A new algorithm for the open-pit mine scheduling problem

Renaud Chicoisne[*]   Daniel Espinoza[†]   Marcos Goycoolea[‡]

Eduardo Moreno[§]   Enrique Rubio[¶‖]

November 16, 2009

Abstract

For the purpose of production scheduling, open pit mines are discretized into three-dimensional arrays known as block models. Production scheduling consists in deciding which blocks should be extracted, when they should be extracted and how each extracted block should be processed. Blocks which are on top should be extracted first, and capacity constraints limit the production each time period. Since the 1960s it has been known that this problem can be modeled with integer programming. However, the large size of real instances (3-10 million blocks, 15-20 time periods) has made these models impractical for use in real planning applications, thus leading to the use of numerous heuristic methods. We propose a new decomposition method for the precedence-constrained knapsack problem, and show how we can use this to solve the LP relaxation of large production-scheduling formulations. Our computations show that we can solve, in minutes, the LP relaxation of real-sized mine-planning applications with up to 5 million blocks and 20 time periods. Combining this with a quick rounding algorithm based on topological sorting, we obtain solutions within less than 6% of optimality in seconds. A second heuristic step, based on local search, allow us to find solutions within 3% in one hour on all instances considered. For most instances we obtain solutions within 1-2% of optimality if we let this heuristic run longer. Previous methods have only been able to tackle instances with up to 150,000 blocks and 15 time periods.

Keywords: open pit mining, optimization, mixed integer programming.

[*]Université Blaise Pascal, Clermont-Ferrand, France

[†]Department of Industrial Engineering, Universidad de Chile, Chile

[‡]School of Business, Universidad Adolfo Ibañez, Chile

[§]Faculty of Engineering and Sciences, Universidad Adolfo Ibañez, Chile

[¶]Department of Mining Engineering, Universidad de Chile, Chile

[‖]Advanced Mining Technology Center, Universidad de Chile

1 Introduction

Open pit mining consists in extracting commercially valuable ore from a mineral deposit by digging from the surface as opposed to tunneling underground. Hustrulid and Kuchta [18] describe the mineral supply process of a mine in terms of three important phases: planning, implementation and production. During the planning phase mining engineers have the highest potential for influencing the ultimate value of the overall production. The ultimate goal of this phase is to prepare a feasibility report used to decide whether or not to proceed with the investment and implementation of the mineral extraction. This feasibility report should contain a tentative production schedule, and is considered a bankable document required by investors. For a recent survey of operations research applications in mine planning see Newman et al. [29]. For a detailed treatment of open-pit mine planning see Hustrulid and Kuchta [18].

The planning phase of a mine is typically subdivided into a number of steps which are solved sequentially in order to obtain a tentative production schedule. Because it is difficult to find a concise and formal description of the mine planning phase in the literature, and because it is important to understand the limitations of the *standard* approach, we briefly outline the process below.

1. *Geological ore body discretization.* This step consists in preparing a discretized model of the physical mineral deposit. By drilling in different locations and depths of the mine, samples of material are obtained and used to interpolate grade and densities throughout the deposit. With this, the mine is divided into cubes of equal size (known as blocks) each of which is assigned an estimated tonnage and estimated mineral grades (see Journel and Huidbregts [23], Isaaks and Srivastava [20]).

2. *Slope angle computations.* This step consists in computing precedence relationships between blocks of the mine. In order to protect from accidents due to rock falls, mines are typically dug in less-than-vertical angles. These angles depend on the structural composition of the rocks (faults, joints, shears, foliations, etc.) and vary depending on location and depth. Before any given block can be extracted, so must all blocks immediately above and within the required wall-slope angles (see Hustrulid et al. [19]).

3. *Economic block model.* This step consists in computing, based on tonnage and grade information, an estimated profit of extraction for each block in the model. Profit depends on cost, and cost depends on how a block is processed after extraction. That is, if the block will be sent to a mill, a waste dump, or a stockpile. Industry practitioners typically use the best possible profit among the different processing options in order to estimate a single a priori profit for each block.

   Block models with such profit attributes are typically referred to as economic block models. Formally, we will describe a block model in terms of a set of blocks $\mathcal{B}$ and a set of destinations $\mathcal{D}$. For each block $b \in \mathcal{B}$ we assume a profit of p_b^d if block b is sent to destination d for processing. We let $p_b = \max_{d \in \mathcal{D}}\{p_b^d\}$ represent the a priori profit estimation. The precedence relationships will be represented in terms of a digraph $G = (\mathcal{B}, \mathcal{A})$ where $(b, a) \in \mathcal{A}$ means that block b must be extracted before block a. We assume that digraph G only contains immediate precedence relationships. That

is, if $a, b, c \in \mathcal{B}$ are such that a must be extracted before b, and b must be extracted before c, then $(a, c) \notin \mathcal{A}$.

4. *Obtaining a final pit contour.* This step consists in delimiting the subregion of the mine in which extraction will take place, or equivalently, the part of the whole inventory which will be considered for mining. This subregion, called a final contour or ultimate pit, is obtained by solving the following problem:

$$UP = \max \sum_{b \in \mathcal{B}} p_b x_b \tag{1a}$$

$$\text{s.t.}$$

$$x_b \leq x_a \qquad \forall (a, b) \in \mathcal{A}, \tag{1b}$$

$$x_b \in \{0, 1\} \quad \forall b \in \mathcal{B}. \tag{1c}$$

In this model, variable x_b takes value one if and only if block b is to be included in the final pit. Under simple assumptions it is easy to prove that the ultimate pit contains the sets of all blocks which should be included in an optimal production schedule (For example, see Caccetta and Hill [6]).

The Ultimate Pit (UP) problem has received much attention in the literature. Lerchs and Grossman [26] first observed that it is equivalent to finding the maximum closure of a graph and proposed a customized algorithm that is still very much in use. Since then, a large number of authors have extended these two approaches in different ways. For modern algorithms and a comprehensive survey, including complexity analysis and computations, see Hochbaum and Chen [17].

5. *Determining a mining sequence.* This step consists in defining an ordering of the blocks inside the ultimate pit. To determine the ordering (typically called a mining sequence) it is standard practice to compute a sequence of nested pits by solving penalized instances of the Ultimate Pit Problem (see problem $UP(\pi)$).

$$UP(\pi) = \max \sum_{b \in \mathcal{B}} (p_b - \pi_b) x_b \tag{2a}$$

$$\text{s.t.}$$

$$x_b \leq x_a \qquad \forall (a, b) \in \mathcal{A}, \tag{2b}$$

$$x_b \in \{0, 1\} \qquad \forall b \in \mathcal{B}. \tag{2c}$$

It is well known (Lerchs and Grossman [26], Matheron [27, 28]) that given a decreasing sequence of penalization vectors $\pi^1 > \pi^2 > \ldots > \pi^k$, there exist optimal solutions x^i for each problem $UP(\pi^i)$ such that $x^1 \leq x^2 \leq \ldots \leq x^k$ (that is, such that the pits are nested), and such that x^k is optimal for UP. The sequence, or ordering of the blocks, follows directly from the nested pits. That is, blocks in the smallest pit are first, then follow blocks that are in the second pit but not in the first, etc.

6. *Computing a production schedule.* This step consists in deciding *which* blocks should be extracted, *when* they should be extracted, and *how* extracted blocks should be processed (Dagdelen [9]). This step assumes an a priori discretization of time into

3

periods, and an a priori definition of production capacity in each time period. In order to determine the time of extraction for each block a subset of nested pits from those computed in the sequencing step is selected. Each of these pits is called a pushback (or phase). It is important to select these pits in such a way that the difference between consecutive pushbacks does not contain too many blocks. Unfortunately, it may not be possible to ensure that the difference in size between consecutive pushbacks is not too large (this is known as the *gapping problem*), in which case heuristics must be used to break these up into smaller sets. After the pushbacks are defined, the blocks in the final pit are grouped together into level (or horizontal) sets called benches. Intersecting the benches and the phases it is possible to obtain a partition of the final pit in minimal connected units called bench-phases. In a final step, the bench-phases are each scheduled a time of extraction. This is done in such a way that in each time period, the scheduled bench-phases do not exceed the capacity constraints. The whole process is illustrated in Figure 1. If the resulting schedule is not satisfactory, the phases are re-defined and the process is repeated. In order to schedule the bench-phases a number of proprietary algorithms exist. Perhaps the best known algorithm for doing this is Milawa, by Gemcom Software. Once the bench-phases are scheduled a detailed plan of extraction is defined in which the material of each bench-phase is assigned a destination. This is typically done by a technique called cutoff grade optimization (see Lane [25] and King [24]). After the detailed plan is made, pushbacks are smoothed out so as to be more compliant with geometric operational requirements and haul-roads and ramps are designed in order to more accurately assess set-up and transportation costs.

The pushback generation method (as described above in its six steps) suffers from a number of important limitations. For example, processing capacity and net-present value are only taken into account after the phase-benches are computed, and multiple possible processing options for each block are only considered after blocks have been scheduled in time. Additionally, block destination is decided based on cutoff criteria rather than by considering capacities, time, and how other blocks are processed. Despite these limitations, most commercial software packages use this method, or variants thereof. Some important examples include Whittle (by Gemcom Software), DataMine, Vulcan, NPVscheduler, Comet, among others.Moreover, most of these packages are not fully automated and require many by-hand computations in steps such as pushback generation.

To our knowledge, the first attempt to address all of these issues in a holistic and exact optimization model was that of Johnson [21, 22]. Johnson considers a very general mathematical programming formulation in which blocks can have multiple destinations, and where multiple capacity constraints are simultaneously enforced. Let R represent the set of resources, $c_{r,t}$ the amount of resource r available at time t, and $q_{r,b}^d$ the amount of resource r consumed if block b is sent to destination d. Define binary variables $x_{b,t}^d$ indicating if block b has been sent to destination d by time t, and assume $p_{b,t}^d$ is of the form $p_b^d/(1+\alpha)^t$ for some non-negative constant α. Johnson's formulation [1] is presented in Figure 2.

---

[1] This model differs in technical details from that presented in Johnson [21] but captures the spirit and main modeling issues.
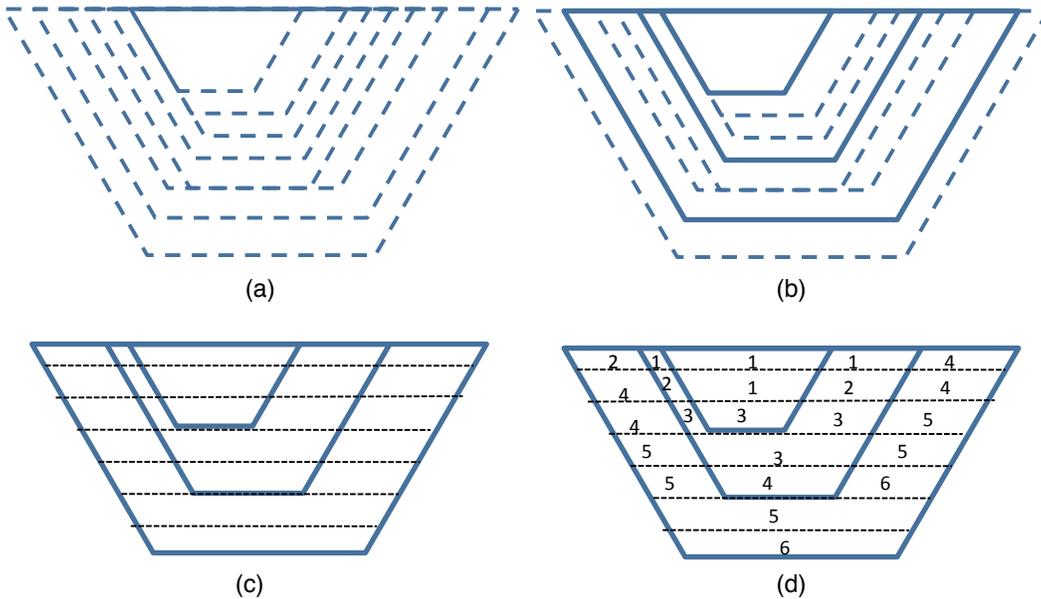
Figure 1: A production plan scheduled by pushbacks. In (a) a sequence of nested pits is defined. In (b) pushbacks are defined by selecting a subset of the nested pits. In (c) the bench-phases are defined. In (d) the bench-phases are assigned a time period of extraction.

Johnson proposed a Dantzig-Wolfe type decomposition for this problem. However, given the large size and inherent difficulty of the model, most articles in the academic literature consider some kind of simplified version. Figure 3 presents a simplified version of Johnson's formulation, which we call C-PIT, in which only a single block destination is considered. Note that in C-PIT it is assumed that the extraction methodology for each block is decided a priori, as in traditional economic block model constructions. Henceforth, we will denote by C-PIT-k the specific class of C-PIT formulations in which there are $r_{max} = k$ resources considered. Further, we will denote the LP relaxation of formulations C-PIT and C-PIT-k by R-PIT and R-PIT-k, respectively.

Observe that if one could solve the C-PIT problem (or Johnson's formulation) one could easily integrate this into current planning methodologies. Instead of scheduling bench-phases in time, one would directly be scheduling blocks. This would be preferable because it would explicitly consider net-present-value and capacity when constructing the schedule. After scheduling the blocks one could continue with the traditional methodologies.

A number of authors have developed linear and integer programming techniques for solving C-PIT. Dagdelen [8] and Dagdelen and Johnson [10] propose Lagrangian relaxation and sub gradient methods in order to solve the LP relaxation. Caccetta and Hill [6] use the solution of the ultimate pit problem for preprocessing. Ramazan and Dimitrakopolous [32] relax the integrality of variables with negative objective function value. Boland et al. [5] and Fricke [12] use cutting planes derived from the precedence constrained knapsack problem. Gaupp [13] uses early and late-start based pre-processing techniques and further

$$\max \sum_{b \in \mathcal{B}} \sum_{d=1}^{D} \sum_{t=1}^{T} p_{b,t}^d (x_{b,t}^d - x_{b,t-1}^d) \tag{3a}$$

s.t.

$$\sum_{d=1}^{D} \sum_{b \in \mathcal{B}} q_{r,b}^d (x_{b,t}^d - x_{b,t-1}^d) \leq c_{r,t} \quad \forall t = 1, \ldots, T, \ \forall r \in R \tag{3b}$$

$$\sum_{t=1}^{T} \sum_{d=1}^{D} (x_{b,t}^d - x_{b,t-1}^d) \leq 1 \qquad \forall b \in \mathcal{B} \tag{3c}$$

$$x_{b,t}^d \leq x_{a,t}^d \qquad \forall (a,b) \in \mathcal{P}, \ \forall t = 1, \ldots, T, \ \forall d \in D \tag{3d}$$

$$x_{b,t-1}^d \leq x_{b,t}^d \qquad \forall b \in \mathcal{B}, \ \forall t = 1, \ldots, T, \ \forall d \in D \tag{3e}$$

$$x_{b,t}^d \in \{0,1\} \qquad \forall b \in \mathcal{B}, \ \forall t = 1, \ldots, T, \ \forall d \in D \tag{3f}$$

$$x_{b,0}^d = 0 \qquad \forall b \in \mathcal{B}, \ \forall d \in D \tag{3g}$$

Figure 2: Johnson's Formulation

explores Lagrangian relaxation methods. Gershon [15, 16] and Fricke [12] propose a series of optimization-based heuristics.

While each of these methods has contributed to faster solution times, solving problem instances with more than $100,000$ blocks remains elusive. Perhaps the only exception to this can be found in the work of Caccetta and Hill [6] who claim to solve instances with up to $250,000$ blocks. Unfortunately, citing commercial interests, the authors do not back this claim with any replicable algorithm or methodology.

Alternative methods to using C-PIT have been proposed for simplifying Johnson's model. Ramazan et al. [31] combine integer programming with aggregation in order to reduce problem size. In their research they define large aggregated blocks called fundamental trees and solve the problem in the reduced-variable space. Gershon [14] allows the partial extraction of blocks in the bottom of the pit. More recently, Boland et al. [4] propose a formulation (henceforth BIN-PIT) in which blocks are grouped into aggregate units called bins, and impose precedence relationships only between bins. In this way they significantly reduce the number of precedence relationships and can instead incorporate multiple processing destinations for blocks. Moreover, they propose a novel decomposition approach to quickly solve the LP relaxation of this formulation. While preparing this document it has come to our attention that Bienstock and Zuckerberg [3] have further built upon this model, proposing an alternative decomposition approach for solving the LP relaxation. Their framework is very general and could also be used to solve LP relaxation of C-PIT. The focus of their article is on solving the LP relaxation of the problem, and they show they can solve relaxations of BIN-PIT instances having hundreds of thousands of blocks in just hours.

For a more detailed survey of exact optimization approaches for Johnson's model and other

6

$$\max \sum_{b \in \mathcal{B}} \sum_{t=1}^{T} p_{b,t}(x_{b,t} - x_{b,t-1}) \qquad (4a)$$

s.t.

$$\sum_{b \in \mathcal{B}} q_{r,b}(x_{b,t} - x_{b,t-1}) \leq c_{r,t} \quad \forall t \in 1 \ldots T, \ \forall r \in 1..r_{max} \qquad (4b)$$

$$x_{b,t} \leq x_{a,t} \qquad \qquad \forall (a,b) \in \mathcal{A}, \ \forall t \in 1 \ldots T \qquad (4c)$$

$$x_{b,t} \leq x_{b,t+1} \qquad \qquad \forall b \in \mathcal{B}, \ \forall t \in 1 \ldots T-1 \qquad (4d)$$

$$x_{b,t} \in \{0,1\} \qquad \qquad \forall b \in \mathcal{B}, \ \forall t \in 1 \ldots T \qquad (4e)$$

$$x_{b,0} = 0 \qquad \qquad \forall b \in \mathcal{B} \qquad (4f)$$

Figure 3: The C-PIT Formulation

variants, see Osanloo et al. [30].

In this paper we present a new methodology for solving instances of C-PIT. Our algorithm follows the ideas outlined in the pioneering work of Dagdelen [8] and Dagdelen and Johnson [10], who observe that by relaxing the capacity constraints it is possible to break-down the problem into single-time period instances. We extend their method by exploiting structural properties of the optimal multipliers, and complementing the LP relaxation algorithm with heuristics to build feasible solutions which approximate the bound. This methodology is described in Section 2, and consists of three steps: First, we solve R-PIT-1 through a new decomposition method, which we call the Critical Multiplier algorithm, that requires solving a sequence of parametrized maximum closure problems whose solutions are subsequently put together to obtain the optimal. Second, we apply a quick rounding heuristic based on topological sorting to obtain a feasible solution of C-PIT-k, and third, we apply a simple local-search heuristic to improve the quality of the obtained solution. In Section 3 we describe an implementation of the algorithm, and the results obtained from testing it on several real mine planning instances. Our computational experiments show that using our proposed methodology we can quickly obtain very good solutions (1% gap) for real-sized instances (millions of blocks) of C-PIT-1 in under two hours, and that we can obtain solutions within 5% of optimality for instances of C-PIT-2 in similar time.

2   Methodology

Observe that a valid upper bound for problem C-PIT-k can be computed by solving R-PIT-1. Unfortunately, if the instance is large enough (millions of blocks), this problem is intractable for standard commercial LP solvers. In Section 2.1 we introduce the Critical Multiplier Algorithm (CMA) for solving R-PIT-1. As we will see in the Computational Results (Section 3) solving R-PIT-1 with this proposed method is fast and results in tight bounds for all our instances of C-PIT-1 and C-PIT-2.

In order to obtain lower bounds we propose two types of heuristics. The first of these types are called TopoSort heuristics, and are described in Section 2.2. These heuristics begin by assigning weights to each block of the mine. Then, they iteratively schedule the best block currently available for extraction of the mine. We show that by using the solution of R-PIT-1 to define weights, we obtain a simple and effective type of TopoSort heuristic called the Expected TopoSort. The second class of heuristics is a simple Local-Search algorithm which works by fixing a subset of variables in a feasible solution, and optimizing over the remaining variables using a standard integer programming solver. As we will see, by combining these two types of heuristics we can quickly obtain near-optimal solutions having previously solved R-PIT-1.

2.1 An upper bound: Solving the LP relaxation of C-PIT-1

In this section we describe the Critical Multiplier Algorithm for solving R-PIT-1. The algorithm is based on two observations. The first is that in order to solve the single-time period R-PIT-1 problem, it suffices to solve two single-time period maximum closure problems and take the convex combination of the solutions. The second observation is that in order to solve the multiple-time period problem it suffices to solve a sequence of single-time period problems in the correct way, and put together the solutions.

2.1.1 The single time period case.

Define problem

$$
\begin{aligned}
CP(\kappa) = \max \quad & px \\
\text{st} \quad & \\
& qx \leq \kappa \\
& x_i \leq x_j \qquad \forall (i,j) \in \mathcal{A} \\
& 0 \leq x_i \leq 1 \qquad \forall i \in \mathcal{B}
\end{aligned}
$$

Where we assume that $q \in \mathbb{R}_+^{\mathcal{B}}, \kappa \in \mathbb{R}_+$. In this section we are concerned with efficiently solving a problem of the form $CP(\kappa)$. Observe that this corresponds to solving a single-time period version of R-PIT-1.

Let us redefine the problem UP from Equation 2 in the following way:

$$
\begin{aligned}
UP(\lambda) = \max \quad & (p - \lambda q)x \\
\text{st} \quad & \\
& x_i \leq x_j \qquad \forall (i,j) \in \mathcal{A} \\
& 0 \leq x_i \leq 1 \qquad \forall i \in \mathcal{B}
\end{aligned}
$$

Note that solutions to $UP(\lambda)$ are integral by total-unimodularity of the constraint matrix. Consider two feasible solutions x, y of $UP(\lambda)$. We say that x *dominates* y (and write $y \prec x$) if $x \neq y$ and $y \leq x$. It is easy to see that exists a maximal (with respect to inclusion) non-dominated optimal solution of $UP(\lambda)$. We henceforth denote this unique solution $x(\lambda)$. It

8

is also known that for $\mu^1, \mu^2 \in \mathbb{R}$ and the optimal non-dominated solutions $x(\mu^1), x(\mu^2)$ of $UP(\mu^1)$ and $UP(\mu^2)$ respectively, if $\mu^2 > \mu^1 \geq 0$ then $x(\mu^2) \leq x(\mu^1)$ (see for example Matheron [27]). Due to this property, we say that λ is a *critical multiplier of* UP if $x(\lambda + \varepsilon) \prec x(\lambda)$ for all $\varepsilon > 0$. Observe that if μ and ν are distinct critical multipliers of UP, by definition, $x(\mu) \prec x(\nu)$ or $x(\nu) \prec x(\mu)$. This means there is only a finite set of critical multipliers. Let $\Lambda = \{\lambda^1, \lambda^2, \ldots, \lambda^m\}$ represent the set of all critical multipliers, sorted in decreasing order.

Proposition 2.1. *For any critical multiplier λ^i, with $i < m$, solution $x(\lambda^i)$ is optimal of $UP(\lambda)$ for all $\lambda \in [\lambda^{i+1}, \lambda^i]$.*

Proof. By definition, it's easy to see that $x(\lambda^i)$ is optimal of $UP(\lambda)$ for all $\lambda \in]\lambda^{i+1}, \lambda^i]$. Note also that $UP(\lambda)$ is a continuous, convex, piece-wise linear function, and since $UP(\lambda) = (c - \lambda q)x(\lambda^i)$ for all $\lambda \in]\lambda^{i+1}, \lambda^i]$; we have, by continuity, that $UP(\lambda^{i+1}) = (c - \lambda^{i+1}q)x(\lambda^i)$. Thus proving the proposition. $\qquad\square$

The following Proposition follows directly from Everett's Theorem [11]:

Proposition 2.2. *Consider a non-negative scalar α^*. If x^* is an optimal solution of problem $UP(\alpha^*)$, then it is an optimal solution of problem $CP(qx^*)$.*

Let $Nx \leq 0$ represent the block of precedence constraints, and define the dual problems associated to $UP(\lambda)$ and $CP(\kappa)$ as follows

$$
\begin{aligned}
DUP(\lambda) = \min \quad & \mathbb{1}y_B \\
\text{st} \quad & \\
& y_B + N^T y_A \geq (p^T - \lambda q^T) \\
& y \geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
DCP(\kappa) = \min \quad & \mathbb{1}y_B + \mu\kappa \\
\text{st} \quad & \\
& y_B + N^T y_A + \mu q^T \geq p^T \\
& y, \mu \geq 0
\end{aligned}
$$

Recall that $\mu > \nu$ implies $qx(\mu) \leq qx(\nu)$. Hence, for $\kappa > 0$, one can define

$$\lambda^u = \max\{\lambda \in \Lambda : qx(\lambda) \geq \kappa\}$$
$$\lambda^l = \min\{\lambda \in \Lambda : qx(\lambda) \leq \kappa\}$$

If $\lambda^u = \lambda^l$, then $x(\lambda^u) = x(\lambda^l)$ is the optimal solution of $CP(\kappa)$. On the contrary, if $\lambda^u < \lambda^l$, then there exist critical multipliers λ^i and λ^{i+1} such that $\lambda^l = \lambda^i$ and $\lambda^u = \lambda^{i+1}$. The following Proposition shows we can still use $x(\lambda^u)$ and $x(\lambda^l)$ to obtain the optimal solution of $CP(\kappa)$.

Proposition 2.3. *Assume $\lambda^u < \lambda^l$, and let $b^u = qx(\lambda^u), b^l = qx(\lambda^l)$. Define*

$$\alpha = \frac{b^u - b}{b^u - b^l}.$$

Then, $\bar{x} = \alpha x(\lambda^l) + (1 - \alpha)x(\lambda^u)$ is optimal for $CP(\kappa)$.

9

Proof. First, note that $\bar{x}$ is feasible of $CP(\kappa)$. In fact, the precedence constraints hold by convexity, and the knapsack condition holds since

$$q\bar{x} = \alpha q x(\lambda^l) + (1-\alpha)qx(\lambda^u) = \alpha b^l + (1-\alpha)b^u = \kappa$$

Second, let (y_A^*, y_B^*) represent the optimal dual solution of $DUP(\lambda^u)$. It is easy to see that $(y_A^*, y_B^*, \lambda^u)$ is feasible for $DCP(r)$ and has objective function value $\mathbb{1}y_B^* + \lambda^u \kappa$. Since both $x(\lambda^u)$ and $x(\lambda^l)$ are optimal for $UP(\lambda^u)$ (see Proposition 2.1), it follows that

$$\mathbb{1}y_B^* = (p - \lambda^u q)x(\lambda^u) \tag{5}$$
$$\mathbb{1}y_B^* = (p - \lambda^u q)x(\lambda^l) \tag{6}$$

Taking $(1-\alpha)\cdot(5) + \alpha\cdot(6)$ we obtain

$$\begin{aligned}
\mathbb{1}y_B^* &= p((1-\alpha)x(\lambda^u) + \alpha x(\lambda^l)) - \lambda^u((1-\alpha)qx(\lambda^u) + \alpha qx(\lambda^l)) \\
&= p\bar{x} - \lambda^u((1-\alpha)b^u + \alpha b^l) \\
&= p\bar{x} - \lambda^u \kappa
\end{aligned}$$

Thus,

$$p\bar{x} = \mathbb{1}y_B^* + \lambda^u \kappa$$

By strong duality we conclude that $\bar{x}$ is optimal of $CP(\kappa)$ and $(y_A^*, y_B^*, \lambda^u)$ of its dual. $\square$

Proposition 2.4. *Let $\kappa_1 > \kappa_2 > 0$ and let $\bar{x}^1$ and $\bar{x}^2$ be its corresponding solutions of $CP(\kappa_1)$ and $CP(\kappa_2)$ constructed as before. Then, $\bar{x}^1 \geq \bar{x}^2$.*

Proof. From Proposition 2.3 we know that for some i, j and α^1, α^2 that

$$\begin{aligned}
\bar{x}^1 &= \alpha^1 x(\lambda^i) + (1-\alpha^1)x(\lambda^{i+1}) \\
\bar{x}^2 &= \alpha^2 x(\lambda^j) + (1-\alpha^2)x(\lambda^{j+1}).
\end{aligned}$$

Recall that the critical multipliers are sorted in decreasing order. That is, $l < m$ implies $\lambda^l > \lambda^m$, and thus $x(\lambda^l) \prec x(\lambda^m)$. First, suppose $i > j$. We have $x(\lambda^j) \prec x(\lambda^{j+1}) \leq x(\lambda^i) \prec x(\lambda^{i+1})$. Hence, $\bar{x}^1 \geq \bar{x}^2$. Second, suppose $i = j$. As in Proposition 2.3, let $\lambda^u = \lambda^{i+1} = \lambda^{j+1}$ and $\lambda^l = \lambda^i = \lambda^j$. Define b^u, and b^l in the same way. We have,

$$\alpha^1 = \frac{b^u - \kappa_1}{b^u - b^l} \qquad \alpha^2 = \frac{b^u - \kappa_2}{b^u - b^l}.$$

Thus, $\kappa_1 > \kappa_2$ implies $\alpha_1 < \alpha_2$. Since $x(\lambda^i) \prec x(\lambda^{i+1})$ it follows that $\bar{x}^1 \geq \bar{x}^2$. $\square$

2.1.2 Extension to the multi-time period case

Theorem 2.5. *Consider a sequence of problems,*

$$SP(t) = \max \quad p(t) \cdot x$$
$$st$$
$$x_i \leq x_j \qquad \forall (i,j) \in \mathcal{A}$$
$$q \cdot x \leq U_t$$
$$0 \leq x \leq 1$$

Assume that: (a) for each $t = 1, \ldots, T$ vector $\bar{x}^t$ is feasible and optimal for problem $SP(t)$, (b) $\bar{x}^1 \leq \bar{x}^2 \leq \ldots \leq \bar{x}^T$, (c) $\bar{x}^0 = 0^n$, and (d) for each $t = 2, \ldots, T$ we have $p^t = \frac{1}{1+r}p^{t-1}$. Then, vector $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^T)$ is optimal for problem,

$$MP = \max \quad \sum_{t=1}^{T} p(t) \cdot (x^t - x^{t-1})$$
$$st$$
$$
\begin{array}{ll}
x_i^t \leq x_j^t & \forall t = 1, \ldots, T \quad \forall (i,j) \in \mathcal{A} \\
x^t \leq x^{t+1} & \forall t = 1, \ldots, T-1 \\
q \cdot x^t \leq U_t & \forall t = 1, \ldots, T \\
0 \leq x^t \leq 1 & \forall t = 1, \ldots, T \\
x^0 = 0 &
\end{array}
$$

Proof. First, observe that we can rewrite the objective function of MP as $z(x) \equiv p(T)x^T + \left(1 - \frac{1}{1+r}\right)\sum_{t=1}^{T-1} p(t) \cdot x^t$. Second, observe that (a) and (b) imply that $\bar{x}$ is feasible for MP. Let $y = (y^1, y^2, \ldots, y^T)$ be feasible for MP. It is easy to see that y^t is feasible for $SP(t)$ for each $t = 1, \ldots, T$. Further,

$$z(\bar{x}) - z(y) = \left(p(T)\bar{x}^T - p(T)y^T\right) + \left(1 - \frac{1}{1+r}\right)\sum_{t=1}^{T-1}\left(p(t)\bar{x}^t - p(t)y^t\right).$$

However, from (a) we know that, $\left(p(t)\bar{x}^t - p(t)y^t\right) \geq 0$ for all t. This implies that $z(\bar{x}) - z(y) \geq 0$ and that $\bar{x}$ is optimal for MP. $\qquad \square$

Proposition 2.6. *If $c_t \geq 0$ and $U_t = \sum_{k=1}^{t} c_k$ for $t = 1, \ldots, T$, then an optimal solution of MP is also an optimal solution of R-PIT-1.*

Proof. Let $\hat{x}$ be the optimal solution of MP. We show that $\hat{x}$ is also feasible for R-PIT-1. For this, it suffices to prove that if $\hat{x}^t \prec \hat{x}^{t+1}$ then $q \cdot \hat{x}^t = U_t$. In fact, let D be the subsets of blocks d such that $\hat{x}_d^t < \hat{x}_d^{t+1}$. By definition of $\hat{x}^{t+1}$, $p(t+1)(\hat{x}^{t+1} - \hat{x}^t) > 0$, hence if $q \cdot \hat{x}^t < U_t$ then we can increase the value of all variables $\hat{x}_d^t$ such that $d \in D$ by ε, obtaining a new solution $\bar{x}$. Note that this new solution $\bar{x}$ has an objective value equal to the objective value of $\hat{x}$ plus $(p(t+1) - p(t))(\hat{x}^{t+1} - \hat{x}^t)\varepsilon$, which is positive number, contradicting the optimality of $\hat{x}$. $\qquad \square$

2.1.3   Putting it all together

We now describe how the results of Section 2.1.1 and Section 2.1.2 can be used to obtain the optimal solution of R-PIT-1.

Let $U_t = \sum_{k=1}^{t} c_k$ for $t = 1, \ldots, T$. By using the following two-step procedure (henceforth, The Critical Multiplier Algorithm) we obtain an optimal solution of R-PIT-1.

1. Compute all of the critical multipliers λ^i of $UP(\lambda)$ and the corresponding solutions $x(\lambda^i)$. This can be done using sensitivity analysis or binary search.
2. Using the critical multipliers computed in Step 1, obtain the optimal solution $\bar{x}^t$ of problem $CP(U_t)$ for each $t = 1, \ldots, T$, as indicated by Proposition 2.3.

To see that this in fact yields an optimal solution of R-PIT-1, first observe that since $p(t)$ is a scalar multiplication of p that $\bar{x}^t$ is also optimal for $SP(t)$ for $t = 1, \ldots, T$. From Proposition 2.4, we know that $\bar{x}^t \leq \bar{x}^{t+1}$ for $t = 1, \ldots, T - 1$. This means that $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^T)$ satisfies the conditions of Theorem 2.5. From Proposition 2.6 we know that $\bar{x}$ is also optimal solution of problem R-PIT-1. The optimal value of the problem is

$$\text{R-PIT-1} = \sum_{t=1}^{T} p(t) \cdot (\bar{x}^t - \bar{x}^{t-1}) = \sum_{t=1}^{T} \gamma_t p \bar{x}^t$$

where

$$\gamma_T = \left(\frac{1}{1+r} \right)^T \quad \text{and} \quad \gamma_t = \left(1 - \frac{1}{1+r} \right) \left(\frac{1}{1+r} \right)^t.$$

2.2   Obtaining a starting lower bound: Toposort Heuristics

Recall that $(\mathcal{B}, \mathcal{A})$ defines a directed graph, and assume $|\mathcal{B}| = n$. Let $\mathcal{B}^-(b)$ denote the set of all blocks in $\mathcal{B}$ that are in the precedence of b, and $\mathcal{B}^+(b)$ denote the set of blocks in $\mathcal{B}$ for which b is in their precedence. An ordering of the blocks $\{b_1, b_2, \ldots, b_n\}$ is a *topological ordering* if $b_i \to b_j$ implies $i < j$, or equivalently, if $\mathcal{B}^-(b_i) \subseteq \{b_1, \ldots, b_{i-1}\}$ for each $i, j \in 1, \ldots, n$. Observe that graph $(\mathcal{B}, \mathcal{A})$ does not have any directed cycles (as there can be no circular precedence relationships). It is well known that if a directed graph does not have any directed cycles, then it contains a topological ordering of the nodes. Given a weight $w(b)$ for each block $b \in \mathcal{B}$ we say that the ordering $\{b_1, b_2, \ldots, b_n\}$ is topologically sorted with respect to w if it is a topological ordering, and in addition, $(|i - j| = 1) \wedge (b_i \nrightarrow b_j) \wedge (b_i \nleftarrow b_j) \wedge (w_i < w_j)$ implies $(i < j)$. Algorithm 1 shows how it is possible to obtain a topological sorting of the blocks in a mine.

Given a topological ordering of the blocks in a mine, it is easy to obtain a feasible solution to C-PIT. In fact, it is simply a matter of sequentially scheduling each block of the mine in the order prescribed by the topological ordering, scheduling each block for extraction as early as possible. In order to determine at what time, exactly, to extract each block,

---

Algorithm 1: $\mathtt{TSort}(G, w)$: Topologically sorting the nodes of a directed graph

---

Require: Directed graph $G = (V, E)$ without directed cycles, $w \in \mathbb{R}^V$.

1: $i \leftarrow 1, n \leftarrow |V|$
2: **while** $i < n$ **do**
3: $\quad u \leftarrow \mathrm{argmin}\{w_u : \delta^-(u) = \emptyset\}$
4: $\quad G \leftarrow G \setminus \{u\}$ (removing any dangling edge).
5: $\quad v_i \leftarrow u, i \leftarrow i + 1$.
6: **return** Order $v_1, \ldots, v_n$ s.t. $\forall (v_i, v_j) \in E, i < j$ and $\forall i = 1, \ldots, n$ either $(v_i, v_{i+1}) \in E$ or $w_i \leq w_j$.

---

first observe that a block can be extracted no earlier than its predecessors and that there must be enough resources available for its extraction. Note that because of the ordering, all of a block's predecessors will have been scheduled before it is reached, so it is possible to compute its earliest possible extraction time. If a predecessor of a block has been scheduled for non-extraction, then that block must be scheduled for non-extraction as well. This suggests a natural heuristic which is described in Algorithm 2.

Algorithm 2: $\mathtt{THeur}(G, w)$: The TopoSort Heuristic

Require: $G = (V, E)$ graph of precedence constraints, $w \in \mathbb{R}^V$.
$\quad T_{max}$ planning horizon, r_{max} number of resource constraints.
$\quad q_{r,b} \in \mathbb{R}^+$ is the amount of resource r that block $b \in V$ uses if extracted.
$\quad c_{r,t}$ availability of resource r in time t.

1: $t_b \leftarrow 1, \forall b \in V, i \leftarrow 1$.
2: $\{b_1, \ldots, b_n\} \leftarrow \mathtt{TSort}(G, w)$.
3: **while** $i < n$ **do**
4: $\quad u \leftarrow v_i, i \leftarrow i + 1$.
5: $\quad t_u \leftarrow \max\{t_v : (v, u) \in E\}$.
6: $\quad t_u \leftarrow \max\{t_u, \min\{t : c_{r,t} \geq q_{r,u} \forall r = 1, \ldots, r_{max}\}\}$.
7: $\quad$ **if** $t_u \leq T_{max}$ **then**
8: $\quad\quad c_{r,t_u} \leftarrow c_{r,t_u} - q_{r,u}, \forall r = 1, \ldots, r_{max}$.
9: **return** $\{t_b\}_{b \in V}$.

Observe that Algorithm 2 describes not just one, but rather an entire class of heuristic algorithms for C-PIT. In fact, different ways of assigning weights to the blocks of the mine could lead to different feasible solutions of the problem. Furthermore, observe that Algorithm 2 generalizes the heuristic of Gershon [15]. In fact, Gershon's heuristic corresponds to the case in which the weight of each block is computed as,

$$w(b) = \sum (p_a : a \in \mathcal{B}^-(b)) \qquad \forall b \in \mathcal{B}. \tag{7}$$

Another natural variant can be defined by assigning weights,

$$w(b) = p_b \qquad \forall b \in \mathcal{B}. \tag{8}$$

13

We henceforth call this the greedy heuristic. We propose a different weight function that use the linear programming relaxation solution of the C-PIT problem. Let x^* be this solution, define weights

$$w(b) = (T + 1) - \sum_{i=1}^{T} x_{b,t}^*$$

The interpretation of this weight is to understand x^* like a cumulative distribution probability of the time of extraction, so $w(b)$ is the expected time of extraction of block b. Note that if $(a, b) \in \mathcal{A}$, then $w(b) \leq w(a)$. Moreover, if x^* is the optimal (integer) solution of C-PIT, then $w(b)$ is either equal to the time of extraction of block b, or equal to $T + 1$ if b is not extracted, hence Toposort using this weight function will obtain the optimal solution. We call this variant the Expected-Time (ET) heuristic.

2.3   Improving the lower bound: A Local-search Heuristic

We now describe a simple (and scalable) local-search method for obtaining a sequence of improving solutions given a starting feasible solution. This algorithm is an enhanced version of that presented by Amaya et al. [2]. Let x^k be a feasible integral solution (henceforth the incumbent) and let D represent a set of blocks in $\mathcal{B}$. At each iteration we determine if there exists a solution x^{k+1} of C-PIT, strictly better than x^k, such that

$$x_{b,t}^{k+1} = x_{b,t}^k \; \forall t = 1, \ldots, T, \; \forall b \notin D,$$

If no such improving solution exists, we define a new set of blocks D and try again. It is easy to see that, given a solution x^k, attempting to find a solution x^{k+1} reduces to solving an instance of C-PIT in which we must additionally consider border conditions for some of the blocks. That is, after substituting out the fixed variables and eliminating constraints in which all variables have been fixed, we obtain an instance of C-PIT in which certain blocks *must* be extracted before (or after) some prescribed time period. Call this constrained problem C-PIT[D]. If D is defined small enough, C-PIT[D] becomes tractable.

Given x^k, we consider three different ways of generating a set D from an empty set:

- Given a random block scheduled for extraction $a \in \mathcal{B}$, add a to D, and add to D blocks in $\mathcal{B}^-(a)$ connected to D up to a certain size. If the set D is too small choose another block a and repeat.
- Given a random block scheduled for extraction $a \in \mathcal{B}$, add a to D, and add to D blocks in $\mathcal{B}^+(a)$ connected to D up to a certain size. If the set D is too small choose another block a and repeat.
- Given a random block scheduled for extraction $a \in \mathcal{B}$ in time t, recursively include in D contiguous blocks that are scheduled for extraction in periods $[t-1, t+1]$, up to a certain size. If D is too small, add contiguous blocks scheduled for extraction in periods $[t-2, t+2]$.

In order to speed up the algorithm we discard neighborhoods that are not promising. For

this, let x^* be the optimal solution of R-PIT-1, and define,

$$\Delta = \left(\sum_{b \in \mathcal{B}} \sum_{t=1}^{T} p_{b,t}(x^*_{b,t} - x^*_{b,t-1}) \right) - \left(\sum_{b \in \mathcal{B}} \sum_{t=1}^{T} p_{b,t}(x^k_{b,t} - x^k_{b,t-1}) \right).$$

Before exploring neighborhood D, compute the value

$$\delta = \left(\sum_{b \in D} \sum_{t=1}^{T} p_{b,t}(x^*_{b,t} - x^*_{b,t-1}) \right) - \left(\sum_{b \in D} \sum_{t=1}^{T} p_{b,t}(x^k_{b,t} - x^k_{b,t-1}) \right).$$

If $\delta < 0.02\Delta$ discard neighborhood D and generate another.

Throughout the local-search we explore different neighborhoods in order to avoid getting stuck in local optima.

3   Implementation Details and Results

All of our algorithms were developed in the C programming language, and all of our computational tests where carried out in a Linux 2.6.9 machine with 32GB of RAM, and two Quad-Core Intel Xeon E5420 processors.

Our computational tests have two goals in mind. Our first goal is to compare the performance of the Critical Multiplier algorithm with CPLEX LP algorithms. Our second goal is to assess the quality of the solutions obtained by using our proposed heuristics, and the time required to obtain them. We do not analyze our solutions from a mine engineering standpoint.

In order to present our results we adopt the following notation. We refer to the Greedy, Gershon and Expected TopoSort heuristics as **GrTS**, **GeTS**, and **ExTS**, respectively. Whenever we had to solve an LP relaxation using CPLEX, we ran the Primal Simplex algorithm, the Dual Simplex algorithm, and the Barrier algorithm, and consider the time of the fastest one. We call this time **CPXbest**. We refer to the Critical Multiplier algorithm as **CMA**. We refer to the Local-Search algorithm as **LS 15m**, **LS 1h**, **LS 4h**and **LS 8h**, depending on if we let it run for 15 minutes, 1 hour, 4 hours or 8 hours. When comparing objective function values, we always present numbers divided by the upper bound obtained with **CMA**. This allows us to assess the proximity of solutions to the optimal value. For example, if a solution objective value is presented as 0.98 it can be inferred that it is within 2% of optimality. Note however, that this is merely a bound on how distant it is to the optimal. Despite being presented as 0.98 it could still be an optimal solution if the LP relaxation is strictly greater than the value of the optimal integral solution.

Before explaining the results of our experiments, we briefly describe some important details concerning the implementation of our algorithms:

CMA. To solve maximum-flow sub-problems, we implemented the highest-label push-relabel algorithm, as described in Ahuja et al. [1], and implemented both global re-

labeling and gap relabeling heuristics as described by Cherkassky and Goldberg [7]. In order to determine all of the critical multipliers we used binary search rather than sensitivity analysis. We always started by identifying the smallest critical multiplier first, then the second smallest, and so on. When determining the k-th smallest critical multiplier we made use of the binary-search iterations made when identifying previous multipliers. We also made use of the fact that the pit associated to the k-th multiplier is contained in the pit corresponding to the $(k-1)$-th multiplier.

LS 15m, LS 1h, LS 4h, LS 8h. All neighborhoods were constructed randomly and had size at most 3250 blocks, regardless of the instance. We used the CPLEX 11.2 IP solver with default settings to solve the C-PIT[D] problems, except for a 200 seconds time-limit. If the 200 seconds expired before obtaining a feasible solution, we aborted the search of this neighborhood and moved on to another. We provided CPLEX with the value of the incumbent solution as a bound for pruning and early termination. We also stop the B&B process as soon as a feasible solution with strictly better objective value than the provided incumbent solution is found.

To test our methodology, we consider four data sets:

Marvin is a fictitious copper gold ore body, included in the Whittle Four-X mine planning software, that has 53,668 blocks and 606,403 precedence constraints.

AmericaMine is a hard rock polymetallic mine. It has 19,320 blocks and 88,618 precedence constraints.

AsiaMine is a polymetallic ore body with a pipe shape. The block-model that we use has 772,800 blocks and 49,507,796 precedence constraints.

Andina is a copper molybdenum ore body in central Chile. The block-model that we use has 4,320,480 blocks and 81,973,942 precedence constraints.

All four mines consider a time horizon of 15 years and a discount rate of 10%. All mines consider two distinct capacity constraints per time period (on total tonnage extracted and processed each year), and precedence constraints with all blocks immediately above and within 45 degrees.

3.1 Single capacity-constraint computations

We first study instances of C-PIT-1. For this, we truncate our original data sets by ignoring extraction capacity constraint and only keeping the processing capacity constraint.

In Table 1 we present the running times of **CPXbest**, **CMA** and the Topological Sorting heuristics on each of our data set instances. It can be seen that in the larger instances **CPXbest** is simply unable to solve the problem. On the other hand, it can be seen that **CMA** manages to solve all LP relaxations to optimality in minutes. It can be seen that **GrTS** is very fast and that the additional amount of time required to run the **ExTS** after having obtained the optimal LP relaxation solution is negligible, even on the Andina instance. On the other hand, it can be observed that the running time of **GeTS** in our implementation is very large (4 days). This is due to the fact that weight of all inverted

cones below each block must be computed before topologically sorting the blocks.

Instance	CMA	CPXbest	GrTS	GeTS	ExTS
Marvin	12 s	1h 3m	< 1s	25s	< 1s
AmericaMine	4 s	19m 26s	< 1s	11s	< 1s
AsiaMine	2m 36s	10d+	< 1s	3h 33m	< 1s
Andina	1h 44m	N/A	< 1s	4d 17h	< 1s

Table 1: Running times of **CPXbest**, **CMA** and **ExTS** on instances of C-PIT-1. The time reported for **ExTS** does not include the time required to obtain the LP relaxation value. In AsiaMine, **CPXbest** was unable to find the optimal LP solution in 10 days. In Andina it was impossible to formulate the problem due to its large size.

In Table 2 we present the objective function values obtained using the TopoSort heuristics and the Local-Search heuristics. It can be seen that values obtained by **GrTS** and **GeTS** are very poor in some of the instances, whereas the values obtained by **ExTS** are very good (all within 6% of optimality). The table also shows that the Local-Search heuristics do a very good job of improving the solution obtained by **ExTS**. In fact, after running the **ExTS** and **LS 1h** all instances were within $3 \sim 4\%$ of optimality.

Instance	GrTS	GeTS	ExTS	LS 15m	LS 1h	LS 4h	LS 8h
Marvin	0.856	0.867	0.957	0.959	0.968	0.983	0.988
AmericaMine	0.819	0.905	0.940	0.997	0.997	0.999	0.999
AsiaMine	0.750	0.861	0.986	0.986	0.991	0.992	0.993
Andina	0.486	0.524	0.977	0.978	0.980	0.982	0.983

Table 2: Objective function values obtained using the TopoSort heuristics and the Local-Search heuristics in C-PIT-1 instances. All values have been divided by the optimal LP relaxation value of the corresponding problem and thus are less than 1.0. The **LS 15m**, **LS 1h**, **LS 4h** and **LS 8h** heuristics all start from the solution found by the **ExTS** heuristic.

3.2 Two capacity-constraint computations

We next study instances of C-PIT-2. That is, we consider our original data sets with both capacity constraints. All runs of the **CPXbest** and TopoSort heuristics include both the processing constraint and the extraction constraint. Runs of the **CMA** algorithm that consider the total extraction capacity constraint will be identified as **CMA-Tot**. Runs of the **CMA** algorithm that consider the processing capacity constraint will be identified as **CMA-Proc**.

First, note that we cannot run algorithm **CMA** on instances with two capacity constraints per time period. Our best alternative is to run both **CMA-Tot** and **CMA-Proc**. Since each of these problems is a relaxation of the problem we wish to solve, they each define a valid upper bound. Thus, we keep the smallest of the two.

In Table 3 we present the objective function values obtained using the TopoSort heuristics and the Local-Search heuristics on the C-PIT-2 instances. It can be seen that the relative performance of these heuristics is pretty much the same as in the C-PIT-1 case (see Table 2). The most important thing to note is that the **ExTS** heuristic obtains very good solutions, even though the LP relaxation solution from which it started was obtained from an instance where only a single-capacity constraint was considered. The same can be said of the Local-Search heuristics. The results show that after running the **ExTS** and **LS 15m** heuristics all instances are within 5% of optimality.

Instance	GrTS	GeTS	ExTS	LS 15m	LS 1h	LS 4h	LS 8h
Marvin	0.682	0.897	0.965	0.965	0.967	0.971	0.977
AmericaMine	0.340	0.823	0.937	0.955	0.988	0.989	0.990
AsiaMine	0.138	0.840	0.972	0.972	0.972	0.972	0.979
Andina	0.487	0.509	0.953	0.954	0.955	0.959	0.960

Table 3: Objective function values obtained using the TopoSort heuristics and the Local-Search heuristics on the C-PIT-2 instances. All values have been divided by the optimal LP relaxation value of the corresponding problem and thus are less than 1.0. The **LS 15m**, **LS 1h**, **LS 4h** and **LS 8h** heuristics all start from the solution found by the **ExTS** heuristic. The **ExTS** heuristic starts from the solution with best value between that obtained by **CMA-Tot** and **CMA-Proc**.

3.3   Additional Observations

Some final observations are as follows:

- In order to compute an upper bound on the value of C-PIT-2, our algorithm uses the best bound obtained from solving the two instances of R-PIT-1 (one for each constraint). How different are the values of this best bound and that of R-PIT-2? We compared this on the two instances on which R-PIT-2 could be solved with CPLEX, and found that the difference was below $3.5 \times 10^{-4}\%$ in both instances.

- In Section 2.3 we presented a heuristic criteria for discarding neighborhoods which were not promising. This was very helpful, especially on the larger problem instances. In fact, in the two smaller instances, approximately half of the iterations was considered, but in AsiaMine and Andina this ratio decreases to 10% and 2% respectively. Of this remaining set, between 2%-10% of these neighborhoods found improvements. When turning off this feature we found that on all instances the number of neighborhoods considered greatly decreased. Also, the success rate (i.e. the number of neighborhoods in which improvements were found) decreased as well.

- As can be seen from the computations, the main factor that contributed to the success of our method is the tight Linear Programming relaxation bound.

4 Final Remarks

In this article we have shown that it is possible to successfully tackle real-sized instances of C-PIT formulation in practical time for one or two resource capacity constraints per time period. We expect that this methodology can lead to new production-scheduling systems where it is no longer necessary to schedule bench-phases as is done today. Rather, using the assignment of blocks in time of the C-PIT solution, one could proceed with production scheduling from this solution as though it had been obtained otherwise.

It is important to note, however, that this is still just a proof of concept. It is likely that the C-PIT solutions are such that blocks scheduled in a same time period are scattered throughout the mine. This might lead to schedules which are not readily operational from an engineering point of view. This problem, which also happens in more traditional mine planning methods, will likely be exacerbated by the fact that our minimal planning units are blocks rather than bench-phases.

A natural next step would consist in extending the Critical-Multiplier method to work explicitly with multiple side-constraints. Another natural extension would consist in replacing the fixed capacity-constraint model with one in which the capacity is variable. The second, larger jump will consist in extending this methodology to work with Johnson's formulation [21]. That is, extending it to consider multiple destinations for each block. In this regard, the recent work of Bienstock and Zuckerberg [3] is very promising, and could complement very well the methods we propose.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, algorithms, and applications.* Prentice Hall, 1993.

[2] J. Amaya, D. Espinoza, M. Goycoolea, E. Moreno, T. Prevost, and E. Rubio. A scalable approach to optimal block scheduling. In *Proceedings of APCOM 2009*, pages 567–575. The Canadian Institute of Mining, Metallurgy and Petroleum, 2009.

[3] D. Bienstock and M. Zuckerberg. A new algorithm for precedence constrained production scheduling. *Optimization Online*, 2009.

[4] N. Boland, I. Dumitrescu, G.Froyland, and A.M. Gleixner. Lp-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers and Operations Research*, 36:1064–1089, 2009.

[5] N. Boland, C. Fricke, and G.Froyland. A strengthened formulation for the open pit mine production scheduling problem. *Available online at Optimization Online*, 2007.

[6] L. Caccetta and S.P. Hill. An application of branch and cut to open pit mine scheduling. *Journal of global optimization*, 27:349–365, 2003.

[7] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

[8] K. Dagdelen. *Optimum multi-period open pit mine production scheduling*. PhD thesis, Colorado School of Mines, Golden, Colorado, 1985.

[9] K. Dagdelen. Open pit optimisation – strategies for improving economics of mining projects through mine planning. *Orebody Modelling and Strategic Mine Planning, Spectrum Series*, 14:125–128, 2007.

[10] K. Dagdelen and T.B. Johnson. Optimum open pit mine production scheduling by lagrangian parameterization. *19th APCOM Symposium of the society of mining engineers (AIME)*, 1986.

[11] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 3:399–417, 1963.

[12] C. Fricke. *Applications of Integer Programming in Open Pit Mining*. PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, April 2006.

[13] M. Gaupp. *Methods for improving the tractability of the block sequencing problem for an open pit mine*. PhD thesis, Division of Economics and Business, Colorado School of Mines, June 2008.

[14] M.E. Gershon. Optimal mine production scheduling: evaluation of large scale mathematical programming approaches. *International journal of mining engineering*, 1:315–329, 1983.

[15] M.E. Gershon. Heuristic approaches for mine planning and production scheduling. *Int. Journal of Mining and Geological Engineering*, 5:1–13, 1987.

[16] M.E. Gershon. An open-pit production scheduler: algorithm and implementation. *AIME Transactions*, 282:793–796, 1987.

[17] D. Hochbaum and A. Chen. Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. *Operations Research*, 48:894–914, 2000.

[18] W. Hustrulid and K. Kuchta, editors. *Open Pit Mine Planning and Design*. Taylor and Francis, London, UK, 2006.

[19] W. A. Hustrulid, M.K. Carter, and D.J.A. Van Zyl, editors. *Slope stability in surface mining*. Society for Mining, Metallurgy, and Exploration, Inc., Litleton, Colorado, 2000.

[20] E.H. Isaaks and R.M. Srivastava. *Applied geostatistics*. Oxford University Press, New York, USA, 1989.

[21] T.B. Johnson. *Optimum open pit mine production scheduling*. PhD thesis, Operations Research Department, Universiry of California, Berkeley, May 1968.

[22] T.B. Johnson. Optimum open-pit mine production scheduling. In A. Weiss, editor, *A decade of digital computing in the mining industry*, chapter 4. AIME, New York, 1969.

[23] A.G. Journel and C.J. Huidbregts. *Mining Geostatics*. Academic Press, San Diego, USA, 1978.

[24] B.M. King. *Optimal mine scheduling policies*. PhD thesis, Royal School of Mines, London University, UK, 2001.

[25] K. Lane. *The economic definition of ore: cutoff grade in theory and practice*. Mining Journal Books Limited, London, 1988.

[26] H. Lerchs and I.F. Grossmann. Optimum design of open-pit mines. *Transactions*, LXVIII:17–24, 1965.

[27] G. Matheron. Le paramétrage des contours optimaux. Technical Report 403, Centre de Géostatistiques, Fontainebleau, France, 1975.

[28] G. Matheron. Le paramétrage technique des réserves. Technical Report 453, Centre de Géostatistiques, Fontainebleau, France, 1975.

[29] A. Newman, E. Rubio, R. Caro, A. Weintraub, and K. Eurek. A review of operations research in mine planning. *Submitted*, 2009.

[30] M. Osanloo, J. Gholamnejad, and B. Karimi. Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, 22:3–35, 2008.

[31] S. Ramazan, K. Dagdelen, and T.B. Johnson. Fundamental tree algorithm in optimising production scheduling for open pit mine design. *Mining Technology (Trans. Inst. Min. Metall. A)*, 114:45–54, 2005.

[32] S. Ramazan and R. Dimitrakopolous. Recent applications of operations research and efficient mip formulations in open pit mining. *Society for mining, metallurgy and exploration meeting transactions*, 316:73–78, 2004.