An Alternative Approach to Push-Back Design

Clemens Mieth and Hooman Askari-Nasab
Mining Optimization Laboratory (MOL)
University of Alberta, Edmonton, Canada

Abstract

Today the standard procedure for developing long term production schedules is based on push-backs derived from parameterization of the ultimate pit. However, the drawback of ultimate pit parameterization is that the push-back size may vary greatly. It is often the case that a small push-back is followed by a very large push-back. This occurrence is generally referred to as the gap problem. In this paper we present an alternative approach for developing push-backs. To create push-backs not exceeding a fixed overall tonnage and ore tonnage we developed an integer optimization model. Iteratively solving the model yields a series of push-backs. Due to the large data volume encountered in mining the model cannot be solved exactly in reasonable time. Instead of employing exact solution methods two heuristics combined with a linear relaxation are used to find an acceptable solution to the optimization model. In each iteration, the linear relaxation of the optimization model is found by using a Lagrangian multiplier based algorithm. The linear relaxation provides an upper bound for the push-back value and also permits a reduction of the problem size. Based on the results of the linear relaxation a greedy heuristic is used to create an initial solution. Subsequently, a local search heuristic is employed to improve the initial solution received from the greedy heuristic. The presented approach is tested on three different deposits, ranging in size and structure. Comparing the results to push-backs received from parameterization it shows that the alternative approach presented is able to overcome the gap problem by creating more homogenous push-backs.

1. Introduction

The development of a mine is an increasingly complex task, requiring considerable upfront investments before any return can be expected. Not only depends the economic success of a mining project on the efficient organization of the mining operations, but also on the quality of the mine planning [1].

Within the planning of a mine the production schedule plays an important part. The schedule is the center piece to which the whole mine will be adjusted including equipment, infrastructure and workforce. It is evident that deficiencies in the long-term production plan will also affect medium-term and short-term plans and can lead to cash shortfalls, equipment underutilization, etc. It may also be the case that mistakes committed in the early stages of the mine become costly to correct in the later stages. On the other hand improvements of the long-term production plan translate into improvements of the subsequent plans as well.

The standard procedure to obtain a production schedule is described by Chicoisne et al. [2] as follows:

1. Geological ore body discretization

Askari-Nasab, Hooman (2012), Mining Optimization Laboratory (MOL) – Report Four, © MOL, University of Alberta, Edmonton, Canada, Pages 340, ISBN: 978-1-55195-301-4, pp. 16-36.

2. Slope angle computation
3. Economic block model development
4. Obtaining final pit limits
5. Ultimate pit parameterization
6. Computing a production schedule.

Today the first four stages are well mastered and they can be accomplished in short time. But due to the large size of many block models it is impossible to optimally solve the production schedule for each individual block [3]. To bypass this problem the ultimate pit is divided into push-backs using ultimate pit parameterization. The push-backs are then used as input for a scheduling algorithm. Therefore the push-backs can be viewed as a preprocessing step for the development of production schedules.

The application of parameterization yields a series of nested ultimate pits. Out of these pits a number of pits are selected and their incremental size defines the push-backs. The size of the nested pits, and subsequently of the push-backs created using parameterization, does not always grow proportional to the multiplier. Therefore, the push-backs may not be uniform in size, but rather can contain large jumps in size between two consecutive push-backs. This occurrence is referred to as the *gap problem*.

In the following an alternative way to create a series of push-backs is presented. By applying operations research techniques push-backs are created, not exceeding a specific overall tonnage and ore tonnage. The technique was implemented in a computer program and tested on various deposit data.

2. Model formulation

2.1. Methodology

A series of nested pits is sought after, which defines the push-backs. Let C^n be the set of all blocks included in nested pit n. Furthermore let C^u be the set of all blocks included in the ultimate pit, then $C^u \supseteq \ldots C^n \supseteq C^{n-1} \ldots C^2 \supseteq C^1$.

Let PB^n be the set of blocks that are included in push-back n then PB^n would be defined as:

$$PB^n = C^n \setminus C^{n-1}. \tag{1}$$

In addition, the push-back defined in Eq. (1) should satisfy several conditions (constraints). It is obvious that the blocks in the push-back will have to satisfy a precedence relationship and slope angles, but to create homogenous push-backs, more constraints have to be introduced. From a production point of view there are two scarce resources in a mine which should be used as efficiently as possible. The mining capacity limits the amount of material that can be extracted and transported to its destination. The processing capacity limits the amount of ore that can be handled at the processing plant.

Since the overall tonnage and ore tonnage in the nested pits derived from parameterization cannot be controlled an alternative approach, where it is possible to introduce tonnage constraints, has to be defined. Instead of creating nested pits, a simple binary integer program (BIP) can be created to find a single push-back. In order to develop a series of push-backs, the simple BIP can be iteratively solved. At each step of the iteration a set of blocks is found that is of maximum value, not violating the tonnage and slope constraints. Furthermore before the next iteration commences the blocks included in the current push-back can be discarded. In practice those blocks are removed from the data before beginning the next iteration.

Solving iteratively a push-back optimization model the push-backs are created directly without the use of nested pits. In the following the algorithm for creating a push-back series is presented:

Step 1: Set $PB^0 = \varnothing$. Set $i = 1$. Set *PB_tonnages*. Set C^u. Set *block_data*.

Step 2: **While** $\bigcup_{j=0}^{i-1} PB^j \neq C^u$ **do**

$$PB^i = \text{solve\_PB}\left(PB\_tonnages, block\_data\right)$$
$$block\_data = \text{remove\_PB}\left(block\_data, PB^i\right)$$
$$i = i + 1$$

end while

Step 3: Save results

The function *solve_PB* above is the core part of the algorithm and it relates to solving the optimization model developed in the following section. The second function in the loop reduces the problem size at each iteration by removing the blocks from the data that belong to the latest push-back found.

2.2. Push-back Optimization Model

2.2.1. Sets

$\mathcal{N}$	set of all nodes in the precedence graph representing all blocks in the block model.
E	set of all edges in the precedence graph.

2.2.2. Parameters

p_i	economic block value of block *i*.
t_i	overall tonnage of block *i*.
o_i	ore tonnage of block *i*.
mc	push-back overall tonnage.
pc	push-back ore tonnage.

2.2.3. Decision Variables

$x_i \in \{0,1\}$	binary integer variable representing block extraction. x_i is equals one if block *i* is extracted otherwise zero.

2.2.4. Optimization Model

$$\max \sum_{i \in \mathcal{N}} p_i x_i \tag{2}$$

Subject to:

$$x_i - x_j \leq 0 \qquad \forall (i,j) \in E \tag{3}$$

$$\sum_{i \in \mathcal{N}} t_i x_i \leq mc \tag{4}$$

$$\sum_{i \in \mathcal{N}} o_i x_i \leq pc \tag{5}$$

$$x_i \in \{0,1\} \qquad \forall i \in \mathcal{N} \tag{6}$$

The model presented above represents a typical BIP with few constraints. Eq. (2) represents the objective function which maximizes the sum of economic block values. If block i is extracted then x_i is equals one and the economic block value of x_i is added to the objective value. In addition the objective function is subjected to several constraints, restricting the feasible solutions. The first constraint Eq. (3) controls the precedence between blocks, characterized by the edge set E of the precedence graph. If there exists an edge from block x_i to block x_j then block x_j has to be extracted before block x_i. This is achieved by restricting the decision variables. If there exists an edge from block x_i to block x_j then x_i has to be equal to or less than x_j which is enforced by formula Eq. (3). The subsequent equations (4) and (5) control the overall tonnage and ore tonnage of the push-back. The last constraint, Eq. (6), is a binary constraint expressing that x_i can only take the value one or zero. The underlying precedence graph used to enforce slope angles and precedence relations is created by using a 1:5:9 precedence pattern where patterns are alternated between even and odd layers. Fig. 1 illustrates the precedence pattern used to create the precedence graph.

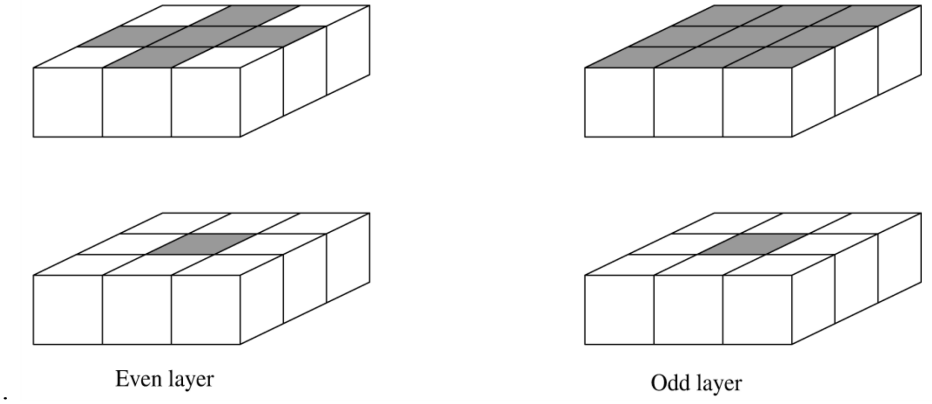

Fig. 1. Precedence pattern 1:5:9.

In Fig. 1 a block from an even layer receives five predecessors whereas a block from an odd layer receives nine predecessors. In the following sections the model defined by Eq. (2)-(6) is referred to as push-back optimization model (POM).

2.3. Structural Properties of the Model

Evaluating the model presented in the previous section, it is possible to derive two distinct features which are important for solving the POM.

In order to gain knowledge about available solution methods and their performance the push-back optimization model has to be classified. Neglecting the precedence constraints enforced by Eq. (3) turns the POM into a classic binary decision problem whose feasibility is determined by the tonnage constraints. The restrictions imposed by Eq. (4) and Eq. (5) can be viewed as a resource contingency and the tonnages t_i and o_i as the amount of resources consumed when block i is extracted ($x_i = 1$). Otherwise ($x_i = 0$) the resource consumption is zero. This structure is commonly referred to as a Knapsack problem (KP). The KP is a well studied optimization problem which can be applied to many decision problems such as portfolio selection, bin packing or cutting patterns. It shows that the POM is rooted in a KP and therefore it can be regarded as an extension of the classical KP. The problem is extended by a second knapsack and by a precedence constraint. KP's

of this structure are called a *precedence constrained knapsack problem* (PCKP) or *partially ordered knapsack problem*. Johnson and Niemi [4] studied the PCKP and showed that it is strongly *NP*-hard, meaning it is very unlikely that an algorithm, solving the problem in pseudo polynomial time, can be found. Considering the data volumes encountered in mining, we can expect long runtimes when solving the POM with exact solution methods.

The second important property of the POM is found within the precedence constraints. The coefficient matrix describing the precedence relations is the incidence matrix of the precedence graph *G*. The incidence matrix of the precedence graph consists only of 0, +1 or -1 entries. This property is called total unimodularity and therefore a significant part of the constraint matrix in the POM is totally unimodular. When the whole coefficient matrix *A* of a LP is totally unimodular and when the right hand side vector *b* is integral, then the vector of the optimal solution will be integral. In the POM the coefficient matrix of the precedence constraints is total unimodular and the right hand side vector is zero. This means that the complexity of the POM is considerably reduced when neglecting the tonnage constraints and which in turn means that the POM is well suited for relaxation techniques.

3. Solution approach

In the following a solution approach, combining reduction, relaxation and heuristic methods is presented. At first the problem size is reduced by finding the ultimate pit. Using a maximum flow algorithm the ultimate pit is computed and all blocks lying outside of the ultimate pit are discarded.

In order to find a series of push-backs the model defined in the previous section is solved iteratively. Since the POM is strongly *NP*-hard it is not always possible to solve the model using exact solution methods without encountering very long runtimes. To derive a solution in every iteration, the linear relaxation of the POM is solved followed by a greedy heuristic and a local search. A somewhat similar approach has been applied by Chicoisne et al. [2] to the multi period production scheduling problem. They also use a local search to improve their initial solution. However, they derive an initial solution using topological sorting.

3.1. Ultimate pit

The majority of commercial software for open pit mining is based on the application of the Lerchs and Grossmann (LG) algorithm to find the ultimate pit [5]. Another way of finding the ultimate pit is presented by Picard [6] and which has been used in our work.

In graph theory finding the ultimate pit can be referred to finding the maximal closure in a graph. Let $G=(\mathcal{N},E)$ be the precedence graph as described in the previous section and assume that to each node in $\mathcal{N}$ a value p_i representing the block value has been assigned. A maximum closure of *G* is a subset *Y* of nodes where $\sum_{n\in Y} p_i$ is maximal and where all predecessors of each node included in *Y* are also included in *Y*. Picard showed that the maximum closure problem can be solved as a maximum flow problem by applying the maximum flow minimum cut theorem established by Ford and Fulkerson [7]. To solve the maximum closure problem as a maximum flow problem, a new graph $\tilde{G}=(\tilde{\mathcal{N}},\tilde{E})$ based on the precedence graph $G=(\mathcal{N},E)$ has to be constructed. In the first step a source node *s* and a sink node *t* are added such that $\tilde{\mathcal{N}}=\mathcal{N}\cup\{s,t\}$.

The idea is to connect each positive valued block with an arc from the source node to the block and to connect each negative valued block with an arc from the block to the sink.

Let $N^+=\{i\in N: p_i>0\}$ and $\mathcal{N}^-=\{i\in\mathcal{N}: p_i<0\}$ then $\tilde{E}$ is created by adding arcs to *E* such that $\tilde{E}=E\cup\{(s,n): n\in\mathcal{N}^+\}\cup\{(n,t): n\in\mathcal{N}^-\}$.

The final step in the preparation of the graph is the assignment of edge capacities to each edge in $\tilde{E}$. The capacities of all original edges, i.e. the edges contained in *E,* are assigned a capacity of ∞. All outgoing edges (*s*,*n*) from the source node *s* to $n \in \mathcal{N}^+$ are assigned the capacity p_i and all incoming edges (*n*,*t*) to the sink node *t* from $n \in \mathcal{N}^-$ are assigned the capacity -p_i. The newly constructed graph $\tilde{G} = (\tilde{\mathcal{N}}, \tilde{E})$ can now be solved using a maximum flow algorithm such as the push-relabel algorithm developed by Edmonds and Karp [8]. To demonstrate the process of finding the ultimate pit, an arbitrary 2D block model has been created which is shown in Fig. 2. The black numbers in the center represent the block number and the uppercase numbers represent the block value. In Fig. 3 the corresponding precedence graph to the block model in Fig. 2 is illustrated.

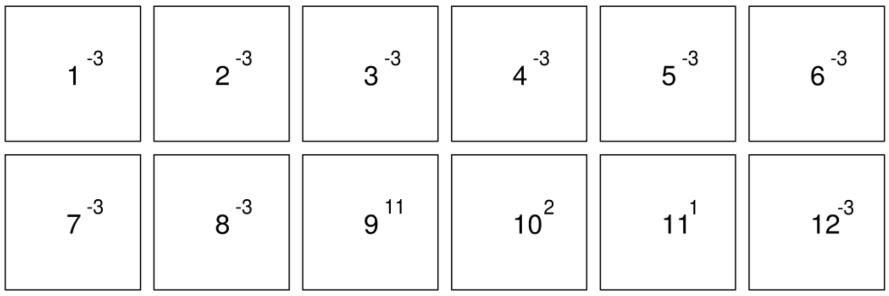

Fig. 2. Arbitrary 2D block model.

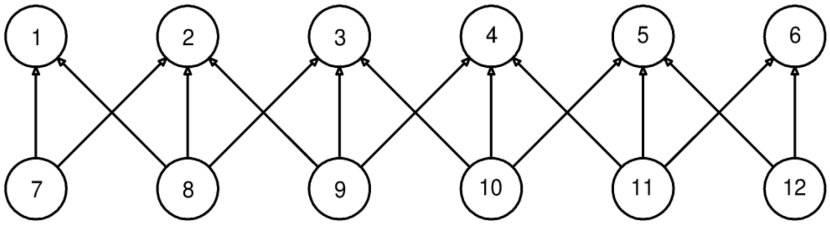

Fig. 3. Precedence graph.

The newly constructed maximum flow graph $\tilde{G} = (\tilde{\mathcal{N}}, \tilde{E})$, i.e. Picard's graph is shown in Fig. 4. The numbers on each arc represent the flow capacity of the arc.

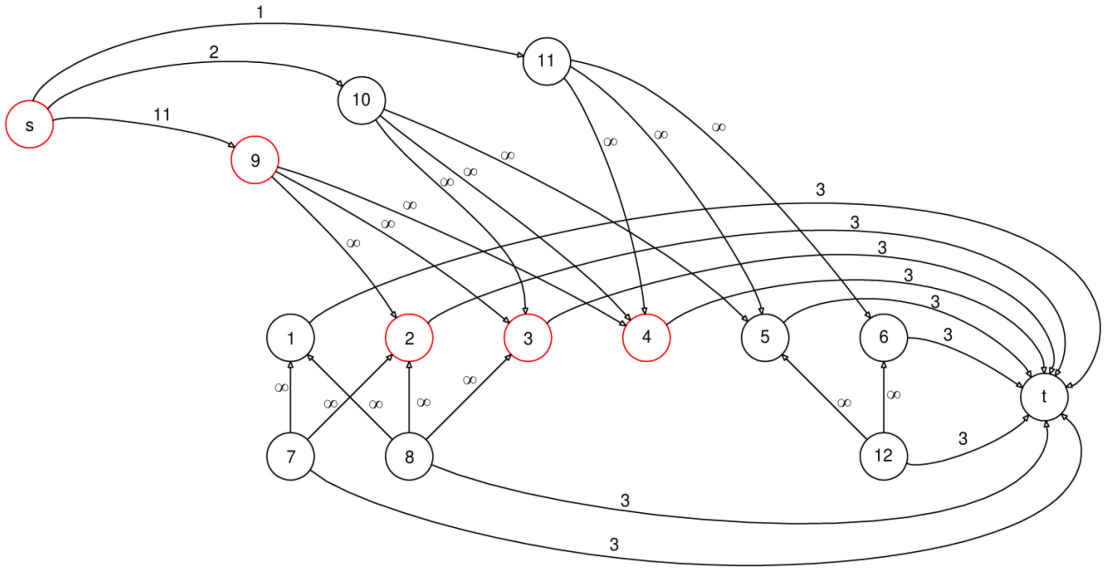

Fig. 4. Picard graph.

Solving the problem defined by Fig. 2 yields a maximum flow of 12. Since the maximum flow problem is the dual problem of the minimum cut and the maximum closure respectively, the result for the ultimate pit is found in the dual variables of the maximum flow solution. In the example defined by Fig. 2 the set *Y* of nodes in the maximum closure, consists of nodes {*s*, 9, 2, 3, 4} and has a maximum value of 2. The nodes colored in read in Fig. 4 represent the blocks which are in the maximum closure, i.e. the blocks forming the ultimate pit.

3.2. Linear relaxation

The linear relaxation yields an approximation for the push-back value which is a theoretical upper bound, meaning that the push-back value cannot exceed this bound. A second benefit of the linear relaxation is that based on the solution received from the linear relaxation it is possible to further reduce the problem size. If the result contains ones, then the blocks whose solution is one can be discarded because they will be part of the desired push-back. Vice versa, if the result contains zeros, then the blocks whose solution is zero can be discarded because they won't be part of the desired push-back. The reduction is possible due to the nestedness of the push-backs. Essentially we find the next largest and the next smallest ultimate pit close to the desired push-back and remove the blocks outside of the next larger pit and inside of the next smallest pit.

Solving the linear relaxation of very large problems such as the problems found in mining can become time consuming and memory intense. Lagrangian relaxation as alternative to linear relaxation faces the same problem of long run times because of slow convergence. In order to speed up the solution time for the POM and to reduce memory usage, the algorithm developed by Bienstock and Zuckerberg [9] is used to find the linear relaxation. Their method uses linear relaxation and Lagrangian relaxation in a hybrid way to solve the linear relaxation of precedence constrained scheduling problems. The steps of the Bienstock and Zuckerberg algorithm applied to the POM are outlined in the following:

Step 1: Set initial lagrangian multipliers $\mu^0 = 0$. Let $\mathcal{C}^0$ be the set of blocks $\mathcal{N}$. Set $k = 1$.

Step 2: Solve Lagrangian relaxation of the POM by dualizing the two tonnage constraints. Let *I* be the set of blocks included in the solution and *O* the set of blocks not included in the solution. If no set in $\mathcal{C}^{k-1}$ overlaps *I* and *O* then STOP. Otherwise partition each set in $\mathcal{C}^{k-1}$ by intersecting it with *I* and *O*. Set $\mathcal{C}^k = \mathcal{C}^{k-1}$.

Step 3: Create a reduced POM by collapsing the blocks in each set of $\mathcal{C}^k$ into a single super node and update the side constraints of the reduced problem. Solve the linear relaxation of the reduced problem and let μ^k be the optimal dual variables corresponding to the tonnage constraints.

Step 4: Step 5: If $\mu^k = \mu^{k-1}$ then STOP.

Step 5: (optional) Update $\mathcal{C}^k$ by merging the sets with the same solution value.

Step 6: Set $k = k + 1$ and GOTO Step 2.

3.3. Greedy heuristic

Greedy heuristics are a standard approach for knapsack problems to create a good initial solution. The concept of a greedy heuristic is to continuously add items to the knapsack until the knapsack capacity is full. In the POM the knapsacks are defined by the overall tonnage restriction and ore tonnage restriction for the push-back. To find a solution blocks are continuously added to the solution until no more blocks can be added without violating the tonnage restrictions. As its name suggest the greedy algorithm makes a greedy choice with respect to which item should be added first. To do so weights have to be assigned to each block, expressing the quality of the selection. This is done by a weight function. The most natural way is to assign the economic block value as

weight to each block. However, when selecting a single block to add to the solution the predecessors of the block also have to be considered and added to the solution. Therefore the weight of a specific block is the sum of economic block values of the predecessor blocks and the considered block as well. Since only profits are considered for the weight function, some choices may consume more resources, i.e. more blocks need to be extracted than a similar profitable choice. To avoid this problem, the sum of economic block values is divided by the sum of tonnages of the blocks. Therefore the blocks are chosen based on efficiency rather than on profit. Since a mining operation is only interested in mining profitable blocks only ore blocks i.e. the blocks above cut-off grade, are considered for assignment of an efficiency value.

Let $\mathcal{N}$ be the set of blocks of the residual problem created with the results of the linear relaxation. Let $\mathcal{V} \subseteq \mathcal{N}$ be the set of blocks that are ore and let $\mathcal{V}^-(v) \subseteq \mathcal{N}$ be a set consisting of the blocks that are predecessors to *v* and *v* itself. Then the efficiency $e(v)$ can be assigned to each ore block *v* as follows:

$$e(v) = \frac{\sum\left(p_i : i \in \mathcal{V}^-(v)\right)}{\sum\left(t_i : i \in \mathcal{V}^-(v)\right)} \qquad \forall v \in \mathcal{V}. \tag{7}$$

Eq. (7) shows a weight function assigning an efficiency value. The numerator in Eq. (7) is the sum of economic block values of the blocks contained in the set $\mathcal{V}^-(v)$ and the denominator in Eq. (7) is the sum of block tonnages of the blocks contained in the set $\mathcal{V}^-(v)$.

Based on the weight function the greedy algorithm can be developed. In the following, the steps of the algorithm are presented:

Step 1: Set $S^0 = \varnothing$. Set overall tonnage restriction *mc* and ore tonnage restriction *pc*. Set $k = 1$.

Step 2: Compute efficiency $e(v)$ for each $v \in \mathcal{V}$.

Step 3: Compute the set of candidates $\mathcal{K}$ by adding Blocks $v \in \mathcal{V}$ where $e(v) > 0$ and where the selection of $\mathcal{V}^-(v)$ would not violate the overall tonnage and ore tonnage restrictions. If $\mathcal{K} = \varnothing$, then STOP.

Step 4: Sort $\mathcal{K}$ such that the ordering $\{v_1, v_2, \ldots, v_n\}$ is sorted in a descending order with respect to efficiency i.e. such that $\{e(v_1) \geq e(v_2) \geq \ldots \geq e(v_n)\}$.

Step 5: Update the solution such that $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \mathcal{V}^-(v_1)$ alternatively select randomly $v_{rand} \in \mathcal{K}$ instead of v_1.

Step 6: Update the overall tonnage and ore tonnage restrictions by subtracting from *mc* and *pc* the tonnages of the blocks that have been added to the solution.

Step 7: Update the predecessors of each ore block *v* such that $\mathcal{V}^-(v) = \mathcal{V}^-(v) \setminus \mathcal{S}^k \ \forall v \in \mathcal{V}$ and remove v_1 and v_{rand} respectively from $\mathcal{V}$.

Step 8: Set $k = k + 1$ and GOTO step 2.

3.4. Local search

Combining the value of the greedy solution with the value of the blocks whose solution value from the relaxation is one yields the value of our current solution. If this value is close to the upper bound obtained from the linear relaxation, then the solution is accepted as push-back and a new

iteration is commenced. Otherwise, we attempt to improve the greedy solution by using a local search.

The algorithmic concept for a local search developed by Chicoisne et al. [4], is used in our solution approach. The idea is to select a small part of the data of the POM and resolve it using exact solution methods. Therefore the underlying model is the same as presented in section 2 but with less blocks. If the solution of the small problem improves the current solution, then the current solution is updated. Otherwise, a new part in the POM is chosen and resolved.

In order to re-optimize parts of the current solution, a neighborhood relation has to be defined. Because the underlying problem is a three dimensional block model, selecting neighborhoods based on spatial distribution would be the most logical decision. The ore blocks which are close to the current solution should be evaluated and re-optimized. Based on this concept a neighborhood can be constructed by selecting ore blocks close to the current solution. When adding a block to the neighborhood, the predecessor relations have to be respected. Therefore the predecessors of a selected block also have to be added to the neighborhood.

To respect the slope constraints of the blocks not in the neighborhood, the predecessors above a selected ore block have to be fixed each time new blocks are added to the neighborhood. The difference between the predecessors above a selected ore block and the resulting neighborhood is illustrated by Fig. 5 which shows the neighborhood selection for one ore block outside the current solution.

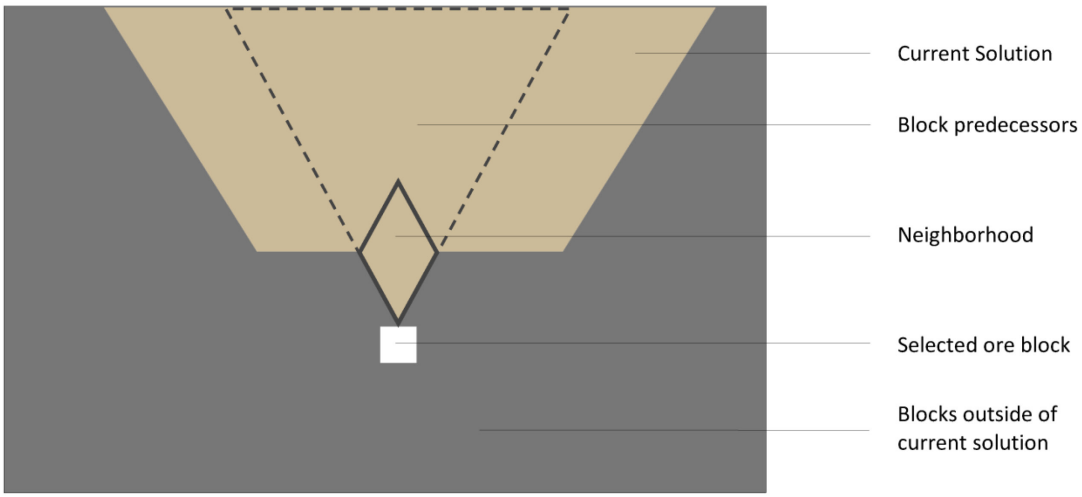

Fig. 5. Illustration of neighborhood selection.

To apply the local search effectively and to enhance the chance of finding an improved solution, the neighborhood has to be as large as possible. On the other hand the neighborhood should have a size small enough to solve the neighborhood quickly because otherwise solving the neighborhood can become time consuming or even intractable.

In the following the local search algorithm is presented:

Step 1: Set $\mathcal{S}^0$ as the set of blocks included in the current solution. Set $\mathcal{V}$ as the set of ore blocks not included in the current solution. Set maximum number of iterations. Set $k=1$. Set maximum allowed size of the neighborhood.

Step 2: Calculate the distance $d(v) \quad \forall v \in \mathcal{V}$ to the closest block in $\mathcal{S}^{k-1}$.

Step 3: Sort $\mathcal{V}$ such that the ordering $\{v_1, v_2, \ldots, v_n\}$ is sorted in an ascending order with respect to distance i.e. such that $\{d(v_1) \leq d(v_2) \leq \ldots \leq d(v_n)\}$.

Step 4: Let $\mathcal{H} = \varnothing$ be the set defining the neighborhood. Add blocks $\{v_1, v_2, \ldots, v_n\}$ and its predecessors to $\mathcal{H}$ until the maximum size of $\mathcal{H}$ is reached.

Step 5: Remove the ore blocks from $\mathcal{V}$ which have been added to $\mathcal{H}$. Remove the blocks from $\mathcal{H}$ which are predecessors to blocks in $\mathcal{S}^{k-1}$.

Step 6: Create a reduced POM defined by the blocks in neighborhood $\mathcal{H}$

Step 7: Solve the reduced POM. If the solution improves the current solution then update the current solution

Step 8: If maximum number of iterations is reached, then STOP. Otherwise GOTO step 4.

4. Implementation and case study

Two separate programs were developed using the programming environment MATLAB [10]. The first program finds the ultimate pit and the second creates the push-backs. In addition to the two main programs, an export program was also developed. A flow chart of the push-back program is illustrated by Fig. 6. To solve IP's, LP's and Network problems the CPLEX solver [11] is used and which is called through TOMLAB optimization environment [12].

The newly developed programs have been tested on three different deposits. The data sets used for creating push-backs series consist of a copper deposit with 18,890 blocks in the ultimate pit, an iron ore deposit with 207,699 blocks in the ultimate pit and a gold deposit with 277,089 blocks in the ultimate pit. The gold deposit also has copper and silver as by-product. Each deposit was provided in the form of a Gemcom Project file. A geological block model, where the ore bodies had been modeled, was also included in the project file. In addition ore and waste blocks had already been assigned by applying a cut-off grade. The mining software package Gemcom Gems [13] was used to access the data and to create economic block models.

All experiments were carried out on a Dell Precision T7500 machine with two quad-core 2.8GHz processors and with 24GB of memory. The parameters illustrated in Table 1 have been used in the push-back program for development of push-back series.

Table 1. Parameters used in case study.

IP acceptance:	35,000 blocks
Early stopping criterion (incl. IP solver):	1% gap
Greedy repetitions:	500
Greedy randomness:	0.3
Max neighborhood size:	20,000 blocks

The IP acceptance parameter sets the maximum allowed input size for using the IP-solver on the reduced problem. If the reduced problem received from the linear relaxation contains more than 35,000 blocks then the greedy algorithm is used instead of the IP-solver. Throughout the push-back program the current solution is tested against the value of the upper bound. If the gap of the push-back to the upper bound is lower than the early stopping criterion, then the current solution is accepted as push-back. Otherwise the program continues. In order to explore different solutions the greedy algorithm is set to be executed 500 times with a randomness of 0.3. Each time an item is randomly selected out of the first 30% of the candidate list instead of choosing always the first (best) item. The last parameter in Table 1 sets the maximum allowed size of the neighborhood used in the local search to 20,000 blocks.

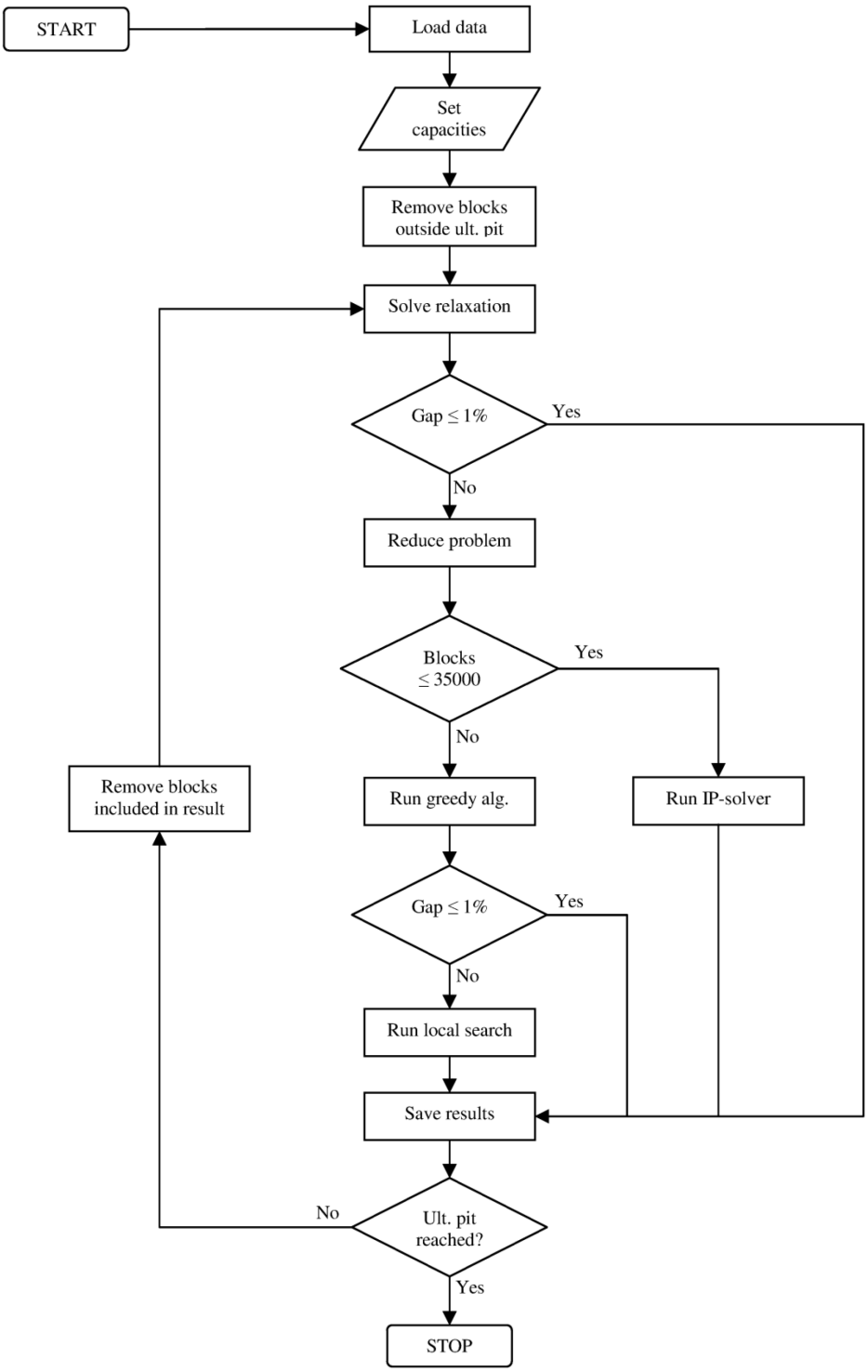

Fig. 6. Flowchart push-back program.

Since the greedy algorithm and the local search are heuristics, it is not possible to guarantee the optimality of the newly developed push-backs. However, it is possible to estimate the quality of the solutions found by comparing it to the approximation in form of the upper bound found by the linear relaxation. The real optimal solution can be close to the approximation but it can also be significantly lower than the approximation. Therefore the comparison of the solution to the

approximation is not ideal but it is the only way to evaluate the solution with regards to optimality. In the case study the comparison of the solution to the approximation is referred to as the *gap*, not to be confused with the gap of the gap problem. Eq, (8) illustrates the *gap* calculation used in the case study where z^u is the upper bound, i.e. the approximation and where z^l is the lower bound, i.e. the value of the current solution.

$$gap = \frac{(z^u - z^l)}{z^l} \tag{8}$$

During the implementation of the method proposed by Bienstock and Zuckerberg for the linear relaxation, it showed that for large deposits the approximation is sometimes slightly lower than the approximation received from solving the linear relaxation directly. Therefore it is possible that a push-back may have a negative gap which otherwise would be impossible.

For each data set three push-back series were created by using different tonnage restrictions. The restrictions are created by dividing the overall tonnage and the ore tonnage of the ultimate pit by 5, 10 and 15. This is comparable to creating a minimum number of 5, 10 and 15 push-backs.

5. Results

5.1. Push-back creation

The results for the copper deposit are illustrated by Fig. 7 and Fig. 8. Each diagram illustrates the net values as well as the overall tonnages and ore tonnages of the individual push-backs. In addition, the underlying tonnage restrictions are illustrated by two horizontal lines. In all three push-back series the ore tonnage restriction is quasi fully used and utilization only decreases in the higher push-backs because of the increasing stripping ratio. This is also reflected by the value of the push-backs which generally decreases with advancement of the push-backs. In contrast to the ore tonnage restriction, the overall tonnage restriction is used less regular. Only in Fig. 7 (a) the utilization of the overall tonnage restriction follows a more regular pattern. It shows that in the beginning the overall tonnage restriction is not fully used due to the low stripping ratio at the beginning. Therefore in the initial push-backs the ore tonnage restriction is the only binding constraint.

In Table 2 the solution time and gap for each push-back are shown. Due to the small size of the problem, almost every push-back is solved using the IP-solver which is indicated by the asterisks. The size of the problem is also reflected in the short runtimes of the program. The smallest push-back series only incurred a total runtime of 69.6 seconds and the largest push-back series incurred a total runtime of 108.9 seconds. Analyzing the gaps in the largest push-back series, it shows that push-back number fifteen has a gap of 150.83% meaning that the approximation is one and a half times greater than the solution found. Nevertheless, the solution found is an optimal solution to the push-back since it was solved using the IP-solver. This shows that a high gap does not necessarily mean that the solution found is of poor quality. Table 2 also shows that the last push-back in each series always has a gap of zero. Since the last push-back completes the push-back series by arriving at the ultimate pit the approximation and the solution value are the same.

Fig. 9 and Fig. 10 illustrate the results for the iron ore deposit. In all three push-back series it shows that there is at least one big drop in ore tonnage and hence in net value. In Fig. 9 (a) the drop occurs in push-back three, in Fig. 9 (b) the drop occurs in push-back 5 and in Fig. 10 the drop occurs in push-back 8. In Table 4 the solution times and gaps for each push-back series are shown. The table shows that fewer push-backs have been solved using the IP-solver because of the large size of the iron ore deposit. Compared to the copper deposit the runtimes have dramatically increased. The smallest push-back series incurred a total runtime of 1.8 hours and the largest push-back series incurred a total runtime of 2.6 hours. Especially those push-backs which lie around the third push-

back in the smallest series experienced long runtimes. The drop in net value witnessed in Fig. 9 and Fig. 10 and the long run times illustrated in Table 4 as well as elevated gaps all occur at the same push-back.

Analyzing the gaps in the largest push-back series it shows that push-back number eight has a highly elevated gap of 316.31%. Table 4 also shows that there were fluctuations in the approximation for the upper bound. The second push-back in the largest push-back series has a gap of -8.32% meaning that the approximation is inaccurate.

The results for the gold deposit are illustrated by Fig. 11 and Fig. 12. The diagrams exhibit well balanced tonnages with only few deviations. Especially the push-back tonnages illustrated in Fig. 11 (a) are well distributed and the tonnage restrictions are efficiently used. However, this may be due to the low resolution when solving only few push-backs. A lower resolution allows for considering larger parts of the deposit and mixing different parts of the deposit.

The solution times and gaps for the push-back series are displayed in Table 3. Compared to the other two deposits the gold deposit incurred an additional increase in runtimes. The smallest push-back series experienced a total runtime of 4.9 hours and the largest push back series incurred a total runtime of 10.6 hours. The elevated runtimes are caused by the size of the problem and the low stripping ratio. In the gold deposit the share of ore blocks is very high. Therefore the greedy algorithm has to consider much more candidate blocks. Analyzing the gaps, it shows that the largest gaps occur at the same spots where the long runtimes are incurred.

5.2. Results verification

In order to estimate the performance of the push-back program and the quality indicator, the optimal solution for some push-backs has been computed. The iron ore deposit is the only deposit in the data series, exhibiting a large gap problem. Therefore the results validation has been focused on the iron ore deposit.

To analyze the quality of the results, the push-back with the worst gap in each push-back series was selected and resolved using an IP-solver. Table 5 shows the gap of each selected push-back to the optimal solution as well as the runtime incurred to find the optimal solution for the push-back. It shows that in reality the solutions are much closer to the optimal solution than indicated by the comparison of the results to the approximation. Push-back number three (original gap 56.62%) of the first series only has a real gap of 13.65%. The fifth push-back (original gap 122.98%) of the second series also shows a much lower real gap of 10.59%. However, the validation of push-back number eight of the last push-back series shows that the push-back is of inferior quality. With a real gap of 125.07% the push-back is less than half as good as the optimal solution. It appears that the program got stuck in a very low local optimum. Table 5 also illustrates the tradeoff between runtime and solution quality. For the first verification run the computer needed 22.6 hours to find the exact solution whereas only 0.5 hours where needed by the push-back program. The second verification run incurred an even longer runtime of 35.3 hours. Only the last verification run incurred a much shorter runtime of 9.4 hours further amplifying the presumption that the program became stuck in a low local optimum.

5.3. Comparison to conventional parameterization

As noted previously, a large gap problem is present in the iron ore deposit, making the deposit an ideal candidate to compare the push-back program against conventional parameterization. The push-back series created from parameterization is based on 100 nested pits. The pits were manually selected by choosing pits of maximum value, not violating the push-back tonnage restrictions. In case that no nested pit exists that complies with the tonnage constraints of a push-back, then the next best pit with the lowest deviation from the tonnage constraints is chosen.

Fig. 13 and Fig. 14 illustrate the comparison of overall push-back tonnage created from parameterization and from the push-back program. It shows that the push-backs from

parameterization are irregular and violate the overall tonnage restriction. On the other hand the push-backs created with the push-back program are more evenly distributed and respect the overall tonnage restriction. Fig. 15 and Fig. 16 illustrate the comparison of ore tonnages of the push-backs created from parameterization and from the push-back program. It shows that the ore tonnage restrictions are also violated in the push-backs created by parameterization. In addition in Fig. 15 (b) the ore tonnage restriction is violated twice. The push-backs created by the push-back program also suffer from irregularities. But overall the ore tonnage restriction is used more efficiently than by the push-backs created using parameterization.

Analyzing Fig. 15 and Fig. 16 it shows that the drop in ore tonnage explained earlier appears in the region where the gap problem occurs. Furthermore the push-backs exhibit elevated gaps. When the gap problem occurs the heuristic has to consider more data for creating a solution. Hence finding a good solution becomes more difficult which is also reflected by the push-back results.

6. Conclusions

Push-back design using ultimate pit parameterization can experience large jumps in push-back size which is referred to as the gap problem. We presented an alternative approach for the ultimate pit parameterization to create a series of push-backs not exceeding a predefined size regarding overall tonnage and ore tonnage.

The comparison of push-backs designed by parameterization to push-backs created with the push-back program showed that it is possible to overcome the gap problem. Especially large jumps in tonnage occurring during ultimate pit parameterization can effectively be removed without encountering very long runtimes. However, the tradeoff is loss of optimality.

To further evaluate the quality of the push-back program a production schedule should be created and compared to a production schedule developed from conventional parameterization.

The results validation in the previous section showed that the push-back program does not always find a solution close to the optimal solution. Therefore, future research should also be focused on improving the heuristics used in the push-back program and make them less susceptible to getting stuck in local optima.

In the POM, only two constraints are used besides the precedence constraints, illustrating the conceptual state of the model. It is evident that the model's relevance to a real life problem is limited. To improve the practicality of the program more constraints such as grade blending constraints could be added to the model in the future.

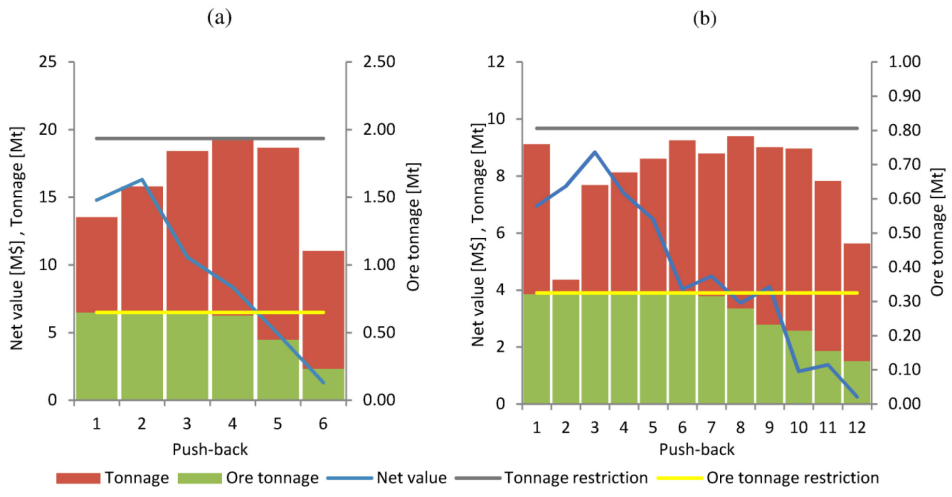

Fig. 7. Results copper deposit: (a) min. 5 push-backs, (b) min. 10 push-backs.

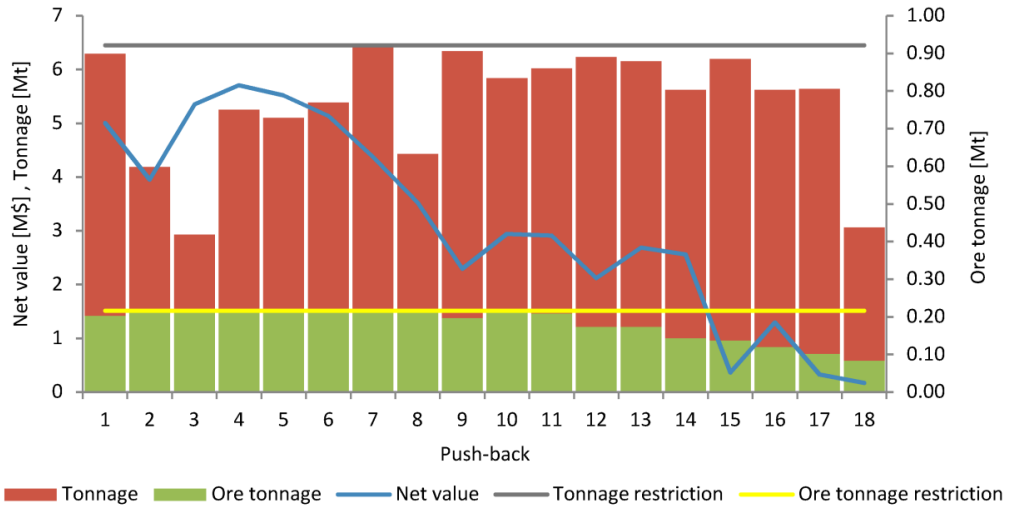

Fig. 8. Results copper deposit min. 15 push-backs.

(a) (b)

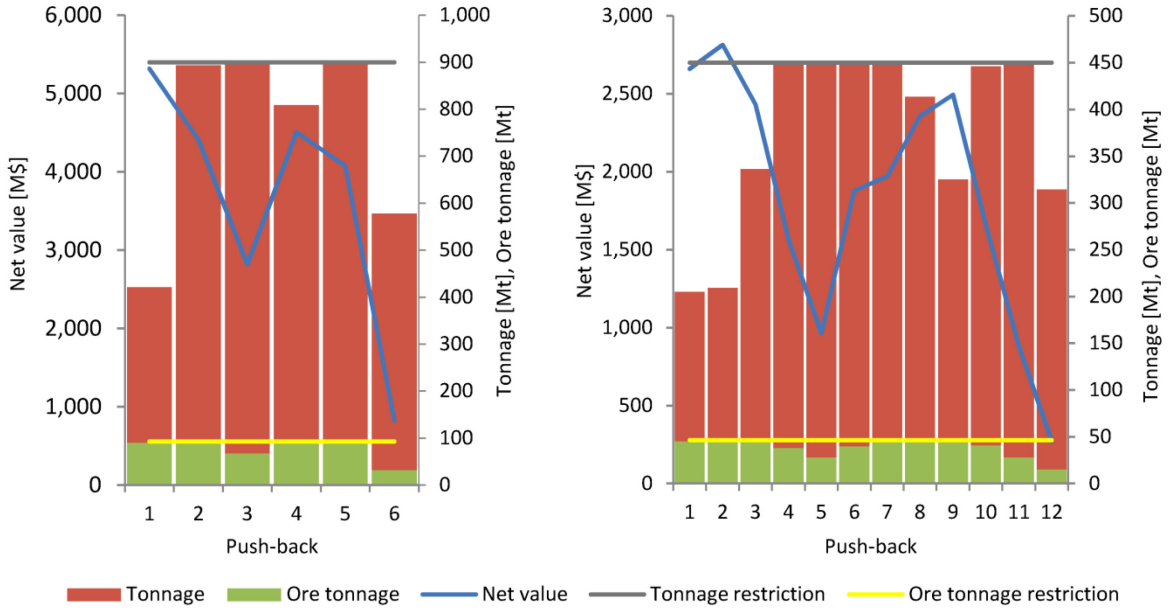

Fig. 9. Results iron ore deposit: (a) min. 5 push-backs, (b) min. 10 push-backs.

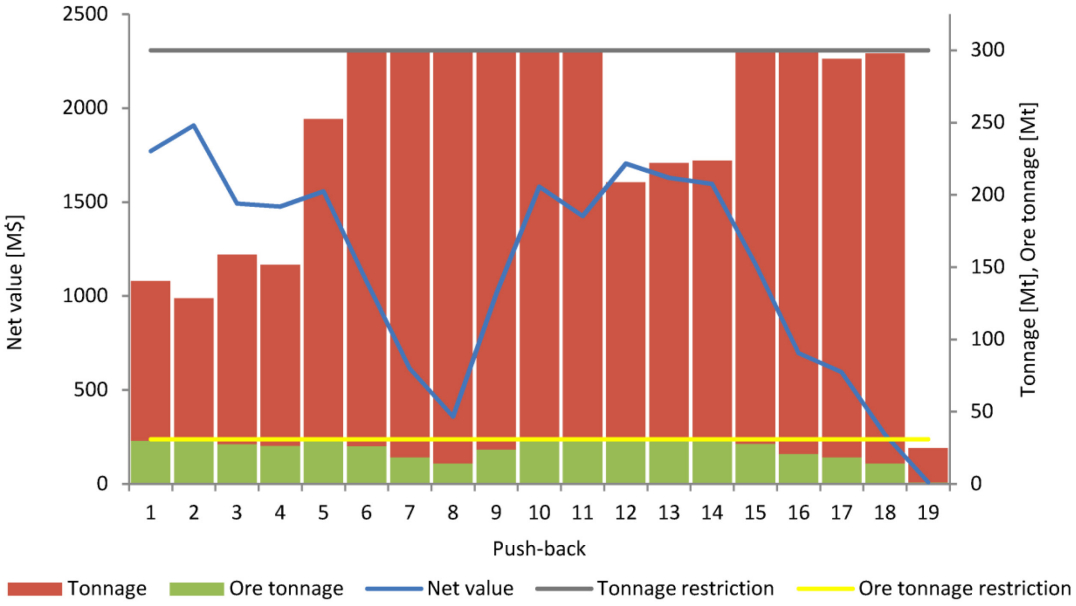

Fig. 10. Results iron ore deposit min. 15 push-backs.

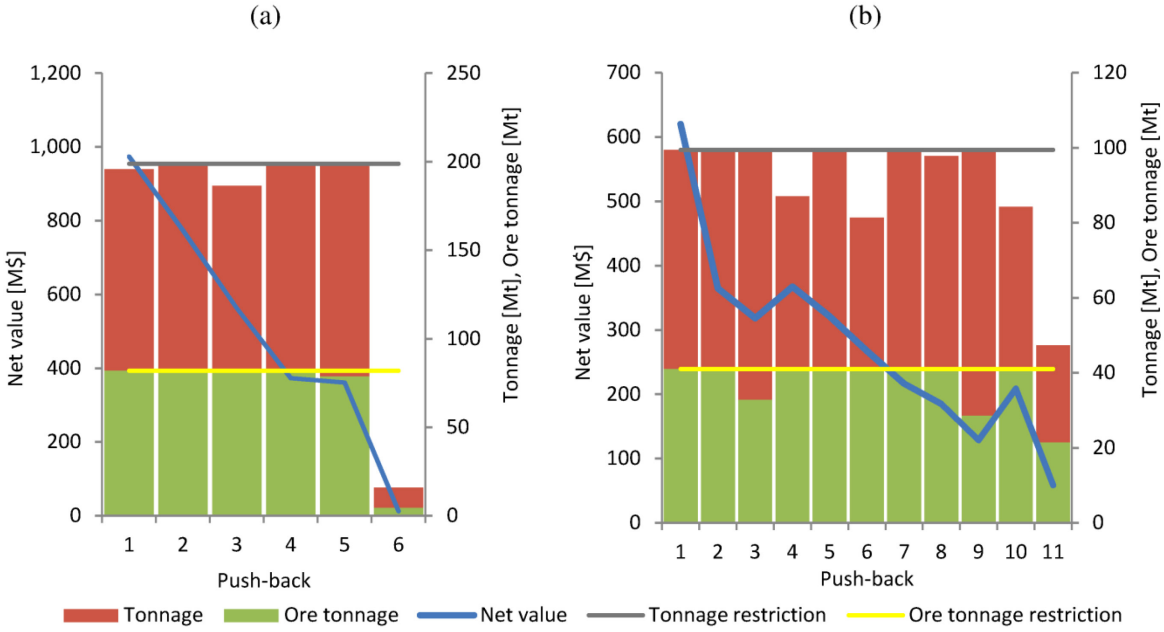

Fig. 11. Results gold deposit: (a) min. 5 push-backs, (b) min. 10 push-backs.

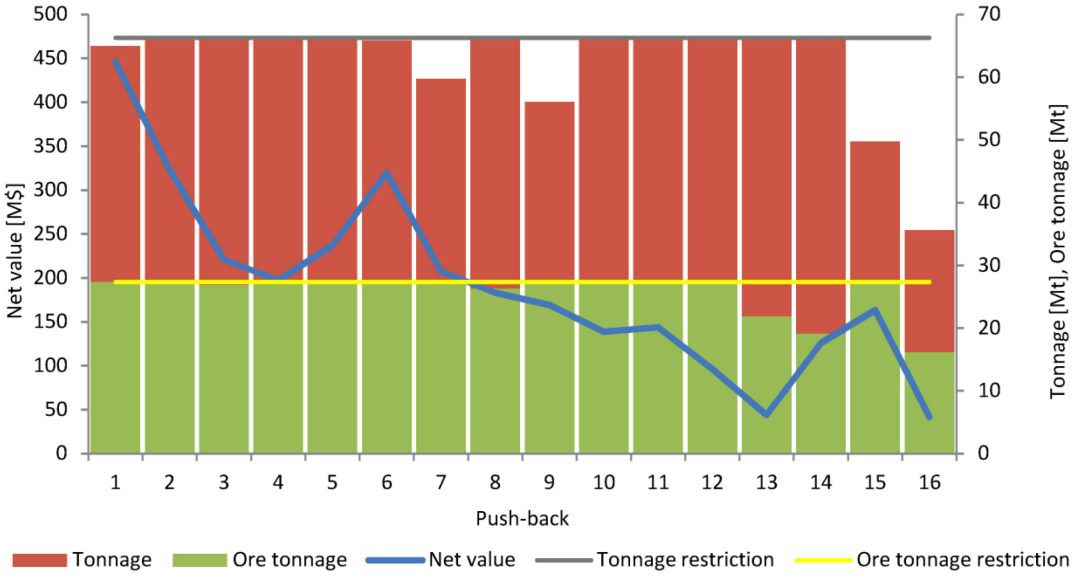

Fig. 12. Results gold deposit min. 15 push-backs.

Table 2. Gaps and solution times for copper deposit

	Min. 5 push-backs		Min. 10 push-backs		Min. 15 push-backs	
	Gap	Time	Gap	Time	Gap	Time
PB	[%]	[s]	[%]	[s]	[%]	[s]
1	*0.08	28.0	*28.20	17.0	*28.43	16.2
2	*5.84	9.3	*9.03	11.3	*32.75	13.1
3	*56.62	22.6	0.96	8.0	*7.56	9.5
4	*3.87	8.1	*2.50	6.3	*5.68	8.1
5	*1.64	1.5	*3.85	5.0	*4.32	6.9
6	0.00	0.1	*17.18	23.8	*1.01	6.1
7			*8.80	8.2	*1.66	5.5
8			*15.04	4.4	*9.55	11.6
9			*5.32	0.5	*38.36	14.6
10			*49.72	1.4	*13.15	2.7
11			*5.93	0.2	*12.48	7.8
12			0.00	0.1	*27.36	3.8
13					*16.37	0.7
14					*4.53	0.5
15					*150.83	1.2
16					*5.00	0.2
17					*20.43	0.3
18					0.00	0.1

* Push-back solved using the IP-solver

Table 3. Gaps and solution times for gold deposit.

	Min. 5 push-backs		Min. 10 push-backs		Min. 15 push-backs	
	Gap	Time	Gap	Time	Gap	Time
PB	[%]	[s]	[%]	[s]	[%]	[s]
1	4.75	3,583	*0.18	748	*-0.30	492
2	*0.13	1,282	10.81	4,661	*-0.09	1,135
3	*-0.35	978	20.16	2,432	13.04	3,178
4	7.41	11,703	*0.24	673	28.50	3,667
5	*0.32	75	*0.20	2,067	*20.80	3,876
6	0.00	1	*0.07	659	*0.14	228
7			*4.64	3,982	*1.19	1,321
8			10.74	11,072	*2.98	2,937
9			46.04	16,209	*-0.35	1,181
10			*-0.64	50	*6.56	3,679
11			0.00	1	*1.17	400
12					26.08	12,797
13					187.32	3,336
14					*42.46	3,651
15					*-0.21	24
16					0.00	1

* Push-back solved using the IP-solver

Table 4. Gaps and solution times for iron ore deposit.

	Min. 5 push-backs		Min. 10 push-backs		Min. 15 push-backs	
	Gap	Time	Gap	Time	Gap	Time
PB	[%]	[s]	[%]	[s]	[%]	[s]
1	0.08	345	0.08	341	*0.10	341
2	5.84	2,435	*-7.32	493	*-8.32	475
3	56.62	1,686	*-1.31	477	*0.07	603
4	3.87	1,506	33.53	2,505	*0.04	511
5	*1.64	311	122.98	1,256	2.35	1,033
6	0.00	1	21.05	1,046	35.64	3,280
7			16.47	703	127.75	1,129
8			*2.72	1,811	316.31	534
9			*-1.80	27	53.08	677
10			0.56	66	0.49	120
11			*1.54	23	*11.36	391
12			0.00	1	*-3.73	207
13					*-2.09	64
14					*-5.45	30
15					*0.36	57
16					*5.10	87
17					*1.92	15
18					2.05	4
19					0.00	1

* Push-back solved using the IP-solver

Table 5. Results validation

Min. 5 push-backs *(push-back no.3)*		Min. 10 push-backs *(push-back no.5)*		Min. 15 push-backs *(push-back no.8)*	
Gap [%]	Time [s]	Gap [%]	Time [s]	Gap [%]	Time [s]
13.65	81,337	10.59	127,143	125.07	33,735

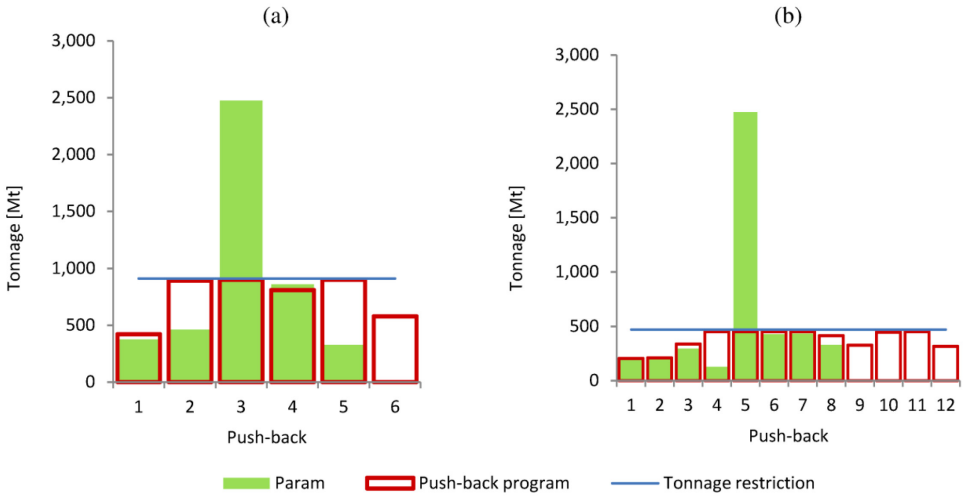

Fig. 13. Iron ore deposit push-back tonnage comparison to conventional parameterization: (a) min. 5 push-backs, (b) min. 10 push-backs.

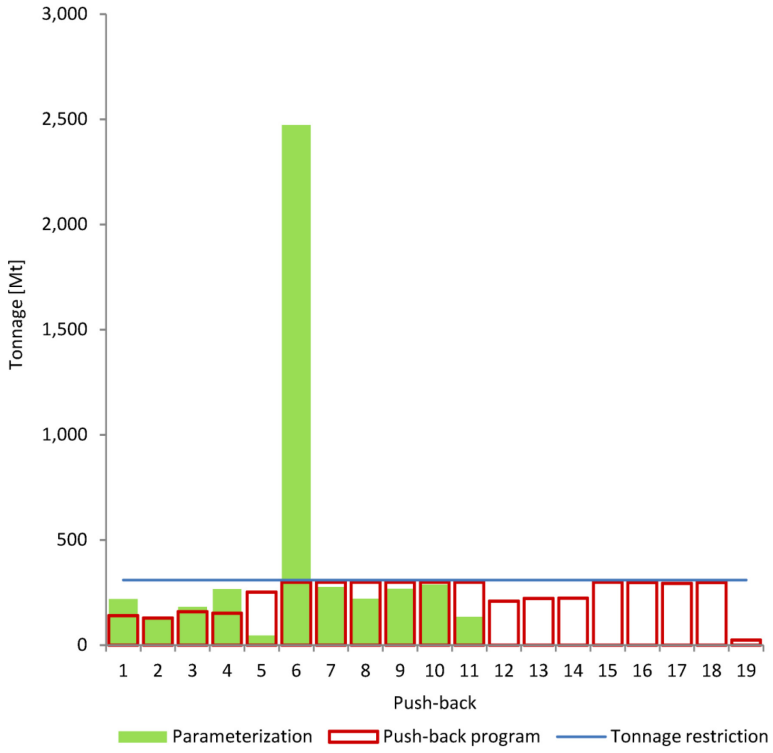

Fig. 14. Iron ore deposit push-back tonnage comparison to conventional parameterization min. 15 push-backs.

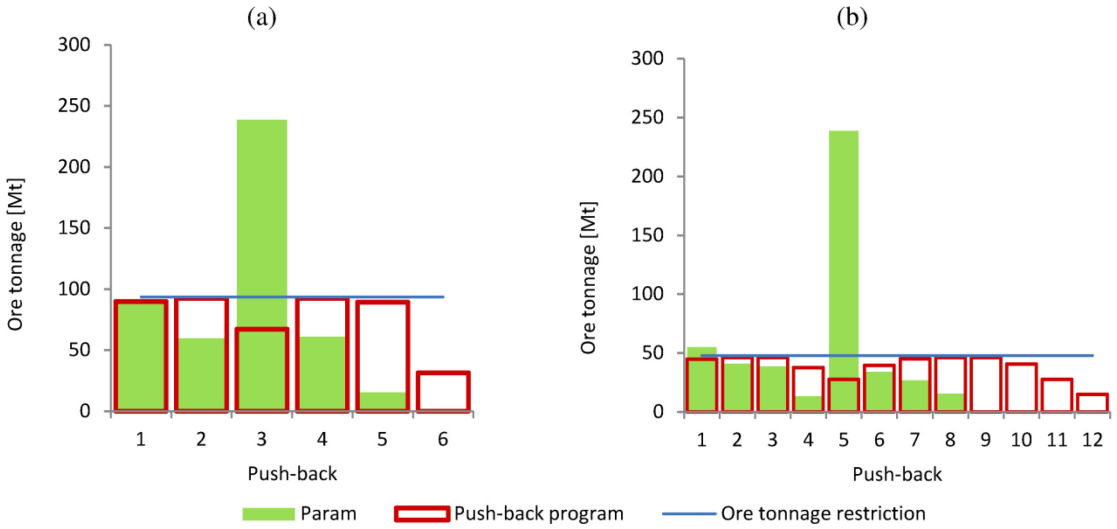

Fig. 15. Iron ore deposit push-back ore tonnage comparison to conventional parameterization: (a) min. 5 push-backs, (b) min. 10 push-backs.

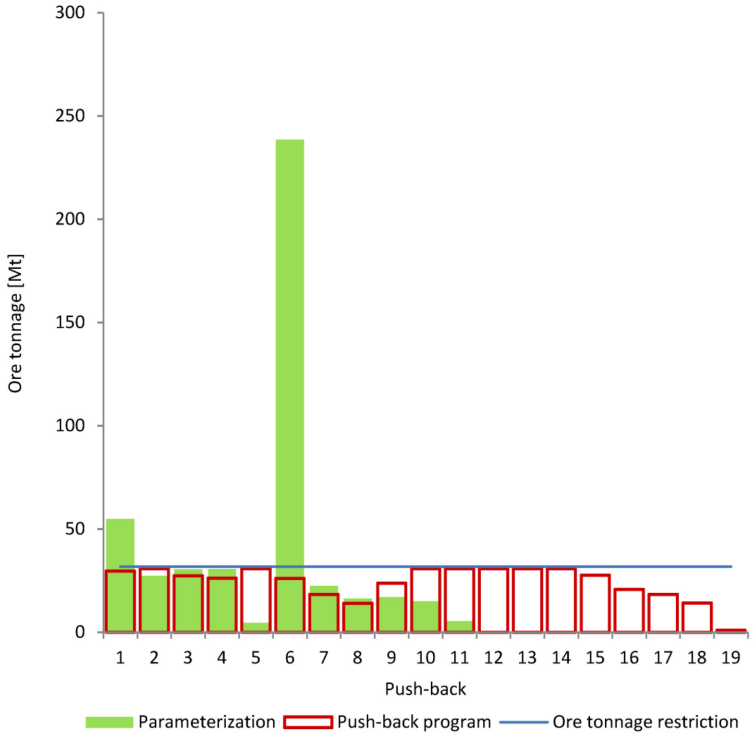

Fig. 16. Iron ore deposit push-back ore tonnage comparison to conventional parameterization min. 15 push-backs.

7. References

[1] Wahl, S. von, Fettweis, G.B., Gentz, H., Gathen, R. von der, Hahn, O., Gschwindt, E., 1990. Bergwirtschaft Verl. Glückauf, Essen.

[2] Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., Rubio, E., 2009. A new algorithm for the open-pit mine scheduling problem, accessed 16 May 2012.

[3] Newman, A.M., Rubio, E., Caro, R., Weintraub, A., Eurek, K., 2010. A Review of Operations Research in Mine Planning. Interfaces 40, pp. 222–245.

[4] Johnson, D.S., Niemi, K.A., 1983. On Knapsacks, Partitions, and a New Dynamic Programming Technique for Trees. Mathematics of Operations Research 8, pp. 1–14.

[5] Hochbaum, D.S., Chen, A., November/December 2000. Performance Analysis and Best Implementations of Old and New Algorithms for the Open-Pit Mining Problem. Operations Research 48, pp. 894–914.

[6] Picard, J.-C., 1976. Maximal Closure of a Graph and Applications to Combinatorial Problems. Management Science 22, pp. 1268–1272.

[7] Ford, L.R., Fulkerson, D.R., 1957. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. Canadian Journal of Mathematics 9, pp. 210–218.

[8] Edmonds, J., Karp, R.M., 1972. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. Journal of the ACM 19, pp. 248–264.

[9] Bienstock, D., Zuckerberg, M., 2009. A New LP Algorithm for Precedence Constrained Production Scheduling, accessed 16 May 2012.

[10] The MathWorks Inc., 2011. MATLAB version 7.13.0.564, Natick, Massachusetts.

[11] IBM Corporation, 2010. CPLEX 12.2, Armonk, New York.

[12] Tomlab Optimization, 2011. TOMLAB v7.7, Seattle, Washington.

[13] Gemcom Software International Inc., 2008. Gems 6.2, Vancouver.