A recursive time aggregation-disaggregation heuristic for the multidimensional and multiperiod precedence-constrained knapsack problem: an application to the open-pit mine block sequencing problem

Pierre Nancel-Penard[a], Nelson Morales[b], Fabien Cornillier[c,d,*]

[a]*Delphos Mine Planning Laboratory, Department of Mining Engineering, Faculty of Physical and Mathematics Sciences, University of Chile, Beauchef 850, Santiago, Chile; Advanced Mining Technology Center, University of Chile, Avda. Tupper 2007, Santiago, Chile*
[b]*Dept. of Civil, Geological and Mining Engineering, Polytechnique Montréal, Montréal, Canada*
[c]*Dept. of Industrial Engineering, Universidad de Ingenieria y Tecnologia – UTEC, Lima, Peru*
[d]*CIRRELT, Montreal, Canada*

Abstract

A recursive time aggregation-disaggregation (RAD) heuristic is proposed to solve large-scale multi-dimensional and multiperiod precedence-constrained knapsack problems (MMPKP) in which a profit is maximized by filling the knapsack in multiple periods while satisfying minimum and maximum resource consumption constraints per period as well as precedence constraints between items. An important strategic planning application of the MMPKP in the mining industry is the well-known open-pit mine block sequencing problem (BSP). In the BSP, a mine is modeled as a three-dimensional grid of blocks to determine a block extraction sequence that maximizes the net present value while satisfying constraints on the shape of the mine and resource consumption over time. Large real-life instances of this problem are difficult to solve, particularly with lower bounds on resource consumption. The advantage of the time aggregation-disaggregation heuristic over a rolling-horizon-based time decomposition is twofold: first, the entire horizon is considered for the resource consumption from the first aggregation; and second, only two-period subproblems have to be solved. This method is applied to a well-known integer programming model and a variant thereof in which blocks can be extracted in parts over multiple periods. Tests on benchmark instances show that near-optimal solutions for both of the models can be obtained for extremely large instances with up to 2,340,142 blocks and 10 periods.

Keywords: Heuristics, Integer programming, Time decomposition, Multidimensional and multiperiod precedence-constrained knapsack, Open pit mine scheduling

1. Introduction

The *multiperiod precedence-constrained knapsack problems* (MPPCKP), a generalization of the knapsack problem, has been introduced in Moreno et al. (2010) to solve a well-known scheduling problem in the mining industry: the open-pit mine production scheduling problem. In a more recent article by Samavati et al. (2017b), a multidimensional MPPCKP without minimum units of resource

*Corresponding author
Email addresses: pierre.nancel@amtc.cl (Pierre Nancel-Penard), nelson.morales@polymtl.ca (Nelson Morales), fcornillier@utec.edu.pe (Fabien Cornillier)

consumption constraints per period has been studied. In the open-pit mine production scheduling problem, profit is maximized by selecting blocks for extraction at each period while satisfying resource consumption constraints for each period as well as precedence constraints between items. Multiple resources such as extracting and processing resources are considered and any constraint corresponds to a knapsack constraint on a resource and a period. In this article, we propose a new recursive time aggregation-disaggregation (RAD) heuristic to solve the *multidimensional and multi-period precedence-constrained knapsack problem* (MMPKP), a multidimensional MPPCKP in which minimum units of resource consumption constraints per period are considered. In this problem, the value of a knapsack is maximized by filling it in multiple periods while satisfying minimum and maximum units of resource consumption per period and precedence constraints between items.

The proposed RAD heuristic is used to solve an important production process scheduling problem in an open-pit mine to maximize the net profit of the mining operation. In open-pit mines, minerals are extracted by moving material from the surface and the extracted materials are processed by plants, stockpiled, or placed in waste dumps depending on the profit potential. To determine the part of the terrain to be extracted, an open-pit mine is represented as a three-dimensional array of equisized blocks, known as the *block model*. Geostatistical methods are applied to the results obtained from exploratory borehole samples to define the geological attributes and to determine the size of the blocks. The mineral content (grade) of a block is one of these geological attributes. The value of a block depends on its geological attributes, its destination, and other parameters and costs related to the operations. Because the opportunity cost affects the net present value of a block, this value depends on the moment of its extraction and processing, which can be determined after discretizing the planning horizon into periods.

The open-pit mine *block sequencing problem* (BSP) seeks to determine the set of blocks to be extracted and the extraction time for each block such as to maximize the net present value while satisfying operational and geomechanical constraints. The main constraints are the slope angle constraints that ensure the stability of the open-pit walls by restricting the slope angles to a maximum limit. These constraints also model the spatial precedence among the blocks by stating that a block cannot be extracted before extracting its overlying blocks. The operational constraints are related either to the amount of material to be extracted and processed at each period, or to the time required to transport and process it. The amount of the material to be extracted and processed, and the time required to transport and process it, are bounded by lower and upper limits that control the flow of blocks between the mine and the destinations. When multiple resources are considered (e.g. mining and processing), this problem corresponds to the MMPKP with time-dependent profits.

Practically, the minimum resource consumption constraints are used to bound the variations of the quantity of material to extract and process among consecutive periods. By smoothing the intensity of the operations, these constraints allow controlling the quantity of the mining equipment required to transport and process the extracted materials. Furthermore, these lower bounds reduce the setup costs incurred whenever operations are stopped and restarted. As stated by Cullenbine et al. (2011), "*contractual agreements and the chemical and physical properties of the milling process may also necessitate positive lower bounds on production and processing rates*". They propose an experiment to analyze the influence of minimum resource consumption constraints on the complexity of the BSP and show that it takes about twenty times longer to find optimal solutions when considering minimum extraction capacities.

The origin of the additional complexity is twofold: the larger number of knapsack constraints for each period and the potential inter-resource conflicts between lower bounds on some resource consumption (e.g. extraction and processing) and upper bounds on other resources (Samavati et al., 2017a).

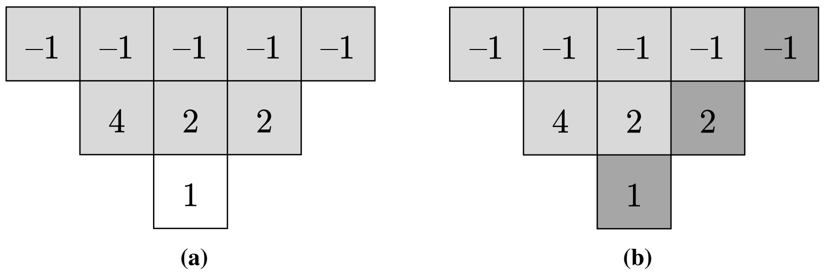

Fig. 1. A 2-dimensional instance with a minimum processing resource consumption of 2 blocks per period and a maximum mining capacity of 8 blocks (numbers represent economic values and grey scale extraction periods) (a) A sliding time window heuristic that only considers one period at a time will identify blocks in grey to be extracted in period 1; thus, it will be unable to produce a feasible solution. (b) An optimal solution.

Figure 1 presents a 2-dimensional example of conflict between a maximum material-extraction capacity (hereafter called maximum *mining capacity*) and a minimum processing resource consumption constraint. A planning horizon of two periods and a discount rate of 10% per period are considered in this example. All blocks have weight equal to 1 and values, in monetary units, represented by the numbers inside the blocks. The minimum processing capacity is 2 per period and the maximum capacity is 8. A sliding time window heuristic that only considers one period at a time will extract the 8 blocks on the top (light grey in Figure 1a) during period 1, leaving only one block for processing in period 2 (white in Figure 1a), i.e., will not find a feasible solution. Conversely, the optimal solution extracts 6 blocks in period 1 (light grey in Figure 1b) and 3 in period 2 (dark grey in Figure 1b) and has a value of 3.8. This example demonstrates that greedy heuristic approaches that only consider a few periods at each iteration may unveil such conflicts and fail to identify a feasible solution.

As for this example and Samavati et al. (2017a), in this article we address the problem in which any extracted block is always processed to maximize its profit or put into waste dumps whenever its ore concentration is below a fixed cutoff grade, i.e., below the minimum ore concentration required to process the block.

We aim to develop a recursive time aggregation-disaggregation (RAD) heuristic to solve large instances of the BSP with minimum extraction and processing constraints. The proposed heuristic is applied to a well-known integer programming model and a variant proposed by Vossen et al. (2016). In the variant version, a mixed-integer programming model (MIP) is used to solve problems in which blocks can be extracted in parts over multiple periods while satisfying the slope angle constraints. They report improvements of between 1% and 1.9% of the net present value compared with the integer programming model. As stated by Vossen et al. (2016) "*some blocks at a mine's surface may be partial at the beginning of period 1 because the undisturbed surface is uneven or because some partial extraction has already taken place.*" Furthermore, extracting entire blocks limits the way in which the discretized profile of the pit approximates the continuous one. Allowing partially mined blocks may offer slope angles closer to the maximum values and a better approximation of the varying slope angles for different depths or sections of the mine. Because the edges of the shapes generated by the block model are smoothed during the design process with the help of CAD software, the solutions of either the integer programming model or the variant proposed by Vossen et al. (2016) could be used as inputs for the design process. Moreover, Morales et al. (2015) compared solutions obtained from software that consider continuous slopes with integer solutions and show a marked difference in pit geometries, particularly for the first periods. Parra et al. (2018) also demonstrate that slope angles may have a considerable impact on the economic value of the first periods. This follows from the fact

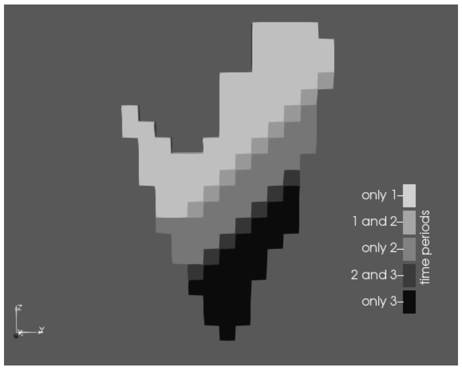

Fig. 2. Section view of the Newman1 instance of the Minelib library (Espinoza et al., 2013) showing the extraction periods for the fractional block sequencing problem (FBSP) model in which blocks can be partially extracted at each period.

that steeper angles require less waste to be extracted to comply with the slope.

Figure 2 shows a section view of the extracted periods of a solution for the Newman1 scheduling instance of the Minelib library (Espinoza et al., 2013) resulting from the MIP. In this example, whenever a block is partially extracted at period t, its remaining fraction is extracted at period $t + 1$.

In this article, we propose new instances based on the library Minelib (Espinoza et al. 2013) introducing minimum resource consumption constraints on material extraction and processing.

The remainder of this article is organized as follows. A literature review is provided in Section 2. Two versions of the problem are modeled in Section 3. The RAD heuristic is described in Section 4, and the computational results are presented in Section 5. The conclusions are drawn in Section 6.

2. Related works

In this section, we present relevant articles on strategic open-pit planning problems and the studies that more specifically consider the difficult minimum resource consumption constraints.

Lerchs & Grossman (1965) were the first to propose an algorithm to solve the *final pit problem* also known as the *ultimate pit problem*. This problem is a simplified version of the BSP in which a value is allotted to each block without considering the extraction period, and where only slope constraints are modeled. Picard (1976) demonstrated that the final pit problem is equivalent to the maximum closure problem, which can be formulated as a minimum cut problem. Hochbaum proposed an efficient polynomial algorithm based on a maximum pseudoflow approach to tackle the final pit problem (Hochbaum & Chen, 2000; Hochbaum, 2008; Hochbaum & Orlin, 2012). Lerchs & Grossman (1965) also proposed an approach to schedule block extraction by solving a series of final pit problems while varying the block revenues to generate a sequence of nested pits. Commercial software programs, including Whittle (Gemcom, 2021), still implement this algorithm. A review of open-pit mine designs with nested pits can be found in Meagher et al. (2014), which focus on the inconsistent sizes that may be observed between successive nested pits, namely, the *gap problem*.

Johnson (1968, 1969) presented a linear programming formulation for a multidestination block scheduling problem under slope, capacity, and blending constraints. Each extracted block containing ore had to be assigned to a processing plant to maximize its value. Since the real instances of this problem are difficult to solve, many algorithms were developed such as the Lagrangian relaxation method proposed by Dagdelen (1986) to solve a problem with fixed thresholds of mineral concentrations (fixed cutoff grades), and where only upper bounds on the capacities are considered.

Caccetta & Hill (2003) proposed a specific branch-and-cut algorithm, which only considered the upper bounds on resource consumption. The algorithm is based on a "by" formulation, in which the extraction time of a block is modeled as a binary variable equal to one if the block is extracted *by* a given period and equal to zero otherwise. As we can switch from an "at" formulation to a "by" formulation via variable substitution, both of the models are equivalent. However, despite this equivalence, the numerical experiments performed by Lambert et al. (2014) demonstrate that improved upper bounds are obtained with the "by" formulation after applying certain variable-reduction operations at the root node of the branch-and-cut algorithm.

As it is often challenging to solve large real instances, Boland et al. (2009) proposed to aggregate blocks into bins, where precedence constraints only applied between bins. By reducing the problem size, this technique helps expedite the resolution time.

Bienstock & Zuckerberg (2009, 2010) achieved breakthrough results using a Lagrangian relaxation-based method to solve the linear relaxation of the BSP. The method substantially decreases the resolution time compared with the standard linear programming solvers.

Moreno et al. (2010) presented the Critical Multiplier Algorithm to solve the LP-relaxation of the multiperiod precedence-constrained knapsack problem as a sequence of single-time period problems applied to the open-pit production scheduling problem.

Bley et al. (2010) used a "by" formulation with upper bounds on resource consumption. Additional clique and cover cuts based on the capacity constraints strengthen the formulation.

Chicoisne et al. (2012) developed a customized algorithm for the linear programming relaxation of the problem with only one destination and one capacity constraint per period. They extended the heuristic method proposed by Gershon (1987) based on expected extraction times to perform a topological sorting (TopoSort) on the blocks. A feasible integer solution is obtained using this procedure for a more general problem with multiple capacity constraints per period.

Espinoza et al. (2013) proposed the Minelib library of standardized instances including real and simulated datasets for three different open-pit mining problems, the final pit problem and the following two variants of the open-pit block-scheduling problem: the fixed cutoff grade problem, referred to as *constrained pit limit problem* (CPIT), where any extracted block is processed only if its ore concentration is equal or greater than the specified minimum grade, and the multidestination optimization problem with blending constraints, referred to as *precedence constrained production scheduling problem* (PCPSP). The PCPSP formulation is a mixed-integer programming model in which different parts of the same extracted block can be sent to different destinations within the same period. Note that, for the CPIT and PCPSP scheduling instances of the Minelib library that do not include positive lower bounds on resource consumption, the null solution is a trivial feasible solution, because it complies with the upper capacities and does not violate any lower bound.

Jélvez et al. (2016) proposed a spatial aggregation and disaggregation heuristic that improved some of the published results of the Minelib library for the fixed cutoff grade problem. The best average relative gaps were obtained with this method for the largest five instances. Others, including Lamghari et al. (2014), Liu & Kozan (2016), and Samavati et al. (2017b,c), proposed heuristics that improve the results for some of the fixed cutoff grade Minelib scheduling instances without minimum resource consumption constraints. In particular, Liu & Kozan (2016) obtained good solutions developing two topological ordering-based algorithms relying on network flow graph and conjunctive graph theory.

Jélvez et al. (2020) presented a hybrid heuristic algorithm that combined a rolling-horizon decomposition algorithm with a block-preselection procedure using the concept of expected extraction times proposed by Gershon (1987). A satisfactory feasible solution was reported for the instance W23 of the Minelib library with blending constraints, an instance for which no feasible solution has been previously reported.

More detailed reviews on general mine planning models and algorithms can be found in Osanloo et al. (2008), Newman et al. (2010) and Zeng et al. (2021).

As stated in the introduction, lower bounds on resource consumption have practical importance in the mining operations but make the problem drastically harder to solve with linear programming models. The articles reviewed in the next paragraphs more specifically address open-pit strategic planning problems with positive lower bounds on resource consumption.

Ramazan et al. (2005) is one of the first authors to specifically solve open-pit planning problems with minimum processing consumption constraints and linear programming for a fixed cutoff grade model. The proposed method, named the *fundamental tree algorithm*, is based on block aggregation to lower the number of variables and constraints of the original problem.

Gaupp (2008) presents a branch-and-cut algorithm with reduction techniques to decrease the number of variables and heuristics based on Lagrangian relaxation to generate feasible solutions for a fixed cutoff grade model. The earliest and latest possible extraction times available for each block are computed to generate cutting planes. These methods are used to solve instances of up to 10,819 blocks and 6 periods.

Askari-Nasab et al. (2011) compared several mixed-integer programming formulations of a problem with a fixed cutoff grade, lower and upper limits on capacity, and blending constraints. They proposed to aggregate blocks into clusters based on the block attributes to solve instances with up to 2,598 blocks and 12 periods. In this study, lower bounds are only considered on the grade blending constraints.

To avoid having a single block at the bottom of the pit, Cullenbine et al. (2011) added horizontal precedence constraints to a "by" formulation of the problem with a fixed cutoff grade. They implemented a rolling time window heuristic that considers the complete horizon while relaxing the integrality constraints for periods beyond the rolling time window. Using this method, instances with lower bounds on resource consumption with up to 25,620 blocks and 15 periods could be solved.

Lambert & Newman (2014) solved the problem formulated as in Cullenbine et al. (2011), without the horizontal precedence constraint. They proposed to identify a sufficient set of blocks to satisfy the minimum resource consumption constraints. This set of blocks is obtained by solving a series of final pit problems while varying the ore price. A Lagrangian relaxation method was proposed, which uses the information obtained while generating the initial solution to select a dualization scheme for some resource constraints. The performance of this algorithm evaluated for instances with up to 25,000 blocks, 10 periods, minimum and maximum use of the mining and processing resources, shows that incorporating lower bounds on resource consumption may increase the computational time by over one order of magnitude.

Vossen et al. (2016) proposed a "hierarchical" Benders decomposition methodology to solve a BSP variant wherein blocks could be extracted in parts over multiple periods. This new methodology generalizes a nested benders decomposition using cumulative variables and time-aggregated resource constraints that consider lower bounds on resource consumption. They solved instances with up to 25,620 blocks and 20 periods.

Samavati et al. (2017a) implemented a local branching procedure that started from a feasible solution obtained with a greedy method to solve a BSP "at" formulation. The blocks are iteratively sorted in a random order to feed a topological sort heuristic similar to the Toposort algorithm proposed by Chicoisne et al. (2012), until obtaining a feasible solution that satisfies the minimum resource consumption constraints. Instances with up to 50,383 blocks, 10 periods, minimum use of the mining and processing resource were solved; unfortunately, the data sets have not been publicly released.

A recent study by Letelier et al. (2020) exposes direct block scheduling models for which preprocessing, cutting plane techniques, heuristic approaches, and a customized branch-and-bound are proposed to obtain better bounds and feasible solutions. Instances with minimum use of the processing resource of up to 2,340,142 blocks and 20 periods are solved; unfortunately, the minimum processing constraints have not been publicly released.

3. Mathematical formulations

In this section, the following two problems are formulated: the *block sequencing problem* (BSP) in which a block should be either totally extracted or not extracted at all, and the *fractional block sequencing problem* (FBSP), based on the model proposed by Vossen et al. (2016), in which a block can be extracted in parts over multiple periods. The formulations presented here are "by" formulations where variables represent blocks that are extracted *by* a specific period. A comparison between the "at" and "by" formulations of the BSP can be found in Lambert et al. (2014).

3.1. Formulation of the block sequencing problem

Let $\mathcal{B}$ denote the set of blocks of the block model, and $\mathcal{R}$ denote the set of renewable operational resources required to extract and process the blocks (i.e., the resources renewed at each period including workforce and equipment, as opposed to consumable resources). The profit p_{bt} is obtained by extracting and processing block $b \in \mathcal{B}$ at period $t \in \mathcal{T}$, where $\mathcal{T} = \{1, \ldots, T\}$ denotes the set of periods. An amount q_{br} of operational resource $r \in \mathcal{R}$ is required to extract and process block b, while a minimum of L_{rt} or a maximum of U_{rt} of resource $r \in \mathcal{R}$ must be used at each period $t \in \mathcal{T}$. The open-pit mine block sequencing problem (BSP) is the problem of maximizing the total profit by determining the subset of blocks to be scheduled and extracted while satisfying the minimum use L_{rt} and the maximum availability U_{rt} of resource r at period t. The precedence constraints are defined by the set $\mathcal{P}$ of the slope precedence arcs. This set contains the arcs (b, b') such that block b' is at a level above b and must be mined before b to ensure the stability of walls. We set the binary variable x_{bt} equal to 1 if block b is extracted *by* period t, and 0 otherwise. Note that $x_{bt} - x_{b,t-1}$ specifies whether block b is mined at time t. The BSP can then be formulated as follows:

$$
\begin{aligned}
&\text{(BSP)} \ \max \sum_{t\in\mathcal{T}, b\in\mathcal{B}} p_{bt}(x_{bt} - x_{b,t-1}) && && (1)\\
&x_{bt} \le x_{b't} && \forall\, (b,b') \in \mathcal{P}, t \in \mathcal{T} && (2)\\
&x_{bt} \ge x_{b,t-1} && \forall b \in \mathcal{B}, t \in \mathcal{T} && (3)\\
&\sum_{b\in\mathcal{B}} q_{br}(x_{bt} - x_{b,t-1}) \le U_{rt} && \forall r \in \mathcal{R}, t \in \mathcal{T} && (4)\\
&\sum_{b\in\mathcal{B}} q_{br}(x_{bt} - x_{b,t-1}) \ge L_{rt} && \forall r \in \mathcal{R}, t \in \mathcal{T} && (5)\\
&x_{bt} \in \{0,1\} && \forall b \in \mathcal{B}, t \in \mathcal{T} && (6)\\
&x_{b0} = 0 && \forall b \in \mathcal{B}. && (7)
\end{aligned}
$$

The objective function (1) maximizes the total profit. Constraints (2) correspond to the precedence constraints given by the slope specifications to ensure the wall's stability. Under constraints (3) a block b extracted by period $t-1$ remains extracted at period $t \le T$ as a block can be extracted only once. Constraints (4) and (5) state that the maximum and minimum resource consumption constraints should be satisfied at each period. Constraints (6) and (7) reflect the nature of the variables.

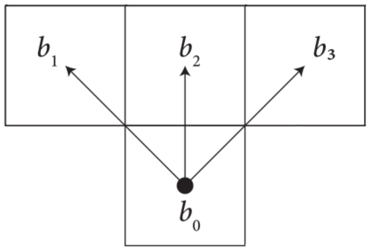

Fig. 3. Example of section view with vertical and oblique precedence arcs

3.2. Formulation of the fractional block sequencing problem

Vossen et al. (2016) proposed a BSP variant in which a block could be extracted in parts over multiple periods, and proved that any feasible solution of this model satisfies the slope precedence constraints.

Let $\mathcal{P}'$ denote the set of vertical precedence arcs, and $\mathcal{P}''$ the set of oblique precedence arcs. In Figure 3, block b_2 is the vertical spatial predecessor of block b_0, i.e., $(b_0, b_2) \in \mathcal{P}'$, and blocks b_1 and b_3 are two oblique spatial predecessors of block b_0, i.e., $(b_0, b_1) \in \mathcal{P}''$ and $(b_0, b_3) \in \mathcal{P}''$. Let us define a continuous decision variable $y_{bt} \in [0, 1]$ equal to the fraction of block b extracted *by* period t, and a binary variable x_{bt} equal to 1 if block b is entirely extracted *by* period t, and 0 otherwise. The FBSP can be formulated as follows:

$$\text{(FBSP)} \ \max \sum_{t\in\mathcal{T}, b\in\mathcal{B}} p_{bt}(y_{bt} - y_{b,t-1}) \tag{8}$$

$$y_{bt} \leq y_{b''t} \qquad \forall\, (b, b'') \in \mathcal{P}'', t \in \mathcal{T} \tag{9}$$

$$y_{bt} \leq x_{b't} \qquad \forall\, (b, b') \in \mathcal{P}', t \in \mathcal{T} \tag{10}$$

$$y_{bt} \geq x_{bt} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{11}$$

$$y_{bt} \geq y_{b,t-1} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{12}$$

$$y_{b0} = 0 \qquad \forall b \in \mathcal{B} \tag{13}$$

$$\sum_{b\in\mathcal{B}} q_{br}\,(y_{bt} - y_{b,t-1}) \leq U_{rt} \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{14}$$

$$\sum_{b\in\mathcal{B}} q_{br}\,(y_{bt} - y_{b,t-1}) \geq L_{rt} \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{15}$$

$$y_{bt} \in [0, 1] \qquad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{16}$$

$$x_{bt} \in \{0, 1\} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T}. \tag{17}$$

Objective function (8) maximizes the total profit of the blocks, whether totally or partially extracted. Under constraints (9), the fraction of a block b extracted by time t cannot exceed the fraction of any of its oblique predecessors extracted by time t. Constraints (10) ensure that block b cannot be extracted before the complete extraction of block b' located directly above block b, i.e., block b' is the vertical predecessor of block b. Constraints (11) reflect the relation between variables y_{bt} and x_{bt} for any block b at period t. Constraints (12) ensure that if a fraction of a block b is extracted by period $t-1$, at least the same fraction remains extracted by period $t \leq T$. Constraints (13) state that no block can be extracted before the beginning of the first period. Constraints (14) and (15) state that the maximum and minimum resource consumption should be satisfied at each period. Constraints (16) and (17) reflect the nature of the variables.

As the LP-relaxations of the FBSP and the BSP only differ in terms of the vertical and oblique predecessors, they are equivalent; therefore, an optimal solution of one of these relaxed models can be used as an upper bound for the non-relaxed model.

4. Algorithm for the MMPKP

In the above formulations, the number of variables is generally a serious challenge in practice. In this section, we propose two size-reduction procedures:

1. The first approach to decrease the number of variables is to decompose the problem into a series of easier-to-solve subproblems. In this section, we propose a recursive time decomposition algorithm in which the periods are aggregated, then disaggregated. This algorithm considers the entire horizon from the first subproblem and thereby avoids using more resources than necessary in the first periods to help to prevent the risk of having an insufficient activity in the subsequent periods. Another advantage of this aggregation-disaggregation heuristic is that it generates only two-period subproblems significantly more tractable for the largest instances.

2. The second classical reduction procedure is to only consider the blocks of the final pit, i.e., the blocks to extract to maximize the economic value while satisfying only the precedence constraints. Since strictly positive minimum resource consumption constraints are considered in this study, the blocks of the final pit may be insufficient to obtain a feasible solution. In such a case, more blocks have to be added to the final pit. In this section, we describe an integer programming model to extend the final pit in an attempt to obtain a feasible solution whenever the overall minimum resource consumption constraints are not satisfied.

4.1. Recursive time aggregation-disaggregation (RAD) heuristic

Let us refer to a *metaperiod* as any subset of consecutive *periods*, and an *elementary metaperiod* as any metaperiod of only one *period*. We propose to solve the BSP and FBSP models by solving a series of two-metaperiod subproblems. In the RAD heuristic, the initial problem is solved as a two-metaperiod problem after aggregating all its periods into two metaperiods as well as their corresponding resource constraints. The periods of each metaperiod are in turn aggregated into two other metaperiods to form a new two-metaperiod subproblem. This procedure is recursively repeated until solving subproblems with only two *elementary* metaperiods, i.e., with only two *periods*. The complete decomposition results in a binary tree structure in which each node corresponds to a two-metaperiod subproblem. All two-metaperiod subproblems of the binary tree are solved with a branch-and-cut procedure to determine the set of blocks (or parts of blocks in the case of the FBSP) to extract within each of their metaperiods. Note that by solving a subproblem, the sole metaperiod during which a block should be extracted is known. Its exact extraction period is revealed only after solving its leaf subproblems. We also point out that the resource constraints aggregated in each metaperiod are recovered as we go down the tree. Indeed, the original constraints have to be satisfied whenever a metaperiod corresponds to a real period.

Figure 4 shows an example of a binary tree generated by the decomposition for an instance of 10 periods. The root node corresponds to the first decomposition of the original problem transformed into a two-metaperiod problem by aggregating all its periods into two metaperiods $\{1, 2, \ldots, 5\}$ and $\{6, 7, \ldots, 10\}$. After solving the two-metaperiod problem P, we obtain two sets of blocks to be extracted: a set $\mathcal{B}_1$ of blocks to be extracted during periods $\{1, 2, \ldots, 5\}$, and a set $\mathcal{B}_2$ of blocks to be extracted during periods $\{6, 7, \ldots, 10\}$. To determine how the extraction periods of the blocks in $\mathcal{B}_1$ are distributed among periods $\{1, 2, \ldots, 5\}$, a new two-metaperiod subproblem P_1 is solved after aggregating the periods of the first metaperiod of the problem P into two metaperiods $\{1, 2\}$ and $\{3, 4, 5\}$. Consequently, we obtain the sets $\mathcal{B}_{1,1}$ and $\mathcal{B}_{1,2}$ of blocks to be extracted during periods $\{1, 2\}$ and $\{3, 4, 5\}$, respectively. A new two-metaperiod subproblem P_2 is solved in the same way after aggregating the periods of the second metaperiod of the problem P into two metaperiods $\{6, 7\}$ and $\{8, 9, 10\}$ to obtain the sets $\mathcal{B}_{2,1}$ and $\mathcal{B}_{2,2}$ of blocks to be extracted during periods $\{6, 7\}$ and $\{8, 9, 10\}$, respectively. When a metaperiod only contains *elementary* metaperiods, i.e., only *periods*, it does not require to be decomposed anymore. Note that to balance the computation time of each subtree a balanced binary tree can be obtained by setting the number of periods in each metaperiod of a two-metaperiod subproblem such that it differs by at most one.

For instance, in Figure 4, the root subproblem P is solved to determine the blocks to extract within each of the two metaperiods $\{1, 2, \ldots, 5\}$ and $\{6, 7, \ldots, 10\}$. Each of these metaperiods is in turn partitioned into two new metaperiods: $\{1, 2, \ldots, 5\}$ into metaperiods $\{1, 2\}$ and $\{3, 4, 5\}$, and $\{6, 7, \ldots, 10\}$ into metaperiods $\{6, 7\}$ and $\{8, 9, 10\}$. Note that because of the "*by*" formulation, if a block b is extracted within the metaperiod $\{1, 2, \ldots, 5\}$ in the solution of the subproblem P, then the same block should also be extracted *by* the second metaperiod of the subproblem $P1$, i.e. the metaperiod $\{3, 4, 5\}$. Formally, $x^{P}_{b,1} = 1$ implies $x^{P1}_{b,2} = 1$, where $x^{P}_{b,m}$ denotes a binary variable equal to 1 if block b is extracted by metaperiod $m \in \{1, 2\}$ in a subproblem P, and 0 otherwise. Moreover, when the FBSP formulation is used, if a fraction λ of block b is extracted by the metaperiod $\{1, 2, \ldots, 5\}$ in the solution of the subproblem P, the same fraction λ should be extracted *by* the second metaperiod of the subproblem $P1$, i.e., metaperiod $\{3, 4, 5\}$. Formally, if we denote by $y^{P}_{b,m} \in [0, 1]$ the fraction of block b extracted by metaperiod $m \in \{1, 2\}$ of a subproblem P, then $y^{P}_{b,1} = \lambda$ implies $y^{P1}_{b,2} = \lambda$. These implications are essential to enforce the solution of the subproblems in their children.

The solution of the BSP corresponds to the blocks to be extracted within each of the elementary metaperiods in the binary tree (in bold in Figure 4).

Illustration of the RAD heuristic. Consider an instance of 4 periods. Figure 5 shows the corresponding binary tree decomposition into subproblems with their respective metaperiods and Figure 6 shows the iterations of the proposed heuristic. The original problem is decomposed into the two metaperiods $\{1, 2\}$ and $\{3, 4\}$. The resulting two-period problem P is then solved to obtain two nested pits that correspond to these two metaperiods, respectively (Figure 6a). A two-metaperiod problem $P1$ is solved to obtain the blocks to be extracted at period 1 or 2 (Figure 6b). Subsequently, a two-period problem $P2$ is solved to obtain the blocks to be extracted at period 3 or 4 (Figure 6c). In each figure, the dotted line represents the solution of the solved two-metaperiod problem solved. The solutions obtained for the subproblems $P1$ and $P2$ yield the global solution.

4.2. *Consolidating the upper and lower resource consumption constraints*

As periods are aggregated into metaperiods, the resource capacities associated with each period of a metaperiod need to be consolidated. Let $U_{r,m}$ and $L_{r,m}$ denote the consolidated upper and lower capacities of resource r for a metaperiod m. These consolidated capacities are computed as the sum of the maximum resource r available at metaperiod m and the sum of the minimum use of resource r at metaperiod m, respectively, over all periods of the metaperiod m, i.e., $U_{r,m} = \sum_{t \in m} U_{r,t}$ and $L_{r,m} = \sum_{t \in m} L_{r,t}$.

Using the BSP formulation in which all the variables are binary and the blocks cannot be extracted in parts over multiple periods, more feasibility issues might arise related to the upper and lower re-

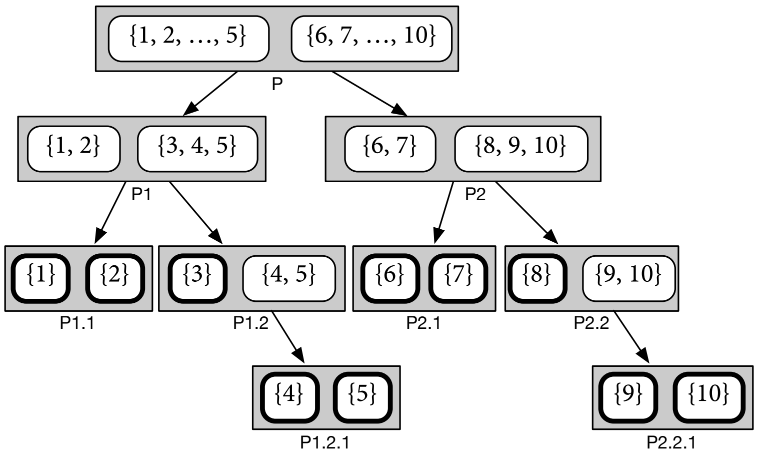

Fig. 4. Resulting binary tree of subproblems for the same instance of 10 periods. The subproblems are represented with grey rectangles, and the metaperiods are depicted using round-edged rectangles

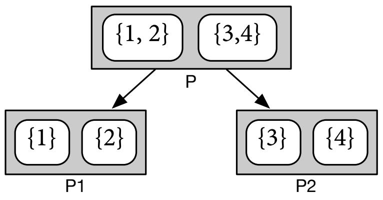

Fig. 5. Resulting binary tree of subproblems for an instance of 4 periods.

source consumption constraints compared with the FBSP formulation which offers more flexibility by allowing blocks to be partially extracted.

Illustration of potential issues related to the upper capacity constraints. Consider an instance of 4 periods with the block model shown in Figure 7. The problem is decomposed as shown in Figure 5. The final pit (in grey in Figure 7) contains 18 blocks to be extracted over 4 periods, i.e., $\mathcal{T} = \{1, 2, 3, 4\}$. For the sake of simplicity, only one resource is considered with constant minimum and maximum mining consumption of $L_{1,t} = 2.0$ and $U_{1,t} = 4.5$ blocks per period $t \in \mathcal{T}$, respectively. In Figure 7, the blocks are represented with their respective undiscounted values, and the present values are computed with a 10% discount rate. Moreover, any extracted block with a positive value is processed.

The horizon of subproblem P is decomposed into two metaperiods $\{1, 2\}$ and $\{3, 4\}$. For both of the metaperiods, the consolidated lower and upper mining capacities are 4 and 9 blocks, respectively. The optimal scheduling of subproblem P is shown in Figure 8 wherein the blocks are represented with their respective extraction metaperiods: a total of 9 blocks are extracted in the first metaperiod $\{1, 2\}$ and 9 blocks in the second metaperiod $\{3, 4\}$.

The metaperiods of subproblem $P1$ are the two elementary metaperiods $\{1\}$ and $\{2\}$. Using the BSP model, all the 9 blocks extracted within the first metaperiod of subproblem P have to be extracted *by* the second metaperiod of $P1$, and we have therefore $x^{P1}_{b,2} = 1$ for these blocks. However, since no block can be extracted in part in the BSP model, only 4 blocks can be extracted at each elementary metaperiod, over the 4.5 blocks allowed by the maximum mining capacity per period (i.e., $U_{1,t}$), i.e., up to 8 blocks can be extracted in each one of the metaperiods $\{1, 2\}$ and $\{3, 4\}$ rather than the 9 blocks per metaperiod scheduled while solving subproblem P. The optimal extraction scheduling for subproblem P makes the subproblem $P1$ unfeasible.

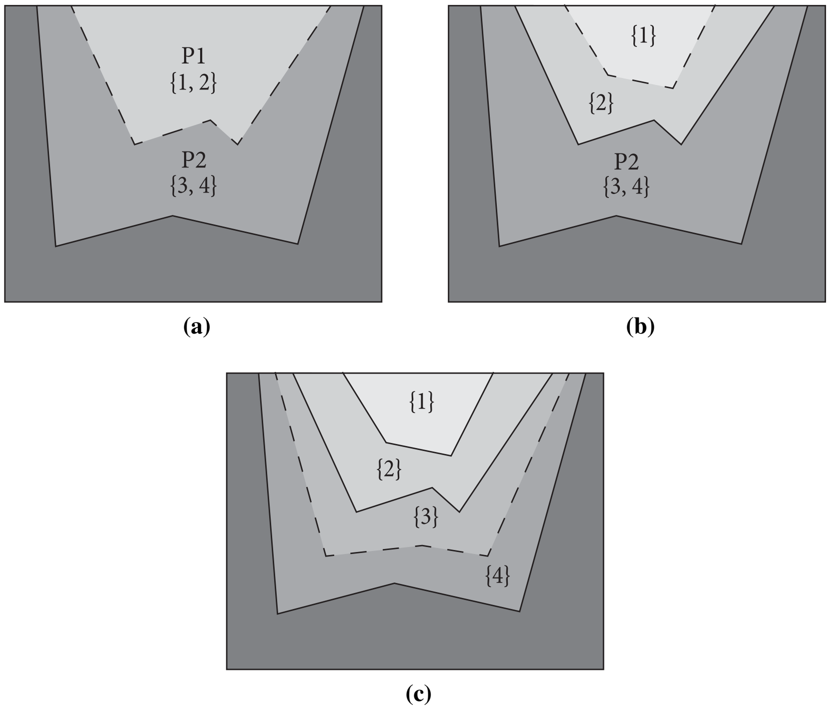

Fig. 6. Illustration of the RAD heuristic for an instance of 4 periods. Figure (a) shows the result of the first subproblem P solved as a two-metaperiod problem into $P1$ and $P2$ after aggregating all the periods into two metaperiods $\{1, 2\}$ and $\{3, 4\}$. Figures (b) and (c) show the respective results of subproblem P_1 and P_2 solved as other two-metaperiod problems. As $P1$ and P_2 only contain elementary metaperiods, no more decomposition is needed.

−1	−1	−1	−1	−1	−1	−1	−1
−1	−1	−1	−1	−1	−1	−1	−1
−1	−1	9	3	3	3	−1	−1

Fig. 7. A 2-dimensional illustration. Numbers represent the economic value of the blocks. Blocks in grey correspond to the final pit.

1	1	1	1	1	2	2	2
	1	1	1	2	2	2	
		1	2	2	2		

Fig. 8. An optimal solution for the BSP and FBSP models without delta adjustment. Numbers and grey scale represent extraction *metaperiods* of blocks. (see Figure 7 for the economic value of the blocks)

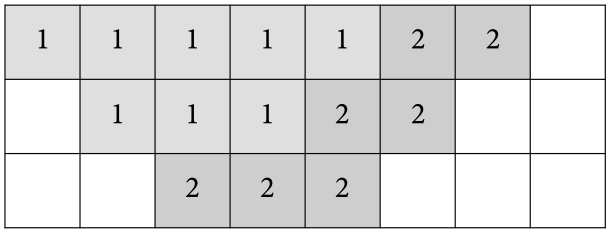

Fig. 9. An optimal solution for the BSP model with delta adjustment. Numbers and grey scale represent extraction *metaperiods* of blocks. (see Figure 7 for the economic value of the blocks)

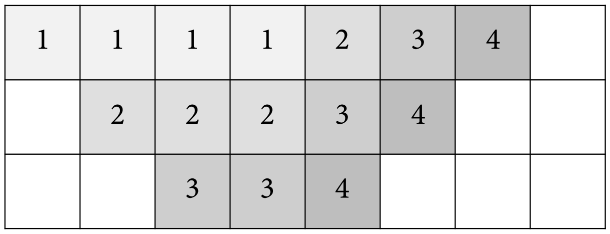

Fig. 10. A solution for the BSP model with delta adjustment. Numbers and grey scale represent extraction *periods* of blocks. The value of this solution is equal to 1.1. (see Figure 7 for the economic value of the blocks)

The probability to face this issue arising with the BSP model increases with the height of the metaperiod in the binary tree, i.e. the length of the longest downward path from the metaperiod to a leaf, which also depends on the size of the binary tree. Indeed, if we denote by $|m|$ the number of elementary periods in a metaperiod m, the height of the metaperiod m is computed as $h_m = \lceil log_2(|m|) \rceil - 1$. Therefore, to lower the probability of such an issue, we propose to solve the subproblems with more restrictive consolidated capacities $U'_{r,m} = U_{r,m} - \delta^U_{r,h_m}$ and $L'_{r,m} = L_{r,m} + \delta^L_{r,h_m}$ in both of the metaperiods $m = 1$ and $m = 2$, while the adjustment values $\delta^U_{r,h_m} > 0$ and $\delta^L_{r,h_m} > 0$ depend on the height h_m.

Let us solve subproblem P of an example wherein $\delta^U_{1,h_{m_1}} = \delta^U_{1,h_{m_2}} = 1$ for the metaperiods $m_1 = \{1, 2\}$ and $m_2 = \{3, 4\}$. The adjusted maximum capacities are $U'_{1,m_1} = U'_{1,m_2} = 8$. In the optimal scheduling, we observe that only 8 blocks are extracted in each metaperiod (Figure 9), rather than 9 when the maximum capacity is not adjusted. Subproblems $P1$ and $P2$ are feasible, and a feasible global solution is obtained with an objective value of 1.1 (Figure 10).

As a block can be partially extracted with the FBSP model, there is no need for more restrictive capacity constraints. In the 2-dimensional example, 4.5 blocks are extracted within each period of subproblems $P1$ and $P2$ (see Figure 11), and the optimal value is equal to 2.7.

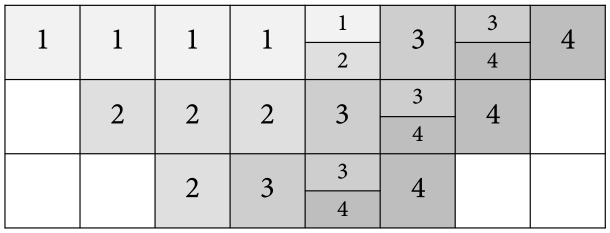

Fig. 11. A solution for the FBSP model. Numbers and grey scale represent extraction periods of *blocks or parts of blocks*. The value of this solution is equal to 2.7. (see Figure 7 for the economic value of the blocks)

4.3. *Final pit extension*

The final pit can be extended whenever the sum of resource capacities to extract is below the sum over the time horizon of the minimum resource consumption for at least one resource. Lambert & Newman (2014) proposed a two-stage extension procedure. In the first stage, the ore price is increased to generate a larger pit containing a sufficient amount of ore to only satisfy the total processing requirements for the time horizon. However, increasing the ore price is not always sufficient to generate a feasible larger pit. When insufficient extracted blocks are obtained by the first expansion stage to satisfy the overall minimum resource consumption, they propose a second expansion stage in which an integer program (PE^{IP}) maximizes the overall value of new blocks to be added to the previous selection. In this integer program (PE^{IP}), the blocks selected in the first stage are forced to be extracted; their values are not considered anymore, and residual capacity constraints are applied to the sole new blocks.

Note that the minimum and maximum constraints on extraction consumption are not considered in the first stage of this algorithm, which may result in unfeasibility in the second stage when too many blocks are selected. Indeed, in such a case, some overall maximum extraction constraints could be violated. The inconsistent size of pit sometimes obtained by varying the ore price is referred to as the *gap problem* in the literature (Meagher et al., 2014).

We propose a single-stage expansion strategy to prevent such pitfalls. It consists in solving an integer program (18)-(23) that forces the extraction of all the blocks belonging to the final pit while satisfying all the resource consumption constraints consolidated over the entire horizon. This strategy may prioritize the extraction of blocks close to the surface that have the advantage of requiring the extraction of fewer blocks to satisfy the slope precedence constraints and identify a feasible solution. Compared with the algorithm proposed by Lambert & Newman (2014), this method generates a more precisely adjusted set of blocks which helps to prevent the gap problem.

Let FP denote the set of blocks in the final pit, L_r and U_r denote the consolidated minimum and maximum resource consumption of resource r over the time horizon, that is $L_r = \sum_{t\in\mathcal{T}} L_{rt}$ and $U_r = \sum_{t\in\mathcal{T}} U_{rt}$. Let δ_r^L and δ_r^U denote the capacity constraint adjustment parameters which help to prevent unfeasibilities arising from the deconsolidated maximum and minimum resource consumption constraints of the original multiperiod problem, the integer programming model to be solved is as follows:

$$\max \sum_{b\in\mathcal{B}} p_b x_b \tag{18}$$

$$x_b \le x_{b'} \qquad \forall\,(b,b') \in \mathcal{P} \tag{19}$$

$$x_b = 1 \qquad \forall b \in FP \tag{20}$$

$$\sum_{b\in\mathcal{B}} q_{br} x_b \le U_r - \delta_r^U \qquad \forall r \in \mathcal{R} \tag{21}$$

$$\sum_{b\in\mathcal{B}} q_{br} x_b \ge L_r + \delta_r^L \qquad \forall r \in \mathcal{R} \tag{22}$$

$$x_b \in \{0,1\} \qquad \forall b \in \mathcal{B}. \tag{23}$$

The objective function (18) maximizes the total profit. Constraints (19) correspond to the precedence constraints given by the slope specifications. Constraints (20) state that all blocks of the final pit FP are extracted. Constraints (21) and (22) state that the adjusted consolidated maximum and minimum resource consumption constraints over the time horizon should be satisfied. Constraints (23) reflect the nature of the variables.

Note that the solution of model (18)-(23) may be a set of blocks for which there is no feasible solution satisfying the minimum resource consumption constraints for each of the periods. It only guarantees that the blocks of the extended pit are sufficient to satisfy the consolidated minimum use of mining and processing capacities without exceeding the consolidated ones over the time horizon. Indeed, we cannot guarantee that in all cases sufficient ore is available at each period to be extracted to meet processing constraints. In the worst case, an instance could be infeasible and its lower bounds would need to be modified to be less restrictive.

5. Computational results

The RAD heuristic was coded in C++ and run on a 64-bit Windows OS workstation with ten 2.6 GHz Intel Xeon E5 2660v3 processors and 120 Gb RAM. It uses the MineLink library (developed at Delphos Mine Planning Laboratory at Universidad de Chile), which implements data structures to store block models and offers an implementation of the pseudoflow algorithm proposed by Hochbaum & Chen (2000) for the final pit problems. The Gurobi 8.0 library was used to solve the MIP subproblems.

A comparison of the RAD heuristic applied to both BSP and FBSP models with a *sliding time window heuristic* (STWH) and the sliding time window heuristic proposed by Lambert & Newman (2014) which relaxes the integrality constraints on the variables that correspond to the periods beyond the incumbent time window, thereafter called *sliding time window heuristic with relaxation (STWHR)*, was performed on 13 modified problems from the 11 instances of the Minelib library (Espinoza et al., 2013) available at http://mansci-web.uai.cl/minelib. Some of these instances are real (KD, P4HD, W23, McLaughlin and McLaughlin limit), others are fictitious (Newman, ZuckSmall, ZuckMedium, ZuckLarge, Marvin, and SM2). Regarding the real instances, KD refers to a copper mine in Arizona, USA; P4HD to a gold and copper mine in Nevada, USA; W23 to a gold mine in North America; and McLaughlin to a gold mine in California, USA. McLaughlin Limit refers to the McLaughlin final pit defined by Somrit (2011).

5.1. *Test instances with minimum resource consumption constraints*

As this study aims to solve difficult instances with positive minimum resource consumption constraints, we propose a set of new test instances based on the constrained pit limit problems (CPIT) instances of the Minelib library. These instances have been modified to include positive lower bounds on mining and processing tonnage capacities. The instances Newman1, Zuck Medium, P4HD, Zuck Large, McLaughlin Limit, and McLaughlin have been modified by reducing the horizon (in bold in Table 1) and by adding or modifying the lower and upper bounds (Table 2).

In the CPIT instances of the Minelib library, each block is associated with two parameters: the *mined* and *processed* tonnages. The mined tonnage of a block corresponds to its actual tonnage, while the processed tonnage corresponds to the tonnage to be processed if the block is extracted. Since the decision to process or not process an extracted block is predetermined, the processed tonnage of a block is equal to the mined tonnage if the block has to be processed once extracted; it is equal to zero otherwise. In the original Zuck Medium instance, the mined tonnage of several blocks is lower than the processed tonnage. This issue has been corrected in the modified Zuck Medium instance by setting the mined tonnage equal to the greater of the mined and processed tonnage of the original instance.

All instances except P4HD and W23 have been modified to generate conflicts between lower bounds on the processing capacities and upper bounds on the mining capacities.

Table 1 summarizes, for each instance, the size of the block model and the number of variables and constraints. The names of modified instances are the original Minelib library instance names prefixed with “m” or “m2”. Notice that modified instances of McLaughlin (resp. Marvin) have the same number of periods, blocks, and variables, but differ in the bounds (see Table 2).

Table 1
Description of the modified instances of the Minelib library (reduced horizons are in bold).

			BSP		FBSP	
Instance Name	T	Blocks	Vars	Cons	Vars	Cons
mNewman1	**4**	1,060	4,240	19,944	8,480	24,184
mZuckSmall	20	9,400	188,000	3,100,880	376,000	3,288,880
mKD	12	14,153	169,836	2,807,220	339,872	2,977,056
mZuckMedium	**14**	29,277	409,878	18,206,832	819,756	18,616,710
mP4HD	**8**	40,947	327,576	6,236,480	655,152	6,564,056
mMarvin	20	53,271	1,065,420	14,078,120	2,130,840	15,143,540
m2Marvin	20	53,271	1,065,420	14,078,120	2,130,840	15,143,540
mW23	12	74,260	891,120	10,068,600	1,782,240	10,959,720
mZuckLarge	**25**	96,821	2,420,525	28,748,250	4,841,050	31,168,775
mSM2	30	99,014	2,970,420	5,869,800	5,940,840	8,840,220
mMcLaughlinLim	**10**	112,687	1,126,870	31,481,740	2,253,740	32,608,610
mMcLaughlin	**10**	2,140,342	21,403,420	742,841,160	42,806,840	774,244,580
m2McLaughlin	**10**	2,140,342	21,403,420	742,841,160	42,806,840	774,244,580

5.2. *Parameter settings*

All subproblems of the proposed heuristic were solved in a depth-first search manner. In this section, we describe how the capacity constraints were adjusted for the BSP formulation. Moreover, we explain the preprocessing procedure used before solving the instances, and how the solver parameters were set.

Capacity constraints adjustments. As reported in Section 4.2, unfeasibilities may occur in the lower nodes of the binary tree when solving the BSP model. In such cases, we use parameters δ^U_{r,h_m} and δ^L_{r,h_m} to set more restrictive capacity constraints for resource r of subproblem m. In the numerical experiments, these parameters were set to:

$$\delta^U_{r,h_m} = 2^{h_m-1} \max_{b\in\mathcal{B}^m} \{q_{b,r}\} \text{ and } \delta^L_{r,h_m} = 2^{h_m} \text{ with } h_m > 0,$$

where $\mathcal{B}^m$ denotes the block model of the subproblem m, h_m is the height of subproblem m in the graph of subproblems, $\delta^U_{r,0}$ and $\delta^L_{r,0}$ are set to 0. All parameters δ^U_{r,h_m} and δ^L_{r,h_m} were set to 0 when solving subproblems with the FBSP model.

With respect to the final pit expansion model presented in Section 4.3, the adjustments δ^L_r and δ^U_r were set to the values of the adjustments δ^U_{r,h_m} and δ^L_{r,h_m} used in the first subproblem of the RAD heuristic.

Preprocessing procedure. The final pit problem is solved for each instance using a pseudoflow algorithm as proposed by Hochbaum & Chen (2000). The final pit eventually needs to be extended whenever the sum of resource capacities to extract it is below the sum over the time horizon of the minimum resource consumption for at least one resource. In such a case, the final pit extension procedure proposed in Section 4.3 is used.

Table 2
Lower and upper bounds on the mining and processing capacities (in tons) for the modified instances.

		Mining capacity		Processing capacity	
Instance	Periods	LB	UB	LB	UB
mNewman1	all	900,000	1,500,000	650,000	900,000
mZuckSmall	all	20,000,000	35,000,000	6,300,000	25,000,000
mKD	1	1,800,000	2,100,000	1,000,000	2,000,000
mKD	2-12	1,800,000	2,100,000	1,710,000	2,000,000
mZuckMedium	all	7,000,000	18,000,000	5,150,000	8,000,000
mP4HD	all	5,000,000	20,000,000	4,000,000	10,000,000
mMarvin	all	15,000,000	30,000,000	8,000,000	21,000,000
m2Marvin	all	27,000,000	30,000,000	9,000,000	21,000,000
mW23	1	8,000,000	10,000,000	2,000,000	3,610,000
mW23	2-12	5,000,000	15,000,000	2,000,000	3,610,000
mZuckLarge	all	1,000,000	2,500,000	700,000	1,200,000
mSM2	1	5,000,000	6,000,000	700,000	800,000
mSM2	2	5,000,000	6,000,000	900,000	1,000,000
mSM2	3	5,000,000	6,000,000	1,250,000	1,300,000
mSM2	4-30	5,000,000	6,000,000	1,100,000	1,300,000
mMcLaughlinLim	all	10,000,000	20,000,000	2,500,000	3,300,000
mMcLaughlin	all	10,000,000	20,000,000	2,500,000	3,300,000
m2McLaughlin	all	11,500,000	20,000,000	2,500,000	3,300,000

Solver parameters. The subproblems of the binary tree were solved with the Gurobi solver by using a value of 1% for the relative MIP optimality gap parameter. Default values were used for all other parameters.

5.3. Comparison of the RAD heuristic with the sliding time window heuristics STWH and STWHR

Table 3 shows the size of the subproblems solved for both the BSP and FBSP models. Note that the proposed methodology successfully decreases the size of the original problems listed in Table 1. Moreover, we observe that FBSP subproblems are larger than BSP ones. The larger generated subproblem corresponds to the root subproblem of the m2McLaughlin instance. For the m2Marvin and m2McLaughlin instances, the integer programming model proposed in Section 4.3 extends the final pit to satisfy the overall minimum use of mining resource constraints while respecting the remaining constraints. For all other cases, only blocks of the final pit are extracted without the need to expand it.

Table 3
Number of variables and constraints in the subproblems solved using the proposed algorithm.

	BSP						FBSP					
	Smallest		Average		Largest		Smallest		Average		Largest	
Instance	Vars	Cons	Vars	Cons	Vars	Cons	Vars	Cons	Vars	Cons	Vars	Cons
mNewman1	956	3,698	1,413	7,142	2,120	8,192	1,916	5,142	2,835	7,863	4,240	12,092
mZuckSmall	1,356	4,292	4,317	43,632	18,800	300,688	2,852	7,008	8,953	51,486	37,600	328,888
mKD	1,588	15,926	14,927	234,822	28,306	453,717	1,184	2,626	6,701	51,512	56,612	496,176
mZuckMedium	6,702	92,593	15,472	492,121	58,554	2,571,699	10,920	77,884	33,164	527,951	117,108	2,659,530
mP4HD	11,542	89,687	28,181	402,248	81,894	1,518,173	23,056	106,780	54,657	432,275	163,788	1,641,014
mMarvin	1,300	3,704	3,924	27,284	17,032	180,332	3,099	10,655	15,430	65,950	68,120	411,760
m2Marvin	1,423	5,084	4,349	31,785	19,874	188,786	3,364	11,302	18,614	73,617	76,124	423,008
mW23	8,790	39,291	34,735	312,071	148,520	1,603,840	28,144	98,358	81,448	436,699	297,040	1,826,620
mZuckLarge	11,094	65,761	38,448	349,358	193,642	2,203,039	21,252	74,744	76,779	410,402	387,284	2,493,502
mSM2	1,286	1,825	4,868	6,915	36,776	53,500	2,764	4,018	9,780	14,291	73,552	108,664
mMcLaughlinLim	39,866	660,785	84,320	1,911,965	225,374	6,183,661	80,364	721,332	168,909	2,034,149	450,748	6,521,722
mMcLaughlin	39,720	595,014	84,110	1,901,729	221,248	6,077,888	79,765	972,549	172,658	2,158,904	461,832	6,623,507
m2McLaughlin	41,362	621,143	95,746	2,135,640	233,554	6,334,862	99,512	1,190,812	195,148	2,213,745	480,127	6,747,346

Table 4
Results of the two-period sliding time window heuristic (STWH), the sliding time window heuristic with relaxation (STWHR) for the BSP model, and the RAD heuristic for the BSP and FBSP models

		Sliding time window heuristics for the BSP model				RAD heuristic					
		STWH		STWHR		BSP		FBSP			
	LP	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Partial blocks	SubPb
Instance	Upper bound	%	[s]	%	[s]	%	[s]	%	[s]	%	#
mNewman1	23,740,544	**1.79**	11	2.72	6	2.14	2	**2.01**	4	14.64	3
mZuckSmall	929,861,634	Not feasible		> 1 day		**1.70**	250	3.46	242	27.72	19
mKD	81,125,108	> 1 day		> 1 day		-	-	**8.25**	5,031	29.75	11
mZuckMedium	710,409,606	Not feasible		> 1 day		**5.00**	6,703	6.88	39,522	11.30	13
mP4HD	235,019,284	**0.03**	1,287	> 1 day		**0.17**	4,220	0.23	6,688	0.25	7
mMarvin	875,461,209	Not feasible		> 1 day		**2.31**	292	4.81	151	17.50	19
m2Marvin	868,404,340	Not feasible		> 1 day		-	-	**3.96**	92	23.21	19
mW23	399,975,146	Not feasible		> 1 day		0.60	7,327	**0.53**	18,794	9.14	10
mZuckLarge	57,388,739	> 1 day		> 1 day		1.59	14,135	**0.90**	23,486	1.18	24
mSM2	1,247,672,349	Not feasible		0.78	44,480	1.46	20	**0.34**	32	0.41	29
mMcLaughlinLim	1,078,802,179	> 1 day		> 1 day		**0.25**	38,533	0.37	21,871	0.38	9
mMcLaughlin	1,078,847,800	> 1 day		> 1 day		0.56	56,613	**0.28**	70,166	0.57	9
m2McLaughlin	1,076,305,701	> 1 day		> 1 day		-	-	**0.39**	22,566	0.26	9

Table 4 shows a comparison of the two-period sliding time window heuristic (STWH), the sliding time window heuristic with relaxation (STWHR) for the BSP model, and the proposed RAD heuristic for the BSP and FBSP models. The sliding time window heuristics were applied for the BSP model with a two-period sliding time window for the STWH, and with a one-period sliding time window for the STWHR.

For each instance, we report:

- The upper bound corresponding to the LP-relaxation (*LP upper bound*). The Bienstock-Zuckerberg algorithm (Bienstock & Zuckerberg, 2010) was used to compute the LP upper bound for the mNewman1, mZuckMedium, mP4HD, mW23, mZuckLarge, mMcLaughlinLimit, mMcLaughlin, and m2McLaughlin instances. Since the LP-relaxations of both BSP and FBSP formulations are equivalent, the same bound is used to compute the relative integrality gaps for both models.

- The relative integrality gap (*Gap*), between the solution obtained by each tested heuristic and the LP upper bound, computed as follows:

$$\text{Relative integrality gap} = \frac{\text{LP upper bound} - \text{objective value}}{\text{LP upper bound}}$$

- The wall clock computing time (*Time*) for each tested heuristic.

- The percentage of partially mined blocks (*Partial blocks*) corresponding to the number of partially mined blocks divided by the number of extracted blocks in the solution obtained with the proposed heuristic on the FBSP model.

- The number of subproblems (*SubPb*) for the BSP and FBSP models, each of which corresponds to a node of the binary tree. This number is the same for both models.

RAD heuristic. When a feasible solution is obtained, the RAD heuristic is generally more efficient in terms of solution quality and computing time with the BSP model than with the FBSP model. The computing time efficiency of the BSP formulation may be explained by the fact that FBSP models are larger than BSP ones, as shown in Table 3, and the tighter upper bounds on the mining and processing capacities (Section 4.2) that may decrease the number of conflicts between knapsack constraints. However, no feasible solutions were obtained using the BSP model for mKD, m2Marvin, and m2McLaughlin instances; an unfeasibility was reported for one of their subproblems. No unfeasibility was reported with the FBSP model, the proposed heuristic always obtained feasible solutions with a relative integrality gap of 2.49% on average over the 13 instances. The percentage of partially mined blocks varies from 0.25% for the mP4HD instance to 29.79% for the mKD instance. For a flat block model as the mSM2, only 0.41% of partially mined blocks are generated for 29 periods. These percentages seem to depend more on the 3-dimensional shape of the block model than on the number of periods of the instance.

Sliding time window heuristics. Only three instances could be solved with the sliding time window heuristics STWH and STWHR within one day of computational time. The mNewman1 instance has been solved with both methods, while the mP4HD and mSM2 instances could only be solved with one method, STWH and STWHR, respectively. We believe that the sliding time window heuristic (STWH) fails to generate feasible solutions because of the lack of information on the later periods in the initial iterations. In the sliding time window heuristic with relaxation (STWHR) the relaxed variables carry information about the future, but we observe that this method has only been able to solve the smallest instances mNewman1 and mSM2 after applying the preprocessing procedure. For all the others instances, it could not find any solution within one day of computational time. Indeed, we observe that the size of the subproblems prevents the mP4HD instance from being solved in a reasonable time with the STWHR, while the lack of information on the latter periods prevents the mSM2 from being solved with the STWH.

Comparing the RAD heuristic with the sliding time window heuristics STWH and STWHR. As the results show, the RAD heuristic addresses both issues: the lack of information on the later periods in the case of the STWH and the size of the subproblems to be solved in the case of the STWHR. Indeed, it reported feasible solutions with low relative integrality gaps even for the largest and most difficult modified instances of the Minelib. Therefore, the RAD heuristic is an interesting alternative to the classical sliding time window heuristics when lower bounds on resource consumption constraints are considered.

6. Conclusions

A recursive time aggregation-disaggregation (RAD) heuristic was introduced to solve the following two versions of the open-pit mine scheduling problem with minimum resource consumption constraints: the block sequencing problem (BSP) and the fractional block sequencing problem (FBSP) proposed by Vossen et al. (2016). Using a binary tree structure, it recursively aggregates and disaggregates the scheduling periods to sequentially solve two-period subproblems. Since the application of the proposed heuristic to the BSP model may generate unfeasible solutions, we proposed tightening the father subproblem's constraint capacities while considering the height of the binary tree. However, if the proposed heuristic is applied to the FBSP model, wherein blocks can be extracted in parts, the lower and upper bounds of the resource consumption constraints need not be adjusted.

For the mMcLaughlin instance, the RAD heuristic with both of the models provided feasible solutions. With 2,340,142 blocks, 10 periods, and 2 minimum resource consumption constraints for each period, mMcLaughlin is, to our knowledge, the largest public instance with lower bounds on resource consumption for which a feasible solution is reported.

While comparing with the more classical sliding time window heuristics STWH and STWHR, the RAD heuristic addresses both pitfalls of the lack of information on the later periods encountered with the STWH, and of the large size of the subproblems encountered with the STWHR.

A single-stage procedure is also introduced to extend the final pit whenever the total use of the mining and processing resources is below the respective required minimums over the time horizon. This procedure prevents the well-documented *gap problem* pitfall.

Moreover, we introduced 13 new instances based on the constrained pit limit problem (CPIT) instances of the Minelib to include lower and upper bounds on resource consumption. Sliding time window heuristics have been implemented, showing the difficulty of solving the scheduling instances with conflicting knapsack constraints. Feasible solutions were obtained by using the proposed heuristic for all 13 instances with the FBSP model, and 10 instances with the BSP model. However, compared with the FBSP model, better average relative integrality gaps were obtained with the BSP model, i.e., 1.58% for the BSP model and 1.98% for the FBSP model over the 10 instances for which feasible solutions were obtained with both of the models.

In the future, the computational times may be reduced by strengthening the formulation of the subproblems by possibly adding clique and cover cuts, as suggested by Bley et al. (2010). To implement those constraints, the algorithm proposed by Bienstock & Zuckerberg (2009) may be used to solve the LP-relaxations of scheduling instances with minimum resource consumption constraints.

The consideration of stochastic versions of these problems is also a relevant research direction for further study.

Acknowledgements

This work was partially funded by the ANID / PIA Project AFB180004. We thank José Fernando da Costa Oliveira and the three anonymous reviewers for their valuable comments and suggestions on earlier versions of this paper.

References

Askari-Nasab, H., Pourrahimian, Y., Ben-Awuah, E., & Kalantari, S. (2011). Mixed integer linear programming formulations for open pit production scheduling. *Journal of Mining Science*, 47(3), 338–359. https://doi.org/10.1134/S1062739147030117

Bienstock, D. & Zuckerberg, M. (2009). A new LP algorithm for precedence constrained production scheduling. *Optimization Online*. http://www.optimization-online.org/DB_FILE/2009/08/2380.pdf. Accessed September 25, 2021

Bienstock, D. & Zuckerberg, M. (2010). Solving LP relaxations of large-scale precedence constrained problems. *Integer Programming and Combinatorial Optimization*, 1–14. https://doi.org/10.1007/978-3-642-13036-6_1

Bley, A., Boland, N., Fricke, C., & Froyland, G. (2010). A strengthened formulation and cutting planes for the open pit mine production scheduling problem. *Computers & Operations Research*, 37(9), 1641–1647. https://doi.org/10.1016/j.cor.2009.12.008

Boland, N., Dumitrescu, I., Froyland, G., & 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. https://doi.org/10.1016/j.cor.2007.12.006

Caccetta, L. & Hill, S. (2003). An application of branch and cut to open-pit mine scheduling. *Journal of Global Optimization*, 27, 349–365. https://doi.org/10.1023/A:1024835022186

Chicoisne, R., Espinoza, D., Goycoolea, M., Moreno, E., & Rubio, E. (2012). A new algorithm for the open-pit mine production scheduling problem. *Operations Research*, 60(3), 517–528. https://doi.org/10.1287/opre.1120.1050

Cullenbine, C., Wood, R., & Newman, A. (2011). A sliding time window heuristic for open pit mine block sequencing. *Optimization Letters*, 5(3), 365–377. https://doi.org/10.1007/s11590-011-0306-2

Dagdelen, K. (1986). Optimum open pit mine production scheduling by lagrangian parameterization. *Proc. 19th Applications for Computers and Operations Research in the Minerals Industries (APCOM) Symposium*, 127–142.

Espinoza, D., Goycoolea, M., Moreno, E., & Newman, A. (2013). Minelib: a library of open pit mining problems. *Annals of Operations Research*, 206(1), 93–114. https://doi.org/10.1007/s10479-012-1258-3

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

Gemcom (2021). *Gemcom Whittle™ Strategic Mine Planning software*. `https://www.3ds.com/products-services/geovia/products/whittle/`. Accessed September 25, 2021

Gershon (1987). Heuristic approaches for mine planning and production scheduling. *Int. Journal of Mining and Geological Engineering*, 5(1), 1–13. `https://doi.org/10.1007/BF01553529`

Hochbaum, D. S. (2008). The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research*, 56(4), 992–1009. `https://doi.org/10.1287/opre.1080.0524`

Hochbaum, D. S. & Chen, A. (2000). Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. *Operations Research*, 48(6), 894–914. `https://doi.org/10.1287/opre.48.6.894.12392`

Hochbaum, D. S. & Orlin, J. B. (2012). Simplifications and speedups of the pseudoflow algorithm. *Networks*, 61, 40–57. `https://doi.org/10.1002/net.21467`

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

Johnson, T. (1969). Optimum open-pit mine production scheduling. *A Decade Of Digital Computing In The Mineral Industry*, 539–562.

Jélvez, E., Morales, N., Nancel-Penard, P., & Cornillier, F. (2020). A new hybrid heuristic algorithm for the precedence constrained production scheduling problem: A mining application. *Omega*, 94, 102046. `https://doi.org/10.1016/j.omega.2019.03.004`

Jélvez, E., Morales, N., Nancel-Penard, P., Peypouquet, J., & Reyes, P. (2016). Aggregation heuristic for the open-pit block scheduling problem. *European Journal Of Operational Research*, 249(3), 1169–1177. `https://doi.org/10.1016/j.ejor.2015.10.044`

Lambert, W., Brickey, A., Newman, A., & Eurek, K. (2014). Open-pit block-sequencing formulations: A tutorial. *INFORMS Journal on Applied Analytics*, 44(2), 127–142. `https://doi.org/10.1287/inte.2013.0731`

Lambert, W. & Newman, A. (2014). Tailored lagrangian relaxation for the open pit block sequencing problem. *Annals of Operations Research*, 222(1), 419–438. `https://doi.org/10.1007/s10479-012-1287-y`

Lamghari, A., Dimitrakopoulos, R., & Ferland, J. A. (2014). A hybrid method based on linear programming and variable neighborhood descent for scheduling production in open-pit mines. *Journal of Global Optimization*, 63(3), 555–582. `https://doi.org/10.1007/s10898-014-0185-z`

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

Letelier, O., Espinoza, D., Goycoolea, M., Moreno, E., & Muñoz, G. (2020). Production scheduling for strategic open pit mine planning: A mixed-integer programming approach. *Operations Research*, 68(5), 1425–1444. `https://doi.org/10.1287/opre.2019.1965`

Liu, S. & Kozan, E. (2016). New graph-based algorithm to efficiently solve large scale open pit mining optimization problems. *Expert Systems With Applications*, 43, 59–65. `https://doi.org/10.1016/j.eswa.2015.08.044`

Meagher, C., Dimitrakopoulos, R., & Avis, D. (2014). Optimized open pit mine design, pushbacks and the gap problem—a review. *Journal of Mining Science*, 50(3), 508–526. `https://doi.org/10.1134/S1062739114030132`

Morales, N.and Jélvez, E., Nancel-Penard, P., Marinho, A., & Guimaraes, O. (2015). A comparison of conventional and direct block scheduling methods for open-pit mine production scheduling. *Proceedings de APCOM 2015, Fairbanks, Alaska, USA*, 1040–1051.

Moreno, E., Espinoza, D., & Goycoolea, M. (2010). Large-scale multi-period precedence constrained knapsack problem: A mining application. *Electronic Notes in Discrete Mathematics*, 36, 407–414. `https://doi.org/10.1016/j.endm.2010.05.052`

Newman, A., Rubio, E., Caro, R., Weintraub, A., & Eurek, K. (2010). A review of operations research in mine planning. *INFORMS Journal on Applied Analytics*, 40(3), 222–245. `https://doi.org/10.1287/inte.1090.0492`

Osanloo, M., Gholamnejad, J., & 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. `https://doi.org/10.1080/17480930601118947`

Parra, A., Morales, N., Vallejos, J., & Nguyen, P. M. V. (2018). Open pit mine planning considering geomechanical fundamentals. *International Journal of Mining, Reclamation and Environment*, 32(4), 221–238. `https://doi.org/10.1080/17480930.2017.1278579`

Picard, J. (1976). Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11), 1268–1272. `https://doi.org/10.2307/2630227`

Ramazan, S., Dagdelen, K., & Johnson, T. (2005). Fundamental tree algorithm in optimising production scheduling for open pit mine design. *Mining Technology*, 114(1), 45–54. `https://doi.org/10.1179/037178405X44511`

Samavati, M., Essam, D., Nehring, M., & Sarker, R. (2017a). A local branching heuristic for the open pit mine production scheduling problem. *European Journal of Operational Research*, 257(1), 261–271. `https://doi.org/10.1016/J.EJOR.2016.07.004`

Samavati, M., Essam, D., Nehring, M., & Sarker, R. (2017b). A methodology for the large-scale multi-period precedence-constrained knapsack problem: an application in the mining industry. *International Journal of Production Economics*, 193, 12–20. `https://doi.org/10.1016/j.ijpe.2017.06.025`

Samavati, M., Essam, D., Nehring, M., & Sarker, R. (2017c). A new methodology for the open-pit mine production scheduling problem. *Omega*, 81, 169–182. `https://doi.org/https://doi.org/10.1016/j.omega.2017.10.008`

Somrit, C. (2011). *Development of a new open pit mine phase design and production scheduling algorithm using mixed integer linear programming*. Dissertation, Colorado School of Mines, Golden, CO.

Vossen, T. W. M., Wood, R. K., & Newman, A. M. (2016). Hierarchical Benders Decomposition for Open-Pit Mine Block Sequencing. *Operations Research*, 64(4), 771–793. `https://doi.org/10.1287/opre.2016.1516`

Zeng, L., Liu, S. Q., Kozan, E., Corry, P. & Masoud, M. (2021). A comprehensive interdisciplinary review of mine supply chain management. *Resources Policy*, 74, 102274. `https://doi.org/10.1016/j.resourpol.2021.102274`