 (2015), pp. 24–33. doi:10.1179/1743286314Y.0000000075
[21] M.W.A. Asad, R. Dimitrakopoulos, and J. Van Eldert, *Stochastic production phase design for an open pit mining complex with multiple processing streams*, Engineering Optimization. 46, 8 (2014), pp. 1139–1152. doi:10.1080/0305215X.2013.819094
[22] A. Lamghari and R. Dimitrakopoulos, *Network-flow based algorithms for scheduling production in multi-processor open-pit mines accounting for metal uncertainty*, European Journal of Operational Research. 250, 1 (2016), pp. 273–290. doi:10.1016/j.ejor.2015.08.051
[23] A. Paithankar, S. Chatterjee, R. Goodfellow, and M.W.A. Asad, *Simultaneous stochastic optimization of production sequence and dynamic cut-off grades in an open pit mining operation*, Resources Policy 66 (2020), pp. 178–184. doi:10.1016/j.resourpol.2020.101634
[24] A. Lamghari and R. Dimitrakopoulos, *Hyper-heuristic approaches for strategic mine planning under uncertainty*, Computers & Operations Research 115 (2020), pp. 1–18. doi:10.1016/j.cor.2018.11.010

[25] A.D. Ajaka, E. Lilford, and E. Topal, *Application of predictive data mining to create mine plan flexibility in the face of geological uncertainty*, Resources Policy 55 (2018), pp. 62–79. doi:10.1016/j.resourpol.2017.10.016
[26] S.C. Bouffard and P. Boggis, *Stochastic optimization of the Jansen potash production and logistics chain*, Mineral Processing and Extractive Metallurgy Review. 40, 3 (2019), pp. 207–217. doi:10.1080/08827508.2018.1528974
[27] S. Ropke and D. Pisinger, *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows*, Transportation Science 40 (4) (2006), pp. 455–472. doi:10.1287/trsc.1050.0135.
[28] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*, 2nd ed., *Springer*, New York, NY, 2011.
[29] R. Dimitrakopoulos and S. Ramazan, *Uncertainty based production scheduling in open pit mining*, SME Transactions 316 (2004), pp. 106–112.
[30] D. Pisinger and S. Ropke, *A general heuristic for vehicle routing problems, 34* Comput, in Oper. Res, 34 (8) (2007), pp. 2403–2435. https://doi.org/10.1016/j.cor.2005.09.012
[31] P. Shaw, *A new local search algorithm providing high quality solutions to vehicle routing problems*, Tech. Rep., Department of Computer Science, University of Strathclyde. Glasgow, UK (1997)
[32] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, and A.H. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys. 21 (6) (1953), pp. 1087–1092. doi:10.1063/1.1699114.
[33] E. Demir, T. Bektas, and G. Laporte, *An adaptive large neighborhood search heuristic for the pollution-routing problem, Eur*, J. Oper. Res. 223 (2) (2012), pp. 346–359. doi:10.1016/j.ejor.2012.06.044.
[34] A.A. Kovacs, S.N. Parragh, K.F. Doerner, and R.F. Hartl, *Adaptive large neighborhood search for service technician routing and scheduling problems, J*, Journal of Scheduling 15 (5) (2012), pp. 579–600. doi:10.1007/s10951-011-0246-9.
[35] R. Masson, F. Lehuede, and O. Peton, *An adaptive large neighborhood search for the pickup and delivery problem with transfers, Transport*, Sci 47 (2013), pp. 344–355.
[36] Y. Adulyasak, J.F. Cordeau, and R. Jans, *Optimization-based adaptive large neighborhood search for the production routing problem, Transport*, Sci 48 (2014), pp. 20–45.
[37] P. Goovaerts, *Geostatistics for Natural Resources Evaluation*, Oxford University Press, New York, 2017.
[38] M. Rossi and C. Deutsch, *Mineral Resource Estimation*, Springer, New York, 2014.
[39] J.J. Gomez-Hernandez and R.M. Srivastava, *One step at a time: The origins of sequential simulation and beyond*, Mathematical Geosciences 53 (2) (2021), pp. 193–209. doi:10.1007/s11004-021-09926-0.
[40] A. Boucher and R. Dimitrakopoulos, *Block Simulation of multiple correlated variables, Math*, Geosci 41 (2009), pp. 215–237.

Annex A. Destroy methods

The 14 destroy methods used to select the β blocks to be removed from the current solution, x, are described below. Unless otherwise specified, all destroy methods follow the general scheme summarised 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.

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 $\beta$ first blocks
  Add these blocks to the list $\mathcal{L}$
  return $\mathcal{L}$.
```

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.
- $\mathcal{P}_{ds}^t(x) = \sum_{i\in\mathcal{N}} w_{is}y_{ids}^t$: 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, the dependence on x will be omitted whenever there is no risk of ambiguity; that is, t_i will be used instead of $t_i(x)$ and so on.

A.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).

A.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.

A.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 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 $\mathcal{Z}_{it_i} - f(x)$.

A.4. Greedy picker (D4)

This method selects the costliest 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.

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

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

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 ($\sum_s \max\left(\in, \underline{E_{t_i}} - \left(M_s^{t_i} - w_{is}\right)\right)$) 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.

A.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 $\alpha_{is} = \sum\limits_{d \in \mathcal{D}} a_{ids}$ 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:

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

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).

A.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:

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

The term ($\sum\limits_{s \in \mathcal{S}} (\alpha_{is} - 1)$) 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 ($\sum\limits_{t = E_i, t \neq t_i}^{L_i} \sum\limits_{s \in \mathcal{S}} \alpha_{is}$) 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.

A.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 extracted 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 extracted in t. Recall that β blocks should be selected. We set $p_i = \left|\gamma_i - \frac{\beta}{\tau}\right|$ and among the blocks currently extracted in t, the block with the smallest value of p_i is selected, as well as its predecessors extracted in t. Ties are broken up randomly. This process is repeated for each of the τ periods.

A.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 extracted in the same period, the procedure to select the blocks is identical to the procedure described in the previous section (A.8).

A.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:

$$p_i = \begin{cases} \min(1, L_i -_i) \max\limits_{s\in n} \frac{n_s^{t_i}}{E^{t_i}} if \sum\limits_{t\in n} x_{it} = 1, \\ 0 \qquad\qquad\qquad otherwise. \end{cases}$$

The blocks are ranked in descending order ofp_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 $\sum\limits_{t\in\mathcal{T}} x_{it} = 1$ 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 $\sum\limits_{t\in\mathcal{T}} x_{it} = 0$) is also set to $p_i = 0$ to avoid selecting it, as such blocks are not extracted and thus have no influence on the mining capacity constraints (changing their periods does not affect the fourth term of the objective function (14)).

A.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:

$$p_i = \begin{cases} \max_{s\in S}\left\{\sigma_{is} \frac{n_{ds}^{t_i}}{F_d^{t_i}}\right\} \qquad if \sum\limits_{t\in T} x_{it} = 1, \\ 0 \qquad\qquad\qquad\qquad otherwise. \end{cases}$$

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 moved to 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.

A.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 $\in$ be a small positive value, and let C be a large positive value. Recall that δ_1 is the economic discount rate. The priority values are computed using the following formula, and the blocks are ranked in descending order of the priority values:

$$p_i = \begin{cases} \frac{1}{(1+\delta_1)^{t_i}} \sum\limits_{s\in n} \frac{v_{id^*s} - v_{ics}}{\max\left(\in, \underline{F_c^{t_i}} - \left(n_{cs}^{t_i} - w_{is}\right)\right)} if \sum\limits_{t\in n} x_{it} = 1, \\ -C \qquad\qquad\qquad otherwise. \end{cases}$$

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 ($\max\left(\in, \underline{F_c^{t_i}} - \left(\mathcal{P}_{cs}^{t_i} - w_{is}\right)\right)$)) 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).

A.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.

A.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, $\pi_{is} = 0 \quad s$ 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:

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

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.

Annex B. Repair methods

Referring to Algorithms 1 and 2, the blocks in the list $\mathcal{L}$, identified by the selected 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, and the remaining variables are 'free'. One of the seven methods described below 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 optimise the 'free' variables. The following notation is used. Given the solution to repair x^{-}, $\mathcal{B}^t = \{i \in \mathcal{N} : x_i^t = 1\}$ denotes the set of blocks extracted in period t. The set of blocks processed in destination d during period t under scenario s is denoted by $\Lambda_{ds}^t = \{i \in \mathcal{N} : y_{ids}^t = 1\}$.

B.1. Random repair (R1)

This method considers one block $i \in \mathcal{L}$ at a time, which is selected randomly, and sequentially chooses the period and the destinations in which i will be scheduled. This is done as follows: 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 extracted 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.

B.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

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

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

$g(\mathcal{B}^t)$ is used to evaluate the cost of extracting a block i in period t accounting for all blocks that are already extracted 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 d 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 extracting i in period t: $\Delta_1(i,t) = g(\mathcal{B}^t \cup^{\{i\}}) - g(\mathcal{B}^t)$. 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: $\Delta_2(i,d,s,t) = h(\Lambda_{ds}^t \cup^{\{i\}}) - h(\Lambda_{ds}^t)$ if i is admissible to d under scenario s, and $\Lambda_2(i,d,s,t)$ is set equal to a large negative value otherwise.

Let $d^*(i,s,t) = arg \max_{d \in \mathcal{D}} \Delta_2(i,d,s,t)$. Clearly, the maximum profit that one can expect if block i is extracted in period t is $\Delta(i,t) = -\Delta_1(i,t) + \frac{1}{S}\sum_{s\in\mathcal{S}} \Delta_2(i,d^*(i,s,t),s,t)$. Once all periods in $\mathcal{FP}(i)$ are considered, the period $t^* = argmax_{t\in\mathcal{FP}(i)}\Delta(i,t)$ is identified and block i is scheduled to be extracted in t^* (i.e. it is included in the set $\mathcal{B}^t$). Finally, for each scenario s, i is sent to destination $d^*(i,s,t)$ (i.e. it is included in the set $\Lambda^t_{d^*(i,s,t)s}$).

B.3. Capacity cautious repair (R3)

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

(1) $g(\mathcal{B}^t)$ and $h(\Lambda^t_{ds})$ 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.

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

$$h(\Lambda^t_{ds}) = \frac{\sum_{i\in\Lambda^t_{ds}} w_{is}}{\overline{F}^t_d}. \tag{32}$$

(2) Accordingly, $d^*(i,s,t) = arg \min_{d\in\mathcal{D}} \Delta_2(i,d,s,t)$ instead of $d^*(i,s,t) = arg \max_{d\in\mathcal{D}} \Delta_2(i,d,s,t)$ (i.e. the best destination for a block in a given scenario and period is the one having the smallest capacity utilisation).
(3) $\Delta(i,t)$is set equal to $\Delta_1(i,t) + \frac{1}{S}\sum_{s\in\mathcal{S}} \ _2(i,d^*(i,s,t),s,t)$ and $t^* = argmin_{t\in\mathcal{FP}(i)}\Delta(i,t)$.

B.4. MCFP repair (R4 and R5)

To repair the solution, this method combines the random repair heuristic (R1) and the MCFP heuristic described in Section 3.2. 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 optimise). 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.

B.5. MIP repair (R6 and R7)

This method is very similar to the previous one. All the extraction 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 extraction 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 maximise the net present economic value to be generated from processing the blocks extracted 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.