Les Cahiers du GERAD ISSN: 0711–2440

Parallel implementation of a tabu search procedure for stochastic mine scheduling

R. Senécal
R. Dimitrakopoulos

G–2014–59

August 2014

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

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

Parallel implementation of a tabu search procedure for stochastic mine scheduling

Renaud Senécal
Roussos Dimitrakopoulos

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

renaud.senecal2@mail.mcgill.ca
roussos.dimitrakopoulos@mcgill.ca

August 2014

Les Cahiers du GERAD
G–2014–59

Abstract: This paper presents a metaheuristic solution to the optimization of open pit long-term production scheduling with a stockpile and geological uncertainty. The optimization formulation is a two stage stochastic integer programming (SIP) model, which determines the optimal mining sequence that maximizes the total discounted cash flow, while penalizing for high deviations from production targets. A parallel implementation of Tabu Search is proposed to accelerate the solution of the SIP formulation and take full advantage of multi-core computer processing. Different variants of the proposed algorithm are applied at a case study to assess the performance of the parallel approach. The proposed algorithm and variants are benchmarked using linear relaxation of the complete problem to determine robustness and provide a better overview of the related performance. The results show a net improvement over the sequential solution and the new proposal seems to be promising when working with a large scale data set.

Key Words: Open pit mine optimization, production scheduling, uncertain supply, Lagrangian relaxation, sub-gradient method, branch and cut algorithm.

1 Introduction

The open pit mine production scheduling problem (OPMPSP) consists of finding the optimal extraction sequence such that the net present value (NPV) is maximized, while respecting the processing and capacity constraints. Over the past several decades, many authors have proposed formulations to solve the OPMPSP in a more exact manner such as in Gershon (1983), Dagdelen and Johnson (1986), Tolwenski and Underwood (1996), Ramazan (2007), Cacetta and Hill (2003). More recently, Meagher (2011) and Bley et al. (2010) has been working on partially ordered knapsack and preprocessing in order to propose a more efficient exact algorithm. Another point of interest is when considering a stockpile with OPMPSP, leading to a natural non-convex non-linear formulation. Since this problem is computationally hard, Bley et al. (2012) proposed an algorithm where they first use a tight linear formulation of the problem and, then, apply a branch and cut algorithm to restrict the maximum violation of the quadratic constraints. Finally, they use a heuristic to transform this integer feasible solution of the relaxed formulation to create a solution that satisfies both the integrality and the quadratic constraints. In the above solutions, all values for a block are treated as if known, without any geological uncertainty, and their approach only looks at maximizing the NPV. However, studies have shown that this assumption can be a major factor in failing mining projects (Vallee, 2000) and it can also be one of the leading factors for not meeting production targets (Baker and Giacomo, 1998). Since then, stochastic optimization frameworks have been developed to assess mineral uncertainty. Ravenscroft (1992) and Dowd (1997) proposed a way of using sequentially simulated stochastic scenarios to assess the risk of a schedule, and Dimitrakopoulos and Ramazan (2004) proposed a risk oriented scheduling approach based on mixed integer programming (MIP).

To overcome considering the values representing mining blocks known and constant, stochastic integer programming (SIP) for OPMPSP is developed. OPMPSP can be formulated as a two stage stochastic integer programming model (Birge and Louveaux, 1997) where the first stage consists of providing the block extraction sequence and the second stage is used to control the deviation from production targets according to each scenario (which accounts for the uncertainty of the grade, metal, ore quality and/or ore tonnages). Ramazan and Dimitrakopoulos (2005) first present a SIP formulation for OPMPSP that maximizes the overall NPV and introduced what is called a geological risk management term in the objective function to minimize the deviations from production target. Later on, Ramazan and Dimitrakopoulos (2012) proposed a new formulation based on a two stage SIP formulation that includes a stochastically managed stockpile. In their paper, the authors consider the scheduling decisions to be the first stage variables and the second stage variables are the penalties for not meeting production targets. The stockpile is considered to be stochastic since the decisions of sending the material to this destination is made at the first stage, regardless of the scenario. The authors apply their approach to a deposit of about 20,000 mining blocks, but need to separate the input into two parts to be able to solve it in a reasonable amount of time. Menabde et al. (2007) proposed the explicit formulation of a cut-off policy into the OPMPSP, where it is modeled by binary variables for each period. They define for each cut-off bin the value of a block for a given scenario and optimized with the constraints that only one cut-off per period must be chosen. They showed that this formulation increases the NPV compared to a conventional schedule, but did not mention the dimension of the input they were working with. Conversely, Boland, Dimitrescu and Froyland (2008) proposed a different approach based on a multistage stochastic programming approach where they use non-anticipativity constraints to model the decision of mining and processing at each period and for each scenario. They use the α-differentiator to reflect the difference between scenarios and allow different decisions for scenarios that have been different enough during the previous year. This formulation allows for flexibility at the operations level; however, no final solution can be provided, only possible outcomes, rendering the approach impractical. Unfortunately no quick methods are known for solving the exact formulation of real-life instance size deposits, which may not find a solution in a practical amount of time. To overcome the computation issues, several authors have proposed metaheuristics to solve large scale problems since these methods have proven to generally lead to a near optimal solution in a more practical amount of time.

Leite and Dimitrakopoulos (2007) and Albor and Dimitrakopoulos (2009) test the simulated annealing method of Godoy and Dimitrakopoulos (2004) that uses quantified geological uncertainty of an orebody to generate a long-term mining schedule. Recently, Lamghari and Dimitrakopoulos (2013) addressed the

OPMPSP with geological uncertainty by using a Variable Neighborhood Descent approach. These authors also present a Tabu Search method for the OPMPSP with geological uncertainty (Lamghari and Dimitrakopoulos, 2012), using as the neighborhood of a solution any feasible solution that differs from one block scheduling at another period. In their implementation, the authors proposed a diversification strategy based on a long term memory given the number of times blocks are schedule to a particular period. When applied to multiple instances, the results provide a good quality solution obtained in a practical amount of time. Because of the easy access to multiple cores for computation, parallel algorithms have received substantial interest in obtaining efficiency. Metaheuristics are usually easy to parallelize because of their structure, which makes this approach a promising way to obtain a near optimal solution in a practical amount of time. Randall and Abransom (1999) review parallelisation strategies for Tabu Search (TS) for combinatorial problems: more generally Cung et al. (2002) present a review of parallel strategies for multiple metaheuristics. Alba (2005) gives a whole survey and description of metaheuristic and parallel metaheuristic development until 2005.

In this paper, a parallel Tabu Search procedure is presented based on the algorithm of Lamghari and Dimitrakopoulos (2012) to solve a modified version of the two stage stochastic approach of Ramazan and Dimitrakopoulos (2012) where the stockpile is formulated as a second stage decision. The processing and stockpiling decisions are scenario dependant since they reflect the fact that at the time of mining, the material type and processing destination is known. Three destinations are considered for a block: the mill, where high grade ore is processed; the leach pad, where low grade material is processed and; a waste dump, where the blocks which contain no valuable material are sent. A mathematical formulation of the problem is first stated, and then a review of the sequential Tabu Search of Lamghari and Dimitrakopoulos (2012) is given. Next, three implementations are given based on different approaches from Randall and Abransom (1999). The first one is Parrallel Independant TS (PITS), the second Parallel Interacting TS (PInTS) and the third is Parallel Neighbor TS (PNTS). A case study showing numerical results for a medium-large size deposit is given after and finally analysis, conclusions and further research are discussed.

2 SIP formulation for stochastic mine scheduling

In this section, the mathematical notation and formulation of the OPMPSP are stated and assumptions stated.

2.1 Model

First, the grade of a block determines where it should be processed under a specific scenario. Fixed marginal cut-offs are used in this approach to classify blocks into a category under a specific scenario, but in future research this assumption will be removed. The process where the block is sent corresponds to the bin where that block's grade falls into. In this paper, the different cut-offs are as follows.

G_i: is the fixed cut-off grade that a block must have to be processed by the leach pad;

$$G_l = \frac{LeachPadProcessingCost}{LeachPadRecovery\,(MetalPrice - SellingCost)}$$

G_m: is the fixed cut-off that a block must have to be processed by the mill.

$$G_m = \frac{MillProcessingCost}{MillRecovery\,(MetalPrice - SellingCost)}$$

2.1.1 Notation

The following notation is used:

- N is the number of blocks considered in the mine.
- i represents a block index, $i \in \{1 \dots N\}$.
- P_i is the set of predecessors of block i.
- w_i is the weight of block i.
- T is the life time of the mine.
- t is a period index, $t \in \{1 \dots T\}$.
- W^t is the maximum weight that can be extracted at period t.
- S is the number of scenarios used for grade uncertainty.
- s is a scenario index, $s \in \{1 \dots S\}$.
- g_{is} is the grade of block i under scenario s.
- d is the financial discount rate.
- r is the geological discount rate.
- C_{msu} is the undiscounted cost per unit of surplus material for the mill.
- $C^t_{msu} = \frac{C_{msu}}{(1+r)^t}$ is the discounted cost per unit of surplus material for the mill.
- C_{msh} is the undiscounted cost per unit of shortage material for the mill.
- $C^t_{msh} = \frac{C_{msh}}{(1+r)^t}$ is the discounted cost per unit of shortage material for the mill.
- C_{lsu} is the undiscounted cost per unit of surplus material for the leach pad.
- $C^t_{lsu} = \frac{C_{lsu}}{(1+r)^t}$ is the discounted cost per unit of surplus material for the leach pad.
- $\delta^t = \frac{\delta}{(1+r)^t}$ is the cost per tonne for sending material to the mill stockpile at period t.
- $\eta^t = \frac{\eta}{(1+r)^t}$ is the unit cost per tonne for taking material to the mill stockpile at period t.
- l_{is} is a binary variable indicating if block i can be send to the leach pad under scenario s:

$$l_{is} = \begin{cases} 1 & \text{if } g_{is} \in (G_l, G_M] \\ 0 & \text{otherwise.} \end{cases}$$

- m_{is} is a binary variable indicating if block i can be send to the mill under scenario s.

$$m_{is} = \begin{cases} 1 & \text{if } g_{is} > G_m \\ 0 & \text{otherwise.} \end{cases}$$

- α^t_{is} is a discounted profit generated if block i is mined during period t and if scenario s occurs. This economic value is given by:

$$\alpha^t_{is} = \begin{cases} \frac{w_i \cdot [g_{is} \cdot MillRecovery \cdot (MetalPrice - SellingCost) - MillProcessingCost - MiningCost]}{(1+d)^t} & \text{if } m_{is} = 1 \\ \frac{w_i \cdot [g_{is} \cdot LeachRecovery \cdot (MetalPrice - SellingCost) - LeachProcessingCost - MiningCost]}{(1+d)^t} & \text{if } l_{is} = 1 \\ \frac{-MiningCost}{(1+d)^t} & \text{otherwise} \end{cases}$$

- The expected profit made if block i is extracted in period t is given by :

$$E\{NPV\}^t_i = \sum_{s=1}^{S} \frac{\alpha^t_{is}}{S}$$

- b_s is the approximation of the grade of the material in the mill stockpile under scenario$s$. The approximation is done using the formula:

$$b_s = \frac{\sum\limits_{i \in B_s} g_{is} \cdot w_i}{\sum\limits_{i \in B_s} w_i} \qquad \text{where} \qquad B_s = \{\, i : m_{is} = 1\}$$

- μ_s^t is the unit discounted profit for material in the mill stockpile at period t under scenario s.

$$\mu_s^t = \frac{b_s \cdot MillRecovery \cdot (MetalPrice - SellingCost) - MillProcessingCost}{(1+d)^t}$$

- $L^{t\circ}$ is the maximum weight of material that can be processed by the leach pad during period t.
- $M^{t\circ}$ is the maximum weight of material that can be processed by the mill during period t.

The following are used to formulate the problem:

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

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

The following positive linear variables are associated with the amount of material for each process:

- d_{msh}^t represents the shortage of the high grade material for the mill for period t under scenario s.
- d_{msu}^t represents the surplus of high grade material for the mill for period t under scenario s.
- d_{lsu}^t represents the surplus of low grade material mined during period t under scenario s.
- k_s^t represents the amount of material sent to the stockpile at period t under scenario s.
- u_s^t represents the amount of material in the mill stockpile at the end of period t under scenario s.
- v_s^t represents the amount of material taken from the mill stockpile during period t under scenario s.

2.1.2 Objective function

$$\sum_{t=1}^{T} \left\{ \underbrace{\sum_{i=1}^{N} E\left\{NPV\right\}_i^t x_i^t}_{Part\,1} - \underbrace{\sum_{s=1}^{S} \left(\mu_s^t + \delta^t\right) k_{ms}^t / S}_{Part\,2} + \underbrace{\sum_{s=1}^{S} \left(\mu_s^t - \eta^t\right) v_s^t / S}_{Part\,3} \right.$$
$$\left. - \underbrace{\left(C_{msh}^t d_{msh}^t + C_{msu}^t d_{msu}^t + C_{lsu}^t d_{lsu}^t\right)}_{Part\,4} \right\} \quad (1)$$

The objective function is separated in four parts: part 1 represents the profit made by extracting and processing the ore blocks; the second part refers to the cost of stockpiling material; part 3 defines the profit made by processing material from the stockpile and finally; part 4 represents the geological risk management, as presented in Ramazan and Dimitrakopoulos (2005), and is added to minimize the deviations from production targets.

2.1.3 Constraints formulation

The following constraints are considered in the present model:

$$\sum_{t=1}^{T} x_i^t \leq 1 \qquad \forall i \tag{2}$$

$$x_i^t - \sum_{\tau=1}^{t} x_j^\tau \leq 0 \qquad \forall i, j \in P_i, t \tag{3}$$

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

$$\sum_{i=1}^{N} l_{is} w_i x_i^t - d_{lsu}^t \leq L^t \qquad \forall t, s \tag{5}$$

$$\sum_{i=1}^{N} m_{is} w_i x_i^t + v_s^t - k_s^t - d_{msu}^t + d_{msh} = M^t \qquad \forall t, s \tag{6}$$

$$u_s^t = u_s^{t-1} - v_s^t + k_s^t \qquad u_s^t \leq S^t \qquad \forall t, s \tag{7}$$

$$x_i^t \in \{0, 1\} \qquad \forall i, t \tag{8}$$

$$k_s^t, u_s^t, v_s^t \geq 0 \qquad \forall s, t \tag{9}$$

$$d_{msu}^t, d_{msh}^t, d_{lsu}^t \geq 0 \qquad \forall t \tag{10}$$

Constraint 2 is the reserve constraint which ensures a block will only be mined at most once. Constraint 3 is the slope constraint, ensuring that each block will be mined after its predecessors. Constraint 4 represents the mining constraint, preventing the equipment capacity from being exceeded. All of these constraints are scenario independent, and are referred to as non-stochastic constraints.

Constraints 5 and 6 are related to the plant capacity. For each scenario, the total weight of ore extracted must be less or equal to the capacity of the plant; all other ore blocks extracted are sent to the stockpile. Constraint 7 is the stockpiling constraint, indicating that the amount at the beginning of a period is the previous stockpile content added or subtracted to the amount taken or sent to the stockpile. It also ensures that the total amount of material sent to the stockpile is not more than a specified upper bound.

3 Implementation of sequential Tabu Search

The proposed model is solved using the Tabu Search procedure described in Lamghari and Dimitrakopoulos (2011), an overview of the algorithm is given here.

3.1 Modified model

In order to allow more extensive search of the solution space, a modification of the objective function is used, subject to the constraints from Eqs. 2 to 9, and is as follows.

$$\sum_{t=1}^{T} \left\{ \underbrace{\sum_{i=1}^{N} E\left(NPV\right)_i^t x_i^t}_{Part\,1} - \underbrace{\sum_{s=1}^{S} \left(\mu_s^t + \delta^t\right) d_{ms}^t / S}_{Part\,2} + \underbrace{\sum_{s=1}^{S} \left(\mu_s^t - \eta^t\right) v_s^t / S}_{Part\,3} \right.$$

$$\left. - \underbrace{\left(C_{msh}^t d_{msh}^t + C_{msu}^t d_{msu}^t + C_{lsu}^t d_{lsu}^t\right)}_{Part\,4} + \underbrace{P^+ \max\left\{ \left(\sum_{i=1}^{N} w_i x_i^t - W^t \right), 0 \right\}^2}_{Part\,5} \right\} \tag{11}$$

In this objective function, a penalty term has been introduced to create a bridge from a feasible solution to another. This is done by using infeasible solutions that can be promising. The adjustments help to return to the feasible space as the penalty coefficient grows. The penalty coefficient P^+ is adjusted using the following criteria:

Starting with $P^+ = 1$ and given a parameter h, for every h iteration check whether all previous solutions have been feasible (for the mining capacity); if yes, then update $P^+ = P^+/2$, else $P^+ = 2P^+$.

3.2 Tabu Search procedure

Given a specific schedule that satisfies the modified model, the neighborhood of the schedule is defined as all the schedules that satisfy the constraints 2 and 3, but have only one block that belongs to a different period. The block/period combinations that are different are called moves. At each iteration, the move with the best improvement of the objective function and which satisfies one of the two criteria given below is selected.

1. The schedule leads to the global maximum objective function found so far.
2. The schedule that leads to the best improvement compared to the current schedule and the move is not considered Tabu.

When applying a move, the reverse of the selected move is considered Tabu, meaning that the combination for the block and period of the move cannot be applied again for a certain amount of iterations, chosen randomly from a given range. This procedure stops after a certain number of iterations without improvement or when all moves are Tabu. Then, diversification is applied.

3.3 Diversification strategy

The diversification strategy used can be summarized as follows.

To generate a new initial solution for the Tabu Search procedure, a diversification procedure is applied based on its long term history. First, a block is chosen randomly and the period of extraction is chosen randomly based on the inverse of the number of times the block has been scheduled to this period. These shifts can lead to a non-feasible schedule in term of constraint 3, and thus a heuristic is used to repair the schedule. Considering the set of all blocks that do not respect slope constraint 3, choose a block at random from this set. To generate a period of extraction from those that do not violates slope constraint 3. The period is chosen based on the long term frequency in order to extensively search the space. Once the block is scheduled remove it from the set and repeat until the set of blocks to repair is empty. Restart the Tabu Search.

4 Implementation of parallel Tabu Search

With the evolution of computing, multi-core programming has become a predominant way of making algorithms faster. In this section, three ways of parallel implementation are proposed to be implemented, and flowcharts as well as the pseudo code are given. All algorithms are implemented based on the approaches found in Randall and Abramson (1998) where they define a general approach to parallelizing the Tabu Search procedure. A similar approach can be found in Crainic et al. (1997) where they define taxonomy for different types of parallel Tabu Search.

In the present study, the first and second implementation, Parallel Independent Tabu Search (PITS) and Parallel Interacting Tabu Search (PInTS) are used to undertake a more extensive search of the solution space. The third implementation is called Parallel Neighbor Tabu Search and aims to do a more intensive search than the original Tabu Search procedure.

4.1 Parallel independent Tabu Search

In this implementation, the Master-Slave paradigm described in Hansen (1993) is used to design the parallel algorithm. The master thread gives to each of the threads a complete Tabu Search Proceduce iteration to

compute. When the process terminates, the master seeks for the best solution over all the threads and finds the globally optimal solution. PITS is the straightforward way of parallelising the Tabu Search algorithm already explained. Its possible efficiency is based on the fact that when more Tabu Searches are launched from different initial solutions in the feasible space, there is a higher probability of getting a better solution for the same amount of time spent. However, this algorithm does not help the Tabu Search to get out of a local optimal, but only provides a more extensive search. The use of multi-core programming is simple: many different threads are launched where each one owns a distinct starting point. Let TL and F and $\oplus(i,t')$ be the Tabu list structure, frequency structures and move operator as described in Lamghari and Dimitrakopoulos (2012). Let THS be the threads available for the algorithm. The following shows the pseudo code for the implemented algorithm:

PITS_Tabu {

1- For each threads in THS {

   a. Generate an initial schedule x_0 and set $x_{best} = x_0$

   b. Apply Tabu Search procedure on x_0

      i. Choose the best available and feasible schedule $x_0 \oplus (i,t')$

      ii. Apply $x_0 \oplus (i,t')$ and forbid $x \oplus (i,t)$, update F and TL

      iii. If $x_0 \oplus (i,t') > x_{best}$ set $x_0 \oplus (i,t') = x_{best}$

      iv. If *stopping criteria* is met stop and go to c.

   c. Apply diversification on $x_{best}$to generate another schedule x_0, if there's time left go to b, else go to 2.

}

2- Select the best schedule x_{best} over all threads in THS.

}

Each thread evolves independently of the others. When every thread has finished, a reduction function is used to get the global minima. The method is summarized in Figure 1:

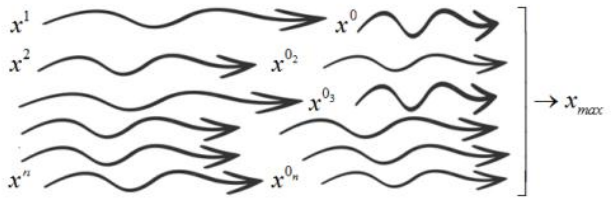


Figure 1: Threading schema of PITS

In the proposed implementation, no thread needs to wait for another and thus there is no race by any thread for any data. Provided there is enough RAM, the expected gain in speed is proportional to the number of threads. Thus with the gain of time through this implementation, one can run more Tabu Searches to cover the search space in a more complete way and expect to have a better final solution. It can also allow more time for a single run in a multi-core approach and get closer to the optimal solution in less time than running the sequential one.

4.2 Parallel interacting Tabu Search

Parallel Interacting Tabu Search (PInTS) is based on the same template design as PITS, but uses the knowledge of more than a single thread to diversify and generate a new solution. Using the same exact notation as in Section 4.1, let x_{best_pool} be the best solution over all threads, F_{best_pool} be the merging

structure associate to the global best solution so far and κ as a weight associated with the frequency of the history. PInTS adds a synchronization step between steps b and c. The following lines represent the pseudo code of the PInTS algorithm used:

PInTS_Tabu {

1- For each threads in THS {

a. Generate an initial schedule x_0 and set $x_{best} = x_0$

b. Apply Tabu Search procedure on x_0

i. Choose the best available and feasible neighbor schedule $x_0 \oplus (i, t')$

ii. Apply $x_0 \oplus (i, t')$ and forbid $x_0 \oplus (i, t)$, update F and TL

iii. If $x_0 \oplus (i, t') > x_{best}$ set $x_0 \oplus (i, t') = x_{best}$

iv. If $x_{best} > x_{best\_pool}, x_{best\_pool} = x_{best}$, $F_{best\_pool} = F$

v. If *stopping criteria* is met stop and go to c.

c. Apply diversification on x_{best_pool} using the following steps

vi. $F = F + \kappa F_{best\_pool}$

vii. Apply diversification on x_{best_pool} using F to generate another x_0

If there is time left, go to step b, else go to 2.

}

2- Select the best schedule x_{best} over all threads in THS.

}

By adding these communication steps, PInTS diversifies the search more effectively since it uses information about the history of all the threads compared to the PITS where only the information about the current thread is available. The vertical lines in Figure 2 represent a barrier of synchronisation; each thread must synchronize its value in order to have the most recent one.

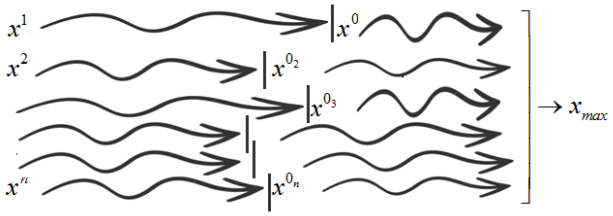


Figure 2: Threading schema of PInTS

4.3 Parallel neighborhood sampling Tabu Search

The Parallel Neighborhood Sampling Tabu Search (PNSTS) is another enhancement that has been made to make Tabu Search more robust for the problem addressed herein. The idea cones from the fact that the presented approach is "deterministic" in the sense that it only takes the move that increases the local optimum the most or that decreases it the least. Let THS be all the threads available, assuming that each thread has its own local structure initialized with the same values, $|THS|$ the number of threads, θ_k a random number generated between $\left[\frac{1}{2|THS|}, \frac{2}{|THS|}\right]$, ν the number of iterations for the parallelized step, x_{best_pool} the best solution over all threads, x_{i_end} the current solution at the end of the parallel step of thread i, the proposed algorithm adds a step executed in parallel to step 2 of the algorithm, as follows:

PNSTS_Tabu {

1- Generate an initial schedule x_0 and set $x_{best} = x_0$

2- Apply Tabu Search procedure on x_0

a. For each thread in THS and during ν iterations:

i. If this is the first thread keep the entire neighbor, else forbid the best neighbor and sample θ_k time the neighborhood of x_0

ii. Choose the best available and feasible neighbor $x_0 \oplus (i, t')$

iii. Apply $x_0 \oplus (i, t')$ and forbid $x \oplus (i, t)$, update F and TL

iv. If $x_0 \oplus (i, t') > x_{best}$ set $x_0 \oplus (i, t') = x_{best}$

v. If $x_{best} > x_{best\_pool}$, $x_{best\_pool} = x_{best}$

b. $x_0 = \max_{i \in THS} \{x_{i\_end}\}$

c. If *stopping criteria* is met stop and go to 3, else go to a.

3- Apply diversification on x_{best} to generate another schedule x_0, if there's time left go to b, else go to 2.

}

In order to search the local space more intensively during the Tabu Search when compared to the other implementations, k threads are assigned a restricted part of the neighborhood where they will focus their search starting from the same solution. After k number of iterations, the threads that have the best current solution broadcasts the necessary information about the search structures to the others so that all threads restart their search from the same place and with the same exact state. Then, a re-sampling of the new neighborhood is done. Figure 3 represents the interaction between the threads evolution during the Tabu Search.

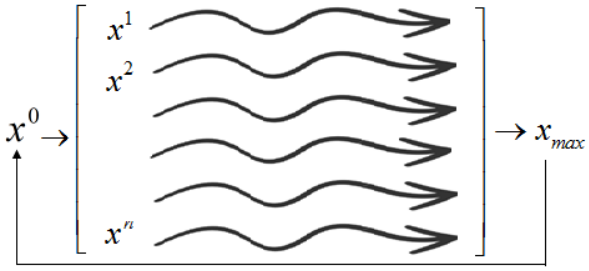


Figure 3: Threads diagram for PNSTS

At the computational level, the master threads split into k children that run in parallel, each one searching for its own local space with its own local structure. When the specified number of iterations is reached, the master threads look over all the children threads to identify the best results and reinitialize each child using the most promising point. The diversification is then done based on the best solution found so far and the long term structure of the threads that found it.

5 Case study

First, an analysis of the parallel efficiency of the proposed algorithms is given by comparing the quality of the solution in terms of the number of threads and the number of runs versus the number of threads for a fixed computational effort(time). Next, a benchmark of the quality of the solution is obtained based on the linear relaxation value (LR). Finally an analysis of the quality of the solution with regards to mine planning is given to assess the model performance. To evaluate the efficiency of the parallelization and provide a benchmark in term of optimality, two different instances are used. The parameters are given in Table 1.

Table 1: Parameters used in this case study

Instance	Initial Sol.	Blocks	Periods	Scenarios	Time (s) (1run)	LR value
P1-H	Heuristic	4073	3	20	60	100.92
P1-C	Exact					
P2-H	Heuristic	20626	5	20	1034	170.18
P2-C	Exact					

In order to test the robustness of the method, two different starting points are generated as in Lamghari and Dimitrakopoulos (2012): one that uses an exact method to get a solution close to what is optimal, and another random heuristic that generates only a feasible solution regardless of the value. The problem that is solved by using the first approach as a starting point are followed by a –C and the other by a –H in the following sections. The reader is referred to the above paper to know more about the initial solution methods.

5.1 Parallel efficiency of the different algorithms

To test the parallel efficiency of the approach presented in Section 4, an analysis of Tabu Searches is made by comparing the number of instances of Tabu Search being launched given a specific time in the function of the number of threads used for the three methods. Results can be seen in Figure 4. As seen, the curve of the number of threads versus the number of runs for each of the algorithms is as good as the theoretical bound, meaning that the implemented methods take full advantage of the multiple cores available. The results also demonstrate that in the present case, covering more space usually gives a better final solution for the PITS and the PInTS and that in this case, PInTS leads to a better solution than the PITS algorithm. It also shows that as the number of threads increase, the three algorithms give almost the same results in terms of objective function but PInTS still leads to a better solution. For PNSTS, as the number of threads increase it appears to perform slightly better but appears to have a critical point where increasing the number of threads does not make a difference, in this example after three threads. Thus, for the small instance, one can see that the multi-threading seems to help in finding a better solution in the same amount of time and thus provides an improvement over the single threading version of the algorithms. It is worth noticing that even though all three methods are theoretically the same when using a single thread, the way they are implemented affects the final solution and so they are different in practice.

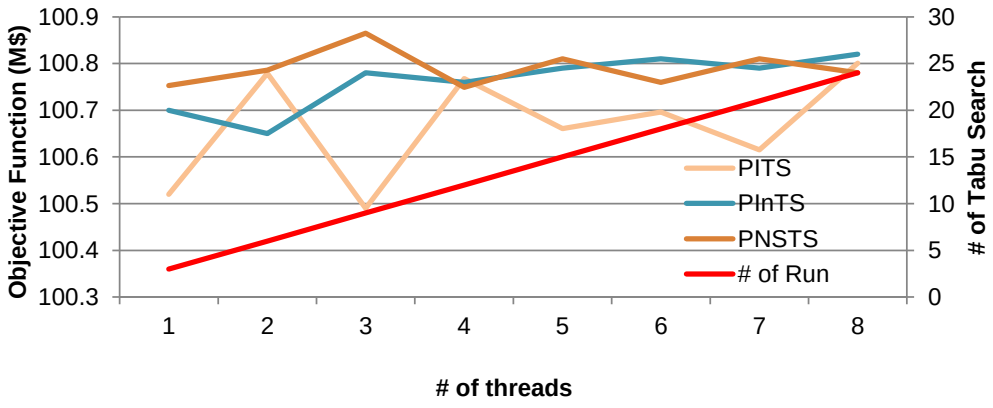
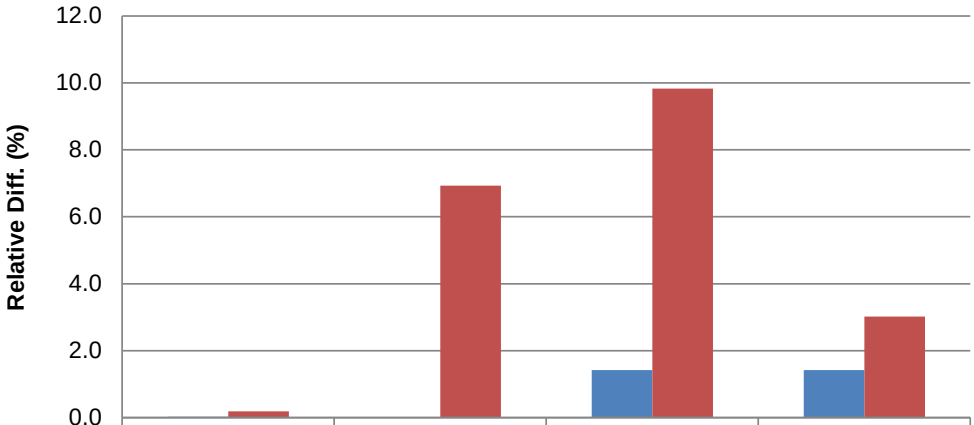


Figure 4: Parallel efficiency for deposit instance P1

To test the effects of multithreading of PITS and PNSTS against their respective sequential version, an experiment is performed by fixing the number of threads and the total running time to 60 s. Figure 5 gives the relative improvement made on the final solution by using the parallel version instead of the sequential one. Note that in the case of PNSTS, the number of samples is fixed and therefore it has to follow the same number of paths as the parallel one. In this benchmark, we used the small instance since the results for the bigger one are not available at this time.



	P1 - H	P1 - C	P2 - H	P2 - C
PITS	0.024964	0.004030	1.420569	1.416152
PNSTS	0.182937	6.933721	9.830910	3.017200

Figure 5: Parallel versus sequential approach

Figure 5 shows the relative difference between using up to 8 threads for PITS and PNSTS algorithms, as compared to using only a single one. The relative difference is calculated by:

$$\text{Relative Diff.} = \frac{(\text{Obj. Func 8 threads} - \text{Obj. Func. single thread})}{\text{Obj. Func single thread}}$$

This measure reflects the improvement made by using multithreading in terms of the quality of the solution. As seen, the difference in terms of quality of the final solution is larger in the case of PNSTS than in the case of PITS. For small instances, using the parallel version of PITS makes almost no difference, less than 1%. Although as the size increases, the parallel PITS seems to be able to provide a better solution in almost every case compared to the sequential one, here more than 1%. However, for PNSTS, the relative difference made by using multithreading seems to be more predominant but does not seem to rely on the initial solution quality. Thus, the parallelization provides improvement for the two algorithms, but seems to lead to higher improvement when using PNSTS. Note that the PInTS algorithm is not shown in this graph, since the single version corresponds to the PITS algorithm and thus it is not comparable.

5.2 Benchmarking

In order to benchmark the quality of the solutions generated by the proposed algorithms, the two instances are tested using a comparable amount of time that is proportional to the number of blocks multiplied by the number of periods. The time used was computed as:

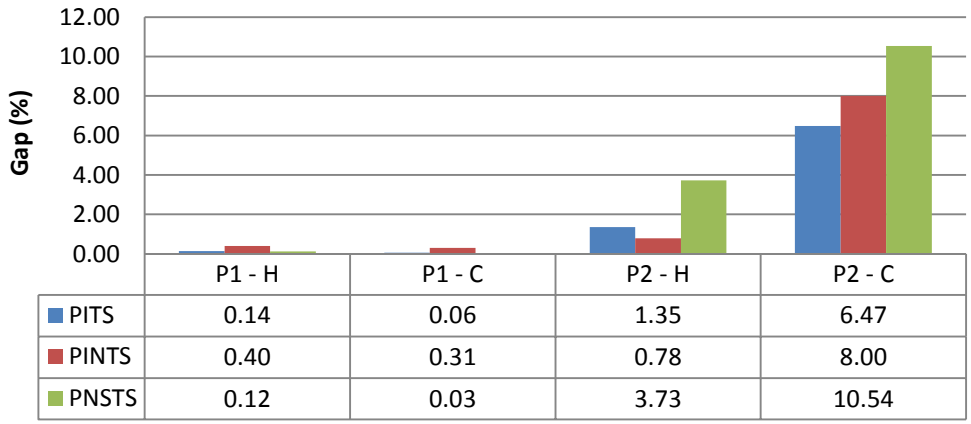
$$Time = 0.01NT$$

where N is the number of blocks and T is the number of periods. To compare the quality of the solution, the gap with respect to the linear relaxation (gap) is used. A table showing the gap is given for all the cases and to test the robustness in terms of the starting solution of each algorithm. The relative difference from the starting point is given to allow the evaluation of the improvement over different types of solution. Figure 6 shows the gap for each method and for each of the two different starting points. As seen in the following table, PNSTS usually gives the worst solutions for the proposed problem since it always result in a higher gap than the two other methods. Overall, looking more intensively around the current solution for other local optima doesn't appear to result in a good parallel Tabu Search approach for this problem. The PITS method seems to be efficient in almost all cases since it leads to a better solution for small and small-medium instances. Covering more space can therefore be a good idea to improve the overall solution quality. However, the table also shows that the PInTS method performs well on all the instances and as the

size of the instance grows, it can even perform better than the simple PITS when starting with a random feasible solution. Another observation is that for the small instance starting with any of the two types of solutions it results in almost the same quality, but for the small-medium instance when starting from a far from optimal solution it seems to provide better final solution. This may mean that the proposed algorithms for the medium-small instance fail to get out of a nearly optimal solution when starting with a better sub optimal solution than one generated randomly. This might be a result of the different history learned during the search. However, this analysis is made only for one specific instance and such conclusions might not hold for further work.

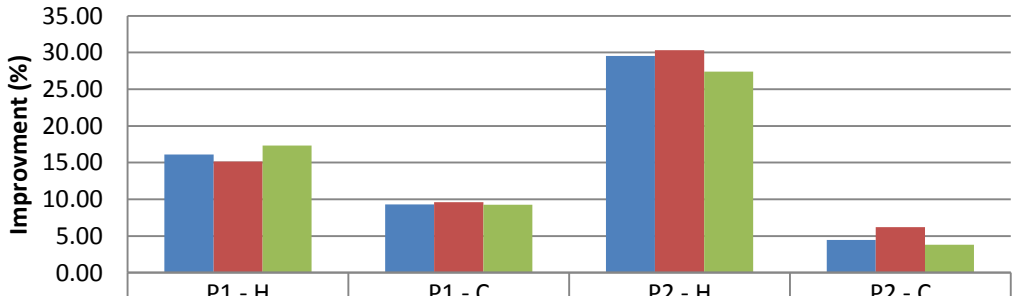
Figure 6 also shows that the proposed methods are suitable for the OPMPSP that contains a stockpile and leach pad, as they provide good results in a practical amount of time (usually less than 2% closer to the optimal solution). The methods are also robust for small instances since starting from either a random solution or from a nearly optimal solution leads to a small gap. As the instance grows in size, each method seems to need more time to be able to compute a solution closer to the optimal one. Figure 7 shows the improvement from the initial solution to the solution found. As seen, in all cases they can improve the initial solution, even when it was already close to optimality.

Figures 6 and 7 combined tell us that the exact method solution is further from optimal as the instance size grows, and the methods are not able to improve by using a random solution. It is expected that the improvement will be less when starting from a better solution since the total improvement depends on the



	P1 - H	P1 - C	P2 - H	P2 - C
PITS	0.14	0.06	1.35	6.47
PINTS	0.40	0.31	0.78	8.00
PNSTS	0.12	0.03	3.73	10.54

Figure 6: Final gap for each method and initial solution



	P1 - H	P1 - C	P2 - H	P2 - C
PITS	16.09	9.28	29.52	4.48
PINTS	15.13	9.59	30.29	6.20
PNSTS	17.30	9.25	27.38	3.81

Figure 7: Improvement of the initial solution

starting point, but we still show that by starting from either solution we can improve the solution by more than 3.5% in all cases.

5.3 Application at a gold deposit

In this section, an application at a gold deposit with a medium-large scale dataset is presented. First, an overview of the parameters is given, and then a risk profile of the solution generated by the algorithm is made.

The previous benchmark was made on a small and medium-small instance of the deposit since the linear relaxation value was obtained within a reasonable amount of time. However for the medium-large scale deposit presented next, the linear relaxation solution was not able to be obtained within a reasonable amount of time and thus only a risk profile of the solution generated is shown. This case study demonstrates that the current algorithm can be applied on a medium-large scale deposit and that a solution with a good risk profile can be generated within a reasonable amount of time. To generate the solution, the second approach, (PInTS) was used with a randomly generated initial solution since it seemed to provide better performance for this bigger dataset.

Table 2 shows the parameters used to schedule the medium-large scale deposit. It shows that to generate the schedule, 14 orebody simulations were used with about 73,000 blocks and a life of mine of maximum 16 years, meaning that a solution with about 16,000,000 variables was generated within 9 hours 45 minutes. The risk profile used consists of the metal content and the ore tonnage processed by the mill, the total rock tonnage extracted from the tonnage of waste, the cumulative NPV and also the cost incurred by deviating from targets. The latter is added in order to evaluate the impact of adding the geological risk management term on the cumulative NPV.

Table 2: Parameters for large scale dataset

Parameters:		
Number of blocks		73130
Horizon	Year	16
Number of scenarios		14
Metal Price	$/g	17.23
Mining Capacity	Mtpa	85
Mill Target Capacity	Mtpa	15
Leaching Capacity	Mtpa	5
Mill Surplus Penalty	$/t	8.00
Mill Shortage Penalty	$/t	10.00
Leachpad Surplus Penalty	$/t	7.00
Financial Discount Factor	%	8
Geological Discount Factor	%	20
Number of Tabu Searches		7
Number of Threads		7
Total time		9 h 45 min

As seen in Figure 8, the ore tonnage input to the mill is very close to the target for the earlier years, but as the time passes deviations begin to increase. This happens because the geological risk discount factor allows for penalizing earlier periods more than later ones. The risk of not meeting production targets is then very low at the beginning of the project and tends to increase at the end. This is normal because more information will be available later and the risk will be reduced at that time. As seen in Figure 9, the metal content sent to the mill process is higher in earlier periods than in later periods, meaning that the profit made by processing the ore will be higher. This is how the model reacts to the application of a discount factor being applied. Additionally, it shows that the risk on the metal content sent to the mill is smaller at the beginning than later on as it begins to be higher in year 12. Combined with the previous graph, this means that the risk is pushed to the later periods for this process. Figure 10 shows the content in the stockpile at each year. As seen, the risk according to the content of the stockpile is low at the beginning and increases with time. The amount of material sent to the stockpile is very volatile at the beginning of the mine life and increases to become more and more volatile as the time passes.

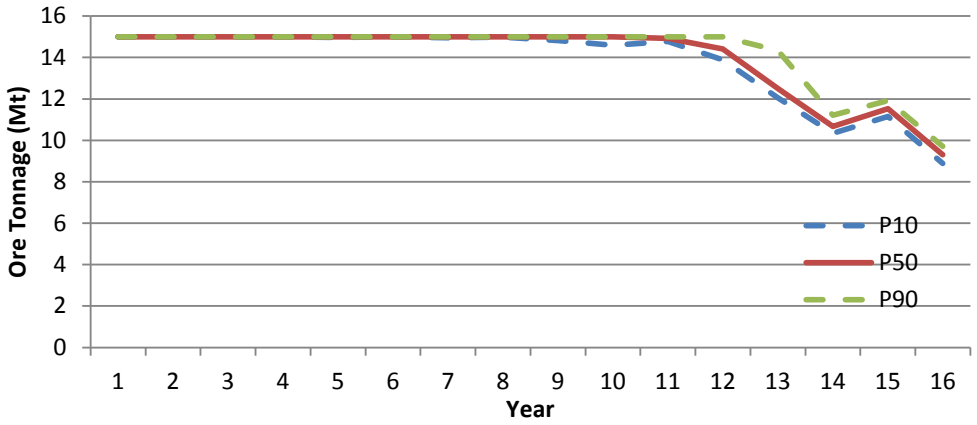


Figure 8: Mill tonnage input

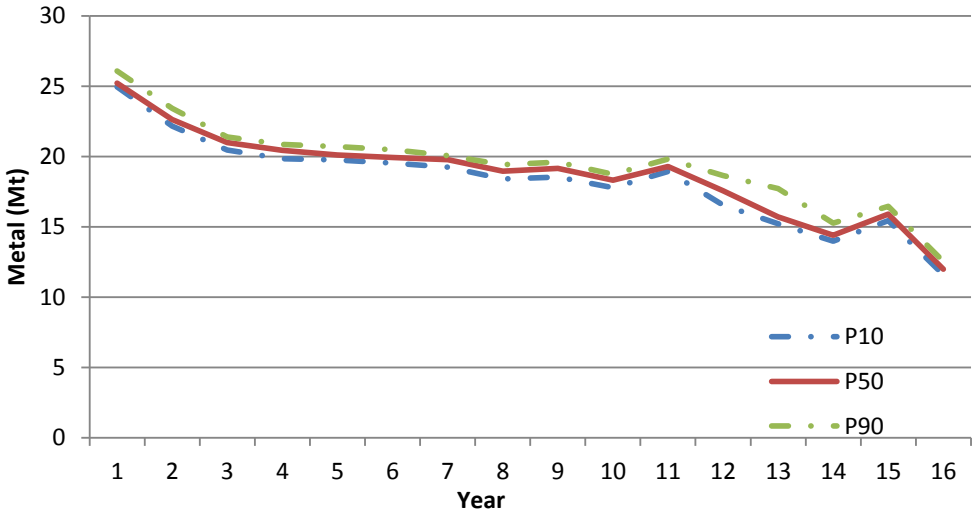


Figure 9: Metal content for the mill

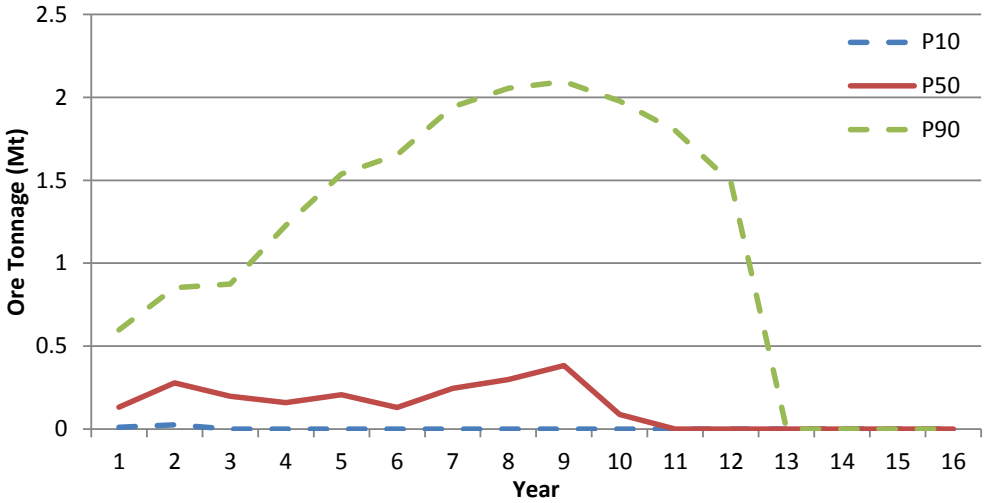


Figure 10: Stockpile content

The SIP formulation incorporates the stockpile only as a buffer for the mill and then the optimizer tries to fill the mill so that every scenario meets the target by scheduling blocks with high variability in the grade early on. It leads to the fact that in a certain scenario blocks are sent to the leaching pad and in others they are sent to the mill. If in one scenario, a certain local area is composed mostly of low grade material and the optimizer does not have any better choice, then, it will extract this part since it helps a couple scenarios to meet the target but more blocks have to be mined in order to meet the target in the other scenarios. This can happen when the bottleneck is the capacity of the mill. Figure 11 shows the rock tonnage extracted and it can be observed that the mining capacity in this case is not the bottleneck for the first periods. As seen in Figure 12, the tonnage of the leach pad does not exceed the fixed target and is reacting as one would expect. However, one may note that even if it is profitable to extract low grade material, the optimizer does not use the full capacity of the process but only the marginal grade that can be extracted.

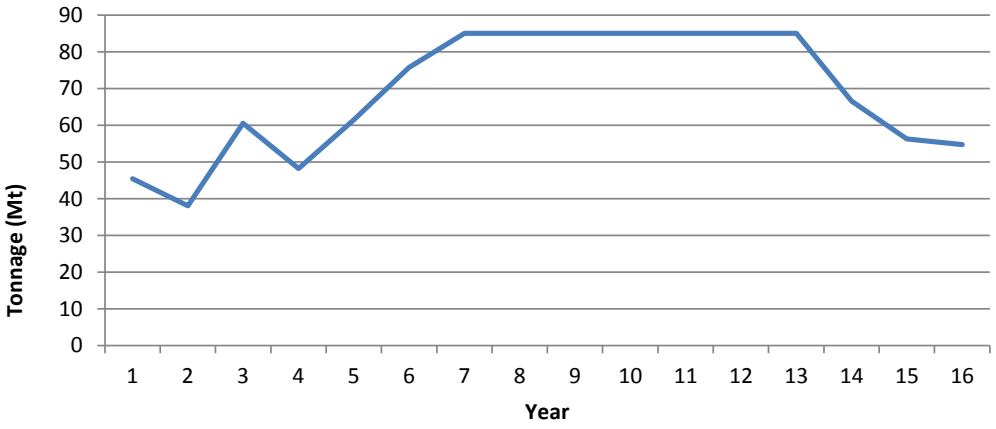


Figure 11: Rock tonnage

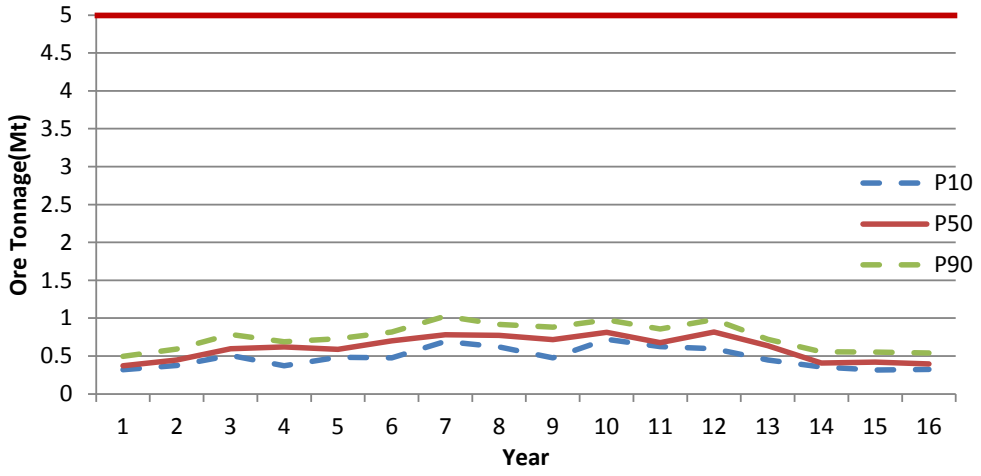


Figure 12: Leach pad tonnage

As discussed, the amount of rock extracted at the beginning is less than in the later periods, meaning that the target is met while extracting fewer blocks. This helps increase the NPV and all the targets are met by extracting high grade blocks. Also from a practical point of view, the tonnage increases as the year increases, but has no decrease or increase leading to a feasible schedule. Figure 13 shows the waste tonnage. There is almost no risk in the waste tonnage, and it increases as the years pass since the discounted factor costs are less in the later periods. The same remark can be made with respect to the rock tonnages, even though it is increasing in the amount of waste tonnage, it is still physically feasible.

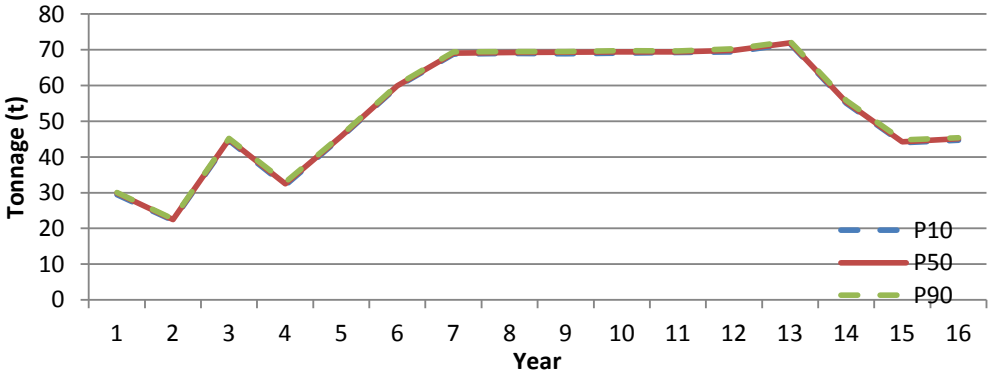


Figure 13: Waste tonnage

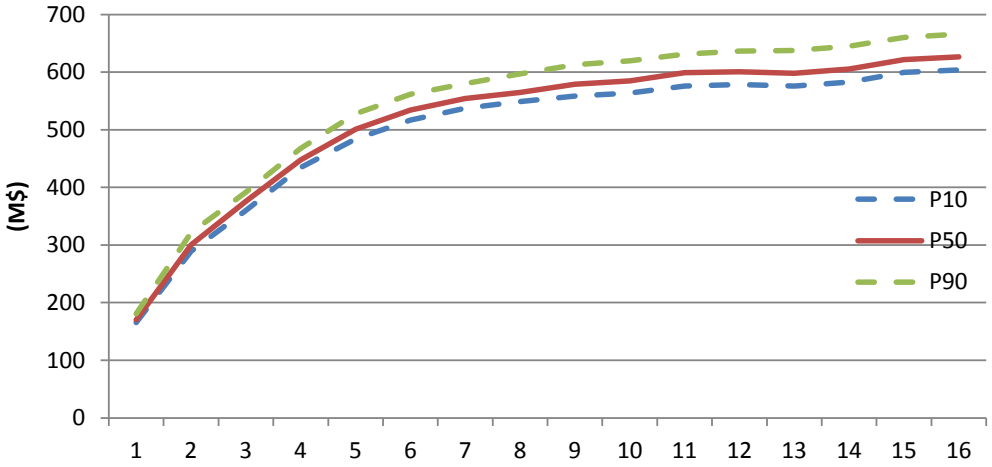


Figure 14: Cumulative NPV

Finally, the total NPV for the project is plotted in Figure 14. It shows that the risk at the beginning of the project is lower than at the end, as it should be. To display the effect of geological risk management, a plot of the value of the deviation by period is given, allowing the magnitude of the penalty applied compared to the NPV to be seen.

Figure 15 shows the effect of the geological risk management on the objective function over the life of mine. It shows that the first periods are severely penalized but as the time passes, the geological discount factor brings the penalty for one ton of material to almost nothing, allowing the model to deviate without incurring a drastic penalty.

6 Conclusions

In this paper, a set of parallel metaheuristics have been developed based on the previous work of Lamghari and Dimitrakopoulos(2012). The proposed metaheuristic incorporates the SIP formulation of Ramazan and Dimitrakopoulos (2013) with the addition of a leach pad. Three algorithms were implemented. The PITS algorithm is a very easy and straightforward way to assess the OPMPSP with multi-destination and a stockpile. As the size instance grows the method is still robust, starting with a random feasible schedule. The second algorithm, PInTS, uses more information than only the information of the current threads, allowing a better coverage of the solution space and better results than the other implemented methods. It solves the OPMPSP problem and can provide a feasible schedule in the terms of optimality and mine planning. The third method, PNSTS, appears to work on small instances but unfortunately provide worst

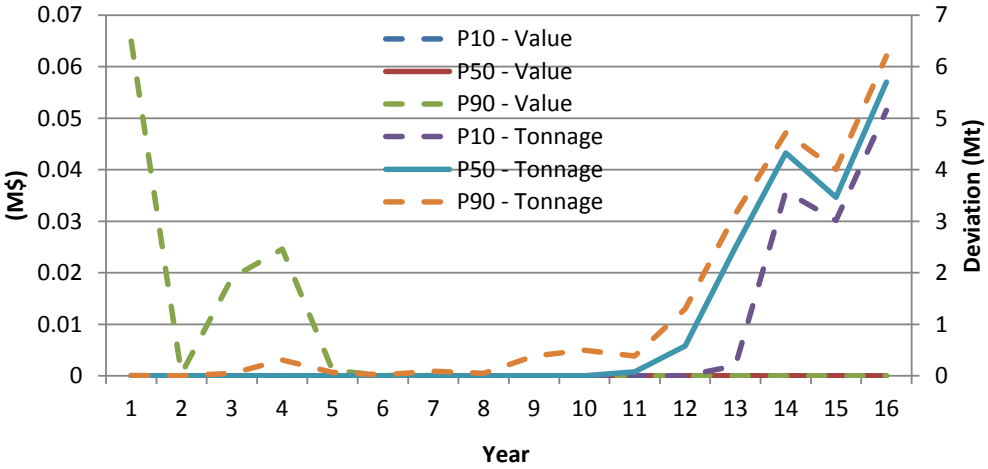


Figure 15: Effect of geological risk management

results than the others in all cases. All of these methods are able to improve the initial solution and as the instance size grows, all methods starting with a random schedule perform better than starting with a close one. The performance gained in time by using parallelism is mostly proportional to the number of threads used for the two instances.

The case study shows that those methods are able to handle a medium-large scale deposit for the SIP proposed. This case study shows that the formulation is able to account for the minimization of the deviations from production targets and provides a schedule where risks are pushed in later periods. It also points out that the stockpile considered now is only there to act as a buffer for the uncertainty at the early period where deviations are important. The analysis in the presented papers has been developed based on a small and medium instance for both parallel efficiency and optimality.

Further improvement of the proposed method can include multi-element deposit, stochastically managed stockpiles or characterisation of the time and space complexity of the algorithms.

7 References

Alba, E. (2005). Parallel Metaheuristics: A New Class of Algorithms. John Wiley & Sons, Inc., vol. 47.

Albor, F., and Dimitrakopoulos, R. (2009). Stochastic mine design optimization based on simulated annealing: Pit limits, production schedules, multiple orebody scenarios and sensitivity analysis. IMM Trans, Mining Technology, 118(2), 80–91.

Baker, C., and Giacomo, S. (1998). Resource and reserves: Their uses and abuses by the equity markets. In: Ore reserves and finance: A joint seminar between Australasian Institute of Mining and Metallurgy and ASX. Sydney, Australia.

Bley, A., Boland, N., Fricke, C., and Froyland, G. (2009). Clique-based facets for the precedence constrained knapsack problem. In: 9th Workshop on Models and Algorithms for Planning and Scheduling Problems, p. 259.

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

Cacetta, L., and Hill, S.P. (2003). An application of branch and cut to open pit mine scheduling. Journal of Global Optimization, 27, 349–365.

Crainic, T., Toulouse, M., and Gendreau, M. (1997). Towards a taxonomy of parallel tabu search heuristic. INFORMS Journal of Computing, 9(1), 61–71.

Cung, V.-D., Martins, S.L., Ribeiro, C.C., and Roucairol, C. (2002). Strategies for the parallel implementation of metaheuristics. Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, 263–308.

Dagdelen, K., and Johnson, T.B. (1986). Optimum Open Pit Mine Production Scheduling by Lagrangian Parameterization. In: 19th International Symposium 1986 – Application of Computers and Operations Research, pp. 127–142.

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

Dowd, P. (1997). Risk in minerals projects: perception, analysis and management. Transactions of the Institution of Mining and Metallurgy, 106, 9–18.

Dowd, P.A., and Kumral, M. (2005). A simulated annealing approach to mine production planning. Journal of the Operational Research Society, 56(80), 922–930.

Gershon, M.E. (1983). Mine Scheduling Optimization With Mixed Integer Programming. AIME Transactions, Preprint No 82-324.

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

Hansen, P. (1993). Model programs for computational science: A programming methodology for multicomputers. Concurrency: Practice and Experience, 5(5), 407–423.

Johnson, T.B. (1969). Optimum open-pit mine production scheduling. A Decade of Digital Computing in the Mineral Industry: A review of the state-of-the-art, AIME, A. Weiss(ed.) New-York, 539–562.

Lamghari, A., and Dimitrakopoulos, R. (2012). A diversified Tabu search for the open-pit mine production scheduling problem with metal uncertainty. European Journal of Operational Research, 222(3), 642–652.

Lamghari, A., Dimitrakopoulos, R., and Ferland, J.A. (2013). A variable neighborhood descent algorithm for the open-pit mine production scheduling problem with metal uncertainty. Journal of the Operational Research Society, Online, http://www.palgrave-journals.com/jors/journal/v65/n9/full/jors201381a.html.

Leite, A., and Dimitrakopoulos, R. (2007). Stochastic optimisation model for open pit mine planning: Application and risk analysis at copper deposit. Mining Technology, 116(3), 109–118.

Lerchs, H., and Grossman, L. (1965). Optimum design of open pit mines. CIM Transactions, vol. LXVIII, CIM, Montreal, 17–24.

Meagher, C. (2011). On the directed cut polyhedra and open pit mining. (Thesis) McGill University.

Menabde, M., Froyland, G., Stone, P. and Yeates, G. (2007). Mining schedule optimisation for conditionally simulated orebodies. In: Orebody Modelling and Strategic Mine Planning, second edition, 353–357.

Ramazan, S., and Dimitrakopoulos, R. (2005). Stochastic optimisation of long-term production scheduling for open pit mines with a new integer programming formulation. AuSIMM, Spectrum Series, 14, 353–360.

Ramazan, S. (2007). The new Fundamental Tree Algorithm for production scheduling of open pit mines, European Journal of Operational Research, 177(2), 1153–1166.

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

Randall, M., and Abramson, D. (1999). A general parallel Tabu Search algorithm for combinatorial optimisation problems. In: Proceedings of 6th Australasian Conference on Parallel and Real Time Systems. Springer-Verlag, Singapore, Cheng, W. and Sajeev, A. (eds), 68–79.

Ravenscroft, P. (1992). Risk analysis for mine scheduling by conditional simulation. Trans Institute of Mining and Metallurgy, 101, 104–108.

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

Vallee, M. (2000). Mineral resource + engineering, economic and legal feasibility = ore reserve. CIM Bulletin, 93, 53–61.

Whittle, J., and Rozman, L.I. (1991). Open pit design in the 90s. Mining Industry Optimisation Conference, Sydney, 13–20.