OPERATIONS RESEARCH
Vol. 60, No. 3, May–June 2012, pp. 517–528
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.1120.1050

A New Algorithm for the Open-Pit Mine Production Scheduling Problem

Renaud Chicoisne, Daniel Espinoza
Department of Industrial Engineering, Universidad de Chile, 8370439 Santiago, Chile
{renaud.chicoisne@gmail.com, daespino@dii.uchile.cl}

Marcos Goycoolea
School of Business, Universidad Adolfo Ibañez, 7941169 Santiago, Chile, marcos.goycoolea@uai.cl

Eduardo Moreno
Faculty of Engineering and Sciences, Universidad Adolfo Ibañez, 7941169 Santiago, Chile, eduardo.moreno@uai.cl

Enrique Rubio
Department of Mining Engineering and Advanced Mining Technology Center, Universidad de Chile, 8370439 Santiago, Chile,
erubio@redcoglobal.com

For the purpose of production scheduling, open-pit mines are discretized into three-dimensional arrays known as block models. Production scheduling consists of deciding which blocks should be extracted, when they should be extracted, and what to do with the blocks once they are extracted. Blocks that are close to the surface should be extracted first, and capacity constraints limit the production in each time period. Since the 1960s, it has been known that this problem can be cast as an integer programming model. However, the large size of some 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. In this article we study a well-known integer programming formulation of the problem that we refer to as C-PIT. We propose a new decomposition method for solving the linear programming relaxation (LP) of C-PIT when there is a single capacity constraint per time period. This algorithm is based on exploiting the structure of the precedence-constrained knapsack problem and runs in $O(mn \log n)$ in which n is the number of blocks and m a function of the precedence relationships in the mine. Our computations show that we can solve, in minutes, the LP relaxation of real-sized mine-planning applications with up to five million blocks and 20 time periods. Combining this with a quick rounding algorithm based on topological sorting, we obtain integer feasible solutions to the more general problem where multiple capacity constraints per time period are considered. Our implementation obtains solutions within 6% of optimality in seconds. A second heuristic step, based on local search, allows 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 been able to tackle only instances with up to 150,000 blocks and 15 time periods.

Subject classifications: open-pit mining; optimization; mixed integer programming.
Area of review: Environment, Energy, and Natural Resources.
History: Received November 2009; revisions received March 2011, June 2011, October 2011; accepted November 2011.

1. Introduction

Open-pit mining consists of extracting commercially valuable ore from a mineral deposit by digging from the surface as opposed to tunneling underground. Hustrulid and Kuchta (2006) describe the open-pit mineral supply process in terms of three important phases: planning, implementation, and production. During the planning phase, mining engineers have the highest potential for influencing the production value. The goal of this phase is to prepare a feasibility report containing a tentative production schedule, that is, to construct a plan outlining which part of the orebody should be extracted, when it should be extracted, and how it should be extracted. The feasibility study of a mine is considered a bankable document required by investors.

The planning phase of a mine typically is 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, we briefly describe it below.

1. *Block model determination*. The first step consists of 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 orebody 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 1978, Isaaks and Srivastava 1989). Based on this information, an estimated extraction profit for each block in the model

is computed. Profit depends on cost, and cost depends on what is done with the block after extraction, that is, if it is sent to a mill, a waste dump, or a stockpile. Block models with such profit attributes typically are referred to as *economic block models*. Formally, we will describe a block model in terms of a set of blocks $\mathscr{B}$ and a set of destinations $\mathscr{D}$. For each block $b \in \mathscr{B}$ we assume a profit of p_{bd} if block b is sent to destination d. Profit computations are done in present value and are done a priori; thus, they do not take into account the time at which the block is actually extracted.

2. *Final pit contour delineation*. This step consists of delimiting the subregion of the mine in which extraction will take place, or equivalently, the part of the whole inventory that will be considered for mining. This subdivision is known as the *final pit contour* or *ultimate pit limit*.

Before any block can be extracted, all blocks immediately above and at certain angles must also be removed (see Hustrulid et al. 2000). These angles are known as wall-slope requirement angles. To determine the ultimate pit limit, it is first necessary to determine these. Slope angle requirements depend on the structural composition of the rocks and vary depending on the location and depth of each block.

Once wall-slope angle requirements are determined, it is possible to define precedence relationships between blocks. These can be represented in terms of a digraph $G = (\mathscr{B}, \mathscr{A})$ in which $(a, b) \in \mathscr{A}$ means that block a must be extracted before block b. We may assume that G contains only immediate precedence relationships. That is, if $a, b, c \in \mathscr{B}$ are such that a must be extracted before b, and b must be extracted before c, then $(a, c) \notin \mathscr{A}$.

To compute the ultimate pit limit, let $p_b = \max_{d\in\mathscr{D}}\{p_{bd}\}$ represent the best possible profit that can be obtained from a block (also known as the estimated undiscounted profit) and solve the following problem:

$$UPL(u) = \max \sum_{b\in\mathscr{B}} u_b x_b, \tag{1a}$$

$$\text{s.t. } x_b \leqslant x_a \quad \forall\,(a, b) \in \mathscr{A}, \tag{1b}$$

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

In this problem, variable x_b takes a value one if and only if block b is to be included in the ultimate pit. Under simple assumptions, it is possible to prove that the ultimate pit limit contains the sets of all blocks that should be included in an optimal production schedule (see Caccetta and Hill 2003). The ultimate pit limit problem has received much attention in the literature. Lerchs and Grossmann (1965) first observed that it is equivalent to finding the maximum closure of a graph and proposed a customized algorithm. For modern algorithms and a comprehensive survey, including complexity analysis and computations, see Hochbaum and Chen (2000) and Chandran and Hochbaum (2009).

3. *Production scheduling*. This consists of deciding *which* blocks should be extracted, *when* they should be extracted, and *how* extracted blocks should be treated (Dagdelen 2007). This step assumes an a priori discretization of time into periods and an a priori definition of capacity in each time period. Capacity can be defined in terms of milling, transportation, and processing capabilities.

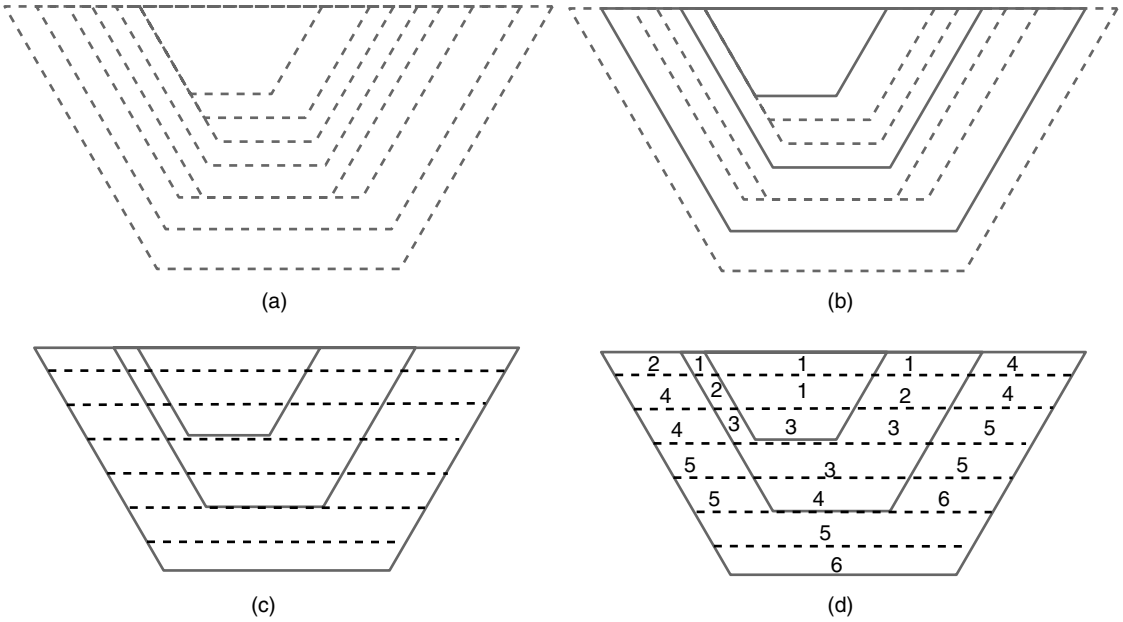
To determine a production schedule, it is common practice to proceed as follows.

First, determine a set of pushbacks (or phases). For this, a decreasing sequence of cost vectors $\pi^1 > \pi^2 > \cdots > \pi^k$ is determined. Then, for $i = 1, \ldots, k$, each of the problems $UPL(\pi^i)$ is solved to optimality to obtain solution x^i. It is well known (Lerchs and Grossmann 1965, Matheron 1975a, b) that $\pi^1 > \pi^2 > \cdots > \pi^k$ implies $x^1 \leqslant x^2 \leqslant \cdots \leqslant x^k$. That is, the solutions $x^1, \ldots, x^k$ correspond to a sequence of pits $P^1, \ldots, P^k$, such that $P^1 \subseteq P^2 \subseteq \ldots, P^k$. For details see Whittle (1999, Appendix E). Pushbacks are defined by taking the difference of consecutive pits. Formally, $F^1 = P^1$, $F^2 = P^2 - P^1, \ldots, F^k = P^k - P^{k-1}$. Ideally, the vectors π^i should be defined in such a way that the pushbacks F^i all have a similar size (in terms of the number of blocks). Unfortunately, it is not always possible to do this (this is known as the *gapping problem*), so in some cases heuristics must be used to separate pushbacks.

After the pushbacks are defined, these are further subdivided into groups of blocks at the same vertical level (or *bench*). These subdivisions are known as *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 capacity limitations. The whole process is illustrated in Figure 1. This process can be repeated when other considerations—such as minimum working space for shovels, trucks, and blending requirements—are not satisfied. To schedule the bench-phases, a number of proprietary algorithms exist. Perhaps the best known algorithm for doing this is Milawa, by Gemcom Software (2011). 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 typically is done by deciding a cutoff grade that discriminates between an ore and waste block (see Lane 1988 and King 2001). After the detailed plan is made, pushbacks are smoothed out to be more compliant with geometric operational requirements. Finally, haul-roads and ramps are designed in order to more accurately assess set-up and transportation costs.

Mine planning, as described in the three steps listed above, suffers from a number of important limitations. For example, processing capacity and net-present value are taken into account only after the bench-phases are computed, and multiple possible processing options for each block are considered only after blocks have been scheduled in time. Additionally, block destination is decided based on cutoff grade criteria rather than by considering capacities, time, and how other blocks are processed. Despite

Figure 1. A production plan scheduled by pushbacks.



Notes. 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.

these limitations, most commercial software packages use this method or variants thereof. Some important examples include Whittle (by Gemcom Software 2011), NPVscheduler (by DataMine Software 2011), Vulcan (by Maptek Software 2011), Comet (by Strategy Optimisation Systems 2011), among others. Moreover, most of these packages are not fully automated and require many manual computations in steps such as pushback generation.

2. Literature Review

To our knowledge, the first attempt to address the open-pit mine production scheduling problem in a holistic and exact optimization model was that of Johnson (1968, 1969). Johnson defines [0-1] variables describing the amount and type of material mined from each block, as well as the method and time of extraction. Johnson considers precedence constraints and a number of different capacity constraints, including (i) hours available per type of equipment; (ii) waste handling capacity; (iii) stockpiling capacity; (iv) capacity of plants (such as concentrators, refineries, and maintenance facilities); (v) blasting man-hours; (vi) lower and upper bounds on mining volumes due to legal constraints, plant-feed requirements, and marketability; (vii) volumetric ratio constraints for concentrator inputs; and others. A generalized form of Johnson's model is presented in Equations (2a)–(2g). In this model, variable x_{bdt} represents the fraction of block b sent to destination d in time period t or before, p_{bdt} represents the profit obtained per unit of block b sent to destination d in period t, a_{bdtr} represents the amount of resource r consumed in time period t per unit of block b sent to destination d, and c_{rt} represents the amount of resource r available in time period t.

$$\max \sum_{b\in\mathcal{B}}\sum_{d=1}^{D}\sum_{t=1}^{T} p_{bdt}(x_{bdt}-x_{b,d,t-1}), \tag{2a}$$

$$\text{s.t.} \sum_{d=1}^{D}\sum_{b\in\mathcal{B}} a_{bdtr}(x_{bdt}-x_{b,d,t-1}) \leqslant c_{rt} \quad \forall\, r\in R,\ \forall\, t=1,\dots,T \tag{2b}$$

$$\sum_{t=1}^{T}\sum_{d=1}^{D}(x_{bdt}-x_{b,d,t-1}) \leqslant 1 \quad \forall\, b\in\mathcal{B} \tag{2c}$$

$$x_{bdt} \leqslant x_{adt} \quad \forall\,(a,b)\in\mathcal{A},\ \forall\, t=1,\dots,T,\ \forall\, d\in D \tag{2d}$$

$$x_{b,d,t-1} \leqslant x_{bdt} \quad \forall\, b\in\mathcal{B},\ \forall\, t=1,\dots,T,\ \forall\, d\in D \tag{2e}$$

$$x_{bd0}=0 \quad \forall\, b\in\mathcal{B},\ \forall\, d\in D, \tag{2f}$$

$$0\leqslant x_{bdt}\leqslant 1 \quad \forall\, b\in\mathcal{B},\ \forall\, d\in D. \tag{2g}$$

Johnson (1968) first observed that in practical applications, the number of constraints of type (2b) is small relative the other constraints. Furthermore, he observed that if these constraints are relaxed, the problem reduces to solving T disjoint maximum closure problems. Because of this reduction, he proposed a Dantzig-Wolfe decomposition for solving problem. Dagdelen (1985) and Dagdelen and Johnson (1986) later developed a Lagrangian relaxation procedure and subgradient methods to tackle this problem. However, because most real-world problems are very large

(in terms of the number of blocks, destinations, and capacity constraints), most articles in the academic literature that follows consider some kind of simplified version. Equations (3a)–(3f) presents a simplified version of Johnson's formulation (C-PIT) in which the destination of each block is decided a priori, as in traditional economic block model constructions. C-PIT also differs from the more general model in that (i) it is assumed that $a \geqslant 0$ and $c_{rt} \geqslant 0$ for all r, t, and (ii) there exists $\eta > 0$ such that $p_{bt} = p_b/((1+\eta)^t)$ for all b and t. When we consider C-PIT formulations with only a single resource (i.e., when $R = 1$), we will denote c_{1t} as c_t.

$$\max \quad \sum_{b\in\mathscr{B}}\sum_{t=1}^{T} p_{bt}(x_{bt} - x_{b,t-1}), \tag{3a}$$

$$\text{s.t.} \quad \sum_{b\in\mathscr{B}} a_{rb}(x_{bt} - x_{b,t-1}) \leqslant c_{rt} \qquad \forall\, t \in 1\ldots T, \forall\, r \in 1\ldots R, \tag{3b}$$

$$x_{bt} \leqslant x_{at} \quad \forall\,(a,b) \in \mathscr{A},\ \forall\, t \in 1\ldots T, \tag{3c}$$

$$x_{bt} \leqslant x_{b,t+1} \quad \forall\, b \in \mathscr{B},\ \forall\, t \in 1\ldots T-1, \tag{3d}$$

$$x_{bt} \in \{0,1\} \quad \forall\, b \in \mathscr{B},\ \forall\, t \in 1\ldots T, \tag{3e}$$

$$x_{b0} = 0 \quad \forall\, b \in \mathscr{B}. \tag{3f}$$

Even though the C-PIT model is much simplified with respect to Johnson's formulation, it is still of much value to current planning methodologies. Instead of scheduling bench-phases, one can use C-PIT to schedule blocks directly. This allows the explicit consideration of net present value and capacity when constructing the schedule.

Caccetta and Hill (2003) use the solution of the ultimate pit limit problem for preprocessing and propose a branch-and-cut framework for solving the problem. Ramazan and Dimitrakopoulos (2004) relax the integrality of variables with negative objective function value. Boland et al. (2010) and Fricke (2006) use cutting planes derived from the precedence constrained knapsack problem. Gaupp (2008) computes bounds on the earliest (and latest) possible times a block can be extracted to pre-process before applying Lagrangian relaxation methods. Gershon (1987a, b), and Fricke (2006) 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 (2003), who claim to solve instances with up to 250,000 blocks. Unfortunately, citing commercial interests, the authors do not describe any replicable algorithm or methodology.

Alternative methods to using C-PIT have been proposed for simplifying Johnson's model. Ramazan et al. (2005) 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 (1983) allows the partial extraction of blocks in the bottom of the pit. More recently, Boland et al. (2009) 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 linear programming (LP) relaxation of this formulation.

While preparing this document it has come to our attention that Bienstock and Zuckerberg (2010) have developed a new decomposition method suited to solving the C-PIT linear programming relaxation with an arbitrary number of side constraints per time period. The computational results they present are very promising.

For a more detailed survey of exact optimization approaches for Johnson's model and other variants, see Osanloo et al. (2008). For a recent survey of operations research applications in mine planning, see Newman et al. (2010). For a detailed treatment of open-pit mine planning, see Hustrulid and Kuchta (2006).

In this paper, we present a new methodology for solving instances of C-PIT with a single resource constraint per time period (i.e., when $R = 1$). This methodology is described in §3. In §4 we describe an implementation of the algorithm and the results obtained from testing it on several real mine planning instances.

3. Methodology

The proposed methodology consists of three steps: First, we solve the linear programming (LP) relaxation of C-PIT with a new decomposition method called the *critical multiplier algorithm*. This algorithm requires that the C-PIT formulation have a single resource constraint per time period. A description of the critical multiplier algorithm is presented in §3.1. Second, we apply a rounding heuristic to the fractional solution obtained from the critical multiplier algorithm. This heuristic is based on topological sorting and is described in §3.2. Third, in §3.3, we apply a local-search heuristic to improve the quality of the solutions obtained by the rounding heuristic. Throughout this section, we assume that all scalars and vectors are rational numbers.

3.1. The Critical Multiplier Algorithm

In this section, we describe the critical multiplier algorithm for solving the linear programming relaxation of C-PIT when there is only a single resource constraint per time period.

We begin by studying the problem $CP(\kappa)$, which can be considered as a single time period version of C-PIT with only a single resource constraint.

$$CP(\kappa) = \max \quad px,$$
$$\text{s.t.} \quad ax \leqslant \kappa$$

$$x_i \leqslant x_j \quad \forall (i,j) \in \mathscr{A}$$
$$0 \leqslant x_i \leqslant 1 \quad \forall i \in \mathscr{B}.$$

Let $z(\lambda)$ correspond to the optimal objective function value of problem $UPL(p-\lambda a)$, as defined in (1a)–(1c). It is well known (Lerchs and Grossmann 1965, Matheron 1975a, b) that

1. $z(\lambda)$ is a convex piecewise linear function with a finite number of break-points

$$\lambda^1 > \lambda^2 > \cdots > \lambda^k;$$

2. for each $i \in 1,\ldots,k$ there exists x^i such that it is an optimal solution of $UPL(p-\lambda a)$ for all $\lambda \in [\lambda^{i+1}, \lambda^i]$ and such that

$$x^1 \leqslant x^2 \leqslant \cdots \leqslant x^k.$$

Proposition 3.1. *Consider a scalar b. For each $i \in 1,\ldots,k$ let $b^i = ax^i$, and define*

$$u = \min\{i \in 1\ldots k\colon b^i \geqslant b\},$$
$$l = \max\{i \in 1\ldots k\colon b^i \leqslant b\}.$$

If $u = l$, then solution $\bar{x} = x^u = x^l$ is optimal for $CP(b)$. Otherwise, define

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

Solution $\bar{x} = \alpha x^l + (1-\alpha)x^u$ is optimal for $CP(b)$.

Proof. Let $Nx \leqslant 0$ represent the set of precedence constraints, and let $\mathbb{1}$ represent the vector of all ones. The dual problems associated to $UPL(p-\lambda a)$ and $CP(b)$ are as follows:

$$DUPL(p-\lambda a) = \begin{aligned} \min\ & \mathbb{1}y_B \\ \text{s.t.}\ & y_B + N^T y_A \geqslant (p^T - \lambda a^T) \\ & y \geqslant 0, \end{aligned}$$

$$DCP(b) = \begin{aligned} \min\ & \mathbb{1}y_B + \mu b \\ \text{s.t.}\ & y_B + N^T y_A + \mu a^T \geqslant p^T \\ & y, \mu \geqslant 0, \end{aligned}$$

where y_A, y_B, and μ are dual variables associated with constraint sets $Nx \leqslant 0$, $x \leqslant \mathbb{1}$, and $ax \leqslant b$, respectively.

First, note that $\bar{x}$ is feasible for $CP(b)$.

Second, let (y_A^*, y_B^*) be an optimal solution of $DUPL(p-\lambda^u a)$; then, it is easy to see that $(y_A^*, y_B^*, \lambda^u)$ is feasible for $DCP(b)$ and has objective function value $\mathbb{1}y_B^* + \lambda^u b$. Because both x^u and x^l are optimal for $UPL(p-\lambda^u a)$, it follows that

$$\mathbb{1}y_B^* = (p-\lambda^u a)x^u, \tag{4}$$
$$\mathbb{1}y_B^* = (p-\lambda^u a)x^l. \tag{5}$$

Taking $(1-\alpha)\cdot(4) + \alpha\cdot(5)$, we obtain

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

Thus,

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

By strong duality, we conclude that $\bar{x}$ is optimal for $CP(b)$. □

A similar version of Proposition 3.1 was proved in Lerchs and Grossmann (1965), but with a focus on parametric analysis of ultimate pit limit solutions.

Proposition 3.2. *Let $\kappa_1 > \kappa_2 > 0$ and let $\bar{x}^1$ and $\bar{x}^2$ be corresponding optimal solutions of $CP(\kappa_1)$ and $CP(\kappa_2)$ as characterized in Proposition* 3.1. *Then, $\bar{x}^1 \geqslant \bar{x}^2$.*

Proof. From Proposition 3.1 we know for some i, j and α^1, α^2 that

$$\bar{x}^1 = \alpha^1 x^i + (1-\alpha^1)x^{i+1},$$
$$\bar{x}^2 = \alpha^2 x^j + (1-\alpha^2)x^{j+1}.$$

First, suppose $i > j$. We have $x^j \leqslant x^{j+1} \leqslant x^i \leqslant x^{i+1}$. Hence, $\bar{x}^1 \geqslant \bar{x}^2$. Second, suppose $i = j$. As in Proposition 3.1, 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

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

Thus, $\kappa_1 > \kappa_2$ implies $\alpha_1 < \alpha_2$. Because $x^i \leqslant x^{i+1}$, it follows that $\bar{x}^1 \geqslant \bar{x}^2$. □

We now present the main result of this section: an algorithm for solving the linear programming relaxation of C-PIT, as defined in Equations (3a)–(3f), when there is only a single resource constraint per time period (i.e., $R = 1$). A summary of the solution methodology is presented in Algorithm 1.

Theorem 3.1. *Consider an instance of C-PIT with a single resource constraint per time period. For each $t \in 1,\ldots,T$ let $U_t = \sum_{s=1}^{t} c_t$, and let $\bar{x}^t$ represent the optimal solution of $CP(U_t)$, obtained as in Proposition* 3.1. *Then, $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^T)$ is an optimal solution to the linear programming relaxation of C-PIT. Furthermore, $\bar{x}$ can be computed in $O(mn\log n)$, where n is the number of blocks and m the number of precedences.*

PROOF. First, note that $\bar{x}$ is feasible for the following problem:

$$AUX = \max \sum_{b\in\mathcal{B}}\sum_{t=1}^{T} p_{bt}(x_{bt} - x_{b,t-1}), \tag{6a}$$

$$\text{s.t. } x_{it} \leqslant x_{jt} \quad \forall\, t = 1,\dots,T \quad \forall\,(i,j)\in\mathcal{A} \tag{6b}$$

$$x_{bt} \leqslant x_{b,t+1}, \quad \forall\, b\in\mathcal{B},\ \forall\, t = 1,\dots,T-1, \tag{6c}$$

$$\sum_{b\in\mathcal{B}} a_b x_{bt} \leqslant U_t \quad \forall\, t = 1,\dots,T, \tag{6d}$$

$$0 \leqslant x_{bt} \leqslant 1 \quad \forall\, b\in\mathcal{B},\ \forall\, t = 1,\dots,T, \tag{6e}$$

$$x_{b0} = 0 \quad \forall\, b\in\mathcal{B}. \tag{6f}$$

In fact, because $\bar{x}^t$ is feasible for $CP(U_t)$ for all t, it follows that constraints (6b), (6d), (6e), and (6f) are all satisfied. On the other hand, because $U_t \leqslant U_{t+1}$ for all $t < T$, from Proposition 3.2, it follows that (6c) is satisfied.

Second, note that $\bar{x}$ is an optimal solution of AUX. In fact, let y be any feasible solution for AUX. We will write $y = (y^1, y^2, \dots, y^T)$, where y_i^t corresponds to y_{it}. In this way, we have that y^t is a feasible solution for $CP(U_t)$ for $t = 1,\dots,T$. For any x feasible for AUX, let $z(x)$ be the corresponding objective function value. It can be seen that

$$z(x) = \sum_{b\in\mathcal{B}}\sum_{t=1}^{T} p_{bt}(x_{bt} - x_{b,t-1}) = \sum_{t=1}^{T-1} \gamma_t p x^t,$$

where $\gamma_t = (1 - 1/(1+\eta))(1/(1+\eta)^t)$ for $t < T$ and $\gamma_T = 1/(1+\eta)^T$. It thus follows that

$$z(\bar{x}) - z(y) = \sum_{t=1}^{T} \gamma_t (p\bar{x}^t - py^t).$$

However, for $t = 1\dots T$, both y^t and $\bar{x}^t$ are feasible for $CP(U_t)$. Because $\bar{x}^t$ is optimal for $CP(U_t)$ we have $(p\bar{x}^t - py^t) \geqslant 0$ for all t, and hence $z(\bar{x}) - z(y) \geqslant 0$.

Finally, note that because $\bar{x}$ is an optimal solution of AUX, it must be feasible for C-PIT. Because $\bar{x}$ is optimal for AUX, it must hold that if $\bar{x}^t \neq \bar{x}^{t+1}$ for some $t < T$, then $a\bar{x}^t = U_t$. However, this implies $\bar{x}$ is feasible for C-PIT.

Because AUX is a relaxation of C-PIT, we conclude that $\bar{x}$ must be optimal for C-PIT.

The complexity $O(mn\log n)$ is the complexity of Hochbaum's parametric pseudoflow algorithm (Hochbaum 2008). This algorithm computes all the break-point scalar values $\lambda^1,\dots,\lambda^k$ and the associated break-point solutions $x^1,\dots,x^k$ for problem $UPL(p - \lambda a)$. Once these break-points are obtained, problem $CP(U_t)$ can be solved to obtain $\bar{x}^t$ in $O(n)$ for each $t\in 1,\dots,T$. □

Algorithm 1 (The critical multiplier algorithm (CMA)).

Inputs: A set of blocks $\mathcal{B}$, where $n = |\mathcal{B}|$, precedence relationships $\mathcal{A}$, a number of time periods T, a positive (scalar) capacity c_t for each $t = 1,\dots,t$, a profit vector $p\in\mathbb{R}^n$, and a nonnegative constraint vector $a\in\mathbb{R}^n$.

Compute break-point values $\lambda^1 > \lambda^2 > \cdots > \lambda^k$ and corresponding break-point solutions $x^1 \leqslant x^2 \leqslant \cdots \leqslant x^k$ of piecewise linear function $z(\lambda)$, where $z(\lambda)$ is the optimal value of $UPL(p - \lambda a)$, as defined in Equations (1a)–(1c). For this, use the parametric version of Chandran and Hochbaum (2009), or any other sensitivity analysis method.
for $t = 1$ to T **do**
  $U_t \leftarrow \sum_{s=1}^{t} c_t$.
  $u \leftarrow \min\{i\in 1\dots k\colon a\cdot x^i \geqslant U_t\}$.
  $l \leftarrow \max\{i\in 1\dots k\colon a\cdot x^i \leqslant U_t\}$.
  if $u = l$ **then**
    $\bar{x}^t \leftarrow x^u$
  else
    $\alpha = \dfrac{a\cdot x^u - U_t}{a\cdot x^u - a\cdot x^l}$.
    $\bar{x}^t \leftarrow \alpha x^l + (1-\alpha)x^u$.
return $\bar{x} = (\bar{x}^1, \bar{x}^2, \dots, \bar{x}^T)$.

3.2. Obtaining a Starting Feasible Solution: The TopoSort Heuristic

Given a block model, we say that an ordering (or permutation) of blocks $\{b_1, b_2, \dots, b_n\}$ defines a *feasible extraction sequence* if each block appearing in the sequence is such that all its predecessors appear before it in the sequence. For example, consider the two-dimensional representation of a mine illustrated in Figure 2(a), and suppose that blocks must be extracted satisfying slope-angle constraints of 45 degrees. The ordering

$$\{b, a, c, g, d, h, e, i, m\},$$

illustrated in Figure 2(b), defines a feasible extraction sequence.

Feasible extraction sequences lead to a natural heuristic for C-PIT. Start with the first block in the sequence. Schedule this block for extraction as early as possible. That is, make sure to schedule the block later in time than all its predecessors, and do so in the first time period for which there are enough resources available. Move to the next block. Repeat until all the blocks are scheduled. Because a feasible extraction sequence is such that all predecessors of a block must come before it in the sequence, this heuristic is correct and leads to a feasible solution.

Consider again the example illustrated in Figure 2. Assume that there are only enough resources to extract two blocks per time period. The solution associated with ordering $\{b, a, c, g, d, h, e, i, m\}$ would consist of extracting $\{b, a\}$ in the first time period, $\{c, g\}$ in the second period, and so on. We call this type of heuristic a TopoSort heuristic for reasons that will soon be clear. The most critical part of a TopoSort heuristic consists of obtaining a

Figure 2. Two-dimensional representation of a mine (a); feasible extraction sequence example (b).

(a)

a	*b*	*c*	*d*	*e*
f	*g*	*h*	*i*	*j*
k	*l*	*m*	*n*	*o*

(b)

a (2)	*b* (1)	*c* (3)	*d* (5)	*e* (7)
f	*g* (4)	*h* (6)	*i* (8)	*j*
k	*l*	*m* (9)	*n*	*o*

Notes. In (a), a two-dimensional example where it is assumed that to extract any block, the blocks directly above, above left, and above right must be previously extracted. This implies that to remove block m, blocks g, h, and i must have been removed before, while we can remove block a in any time period. In (b), a feasible extraction sequence example: For the mine defined in (a), the order $b, a, c, \ldots, m$ is a feasible extraction sequence, in the sense that if we extract the blocks in this order we will not violate the precedence requirements.

good feasible extraction sequence. We next describe several methods by which such a sequence can be obtained. Before doing so, we introduce some notation and formalize the algorithm just described.

Recall the precedence graph $G = (\mathcal{B}, \mathcal{A})$ defined in the introduction. In this graph, nodes correspond to blocks, and arcs correspond to immediate precedence relationships. That is, $(a, b) \in \mathcal{A}$ if block a is an immediate predecessor of block b. Assume $|\mathcal{B}| = n$. Given $a, b \in \mathcal{B}$, we say that $a \to b$ if there is a directed path from a to b in G, or equivalently, if a is a predecessor of b.

Let $\mathcal{B}^-(b)$ denote the set of all blocks in $\mathcal{B}$ that are predecessors of b. We say that an ordering of the blocks $\{b_1, b_2, \ldots, b_n\}$ defines a *topological ordering* of $\mathcal{B}$ 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 $i, j \in 1, \ldots, n$. Note that a sequence is a topological ordering in $\mathcal{B}$ if and only if it defines a feasible extraction sequence in the mine.

Because arcs $\mathcal{A}$ correspond to geometric precedence relationships, G cannot contain any directed cycles. An important property of acyclic directed graphs is that they always admit topological orderings (see Cook et al. 1998). Thus, we can say that (1) the TopoSort heuristic simply reduces to finding a topological ordering of the blocks in the mine, and (2) such an ordering always exists.

Because a directed acyclic graph can have many different topological orderings, we need a mechanism by which to select good orderings. If we assume that every block b has an associated weight w_b, our heuristics will prefer those topological orderings in which blocks with bigger weights appear earlier in the sequence. This idea leads to the following definition.

Consider a topological ordering $\{b_1, \ldots, b_n\}$. We say that the ordering is weighted with respect to w if every pair of consecutive blocks b_i and b_{i+1} satisfies either (1) b_i is a predecessor of b_{i+1} (i.e., $b_i \to b_{i+1}$), or (2) $w_i < w_j$. Algorithm 2 shows how it is possible to obtain a topological sorting of the blocks in a mine. In this algorithm, given a directed graph $G(V, E)$ and $u \in V$, set $\delta^-(u)$ represents the arcs pointing to u.

Algorithm 2 (`TSort`(G, w), topologically sorting the nodes of a directed graph).

Inputs: An acyclic directed graph $G = (V, E)$, a node cost vector w.

1: Let $i \leftarrow 1$ and let $n \leftarrow |V|$.
2: **while** $i < n$ **do**
3: $u \leftarrow \arg\max\{w_u\colon \delta^-(u) = \varnothing\}$ (u is any min-weight node with no incoming arcs).
4: $G \leftarrow G \backslash \{u\}$ (remove u from G and all arcs outgoing from u).
5: $v_i \leftarrow u$ and $i \leftarrow i + 1$.
6: **return** A topological ordering $v_1, \ldots, v_n$ in G that is weighted with respect to w.

Given a weighted topological ordering, it is simple to schedule the blocks of a mine in such a way as to obtain a feasible solution to C-PIT. This can be seen in Algorithm 3. There is but one key issue remaining in our discussion: how to choose an appropriate set of weights for the blocks in the mine.

Algorithm 3 (`THeur`(G, w): the TopoSort heuristic).

Inputs: An acyclic directed graph $G = (\mathcal{B}, \mathcal{A})$ representing blocks and precedences, a weight w_b for each block $b \in \mathcal{B}$, a number of time periods T, a number of resources R, and for each block b, each time period t and each resource r, the amount of resource r required to extract block b (represented by q_{rb}),and the availability of resource r available in time t (represented by c_{rt}).

1: $t_b \leftarrow 1,\ \forall b \in \mathcal{B},\ i \leftarrow 1$.
2: $\{b_1, \ldots, b_n\} \leftarrow$ `TSort`(G, w).
3: **while** $i < n$ **do**
4: Let $u = b_i$ and let $i \leftarrow i + 1$.
5: $t_u \leftarrow \max\{t_v\colon (v, u) \in E\}$. (block u cannot be extracted before its predecessors).
6: $t_u \leftarrow \max\{t_u, \min\{t\colon c_{rt} \geqslant q_{ru}\ \forall r = 1, \ldots, R\}\}$. (extract in the earliest time period following t_u with sufficient resources).
7: **if** $t_u \leqslant T_{\max}$ **then**
8: $c_{rt_u} \leftarrow c_{rt_u} - q_{ru},\ \forall r = 1, \ldots, R$. (update available resources).
9: **return** A time of extraction t_b for each block $b \in \mathcal{B}$.

In fact, Algorithm 3 describes not just one, but rather an entire class of heuristic algorithms for C-PIT. Different assignments of weights lead to different feasible solutions of the problem.

A well-known TopoSort heuristic was first proposed by Gershon (1987a). Gershon's heuristic corresponds to the case in which the weight of each block is computed as

$$w(b) = \sum_{a \in \mathcal{B}^-(b)} p_a \quad \forall\, b \in \mathcal{B}. \tag{7}$$

Another natural variant can be defined by assigning weights

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

We henceforth call this the greedy heuristic.

We propose a different weight function that uses the linear programming relaxation solution of the C-PIT problem. For each $b \in B$ and $t \in 1, \ldots, T$, let $P_{b,t} = (x^*_{b,t} - x^*_{b,t-1})$. The fact that $0 \leqslant x_{b,0} \leqslant x_{b,1} \leqslant \ldots x_{b,T} \leqslant 1$ implies $0 \leqslant P_{b,t} \leqslant 1$ for $t = 1, \ldots, T$, and, $\sum_{t=1}^{T} P_{b,i} \leqslant 1$. We can interpret these two properties as the fact that the LP model is uncertain regarding when each block should be extracted, and that $P_{b,t}$ is the probability that block b will be extracted in time t. We can assume that blocks not extracted by time T will be extracted in time $T+1$. This suggests defining

$$E_b = \sum_{t=1}^{T} t(x^*_{b,t} - x^*_{b,t-1}) + (T+1)(1 - x^*_{b,T}),$$

that is, the expected time of extraction for block b. Note that if $(a, b) \in \mathcal{A}$, then $E_a \leqslant E_b$. Moreover, if x^* is the optimal (integer) solution of C-PIT, then E_b is either equal to the time of extraction of block b, or equal to $T+1$ if b is not extracted.

We define the expected-time (ET) TopoSort heuristic by defining weights

$$w_b = -E_b \quad \forall\, b \in \mathcal{B}.$$

As an example, the solution obtained by applying this heuristic to the instance depicted in Figure 2 is described in Table 1.

3.3. 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. (2009). 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, having strictly better objective function value than x^k, that coincides with x^k outside of D. Formally, this is equivalent to

$$x^{k+1}_{b,t} = x^k_{b,t} \quad \forall\, t = 1, \ldots, T,\ \forall\, b \notin D.$$

If no such improving solution exists, we define a new set of blocks D and try again. Given a solution x^k, obtaining x^{k+1} reduces to solving an instance of C-PIT in which we have a number of variables inversely proportional to the size of D. Moreover, fixing these variables does not change the structure of the problem. After substituting out the fixed variables and eliminating constraints in which all variables have been fixed, we obtain a smaller 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. See §4 for details regarding an appropriate size of D.

Given x^k, we consider three different ways of building the neighborhood set D:

- Method 1. Choose a random block $a \in \mathcal{B}$ that has been scheduled for extraction in the incumbent solution. Add a to D. If $|\mathcal{B}^-(a)| \geqslant d_{\max}$, build D so that it is a connected subset of $\mathcal{B}^-(a)$ having $d_{\max}$ blocks, and stop. Otherwise, add all of $\mathcal{B}^-(a)$ to D.
- Method 2. Choose a random block $a \in \mathcal{B}$ that has been scheduled for extraction in the incumbent solution. Add a to D. If $|\mathcal{B}^+(a)| \geqslant d_{\max}$, build D so that it is a connected subset of $\mathcal{B}^+(a)$ having $d_{\max}$ blocks, and stop. Otherwise, add all of $\mathcal{B}^+(a)$ to D.
- Method 3. Choose a random block $a \in \mathcal{B}$ that has been scheduled for extraction in the incumbent solution. Assume that t is the time in which a is extracted. Build D in such a way that: (i) a is contained in D; (ii) all the blocks in D are scheduled for extraction in times $t-1$, t, or $t+1$; and (iii) D does not contain more than $d_{\max}$ blocks.

Throughout the local-search algorithm, we explore different neighborhoods in order to avoid local optima. For details on how we defined the parameters such as $d_{\max}$, or how long we explored each neighborhood, see §4.

4. Implementation Details and Results

All our algorithms were developed in the C programming language, and all our computational tests were carried out on a Linux 2.6.9 machine with 32 GB 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.

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. When using CPLEX to solve a linear program, we ran the primal simplex algorithm, the dual simplex algorithm, and the barrier algorithm, and we considered the time of the fastest one. For each of these three runs, we used the default CPLEX settings. We call this time **CPXbest**. We refer to the critical multiplier algorithm as **CMA**. We refer to the

Table 1. Solution of the example in Figure 2 for $T = 5$, $c_t = 3\ \forall t$, $\eta = 5\%$, weights a_b and profits p_b.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p_b	1	1	1	−1	−1	−1	2	2	3	−1	−2	−2	5	−2	−2
a_b	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$x_{b,1}$	$\frac{6}{10}$	$\frac{6}{10}$	$\frac{6}{10}$	0	0	0	$\frac{6}{10}$	0	0	0	0	0	0	0	0
$x_{b,2}$	1	1	1	$\frac{1}{9}$	$\frac{1}{9}$	0	1	$\frac{1}{9}$	$\frac{1}{9}$	0	0	0	$\frac{1}{9}$	0	0
$x_{b,3}$	1	1	1	$\frac{4}{9}$	$\frac{4}{9}$	0	1	$\frac{4}{9}$	$\frac{4}{9}$	0	0	0	$\frac{4}{9}$	0	0
$x_{b,4}$	1	1	1	$\frac{7}{9}$	$\frac{7}{9}$	0	1	$\frac{7}{9}$	$\frac{7}{9}$	0	0	0	$\frac{7}{9}$	0	0
$x_{b,5}$	1	1	1	1	1	0	1	1	1	0	0	0	1	0	0
E_b	1.4	1.4	1.4	3.67	3.67	6	1.4	3.67	3.67	6	6	6	3.67	6	6
Order	1	2	3	5	6	10	4	7	8	11	12	13	9	14	15
t_b	1	1	1	2	3	—	2	3	4	—	—	—	5	—	—

Notes. We show in row E_b the expected time for each block, in row *Order* a topological order obtained by `TSort` using weights $w_b = -E_b$, and in row t_b the time of extraction of each block obtained by the TopoSort heuristic.

local-search algorithm as **LS 15 m**, **LS 1 h**, **LS 4 h**, and **LS 8 h**, 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 close the objective function value of the solution is to the optimal objective function value. 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 subproblems, we implemented the highest-label push-relabel algorithm, as described in Ahuja et al. (1993), and implemented both global relabeling and gap relabeling heuristics as described by Cherkassky and Goldberg (1997). To determine all the critical multipliers, we used binary search rather than Hochbaum's parametric algorithm (Hochbaum and Chen 2000). Although in theory this should be less efficient, in practice we found it worked fast enough because we are determining only T break-points. We always started by identifying the smallest critical multiplier first, then the second smallest, and so on. When determining the kth smallest critical multiplier, we made use of the previous binary-search iterations. We also used the fact that the pit associated with the kth multiplier is contained in the pit corresponding to the $(k-1)$th multiplier.

LS 15 m, **LS 1 h**, **LS 4 h**, **LS 8 h**. In each iteration of the local-search heuristics, we randomly chose among the three methods proposed in §3.3 in order to build a neighborhood. The probability of choosing each of the three neighborhoods was one third. In all our runs, we defined $d_{\max} =$ 3,250 blocks. To solve each instance of C-PIT$[D]$, we used the CPLEX 11.2 IP solver (with default settings), except for the fact that we imposed a 200-second time limit. If the 200 seconds expired before CPLEX was able to find a feasible solution, we aborted the search of the neighborhood and moved on to another. The selection of $d_{\max}$ was made to generate neighborhoods that could be solved to near-optimality in the 200-second time limit. In each iteration, we provided CPLEX with the value of the incumbent solution as a bound for pruning and early termination. We also stopped the B&B process as soon as a feasible solution with a strictly better objective value than that provided by the incumbent solution was found. These settings were found to work well after testing various different options.

To test our methodology, we consider four data sets.

Marvin is a fictitious copper and gold orebody, 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 orebody with a pipe shape. The block model we use has 772,800 blocks and 49,507,796 precedence constraints.

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

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

4.1. Single Resource Constraint Computations

We first study instances of C-PIT in which we consider only a single resource constraint per time period (i.e., when $R = 1$). For this, we truncate our original data sets by

Table 2. Running times of **CPXbest**, **CMA**, and **ExTS** on instances of C-PIT where only a single resource constraint per time period was considered.

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

Notes. 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 for us to formulate the problem because the problem size is beyond architectural limits of the computer (CPLEX Error 1012: problem size limits too large).

ignoring the extraction capacity constraint and keeping only the processing capacity constraint.

In Table 2, we present the running times of **CPXbest**, **CMA** and the topological sorting heuristics on each of our data set instances. In the larger instances, **CPXbest** is simply unable to solve the problem. On the other hand, **CMA** manages to solve all LP relaxations to optimality in minutes. **GrTS** is very fast and 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. Finally, the running time of **GeTS** in our implementation is very large (4 days). This is because computing the weight of each block using this method is very time-consuming.

In Table 3, we present the objective function values obtained using the TopoSort heuristics and the local-search heuristics. The 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 1 h**, all instances were within 3%–4% of optimality.

4.2. Two Resource Constraint Computations

We next study instances of C-PIT in which we consider two resource constraints per time period (i.e., $R = 2$). That is, we consider our original data sets with both capacity constraints.

First, recall that we cannot run algorithm **CMA** on instances with two capacity constraints per time period. However, given an instance with two constraints per time period, if we relax one of these, we obtain a problem whose linear programming relaxation we can solve with **CMA**. This linear relaxation is still an upper bound of the optimal solution we are seeking, and moreover, can be used to start the **ExTS** heuristic. In contrast to the **CMA** algorithm, the **THeur** algorithm runs correctly when there are two resource constraints per time period, and is guaranteed to yield a solution valid for the original problem (i.e., satisfying both constraints). This discussion suggests proceeding as follows: run **CMA** twice, once for each resource constraint. Starting from each of the fractional solutions obtained, run the **ExTS** heuristic. Use the fractional solution having the smallest objective function value to compute an upper bound. Use the integer-feasible solution having the highest value to compute a lower bound. This procedure is detailed in Algorithm 4. It is easy to see that this procedure could be extended to any number of resource constraints.

Algorithm 4 (**ExTS** heuristic for two resource constraints ($R = 2$)).

1: Solve **CMA** using only the extraction capacity constraint. Let $\hat{x}^{\text{Ext}}$ be its solution with objective value $\hat{z}^{\text{Ext}}$.
2: Compute expected times E_b^{Ext} for each block $b \in B$ using $\hat{x}^{\text{Ext}}$.
3: Use `THeur` with weights $w_b = -E_b^{\text{Ext}}$ to obtain a feasible solution $\bar{x}^{\text{Ext}}$ with objective value $\bar{z}^{\text{Ext}}$.
4: Solve **CMA** using only processing capacity constraint. Let $\hat{x}^{\text{Proc}}$ be its solution with objective value $\hat{z}^{\text{Proc}}$.
5: Compute expected times E_b^{Proc} for each block $b \in B$ using $\hat{x}^{\text{Proc}}$.
6: Use `THeur` with weights $w_b = -E_b^{\text{Proc}}$ to obtain a feasible solution $\bar{x}^{\text{Proc}}$ with objective function $\bar{z}^{\text{Proc}}$.
7: $\hat{z}^* \leftarrow \min\{\hat{z}^{\text{Ext}}, \hat{z}^{\text{Proc}}\}$.
8: **if** $\bar{z}^{\text{Ext}} > \bar{z}^{\text{Proc}}$ **then**
9: **return** Feasible solution $\bar{x}^{\text{Ext}}$ and upper bound $\hat{z}^*$.
10: **else**
11: **return** Feasible solution $\bar{x}^{\text{Proc}}$ and upper bound $\hat{z}^*$.

All runs of the **CPXbest** and TopoSort heuristics include both the processing constraint and the extraction constraint.

Table 3. Objective function values obtained using the TopoSort heuristics and the local-search heuristics in C-PIT instances where only a single resource constraint per time period was considered.

Instance	GrTS	GeTS	ExTS	LS 15 m	LS 1 h	LS 4 h	LS 8 h
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

Notes. All values have been divided by the optimal LP relaxation value of the corresponding problem and thus are less than 1.0. The **LS 15 m**, **LS 1 h**, **LS 4 h**, and **LS 8 h** heuristics all start from the solution found by the **ExTS** heuristic.

Table 4. Objective function values obtained using the TopoSort heuristics and the local-search heuristics on the C-PIT instances in which two resource constraints per time period were considered.

Instance	GrTS	GeTS	ExTS	LS 15 m	LS 1 h	LS 4 h	LS 8 h
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

Notes. All values have been divided by the optimal LP relaxation value of the corresponding problem and thus are less than 1.0. The **LS 15 m**, **LS 1 h**, **LS 4 h**, and **LS 8 h** heuristics all start from the solution found by the **ExTS** heuristic.

In Table 4, we present the objective function values obtained after using the TopoSort heuristics and the local-search heuristics. The relative performance of our heuristics when $R = 2$ is similar to that when $R = 1$ (see Table 3). 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 in which 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 15 m** heuristics, all instances are within 5% of optimality.

5. Final Remarks

In this article, we have shown that it is possible to successfully tackle real-sized instances of the 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 in which it is no longer necessary to schedule bench-phases, as is done today. Rather, using the assignment of blocks over time from 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 that require manual intervention by mining engineers to consider additional operational constraints. This problem, which also occurs 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 of extending the critical multiplier method to work explicitly with multiple side constraints (see Johnson 1968 for a list of possible examples). Another natural extension would consist of replacing the fixed capacity-constraint model with one in which the capacity is variable. The second, larger jump will consist of extending this methodology to work with multiple destinations for each block (Johnson 1968). In this regard, the recent work of Bienstock and Zuckerberg (2010) makes important advances and should be further studied and compared to this work.

Acknowledgments

Enrique Rubio thanks Codelco Chile for support through the Mining Technology Chair and BHP Billiton for the Mine Planning Chair. The authors thank Alexandra Newman for providing the AmericaMine data set. Part of this work is included in the master's thesis of Renaud Chicoisne, who visited Universidad Adolfo Ibañez during 2009. The authors thank IBM CPLEX for their licensing support. This work was partially funded by the Chilean Government [Projects FONDEF D06I1031, FONDECYT 110674 (M.G., D.E.), FONDECYT 1070749 (D.E.), ICM P05-004F (D.E.), CMM Basal (M.G., E.M.), and ANILLO ACT-88 (M.G., E.M.)].

References

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Englewood Cliff, NJ).

Amaya J, Espinoza D, Goycoolea M, Moreno E, Prevost T, Rubio E (2009) A scalable approach to optimal block scheduling. *Proc. APCOM 2009* (The Canadian Institute of Mining, Metallurgy and Petroleum, Vancouver, British Columbia, Canada), 567–575.

Bienstock D, Zuckerberg M (2010) Solving LP relaxations of large-scale precedence constrained problems. *Proc. 14th Conf. Integer Programming Combin. Optim. (IPCO)*. Lecture Notes in Computer Science, Vol. 6080. (Springer-Verlag, Berlin) 1–14.

Boland N, Fricke C, Froyland G (2010) A strengthened formulation and cutting planes for the open pit mine production scheduling problem. *Comput. Oper. Res.* 37(9):1641–1647.

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

Caccetta L, Hill SP (2003) An application of branch and cut to open pit mine scheduling. *J. Global Optim.* 27(2–3):349–365.

Chandran BG, Hochbaum DS (2009) A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Oper. Res.* 57(2):358–376.

Cherkassky BV, Goldberg AV (1997) On implementing the push-relabel method for the maximum flow problem. *Algorithmica* 19(4):390–410.

Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1998) *Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization (Wiley-Interscience, New York).

Dagdelen K (1985) Optimum multi-period open pit mine production scheduling. Ph.D. thesis, Colorado School of Mines, Golden, CO.

Dagdelen K (2007) Open pit optimisation—Strategies for improving economics of mining projects through mine planning. *Orebody Modelling and Strategic Mine Planning, Spectrum Ser.* 14:125–128.

Dagdelen K, Johnson TB (1986) Optimum open pit mine production scheduling by Lagrangian parameterization. *19th APCOM Sympos. Soc. Mining Engineers (AIME)* (Jostens Publications, State College, PA).

DataMine Software (2011) Accessed February 2011, http://www.datamine.co.uk.

Fricke C (2006) Applications of integer programming in open pit mining. Ph.D. thesis, Department of Mathematics and Statistics, The University of Melbourne, Melbourne, Australia.

Gaupp M (2008) Methods for improving the tractability of the block sequencing problem for an open pit mine. Ph.D. thesis, Division of Economics and Business, Colorado School of Mines, Golden, CO.

Gemcom Software (2011) Accessed February 2011, http://www.gemcom.com.

Gershon ME (1983) Optimal mine production scheduling: Evaluation of large scale mathematical programming approaches. *Internat. J. Mining Engrg.* 1(4):315–329.

Gershon ME (1987a) Heuristic approaches for mine planning and production scheduling. *Internat. J. Mining and Geological Engrg.* 5(1):1–13.

Gershon ME (1987b) An open-pit production scheduler: Algorithm and implementation. *AIME Trans.* 282:793–796.

Hochbaum DS (2008) The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Oper. Res.* 56(4):992–1009.

Hochbaum DS, Chandran BG (2009) A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Oper. Res.* 57(2):358–376.

Hochbaum DS, Chen A (2000) Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. *Oper. Res.* 48(6):894–914.

Hustrulid W, Kuchta K, eds. (2006) *Open Pit Mine Planning and Design* (Taylor and Francis, London).

Hustrulid WA, Carter MK, Van Zyl DJA, eds. (2000) *Slope Stability in Surface Mining* (Society for Mining, Metallurgy, and Exploration, Inc., Littleton, CO).

Isaaks EH, Srivastava RM (1989) *Applied Geostatistics* (Oxford University Press, New York).

Johnson TB (1968) Optimum open pit mine production scheduling. Ph.D. thesis, Operations Research Department, University of California, Berkeley, Berkeley.

Johnson TB (1969) Optimum open-pit mine production scheduling. Weiss A, ed. *A Decade of Digital Computing in the Mining Industry*, Chapter 4 (AIME, New York), 539–562.

Journel AG, Huidbregts CJ (1978) *Mining Geostatics* (Academic Press, San Diego).

King BM (2001) Optimal mine scheduling policies. Ph.D. thesis, Royal School of Mines, Imperial College London, London.

Lane K (1988) *The Economic Definition of Ore*: *Cutoff Grade in Theory and Practice* (Mining Journal Books Limited, London).

Lerchs H, Grossmann IF (1965) Optimum design of open-pit mines. *Transactions* LXVIII:17–24.

Maptek Software (2011) Accessed February 2011, http://www.maptek.com.

Matheron G (1975a) Le paramétrage des contours optimaux. Technical Report 403, Centre de Géostatistiques, Fontainebleau, France.

Matheron G (1975b) Le paramétrage technique des réserves. Technical Report 453, Centre de Géostatistiques, Fontainebleau, France.

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

Osanloo M, Gholamnejad J, Karimi B (2008) Long-term open pit mine production planning: A review of models and algorithms. *Internat. J. Mining, Reclamation and Environ.* 22(1):3–35.

Ramazan S, Dimitrakopoulos R (2004) Recent applications of operations research and efficient MIP formulations in open pit mining. *Soc. Mining, Metallurgy and Exploration Meeting Trans.* 316:73–78.

Ramazan S, Dagdelen K, Johnson TB (2005) Fundamental tree algorithm in optimising production scheduling for open pit mine design. *Mining Technology (Trans. Inst. Min. Metallurgy A)* 114:45–54.

Strategy Optimisation (2011) Accessed February 2011, http://www.stops.com.au.

Whittle J (1999) *Four-x Strategic Planning Software for Open Pit Mines.* Reference manual (Whittle Programming Pty Ltd., Victoria, Australia).

Renaud Chicoisne is a Ph.D. student in operations research in the Department of Industrial Engineering of the Universidad de Chile, Santiago, Chile. His research focuses on combinatorial optimization, linear and integer programming, and graph theory, with application within the areas of open-pit mining and transportation.

Daniel Espinoza is an assistant professor in the Industrial Engineering Department at the Universidad de Chile, Santiago, Chile. His research focuses on combinatorial optimization, mixed-integer programming, stochastic programming, and large-scale optimization problems.

Marcos Goycoolea is associate professor of operations research at the Universidad Adolfo Ibañez School of Business. His research interests are mostly on computational aspects of mixed-integer programming, specifically on cutting plane methods, decomposition methods, and natural resource management applications.

Eduardo Moreno is an associate professor in the Faculty of Engineering and Sciences at the Universidad Adolfo Ibañez, Santiago, Chile. His research focuses on combinatorial optimization, linear and integer programming, and graph theory, with application within the areas of mining, transportation, and telecommunications.

Enrique Rubio is chairman of the Delphos Mine Planning Laboratory at the University of Chile and director of Technology Transfer of the Advanced Mining Technology Center (AMTC). He also directs research and development of the recently created CSIRO-Chile mining center. His research is related to the application and development of operations research models in mine planning to support strategic, tactical, and operational decisions in mining.