Aggregation heuristic for the open-pit block scheduling problem

Enrique Jélvez[a,b,c], Nelson Morales[a,b], Pierre Nancel-Penard[a], Juan Peypouquet[d,*], Patricio Reyes[e]

[a] *Advanced Mining Technology Center, Universidad de Chile, Santiago, Chile*
[b] *Delphos Mine Planning Laboratory, Department of Mining Engineering, Universidad de Chile, Santiago, Chile*
[c] *CSIRO Chile International Centre of Excellence, Santiago, Chile*
[d] *Department of Mathematics, Universidad Técnica Federico Santa María, Valparaíso, Chile*
[e] *Department of Statistics and Operations Research, Universidad Carlos III de Madrid, Madrid, Spain*

Abstract

In order to establish a production plan, an open-pit mine is partitioned into a three-dimensional array of blocks. The order in which blocks are extracted and processed has a dramatic impact on the economic value of the exploitation. Since realistic models have millions of blocks and constraints, the combinatorial optimization problem of finding the extraction sequence that maximizes the profit is computationally intractable. In this work, we present a procedure, based on innovative aggregation and disaggregation heuristics, that allows us to get feasible and nearly optimal solutions. The method was tested on the public reference library `MineLib` and improved the best known results in the literature in 9 of the 11 instances of the library. Moreover, the overall procedure is very scalable, which makes it a promising tool for large size problems.

Keywords: mine scheduling, mine planning, block aggregation, open-pit block scheduling, surface mine production scheduling, integer programming, heuristics.

1. Introduction

The mining industry is a very relevant economic sector. In Chile, where this research has been carried out, copper exports account for about 62.5% of the total exports and represent a 12% of the GDP [9].

Mines can be either open-pit or underground, the actual decision depending on different economic and technical considerations. In this paper we focus on *open-pit* mines, in which mineral is extracted by digging from the surface. Open-pit mines are preferred to underground mines because they can reach higher production levels, and have smaller operational costs. However, most of the time, it is necessary to remove material with poor or none ore content (*waste*) in order to have access to economically profitable material.

The actual value of a mine strongly depends on the order in which the material is extracted and processed. In order to define what portions of the terrain must be mined at different moments during the life-time of the mine, the planning horizon is discretized into *time-periods* (or *time-slots*). In turn, the terrain is divided into regular *blocks*, which are arranged in a 3-dimensional array. For each block, estimations on the ore content, density and other relevant attributes are constructed by using geostatistical methods. A *block model*, namely, the set of all blocks and their attributes, is the main input to the mine planning process. Using this information, it is possible to build a *block scheduling*, which specifies an extraction time-period for each block. The final value of a mine is therefore determined by the block model and the block scheduling.

The feasibility of a block scheduling for the open-pit method depends on *accessibility* and *extraction* constraints. First, before extracting one block, all the blocks above it must have been extracted. Moreover, stability of the walls must be ensured. This is expressed in terms of slope angles that must be satisfied at each moment. All these constraints are translated into *precedences* between blocks. On the other hand,

*Correspondence: Juan Peypouquet, Department of Mathematics, Universidad Técnica Federico Santa María, Valparaíso, postal code 2390123, Chile. E-mail address: juan.peypouquet@usm.cl

there are certain capacity constraints, as well as other limitations, that are inherent to the process. The amount of material to be transported and processed at each period is subject to upper bounds given by transportation and plant capacity, respectively, which are usually expressed either in tons or hours. Further on, processed material must satisfy some *blending* constraints as well. The efficiency (or even feasibility) of the plant process depends on the attributes of the combination of blocks that are processed at a given period. For example, it may not be feasible to process alone a block with a high content of a certain pollutant (say arsenic), even it has a very high ore grade. Mixing it with another block (even a low ore-grade one) and processing them simultaneously may be possible because the blending provides an acceptable amount of the pollutant. Blending constraints can be either upper or lower bounds and apply to certain attributes of the blocks to be processed. Finally, the decision of how to process a block may depend on different parameters. Indeed, it is quite common for a mine to have more than one processing possibility (considering a block as waste and sending it to a waste pile is already one possibility). Depending on the final process or destination of the block, the net profit perceived by the mine is different, as are the blending constraints that apply to the process and the resource required to achieve this processing (different plant capacities, for example).

In this work, we propose and test a new numerical method to determine the block scheduling that maximizes the net present value for the exploitation of an open-pit mine. Our proposal is based on a combination of two approaches, that allow us to reduce the size of the problem and make it computationally tractable. The procedure aggregates blocks, uses integer programming techniques to solve incrementally the aggregated problem and produces solutions for the original instance in an innovative fashion. Using this methodology, we are able to provide nearly optimal solutions for some realistic-size problems that are otherwise numerically inaccessible.

The procedure was proved on the instances of the public reference library named `MineLib` [15], which has three different types of open-pit mine planning problems for which good feasible solutions have been reported: the ultimate pit limit problem and two variants of open-pit production scheduling problems, for fictional cases, but also for real-life mine (for example, the instances KD, P4HD, W23 and McLaughlin correspond to actual copper and gold mines located in North America). We focus on the Constrained Pit Limit Problem (`CPIT`), which consists of the maximization of the net present value (NPV) of the exploitation over the time horizon, subject to precedence and operational constraints. The results obtained by our procedure improve nine out of the eleven instances available in the `MineLib` library. Moreover, the remaining two cases are within a gap of 0.2% of the optimum solution.

The paper is organized as follows: In Section 2 we provide a brief summary of the most relevant (and best-known) approaches found in the literature. Section 3 contains all the details concerning the modeling, notation and problem statement. The description of our methodological proposal, as well as the different heuristics involved, are presented in Section 4. All the implementation details, and the numerical results obtained are given and commented in Section 5. Finally, Section 6 contains some concluding remarks and perspectives.

2. Related work

A very general formulation, due to Johnson [22], presents the block scheduling problem under slope, capacity and blending constraints (the last ones given by ranges of the processed ore grade) within a multi-destination setting, i.e., the optimization model decides what is the best process to apply to a given block. Unfortunately, at the time of its publication, the model was too complex to be solved in realistic case studies.

As an alternative to the work of Johnson, Lerchs and Grossman [25] proposed a very simplified version of the problem in which block destinations are fixed in advance, slope constraints are considered, but capacity or blending constraints are not. In this case, the problem reduces to selecting a subset of blocks such that the contained value is maximized while the precedence constraint induced by the slope angles are held. This problem is known as the *ultimate* or *final pit* problem. Lerchs and Grossman presented an efficient (polynomial) algorithm for solving the ultimate pit problem, and showed that reducing the economic value of any given block makes the optimal solution of the ultimate pit problem to *shrink*, in the sense that, if the values of the blocks decrease, the new solution is a subset of the original one. Therefore, it is possible

to produce *nested pits* and, by trial and error, construct block schedules that satisfy other constraints like capacity. Present-day commercial software, like Whittle [18], is based on these facts.

As it turns out, while the model proposed by Johnson (and others) has always been regarded as superior in terms of the value it can add to a mining plan, it has been only recently that new developments (specially in algorithms) have allowed to solve or approximate this kind of models. Indeed, a main motivation of this work is to contribute to transform the theoretical superiority of these mathematical models into a practical one.

Picard [30] showed that the ultimate pit problem is equivalent to the *maximum closure problem* in which, given a directed graph $G = (V, A)$ with weight function w defined over the nodes, one looks for a subset of nodes $U \subset V$ such that $\sum_{u \in U} w(u)$ is maximal but $u \in U, (u, v) \in A \Rightarrow v \in U$. The maximum closure problem, in turn, can be reduced to the *min cut* problem (for more details see [27]). Using this fact, Hochbaum [21] proposes to attack the ultimate pit problem by means of existing efficient algorithms for the min cut problem.

Caccetta and Hill [7] use a customized version of the *branch-and-bound* algorithm to solve problems up to a few hundred of thousands of blocks under blending and capacity constraints. Their method can be used only for upper bounds. Bley et al. [5] use a similar model but incorporating additional cuts based on the capacity constraints that strengthen the formulation of the problem, in the sense that the value of the linear relaxation provides a tighter bound. They test this approach on small instances (up to 500 blocks and 10 time-periods) on which they show very interesting improvements in the computational time. Unfortunately, it is not clear how to scale the technique for larger instances, as the number of cuts may explode very quickly. A closely related strategy is used by Fricke [16], in order to find inequalities that improve various integer formulations of the same model. Gaupp [17] reduces the size of the problem by deriving minimum and maximum extraction periods for each block, from the capacity constraints, and eliminating some of the variables. The method then applies Lagrangian relaxation to solve the problem.

The next two papers address the problem under consideration, but considering only upper bounds on resources consumption constraints: First, Amaya et al. [3] starting from an initial feasible solution and then iteratively fix parts of the incumbent solution and re-optimize the complement. At each iteration, this defines an integer programming sub-problem that is solved exactly. They are able to solve instances of up to 4 millions blocks and 15 time periods in 4 hours. In turn, Lamghari et al. [24] use a hybrid method based on linear programming and variable neighborhood descent. The authors introduce a two-phase solution method: in the first one, they solve a series of linear programming problems to generate an initial solution. In the second phase, a variable neighborhood descent procedure is applied to improve the solution. The method is tested on some benchmark instances from the literature (some of `MineLib`), showing new best-known solutions for almost all of the instances, when compared to the solutions reported in [24] and [15]. Indeed, only in two of these instances the solutions obtained have a larger gap, but this is still at much 0.2%.

Following Picard and Hochbaum ideas, Chicoisne et al. [8], and Bienstock and Zuckerberg [4] address a problem which is very close to the one considered in this paper. However, they use Lagrangian relaxation on all but the precedence constraints (in this case the problem reduces to the ultimate pit problem). Using this approach, Chicoisne et al. focus on the case where there exists only one destination and one capacity constraint per period, and develop a customized algorithm (`CMA`) for the linear relaxation and a procedure (based upon topological sorting) to obtain an integer feasible solutions from it. They report good solutions for the problem in large instances (over one million blocks). Moreover, Espinoza et al. [15] published a library of standardized instances, named `MineLib`, for which they apply the above techniques and obtain very good results as well. Respectively, Bienstock and Zuckerberg consider all types of constraints, but focus on the resolution of the linear relaxation only, and report very good improvements in resolution time with respect to the standard LP solvers.

Other work which is close to ours is due to Cullenbine et al. [11]. They propose a heuristic using Lagrangian relaxation on capacity constraints (lower and upper bounds) plus a *sliding window* strategy in which extraction variables for late periods are relaxed while variables corresponding to early periods are fixed incrementally. They work on a slightly different problem in which the bottom of the pit must contain at least two adjacent blocks, and report improvements in the execution time with respect to standard solvers. A

recent work of Lambert and Newman [23] employs a tailored Lagrangian relaxation, which uses information obtained while generating the initial solution to select a dualization scheme for the resource constraints. They report solutions for models having up to 25.000 blocks and 10 time-periods at 36,000 seconds, with an optimality gap of 6% in the largest case.

Another approach to tackle large-scale problems is based on aggregation procedures. Dagdelen et al. [13, 12], and Ramazan et al. [31], work on a model with fixed cut-off grades, upper and lower bounds for blending, but only upper bounds for the capacity. They aggregate blocks into what they call *fundamental trees* (subsets that have positive value, respect slope constraints and are minimal in some sense) and present a relatively small case study (less than 15,000 blocks). Boland et al. [6] propose a different procedure, in which they aggregate blocks into what they call *bins*. The extraction of individual blocks is controlled with continuous variables, but binary variables are used at the bin level to impose slope constraints. They are able to solve instances of about 100,000 blocks.

Approaches not based in linear programming include *genetic algorithms* and *tabu search*. Zhang [33] uses Genetic algorithm combined with a block aggregation technique based on topological sort to reduce of number of variables in the model. The method simultaneously determines an ultimate pit and an optimal block extraction schedule that maximizes the net present value by specifying whether a block should be extracted and where it should be sent (waste dump or processing plant), subject to a number of constraints including maximum wall slope, as well as mining and processing capacities. By this approach, Zhang concludes that the computational time can be effectively reduced without compromising optimality. The method was implemented and tested against BHP-Billiton's existing industrial benchmark achieved by the commercial optimizer `ILOG CPLEX` [10]. According to Amankwah [2], the nature of the constraints poses a major difficulty in the use of genetic algorithms to solve the mine planning problem. Newman et al. [28] point out that Zhang does not assess the practical consequences of aggregation and does not provide a disaggregation procedure. In turn, Tabesh and Askari-Nasab [32], propose an algorithm that aggregates blocks into mining units and uses tabu search to calibrate the number of final units. The resulting problem is then solved using standard integer programming algorithms. The aggregation technique is interesting, because it is based on a similarity index that considers attributes like rock type, ore grades and the distance between the blocks. The tabu search procedure is then used to further aggregate the blocks, while trying to balance the resulting loss of selectivity. The procedure is tested on 5 different instances (with up to 20,000 blocks), which show a variety of results where improvements in the objective function value (NPV) and computational time are not completely consistent. The authors indicate that further research is required.

A completely different approach, suggested by Matheron [26], uses continuous models to describe the problem. The pits are described by *profiles*, which are functions that determine the current surface of the pit. Block attributes are modeled as *density* functions, whose integrals have to be either maximized or kept within certain ranges. This approach was followed by Alvarez et al. [1], showing existence of solutions under some hypotheses on the mass density function.

Additional reviews of operations research in mining can be found in Osanloo et al. [29] for models and algorithms and Newman et al. [28] for mining in general.

3. Modeling, notation and problem statement

We consider a set $\mathcal{B}$ of blocks and set $N = |\mathcal{B}|$. We denote the blocks with indices $i, j \in \mathcal{B}$, unless otherwise stated. Similarly, we consider $T \in \mathbb{N}$ time-periods and denote individual time-periods with $s, t = 1, 2, \ldots, T$. The number T is called the *time horizon*. The set of time-periods is denoted by $\mathbf{T} = \{1, 2, \ldots, T\}$.

Slope (precedence) constraints are encoded as a set of *arcs* $\mathcal{A} \subset \mathcal{B} \times \mathcal{B}$: $(i, j) \in \mathcal{A}$ means that block j has to be extracted before block i. We say, in this case, that block j is a predecessor of block i, which is a successor of j. Notice that arc (i, j) goes from the successor to the predecessor.

In this work we address a simplified version of the problem in which the decision of the destination of the block is done beforehand. Therefore, the net profit (which can be negative) of processing block i is already known and noted as $v_i \in \mathbb{R}$. The discounted net profit of processing block i at time-period t is $\rho^t v_i \in \mathbb{R}$, with discount factor $\rho = \frac{1}{1+dr}$, where dr represents the discount rate. We define a set of resources $\mathcal{R}$, and

$a(i,r) \in \mathbb{R}$ as the quantity of resource $r \in \mathcal{R}$ used when block $i \in \mathcal{B}$ is processed. For each time-period t, lower and upper bounds on the consumption of resource r is given by the quantities $C^-_{rt} \in \{-\infty\} \cup \mathbb{R}$ and $C^+_{rt} \in \{+\infty\} \cup \mathbb{R}$, respectively.

Each block is processed in the same time-period in which it is extracted from the mine (that is, we do not allow to stock material for future processing). As usual in these models, we assume that extraction, handling and processing of a block is done within a time-period length. While the modeling can be easily extended to the general case, the heuristics presented in this article do not always work in the case in which blending constraints apply, therefore, we assume there are no such constraints. Table 1 summarizes the notation.

Symbol	Meaning
$\mathcal{B}$	Set of blocks
i, j	Blocks (elements of $\mathcal{B}$)
s, t	Time-periods
T	Time horizon (number of time-periods)
$\mathbf{T}$	Set of time-periods
$\mathcal{A}$	Set of precedence arcs
$\mathcal{R}$	Set of resources
v_i	Economic value (net profit) of block i
ρ	Discount factor (depends on discount rate)
$a(i,r)$	Consumption of resource r by block i
C^-_{rt}	Lower bound on resource r at time-period t
C^+_{rt}	Upper bound on resource r at time-period t

Table 1: Main notation used in the article.

A *block scheduling* is a function $\tau : \mathcal{B} \to \{1, 2, \ldots, T, \infty\}$ where $\tau(i)$ is the time-period in which block i is extracted, hence, a block scheduling must satisfy the precedence constraints, that is, if $(i,j) \in \mathcal{A}$ then $\tau(i) \geq \tau(j)$. If τ is a block scheduling then the sets $P_1 = \tau^{-1}(1)$ and $P_t = P_{t-1} \cup \tau^{-1}(t)$ for $t > 1$ are called *pits*. We observe that $P_t \subset P_{t+1}$ and say that the pits are *nested*.

The open-pit block scheduling problem is defined on the following variables. For each $i \in \mathcal{B}, t \in \mathbf{T}$:

$$x_{it} = \begin{cases} 1 & \text{if block } i \text{ is extracted by time-period } t, \\ 0 & \text{otherwise.} \end{cases}$$

The interpretation of variable x_{it} is *by* time-period, that is $x_{it} = 1$ if and only if block i has been extracted (and processed) at some period s with $1 \leq s \leq t$. It is useful to introduce the following auxiliary variables for $i \in \mathcal{B}$: $\Delta x_{i1} = x_{i1}$, and $\Delta x_{it} = x_{it} - x_{i,t-1}$ for $t = 2, 3, \ldots, T$. We have $x_{it} = \sum_{s=1}^{t} \Delta x_{is}$ and $\Delta x_{it} = 1$ if, and only if, block i is extracted exactly at time-period t.

The mathematical program we address is the following:

$$(\texttt{OPBSP}) \max \sum_{t=1}^{T} \sum_{i=1}^{N} \rho^t v_i \Delta x_{it} \tag{1}$$

$$x_{it} \leq x_{jt} \qquad (\forall (i,j) \in \mathcal{A})(\forall t \in \mathbf{T}) \tag{2}$$

$$\Delta x_{it} \geq 0 \qquad (\forall i \in \mathcal{B})(\forall t \in \mathbf{T}) \tag{3}$$

$$\sum_{i \in \mathcal{B}} a(i,r) \Delta x_{it} \leq C^+_{rt} \qquad (\forall r \in \mathcal{R})(\forall t \in \mathbf{T}) \tag{4}$$

$$\sum_{i \in \mathcal{B}} a(i,r) \Delta x_{it} \geq C^-_{rt} \qquad (\forall r \in \mathcal{R})(t \in \mathbf{T}) \tag{5}$$

$$x_{it} \in \{0,1\} \qquad (\forall i \in \mathcal{B})(\forall t \in \mathbf{T}) \tag{6}$$

Expression (1) presents the objective function, which is the discounted value of extracted blocks over the time horizon T. In turn, (2) corresponds to the precedence constraints given by the slope angle and (3) means that blocks can be extracted only once. Moreover, (4) and (5) state the maximum and minimum resource consumption in each time-period, respectively. Finally, (6) states that all the variables assume binary values.

For a block model $\mathcal{B}$, precedence arcs $\mathcal{A}$, block values $V = (v_i)_{i\in\mathcal{B}}$ and attribute matrix $A = (a(i,r))_{i\in\mathcal{B},r\in\mathcal{R}}$, we will use the notation $\texttt{OPBSP}(\mathcal{B},\mathcal{A},V,A,T,\rho,C^+,C^-)$ to denote an instance of the open-pit block scheduling problem for a certain time horizon T, discount factor ρ, and resource limit matrices $C^+ = (C^+_{r,t})_{r\in\mathcal{R},t\in\mathbf{T}}$ and $C^- = (C^-_{r,t})_{r\in\mathcal{R},t\in\mathbf{T}}$. We will omit some of the parameters if they are clear from the context.

It is important to emphasize that we present the model with lower bounds on operational resource capacities (5) to keep the model consistent with `CPIT`. However, all of the instances published in `MineLib` have these bounds equal to zero.

4. Block Aggregation Algorithm

We present now the *Block Aggregation Algorithm* to approximately solve the open-pit block scheduling problem, when the original instance of `OPBSP` can not be solved directly.

The procedure first aggregates blocks, then it solves incrementally the aggregated problem (using integer programming techniques) and finally, it produces feasible solutions for the original instance. The algorithm has two stages: A **forward** stage, where the procedure tries to solve the problem using a simplified block model (obtained by the reblocking procedure described in 4.1.2); and a **backward** stage, in which the solution from the forward stage is used to fix some variables in the original problem. Figure 1 outlines the procedure.

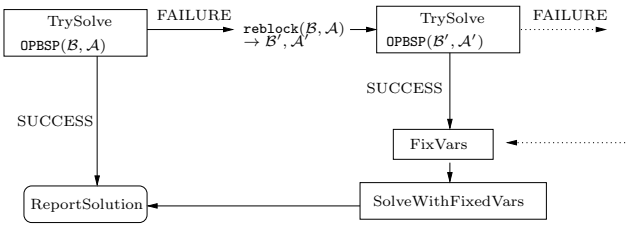

Figure 1: Diagram of iterations for the block aggregation heuristic.

Although block aggregation appears to be simple, the reverse step (the forming of a block scheduling for the original disaggregated model) is not so straightforward. In fact, it is challenging in terms of implementation.

4.1. The forward stage

The goal of the forward stage is to reduce the number of variables by means of block aggregation. The stage can be described as follows:

0. Set `currentInstance` $\leftarrow (\mathcal{B},\mathcal{A},V,A,T,\rho,C^+,C^-)$
1. `SOLVE` $\texttt{OPBSP}($`currentInstance`$)$. (See section 4.1.1)
2. `If` step 1 found a feasible solution, `then` go to **backward stage** (See section 4.2).
3. `else REBLOCK` (See section 4.1.2) to produce a new instance

$$\texttt{currentInstance} \leftarrow (\mathcal{B}^R,\mathcal{A}^R,V(\mathcal{B}^R),A(\mathcal{B}^R),T,\rho,C^+,C^-).$$

and go to step 1.

This recursive procedure is very flexible: it can use any subroutine to `SOLVE` the instance of the problem in step 1, and to `REBLOCK` the model in step 3. In the following, we will describe the procedures used in our case studies (Section 5) which proved to be very effective.

4.1.1. Solving $\texttt{OPBSP}(\mathcal{B}, \mathcal{A}, V, A, T, \rho, C^+, C^-)$.

The standard approach of solving directly the IP formulation is highly unpractical for large-scale problems. Since the number of blocks is considerably larger than the number of periods, a reduction in the number of time-periods can have a great impact in reducing the overall number of variables and constraints. We propose the following:

Incremental Heuristic (`HInc`). This heuristic tries to solve the problem incrementally: it considers only fewer time-periods (which we call *sliding time window* or just *time window*). Then, the heuristic solves and removes the scheduled blocks in the current time window. The process is repeated with the remaining blocks and time-periods, adjusting the constraints accordingly and moving the time window until the horizon planning is completed. In order to fix ideas, the simplest approach is to consider the time window as only one time-period and then solve the problem period by period.

More precisely, `HInc` considers an instance of the open-pit block scheduling problem $\texttt{OPBSP}(\mathcal{B}, \mathcal{A}, V, A, T, \rho, C^+, C^-)$. Given $B \subset \mathcal{B}$, define:

- $\mathcal{A}(B) = \mathcal{A} \cap (B \times B)$,
- $V(B) = (v_i)_{i \in B}$, and
- $A(B) = (a(i,r))_{i \in B, r \in \mathcal{R}}$

Suppose that $\texttt{OPBSP}(T)$ can not be solved, but $\texttt{OPBSP}(T')$ can, for $T' < T$. For simplicity, we describe the heuristic when $T/T' = n \in \mathbb{N}$. Thus, let also C_k^+ and C_k^- be the maximum and minimum resource limits for time-periods $(k-1)T'+1, \dots, kT'$, with $1 \leq k \leq n$. The heuristic then works as follows:

1. Set $n = T/T', k = 1, P_0 = \emptyset$.
2. While $k \leq n$:
   (a) Set $B_k = \mathcal{B} \setminus P_{k-1}$.
   (b) Solve $\texttt{OPBSP}(B_k, \mathcal{A}(B_k), V(B_k), A(B_k), T', \rho', C_k^+, C_k^-)$.
   (c) Let $\Gamma_k = \bigcup_{s=(k-1)T'+1}^{kT'} \tau^{-1}(s)$ be the set of extracted blocks and set $P_k = P_{k-1} \cup \Gamma_k$.
   (d) $k \leftarrow k+1$.
3. Return the block scheduling τ given by $\tau(i) = t_0$ if $i \in P_{t_0}$ for some t_0, or $\tau = \infty$ if there is no t such that $i \in P_t$.

Notice that τ is well-defined as the sets $P_1, P_2, \dots, P_n$ are nested.

Figure 2 illustrates how the block scheduling is generated, with a time-horizon of $T = 3$ and a time window $T' = 1$: in the first time-period, the extracted blocks are the set P_1. Then these blocks are removed and the procedure is repeated for the second period. During the second period, the extracted blocks correspond to the set $P_2 \setminus P_1$. Finally, the process is repeated for the third and last period.

Although this procedure does not necessarily produce optimal results (see Example 1), it can give a significant improvement in practice, as we describe in Section 5.

Example 1. The small 2-dimensional mine displayed in Figure 3 is to be exploited in a 5-period time horizon. Each number represents the economic value of the corresponding block. For simplicity we assume that each block has an unit tonnage, the capacity per time-period is also one unit (hence exactly one block can be extracted in each time-period), and there is no discount rate.

We index the blocks by integer coordinates as in a matrix, so the top-left block is identified with coordinates $(1,1)$ and the bottom-right block has coordinates $(2,4)$. If the slope angle is $45°$, block (x,y) has predecessors $(x-1, y-1), (x-1, y)$ and $(x-1, y+1)$ if $x = 2$ (when such blocks exist) and no predecessors if $x = 1$.

The optimal scheduling is to extract blocks $(1,1), (1,2), (1,3), (1,4)$ in any order during the first four periods, and then to extract block $(2,3)$ at period 5. The corresponding value is $\text{NPV} = 1 + (-1) + (-1) + (-1) + 10 = 8$. However, the incremental heuristic ($T' = 1$) will select block $(1,1)$ at the first iteration, and no blocks afterwards. The value of this solution is 1, which is suboptimal.

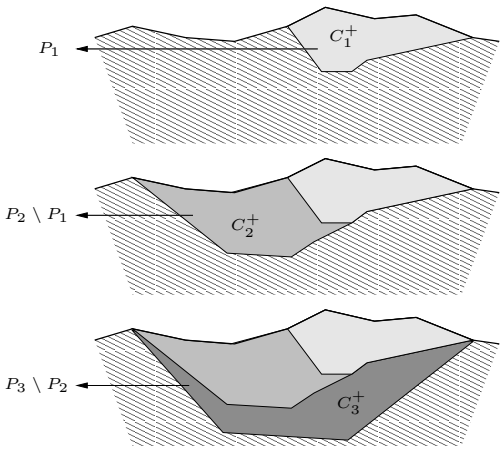

Figure 2: A graphic example of incremental heuristic `HInc` with $T' = 1$ and $T = 3$.

1	-1	-1	-1
-1	-1	10	-1

Figure 3: A small 2-D mine example.

4.1.2. Reblocking

In this step, we do not reduce the number of time-periods, but we reduce the number of blocks by aggregating (or *reblocking*) them into larger units. The motivation comes from the fact that while a block model consists of thousands to millions of blocks, the decision of extracting a specific block at a certain time-period may be too *atomic*.

We consider a procedure `reblock` that takes a block model $\mathcal{B}$ and produces a new one, say $\tilde{\mathcal{B}}$, so that the number of blocks in the new model is small compared to the original block model. The idea is that $\tilde{\mathcal{B}}$ keeps certain properties of the original one, then solving `OPBSP` in $\tilde{\mathcal{B}}$ can be used to produce solutions for the original model. This is done using the following:

Block aggregation heuristic (`HReb`). Let us introduce some notation first. We consider that `reblock` : $\mathcal{B} \to 2^{\mathcal{B}}$ thus $Im(\texttt{reblock})$ is a partition of $\mathcal{B}$. We identify $Im(\texttt{reblock})$ as a new block model $\mathcal{B}^R$, so that each block in $\mathcal{B}^R$ has the same set of attributes of the blocks in the original block model. We will use notation i^R, j^R to refer to elements of $\mathcal{B}^R$ in order to prevent any confusion with blocks of the original model. Finally, we will also assume that $\mathcal{A}$ is also *translated* into $\mathcal{A}^R$ in a "compatible" way that allows us to use a scheduling in $\mathcal{B}^R$ to construct a scheduling in $\mathcal{B}$ respecting the slope constraints.

Note that `reblock` is defined in a very general way. It may depend on geometric arguments (distances between blocks, for example) or block characteristics (lithology and ore grades). In this work, we consider geometric aspects, defining how many blocks $i \in \mathcal{B}$ are included in each aggregated block $i^R \in \mathcal{B}^R$. Considering that each block is identified by coordinates (x, y, z), each aggregated block $i^R = (x_i^R, y_i^R, z_i^R)$ will be composed by the $n^R \times n^R \times n^R$ original blocks with coordinates $(x_i^R n^R + j_x, y_i^R n^R + j_y, z_i^R n^R + j_z)$, with $j_x, j_y, j_z \in \{0, \cdots, n^R - 1\}$. An example (only in 2 dimensions for clarity) can be seen in Figure 4.

4.2. The backward stage

This is the most important part, since it allows us to obtain a scheduling for the original block model starting from the aggregated model solution. We assume that the problem was solved in the forward stage for the instance

$$\texttt{OPBSP}(\mathcal{B}^R, \mathcal{A}^R, V(\mathcal{B}^R), A(\mathcal{B}^R), T, \rho, C^+, C^-),$$

so we have a block scheduling τ^R for the reblocked model and we want to produce a solution for the original instance. This involves the following steps:

1. Partition $\mathcal{B}^R$ into ∂B and $\mathring{B}$ where ∂B is the set of aggregated blocks in the *borders*, which are those with neighboring blocks extracted at a different time period, and $\mathring{B}$ is the complement.
2. Fix the extracting period for the original blocks that correspond to elements of $\mathring{B} \subset \mathcal{B}^R$, i.e. the set of blocks such that all direct predecessors and successors are extracted at the same time-period. This is, $i^R \in \mathring{B}$ if and only if the set of aggregated $i^R \in \mathcal{B}^R$ satisfies that
   (a) $\forall (i^R, j^R) \in \mathcal{A}^R,\ \tau^R(i^R) = \tau^R(j^R)$
   (b) $\forall (j^R, i^R) \in \mathcal{A}^R,\ \tau^R(i^R) = \tau^R(j^R)$.
   Then if $i^R = \texttt{reblock}(i)$, we will assume that block $i \in \mathcal{B}$ is extracted at time-period $\tau^R(i^R)$.
   Add the constraints $x_{it} = 0$ for $t < \tau^R(i^R)$ and $x_{it} = 1$ for $t \geq \tau^R(i^R)$, for any $i \in \mathcal{B}$ such that $\texttt{reblock}(i) \in \mathring{B}$.
3. For the original blocks i that correspond to elements of ∂B, compute the following bounds, considering that $j^R = \texttt{reblock}(j)$:
   (a) $t^-(i) = \max\{\tau^R(j^R) : (i,j) \in \mathcal{A}, j^R \in \partial B\}$
   (b) $t^+(i) = \min\{\tau^R(j^R) : (j,i) \in \mathcal{A}, j^R \in \partial B\}$
   Add the constraints $x_{it} = 0$ for $t < t^-(i)$ and $x_{it} = 1$ for $t > t^+(i)$, for any $i \in \mathcal{B}$ such that $\texttt{reblock}(i) \in \partial B$.
4. Adjust capacities as:
   (a) $\partial C^+_{rt} = C^+_{rt} - \sum_{i \in I_t} a(i,r)$, $(\forall r \in \mathcal{R})(\forall t \in \mathbf{T})$, where $I_t = \{i \in \mathring{B} : \tau^R(i^R) = t\}$
   (b) $\partial C^-_{rt} = \max\{C^-_{rt} - \sum_{i \in I_t} a(i,r), 0\}$, $(\forall r \in \mathcal{R})(\forall t \in \mathbf{T})$, where $I_t = \{i \in \mathring{B} : \tau^R(i^R) = t\}$
5. Solve $\texttt{OPBSP}(\mathcal{B}, \mathcal{A}, V, A, T, \rho, \partial C^+, \partial C^-)$ under the additional constraints.

An illustration of this heuristic in a 2D-example can be found in Figure 4, in which individual blocks in $\mathcal{B}$ (a) are aggregated into big blocks containing 9 of them ($n^R = 3$). While the original block model has 315 blocks, the reblocked model contains only 35 blocks (b). The problem is then solved over these 35 blocks (c) and we set the blocks at the inner part (denoted $\mathring{B}$) of each period (d). The original blocks corresponding to these aggregated blocks are removed from the original model, the capacities are updated and the problem is solved for the remaining blocks (e). Finally, all the blocks are scheduled by mixing the solutions (f).

It is important to note that the heuristic proposed is myopic. Thus, it cannot guarantee that a feasible solution will be found, if exists. If an auxiliary problem becomes infeasible in the Backward Stage, it will be necessary to extend T' in order to try to find feasibility.

5. Implementation and results

In this section we discuss some specific details regarding the implementation of the Block Aggregation Algorithm introduced in Section 4. We also present the best known results obtained for the reference dataset `MineLib`. The idea is to illustrate how the different heuristics scale when the size and number of blocks change. We will see that a relevant feature of this approach is the ability to solve large problems.

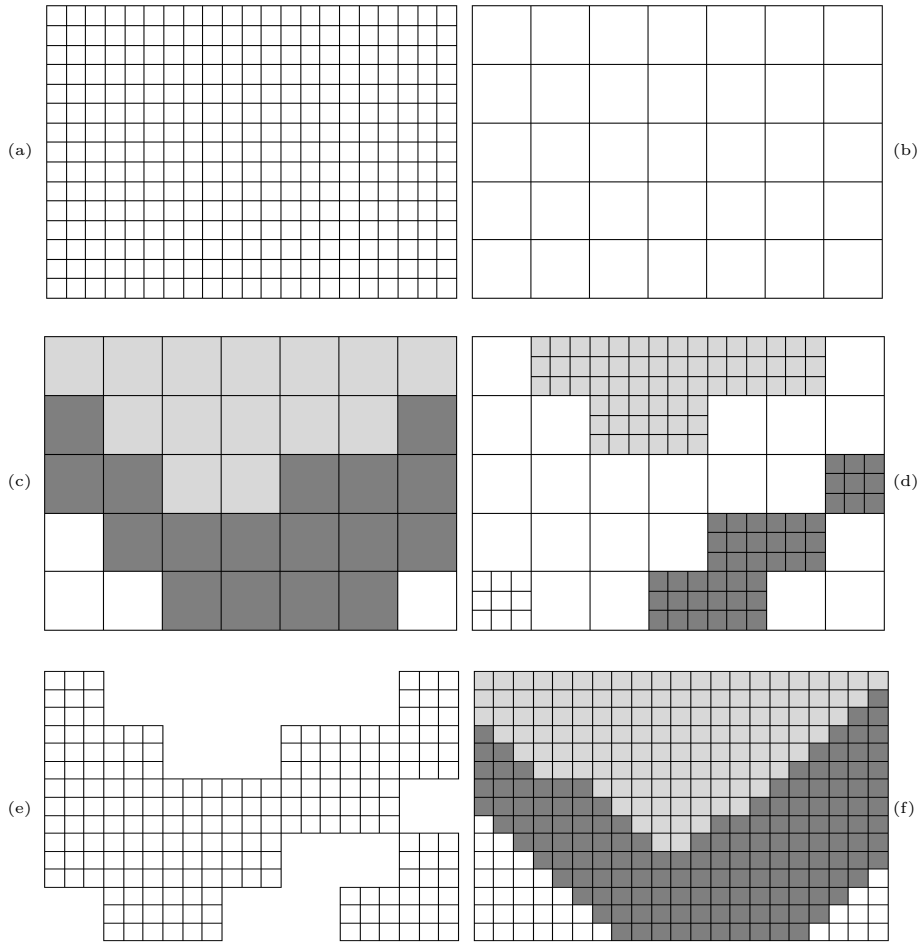

Figure 4: Example of the block aggregation heuristic.(a) Original block model. (b) The aggregated model. (c) The scheduling problem is solved. (d) The extraction period is fixed for the *Inner* blocks. (e) The *Border* blocks are left for solving at the original scale. (f) The final solution is made.

5.1. Software and hardware

In order to implement the different algorithms, we used the `MineLink` library developed at Delphos Mine Planning Laboratory at Universidad de Chile [14], which implements data structures to store the block model, precedence arcs and reblocking routines. The library is written in C++, but there also exists a wrapper to use it from `Python`, a general-purpose free available scripting language, version 2.7. Within this library, the `BOS2`® module implements the mathematical model and the heuristics described earlier. The module uses the `GUROBI`® [19] version 5.6.3 for mixed integer linear optimization. Execution of the code was done on an Intel Core i5-3570 machine with 16Gb running Ubuntu version 12.10. This machine has 4 processors that run at 3.40 Ghz.

5.2. Datasets and pre-existing solutions

The case studies were obtained from `MineLib`, a library of publicly available test problem instances for open-pit mining. The dataset can be found at http://mansci-web.uai.cl/minelib/. These instances come from real-world mining projects and simulated data. For each instance, the database contains the block model, precedence relations, additional constraints and parameters needed to formulate the optimization

problem. A description of the instances (time-horizon, number of: blocks, precedence arcs, variables and constraints) can be found in Table 2. For more details on `MineLib` see [15].

Instance Name	T	Blocks	Prec. arcs	Vars.	Constr.
`Newman1`	6	1,060	3,922	6,360	10,294
`Zuck small`	20	9,400	145,640	188,000	333,680
`KD`	12	14,153	219,778	169,836	389,626
`Zuck medium`	15	29,277	1,271,207	439,155	1,710,392
`P4HD`	10	40,947	738,609	409,470	1,148,099
`Marvin`	20	53,271	650,631	1,065,420	1,716,091
`W23`	12	74,260	764,786	891,120	1,655,942
`Zuck large`	30	96,821	1,053,105	2,904,630	3,957,795
`SM2`	30	99,014	96,642	2,970,420	3,067,122
`McLaughlin lim`	15	112,687	3,035,483	1,690,305	4,725,803
`McLaughlin`	20	2,140,342	73,143,770	42,806,840	115,950,630

Table 2: Summary of problem instances available in `MineLib`[15]. Columns indicate time-horizon, number of blocks, precedence arcs, number of variables and constraints.

Instance	Forward stage				Backward stage			
name	Blocks	Prec. arcs	Vars.	Constr.	Blocks	Prec. arcs	Vars.	Constr.
`Newman1`	242	745	1,452	2,209	862	3,052	5,172	8,236
`Zuck small`	1,494	89,270	29,880	119,190	9,193	139,345	183,860	323,245
`KD`	2,239	27,949	26,868	54,829	10,666	143,168	127,992	271,172
`Zuck medium`	4,280	209,526	64,200	273,756	29,277	1,271,207	439,155	1,710,392
`P4HD`	6,133	188,363	61,330	249,713	32,134	509,243	321,340	830,603
`Marvin`	1,360	25,529	27,200	52,769	8,336	82,788	166,720	249,548
`W23`	7,638	82,416	91,656	174,108	41,998	395,517	503,976	899,529
`Zuck large`	13,949	101,260	418,470	519,790	78,696	739,247	2,360,880	3,100,187
`SM2`	3,516	3,272	105,480	108,812	18,278	17,445	548,340	565,845
`McLaughlin lim`	16,531	512,165	247,965	760,145	76,491	1,583,404	1,147,365	2,730,784
`McLaughlin`	16,315	505,101	326,300	831,421	73,948	1,512,652	1,478,960	2,991,632

Table 3: Summary of forward (`HInc`+`HReb`) and backward stages, for each instance available in `MineLib`. "Forward stage" corresponds to the problem settings after aggregation. "Backward stage" corresponds to the remaining original blocks after solving the "Forward stage" (See sections 4.1 and 4.2 for more details)

We compare against pre-existing solutions for these instances that have been published in [15] and [24]. We briefly describe the algorithms involved in these two references.

In [15], the results were obtained by creating a *rank* of blocks, constructing a block-by-block sequence and then following this sequence up to saturate the capacities per period as follows:

1. Firstly, a solution $\mathbf{x}$ to the integrality relaxation of the problem is obtained.
2. For each (possibly partially) extracted block, one computes $Q(i) = \sum_{t=1}^{T} t\Delta x_{it} + (T+1)(1 - x_{iT})$. The values $Q(i)$ can be interpreted as expected times of extraction.
3. A block-by-block sequence is produced by extracting the blocks from the surface in increasing values of $Q(i)$ while respecting the precedence constraints.
4. Finally, the block sequence is used to construct a block schedule by filling each time period with blocks as long as the capacity constraints allow it.

Instance	LP upper bound	Current best results		HInc+HReb results		
		Source	GAP [%]	Value ($)	GAP [%]	Time [s]
Newman	24,486,184	LDF14	1.68%	24,173,521	**1.28%**	3
Zuck small	854,182,396	LDF14	1.80%	846,850,343	**0.86%**	913
KD	409,498,555	EGMN13	3.09%	408,718,363	**0.19%**	45
Zuck medium	710,641,410	LDF14	8.08%	669,191,136	**5.83%**	16,304
P4HD	247,415,730	LDF14	**0.19%**	246,917,483	0.20%	159
Marvin	863,916,131	LDF14	1.87%	856,191,778	**0.89%**	771
W23	400,653,199	LDF14	1.06%	398,457,833	**0.55%**	395
Zuck large	57,389,094	LDF14	0.56%	57,253,697	**0.24%**	1,825
SM2	1,648,051,083	LDF14	**0.04%**	1,646,784,822	0.08%	30
McLaughin lim	1,078,979,501	EGMN13	0.52%	1,078,261,873	**0.07%**	852
McLaughin	1,079,024,268	EGMN13	0.53%	1,078,378,642	**0.06%**	1,229

Table 4: Comparison between pre-existing results and the ones obtained using the HInc+HReb heuristic.

In [24], they present two ideas. A method to produce an initial solution, and a method to improve this solution based on a type of local search algorithm called Variable Neighborhood Descent(VND) introduced in [20] to approach what they call the Mine Production Scheduling Problem (MPSP), which corresponds to the CPIT, but with no lower bounds for the capacity constraints.

Using the VND method on the solutions provided in MineLib, they are able to improve 4 of the 11 cases. Additional 4 cases are improved by using their method to produce the starting solution and then applying VND.

Finally, notice that an upper bound to the value of any feasible solution of the problem can be obtained by relaxing the integrality constraints of the variables. These upper bounds are reported in [15].

5.3. Results

We tested the aggregation heuristic in all of the CPIT instances of MineLib. In all of them, the problem was solved using one Forward Stage and its corresponding Backward Stage. Table 3 shows the instances obtained in the Forward and Backward stages.

For all of the instances, Forward Stage was implemented using HInc and HReb. HInc was set to start with a sliding time window $T' = 1$ period (See Section 4.1.1). If no feasible solution was found, the sliding window was increased in one period until a feasible solution was achieved. The reblocking was implemented with HReb as described in section 4.1.2. The Backward Stage was implemented as explained in section 4.2.

Finally, the complete procedure will be denoted as HInc+HReb.

Table 4 shows the results for the MineLib instances. For each case we report:

1. *LP upper bound value*, obtained by relaxing the integrality constraints on the variables, as reported in [15]. This is used to compute the GAPs of the feasible solutions.
2. *Source*, indicating the paper in which the current best solution was reported: "LDF14" corresponds to [24] and "EGMN13" to [15].
3. *GAP*, the integrality GAP between the currently best known solution and the LP upper bound.
4. *Value*, the value of the solution obtained using our heuristic.
5. *GAP*, the integrality GAP between the solution obtained by the heuristic and the LP upper bound.
6. *Time*, the solution time of the heuristic. The time reported is the real time (also known as wall-clock time) and it includes all the pre-processing steps (precedence computations, Final-Pit when needed).

The MineLib instances for the CPIT problem do not have lower bounds on the operational resource constraints (they are equal to zero and the coefficients of these constraints are positive). Then, it is possible to apply a pre-processing, using a well known result such that any block schedule will be inside the final pit outline [7]. Indeed, most of the instances in MineLib are already solutions for the Final Pit. For the

remaining instances (`Marvin`, `SM2`, `W23` and `McLaughlin`), the pre-processing considered first to solve the Final Pit problem. Then, only the blocks in the final pit were given as the input for the heuristics. Thus, the problem size was considerably reduced.

5.4. Discussion

The aggregation heuristic `HInc+HReb` was able to find feasible solutions for all of the problems in the `MineLib` dataset. Even more, our heuristic obtained the best known feasible solutions in all of the instances, except for two instances (P4HD and SM2). Still, in both cases our heuristic reported solutions with GAP of at most 0.2%.

`MineLib` does not provide information about block size and precedence type for the instances `Zuck small`, `Zuck medium` and `Zuck large`. Thus, it was difficult to find out the right precedence model for the aggregated instances (However, the feasibility of the solution was verified with respect to the original precedence relationships). This explains in part why the Forward Stage was not as effective as expected compared to all the remaining instances: the problem size was not reduced as expected. For example, in `Zuck medium`, the number of blocks in the Backward Stage is still 99% with respect to the original number of blocks. This explains that the relative gap is the highest of the instances considered.

Apart from `Zuck small`, `Zuck medium` and `Zuck large` (block size and precedences are not clear), the Forward Stage considerably reduced the size of the problem. For example, in `KD`, the Backward Stage has 26% fewer blocks and 32% fewer constraints than the original problem. Even more, in 3 out of 4 of the instances pre-solved with Final-Pit (`Marvin`, `SM2` and `McLaughlin`), the reduction in size was more than 80% (variables and constraints in the Backward Stage compared to the original problem). In the remaining instance, `W23`, the reduction was around 23%.

Due to the successful reduction in the size of the problem, the problem was solved in a reasonable time. Except `Zuck medium`, which needed less than 5 hours, all of the instances were solved in less than 30 minutes wall-clock time. (See hardware specifications in section 5.1.)

6. Conclusions and further work

We have presented a number of heuristics to tackle the open-pit block scheduling problem. Our approach is based mainly on the reduction of the problem to be solved, that is, the size of the binary linear formulation.

The heuristics were applied to `MineLib`, a library of publicly available test problem instances. For nine of the eleven instances in the library, the gap obtained between the solution reported by our heuristic and the linear relaxation was lower than the gap currently known in the literature. While other references do not provide execution times for comparison, we consider that our running times are reasonable for application purposes. Further on, these can be improved for example by trying different levels of aggregation simultaneously on several computers, therefore reducing forward stage running time.

Furthermore, we believe that the approach described is very scalable in terms of problem size. Moreover, our approach is compatible with other algorithms described in the literature, so it can be easily combined to benefit other approaches, as well as extensions of the heuristics.

Regarding the combination with other approaches, the most promising is the one presented in Chicoisne et al. [8]: an `HReb+CMA` (Critical Multiplier Algorithm). This approach seems interesting, because it is not difficult to implement. Another approach, more difficult to code because of solver limitations, is the Bienstock & Zuckerberg [4] algorithm within the branching procedure.

There are also several possible extensions. One could be using more than one time-period at each iteration of the incremental heuristic. Another is to introduce information about the future when producing incremental solutions. For example using Lagrangian relaxation like in Cullenbine et al. [11] or Lambert and Newman [23].

Acknowledgments

J. Peypouquet was partly supported by FONDECYT grant 1140829, Basal Project CMM Universidad de Chile, Millenium Nucleus ICM/FIC P10-024F, Anillo Project ACT-1106, ECOS grant C13E03 and STIC AmSud Project OVIMINE.

References

[1] Alvarez, F., Amaya, J., Griewank, A., and Strogies, N. (2011). A continuous framework for open pit mine planning. *Mathematical Methods of Operations Research*, 73(1):29–54.

[2] Amankwah, H. (2011). *Mathematical Optimization Models and Methods for Open-Pit Mining*. PhD thesis, Linköping.

[3] Amaya, J., Espinoza, D., Goycoolea, M., Moreno, E., Prevost, T., and Rubio, E. (2009). A scalable approach to optimal block scheduling. In *Proceedings of APCOM*, pages 567–575.

[4] Bienstock, D. and Zuckerberg, M. (2010). Solving LP relaxations of large-scale precedence constrained problems. In *IPCO*, pages 1–14.

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

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

[7] Caccetta, L. and Hill, S. (2003). An application of branch and cut to open-pit mine scheduling. *Journal of Global Optimization*, 27:349–365.

[8] Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., and Rubio, E. (2012). A new algorithm for the open-pit mine production scheduling problem. *Operations Research*, 60(3):517–528.

[9] COCHILCO (2013). *Anuario de estadísticas del cobre y otros minerales: 1993-2012*. Recopilación de estudios, COCHILCO.

[10] CPLEX, I. I. (2013). Ibm ilog cplex optimization studio. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[11] Cullenbine, C., Wood, R., and Newman, A. (2011). A sliding time window heuristic for open pit mine block sequencing. *Optimization Letters*, 5(3):365–377.

[12] Dagdelen, K. and Akaike, A. (1999). A strategic production scheduling method for an open pit mine. In C. Dardano, M. Francisco, J. P., editor, *Proc. 28th Internat. Appl. Comput. Oper. Res. Mineral Indust. (APCOM) Sympos.*, pages 729 – 738, Littleton, CO. SME.

[13] Dagdelen, K. and Johnson, T. (1986). Optimum open pit mine production scheduling by lagrangian parameterization. In *Proc. 19th Internat. Appl. Comput. Oper. Res. Mineral Indust. (APCOM) Sympos.*, pages 127 – 141, Littleton, CO. SME.

[14] Delphos, M. P. L. (2013). Minelink library. http://delphos.dmi.uchile.cl.

[15] Espinoza, D., Goycoolea, M., Moreno, E., and Newman, A. (2013). Minelib: a library of open pit mining problems. *Annals of Operations Research*, 206(1):93–114.

[16] Fricke, C. (2006). *Applications of integer programming in open pit mining*. PhD thesis, University of Melbourne, Department of Mathematics and Statistics.

[17] Gaupp, M. (2008). *Methods for improving the tractability of the block sequencing problem for open pit mining*. PhD thesis, Colorado School of Mines.

[18] Gemcom (2011). Gemcom Whittle^{TM} Strategic Mine Planning software. http://www.gemcomsoftware.com/products/whittle.

[19] Gurobi Optimization, I. (2013). Gurobi optimizer reference manual. http://www.gurobi.com.

[20] Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

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

[22] Johnson, T. (1968). *Optimum open-pit mine production scheduling*. PhD thesis, Operations Research Department, University of California, Berkeley.

[23] Lambert, W. and Newman, A. (2014). Tailored lagrangian relaxation for the open pit block sequencing problem. *Annals of Operations Research*, 222(1):419–438.

[24] Lamghari, A., Dimitrakopoulos, R., and Ferland, J. (2014). A hybrid method based on linear programming and variable neighborhood descent for scheduling production in open-pit mines. *Journal of Global Optimization*, pages 1–28.

[25] Lerchs, H. and Grossman, H. (1965). Optimal design of open-pit mines. *Transactions C.I.M.*, 58:47–54.

[26] Matheron, G. (1975). Paramétrage de contours optimaux. Technical report, Note Geostatistics, Fontainebleau.

[27] Nemhauser, G. and Wolsey, L. (1988). *Integer and combinatorial optimization*, volume 18. Wiley New York.

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

[29] Osanloo, M., Gholamnejad, J., and Karimi, B. (2008). Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, 22(1):3–35.

[30] Picard, J. (1976). Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11):pp. 1268–1272.

[31] Ramazan, S., Dagdelen, K., and Johnson, T. (2005). Fundamental tree algorithm in optimising production scheduling for open pit mine design. *Mining Technology*, 114(1):45–54.

[32] Tabesh, M. and Askari-Nasab, H. (2011). Two-stage clustering algorithm for block aggregation in open pit mines. *Mining Technology*, 120(3):158–169.

[33] Zhang, M. (2006). Combining genetic algorithms and topological sort to optimize open-pit mine plans. *Proc. 15th Internat. Sympos. Mine Planning Equipment Selection (MPES), M. Cardu, R. Ciccu, E. Lovera, E. Michelotti, eds., FIORDO S.r.l., Torino, Italy*, pages 1234 – 1239.