Les Cahiers du GERAD ISSN: 0711–2440

A Network-Flow Based Algorithm for Scheduling Production in Multi-Processor Open-Pit Mines Accounting for Metal Uncertainty

A. Lamghari
R. Dimitrakopoulos

G–2013–63

September 2013

A Network-Flow Based Algorithm for Scheduling Production in Multi-Processor Open-Pit Mines Accounting for Metal Uncertainty

Amina Lamghari

Roussos Dimitrakopoulos*

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

amina.lamghari@mail.mcgill.ca

roussos.dimitrakopoulos@mcgill.ca

* *and GERAD*

September 2013

Les Cahiers du GERAD

G–2013–63

Abstract: This paper proposes a heuristic approach based on network flow techniques to schedule the production in open-pit mines, while accounting for metal uncertainty and considering multiple destinations for the material mined. The method is tested on small and medium size orebodies with up to 63,424 mining blocks, and near-optimal solutions are obtained from a few minutes up to a few hours. This is an improvement of orders of magnitude when compared with other fully stochastic implementations.

Key Words: Production scheduling; open-pit mining; multiple processors, stockpiles; metal uncertainty; network-flow algorithms.

1 Introduction

Production scheduling of open-pit mining operations is a challenging and critical issue for mining companies, a key factor in determining returns on investments in the order of hundreds of millions of dollars. In scheduling mine production, the mineral deposit is represented as a three-dimensional array of blocks. Each block has a weight and a metal content interpolated using information obtained from drilling.

In practice, an open-pit mine has multiple processors operating simultaneously. Each processor has a limited capacity, involves specific recoveries and costs, and requires blocks with a grade above a specific threshold value (the grade being the proportion of metal to rock). A block is processed in the processor that gives the highest recovered economic value, defined as the revenue from selling the metal recovered minus the processing cost. Blocks that do not contain enough metal content to make them profitable when processed in any of the processors are discarded as waste. In addition, the discount factor makes the early periods more profitable, such that it is preferable to process the blocks with the highest grade as early as possible. However, the blocks with highest grade might be found at the bottom of the pit, and thus the overlying blocks (their predecessors) might have to be mined faster than they can be processed due to the limited capacity of the processors. Therefore, blocks with a grade sufficiently high to allow for profitable processing are stockpiled and saved for processing during later periods, while the higher grade blocks are processed as early as they can be, given the limited extraction capacity, so as to maximize the net present value (NPV) over the life of the mine. Any material sent to or taken from the stockpiles incurs additional costs, so material is stockpiled if and only if there is more ore mined in a given period than can be processed.

Decisions on block scheduling are thus subject to various types of constraints. The production schedule not only must respect the limits on extraction capacity (mining constraints), the capacity of the processors (processing constraints), and the availability at the stockpiles (stockpiling constraints) at each period of the life of the mine, but also must take into consideration the order in which blocks can be removed from the orebody to ensure that a block is not mined before any of its predecessors (slope constraints). Additionally, any block can be mined only once (reserve constraints). The problem is to determine which blocks to extract and when to extract them (mining sequence) in order to maximize the net present value (NPV) of the mine while respecting the various constraints.

A major complexity in the open-pit mine production scheduling problem (MPSP) is that the number of blocks is large, in general in the order of tens to hundreds of thousands, yielding a large-scale optimization problem. Another factor adding to the complexity of the mine scheduling problem is metal uncertainty, for the metal content of the blocks is not known precisely at the time decisions are made but is inferred from limited drilling information. Up to now, most methods developed to solve the MPSP either ignore the metal uncertainty issue or are not able to solve large-scale instances in which metal uncertainty is accounted for.

Indeed, the MPSP has been frequently studied since the 1960s. Different methods have been applied to solve the deterministic version of the problem, which assumes that all the problem parameters are well known. These methods can be classified into three categories: exact methods (Dagdelen and Johnson, 1986; Caccetta and Hill, 2003; Ramazan, 2007; Boland et al., 2009), heuristic and metaheuristic methods (Gershon, 1987; Denby and Schofield, 1994; Ferland et al., 2007; Chatterjee et al., 2010), and hybrid methods (Tolwinski and Underwood, 1996; Sevim and Lei, 1998; Moreno et al., 2010). However, the uncertain nature of the problem is ignored in the deterministic version of the MPSP, resulting in misleading assessments as it has been shown in the technical literature (Dowd, 1994; Dimitrakopoulos et al., 2002).

The stochastic version of the problem, which is more realistic and more relevant, has been considered by an increasing number of researchers in the last decade (Godoy, 2003; Menabde et al., 2004; Leite and Dimitrakopoulos, 2007; Ramazan and Dimitrakopoulos, 2005, 2012; Boland et al., 2008; Albor and Dimitrakopoulos, 2010; Goodfellow and Dimitrakopoulos, 2011). A review of stochastic approaches in the context of mine scheduling can be found in Dimitrakopoulos (2011). While all the papers cited above account for metal uncertainty in the mine scheduling process, they consider different variants of the stochastic version of the MPSP, in the sense that they consider different constraints and/or different objective functions. The variants studied in Goodfellow and Dimitrakopoulos (2011) and Ramazan and Dimitrakopoulos (2012) are the closest to the one addressed in this paper. Indeed, Goodfellow and Dimitrakopoulos (2011) consider

multiple destinations for the material mined, but they do not consider stockpiling as an option. To solve the problem with multiple processors, they extend the simulated annealing based approach proposed by Godoy (2003). Ramazan and Dimitrakopoulos (2012) do account for stockpiling. They consider a mine with a single processor and a single stockpile. Their problem formulation includes some constraints that we do not consider here, namely, lower bounds on mining, grade blending constraints, and constraints specifying targets for ore and metal production. They solve it using the mixed integer programming solver CPLEX. This solution approach is limited by its inability to solve instances of realistic size in reasonable amount of time.

In this paper, we propose a network-flow based algorithm, which can handle the different complexities of the problem studied herein (large size, metal uncertainty, multiple processors, stockpiles) and is flexible enough to incorporate other constraints. To generate the initial solution to be improved by this algorithm, we consider two different alternatives. Both are based on a decomposition approach separating the problem into a series of sub-problems, each associated with one period. They differ in the method used to solve the sub-problems. In the first alternative, the sub-problems are solved exactly, while in the second one, the sub-problems are solved approximately with a greedy heuristic. The algorithm used to generate the initial solution is an extension of the work developed in Lamghari and Dimitrakopoulos (2011) to deal with the deterministic version of the MPSP. The improvement algorithm can be seen as a very large-scale neighborhood search method (VLSN), which uses network flow techniques to efficiently search for improving solutions in a very large neighborhood. A survey on VLSN algorithms can be found in Ahuja et al. (2002).

We provide numerical results allowing us to evaluate and compare the performance of the two variants of the proposed solution method, specified according to the process used for generating the initial solution. These results indicate that both variants generate very good solutions in reasonable computational times, but the first variant, where the sub-problems are solved exactly, in general outperforms the second one in terms of solution quality, but at the expense of higher computational times.

The remainder of the paper is organized as follows: In Section 2, the approach used to deal with metal uncertainty is outlined, and a mathematical formulation of the problem is introduced. The following section presents the methods used to generate the initial solution. Section 4 describes in detail the network-flow based algorithm used to improve the initial solution. Computational results are reported and discussed in Section 5. Finally, conclusions are drawn in Section 6.

2 Notation and Mathematical Formulation

Referring to the description given in the previous section, we deal with a decision problem where the mining sequence (or the schedule) must be determined before the metal content of the blocks is known exactly. However, we have some probabilistic information about the metal content. More specifically, for each block, there is a finite number of potential outcomes or scenarios for the grade (proportion of metal to rock), each having an associate probability of occurrence. The notation used to formulate the mathematical programming model is presented below. First are listed the indices and parameters. Second the variables are presented.

- N: the number of blocks considered for scheduling.
- i: block index, $i = 1, \ldots, N$.
- T: the number of periods over which blocks are being scheduled (horizon).
- t: period index, $t = 1, \ldots, T$.
- P: the number of processors. Note that since a stockpile is associated with each processor, the number of stockpiles is equal to the number of processors.
- p: processor index, $p = 1, \ldots, P$.
- S: the number of scenarios used to model metal uncertainty.
- s: scenario index, $s = 1, \ldots, S$.
- π_s: probability that scenario s occurs, with $\sum_{s=1}^{S} \pi_s = 1$.
- $Pred(i) \subseteq \{1, \ldots, N\}$: the set of predecessors of block i; i.e., blocks that have to be removed to have access to block i.

- w_i: the weight of block i in tonnes.
- W^t: maximum amount of material (waste and ore) that can be mined during period t (mining capacity).
- $\bar{c}$: undiscounted cost of mining a tonne of rock.
- c_p: undiscounted cost of processing a tonne of ore in processor p.
- α_p: recovery of metal if the ore is processed in processor p.
- ξ: undiscounted selling cost (includes the refining and marketing costs).
- μ: undiscounted metal price.
- g_{is}: the grade of block i under scenario s.
- $G_p = \frac{c_p}{\alpha_p(\mu-\xi)}$: marginal cutoff grade for processor p. In other words, G_p represents the grade of ore below which processing is no longer profitable in processor p. Without loss of generality, we assume that the processors are indexed in such a way that $G_1 < G_2 < G_3 < \ldots < G_P$.
- θ_{ips}: a parameter indicating the most profitable processor p to which block i can be sent if scenario s occurs

$$\theta_{ips} = \begin{cases} 1 & \text{if } g_{is} \in]G_p, G_{p+1}] \text{ (if } p = P, \text{ then } G_{P+1} \text{ is a large constant)}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that this implies that $\forall i, s$, $\sum_{p=1}^{P} \theta_{ips} \in \{0, 1\}$; i.e., for each block i and each scenario s, either i is a waste block (if $g_{is} \leq G_1$) or there is only one processor p where i can be processed (processor p such that $g_{is} \in]G_p, G_{p+1}]$).

- Θ_p^t: maximum amount of ore that can be processed in processor p during period t (processing capacity of p).
- $r_{is} = \sum_{p=1}^{P} \theta_{ips} w_i [g_{is}\alpha_p(\mu - \xi) - c_p]$: undiscounted revenue of an already mined block i if sent directly to processing, and if scenario s occurs.

Note that since $\sum_{p=1}^{P} \theta_{ips} \in \{0, 1\}$, this means that $r_{is} = 0$ if i is a waste block under scenario s; otherwise, r_{is} is equal to the undiscounted revenue to be generated if i is processed in p, p being the processor such that $\theta_{ips} = 1$.

- c_p^+: undiscounted cost of sending a tonne of ore to the stockpile associated with processor p (transportation cost). We assume that when a block arrives at the stockpile, it is mixed with the other material already there. The cost of this operation is included in c_p^+.
- c_p^-: undiscounted cost of taking a tonne of ore from the stockpile associated with processor p (transportation cost plus loading cost).
- $\tilde{G}_{ps} = \frac{\sum_{i:\, \theta_{ips}=1} w_i g_{is}}{\sum_{i:\, \theta_{ips}=1} w_i}$: an approximation of the average grade in the stockpile associated with processor p, if scenario s occurs. Note that assuming that the average grade in the stockpiles is known is a simplifying assumption to bypass the nonlinearity of the problem that stems from the use of the stockpiles. Indeed, without this assumption, and assuming that the material in the stockpiles is mixed, the average grade in the stockpiles at a given period is the average grade of the blocks extracted and sent to the stockpiles in the previous periods. But, blocks to extract are decision variables of the problem and the average grade in the stockpiles is used to compute the revenue to be generated from processing ore taken from the stockpiles, a coefficient of the objective function. This interaction gives rise to a non-linear term in the objective function.
- $\tilde{r}_{ps} = \tilde{G}_{ps}\alpha_p(\mu - \xi) - c_p$: an approximation of the revenue to be generated if a tonne of ore in the stockpile associated with p is processed, and if scenario s occurs.
- d: the discount rate per period.

The variables used to formulate the problem are as follows:

- A binary variable is associated with each block i for each period t:

$$x_i^t = \begin{cases} 1 & \text{if block } i \text{ is mined by period } t, \\ 0 & \text{otherwise.} \end{cases}$$

This means that if block i is mined in period τ, then $x_i^t = 0$ for all $t = 1, \ldots, \tau - 1$ and $x_i^t = 1$ for all $t = \tau, \ldots, T$. If i is not mined during the horizon, then $x_i^t = 0$ for all $t = 1, \ldots, T$. To simplify the notation in the rest of this section, we introduce a set of N dummy decision variables $x_i^0\,(i = 1, \ldots, N)$, each having a fixed value equal to 0.

- In modeling the *processing constraints*, we use the variables y_{ps}^{t+} to denote the surplus in the amount of ore mined during period t that can be processed in p if scenario s occurs (i.e., the amount to send to the stockpile).
- y_{ps}^{t-} measures the amount of ore to take at period t from the stockpile associated with processor p, if scenario s occurs.
- Finally, the continuous variables y_{ps}^t denote the amount of ore in the stockpile associated with processor p at the end of period t under scenario s. We assume that the stockpile is empty at the beginning of the first period.

The proposed model is given below:

$$\begin{aligned} \max & -\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} w_i \bar{c}(x_i^t - x_i^{t-1}) + \sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} \sum_{s=1}^{S} \pi_s r_{is}(x_i^t - x_i^{t-1}) \\ & - \sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s (\tilde{r}_{ps} + c_p^+) y_{ps}^{t+} + \sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s (\tilde{r}_{ps} - c_p^-) y_{ps}^{t-} \end{aligned} \tag{1}$$

(M) **Subject to**

$$x_i^{t-1} \leq x_i^t \qquad \forall i, t \tag{2}$$

$$x_i^t \leq x_j^t \qquad \forall i, j \in pred(i), t \tag{3}$$

$$\sum_{i=1}^{N} w_i (x_i^t - x_i^{t-1}) \leq W^t \qquad \forall t \tag{4}$$

$$\sum_{i=1}^{N} \theta_{ips} w_i (x_i^t - x_i^{t-1}) - y_{ps}^{t+} + y_{ps}^{t-} \leq \Theta_p^t \qquad \forall t, p, s \tag{5}$$

$$y_{ps}^{t-1} + y_{ps}^{t+} - y_{ps}^{t-} = y_{ps}^t \qquad \forall t, p, s \tag{6}$$

$$x_i^t = 0 \text{ or } 1 \qquad \forall i, t \tag{7}$$

$$x_i^0 = 0 \qquad \forall i \tag{8}$$

$$y_{ps}^{t+}, y_{ps}^{t-}, y_{ps}^t \geq 0 \qquad \forall t, p, s \tag{9}$$

$$y_{ps}^0 = 0 \qquad \forall p, s. \tag{10}$$

The objective function (1) maximizes the NPV of the mine. It includes four different terms:

1. The first term ($\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} w_i \bar{c}(x_i^t - x_i^{t-1})$) evaluates the total discounted cost of the extraction (mining operations).
2. The second term ($\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} \sum_{s=1}^{S} \pi_s r_{is}(x_i^t - x_i^{t-1})$) gives the total expected discounted revenue generated if all the ore mined is sent directly for processing during the period in which it is mined. Note that waste blocks do not contribute to this term because, as mentioned earlier, if i is a waste block under scenario s, then $r_{is} = 0$.

3. The third term ($\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s(\tilde{r}_{ps} + c_p^+) y_{ps}^{t+}$) gives the total expected discounted cost of sending ore to the stockpiles, including both the revenue lost because the ore is not processed in the period where it is available and the cost of transportation to the stockpile.
4. The fourth term ($\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s(\tilde{r}_{ps} - c_p^-) y_{ps}^{t-}$) represents the total expected discounted net revenue to be generated from processing ore taken from the stockpiles (revenue minus loading and transportation costs).

Constraints (2)–(4) are scenario-independent. Constraints (2) guarantee that each block i is mined at most once during the horizon (*reserve constraints*). The mining precedence (*slope constraints*) is enforced by constraints (3). Constraints (4) impose an upper bound W^t on the amount of material (waste and ore) mined during each period t (*mining constraints*).

Constraints (5) are related to the requirements on the processing levels (*processing constraints*). They are scenario-dependent for the following reason: the grade of a block determines whether a block can be processed in a given processor or not. But, the grade of a block may differ from one scenario to another. Therefore, each scenario may have a different impact regarding the amount of ore available for processing. Constraints (5) stipulate that for each scenario s and each processor p, if the total weight of ore blocks mined during any period t is greater than the processing capacity at that period, Θ_p^t, then the surplus y_{ps}^{t+} is sent to the stockpile associated with the processor p, inducing a penalty cost equal to $\frac{(\tilde{r}_{ps}+c_p^+)}{(1+d)^t} y_{ps}^{t+}$. If it is smaller than Θ_p^t and there is material in the stockpile, then an amount equal to y_{ps}^{t-} (maximum possible such that neither the capacity of the processor nor the amount available in the stockpile is exceeded) is taken from the stockpile and added to feed the processor, generating a net profit equal to $\frac{(\tilde{r}_{ps}-c_p^-)}{(1+d)^t} y_{ps}^{t-}$.

Finally, constraints (6) balance the flow at each stockpile (*stockpiling constraints*). Recall that a stockpile is associated with each processor p, and it contains ore that can be processed in p. Therefore, *stockpiling constraints* are also scenario-dependent. They ensure that for each scenario s, at the end of any period t, the amount of ore in the stockpile associated with each processor p is equal to the amount that was in the stockpile at the end of the previous period $(t-1)$ plus the amount added to the stockpile during t minus the amount taken from the stockpile during t (i.e., the amount sent from the stockpile for processing, if any). The initial amount in the stockpile, y_{ps}^0, is assumed to be equal to 0. Note that it is not necessary to add constraints to ensure that the amount taken from any stockpile should not exceed the amount available in this stockpile. Indeed, in an optimal solution, during any period t, we will not both take ore from the stockpile and send ore to it since these operations induce costs (see objective function (1)). Hence, at most one of the 2 variables y_{ps}^{t+} and y_{ps}^{t-} will take a positive value. Assume that $y_{ps}^{t-} > 0$ (i.e., under scenario s, the amount of ore mined during period t that can be processed in p is less than Θ_p^t, the capacity of processor p during that period), then $y_{ps}^{t+} = 0$ (i.e., no ore will be sent to the stockpile associated with p). Constraints (6) imply then that

$$y_{ps}^{t-1} - y_{ps}^{t-} = y_{ps}^{t} \geq 0$$

and consequently, $y_{ps}^{t-1} \geq y_{ps}^{t-}$.

Note that the proposed model can be seen as the deterministic equivalent of a two-stage stochastic integer programming model (Birge and Louveaux, 1997). The variables x_i^t specifying the mining sequence are the first-stage decision variables, and the variables y_{ps}^{t+} and y_{ps}^{t-} measuring respectively the amount sent to and taken from the stockpiles are the second-stage decision variables. The costs of sending the surplus of ore to the stockpiles and the profit generated from processing ore taken from the stockpiles will obviously influence the choice of the mining sequence. Thus, first-stage decisions are taken in the presence of uncertainty, yet they are chosen by taking their future effects into account.

Note also that if we eliminate the *mining*, the *processing*, and the *stockpiling constraints*, and if the scheduling horizon reduces to a single period, then the model reduces to the classical maximum closure problem. This problem is reducible to the minimum-cut problem (Picard, 1976), and thus it can be solved efficiently using any of the known polynomial maximum-flow algorithms. However, as Bienstock and Zuckerberg (2010) note, "it can be shown by reduction from max clique that adding a single cardinality constraint to a maximum closure problem is enough to make it NP-hard." Because the MPSP is more complex than a

constrained maximum closure problem, and as real-world MPSP instances are very large, having typically tens to hundreds of thousands blocks, it is most likely not appropriate to solve these large-scale realistic instances using an exact method. Instead, we propose a heuristic where an initial feasible solution is first obtained and then improved with a network-flow based algorithm. The methods used to generate the initial solution as well as the network-flow improvement algorithm are described in the following sections.

3 Initial Solution

We propose to generate the initial solution using two different heuristics based on a decomposition approach where the global problem is divided into smaller sub-problems, each associated with a period t ($t = 1, \ldots, T$). The sub-problems are solved sequentially in increasing order of t, and their solutions are combined to generate the initial solution. The heuristics differ in the method used to deal with the sub-problems. In the first one, the sub-problems are solved exactly, while in the second one, the sub-problems are solved approximately with a greedy heuristic. Note that since the sub-problems are smaller than the global problem, an exact method can be applied to solve them.

3.1 Sub-problem formulation

A solution of the sub-problem associated with period t is characterized by the following three decision variables: the blocks to mine, X, the amount of ore to send to the stockpiles, Y^+, and the amount of ore to take from the stockpiles, Y^-. Y^+ is dependent on X. Y^- depends on X as well as on the amount on hand in the stockpiles at the beginning of t. The latter is known when the sub-problems are solved sequentially. Therefore, the main decision consists of determining a set of blocks B^t to be mined in period t. Let us denote by $\mathrm{R}^t = \{block\ i : x_i^t = 0\}$ the set of blocks not mined yet (if $t = 1$, then $|\mathrm{R}^t| = N$; otherwise, $|\mathrm{R}^t| = N - |\bigcup_{\tau<t} \mathrm{B}^\tau|$). In order to satisfy the *reserve constraints*, the blocks to be included in B^t should be selected from R^t. Logical implications of the constraints are used to reduce the size of the set of candidate blocks, R^t, and make the sub-problems easier to solve. To be more specific, consider any block $i \in \mathrm{R}^t$. On the one hand, the *slope constraints* require that to include i in B^t, we must also include all blocks $j \in \mathrm{N}_i = Pred(i) \cap \mathrm{R}^t$ (the set of blocks that are predecessors of i and not mined yet). On the other hand, i should not be included in B^t if $w_i + \sum_{j\in \mathrm{N}_i} w_j > W^t$ because this would lead to violation of the *mining constraints*. Hence, we eliminate from R^t any block such that $w_i + \sum_{j\in \mathrm{N}_i} w_j > W^t$. The sub-problem associated with period t can then be summarized as follows:

$$\max -\sum_{i\in\mathcal{R}^t} w_i \bar{c} x_i + \sum_{i\in\mathcal{R}^t}\sum_{s=1}^{S} \pi_s r_{is} x_i - \sum_{p=1}^{P}\sum_{s=1}^{S} \pi_s(\tilde{r}_{ps} + c_p^+) y_{ps}^+ + \sum_{p=1}^{P}\sum_{s=1}^{S} \pi_s(\tilde{r}_{ps} - c_p^-) y_{ps}^- \tag{11}$$

(SP^t) **Subject to**

$$x_i \le x_j \qquad \forall i \in \mathcal{R}^t, j \in Pred(i) \cap \mathcal{R}^t \tag{12}$$

$$\sum_{i\in\mathcal{R}^t} w_i x_i \le W^t \tag{13}$$

$$\sum_{i\in\mathcal{R}^t} \theta_{ips} w_i x_i - y_{ps}^+ + y_{ps}^- \le \Theta_p^t \qquad \forall p, s \tag{14}$$

$$y_{ps}^+ \le \mathcal{S}_{ps} \qquad \forall p, s \tag{15}$$

$$x_i = 0 \text{ or } 1 \qquad \forall i \in \mathcal{R}^t \tag{16}$$

$$y_{ps}^+, y_{ps}^- \ge 0 \qquad \forall p, s \tag{17}$$

where:

- $x_i = \begin{cases} 1 & \text{if block } i \text{ is included in the set } \mathrm{B}^t, \\ 0 & \text{otherwise.} \end{cases}$

- y_{ps}^{+} and y_{ps}^{-} are continuous variables indicating respectively the amount sent to and taken from the stockpile associated with processor p if scenario s occurs.
- S_{ps}: a constant representing the content of the stockpile associated with processor p at the beginning of the current period (t) if scenario s occurs. The value of S_{ps} is initially set equal to 0 (when solving the first sub-problem (SP^1)). It is updated each time a sub-problem is solved as follows: Let the optimal solution to (SP^t) be (x_i^*, y_{ps}^{+*}, y_{ps}^{-*}). Then $\mathrm{S}_{ps} := \mathrm{S}_{ps} + y_{ps}^{+*} - y_{ps}^{-*}$. Constraints (15) guarantee then that for each scenario s, the amount of ore taken from the stockpile associated with any processor p does not exceed the amount available in this stockpile. The rest of the formulation is self-explanatory given the previous discussion in Section 2. Note that we omit the discount factor $\frac{1}{(1+d)^t}$ in the objective function (11) because the same factor appears in all of the four terms and thus does not affect the optimal solution.

3.2 Sub-problem solution methods

Two different methods are introduced to solve the sub-problems. In the first method, the formulation (11)–(17) is solved using the branch-and-cut algorithm (BC) implemented in the mixed integer programming solver CPLEX.

The second method is a sequential greedy heuristic procedure (GH) where at each iteration we try to include in the set B^t an inverted cone formed by a "base" block in R^t and all its predecessors not mined yet. Let us analyze a typical iteration.

First, consider the case where $t \leq T-1$; i.e., when the period we are dealing with is not the last period of the horizon. Let $\mathrm{V}^t = W^t - \sum_{b\in \mathrm{B}^t} w_b$ be the residual mining capacity and $f(\mathrm{B}^t)$ be the value of the objective function (11) at the current iteration. For each block $i \in \mathrm{R}^t$, we denote:

- $\Upsilon_i = \{i\} \cup \{j \,:\, j \in \mathrm{N}_i\}$ the set formed by block i and all its unmined predecessors (i.e., the inverted cone whose base is i).
- $\beta_i = \sum_{v\in\Upsilon_i} w_v$: the total weight of blocks in Υ_i.
- $\delta_i = f(\mathrm{B}^t \cup \{\Upsilon_i\}) - f(\mathrm{B}^t)$: the modification induced on the objective function (11) if blocks in Υ_i are added to B^t.
- $\rho_i = \sum_{v\in\Upsilon_i}(-w_v\bar{c} + \sum_{s=1}^{S}\pi_s r_{vs})$: the total expected profit of blocks in Υ_i if all ore blocks in Υ_i are sent directly for processing once mined.

Consider the set $\mathrm{A} = \left\{ i \in \mathrm{R}^t \,:\, \alpha_i \leq \mathrm{V}^t, \delta_i \geq 0, \text{ and } \frac{\delta_i}{(1+d)^t} \geq \frac{\rho_i}{(1+d)^{t+1}} \right\}$ of blocks $i \in \mathrm{R}^t$ whose weight plus the weight of their predecessors not mined yet does not exceed the residual mining capacity, which, if mined with their predecessors, leads to an improvement of the objective function value, and which are, along with their predecessors, more profitable to mine in the current period than to leave for the next period. Clearly, cones having blocks in A as their base can be added to B^t while satisfying the *slope constraints* and the *mining constraints* and improving the value of the current solution. We discard blocks with $\frac{\delta_i}{(1+d)^t} < \frac{\rho_i}{(1+d)^{t+1}}$, for it's better to leave them until a later period because they should generate more profit if they are. Select the block $i^* \in \mathrm{A}$ maximizing the value of δ_i. Remove all blocks $v \in \Upsilon_{i^*}$ from R^t, add them to B^t, and update V^t, $f(\mathrm{B}^t)$, and A. The process terminates when the set A is empty.

Now let us consider the case where $t = T$. The same process described above is applied except that the condition $\frac{\delta_i}{(1+d)^t} \geq \frac{\rho_i}{(1+d)^{t+1}}$ is dropped when identifying the set of candidate blocks A since we are dealing with the last period of the horizon, and therefore, no revenue can be generated if the blocks are left behind. Blocks left unselected at the end of the T^{th} iteration are included in B^{T+1}, $(T+1)$ being a fictitious period.

Note that choosing blocks along with their predecessors allows a look ahead feature generating better solutions than the myopic approach of choosing blocks one by one. Comparing $\frac{\delta_i}{(1+d)^t}$ to $\frac{\rho_i}{(1+d)^{t+1}}$ also allows a long-term vision of the scheduling process.

4 Network-flow Improvement Algorithm

Let x^0 be the initial schedule generated using one of the heuristics described in the previous section. We wish to obtain a schedule having a higher NPV than x^0. One way to achieve this is to consider blocks having a negative contribution to the objective function (1) that we aim to maximize, and try to delay their extraction. Another way would be to advance the extraction of blocks having a positive contribution. To identify such blocks (to be extracted later or earlier), we solve a series of longest path problems (LPP). Each problem is defined on a graph $G = (V, E)$, V and E being the set of nodes and arcs, respectively.

In what follows, we first introduce the notation to be used throughout this section and define the graph structure. Then, we give the mathematical model (LPP) to be solved, followed by its explanation and how the new schedule can be obtained from the optimal solution of (LPP). Finally, we outline the algorithm used to improve the initial solution.

4.1 Notation and graph structure

Recall that the scheduling horizon consists of T periods. We consider an additional fictitious period $(T + 1)$ to identify blocks that are not mined. To simplify the presentation, we will refer to a block not mined during the horizon as a block mined in period $(T + 1)$. We associate with this fictitious period an infinite mining capacity ($W^{T+1} = \infty$), and obviously, any block mined in $(T+1)$ neither incurs costs nor generates revenue. Recall also that B^t denotes the set of blocks mined in period t ($t = 1, \ldots, T + 1$). We add to each set B^t a fictitious block f^t, whose weight and revenue for each scenario are equal to zero (i.e., $w_{f^t} = 0$ and $r_{f^t s} = 0 \quad \forall s$). Moreover, such block has no predecessors (i.e., $Pred(f^t) = \emptyset$) nor does it have successors. We will explain later the motivation behind adding the fictitious blocks, f^t.

Let us first describe how we construct the graph $G = (V, E)$ in the case where we delay the extraction of some blocks having a negative contribution to the objective function (1) before moving to the case where we advance the extraction of some blocks having a positive contribution. The latter case is a straightforward extension of the first case.

First, consider a period $t \neq (T + 1)$. Let i be a block in B^t, and let Γ_i be the set including i and its successors mined in the same period. Denote by $p(\Gamma_i) = \sum_{\gamma \in \Gamma_i}(-w_\gamma \bar{c} + \sum_{s=1}^{S} \pi_s r_{\gamma s})$ the total undiscounted expected profit of blocks in Γ_i. Define:

- $\Omega^t = \{i \in \mathrm{B}^t : p(\Gamma_i) \leq 0\}$
- $V^t = \{i \in \Omega^t : \exists j \in \Omega^{t+1}$ such that $\sum_{b \in \mathrm{B}^{t+1}} w_b - \sum_{\gamma \in \Gamma_j} w_\gamma + \sum_{\gamma \in \Gamma_i} w_\gamma \leq W^{t+1}\}$.

Basically, this means that any block $i \in V^t$ is a block currently mined in t that could be extracted later (moved to the next period $(t + 1)$). This move can be feasibly done; i.e., without violating the *slope* and the *mining constraints* in both periods t and $(t + 1)$ involved in the move. Moreover, such move is more likely to improve the value of the objective function (1) (NPV of the schedule) because $p(\Gamma_i) \leq 0$. The reasons why the constraints remain satisfied are as follows:

- The mining capacity at period t cannot be exceeded because we remove blocks from that period. Nor is the mining capacity in period $(t + 1)$ exceeded because it is possible to delay the extraction of some blocks currently mined in $(t + 1)$ to make room for the new blocks (i and its successors), if necessary.
- The *slope constraints* remain satisfied because all blocks involved in the move are moved along with their successors mined in the same period.

Now let us consider the fictitious period $(T + 1)$. We define $V^{T+1} = \{f^{T+1}\}$. The reason behind that is that the blocks mined in $(T + 1)$ are in fact those that are not mined, so we cannot delay their extraction. We now explain how to construct the graph $G = (V, E)$. The set of nodes V is obtained by defining a node for each block $i \in V^t$, $t = 1, \ldots, T + 1$. In what follows, we will refer indifferently to a block as a node or a block. We add to V a source σ and a sink τ. The set of arcs E consists of all possible connections between two nodes $i^t \in V^t$ and $i^{t+1} \in V^{t+1}$, $t = 1, \ldots, T$. We add to E arcs connecting the source σ to nodes in V^1 as well as arcs connecting nodes in V^{T+1} to the sink, τ.

Consider a path from the source to the sink ($\sigma \to i^1 \to i^2, \ldots, i^T \to i^{T+1} \to \tau$), where $i^t \in V^t \quad \forall t = 1, \ldots, T+1$. Recall that Γ_i denotes the set including i and its successors mined in the same period. Simultaneously removing blocks in all sets Γ_i^t from t, their current period, and adding them to the next period $(t+1)$ will yield a new solution of the problem. Each path thus identifies a new solution of the problem. Moreover, by definition of the sets V^t, there exists at least one feasible path (i.e., yielding a feasible solution). Our objective is to find a feasible path that improves the value of the objective function (1). We need thus to differentiate between feasible and infeasible paths and to evaluate the impact on the objective function of moving blocks identified by a feasible path. To this end, we associate with each path a profit. More specifically, each arc $(i,j) \in E$ has a profit p_{ij}, and in order to obtain the profit associated with a path, we simply sum the profit of the arcs in the path. Before explaining how we define the p_{ij}, note that the definition of a path requires moving blocks from each period of the horizon. But, it might not be profitable to delay the extraction of any block mined in a certain period t. So, we should allow the possibility of not moving blocks from the periods. That's the reason why we add the fictitious blocks $f^1, \ldots, f^{T+1}$: a path including the node f^t signifies that no block is moved from period t.

We define the profit p_{ij} of an arc $(i,j) \in E$ as follows:

- Arcs connecting nodes in V^t to nodes in V^{t+1} $(1 \leq t \leq T)$: Each arc (i,j) of this type signifies that block j and its successors (blocks in Γ_j), currently mined in period $(t+1)$, will be left behind to be mined in period $(t+2)$. In their place, block i and its successors (blocks in Γ_i), currently mined in t, will be moved to $(t+1)$. If this move results in a violation of the mining capacity at $(t+1)$, then we set p_{ij} equal to a large negative value to penalize any path including the arc (i,j) (i.e., infeasible paths). Otherwise, we define the profit p_{ij} to be the change in the first three terms of the objective function (1) due to simultaneously adding blocks in Γ_i to and removing blocks in Γ_j from $(t+1)$. Note that when computing p_{ij} we do not include the change due to adding j and its successors to period $(t+2)$. That's because if we include it, then when we sum the profit of the arcs in a path (to obtain the profit associated with the path), we will be double counting the change in period $(t+2)$ (p_{jk}, the profit associated with the arc following (i,j) in the path, includes this change). Note also that we do not account for the fourth term of the objective function (1), which is the revenue generated by taking ore from the stockpiles. The reason is that this revenue cannot be evaluated independently for each period. Since it depends on the amount available in the stockpile at the beginning of $(t+1)$, it is affected not only by the blocks added to and removed from $(t+1)$, but also by the moves made in periods $1, \ldots, t$ (all arcs preceding (i,j)).
- Arcs connecting the source σ to nodes j in V^1: Each arc (σ, j) signifies that blocks in Γ_j will be extracted in period 2 rather than in period 1. Obviously, the mining capacity at period 1 is satisfied if we perform such a move. Therefore, the profit $p_{\sigma j}$ is defined as the change in the first three terms of the objective function (1) resulting from removing blocks in Γ_j from the first period.
- Arcs connecting nodes in V^{T+1} to the sink τ: As mentioned before, V^{T+1} comprises a single node associated with the fictitious block f^{T+1}. Hence, we define $p_{f^{T+1}\tau} = 0$.

The discussion above concerns the case where we want to delay the extraction of some blocks in an attempt to improve the current solution. Now let us consider the case where we want to identify blocks to extract earlier. The way the graph is constructed is very similar to what has been described above, with the following few differences:

i. Γ_i is defined as the set including i and its predecessors mined in the same period rather than i and its successors.
ii. $\Omega^t = \{i \in \mathrm{B}^t : p(\Gamma_i) \geq 0\}$ instead of $\{i \in \mathrm{B}^t : p(\Gamma_i) \leq 0\}$.
iii. $V^1 = \{f^1\}$ because blocks mined in the first period cannot be extracted earlier.
iv. For the other periods $t = 2, \ldots, T+1$ we define

$$V^t = \left\{ i \in \Omega^t : \exists j \in \Omega^{t-1} \text{ such that } \sum_{b \in \mathrm{B}^{t-1}} w_b - \sum_{\gamma \in \Gamma_j} w_\gamma + \sum_{\gamma \in \Gamma_i} w_\gamma \leq W^{t-1} \right\}$$

which guarantees that it is possible to extract blocks $i \in V^t$ earlier, in period $(t-1)$ rather than in period t, while maintaining the satisfaction of the *slope* and the *mining* constraints in both periods.

v. While the set of nodes is defined in a similar manner as above (a node is associated with each block $i \in V^t$, $t = 1, \ldots, T+1$), the set of arcs E is defined slightly differently. It now consists of all possible connections between two nodes $i \in V^t$ and $j \in V^{t-1}$, $t = 2, \ldots, T+1$. We add to E arcs connecting the source σ to nodes in V^{T+1} and arcs connecting nodes in V^1 to the sink, τ.

vi. The profit p_{ij} associated with an arc $(i,j) \in V^t \times V^{t-1}$ $(t = 2, \ldots, T+1)$ is the change in the three terms of the objective function (1) due to simultaneously adding i and its predecessors to and removing j and its predecessors from period $(t-1)$. The profit associated with the remaining arcs is equal to 0.

4.2 Longest path problem

Once the graph $G = (V, E)$ is constructed, we solve a longest path problem defined on G to find the path that most improves the value of the three terms of the objective function (1). We will use the following additional notation to formulate this problem. We associate a binary variable z_{ij} with each arc $(i,j) \in E$. This variable takes value 1 if arc (i,j) is present in some valid path, and takes value 0 otherwise. For each node $i \in V$, we denote by $I(i)$ and $O(i)$ the set of incoming and outgoing arcs at the node i, respectively. We now give the mathematical model:

$$\max \sum_{(i,j)\in E} p_{ij} z_{ij} \tag{18}$$

(LPP) **Subject to**

$$\sum_{(\sigma,j)\in O(\sigma)} z_{\sigma j} = 1 \tag{19}$$

$$\sum_{(i,j)\in O(i)} z_{ij} - \sum_{(j,i)\in I(i)} z_{ji} = 0 \qquad \forall i \neq \sigma, \tau \tag{20}$$

$$\sum_{(i,\tau)\in I(\tau)} z_{i\tau} = 1 \tag{21}$$

$$z_{ij} = 0 \text{ or } 1 \qquad \forall (i,j) \in E \tag{22}$$

Constraints (19)–(21) are the network flow conservation constraints. If we are delaying the extraction of the blocks, they ensure that, for each period t $(t = 1, \ldots, T)$, either the fictitious block f^t or a block mined in t and all its successors mined in the same period are removed from t and added to the next period $(t+1)$. Conversely, if we are advancing the extraction of the blocks, they ensure that, for each period t $(t = 2, \ldots, T+1)$, either the fictitious block f^t or a block mined in t and all its predecessors mined in the same period are removed from t and added to the previous period $(t-1)$. The objective function (18) maximizes the total change in the first three terms of the objective function (1) resulting from such moves.

Let the optimal solution of (LPP) be $z^* = (z^*_{ij})$. Consider first the case where we try to delay the extraction of the blocks to improve the current solution x. For each arc $(i,j) \in V^t \times V^{t+1}$ $(t = 1, \ldots, T)$, if $z^*_{ij} = 1$ and $i \neq f^t$, then we move i and all its successors mined in the same period to the next period $(t+1)$. Now let us consider the other case where we try to advance the extraction of the blocks. For each arc $(i,j) \in V^t \times V^{t-1}$ $(t = 2, \ldots, T+1)$, if $z^*_{ij} = 1$ and $i \neq f^t$, we move i and all its predecessors mined in the same period to the previous period $(t-1)$. We thus obtain a new solution of the original problem (a new schedule), which is feasible, as discussed in the previous section.

The number of arcs and hence the number of binary variables in the model (LPP) might be very large if the number of blocks N is large. However, since the constraints matrix of (LPP) is unimodular, we can drop the integrality constraints and only restrict $z_{ij} \in [0,1] \quad \forall (i,j) \in E$. Thus, (LPP) can be solved efficiently using network-flow or linear programming techniques. On the other hand, there is no guarantee that the new obtained solution will be better than the current solution x. It might be similar to x (that's the case when

all the blocks in the optimal path are fictitious) as it might have a value less than or equal to the value of x, if we consider all the four terms of the objective function (1) because, as mentioned in the previous section, when computing the p_{ij}, we don't account for any revenues generated from processing ore taken from the stockpiles (the last term of the objective function (1)). Note, however, that the way the coefficients p_{ij} are defined allows us to minimize the surplus of ore production. This should allow us to minimize the use of the stockpiles and will implicitly minimize the costs incurred whenever the stockpiles are involved.

4.3 Algorithm framework

We will now describe the algorithm we propose to improve the initial solution x^0. To simplify the presentation, we will say that we perform *a backward pass* if we try to delay the extraction of the blocks, and that we perform *a forward pass* if we try to advance the extraction of the blocks. *A full pass* means that we perform two consecutive *backward forward passes* or conversely two consecutive *forward backward passes*.

Denote by *xbest* the best solution found so far. We use the same notation as in the previous section for the current solution (i.e., x). Set $x := x^0$ and $xbest := x^0$. We start by performing *a backward pass*. We construct the appropriate graph $G = (V, E)$, solve the corresponding (LPP), and use the so-obtained optimal path to obtain the new solution, as described in the previous section. If the new solution is better than *xbest*, then it replaces *xbest*. We repeat this process with the new solution, and terminate when the new solution is similar to the current one (i.e., if all the blocks in the optimal path are fictitious). Then, we perform *forward passes* in a similar manner using the same stopping criterion. When it is not possible to advance any blocks further, thus terminating the step, we again perform *backward passes*. The algorithm switches between *backward passes* and *forward passes* and terminates when in one *full pass* it finds that the current solution has not changed.

5 Numerical Results

Numerical tests were conducted on four different problems P_1, P_2, P_3, and P_4, representing four different deposits. P_1 and P_2 are two actual copper deposits where blocks are $15 \times 15 \times 15$ and $20 \times 20 \times 10$ meters in size, respectively. P_3 and P_4 are two actual gold deposits where blocks are $15 \times 15 \times 10$ and $10 \times 10 \times 5$ meters in size, respectively. Blocks in P_1 have different weights varying between 5805 and 9213.75 tonnes, while blocks in P_2, P_3, and P_4 weigh $10,800$, 5625, and 1250 tonnes each, respectively.

Each problem is characterized by a number of blocks N, a number of periods T, and a number of scenarios S for the metal content, specified in Table 1. Each period is one year long. The number of periods is set to $T = \lceil \frac{w_i N}{20{,}000{,}000} \rceil$. Each scenario has an equal probability of occurrence; i.e., $\pi_s = \frac{1}{S}$.

We assume that the production capacities are identical in all periods. For each problem, we assume that it is possible to extract a total of $W^t = 1.20 \sum_{i=1}^{N} \frac{w_i}{T}$ tonnes per year, of which the waste is sent to the waste dump (having an unlimited capacity), the low-grade material is sent to processor p_1 (having a capacity of $\Theta_{p_1}^t = \sum_{i=1}^{N} \sum_{s=1}^{S} \pi_s \frac{w_i \theta_{ip_1 s}}{T}$), and finally the high-grade material is sent to a second processor p_2 (having a capacity of $\Theta_{p_2}^t = \sum_{i=1}^{N} \sum_{s=1}^{S} \pi_s \frac{w_i \theta_{ip_2 s}}{T}$). Recall that $\theta_{ips} = 1$ if p is the most profitable processor to which block i can be sent if scenario s occurs and 0 otherwise, and that π_s denotes the probability that scenario s occurs. The processing capacities are thus set so as to make the satisfaction of the processing constraints more difficult. The economic parameters used to compute the mining costs, the blocks' revenues r_{is}, and the costs for sending and taking ore from the stockpiles are summarized in Table 2.

All the numerical experiments were performed on an Intel(R) Xeon(R) CPU X7350 computer (2.93 GHz) with 64 GB of RAM running under Linux. Version 12.2 of the solver CPLEX was used to solve the mathematical models (SP^t) and (LPP) introduced in Sections 3.1 and 4.2, respectively.

In the following, **BC-NF** and **GH-NF** denote the two variants of the proposed solution procedure, obtained by combining the two methods for generating the initial solution in Section 3.2 (the exact method and the greedy heuristic, denoted **BC** and **GH**, respectively) with the network-flow based improvement algorithm in Section 4.3 (denoted **NF**) .

Table 1: Characteristics of the problems used in the numerical experiments

Problem	Number of blocks (N)	Number of periods (T)	Number of scenarios (S)
P_1 Metal type: copper Block size: $15 \times 15 \times 15$ meters Block weight: variable $5805 \leq w_i \leq 9213.75$ tonnes	14,118	6	25
P_2 Metal type: copper Block size: $20 \times 20 \times 10$ meters Block weight: $w_i = 10,800$ tonnes	28,154	16	20
P_3 Metal type: gold Block size: $15 \times 15 \times 10$ meters Block weight: $w_i = 5625$ tonnes	48,821	14	20
P_4 Metal type: gold Block size: $10 \times 10 \times 5$ meters Block weight: $w_i = 1250$ tonnes	63,424	4	20

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

Parameters	Copper (P_1 and P_2)	Gold (P_3 and P_4)
Mining cost ($\bar{c}$)	\$1/t	\$1/t
Low-grade processor p_1		
Processing cost (c_{p1})	\$2.25/t	\$6/t
Recovery (α_{p1})	55%	45%
Cost of sending ore to the stockpile (c^+_{p1})	\$0.25/t	\$0.25/t
Cost of taking ore from the stockpile (c^-_{p1})	\$0.45/t	\$0.45/t
High-grade processor p_2		
Processing cost (c_{p2})	\$9/t	\$15/t
Recovery (α_{p2})	90%	95%
Cost of sending ore to the stockpile (c^+_{p2})	\$0.25/t	\$0.25/t
Cost of taking ore from the stockpile (c^-_{p2})	\$0.45/t	\$0.45/t
Metal price (μ)	\$2/lb	\$30/g
Selling cost (ξ)	\$0.3/lb	\$0.2/g
Discount rate (d)	10%	10%

In Table 3, we compare **BC-NF** and **GH-NF** in terms of the value *Zbest* of the best solution generated (value of the objective function (1)) and the solution time, CPU, given in minutes. Furthermore, for each criterion, we report the relative performance of **BC-NF** over **GH-NF** (the ratio of the value obtained by **BC-NF** to that obtained by **GH-NF**).

As can be observed from the results in Table 3, both variants solve the problems in reasonable computational times. In terms of solution quality, **BC-FN** in general produces results slightly better than **GH-FN** (except for the smallest problem, P_1), but at the expense of higher computational times. The performance differences are more pronounced for the smaller problem P_1, where the CPU is reduced by a factor of 70 when **GH-FN** is used.

To further analyze the performance of the two variants, we compare the results they produce with those obtained with CPLEX 12.2. Since none of the problems was solved to optimality within one day, we have solved the linear relaxation of the problem to obtain an upper bound on the optimal value. In Table 4, we evaluate the effectiveness of the two variants with respect to the upper bounds provided by CPLEX. For each problem, except for problems P_2 and P_3, for which CPLEX was unable to solve the linear relaxation

Table 3: Comparison between BC-NF and GH-NF

Problem	N	T	S	*Zbest*			CPU time (minutes)		
				BC-FN	**GH-FN**	$\frac{\textbf{GH-FN}}{\textbf{BC-FN}}$	**BC-FN**	**GH-FN**	$\frac{\textbf{GH-FN}}{\textbf{BC-FN}}$
P_1	14,118	6	25	27,861,500	27,892,900	0.999	135.58	1.93	70.16
P_2	28,154	16	20	226,008,000	222,152,000	1.017	52.79	13.35	3.95
P_3	48,821	14	20	473,877,000	470,405,000	1.007	371.97	83.89	4.43
P_4	63,424	4	20	163,753,000	160,391,000	1.021	1964.32	1516.50	1.29

Table 4: Comparison between BC-NF, GH-NF and CPLEX

Problem	N	T	S	*%Gap*		CPU time (minutes)		
				BC-FN	**GH-FN**	**BC-FN**	**GH-FN**	**CPLEX**
P_1	14,118	6	25	6.48	6.37	135.58	1.93	2421.95
P_2	28,154	16	20	–	–	52.79	13.35	CPU > 40,320.00
P_3	48,821	14	20	–	–	371.97	83.89	CPU > 40,320.00
P_4	63,424	4	20	0.23	2.28	1964.32	1516.50	8639.60

within 4 weeks, we indicate the value of the relative gap $\%Gap$ between the value *Zbest* of the best solution generated (as reported in Table 3) and the linear relaxation optimal value Z_{LR}:

$$\%Gap = \frac{Z_{LR} - Zbest}{Z_{LR}} \times 100.$$

The CPU times required by **BC-FN** and **GH-FN** to solve the problems as well as the CPU time required by **CPLEX** to solve the linear relaxation of the problems are given in minutes in the last three columns of the table.

The results show that both variants outperform **CPLEX**, in the sense that, for all the tested problems, they require significantly shorter solution times than the time required by **CPLEX** to solve only the linear relaxation of the problems. In particular, for the smallest problem, P_1, the computational times for **GH-FN** and **BC-FN** are only about 1 minute and 2 hours, respectively, while the CPU time required by CPLEX to solve the linear relaxation of the problem is approximately 2 days. Furthermore, the solutions obtained by the variants in these relatively short computational times are of very good quality, as can be seen from the small values of $\%Gap$. Note that, for problems P_2 and P_3, we cannot assess the quality of the solutions obtained by the variants because **CPLEX** was not able to solve their linear relaxation within 4 weeks. For these two problems, we give below detailed results, namely: sample cross-sections of the physical schedules and risk profiles for the waste tonnage, the ore tonnage processed in each processor, the tonnage sent to each stockpile, the tonnage taken from each stockpile, and the tonnage in each stockpile. Detailed results for the other two problems are given in Appendix A.

Figures 1 and 2 show sample cross-sections of the physical schedules obtained for problems P_2 and P_3, respectively. The physical schedules obtained by the two methods are different. As we can see from Figure 1, for problem P2, the areas mined in the first periods of the life of the mine are not all contiguous, especially in the schedule generated by **BC-FN**, because the variants try to find the mining sequence that gives the highest NPV in each period. But, in both cases the schedules respect the *slope constraints*.

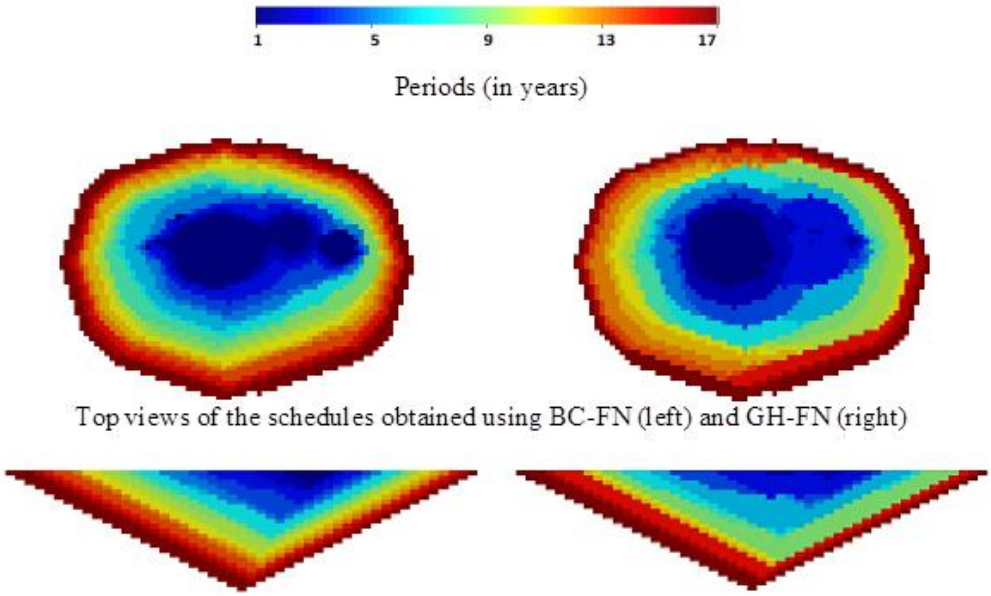

Figure 1: Cross-sections of the physical schedules of problem P_2

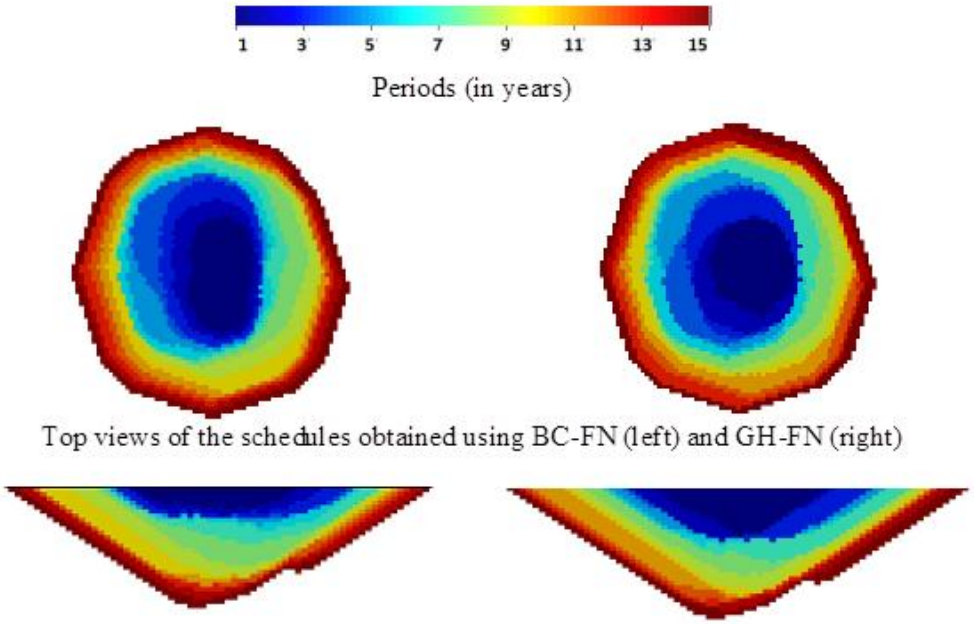

Figure 2: Cross-sections of the physical schedules of problem P_3

Figures 3–10 show the risk profiles for the two problems. Overall, the two methods perform similarly in terms of reducing the risk. Regarding the tonnage processed in each processor, Figures 4 and 8 indicate that the two methods perform extremely well, in the sense that there is very small risk in the first periods (small difference between the minimum and maximum curves), but the risk increases towards the end of the life of the mine. However, for the tonnage stockpiled, the differences between the minimum, maximum, and expected curves are more pronounced. From these figures, it also appears that, at the end of the life of the mine, the stockpile associated with the high-grade material is empty whatever the scenario is, indicating that all the ore stockpiled during the first periods (surplus of ore production) has been processed. This is not the case for the stockpile associated with the low-grade material, where a small tonnage remains unprocessed in the stockpile. Figures 5 and 9 show that for both stockpiles, the ore is sent during the first periods and taken during the last periods. From Figures 3 and 7 it appears indeed, 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. Finally, regarding the net present value, Figures 6 and 10 show that the risk is higher in the last periods of the life of the mine than it is in the first periods, which is in concordance with the results in Figures 4 and 8.

6 Conclusions

In this paper, we have considered a complex version of the open-pit mine production scheduling problem that accounts for geological uncertainty and considers multiple destinations for the mined material, including stockpiles. We have developed a mathematical formulation of it and an efficient network-flow based algorithm to solve it. We have implemented two variants of the algorithm, which differ in terms of the process used to generate the initial solution – an exact method and a greedy heuristic.

The experiments conducted on small to medium size instances, with up to 63,424 blocks, are promising. They show that the two proposed variants outperform CPLEX and can produce very good quality solutions in relatively short computational times. When comparing the two variants, the results indicate that, in general, the variant based on the exact method dominates the one based on the greedy heuristic in terms of solution quality, but, with respect to solution time, the results indicate that, for the tested problems, the latter variant dominates.

The proposed solution approach is flexible enough to handle other constraints. We are currently testing it on large size instances. Future research will be devoted to developing other efficient approaches for the problem. One approach we are investigating is a heuristic based on the adaptive large neighborhood search framework. Briefly, this method repeatedly destroys a part of the current solution and reconstructs it in the hope of achieving an improvement.

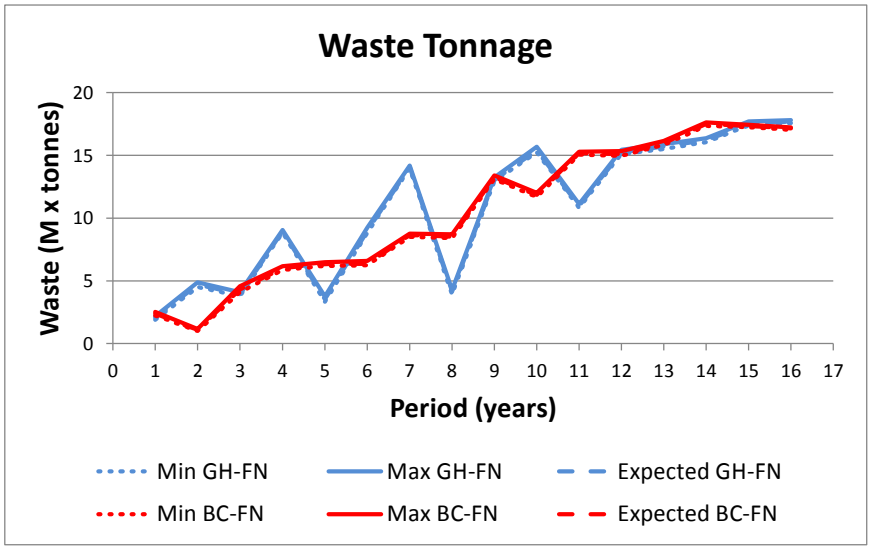

Figure 3: Risk analysis for waste tonnage for problem P_2

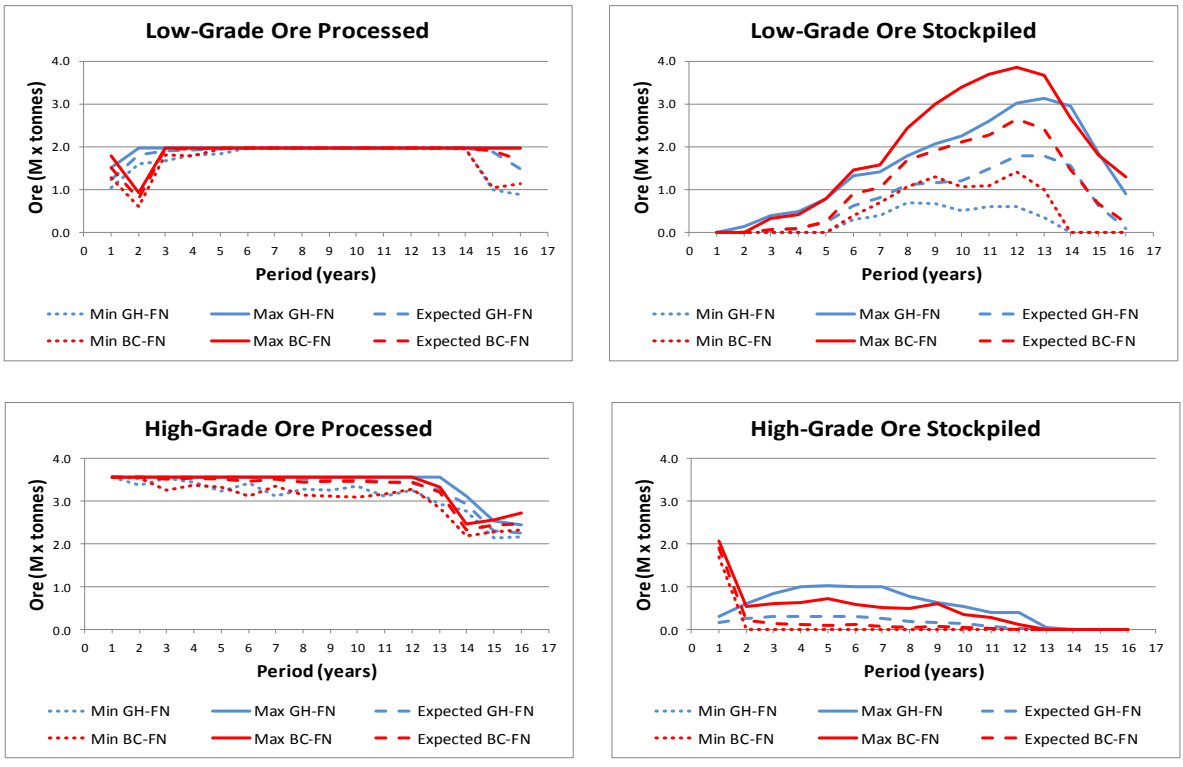

Figure 4: Risk analysis for ore tonnage processed and stockpiled for problem P_2

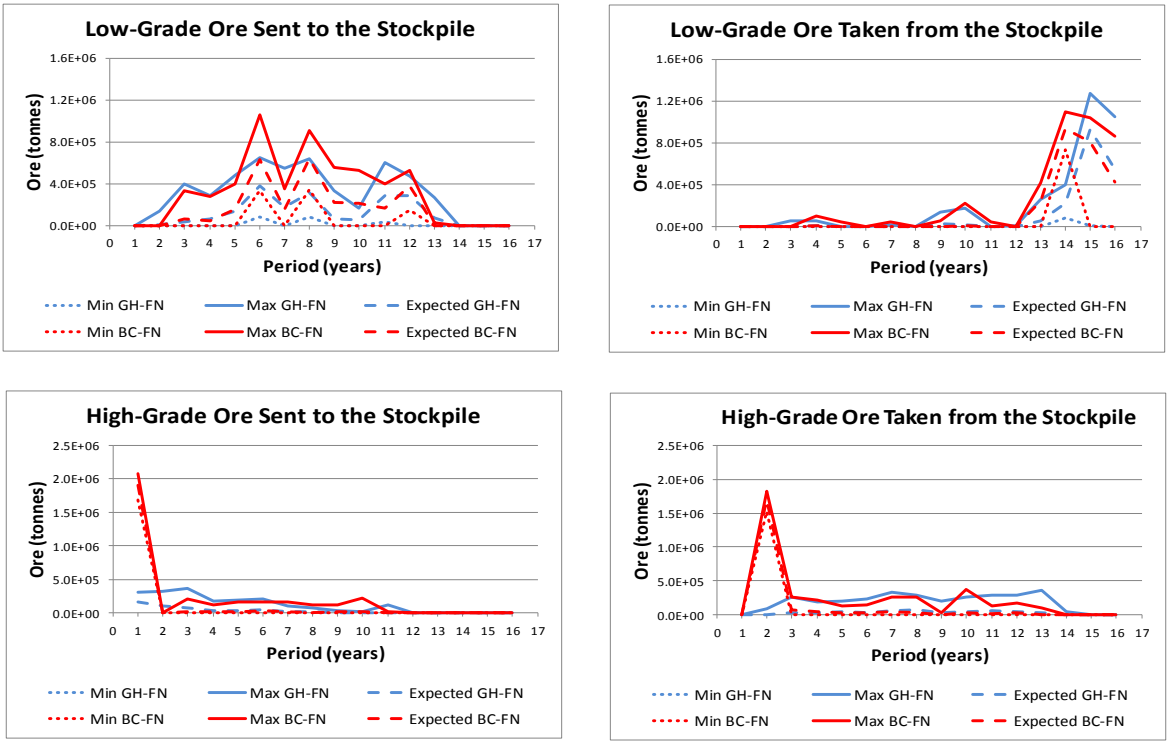

Figure 5: Risk analysis for ore tonnage sent and taken from the stockpiles for problem P_2

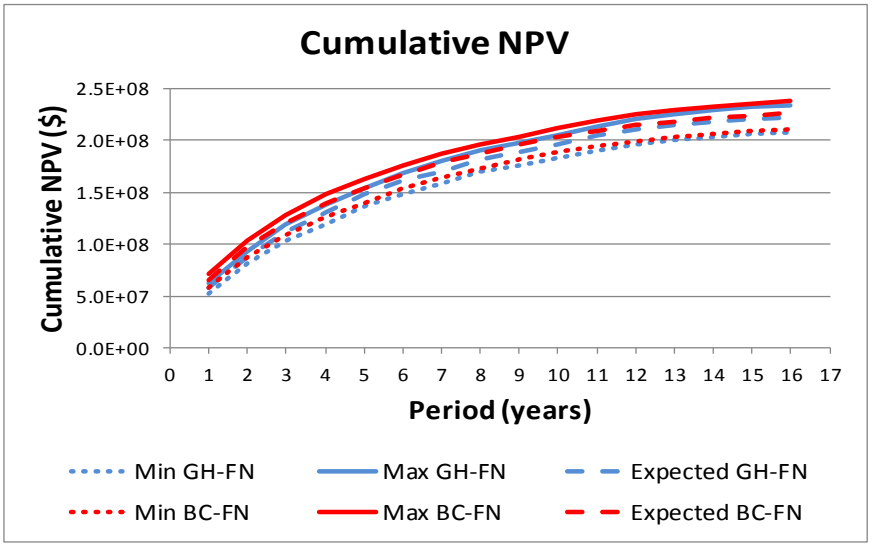

Figure 6: Risk analysis for cumulative discounted cash values for problem P_2

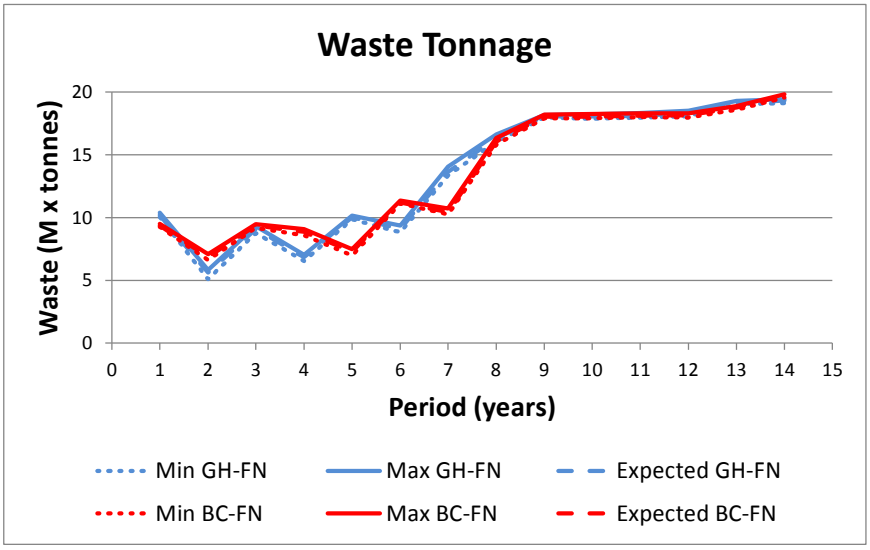

Figure 7: Risk analysis for waste tonnage for problem P_3

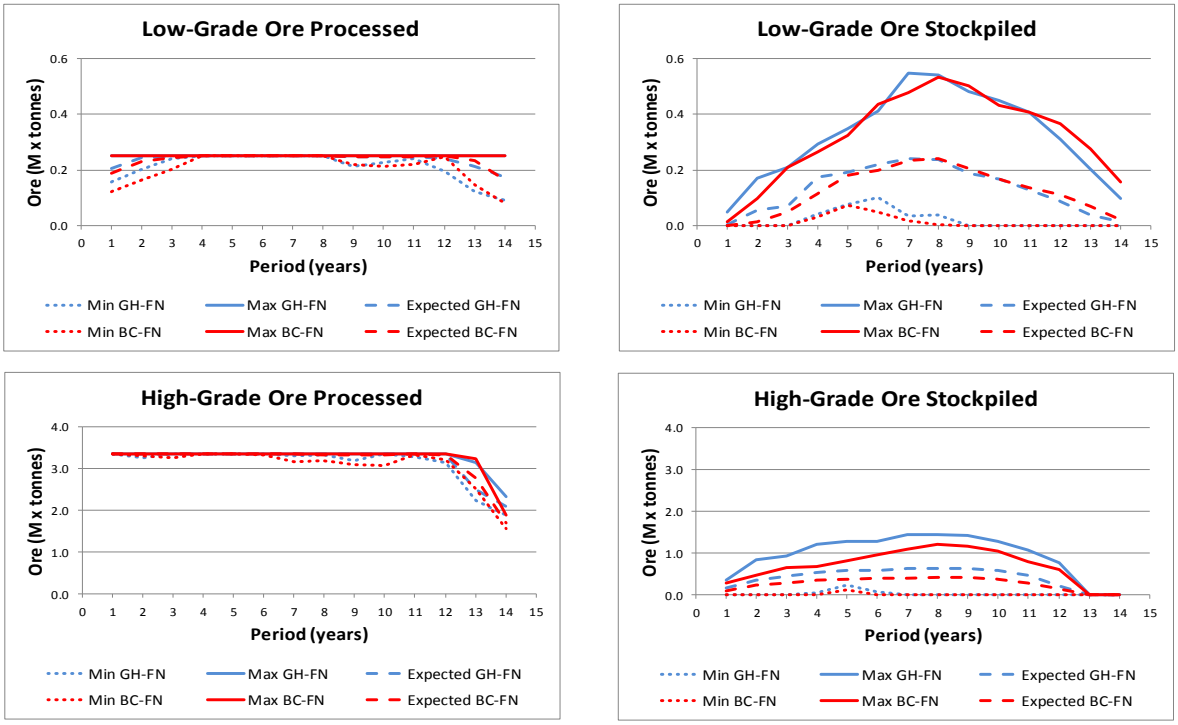

Figure 8: Risk analysis for ore tonnage processed and stockpiled for problem P_3

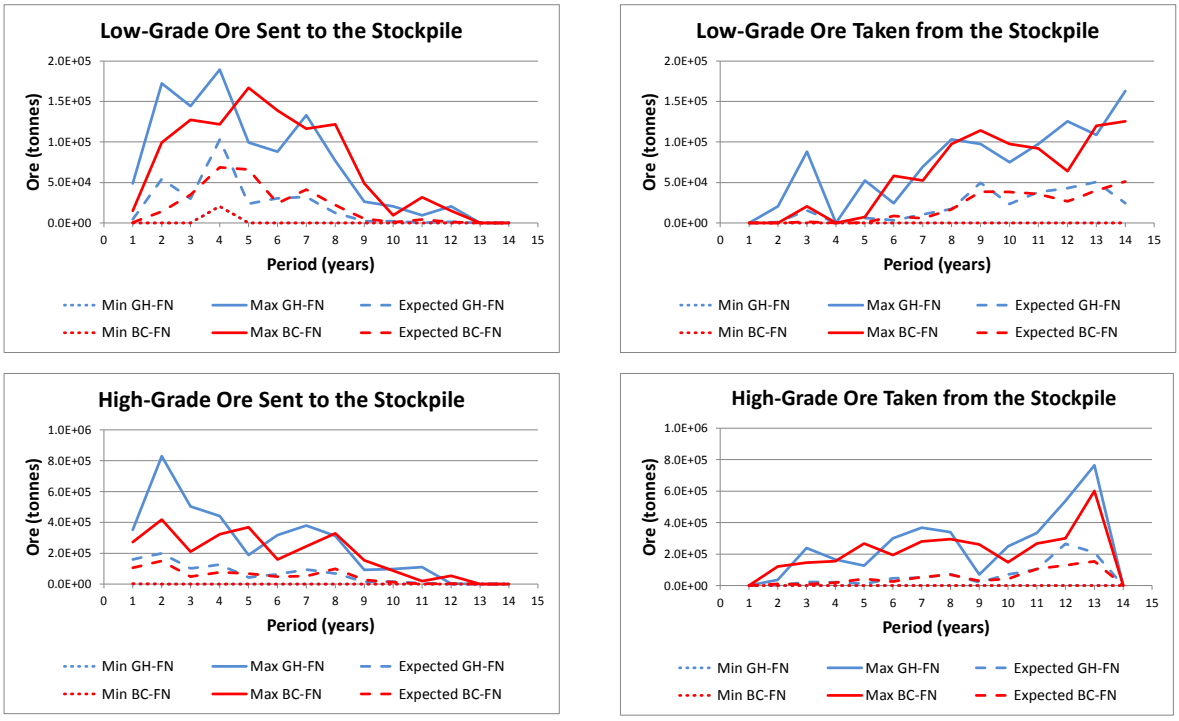

Figure 9: Risk analysis for ore tonnage sent and taken from the stockpiles for problem P_3

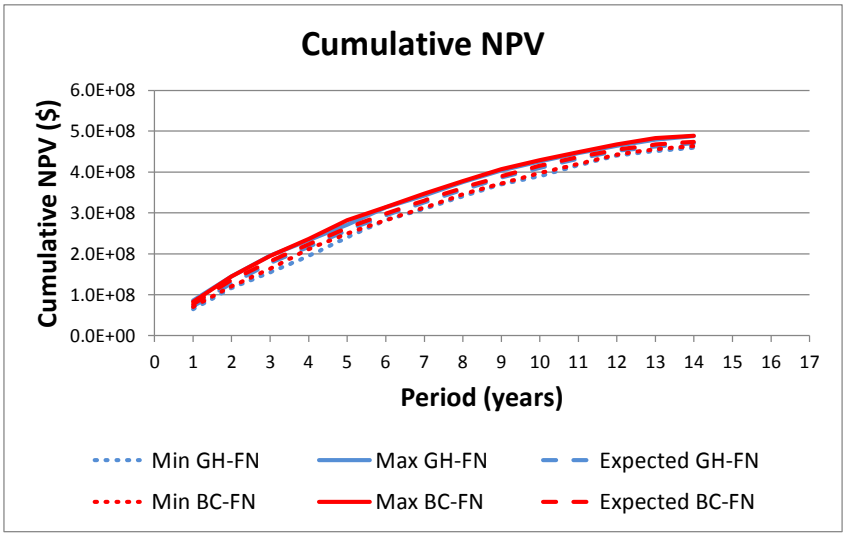

Figure 10: Risk analysis for cumulative discounted cash values for problem P_3

Appendix A – Risk Analysis

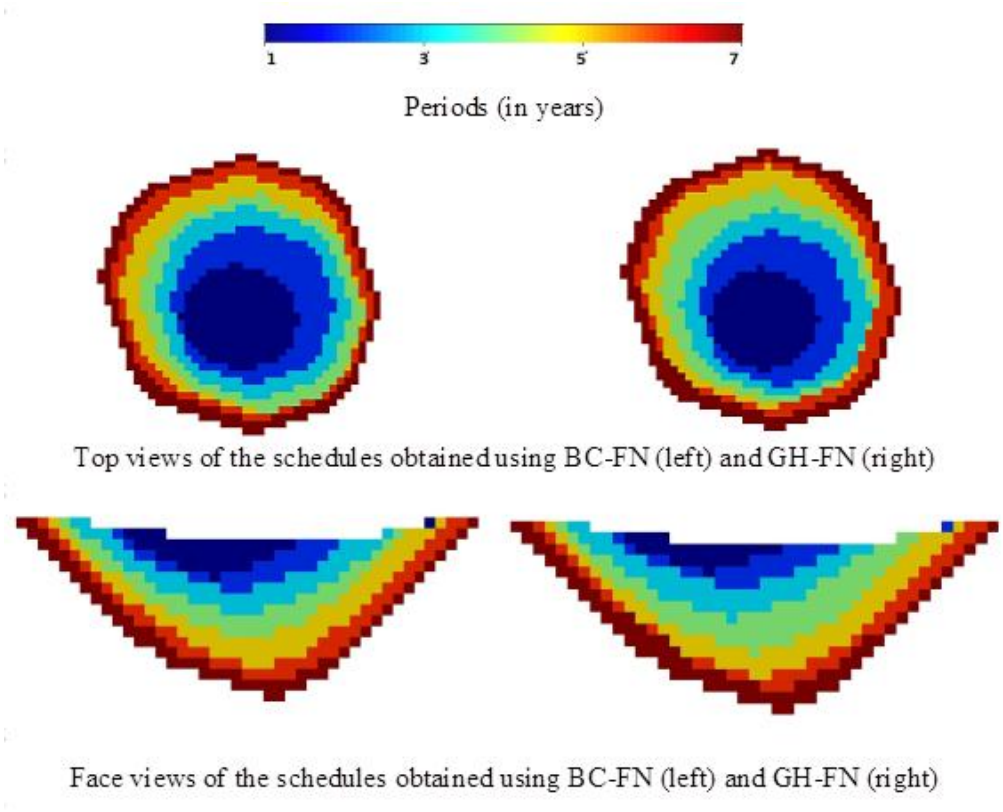

Figure 11: Cross-sections of the physical schedules of problem P_1

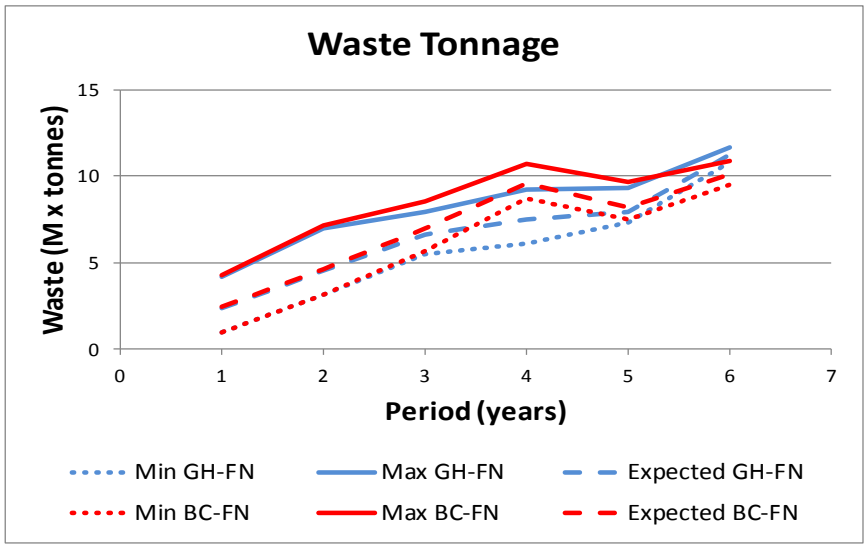

Figure 12: Risk analysis for waste tonnage for problem P_1

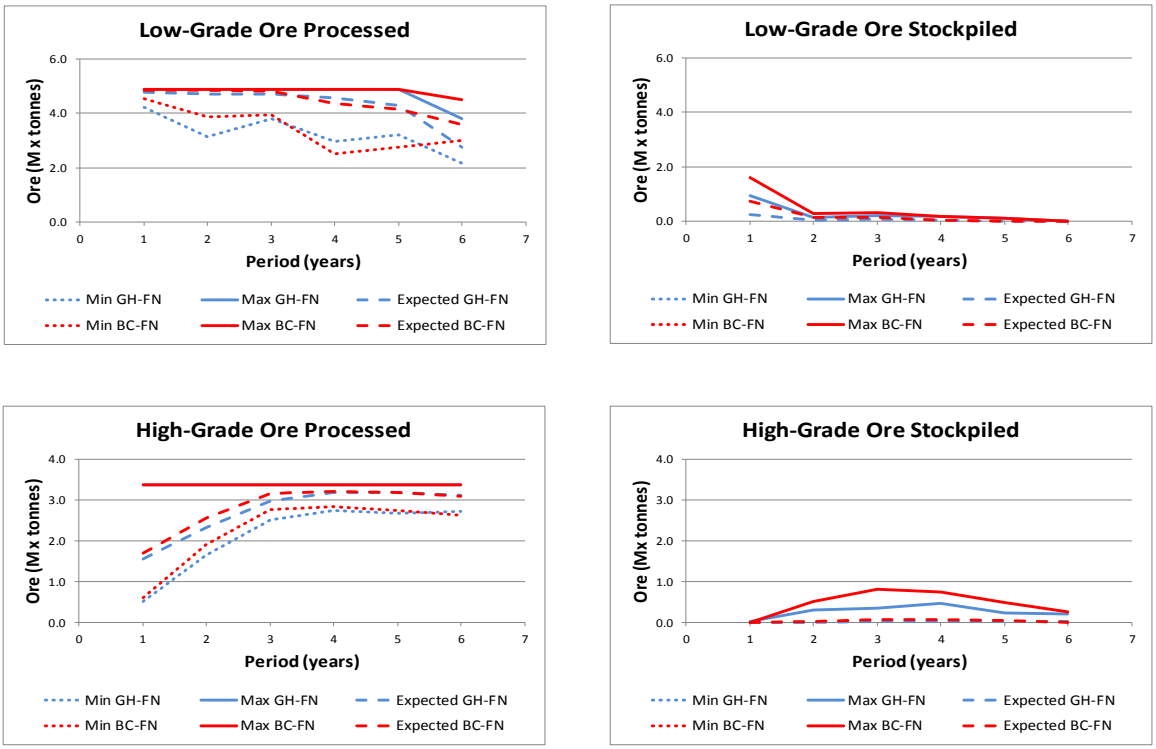

Figure 13: Risk analysis for ore tonnage processed and stockpiled for problem P_1

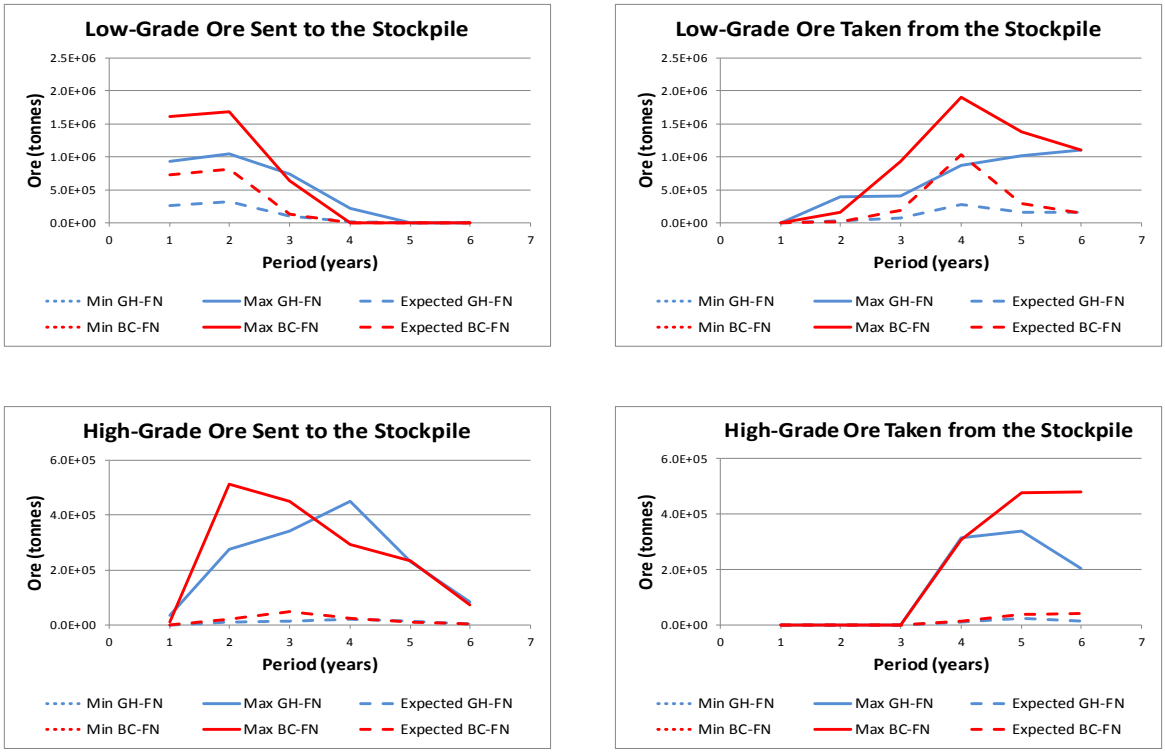

Figure 14: Risk analysis for ore tonnage sent and taken from the stockpiles for problem P_1

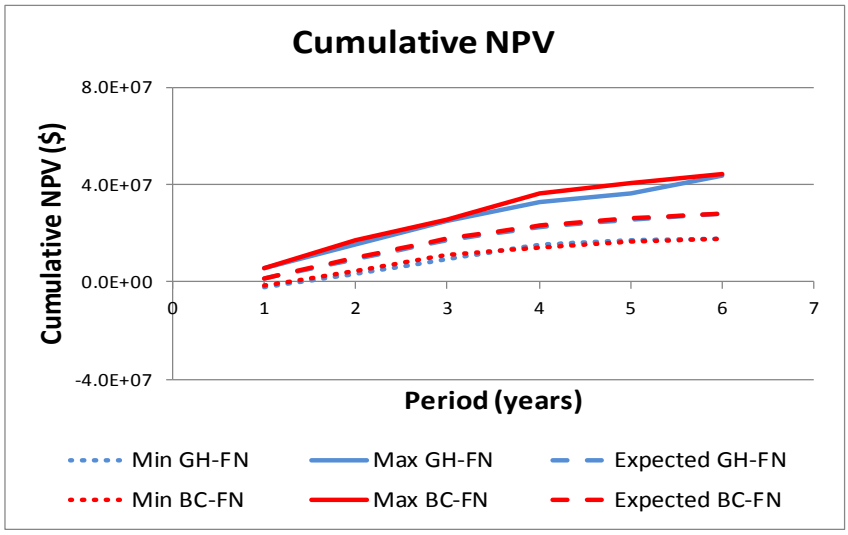

Figure 15: Risk analysis for cumulative discounted cash values for problem P_1

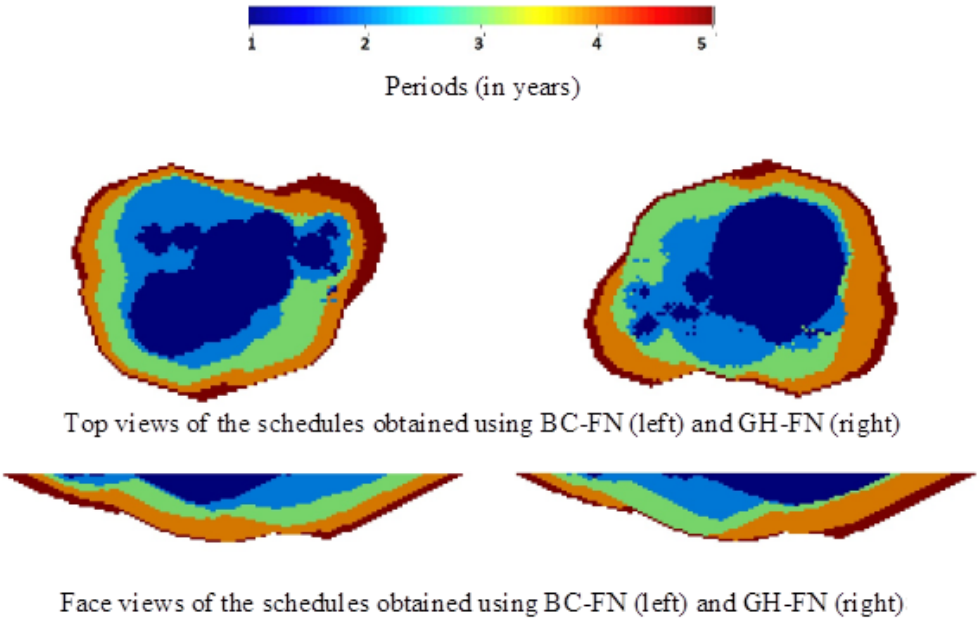

Figure 16: Cross-sections of the physical schedules of problem P_4

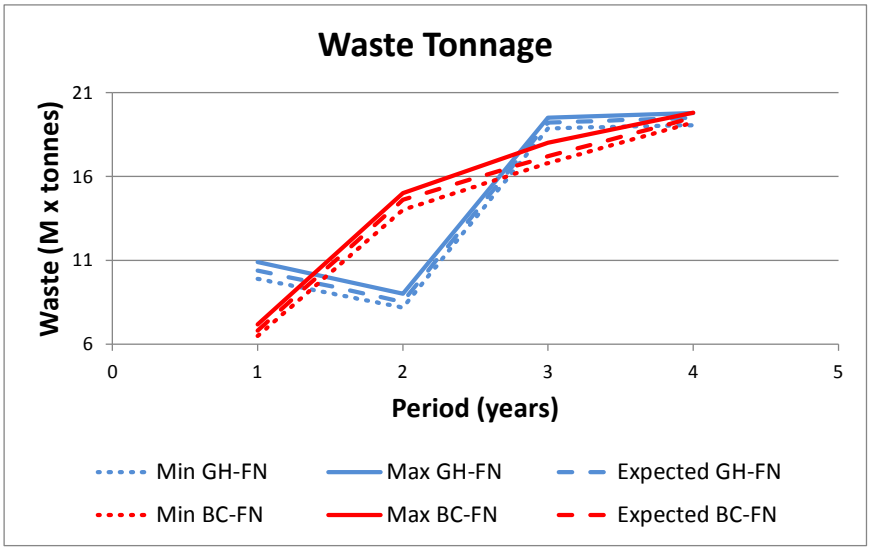

Figure 17: Risk analysis for waste tonnage for problem P_4

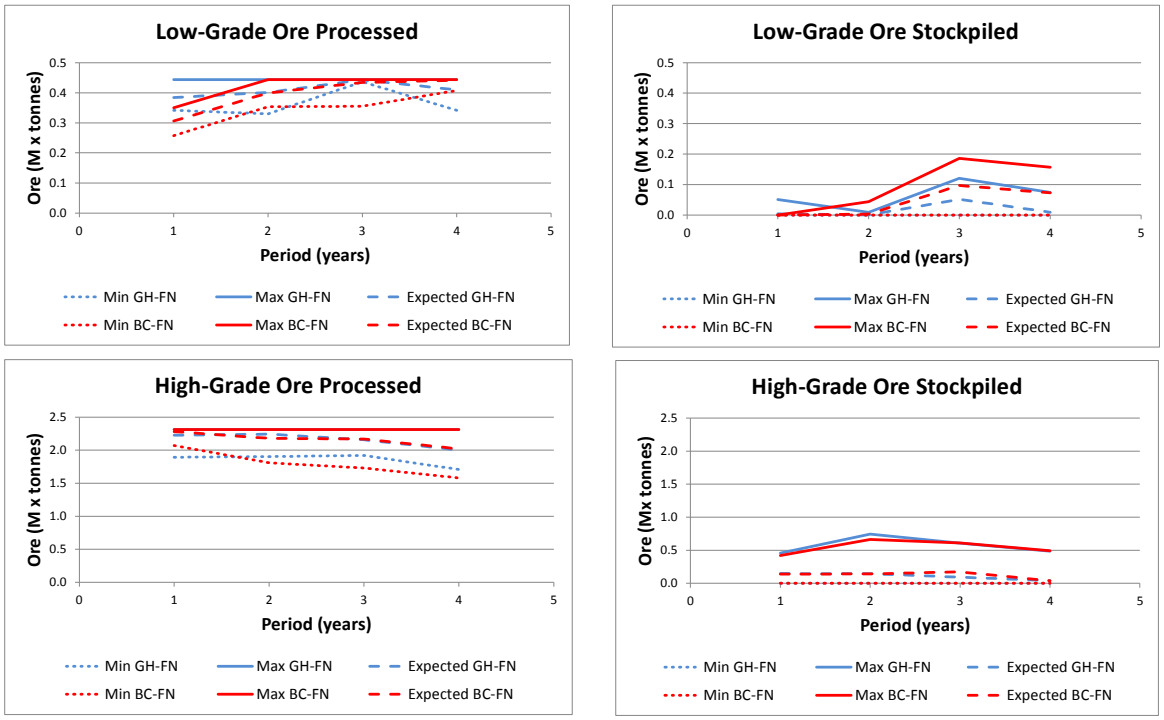

Figure 18: Risk analysis for ore tonnage processed and stockpiled for problem P_4

Figure 19: Risk analysis for ore tonnage sent and taken from the stockpiles for problem P_4

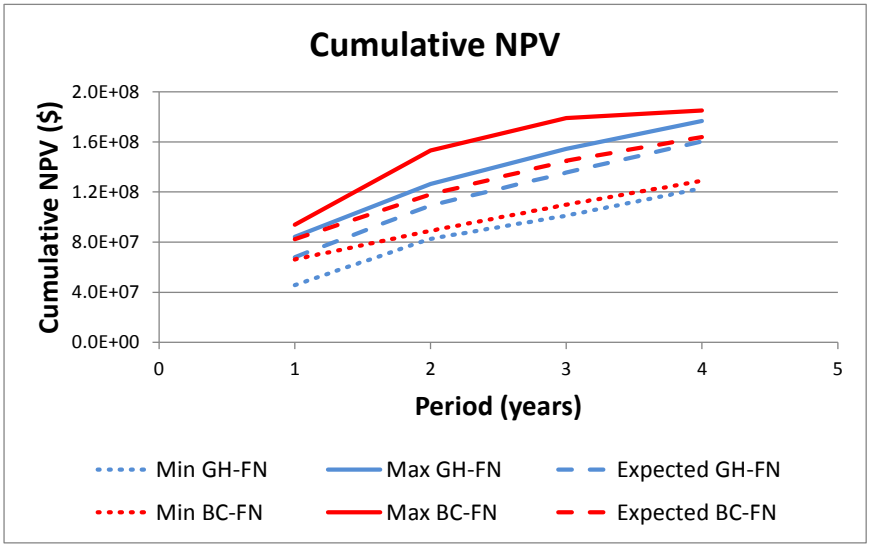

Figure 20: Risk analysis for cumulative discounted cash values for problem P_4

References

Ahuja RK, Ergun O, Orlin J, Punnen A (2002) A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 123:75–102.

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.

Bienstock D, Zuckerberg M (2010) Solving LP relaxations of large-scale precedence constrained problems. Lecture Notes in Computer Science 6080:1–14.

Birge J, Louveaux F (1997) Introduction to Stochastic Programming. Springer.

Boland N, Dumitrescu I, Froyland G (2008) A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology. Optimization Online, last accessed: June 21, 2012. URL http://www.optimization-online.org/DB_FILE/2008/10/2123.pdf.

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

Caccetta L, Hill SP (2003) An application of Branch and Cut to open pit mine scheduling. Journal of Global Optimization 27:349–365.

Chatterjee S, Lamghari A, Dimitrakopoulos R (2010) A two-phase heuristic method for constrained pushback design. In: Proceedings of MININ2010, Conference on Mining Innovation (R. Castro, X. Emery, and R.Kuyvenhoven eds.). Santiago, Chile:195–204.

Dagdelen K, Johnson TB (1986) Optimum open pit mine production scheduling by lagrangian parameterization. In: Proceedings of 19th International APCOM Symposium. Littleton, CO:127–142.

Denby B, Schofield D (1994) Open-pit design and scheduling by use of genetic algorithms. Transactions of the Institution of Mining and Metallurgy 103:A21–A26.

Dimitrakopoulos R (2011) Stochastic optimization for strategic mine planning: a decade of developments. Journal of Mining Science 84:138–150.

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.

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

Ferland JA, Amaya J, Djuimo MS (2007) Application of a particle swarm algorithm to the capacitated open pit mining problem. Autonomous Robots and Agents, Mukhopadhyay S. and Sen Gupta G. Ed. Springer-Verlag:127–133.

Gershon M (1987) Heuristic approaches for mine planning and production scheduling. International Journal of Mining and Geological Engineering 5 (1):1–13.

Godoy M (2003) The effective management of geological risk. Ph.D. thesis, University of Queensland, Australia.

Goodfellow R, Dimitrakopoulos R (2011) Algorithmic integration of geological uncertainty in pushback designs for complex multi-process open-pit mines. COSMO Report 5 (1):1–30.

Lamghari A, Dimitrakopoulos R (2011) Hybridization of exact and metaheuristic methods for scheduling production in large-scale open-pit mines. COSMO Report 5 (2):168–190.

Leite A, Dimitrakopoulos R (2007) A stochastic optimisation model for open pit mine planning: Application and risk analysis at a copper deposit. Transactions of the Institution of Mining and Metallurgy 116:A109–A118.

Menabde M, Froyland G, Stone P, Yeates G (2005) Mining schedule optimization for conditionally simulated orebodies. Orebody Modelling and Strategic Mine Planning. AusIMM Spectrum Series 14:347–352.

Moreno E, Espinoza D, Goycoolea M (2010) Large-scale multi-period precedence constrained knapsack problem: A mining application. Electronic Notes in Discrete Mathematics 36:407–414.

Picard C (1976) Maximal closure of a graph and application to combinatorial problems. Management Science 22:1268–1272.

Ramazan S (2007) The new fundamental tree algorithm for production scheduling of open pit mines. European Journal of Operational Research 177:1153–1166.

Ramazan S, Dimitrakopoulos R (2005) Stochastic optimization of long term production scheduling for open pit mines with a new integer programming formulation. Orebody Modelling and Strategic Mine Planning. AusIMM Spectrum Series 14:359–365.

Ramazan S, Dimitrakopoulos R (2012) Production scheduling with uncertain supply: a new solution to the open pit mining problem. Optimization and Engineering DOI: 10.1007/s11081-012-9186-2.

Sevim H, Lei D (1998) The problem of production planning in open pit mines. INFOR 36 (1-2):1–12.

Tolwinski B, Underwood R (1996) A scheduling algorithm for open pit mines. IMA Journal of Mathematics Applied in Business & Industry 7:247–270.