European Journal of Operational Research 222 (2012) 642–652

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

A diversified Tabu search approach for the open-pit mine production scheduling problem with metal uncertainty

Amina Lamghari *, Roussos Dimitrakopoulos

COSMO – Stochastic Mine Planning Laboratory, McGill University, FDA Building, 3450 University Street, Montreal, Quebec, Canada H3A 2A7

ARTICLE INFO

Article history:
Received 6 June 2011
Accepted 11 May 2012
Available online 28 May 2012

Keywords:
Scheduling
Open-pit mining
Uncertainty
Metaheuristics
Tabu search
Diversification

ABSTRACT

This paper presents a metaheuristic solution approach based on Tabu search for the open-pit mine production scheduling problem with metal uncertainty. To search the feasible domain more extensively, two different diversification strategies are used to generate several initial solutions to be optimized by the Tabu search procedure. The first diversification strategy exploits a long-term memory of the search history. The second one relies on the variable neighborhood search method. Numerical results on realistic large-scale instances are provided to indicate the efficiency of the solution approach to produce very good solutions in relatively short computational times.

1. Introduction

Production scheduling of open-pit mines is an important problem that arises in surface mine planning. It can be summarized as follows: The ore body is represented as a three-dimensional array of blocks. Each block has a set of attributes including its weight and its metal content estimated using drill hole data. Blocks have to be mined from the ground before they can be processed in a plant where the metal that they contain is recovered. A block is not processed unless it is economically profitable to do so; that is, only if the expected revenue from selling the metal contained within the block is greater than the processing cost. This leads to a partition of the blocks into two groups: the group of *ore* blocks includes blocks that can be processed profitably, and the group of *waste* blocks includes all the remaining blocks. In addition, each block has an economic value representing the net profit associated with it. The open-pit mine production scheduling problem (MPSP) consists of identifying which blocks should be mined during each period of the life of the mine so as to maximize the net present value of the mining operation. Different physical and operational constraints have to be met when scheduling blocks. Those considered in this paper are as follows:

- *Reserve constraints*: a block can be mined at most once during the horizon.
- *Slope constraints*: a block cannot be mined before its predecessors. Indeed, to have access to a given block, all the blocks overlying it, referred to as its predecessors, have to be removed beforehand.
- *Mining constraints*: the total weight of blocks (*waste* and *ore*) mined during each period should be at least equal to a minimum value to avoid having an unbalanced mining flow throughout the periods. On the other hand, it should not exceed the mining equipment capacity available during that period.
- *Processing constraints*: the total weight of *ore* blocks mined during each period should be at least equal to a minimum amount required to feed the processing plant, but it should not exceed the processing plant capacity. Note that we assume that *ore* blocks are processed during the same period when they are mined.
- *Metal production constraints*: during each period, the amount of metal recovered from the *ore* blocks processed should not exceed the amount that can be sold during this period and should not be less than a minimum amount.

Production scheduling decisions are critical for a mining company, as they serve as a baseline for determining the periodical metal production and are a key factor in determining the financial returns of significant investments in the order of hundreds of millions of dollars. Finding the most profitable production schedule is a complex task since it involves large data sets, multiple constraints, and uncertainty. Uncertainty stems mainly from the fact that the metal content of the blocks is not known precisely at the time decisions are made: it is inferred from limited drilling

* Corresponding author. Tel.: +1 514 398 4755x089638; fax: +1 514 398 7099.
E-mail addresses: amina.lamghari@mail.mcgill.ca (A. Lamghari), roussos.dimitrakopoulos@mcgill.ca (R. Dimitrakopoulos).

0377-2217
http://dx.doi.org/10.1016/j.ejor.2012.05.029

information. This type of uncertainty is referred to in the literature as grade, geological, or metal uncertainty. Another source of uncertainty is the volatility of the metal prices. The considerations above clearly explain why the MPSP has been a frequently studied optimization problem since the 1960s (Johnson, 1969).

Different optimization methods have been applied to solve the deterministic version of the MPSP, which assumes that all the problem parameters are well known. Dagdelen and Johnson (1986) propose an exact method based on Lagrangian relaxation. A branch-and-cut algorithm was developed by Caccetta and Hill (2003). Another exact approach based on cutting plane techniques was recently proposed by Bley et al. (2010). The major limitation with exact methods is that they can only be applied to instances of relatively small size. Solving instances of realistic size, where typically the number of blocks is in the order of tens to hundreds of thousands, requires prohibitive computational times. To reduce the size of the problem and thus make large instances of practical interest computationally tractable by exact methods, Ramazan (2007) exploits the structure of the problem to aggregate blocks into groups. Other approaches to tackling realistic large-scale instances rely on heuristics (Gershon, 1987), a combination of dynamic programming and heuristics (Tolwinski and Underwood, 1996), genetic algorithms (Denby and Schofield, 1994), and particle swarm algorithms (Ferland et al., 2007). A more detailed review of the different solution approaches for the MPSP can be found in Newman et al. (2010). This paper also provides a review of other optimization problems that arise in the mining context such as fleet allocation (Souza et al., 2010; Topal and Ramazan, 2010).

The deterministic version of the MPSP ignores the uncertain nature of the problem, which leads to misleading assessments (Ravenscroft, 1992; Dowd, 1994; Dimitrakopoulos et al., 2002; Godoy and Dimitrakopoulos, 2004). Stochastic versions of the problem, which are more realistic and more relevant, have been attracting the attention of an increasing number of researchers in the last decade. The literature has mostly focused on addressing metal uncertainty. Formulations minimizing deviations from production targets over multiple ore body scenarios describing the metal uncertainty have been proposed by Godoy (2003), Ramazan and Dimitrakopoulos (2007, 2012), and Albor and Dimitrakopoulos (2010). Menabde et al. (2007) develop a formulation maximizing the expected net present value over several scenarios while satisfying the production targets in an average sense. Golamnejad et al. (2006) propose a chance-constrained formulation; however, chance-constrained formulations make severe and unrealistic assumptions, such as a Gaussian distribution and independence of the metal content of mining blocks (Ramazan and Dimitrakopoulos, 2012). Boland et al. (2008) take into account the metal uncertainty via a multistage stochastic programming approach.

While different approaches for modeling metal uncertainty have been developed, solution methods have received relatively less attention. A solution method based on simulated annealing is described in Godoy (2003). The method used in Albor and Dimitrakopoulos (2010) consists of generating a set of nested pits, grouping these pits into pushbacks, and then generating a schedule based on the pushback designs obtained. The stochastic models proposed in the studies by Menabde et al. (2007), Ramazan and Dimitrakopoulos (2007, 2012), and Boland et al. (2008) are solved using the commercial mixed integer programming software CPLEX, which restricts these approaches to instances of relatively small size (typically, instances with less than 20,000 blocks).

The objective of this paper is to propose an efficient solution method to tackle large instances of the MPSP with metal uncertainty. More specifically, we propose a metaheuristic method based on a Tabu search procedure (Glover and Laguna, 1998; Hansen, 1986). To search the feasible domain more extensively, we use two different diversification strategies to generate several initial solutions to be optimized by the Tabu search procedure. The first diversification strategy exploits a long-term memory of the search history. The second one relies on the variable neighborhood search method (Hansen and Mladenovic, 2001). Even if the solution method is introduced for the specific problem studied in this paper and the specific approach used to address metal uncertainty, its flexibility should allow it to be easily adapted to dealing with other uncertainty modeling approaches and to account for additional scheduling constraints.

We provide numerical results allowing us to evaluate the efficiency of the method with respect to the upper bounds provided by CPLEX. These results indicate that the proposed approach can generate very good solutions in relatively short computational times. Furthermore, comparing the two diversification strategies indicates that the first one generates better solutions than the second one.

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. Sections 3 and 4 describe the Tabu search procedure and the two diversification strategies, respectively. Computational results on real-life data are reported and discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. Problem formulation

We assume that a finite set of possible scenarios, each specifying the metal content of each block, is available, and that each scenario has an equal probability of occurrence.

Referring to the description given in the previous section, the problem can be formulated as a two-stage stochastic integer programming model (Birge and Louveaux, 1997). In the first stage, one determines for each period of the horizon a set of blocks to be mined respecting the minimum and maximum mining limits, such that each block in each set is scheduled exactly once after all its predecessors. The metal content of the blocks, and thus their group (*ore* or *waste*), is uncertain at this stage. In the second stage, the uncertainty is revealed according to each scenario. In some periods, *ore* blocks available, requiring processing, may have a total weight exceeding the processing plant capacity, while in other periods, they may not meet the minimum requirement. The same may happen with the metal recovered from the *ore* blocks processed. To adapt to the situation at hand, some recourse actions should be taken. For instance, if the plant capacity is not sufficient to process all *ore* blocks mined during a given period, the recourse action would be to stockpile the excess so that it could be used later when shortage occurs. In each period, each recourse action yields a cost. Hence, the problem consists of identifying a first-stage solution that minimizes the expected cost of the second-stage solution; i.e., a schedule that maximizes the expected net present value of the mining operation minus the expected recourse costs incurred due to the violation of the *processing constraints* and the *metal production constraints*. Note that the proposed formulation is similar to that presented in Ramazan and Dimitrakopoulos (2012) except that these authors minimize the total penalty costs for deviating from production targets.

In the rest of the paper, we will refer to the *processing constraints* and the *metal production constraints* as stochastic constraints. The other constraints (*reserve constraints*, *slope constraints*, and *mining constraints*) are referred to as non-stochastic constraints.

We use the following notation to formulate the first stage of the problem:

- 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_i: the set of predecessors of block i; i.e., blocks that should be removed before i can be mined. Note that if block p is a predecessor of block i, then i is called a successor of p.
- S_i: the set of successors of block i.
- w_i: the weight of block i.
- $\underline{W}_t$: minimum weight that should be mined during period t (considering both *ore* and *waste* blocks).
- $\overline{W}_t$: maximum weight that can be mined during period t (mining equipment capacity).

To formulate the second stage, the following notation is used for each scenario:

- S: the number of scenarios used to model metal uncertainty.
- s: scenario index, $s = 1, \ldots, S$.
- o_{is}: parameter indicating the group of block i under scenario s

$$o_{is} = \begin{cases} 1 & \text{if block } i \text{ is an } \textit{ore} \text{ block under scenario } s, \\ 0 & \text{otherwise(i.e., if } i \text{ is a } \textit{waste} \text{ block under scenario } s). \end{cases}$$

- m_{is}: the metal content of block i under scenario s.
- v_{its}: the discounted economic value of block i if mined during period t, and if scenario s occurs. If we denote by d_1 the discount rate and by p_{is} the economic value of block i under scenario s, then v_{its} is given by the following formula:

$$v_{its} = \frac{p_{is}}{(1+d_1)^t}.$$

Recall that we have defined in Section 1 the economic value of a block as being the net profit associated with it. The net profit differs according to the group (*ore* or *waste*). In the first case, it is equal to the value of the metal content of the block less the mining, processing, and selling costs. In the second case, it is equal to minus the cost of mining the block. Furthermore, it is assumed that *ore* blocks are processed during the same period when they are mined and that the profit is also generated during that period.

For each period t, the following notation is used:

- $\underline{O}_t$: minimum weight of *ore* required to feed the processing plant during period t.
- $\overline{O}_t$: maximum weight of *ore* that can be processed in the plant during period t (processing plant capacity).
- $c_t^{o-} = \frac{c^{o-}}{(1+d_2)^t}$: unit shortage cost associated with the failure to meet $\underline{O}_t$ during period t (c^{o-} is the undiscounted unit shortage cost, and d_2 represents the risk discount rate).
- $c_t^{o+} = \frac{c^{o+}}{(1+d_2)^t}$: unit surplus cost incurred if the total weight of *ore* blocks mined during period t exceeds $\overline{O}_t$.
- $\underline{M}_t$: minimum amount of metal that should be produced during period t.
- $\overline{M}_t$: maximum amount of metal that can be sold during period t (metal demand).
- $c_t^{m-} = \frac{c^{m-}}{(1+d_2)^t}$: unit shortage cost associated with the failure to meet $\underline{M}_t$ during period t.
- $c_t^{m+} = \frac{c^{m+}}{(1+d_2)^t}$: unit surplus cost incurred if the metal production during period t exceeds $\overline{M}_t$.

The variables used to formulate the problem are as follows:

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

$$x_{it} = \begin{cases} 1 & \text{if block } i \text{ is mined during period } t, \\ 0 & \text{otherwise.} \end{cases}$$

- In modeling the *processing constraints*, we use the variables d_{ts}^{o-} and d_{ts}^{o+} to denote the shortage and the surplus in the amount of *ore* mined during period t if scenario s occurs, respectively.
- Finally, the variables d_{ts}^{m-} and d_{ts}^{m+} measure the shortage and the surplus in metal production during period t under scenario s, respectively.

The proposed model is summarized as follows:

$$\max \frac{1}{S}\left\{\sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{i=1}^{N} v_{its}x_{it} - \sum_{s=1}^{S}\sum_{t=1}^{T}\left(c_t^{o-}d_{ts}^{o-} + c_t^{o+}d_{ts}^{o+} + c_t^{m-}d_{ts}^{m-} + c_t^{m+}d_{ts}^{m+}\right)\right\} \quad (1)$$

$$(M) \quad \text{s.t.} \sum_{t=1}^{T} x_{it} \leqslant 1 \quad i = 1, \ldots, N \quad (2)$$

$$x_{it} - \sum_{\tau=1}^{t} x_{p\tau} \leqslant 0 \quad i = 1, \ldots, N, p \in P_i, \quad t = 1, \ldots, T \quad (3)$$

$$\sum_{i=1}^{N} w_i x_{it} \leqslant \overline{W}_t \quad t = 1, \ldots, T \quad (4)$$

$$\sum_{i=1}^{N} w_i x_{it} \geqslant \underline{W}_t \quad t = 1, \ldots, T \quad (5)$$

$$\sum_{i=1}^{N} o_{is} w_i x_{it} + d_{ts}^{o-} \geqslant \underline{O}_t \quad t = 1, \ldots, T, \quad s = 1, \ldots, S \quad (6)$$

$$\sum_{i=1}^{N} o_{is} w_i x_{it} - d_{ts}^{o+} \leqslant \overline{O}_t \quad t = 1, \ldots, T, \quad s = 1, \ldots, S \quad (7)$$

$$\sum_{i=1}^{N} o_{is} m_{is} x_{it} + d_{ts}^{m-} \geqslant \underline{M}_t \quad t = 1, \ldots, T, \quad s = 1, \ldots, S \quad (8)$$

$$\sum_{i=1}^{N} o_{is} m_{is} x_{it} - d_{ts}^{m+} \leqslant \overline{M}_t \quad t = 1, \ldots, T, \quad s = 1, \ldots, S \quad (9)$$

$$x_{it} = 0 \text{ or } 1 \quad i = 1, \ldots, N, \quad t = 1, \ldots, T \quad (10)$$

$$d_{ts}^{o-}, d_{ts}^{o+}, d_{ts}^{m-}, d_{ts}^{m+} \geqslant 0 \quad t = 1, \ldots, T, \quad s = 1, \ldots, S. \quad (11)$$

x_{it} are the first-stage decision variables. They are scenario-independent since they must be fixed before knowing the values of the uncertain parameters. The deviation variables d_{ts}^{o-}, d_{ts}^{o+}, d_{ts}^{m-}, and d_{ts}^{m+} are the second-stage (recourse) decision variables. Their values depend both on the realization of the uncertain parameters and on the values of the first-stage decision variables.

The objective function (1) includes two elements to maximize the expected net present value of the mining operation, and to minimize the expected recourse costs incurred whenever the stochastic constraints are violated due to metal uncertainty. In this presentation, we assume that all scenarios have an equal probability of occurrence and hence the coefficient $\frac{1}{S}$ represents the probability that scenario s occurs.

Constraints (2)–(5) are related to the non-stochastic constraints and thus 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) and (5) ensure that the requirements on the mining levels are respected during each period of the horizon (*mining constraints*).

Constraints (6)–(9) are related to the stochastic constraints and thus are scenario-dependent. Constraints (6) and (7) are related to the requirements on the processing levels (*processing constraints*). For each scenario s, the target is to have the total weight of *ore* blocks mined during any period t in the interval $[\underline{O}_t, \overline{O}_t]$. If it is equal to a value smaller than $\underline{O}_t$ (respectively, larger than $\overline{O}_t$), then the shortage penalty cost is equal to $c_t^{o-} d_{ts}^{o-}$ (respectively, the surplus penalty cost is equal to $c_t^{o+} d_{ts}^{o+}$). Finally, constraints (8) and (9) indicate that, during any period t, the target is to have the metal production in the interval $[\underline{M}_t, \overline{M}_t]$. Otherwise, the shortage penalty cost is equal to $c_t^{m-} d_{ts}^{m-}$ or the surplus penalty cost is equal to $c_t^{m+} d_{ts}^{m+}$ (*metal production constraints*).

Suppose that the scheduling horizon consists of a single period, that the number of scenarios is reduced to one, and that the constraints (5)–(9) are dropped. Then, our problem would reduce to the well-known *Precedence-Constrained Knapsack Problem* (PCKP) (Kellerer et al., 2004) and would be simplified considerably. But it is known that the PCKP is NP-hard (You and Yamada, 2007), which implies that the resolution of large instances with an exact method is very time consuming. Since our problem is even more complex, and since the size of real-world MPSP instances is very large, then heuristic and metaheuristic methods are indicated to address these large-scale realistic instances. We have developed a Tabu search procedure and two different diversification strategies to search the feasible domain more extensively.

3. Tabu search

In this section, we first introduce a modified version of the original model, where we allow violations of the *mining constraints* (4) and (5) at the expense of a penalty cost added to the objective function. We then describe the solution procedure.

3.1. Modified model

A solution x is encoded as an array of length N where the i^{th} entry is associated with block i and represents the period in which i is scheduled. Referring to the model (M) in Section 2, this means that $x_i = t$ if $x_{it} = 1$. Now if $x_{it} = 0$ for all $t = 1, \ldots, T$ (i.e., block i is not mined during the horizon), then x_i is set to $(T+1)$, a fictitious period. Obviously, in this case block i does not incur costs nor does it generate revenue. Furthermore, constraints (4)–(9) related to the mining, processing, and metal production levels do not apply to the fictitious period $(T+1)$.

For the other periods $t \neq (T+1)$, we allow violations of the *mining constraints* at the expense of the following penalty cost to be added to the objective function:

$$P(x) = \sum_{t=1}^{T} \left[P^{+} \max\left\{ \sum_{i=1}^{N} w_i x_{it} - \overline{W}_t, 0 \right\}^2 + P^{-} \max\left\{ \underline{W}_t - \sum_{i=1}^{N} w_i x_{it}, 0 \right\}^2 \right] \quad (12)$$

where P^{+} and P^{-} are parameters specifying how much to penalize the violation of constraints (4) and (5), respectively.

The modified model (MM) is summarized as follows:

$$\max f(x) = \frac{1}{S} \left\{ \sum_{s=1}^{S} \sum_{t=1}^{T} \sum_{i=1}^{N} v_{its} x_{it} - \sum_{s=1}^{S} \sum_{t=1}^{T} \left(c_t^{o-} d_{ts}^{o-} + c_t^{o+} d_{ts}^{o+} + c_t^{m-} d_{ts}^{m-} + c_t^{m+} d_{ts}^{m+} \right) \right\} - P(x)$$

(MM) s.t.

$$\sum_{t=1}^{T+1} x_{it} = 1 \quad i = 1, \ldots, N \quad (13)$$

and constraints (3) and (6)–(11).

The solution procedure is initiated with a feasible solution of (MM). Then, a Tabu search procedure is used to improve the quality of this solution by increasing the expected net present value and reducing constraint violations. When the Tabu search terminates, a diversification strategy is used to generate a new starting solution for the Tabu search. This process is iterated as long as the elapsed time is less than a specified maximum time *timemax*.

Note that to determine the penalty cost $P(x)$, we use the square of the deficient weight and the square of the exceeding weight rather than the deficient and the exceeding weights because it appears reasonable to prefer a solution having different periods with a small infeasibility (with respect to the original model (M)) rather than a solution having few periods with a large infeasibility. Furthermore, the values of the parameters P^{+} and P^{-} are initially set equal to 1. They are adjusted every 10 iterations, as in Gendreau et al. (1994): if the 10 previous solutions are feasible, then P^{+} and P^{-} are divided by 2; if they are all infeasible, then P^{+} and P^{-} are multiplied by 2; otherwise P^{+} and P^{-} remain unchanged.

3.2. Initial solution

The method used to generate the first initial solution is a sequential heuristic procedure consisting of $(T+1)$ major iterations. During the procedure, denote by $\mathcal{E}$ the current set of blocks that are *eligible* to be mined because they are not scheduled yet, and all their predecessors are already scheduled. At each major iteration t $(t = 1, \ldots, T)$, a certain number of blocks are selected from $\mathcal{E}$ and scheduled at period t. The set of blocks scheduled at t is denoted B_t and is built up by sequentially adding blocks from $\mathcal{E}$ as follows: Select randomly a block i among those in $\mathcal{E}$, add it to B_t, and update $\mathcal{E}$. Repeat the process until either the total weight of blocks in B_t is greater or equal than $W_t = \frac{\underline{W}_t + \overline{W}_t}{2}$, or $\mathcal{E}$ is empty. Thereafter, perform a new iteration to deal with the next period $(t+1)$. Blocks left unselected at the end of the T^{th} iteration are included in B_{T+1}.

Note that choosing blocks from $\mathcal{E}$ ensures that the solution generated by the heuristic satisfies the *reserve constraints* and the *slope constraints*. On the other hand, while some notice is taken of the satisfaction of the *mining constraints*, the other objectives and constraints are ignored. Indeed, the purpose at this stage is to quickly identify a solution satisfying the non-stochastic constraints. Now this solution is improved using the Tabu search procedure described in the following section.

3.3. Tabu search procedure

The neighborhood of a feasible solution x of the modified model (MM) is generated by shifting a block i currently scheduled at period x_i to another period $t \neq x_i$. This structure allows a new block to be inserted into the schedule (if $x_i = T+1$), or an existing block to be removed from the schedule (if $t = T+1$), or a block to be moved to a different period (if $x_i, t \neq T+1$).

The new solution generated is denoted $x \oplus (i,t)$. The shift is feasible and the solution generated belongs to the neighborhood of x if and only if the *slope constraints* are satisfied; i.e., if and only if $x_p \in [1, t]$ for all $p \in P_i$ and $x_s \in [t, T+1]$ for all $s \in S_i$ (recall that P_i and S_i denote the set of predecessors and successors of block i, respectively). Hence the shift is feasible if $t \in [e(x_i), l(x_i)]$ where $e(x_i) = \max_{p \in P_i}\{x_p\}$ (the closest predecessors of i are scheduled at $e(x_i)$) and $l(x_i) = \min_{s \in S_i}\{x_s\}$ (the closest successors of i are scheduled at $l(x_i)$). To keep track more easily of the admissible shifts for each block i, we define a $2 \times N$ matrix $\mathcal{C}$ where the elements in column i are specified as $\mathcal{C}_{1i} = e(x_i)$ and $\mathcal{C}_{2i} = l(x_i)$. The matrix $\mathcal{C}$ is easily updated at each iteration since a shift involving block i induces modifying the information in column i and in columns associated with its closest predecessors and its closest successors.

To avoid cycling, a short-term Tabu list *TL* that forbids reversing recent shifts is used. To be more specific, if we move from x to $x \oplus (i,t)$, then we forbid block i to be scheduled at x_i (the shift

(i, x_i) is declared Tabu) during the next θ iterations, where θ is a random integer number chosen in $[\theta_{min}, \theta_{max}]$. However, a Tabu shift can be applied if it leads to a solution better than the best solution found so far denoted *xbest* (classical aspiration criterion).

At each iteration, we select one of the best non-Tabu neighbor solutions or one of the best Tabu neighbor solutions satisfying the aspiration criterion to be the current solution for the next iteration. The modification value $\Delta_x(i,t) = f(x \oplus (i,t)) - f(x)$ associated with a neighbor solution $x \oplus (i,t)$ is easy to evaluate since only periods x_i and t are affected by the shift (i,t).

Note that there may exist several shifts with the same best modification value. To break ties, we use a secondary selection criterion based on a frequency memory. To be more specific, denote by $\mathcal{F} = [\mathcal{F}_{it}]$ an $N \times (T+1)$ frequency matrix. Each entry $\mathcal{F}_{it}$, associated with the pair of block i and period t, represents the number of times that i has been scheduled at t since the beginning of the solution process. This frequency matrix is updated whenever a shift (i,t) is applied by increasing the value of the entry $\mathcal{F}_{it}$ by 1. To break ties, the best candidate solution $x \oplus (i,t)$ with the smallest frequency value $\mathcal{F}_{it}$ is selected. This secondary selection criterion can be seen as a diversification strategy used to drive the search towards less explored regions of the search space.

Finally, the Tabu search procedure terminates when the number of successive non-improving iterations reaches a specified value *nitermax*. Denote by x^* the current best solution generated by the Tabu search. If x^* is better than the solution *xbest* generated in previous applications of the search (the best solution found so far), then *xbest* is replaced by x^*.

4. Diversification strategies

Two different diversification strategies are compared numerically. The first strategy exploits a long-term memory of the search history while the second one is based on the variable neighborhood search method. If the time elapsed is smaller than *timemax*, these strategies can be used to generate a new initial solution x^0 to reinitialize the Tabu search procedure.

4.1. Long-term memory diversification strategy

Consider the current best solution x^*. Since the purpose of any diversification strategy is to search the feasible domain more extensively, one way to achieve this is by moving some blocks to the periods where they rarely have been scheduled to date. To identify such periods, we refer to the frequency matrix $\mathcal{F} = [\mathcal{F}_{it}]$ defined in Section 3.3. For each block i, let $t_i = \underset{t \neq x_i^*, T+1}{\operatorname{argmin}}\{\mathcal{F}_{it}\}$ be the period (different from x_i^* and $(T+1)$) at which i has been scheduled the least frequently, and denote $\varphi_i = \mathcal{F}_{it_i}$.

To generate the new initial solution x^0, start with $x^0 := x^*$. Associate with each block i a probability of being selected inversely proportional to its φ_i value. Select a block j randomly according to these probabilities, and move it to period t_j (i.e., set $x^0 := x^0 \oplus (j, t_j)$). Now such a shift may create infeasibility. Indeed, if $t_j > x_j^*$, it may happen that some successors s of j are now scheduled before t_j in x^0 (i.e., $x_s^0 < x_j^0$). On the other hand, if $t_j < x_j^*$, then some predecessors p of j may be now scheduled after t_j in x^0 (i.e., $x_p^0 > x_j^0$). In this case, we apply the following sequential process that allows the retrieval of feasibility.

To illustrate the process, consider the case where $t_j > x_j^*$. Let $\Gamma = \left\{ s \in S_j : x_s^0 < x_j^0 \right\}$. At each iteration, select randomly s in Γ. Identify $\alpha_s = \max_{k \in P_s, k \notin \Gamma} \{x_k^0\}$ and $\beta_s = \min_{l \in S_s, l \notin \Gamma} \{x_l^0\}$ the periods at which the closest predecessors and the closest successors of s are scheduled, but without considering the predecessors and the successors of s that are in Γ. Determine a period τ_s in $[\alpha_s, \beta_s]$ where s has been less frequently scheduled: $\tau_s = \underset{t \in [\alpha_s, \beta_s]}{\operatorname{argmin}}\{\mathcal{F}_{st}\}$. Move s to τ_s (i.e., set $x^0 := x^0 \oplus (s, \tau_s)$). Then eliminate s from Γ. The process terminates when the set Γ is empty.

Note that during the diversification stage, each time a shift (i,t) is applied, the Tabu list, initially empty, is updated as described in Section 3.3 in order to avoid returning to x^* during the Tabu search. The frequency matrix $\mathcal{F}$ is also updated.

4.2. Variable neighborhood diversification strategy

This strategy relies on principles found in the basic variable neighborhood search method presented by Hansen and Mladenovic (2001). A set of k_{max} neighborhood structures $N^k (k = 1, \ldots, k_{max})$ has to be specified a priori. In our implementation, the structure N^1 is the neighborhood used in the Tabu search described in Section 3.3. $N^k (k \geqslant 2)$ are straightforward extensions of N^1: given a solution x, the elements of $N^k(x)$ are generated by successively applying k shifts to x. In doing so, we take care not to move a block more than once.

To modify the value of k in order to determine the neighborhood used to specify the diversification, we proceed as follows: When the diversification strategy is applied for the first time, k is set equal to 1. Then, whenever the Tabu search is completed with the current best solution x^*, we modify the value of k as follows: If x^* is better than *xbest*, then $xbest := x^*$ and $k := 1$; otherwise, *xbest* is not modified and

$$k := \begin{cases} k+1 & \text{if } k < k_{max}, \\ 1 & \text{if } k = k_{max}. \end{cases}$$

To specify the new initial solution x^0, consider the best solution *xbest*, and determine $x^0 \in N^k(xbest)$ (the neighborhood of *xbest* in the structure N^k). Start with $x^0 := xbest$. Let $\mathcal{B} = \{ i : e(x_i^0) \neq l(x_i^0) \}$ be the set of blocks i that can be moved from their current period x_i^0 to a new period where the *slope constraints* can still be satisfied.

The selection of the blocks to be moved relies on additional information included in a N-dimensional vector $\mathcal{T}$ where the i^{th} component $\mathcal{T}_i$ is associated with block i. This vector is updated at each iteration of the Tabu search or the diversification strategy as follows: Whenever a shift (i,t) is applied, then $\mathcal{T}_i := \mathcal{T}_i + 1$; i.e., $\mathcal{T}_i$ indicates the number of times that block i has been moved.

At each iteration of the diversification strategy, select a block j in the set $\mathcal{B}$. The selection is probabilistically biased towards blocks with lower values of $\mathcal{T}_i$. Next, consider the periods in $\left[e\left(x_j^0\right), l\left(x_j^0\right) \right]$ feasible for block j. Select a period t in this interval either randomly or in a greedy manner by selecting the period inducing the best improvement of the objective function. Then, move block j to t (i.e., set $x^0 := x^0 \oplus (j,t)$) and update the set $\mathcal{B}$, the array $\mathcal{T}$, and the Tabu list, which is initially empty. This process is repeated k times.

5. Numerical results

Two different sets of problems P_1 and P_2 are used to complete the numerical experimentation. Each set includes 5 different problems based on real-life data from our industry partners. Problems in P_1 are from a copper deposit where blocks are of size $20 \times 20 \times 10$ meters and weigh 10,800 tons each. Problems in P_2 are from a gold deposit where blocks are of size $15 \times 15 \times$ 10 meters and weigh 5625 tons each. The 10 problems are specified in Table 1.

The largest problem in each set is a real-life size problem (problems C5 and G5). The N blocks in these problems are those within the *pit limits* corresponding to the blocks to mine so as to maximize

Table 1
Characteristics of the problems in the two data sets.

Set	Problem	D	Number of blocks (N)	Number of periods (T)	Number of scenarios (S)
P_1 Metal type: copper	C1	20,000	4273	3	20
Block size: 20 × 20 × 10 meters Block weight: 10,800 tons	C2	15,000	7141	4	20
	C3	10,000	12,627	7	20
	C4	5000	20,626	10	20
	C5	0	26,021	13	20
P_2 Metal type: gold	G1	20,000	18,821	5	20
Block size: 15 × 15 × 10 meters Block weight: 5625 tons	G2	15,000	23,901	7	20
	G3	10,000	30,013	8	20
	G4	5000	34,981	9	20
	G5	0	40,762	11	20

the profit expected from the mining operation while satisfying the *slope constraints*. These blocks were identified by solving the following problem:

$$\max \sum_{i=1}^{\overline{N}} v_i y_i \quad (14)$$

(PL) s.t.

$$y_i \leqslant y_p \quad i = 1, \ldots, \overline{N}, \quad p \in P_i \quad (15)$$

$$y_i = 0 \text{ or } 1 \quad i = 1, \ldots, \overline{N} \quad (16)$$

where $\overline{N}$ is the number of blocks in the mineral deposit, and $v_i = \frac{\sum_{s=1}^{S} p_{is}}{S}$ is the expected economic value of block i. Recall that p_{is} denotes the economic value of block i under scenario s, and S denotes the number of scenarios used to model metal uncertainty. P_i the set of predecessors of block i. $y_i = \begin{cases} 1 & \text{if block } i \text{ is part of the pit,} \\ 0 & \text{otherwise.} \end{cases}$

For the other problems (C1–C4 and G1–G4), the blocks considered were also obtained by solving the mathematical model (PL); however, the economic values of the blocks have been decreased in order to obtain smaller pits (i.e., in the objective function (14), v_i is replaced by $(v_i - D)$ where the decreasing factor D is given in the third column of Table 1). The purpose of using problems C1–C4 and G1–G4 is twofold: to assess how the proposed solution method scales with the problem size, and to determine the limit of problem size up to which the commercial solver CPLEX is able to solve the problem in a reasonable amount of time.

For each deposit (copper and gold), 20 equiprobable scenarios representing the mineral deposits were generated from a limited number of drilling information and the geostatistical techniques of conditional simulation (Goovaerts, 1997; Scheidt and Caers, 2009; Horta and Soares, 2010; Chiles and Delfiner, 2012), which can be seen as a complex Monte Carlo simulation framework. The scenarios generated reproduce all available data and information as well as spatial statistics of the data. The specific technique utilized herein is retailed in Boucher and Dimitrakopoulos (2009).

The economic parameters (unit costs, unit revenues, discount factor) are based on real-life data and are summarized in Table 2. The reasoning behind the choice of the values of the undiscounted shortage and surplus costs is as follows:

- If the *ore* produced during period t is less than $\underline{O}_t$, there is a shortage, and the mining company "loses" some of the capital invested to build a processing plant with a large capacity. The investment is typically equal to \$15/t and \$17/t per year for copper and gold, respectively. Hence, we use these values for the undiscounted shortage cost for *ore*, c^{o-}.

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

Parameters	P_1	P_2
Mining cost	\$1/t	\$1/t
Processing cost	\$9/t	\$15/t
Metal price	\$0.125/oz	\$900/oz
Selling cost	\$1.875E−02/oz	\$7/oz
Undiscounted shortage cost for ore (c^{o-})	\$15/t	\$17/t
Undiscounted surplus cost for ore (c^{o+})	\$15/t	\$17/t
Undiscounted shortage cost for metal (c^{m-})	\$1.25E−02/oz	\$90/oz
Undiscounted surplus cost for metal (c^{m+})	\$6.25E−03/oz	\$45/oz
Discount rate (d_1)	10%	10%
Risk discount rate (d_2)	10%	10%

- If the amount of *ore* mined during period t exceeds the processing plant capacity, the mining company must increase the plant capacity at a cost of \$15/t and \$17/t for copper and gold, respectively. Hence, we use these values for the undiscounted surplus cost for ore, c^{o+}.
- If there is a shortage of metal during period t, and the contract with the buyer stipulates that a penalty of 10% of the value of the "undelivered" metal is to be charged, then the undiscounted shortage cost for the metal, c^{m-}, is defined as: $0.1 \times$ metalprice.
- In case of an excess of metal production over demand, metal prices may fall. Assuming sales of the excess metal at a price 5% below the original price, the undiscounted surplus cost for metal, c^{m+}, is defined as: $0.05 \times$ metal price.

Finally, for the 10 problems, each period is one year long. Lower and upper bounds on processing at each period are defined as: $\frac{\text{total expected amount of ore}}{\text{number of periods}} \pm 5\%$. The same margin is considered for metal production levels, while a higher margin (20%) is considered for mining levels.

The numerical tests were completed on an AMD Opteron 250 computer (2.4 gigahertz) with 16 gigabytes of RAM running under Linux. Preliminary tests were conducted to determine appropriate parameter values for the different procedures introduced in this paper. On the basis of the results of these tests, we have chosen the following values that exhibit a good overall performance:

- Parameters of the solution procedure in Section 3.1:
  - Maximum allotted time: $timemax = 0.02NT$ seconds (N being the number of blocks in the problem and T the number of periods over which blocks are being scheduled).
- Parameters of the Tabu search procedure in Section 3.3:
  - Interval in which the duration of the Tabu status of the shifts is chosen: $[\theta_{min}, \theta_{max}] = [\lfloor 0.8\eta \rfloor, \lceil 1.2\eta \rceil]$, where η denotes the

number of blocks that can be moved from their current period in the initial solution x^0 to a new period where the *slope constraints* can still be satisfied (i.e., blocks i such that $e(x_i^0) \neq l(x_i^0)$).
 – Maximum number of successive non-improving iterations: $nitermax = \eta$.
- Parameters of the variable neighborhood diversification strategy in Section 4.2:
 – Maximal number of neighborhood structures: $k_{max} = \lceil \sqrt{N} \rceil$.
 – Probability of choosing the random strategy: $\alpha = 0.3$. Recall that in the procedure described in Section 4.2, once a block j is selected in the set $\mathcal{B}$, it is moved to a period $t \in \left[e\left(x_j^0\right), l\left(x_j^0\right)\right]$ chosen either randomly or in a greedy manner. The period t is chosen at random with a probability of $\alpha \in [0, 1]$. The selection bias in favor of the best shifts can therefore be increased by setting the α value closer to 0, or be reduced by setting α closer to 1.

In the following, **TS-LTM** and **TS-VN** denote the two variants of the solution procedure in Section 3.1, obtained by combining the Tabu search procedure in Section 3.3 (denoted **TS**) with the two diversification strategies in Section 4 (the long-term memory diversification strategy and the variable neighborhood diversification strategy, denoted **LTM** and **VN**, respectively). We compare the variants numerically with the 10 problems specified in Table 1, using the values above for the parameters. Since both variants contain stochastic features, each problem is solved 10 times by each variant using different initial solutions. All reported results correspond to averages over the 10 runs.

In Table 3, we compare the variants globally in terms of the average value *Zbest* of the best solutions generated, the average value *NPV* of the expected net present value for the best solutions found (i.e., the average value of the first term of the objective function (1)), and the average number *IterDiv* of diversification iterations performed (i.e., number of times that new solutions have been generated to reinitialize the Tabu search). Note that we do not compare CPU times since they are equivalent (both variants stop after $timemax = 0.02NT$ seconds). Furthermore, for the first two criteria (*Zbest* and *NPV*), we report the relative performance of **TS-LTM** over **TS-VN** (the ratio of the value obtained by **TS-LTM** to that obtained by **TS-VN**) rather than the values obtained by the two variants.

As can be observed from the results in Table 3, **TS-LTM** always produces better results than **TS-VN**. The performance differences

Table 3
Comparison between **TS-LTM** and **TS-VN**.

Set	Problem	*NPV*($)	*Zbest*($)	*IterDiv*	
		$\frac{\text{TS-LTM}}{\text{TS-VN}}$	$\frac{\text{TS-LTM}}{\text{TS-VN}}$	**TS-LTM**	**TS-VN**
P_1	C1	1.003	1.002	867.7	766.9
	C2	1.008	1.005	664.3	744.6
	C3	1.027	1.025	184.6	346.9
	C4	1.033	1.031	103.0	253.0
	C5	1.077	1.193	57.3	111.2
P_2	G1	1.028	1.186	739.9	1346.4
	G2	1.117	1.909	394.3	430.2
	G3	1.157	2.286	239.1	204.1
	G4	1.176	2.230	149.3	156.7
	G5	1.217	2.258	68.5	113.4

increase with the problem size. In particular, for the smallest problem C1, *NPV* and *Zbest* are improved respectively by 0.30% and 0.16% when **TS-LTM** is used. Even higher improvements of 21.70% and 125.82%, respectively, are obtained for the largest problem, G5.

The values of *IterDiv* indicate that, in general, **TS-VN** restarts the Tabu search procedure more frequently than **TS-LTM**. This can be explained by two reasons. First, **VN** is often less time consuming than **LTM** (cf. Sections 4.1 and 4.2). Second, the results indicate that as **TS-VN** progresses, the number of **TS** iterations performed between two diversifications tends towards *nitermax*, which means that the Tabu search fails to improve the new initial solution, much less the best solution found so far. **TS-LTM** often exhibits the opposite behavior and performs more than *nitermax* iterations between two diversifications.

To summarize, the **VN** diversification strategy does not appear to be very effective for escaping local optima. We think that the fact that this strategy generates solutions in the neighborhood of the best solution found so far *xbest* (cf. Section 4.2) is the main reason for the poor performance of **TS-VN**. Indeed, the consequence of this is that the search is performed in a restricted region of the feasible domain (around *xbest*), and it fails to explore other regions that may contain better solutions. On the other hand, **LTM** generates solutions that are significantly different from *xbest* (cf. Section 4.1). Since it is based on the frequency matrix, which enables keeping track of the entire history of the search, this strategy drives the search towards unexplored or less explored regions of the feasible domain and thus provides better opportunities for an extensive exploration of it.

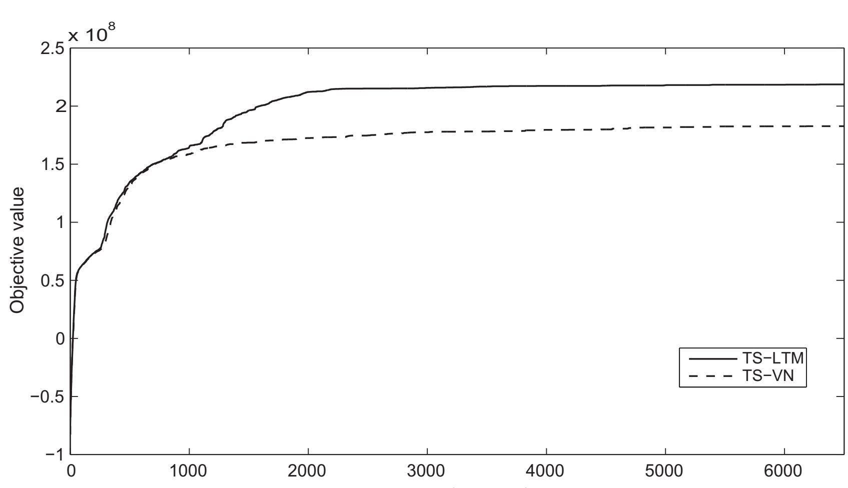

Fig. 1. Evolution of the best solution found over time for problem C5.

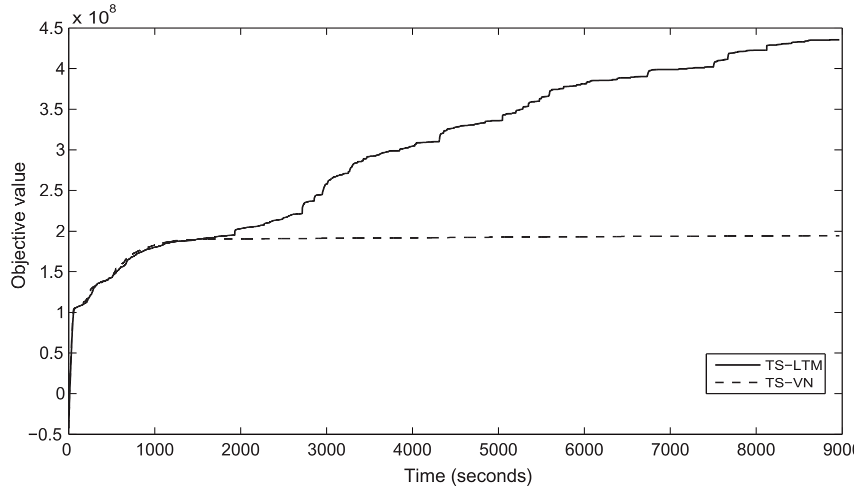

Fig. 2. Evolution of the best solution found over time for problem G5.

Figs. 1 and 2 illustrate even more strongly that the use of the **VN** diversification strategy is not very useful for escaping local optima. These figures show the results for the largest problems in each set (problems C5 and G5, respectively). Each curve associated with a variant indicates the average values of the best solutions generated over the 10 resolutions calculated at different times of the resolution.

Globally, for problem C5 (cf. Fig. 1), the curves associated with the two variants have almost the same shape. During the first 200 seconds, they are exactly alike (recall that the initial solutions are identical and that they are improved using the same Tabu search procedure). During the period between 200 and 1000 seconds, the values of the objective function continue to increase, but **TS-LTM** performs slightly better than **TS-VN**. The gap between the two variants increases during the period between 1000 and 2000 seconds. Improvements rarely occur with **TS-VN**, indicating that this variant can easily be trapped in local optima. After 2000 seconds, the curve associated with **TS-LTM** also levels, but at higher values than **TS-VN**.

For problem G5, the curves in Fig. 2 indicate that **TS-LTM** consistently improves the quality of the solutions found, while **TS-VN** has the same behavior as for problem C5: after improving the solution at the beginning of the process, it reaches a point where it cannot move out of local maxima.

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 C5 and G5, for which CPLEX was unable to solve the linear relaxation within 4 weeks, we indicate the value of the average relative gap $\%Gap$ between the average value *Zbest* of the best solutions generated (as reported in Table 3) and the linear relaxation optimal value Z_{LR}:

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

The standard deviation of $\%Gap$ for the 10 runs is indicated in parentheses. The CPU times required by **TS-LTM**, **TS-VN**, and CPLEX are given in minutes in the last three columns of the table. Note that CPU times for **TS-LTM** and **TS-VN** are identical since both stop after $timemax = 0.02NT$ seconds.

Notice first the effectiveness of **TS-LTM** in producing very good quality solutions for all problem sizes. The results indicate that $\%Gap$ tends to increase with the problem size (this is probably due to the fact that *IterDiv* decreases with the problem size). However, its average value is only 1.80%, and in 92.5% of all cases (considering the 80 runs), it is smaller than 4%. The small values of standard deviation indicate even more clearly the robustness of **TS-LTM**. **TS-VN** is not competitive with **TS-LTM**, especially when the problem size increases (problems in P_2). The results also show that both variants require significantly shorter solution times than CPLEX. In particular, for problems C4 and G4, the computational times for **TS-LTM** and **TS-VN** are only about 1 and 2 hours, respectively, while the CPU time required by CPLEX to solve the linear relaxation of the problem is approximately 9 and 11 days, respectively.

Table 4
Comparison between **TS-LTM**, **TS-VN** and **CPLEX**.

Set	Problem	%Gap		CPU time (minutes)		
		TS-LTM	**TS-VN**	**TS-LTM**	**TS-VN**	**CPLEX**
P_1	C1	0.23 (0.02)	0.40 (0.08)	4.28	4.28	8.97
	C2	0.67 (0.06)	1.17 (0.19)	9.53	9.53	73.64
	C3	1.98 (0.21)	4.39 (0.17)	29.47	29.47	1,457.05
	C4	4.17 (0.49)	7.09 (0.18)	68.77	68.77	12,115.63
	C5	–	–	112.77	112.77	–
P_2	G1	1.15 (0.05)	16.64 (14.15)	31.38	31.38	855.50
	G2	1.72 (0.06)	48.52 (7.19)	55.78	55.78	3,786.73
	G3	2.07 (0.11)	57.17 (2.51)	80.05	80.05	7,902.27
	G4	2.40 (0.21)	56.23 (1.97)	104.95	104.95	15,230.50
	G5	–	–	149.47	149.47	–

Table 5
Comparison between **TS-LTM**, **TS-VN**, and pure **TS**.

Set	Problem	G Zbest		G %Gap	
		TS-LTM	**TS-VN**	**TS-LTM**	**TS-VN**
P_1	C1	1.004	1.003	2.797	1.629
	C2	1.001	0.996	1.106	0.635
	C3	1.023	0.998	2.104	0.947
	C4	1.033	1.001	1.726	1.015
	C5	1.039	0.871	–	–
P_2	G1	0.996	0.840	0.668	0.047
	G2	1.005	0.526	1.234	0.045
	G3	1.950	0.853	23.166	0.871
	G4	1.935	0.868	20.504	0.881
	G5	1.834	0.812	–	–

To evaluate the impact of the two proposed diversification strategies, we solve the same problems without them; i.e., using a pure Tabu search. To be more specific, each problem is solved 10 times using the Tabu search procedure in Section 3.3 (**TS**), but instead of stopping the procedure when the number of consecutive iterations without improving *xbest* reaches the value *nitermax*, the procedure is stopped when the CPU time reaches $timemax = 0.02NT$ seconds (i.e., using the same stopping criterion as for **TS-LTM** and **TS-VN**). Table 5 includes results comparing the two approaches where:

- *G Zbest* = (value of *Zbest* with diversification)/(value of *Zbest* without diversification).
- *G* $\%Gap$ = (value of $\%Gap$ without diversification)/(value of $\%Gap$ with diversification).

The numerical results in Table 5 indicate that the **LTM** diversification strategy has a positive impact in improving the quality of the solutions (except for problem G1). On the other hand, using the **VN** diversification strategy has a negative impact, in general. To assess whether these differences in performance are statistically significant, for each pair of methods, we apply Wilcoxon signed-rank tests (see the website http://www.R-project.org) with a 5% level of confidence over their numerical results. These tests confirm the results in Tables 4 and 5, indicating that **TS-LTM** outperforms both **TS** and **TS-VN**. On the other hand, **TS** is statistically better than **TS-VN**. So overall, with regards to solution quality, **TS-LTM** would rank first, **TS** second, and **TS-VN** last. Note that among the three methods, **TS-VN** is the only one that focuses the search around *xbest*. Therefore, these results corroborate the observations made above: for the problem studied in this paper, focusing the search around the best solution found so far is not the best strategy to escape local optima.

To gain a deeper insight into the performance of the three methods, we conduct additional experiments and perform an analysis of run-time distributions. Run-time distributions (RTDs) or time-to-target plots (TTT plots) are useful tools to characterize the running times of stochastic algorithms for combinatorial optimization and have been widely used as a tool for algorithm design and comparison. They display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis (Ribeiro et al., 2011). The methodology used to produce the TTT plots can be summarized as follows: *n* independent runs on a given instance are performed. For each run, the CPU time required to obtain a solution with a value at least as good as a given target value is recorded. After sorting these times in ascending order, a proba-

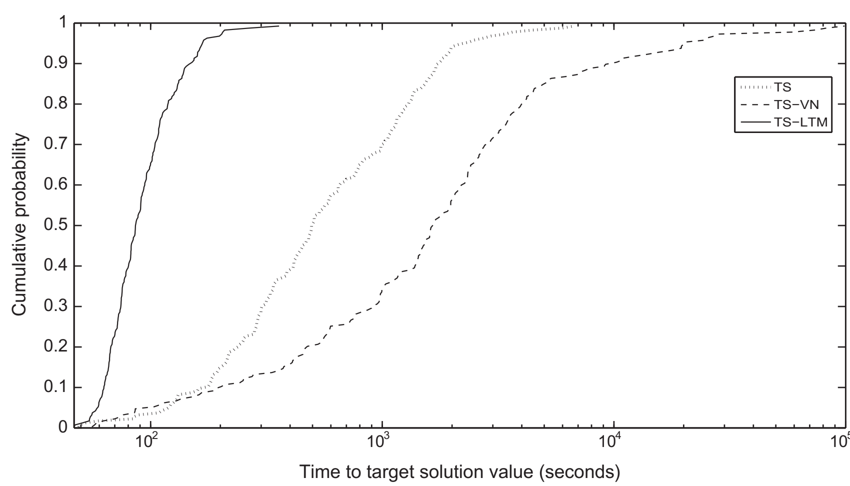

Fig. 3. Superimposed run time distributions of **TS-LTM**, **TS-VN**, and pure **TS** for problem C3.

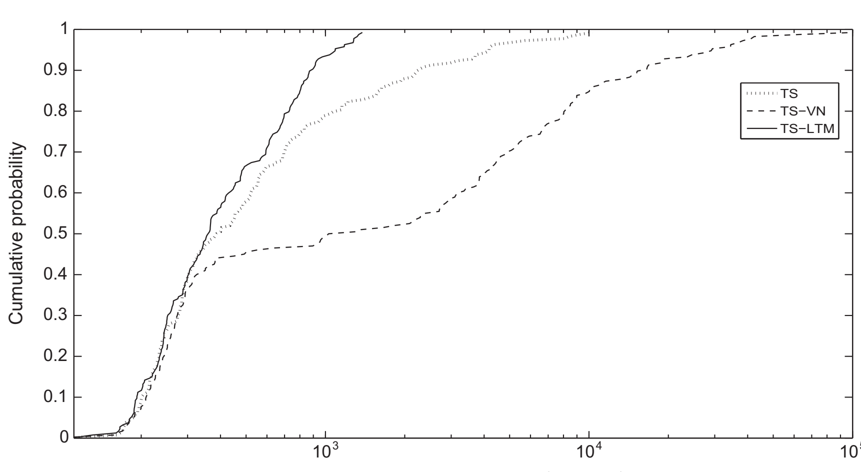

Fig. 4. Superimposed run time distributions of **TS-LTM**, **TS-VN**, and pure **TS** for problem G3.

bility $p_i = \frac{i-0.5}{n}$ is associated with the i^{th} sorted running time t_i. Then, the points (t_i, p_i) are plotted. Next, we present illustrative results for two problems: problem C3 from the set P_1 and problem G3 from the set P_2.

For each problem, each method (**TS**, **TS-LTM**, and **TS-VN**) was run 200 times with different seeds (thus different initial solutions). The target was set to the value of the best solution found by the worst method, **TS-VN**. Run-time distributions of the three methods were obtained using the Perl program described in Aiex et al. (2007). They are superimposed in Figs. 3 and 4.

From these figures, we can observe that **TS-LTM** is more likely to find a target solution faster than the other two methods. For instance, for problem C3, Fig. 3 shows that the probability of finding the target value in at most 2 minutes is approximately 5% for both **TS** and **TS-VN**, while it is 80% for **TS-LTM**. For the larger problem, G3, Fig. 4 shows that the probabilities of reaching the target value in at most 15 minutes are approximately 48%, 78%, and 95% for **TS-VN**, **TS**, and **TS-LTM**, respectively. The analysis of the run-time distributions leads therefore to results that are consistent with the previous ones obtained using the other comparison criteria, in the sense that **TS** as well as **TS-LTM** perform much better than **TS-VN**, and that **TS-LTM** outperforms **TS**. It also indicates that **TS-LTM** is robust, and the quality of the initial solution does not make any significant difference in terms of the quality of the final solution.

In conclusion, it seems to be worth using the proposed approach to solve the open-pit mine production scheduling problem with metal uncertainty. The results indicate that when problems are relatively small, the three methods introduced in this paper can generate very good solutions. For larger problems, the comparison indicates that **TS-LTM** is more effective and robust than **TS-VN** and **TS**. Solution times required by the three methods are reasonable even for problems of large size. For problems C5 and G5 of realistic size, CPU times are approximately 2 and 2.5 hours, respectively, while CPLEX is not able to solve even the linear relaxation of any of these problems within 4 weeks.

6. Conclusions

In this paper, we have proposed a metaheuristic method based on a Tabu search procedure to solve an important real-world problem that arises in surface mine planning, namely the open-pit mine production scheduling problem with metal uncertainty. To search the feasible domain more extensively, we have used two different diversification strategies to generate several initial solutions to be optimized by the Tabu search procedure. The first diversification strategy, **LTM**, exploits a long-term memory of the search history. The second one, **VN**, relies on the variable neighborhood search method.

Two variants of the solution method **TS-LTM** and **TS-VN**, obtained by combining the Tabu search procedure **TS** with the two diversification strategies **LTM** and **VN**, respectively, were compared on 10 problems based on real-life data and having different sizes. This comparison shows that for problems of relatively small sizes, **TS-VN** produces solutions as good as **TS-LTM**. It is, however, not competitive with **TS-LTM** on larger problems, nor is it competitive with a pure Tabu search.

The upper bounds provided by CPLEX allowed us to evaluate the quality of the solutions generated by the two variants. Results indicate that the variant **TS-LTM** performs very well on all the tested problems and is very robust. Indeed, the gap between the solutions generated by this variant and the upper bounds obtained using CPLEX is smaller than 4% in 92.5% of all runs. Results also indicate that the computational times of the proposed solution method are reasonable and significantly smaller than those required by CPLEX to solve the linear relaxation of the problem.

Another interesting feature of the proposed solution method is its flexibility. Even though it is introduced for the specific problem studied in this paper and the specific approach used to address metal uncertainty, it can be easily adapted to deal with other uncertainty modeling approaches and additional scheduling constraints. Future research will be devoted to adapting it to solve more complex versions of the problem that include additional operational constraints and other sources of uncertainty. We are also interested in designing parallel algorithms to improve the performance of the solution method for large problems.

Another important research direction is the development of other efficient solution approaches. Since it has been observed empirically that the problem formulation often achieves small integrality gaps, one approach could be to solve the linear relaxation of the problem using an efficient algorithm and then to use an LP-rounding procedure to get an integer solution. Similar approaches are proposed in Meagher (2010) and Moreno et al. (2010) to solve the deterministic version of the open pit mine production scheduling problem.

Acknowledgments

The work in this paper was funded by NSERC CDR Grant 335696, NSERC Discovery Grant 239019, and the industry members of the COSMO Stochastic Mine Planning Laboratory: AngloGold Ashanti, Barrick, BHP Billiton, De Beers, Newmont, and Vale. Thanks are in order to Brian Baird, Peter Stone, Darren Dyck, and Gavin Yates of BHP Billiton for their support, collaboration, and technical comments.

References

Aiex, R., Resende, M., Ribeiro, C., 2007. TTT plots: a perl program to create time-to-target plots. Optimization Letters 1, 355–366.

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

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

Bley, A., Boland, N., Fricke, C., Froyland, G., 2010. A strengthened formulation and cutting planes for the open pit mine production scheduling problem. Computers & Operations Research 37 (9), 1641–1647.

Boland, N., Dumitrescu, I., Froyland, G., 2008. A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology. Optimization Online <http://www.optimization-online.org/DB_FILE/2008/10/2123.pdf> (last accessed 06.06.11).

Boucher, A., Dimitrakopoulos, R., 2009. Block simulation of multiple correlated variables. Mathematical Geosciences 41 (2), 215–237.

Chiles, J.P., Delfiner, P., 2012. Geostatistics: Modeling Spatial Uncertainty, Second ed. John Wiley & Sons.

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

Dagdelen, K., Johnson, T.B., 1986. Optimum open pit mine production scheduling by lagrangian parameterization. In: Proceedings of 19th International APCOM Symposium, Littleton, CO, pp. 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., Farrelly, C.T., 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.A., 1994. Risk assessment in reserve estimation and open-pit planning. Transactions of the Institution of Mining and Metallurgy 103, A148–A154.

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

Glover, F., Laguna, M., 1998. Tabu Search. Kluwer Academic Publishers.

Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Management Science 40 (10), 1276–1290.

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.

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

Golamnejad, J., Osanloo, M., Karimi, B., 2006. A chance-constrained programming approach for open pit long-term production scheduling in stochastic environments. The Journal of the South African Institute of Mining and Metallurgy 106, 105–114.

Goovaerts, P., 1997. Geostatistics for Natural Resources Evaluation. Oxford University Press.

Hansen, P., 1986. The steepest ascent mildest descent heuristic for combinatorial programming. In: Presented at Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: principles and applications. European Journal of Operational Research 130 (3), 449–467.

Horta, A., Soares, A., 2010. Direct sequential co-simulation with joint probability distributions. Mathematical Geosciences 42 (3), 269–292.

Johnson, T.B., 1969. Optimum production scheduling. In: Proceedings of the 8th International Symposium on Computers and Operations Research, Salt Lake City, pp. 539–562.

Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack Problems. Springer-Verlag.

Meagher, C.J., 2010. On the direct cut polyhedral and open pit mining. Ph.D thesis, McGill University, Canada.

Menabde, M., Froyland, G., Stone, P., Yeates, G., 2007. Mining schedule optimization for conditionally simulated orebodies. The Australasian Institute of Mining and Metallurgy, Spectrum Series 14, 379–384.

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.

Newman, A.M., Rubio, E., Caro, R., Weintraub, A., Eurek, K., 2010. A review of operations research in mine planning. Interfaces 40 (3), 222–245.

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., 2007. Stochastic optimization of long term production scheduling for open pit mines with a new integer programming formulation. The Australasian Institute of Mining and Metallurgy, Spectrum Series 14, 385–392.

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

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

Ribeiro, C.C., Rosseti, I., Vallejos, R., 2011. Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms. Journal of Global Optimization. http://dx.doi.org/10.1007/s10898-011-9769-z.

Scheidt, C., Caers, J., 2009. Representing spatial uncertainty using distances and kernels. Mathematical Geosciences 41 (4), 397–419.

Souza, M.J.F., Coelho, I.M., Ribas, S., Santos, H.G., Merschmann, L.H.C., 2010. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European Journal of Operational Research 207, 1041–1051.

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

Topal, E., Ramazan, S., 2010. A new MIP model for mine equipment scheduling by minimizing maintenance cost. European Journal of Operational Research 207, 1065–1071.

You, B., Yamada, T., 2007. A pegging approach to the precedence-constrained knapsack problem. European Journal of Operational Research 183, 618–632.