Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

A new hybrid heuristic algorithm for the Precedence Constrained Production Scheduling Problem: A mining application[☆]

Enrique Jélvez[a,*], Nelson Morales[a], Pierre Nancel-Penard[a], Fabien Cornillier[b]

[a] Delphos Mine Planning Laboratory, Advanced Mining Technology Center & Department of Mining Engineering, Universidad de Chile, Santiago, Chile
[b] Department of Industrial Engineering, Universidad de Ingenieria y Tecnologia - UTEC, Lima, Peru

ARTICLE INFO

Article history:
Received 13 February 2018
Accepted 9 March 2019

Keywords:
Precedence constrained production scheduling
Open-pit mine production planning
Hybrid heuristic
Mixed-integer linear programming

ABSTRACT

In this work we address the Precedence Constrained Production Scheduling Problem (PCPSP), the problem of scheduling tasks in such a way that total profit is maximized, while satisfying conditions such as precedence constraints among tasks and side constraints. A motivation for addressing this problem comes from open-pit mining industry, where the PCPSP seeks to maximize the net present value of an ore deposit by selecting the blocks (tasks) to extract, their extraction periods and their processing options, while satisfying constraints as precedences among blocks, limited availability of operational resources and maximum and/or minimum allowable concentrations of ore-grade or pollutants. Since real-world models have millions of blocks and constraints, the monolithic problem is computationally intractable. This article presents a hybrid heuristic algorithm that combines a rolling horizon decomposition with a block preselection procedure, allowing near-optimal solutions to be quickly determined. The proposed heuristic was tested on all the PCPSP instances of the MineLib library and has shown a significant improvement over the previous reported results. Moreover, a good feasible solution has been found for the instance W23, for which no solution has been previously reported.

1. Introduction

The Precedence Constrained Production Scheduling Problem (PCPSP) belongs to a special class of problems commonly found in operations management and production planning, where tasks must be scheduled over a time horizon and assigned to a destination (i.e. a processing facility), satisfying production capacity constraints and precedence constraints, while maximizing profits. Simplified versions of this problem only consider a single-period, i.e., they do not take into account the temporal dimension [1–6]. In other versions of this problem the processing facility is preassigned [7–10]. Among all applications of this problem and its extensions, the mine production planning seems to be the most challenging due to the huge size of instances [11]. In this article, we propose applying the PCPSP to open-pit mine production planning.

In long-term open-pit mine production planning the goal is to maximize the net present value of the extracted and processed material. In such mines, the mineral is reached by digging material from the surface. Depending on its composition, its profitability, and the availability of the processing facilities, the extracted material is either assigned to a processing facility, accumulated into stockpiles for later processing, or put into waste dumps. To define what portions of the terrain must be mined at each time-period, the terrain is modeled as a three-dimensional grid of blocks and the planning horizon is discretized into periods. In this application, tasks correspond to blocks and the objective is to find the best strategy to extract and process the blocks.

For each block, estimations on the ore content, tonnage, and other relevant attributes are constructed by using geostatistical methods (see [12]) based on terrain samples. The location and attributes of the blocks form the so-called *block model*. The contribution of a block to the overall value mainly depends on its geological attributes, its extraction period, and how it is processed, in addition to external variables such as commodity price and mining costs.

In mining context, the PCPSP is the problem of determining which blocks to extract at each period in the planning horizon, and of assigning each extracted block to a processing facility, while maximizing the discounted profit satisfying technical and operational constraints. Examples of technical constraints are the slope precedences, by which the extraction of a block is feasible when a determined set of blocks located above it has been extracted, respecting maximum slope angles to ensure the stability of the pit

☆ This manuscript was processed by Associate Editor Singh.

* Corresponding author at: Advanced Mining Technology Center, Universidad de Chile. Avenida Tupper 2007, Santiago, postal code 8370451, Chile.

E-mail address: enrique.jelvez@amtc.cl (E. Jélvez).

https://doi.org/10.1016/j.omega.2019.03.004
0305-0483/© 2019 Elsevier Ltd. All rights reserved.

walls. Operational constraints are inherent to the extraction process: the amount of material to be transported and processed (operational resources) at each period is generally subject to upper and possibly lower limits. Processing material also implies satisfying blending constraints associated with its quality. Indeed, the efficiency of the processing, even its feasibility, depends on the combination of blocks processed simultaneously at a facility. In particular, it may not be feasible to process alone a block with a high content of pollutants such as arsenic, even with a high ore-grade. In such case, it could be possible to process it by mixing it with other blocks (even low ore-grade ones) whenever the blending provides an acceptable amount of pollutants.

Most of the real instances of the PCPSP in the mining industry are difficult to solve with block models containing millions of blocks for a planning horizon as long as several decades. The main contribution of this article is to propose a hybrid heuristic based on a sliding time-window and a linear relaxation to preselect a small subset of blocks to be scheduled within each time-window. Contrary to other existing heuristics applied to the PCPSP, the proposed algorithm is able to tackle problems with blending constraints.

The remainder of this article is organized as follows: Section 2 provides a brief summary of the most relevant or best-known approaches and results from the literature. Section 3 provides a mathematical model for the PCPSP. A description of the proposed heuristic algorithm is given in Section 4, followed by computational results in Section 5 and conclusions in Section 6.

2. Related work

Lerchs and Grossmann [1] presented an algorithm to solve the *Final Pit Problem*, a simplified version of the PCPSP in which a single value – positive or negative – is associated with each block without consideration to its extraction period and its final destination: the temporal dimension is ignored. In this problem, the objective is to identify the set of blocks to be extracted to maximize the total undiscounted profit while satisfying only slope precedence constraints. In the same article, Lerchs and Grossmann described how a sequence of nested pits can be generated with their algorithm and used as a guide to schedule the extraction of blocks over time (see Jélvez et al. [13] for a review). Commercial software, such as Whittle from Geovia, are based on this algorithm.

Closer to the problem under consideration in this article, the pioneer work of Johnson [14] proposed for the first time a linear programming formulation under slope precedence, capacity and specific blending constraints within a multi-destination setting, where the optimization model determines the best process to apply to each extracted block. Due to the nature of variables used in this model, it may happen that a portion of a block is extracted while all the overlying blocks have not been mined, making the solution unfeasible in practice.

Because of the difficulty to directly solve PCPSP instances of practical sizes, many algorithms have been proposed to find good feasible solutions of this problem and its variations. A well-studied variation consists in setting the destination of each block *a priori* (i.e., the destination of each block is not a decision variable) and ignoring blending requirements (see for example Ramazan [15], Cullenbine et al. [16], Chicoisne et al. [17], Jélvez et al. [18] and Samavati et al. [19]).

The particular case of the PCPSP, which includes block destinations, slope precedence, capacity and blending constraints, has been previously studied. Bienstock and Zuckerberg [11] addressed the linear relaxation of the PCPSP and proposed a method based on a Lagrangian relaxation evidencing a substantial computation-time improvement with regards to the standard linear programming solvers. As such, this important method does not give any feasible solution, but rather gives an upper-bound on the objective value.

Espinoza et al. [20] applied a heuristic based on a topological sorting to solve both, the PCPSP considered in this paper, where the destination of a block is a decision variable, and the *Constrained Pit Limit Problem* (CPIT), a simplified version of the PCPSP where block destinations are preassigned. They also proposed standardized testbed instances (MineLib library) for these problems.

While some authors have proposed outperforming solutions for the MineLib CPIT instances (see Lamghari et al. [21], Liu and Kozan [22], Jélvez et al. [18] and Samavati et al. [19,23]), to the best of our knowledge, only Kenny et al. [24] have reported improved solutions for some MineLib PCPSP instances by using a Greedy Randomized Adaptive Search Procedure (GRASP). However, neither lower limits on resources consumption nor general side constraints as blending are considered in their model, which makes it impossible to apply to some MineLib instances.

This article tackles the PCPSP as introduced by Espinoza et al. [20], proposes a hybrid heuristic algorithm and compares the results with those published in Espinoza et al. [20] and Kenny et al. [24].

3. The Precedence Constrained Production Scheduling Problem

Denote by $\mathcal{B}$ the set of blocks, by $\mathcal{B}_b$ the subset of predecessors of block $b \in \mathcal{B}$, by $\mathcal{D}$ the set of destinations, and by $\mathcal{R}$ the set of operational resources needed to extract and process the blocks. A profit p_{bdt} is obtained by extracting block b and processing it at destination d at period $t \in \mathcal{T} = \{1, \dots, T\}$, where T denotes the planning horizon, while an amount q_{bdr} of operational resource r is used to extract block b and process it at destination $d \in \mathcal{D}$. $\underline{R}_{rt}$ represents the minimum use and $\bar{R}_{rt}$ the maximum availability of resource r at period t.

We define binary variables z_{bt} equal to 1 if block b is extracted by period t, and 0 otherwise; and continuous variables y_{bdt} represent the portion of block b sent at destination d at period t. The PCPSP can be formulated as follows:

$$\text{(PCPSP)} \qquad \max \sum_{b\in\mathcal{B}} \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} p_{bdt}\, y_{bdt} \tag{1}$$

$$\text{s.t.} \quad z_{b,t-1} \le z_{bt} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{2}$$

$$z_{bt} - z_{b,t-1} = \sum_{d\in\mathcal{D}} y_{bdt} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{3}$$

$$z_{bt} \le z_{b't} \qquad \forall b \in \mathcal{B}, b' \in \mathcal{B}_b, t \in \mathcal{T} \tag{4}$$

$$\underline{R}_{rt} \le \sum_{b\in\mathcal{B}} \sum_{d\in\mathcal{D}} q_{bdr} y_{bdt} \le \bar{R}_{rt} \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{5}$$

$$\underline{a} \le Ay \le \bar{a} \tag{6}$$

$$z_{bt} \in \{0, 1\} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{7}$$

$$z_{b0} = 0 \qquad \forall b \in \mathcal{B} \tag{8}$$

$$y_{bdt} \in [0, 1] \qquad \forall b \in \mathcal{B}, d \in \mathcal{D}, t \in \mathcal{T} \tag{9}$$

In this formulation, the objective function (1) maximizes the discounted total profit. Constraints (2) ensure that a block is scheduled in one period at most. Constraints (3) require that if a block is extracted, it must be fully sent to one or more destinations. Constraints (4) prevent the extraction of any block for which all the predecessors have not been previously extracted. Constraints (5) state that the minimum $\underline{R}_{rt}$ and maximum $\bar{R}_{rt}$ use of every operational resource r are satisfied for each period t, and constraints (6) correspond to the general side constraints with lower and upper bounds $\underline{a}$ and $\bar{a}$, respectively. Finally, constraints (7) to (9) reflect the nature of the variables.

General side constraints may represent blending requirements to feed processing plants, but other examples are presented in Espinoza et al. [20] and could include: (i) a minimum number of blocks that must be extracted on a given level; (ii) ore allowed to be stockpiled; (iii) a variable production and/or processing rate, e.g., it is possible to purchase extraction equipment and/or increase the capacity of the processing plant(s); (iv) a minimum number of blocks at the bottom of the pit; and (v) a limitation to the number of areas that can be simultaneously mined due to geotechnics and equipment availability.

We use a *by*-formulation equivalent to the stronger formulation proposed by Bienstock and Zuckerberg [11] and Espinoza et al. [20] where decision variables x_{bt} take value 1 if block b is extracted *at* period t. In the proposed formulation, we apply the variable substitution $x_{bt} = z_{bt} - z_{b,t-1}$ with $z_{b0} = 0$ in order to obtain a better representation of the precedence constraints making explicit the underlying network structure, as reported in the literature (Caccetta and Hill [7] and Lambert et al. [25]). In this representation, decision variables z_{bt} take value 1 if block b is extracted *by* period t. As a final comment, a full-binary formulation of the PCPSP can be found in Jélvez et al. [26].

4. A hybrid heuristic based on rolling horizon and block preselection

The PCPSP is a strongly NP-hard problem [20] and solving it to optimality with an optimization solver is intractable for real size instances involving a prohibitive number of blocks and periods. We propose decomposing the monolithic problem into a series of smaller subproblems on a rolling horizon basis, preselecting a subset of candidate blocks to consider in each subproblem. The significantly reduced number of variables and constraints of the subproblems generally allows building a feasible solution, if one exists.

4.1. Rolling horizon

Starting from the first period $t = 1$ of the planning horizon, this method iteratively constructs a schedule for each period by solving the PCPSP for a reduced time-window $\{t, \dots, \min\{t + w - 1, T\}\}$, where the maximum length of time-window w is an integer parameter to be determined. Each time the subproblem is solved, the variables z_{bt} and y_{bdt} are fixed for the first Δ periods of the incumbent time-window, where $\Delta \le w$ is another parameter to be determined. The time-window is then moved forward by Δ periods, and the new subproblem is solved for the respective new time-window. The procedure stops when the last subproblem that includes the period $t = T$ has been solved and fixed. Note that when a solution is obtained for a subproblem, the procedure allows a partial or complete fix of the time-window as part of the final feasible solution.

Similar approaches have been explored by Cullenbine et al. [16], and Lambert and Newman [27] for a simpler problem (see Section 2), where the authors consider a sliding time-window, but additionally relax the integrality constraints on the variables corresponding to the periods beyond the incumbent time-window. Therefore, the subproblems always have the same number of periods than the monolithic instance. The proposed heuristic ignores these periods to reduce the number of variables considered in each iteration. Indeed, preliminary experiments on several PCPSP instances show that the impact on the objective value when relaxing the integrality constraints on the variables corresponding to the subsequent periods does not justify the major increase of its computation-time. Appendix A shows that no feasible solution has been found with this heuristic within 24 hours of computational time for 8 out of 10 MineLib instances. In these articles, capacity constraints are also approached using a Lagrangian relaxation, while the proposed approach keeps them intact in the subproblems formulation.

4.2. Block preselection

Despite a significant reduction of the number of decision variables and constraints when the problem is decomposed into simpler subproblems as described in Section 4.1, the resulting subproblems may still be difficult to solve. Indeed, the number of blocks in a mine can be considerable and easily exceeds hundreds of thousands, or even millions of blocks. To overcome this difficulty, we propose a heuristic based on the *expected extraction times* introduced by Chicoisne et al. [17] to preselect the subset of blocks to be included in the model.

Let $\tilde{z}^*_{bt}$ be the solution of the LP relaxation of the monolithic PCPSP instance. The expected extraction time of any block b is given by:

$$ET_b = \sum_{t\in\mathcal{T}} t\,(\tilde{z}^*_{bt} - \tilde{z}^*_{b,t-1}) + (T+1)(1 - \tilde{z}^*_{bT}). \tag{10}$$

The idea is to interpret the fractional values $\tilde{z}^*_{b\cdot}$ as the cumulative distribution probability of the extraction time, therefore $\tilde{z}^*_{bt} - \tilde{z}^*_{b,t-1}$ represents the probability of extraction of block b at period t. We assume that any block b not extracted by time T is extracted at time $T + 1$, and we set $ET_b = T + 1$.

We define B as the set of blocks not yet extracted at period t and for which the expected extraction time ET_b is smaller than $\min\{t + w - 1, T\} + s$, where $s > 0$ is a continuous parameter to be determined that represents a tolerance for a block to be considered in each subproblem. This tolerance parameter gives some control over the selected blocks, however in our proposal we take advantage of fast algorithms such as the Bienstock-Zuckerberg algorithm [11] to solve the LP relaxation of the PCPSP, and based on expected extraction times, to choose a value of s to preselect a reduced set of blocks to be considered in each subproblem on a rolling horizon.

It is worth noting that in this procedure the expected times are used as a block preselection tool to reduce the size of the subproblems: they are not used to generate a sequence of blocks as proposed in the TopoSort heuristic developed by Chicoisne et al. [17]. Note that the TopoSort approach may have a high risk for producing infeasible solutions when, for example, there are lower bounds on resource constraints. Besides, by construction it cannot tackle general side constraints such as blending.

4.3. Description of the hybrid heuristic algorithm

The proposed hybrid heuristic combines a rolling horizon method in order to reduce the number of periods in each subproblem (Section 4.1) and a block preselection procedure based on expected extraction times in order to reduce the number of blocks within each subproblem (Section 4.2). Then a feasible solution is constructed iteratively. Note that the hybrid heuristic depends on three parameters: w (length of the sliding time-window), Δ (sliding shift length) and s (tolerance to select blocks from expected extraction times). In particular, when $w = \Delta = T$ and $s \ge T + 1$ this heuristic is equivalent to solving the monolithic version of the PCPSP, therefore, the algorithm is exact if the optimality gap is set to zero in the Branch & Cut resolution. For other cases, only a sub-optimal feasible solution would be obtained. It is important to point out that the nearsightedness of our heuristic does not guarantee finding a feasible solution, if one exists, however, the parameters could be modified whenever a subproblem is infeasible in an attempt to recover feasibility.

Basically, the algorithm has the following steps:

1. Select a time-window, according to Section 4.1.
2. Select a sub-block model, in accordance with Section 4.2.
3. Construct a subproblem.

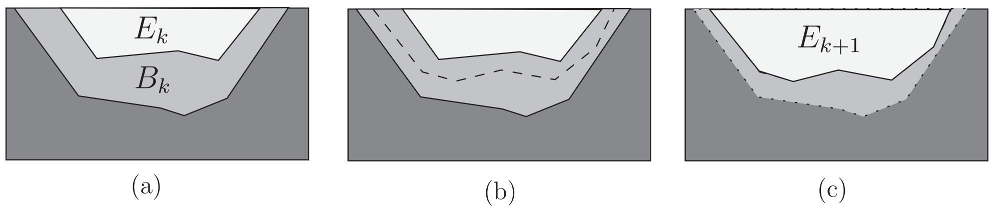

Fig. 1. A k-th iteration of the proposed algorithm. E_k is the set of already extracted blocks and B_k is the set of preselected blocks used for the auxiliary problem. Firstly (a) the set of blocks B_k are computed according to expected extraction times and to the tolerance parameter s, then (b) a subset of B_k is scheduled for extraction and processing, and finally (c) the scheduled blocks are updated.

```
Algorithm: Hybrid heuristic

Input: A PCPSP instance: Block model $\mathcal{B}$, sets of precedence arcs $\mathcal{B}_b$, set of destinations $\mathcal{D}$, a number
       of time-periods $T$, block values, number of resources $R$, amount of resource required per block,
       bounds on resources and general side constraints.
       Heuristic parameters: Sliding time-window size $w$, sliding step $\Delta$ and tolerance parameter $s$.
Output: Set of extracted blocks, extraction periods and block destinations.

1  $(\tilde{z}^*, \tilde{y}^*) \leftarrow$ LPSolve(PCPSP($1, T, \mathcal{B}$))
   for $b \in \mathcal{B}$ do
2      $ET_b \leftarrow$ ExpectedTime($b, \tilde{z}^*$)
3      period($b$) $\leftarrow +\infty$
4      destination($b$) $\leftarrow (0)_{d\in\mathcal{D}}$
   end
5  $t_1 \leftarrow 1$
   while $t_1 \leq T$ do
6      $t_2 \leftarrow \min\{t_1 + w - 1, T\}$
7      $B \leftarrow \{b \in \mathcal{B} \mid$ period($b$) $= +\infty,\ ET_b \leq t_2 + s\}$
8      $(z^*, y^*) \leftarrow$ IPSolve(PCPSP($t_1, t_2, B$))
9      if $(z^*, y^*)$ unfeasible then
           return UNFEASIBLE
       end

       for $\{(b,t) \mid b \in B,\ t \in \{t_1, \dots, \min\{t_1 + \Delta - 1, T\}\},\ z^*_{bt} - z^*_{b,t-1} = 1\}$ do
10         period($b$) $\leftarrow t$
11         destination($b$) $\leftarrow (y^*_{bdt})_{d\in\mathcal{D}}$
       end
12     $t_1 \leftarrow t_1 + \Delta$
   end
   return (period, destination)
```

Fig. 2. Algorithm of the hybrid heuristic.

4. Solve the subproblem to select the blocks to be extracted and processed.
5. Stop if the termination condition (time limit or gap) is satisfied or if no feasible solution can be identified, otherwise go to the Step 1.

Fig. 1 depicts a k-th iteration of the algorithm associated with the time-window $\mathcal{T}_k = \{t_1, \dots, t_2\}$ which goes from period $t_1 = 1 + (k-1)\Delta$ to period $t_2 = \min\{1 + (k-1)\Delta + w, T\}$:

1. The algorithm keeps a set E_k of extracted blocks and selects a set $B_k = \{b \in \mathcal{B} - E_k : ET_b \leq k \cdot w + s\}$ of block candidates to extract in the current time-window.
2. The subproblem is solved over blocks B_k and periods $\mathcal{T}_k$. The subproblem's solution is used to set the extraction period and the destination of the scheduled blocks of B_k for the first Δ periods of $\mathcal{T}_k$.
3. The set of scheduled blocks E_{k+1} is updated.
4. The time-window moves forward by Δ periods.

A more detailed description of the hybrid heuristic algorithm is presented in Fig. 2. As inputs, the algorithm takes (i) a PCPSP instance, and (ii) the parameters w, Δ and s. The output of the algorithm is the production schedule, i.e., the set of extracted blocks, their extraction periods and their destinations.

At Step 1 the LP relaxation of the PCPSP model is solved with the Bienstock-Zuckerberg algorithm [11], and the optimum value of the continuous decision variables $\tilde{z}_{bt}$ and $\tilde{y}_{bdt}$ are returned. For each block $b \in \mathcal{B}$ the expected extraction time is computed (Step 2), and the period and the destination are initialized (Steps 3 and 4). The first time-window is initialized so as to start at period $t_1 = 1$ (Step 5). For any time-window starting at period $t_1 \leq T$, the last period t_2 of the time-window and the set B of preselected blocks are initialized (Steps 6 and 7), the incumbent PCPSP subproblem is solved, and the optimal value of the binary decision z_{bt} and continuous decisions y_{bdt} are returned (Step 8). In Step 9, the algorithm stops with the unfeasibility status if no feasible solution is found for the incumbent subproblem with the current parameters Δ, w, s. Otherwise, the values of z^* and y^* are used to fix the period and the destination of all the blocks scheduled in the first Δ periods of the incumbent time-window (Steps 10 and 11). Finally, a new time-window is initialized so as to start Δ periods later if $t_1 + \Delta \leq T$, otherwise the algorithm ends. Finally, it is worth noting that feasibility of the solutions generated by the heuristic depend on feasibility of the subproblems. If for each time-window the algorithm finds a feasible solution, then feasibility is guaranteed for the monolithic version of the PCPSP. Otherwise, feasibility on a subproblem may be recovered by increasing either the parameter w and by restarting the algorithm from the beginning. Because of this, for the numerical experiments, we embed the heuristic into a solving strategy that adapts the values of Δ and w to make sure it finds a solution. While such strategy may end up trying to solve the monolithic version of the problem, in practice the results show that good feasible solutions are obtained with relatively small values of w.

4.3.1. Parameters setting

As the success of the proposed hybrid heuristic depends on the choice of parameters s, w and Δ, in this section we give a detailed explanation of these parameters and some guidelines for choosing their values.

The procedure used to determine the value of the parameter s is designed to ensure that all time-windows have a sufficient number of preselected blocks to allow a fully use of the maximum capacity. This parameter is also used to limit the risk of infeasibility of the subproblems in instances with minimum resource constraints or blending constraints. If $s = 0$, a block is considered in the subproblem whenever its expected extraction time is less than or equal to the upper limit of the incumbent time-window. Increasing the parameter s allows the consideration of additional blocks with higher expected extraction time than the strict upper limit of the incumbent time-window. Setting its value is a trade-off between feasibility and the number of decision variables.

The length w of the sliding time-window has a great impact on both quality of solution (even feasibility) and computation time. Increasing w allows the algorithm to work in a wider search space or a more diverse set of solutions at once, making it less likely to get trapped in local optima and therefore produces higher quality solutions [24]. Unfortunately, it also implies an increased number of variables in the subproblem as well as a larger computation time. As such, choosing a value for this parameter is a trade-off between quality of solution and computation time.

The parameter Δ is the length of the sliding shift. It is also the number of periods of the incumbent time-window whose respective decision variables will be fixed.

4.3.2. Numerical example

In this section a small numerical example of the PCPSP with only one destination is presented to compare the proposed heuristic with a pure rolling horizon heuristic. Fig. 3 describes a small 2-D block model with the value of each block. We assume a slope angle of 45^o, a tonnage of 1 for all blocks, an extraction capacity of 4 blocks per period, and a 10% discount rate.

The pure rolling horizon heuristic is used with parameters $w = 1$ and $\Delta = 1$, then the generated solution starts with the small pit on the left of the block model, then moves to the right (see Fig. 4). The cumulative discounted objective values obtained for the three periods are 2.00, 3.36 and 6.25, respectively.

The hybrid heuristic is applied on the same block model with the parameters $w = 1$, $\Delta = 1$ and $s = 0.5$. First, the linear relaxation is computed and a LP upper bound of 6.58 is obtained. Then the expected extraction time values are computed as shown in Fig. 5. In the first subproblem 6 blocks with expected extraction times lower than $w + s = 1.5$ are preselected. The resulting production schedule is as shown in Fig. 6. The cumulative discounted objective values obtained for the three periods are 1.50, 4.68 and 6.33, respectively.

-1	-1	-1	-1	-1	-1	-1
-1	5	-1	-1	4.5	4.5	-1

Fig. 3. 2-D block model with block values.

1	1	1	2	2	2	3
	1			2	3	

Fig. 4. Extraction periods with the pure rolling horizon heuristic.

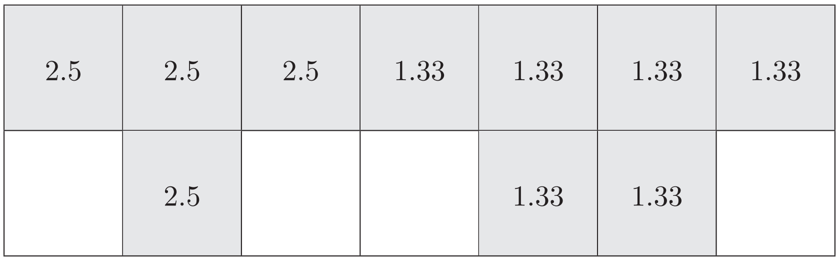

Fig. 5. Expected extraction times.

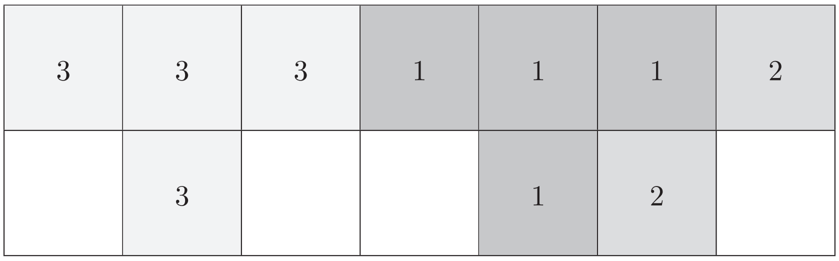

Fig. 6. Extraction periods with the proposed heuristic.

A 3.79% gap solution is obtained with the proposed heuristic compared with a 4.98% gap for the pure rolling horizon approach. We observe in this example that the pure rolling horizon heuristic tends to act in a more greedy fashion than the proposed heuristics guided by the expected extraction times.

It is worth noting that the proposed heuristic also tends to run faster than the pure rolling horizon heuristic. Indeed, for each subproblem the proposed heuristic considers fewer blocks than the pure rolling horizon heuristic. In the first subproblem of this example, 14 blocks are considered in the pure rolling horizon heuristic when only 6 blocks are considered in the proposed heuristic (see Fig. 5).

Table 1
MineLib PCPSP instances.

Instance Name	$\|\mathcal{B}\|$	#Precedence	T	$\|\mathcal{D}\|$	$\|\mathcal{R}\|$
`Newman1`	1060	3922	6	2	2
`Zuck small`	9400	145,640	20	2	2
`KD`	14,153	219,778	12	2	1
`Zuck medium`	29,277	1,271,207	15	2	2
`Marvin`	53,271	650,631	20	2	2
`W23`	74,260	764,786	12	4	7
`Zuck large`	96,821	1,053,105	30	2	2
`SM2`	99,014	96,642	30	2	2
`McLaughlin limit`	112,687	3,035,483	15	2	1
`McLaughlin`	2,140,342	73,143,770	20	2	1

5. Computational results

This section presents the details regarding the implementation of the proposed heuristic and the results obtained on a set of instances from the MineLib library [20].

5.1. Computational setting

The proposed algorithm was coded in C++ with Gurobi 6.5.2 API and executed on a 64-bit Windows OS workstation with ten 2.6 GHz Intel Xeon E5 2660 v3 processors and 120 Gb RAM. To solve the partitioning problem, we implemented the BZ algorithm [11] for the integer relaxation with the default Gurobi parameters and our own implementation of the pseudo-flow algorithm [6] to solve the sub-problems. The only Gurobi parameters set differently from the defaults were `TimeLimit=20,000` and `MIPGap=0.01` to solve each subproblem.

The instances come from the publicly available MineLib library [20]. A complete list of the MineLib instances is presented in Table 1. For each instance, Table 1 gives the number $|\mathcal{B}|$ of blocks, the number of precedences, the number T of periods, the number $|\mathcal{D}|$ of destinations and the number $|\mathcal{R}|$ of operational resources.

As mentioned before, the hybrid heuristic may fail to find a feasible solution, and for this reason a solving strategy is used as a subroutine. The solving strategy first sets the parameter s to ensure that all time-windows have a sufficient number of preselected blocks at each iteration. This parameter is set such as to have the number of blocks $|B_k|$ considered in the subproblem k as close as possible than $w|\mathcal{B}|/T$ to better distribute the considered blocks among the subproblems. The parameter s is initially set to 0.5 and increased whenever the number of blocks $|B_k|$ is less than $w|\mathcal{B}|/T$ while there are sufficiently blocks not yet scheduled (i.e., $|\mathcal{B}-E_k| \geq w|\mathcal{B}|/T$). The parameter s is set as following:

```
$s \leftarrow 0.5$
while $|B_k| < w|\mathcal{B}|/T$ and $|\mathcal{B} - E_k| > w|\mathcal{B}|/T$ do
    $s \leftarrow s + 1.0$
end
```

The parameters w and Δ are initially set to $w = \Delta = 2$. They are both increased by 2 whenever a gap greater than 1% is found for the global problem, or the subroutine reports an infeasibility. In such case, the solution process to generate a feasible solution is restarted from the first time-window.

5.2. Results and discussion

Table 2 shows the set of values for w, Δ and s at each iteration, with a maximum computation time limit of 36,000 s for the complete process. The algorithm returns the best feasible solution found whenever the 1% gap criterion is satisfied or the time limit is reached. For each of the MineLib instances we report: (i)

Table 2
Detailed results of the proposed heuristic with $w = \Delta$ and s for each iteration (values in bold correspond to best solutions found)

Instance			Results					
Name	Parameters		LP upper	LP relaxation	Heuristic	Cumulative	Objective	GAP
	$w = \Delta$	s	bound	Time (s)	Time (s)	Time (s)	value	%
`Newman1`	2	0.5	24,486,549	1	6	7	24,150,103	1.37
	4	**0.5**			10	**17**	**24,175,453**	**1.27**
	6	0.5			19	36	24,163,607	1.32
`Zuck small`	2	2.5	905,878,194	85	190	275	833,155,317	8.03
	4	0.5			554	829	842,284,279	7.02
	6	**0.5**			5557	**6,406**	**898,931,342**	**0.77**
`KD`	2	1.5	410,891,003	54	66	120	390,773,878	4.90
	4	0.5			765	885	383,404,031	6.69
	6	**0.5**			4335	**5,240**	**409,319,677**	**0.38**
`Zuck medium`	**2**	**1.5**	750,519,188	251	25,755	**26,006**	**713,051,791**	**4.99**
`Marvin`	2	2.5	911,704,801	92	133	225	830,130,522	8.95
	4	0.5			366	591	864,880,388	5.14
	6	**0.5**			2419	**3,010**	**904,519,813**	**0.79**
`W23`	**2**	**0.5**	387,678,103	25,134	1458	**26,592**	**380,861,353**	**1.76**
`Zuck large`	**2**	**1.5**	57,938,798	3771	25,739	**29,510**	**56,426,079**	**2.61**
`SM2`	**2**	**1.5**	1,652,393,887	1679	25	**1,704**	**1,650,878,860**	**0.09**
`McLaughlin lim`	**2**	**0.5**	1,324,829,834	1884	4179	**6,063**	**1,321,480,663**	**0.25**
`McLaughlin`	**2**	**0.5**	1,512,971,541	6762	9354	**16,116**	**1,511,899,590**	**0.07**

Table 3
Comparison between the previously reported solutions and the solutions obtained with the proposed heuristic.

Instance	LP upper bound	Best-known solution		Proposed heuristic		
		Source	GAP (%)	Objective value	GAP (%)	Time (s)
`Newman1`	24,486,549	KLET17	1.58	**24,175,453**	**1.27**	17
`Zuck small`	905,878,194	KLET17	1.64	**898,931,342**	**0.77**	6406
`KD`	410,891,003	EGMN13	0.98	**409,319,677**	**0.38**	5240
`Zuck medium`	750,519,188	KLET17	3.00	713,051,791	4.99	26,006
`Marvin`	911,704,801	KLET17	1.61	**904,519,813**	**0.79**	3010
`W23`	387,678,103	–	100.00	***380,861,353**	***1.76**	26,592
`Zuck large`	57,938,798	EGMN13	1.04	56,426,079	2.61	29,510
`SM2`	1,652,393,887	EGMN13	0.12	**1,650,878,861**	**0.09**	1704
`McLaughlin lim`	1,324,829,834	EGMN13	0.24	1,321,480,663	0.25	6063
`McLaughlin`	1,512,971,541	EGMN13	0.19	**1,511,899,590**	**0.07**	16,116

the name of the instance and the parameters (w, Δ, s); (ii) the LP upper bound found with the BZ algorithm [11] and (iii) the times to compute each step of the proposed heuristic (the computation time to solve the LP relaxation and the time to find a feasible solution for the global problem) and the cumulative times from the LP relaxation, (iv) the objective values and (v) the optimality gaps of the feasible solution found relative to LP upper bound.

For example, for the instance `Zuck small`, 85 s are needed to solve the LP relaxation with the BZ algorithm, then 190 s to generate a solution with a 8.03% gap with $w = \Delta = 2$, then 554 additional s to generate a solution with a 7.02% gap with $w = \Delta = 4$. Because neither the time limit was reached nor the 1% gap criterion was satisfied, the parameters w and Δ were increased to 6, for which the algorithm generated a 0.77% gap solution in 5557 additional seconds before stopping, for a total of 6,406 s.

Table 3 shows the main results for the PCPSP instances and a comparison with the best-known results from literature. For each case we report: (i) the LP upper bound; (ii) the source of the current best-known solution: EGMN13 corresponds to Espinoza et al. [20] and KLET17 to Kenny et al. [24]; (iii) the best-known optimality gap; (iv) the objective value obtained with the proposed heuristic; (v) its optimality gap; and (vi) the total solution time in seconds (wall-clock time including the preprocessing step that solves the LP relaxation of the PCPSP and the heuristic's iterations up to find a feasible solution to the complete problem).

It is noteworthy that no feasible solution has been previously reported for the instance `W23`. This instance is precisely the only MineLib instance that considers general side constraints (blending requirements): it should be recalled that neither the TopoSort heuristic used in Espinoza et al. [20] nor the Greedy Randomized Adaptive Search Procedure (GRASP) proposed in Kenny et al. [24] consider this kind of constraints.

While running the proposed heuristic, a feasible solution with 1.76% optimality gap has been found for the instance `W23`. An improvement is also observed for 6 of the 9 other instances (in bold without asterisk in Table 3), with a 45.5% gap reduction on average among these 6 instances, the gap reduction being computed as:

$$\frac{\text{GAP Best-known solution (\%)} - \text{GAP Proposed heuristic (\%)}}{\text{GAP Best-known solution (\%)}}.$$

Among all instances we observe a 14.98% gap reduction on average when the instance `W23` is included, and a 5.72% gap reduction on average when it is excluded.

Regarding the parameters setting, the tolerance parameter s has been increased for the instances `Zuck small`, `KD`, `Zuck medium`, `Marvin`, `Zuck large` and `SM2` to obtain a sufficient quantity of blocks in the first iteration ($w = \Delta = 2$). To satisfy the gap criterion described in Section 5.1, it was necessary to increase the length w of the sliding time-window for the instances `Newman1`, `Zuck small`, `KD` and `Marvin` (see Table 2).

The more time-consuming instances are `Zuck medium`, `W23` and `Zuck large`. For these instances, the gap criterion was not satisfied when using the default values of the parameters w and Δ within the time limit set at 36,000 s (see Appendix B). These parameters had to be increased applying the rules proposed in Section 4.3.1. For the instances `Zuck medium` and `Zuck large`, most of the CPU time is used to find a feasible solution while, for

Table 4
Comparison of a pure rolling horizon approach with a rolling horizon heuristic including the block preselection procedure (i.e. hybrid heuristic) (values in bold correspond to better solutions).

Instance	LP upper bound	Pure rolling horizon			Proposed heuristic		
		Objective value	GAP (%)	Time (s)	Objective value	GAP (%)	Time (s)
`Newman1`	24,486,549	24,131,200	1.45	26	24,175,453	**1.27**	17
`Zuck small`	905,878,194	854,062,473	5.72	11,155	898,931,342	**0.77**	6406
`KD`	410,891,003	382,066,711	7.02	3741	409,319,677	**0.38**	5240
`Zuck medium`	750,519,188	–	100.00	–	713,051,791	**4.99**	26,006
`Marvin`	911,704,801	857,780,482	5.91	4110	904,519,813	**0.79**	3010
`W23`	387,678,103	338,226,394	12.76	13,203	380,861,353	**1.76**	26,592
`Zuck large`	57,938,798	–	100.00	–	56,426,079	**2.61**	29,510
`SM2`	1,652,393,887	1,621,601,503	1.86	107	1,650,878,861	**0.09**	1704
`McLaughlin lim`	1,324,829,834	1,210,454,961	8.63	14,553	1,321,480,663	**0.25**	6063
`McLaughlin`	1,512,971,541	–	100.00	–	1,511,899,590	**0.07**	16,116

the instance `W23`, most time is used to solve the LP relaxation (see Table 2 for more details).

As a final comment, the reported results were found by following the procedure according to Section 4.3.1, obtaining better results for 7 out of 10 PCPSP instances from MineLib.

In Appendix C, additional results with $\Delta = 1$ and various values of the parameter w are presented.

5.2.1. *Added value from the block preselection procedure*

In Section 4.3.2 a numerical example has been proposed to compare the pure rolling horizon heuristic with the proposed heuristic. In this section we propose the same comparison with all the instances of the MineLib library.

Table 4 shows the results obtained with both approaches while using the parameters w and Δ stated in Table 2. For each, we present the objective value (in dollars), the relative gap (as percentage) and the computation time (in seconds).

With a computing time limit set to 36,000 s, the pure rolling horizon approach is able to provide feasible solutions for 7 out of 10 MineLib instances, and the heuristic runs out of time for 3 remaining instances. Comparing the results of the pure rolling horizon approach with the best known solutions, better results are obtained while applying the pure rolling horizon approach to instances `Newman1` and `W23`. The pure rolling horizon approach is able to find a 12.76% gap solution to instance `W23`, for which no solutions have been previously reported to date. However, no feasible solutions are found for instances `Zuck medium`, `Zuck large` and `McLaughlin`.

Compared with the pure rolling horizon heuristic, the proposed hybrid heuristic finds better results for all instances, with an average of 79.8% gap reduction for the instances for which a feasible solution is found with the pure rolling horizon heuristic.

6. Conclusions

We introduce a hybrid heuristic algorithm using a block preselection procedure based on expected extraction times to solve the PCPSP. The problem is decomposed into smaller and easier subproblems on a rolling horizon basis, where a reduced set of blocks is preselected according to the LP relaxation solution of the complete problem. When applied to the all PCPSP instances of the MineLib library without blending constraints, the results obtained by the proposed heuristic show a significant improvement for 6 out of 9 pre-existing results reported by Espinoza et al. [20] and Kenny et al. [24]. This heuristic is also able to handle blending constraints, a special kind of general side constraints of the PCPSP, which is not the case of both TopoSort heuristic [20] nor GRASP algorithm [24]. The proposed hybrid heuristic was able to generate the first feasible solution to date for the only instance with blending constraints, i.e. `W23`, with a 1.76% optimality gap.

As a future direction of research it would be interesting to improve the expected extraction times by strengthening the formulation for the initial integer relaxation and the subproblems, for example by adding clique cuts as suggested by [9] and [23].

Acknowledgements

E. Jélvez, N. Morales and P. Nancel-Penard were supported by CONICYT Basal Project FB0809 – Advanced Mining Technology Center – Universidad de Chile.

Appendix A. An extended approach for pure rolling horizon

An extension of the pure rolling horizon approach is presented by [16], where the complete problem is solved by using (i) fixed variables in early time periods, (ii) an exact submodel defined over a window of middle time periods, and (iii) a relaxed submodel in later time periods, where a Lagrangian relaxation is used for the capacity constraints. Table 5 shows the results on the MineLib instances when this heuristic is applied. For 8 out of 10 PCPSP instances the algorithm is unable to generate a feasible solution, either because they exceeded the limit of one-day computation time (7 instances) or because the maximum available memory was insufficient (`McLaughlin`). Only 2 out of 10 instances were solved: `Newman` for which a lower gap is obtained, and `SM2` for which the obtained solution improves the current best-known solution.

Appendix B. Detailed results of the proposed heuristic with $w = \Delta$

Table 6 shows the detailed results for each of the MineLib instances with $w = \Delta$. For each case we report: (i) the name of the instance and the parameters w, Δ and s; (ii) the relative gaps (compared to the LP relaxation objective value given by the BZ algorithm) and the cumulative times (in seconds) which include the computation time for the LP relaxation of the complete problem, the computation time of the expected extraction times, and the computation time to solve the subproblems within a total time limit of 36,000 s. We also provide the number of blocks, the number of precedence constraints, the number of variables and the total number of constraints for the smallest, the average and the largest subproblems. With $w = \Delta$, the extraction times obtained for the preselected blocks of each subproblem are part of the final global solution.

In general, there is a significant variability in both size and solving-time among the subproblems, specially between the first and the last subproblem as the last ones contains few blocks. For

Table 5
Comparison of the previously reported solutions with the solutions of the extended approach for the pure rolling horizon proposed by [16] considering a sliding-time window with $w = 2$ (values in bold correspond to new best-known solutions).

Instance	LP upper bound	Best-known solution		[16]		
		Source	GAP (%)	Objective value	GAP (%)	Time (s)
Newman1	24,486,549	KLET17	1.27	24,149,615	1.38	49
Zuck small	905,878,194	KLET17	1.64	–	100.00	> 1 day
KD	410,891,003	EGMN13	0.98	–	100.00	> 1 day
Zuck medium	750,519,188	KLET17	3.00	–	100.00	> 1 day
Marvin	911,704,801	KLET17	1.61	–	100.00	> 1 day
W23	387,678,103	–	100.00	–	100.00	> 1 day
Zuck large	57,938,798	EGMN13	1.04	–	100.00	> 1 day
SM2	1,652,393,887	EGMN13	0.12	**1,650,662,943**	**0.10**	22,261
McLaughlin lim	1,324,829,834	EGMN13	0.24	–	100.00	> 1 day
McLaughlin	1,512,971,541	EGMN13	0.19	–	100.00	OOM

Table 6
Detailed results of the proposed heuristic with $w = \Delta$ (values in bold correspond to new best-known solutions).

Instance			Results		Subproblems											
Name	$w = \Delta$	s	Gap (%)	Cumulative Time (s)	smallest				average				largest			
					#blocks	#precs	#vars	#constrs	#blocks	#precs	#vars	#constrs	#blocks	#precs	#vars	#constrs
Newman1	2	0.5	1.37	7	296	903	1776	3294	643	2333	3855	7886	989	3762	5934	12,477
	4	**0.5**	**1.27**	**17**	1059	3922	12,708	27,353	1059	3922	12,708	27,353	1059	3922	12,708	27,353
	6	0.5	1.32	31	1059	3922	19,062	41,559	1059	3922	19,062	41,559	1059	3922	19,062	41,559
Zuck small	2	2.5	8.03	275	1270	6950	7620	20,388	1738	12,534	10,428	33,766	3047	35,070	18,282	85,383
	4	0.5	7.02	829	1541	9166	18,492	54,307	2351	19,395	28,209	103,454	3047	35,070	36,564	173,813
	6	**0.5**	**0.77**	**6406**	1807	9166	32,526	85,739	3134	30,382	56,418	235,600	4304	42,955	77,472	330,922
KD	2	1.5	4.90	120	658	2028	3948	7350	2500	25,697	14,999	63,898	3675	49,332	22,050	117,043
	4	0.5	6.69	885	2173	13,148	26,076	76,503	4065	47,914	48,780	236,380	6398	90,644	76,776	432,962
	6	**0.5**	**0.38**	**5240**	3830	29,394	68,940	241,486	6077	75,723	109,386	557,659	8324	122,052	149,832	873,832
Zuck medium	2	1.5	4.99	36,000	1913	10,255	11,478	30,083	5425	123,076	32,549	273,284	13,586	468,845	81,516	1,005,628
Marvin	2	2.5	8.95	225	1151	3613	6906	13,454	1543	7927	9258	23,577	2723	22,126	16,338	57,875
	4	0.5	5.14	591	1244	3613	14,928	28,152	2129	12,581	25,548	73,759	2914	22,126	34,968	118,473
	6	**0.5**	**0.79**	**3010**	1244	3613	22,392	42,850	2839	19,300	51,096	164,079	4065	27,426	73,170	233,685
W23	**2**	**0.5**	**1.76**	**36,000**	14	4	140	92	7808	61,489	78,083	162,036	17,591	149,678	175,910	382,389
Zuck large	2	1.5	2.61	36,000	4583	13,401	27,498	49,725	9460	65,324	56,758	177,953	14,517	134,761	87,102	342,115
SM2	**2**	**1.5**	**0.09**	**1704**	1188	1107	7128	8162	1770	1607	10,623	12,074	2560	2224	15,360	17,256
McLaughlin lim	2	0.5	0.25	6063	846	2980	5076	10,194	15,824	266,664	94,945	612,453	28,880	681,500	173,280	1,507,404
McLaughlin	**2**	**0.5**	**0.07**	**16,116**	13,952	114,911	83,712	305,916	22,594	365,484	135,562	843,939	35,115	828,562	210,690	1,832,703

example, the instance Zuck large presents the most significant solving-time variability: in this instance a subproblem reached the 20,000 s time limit and the subproblem in the last time window was solved in 1 second, with an average time per subproblem of 2139 s. On the other side, some instances present smaller variability, such as SM2, where all subproblems need a short solving-time, with a minimum of 0.2 s, a maximum of 6.5 s and an average of 2.4 s, or the instance W23, with a minimum time for solving a subproblem of 1 s, other with a maximum of 623 s and an average of 291 s.

The most important results from Table 6 are summarized in Table 3. An improvement is observed in 7 out of 10 cases when comparing to the best-known results from literature, including an instance for which the first feasible solution has been reported (W23 with a gap of 1.76%).

In the instances Zuck medium, W23 and Zuck large the total time spent to find the feasible solution for $w = 2,\ \Delta = 2$ was 26,006, 26,592 and 29,510 s, respectively. The remaining time to complete 36,000 s time limit was trying to find a feasible solution for the parameters $w = 4,\ \Delta = 4$.

Appendix C. Detailed results of the proposed heuristic with $w \geq 1$ and $\Delta = 1$

Similarly to Appendix B, Table 7 shows the detailed results for each of the MineLib instances with $w \geq 1$ and $\Delta = 1$.

When comparing with the performance obtained from the proposed heuristic with $w = \Delta$, the results show a significant improvement on the instance W23 for $w = 2,\ \Delta = 1$ and $s = 0.5$, with a gap of 0.74% (i.e. a 58% gap reduction). Other instances as Newman1, Marvin, SM2 and McLaughlin lim present similar gaps, but a poorer performance is obtained for instances Zuck small, KD and McLaughlin (respectively 30%, 79% and 243% higher gap).

Similar to the case with parameters $w = \Delta$, there is a significant variability in both size and solving-time among the subproblems. Since in this case more iterations have to be performed, for those instances where the gaps are similar (Newman1, Marvin, SM2 and McLaughlin lim) the total time is longer.

For the instances Zuck medium and Zuck large a first intent was done with parameters $w = 1,\ \Delta = 1$. As the time spent was below the time limit and the optimality gap was greater than 1%, a new intent has been done with $w = 2, \Delta = 1$. As the time limit of 36,000 s was reached the gap being still greater than 1%, the result of the previous intent (i.e., with parameters $w = 1, \Delta = 1$) has been retained.

As an attempt to find better solutions given the time available, in the instances SM2, McLaughlin lim and McLaughlin a new iteration was performed trying to improve the available gap, which was already less than 1%. The results are shown in Table 7. For SM2 the solution found for $w = 3,\ \Delta = 1$ results in a higher gap (0.21%). In the case of McLaughlin lim the solution found is the same

Table 7
Detailed results of the proposed heuristic with $w \geq 1$ and $\Delta = 1$ (values in bold correspond to new best-known solutions).

Instance			Results		Subproblems											
Name	w	s	Gap (%)	Cumulative Time (s)	smallest				average				largest			
					#blocks	#precs	#vars	#constrs	#blocks	#precs	#vars	#constrs	#blocks	#precs	#vars	#constrs
Newman1	1	1.5	1.27	6	287	986	861	1564	643	2345	1929	3635	989	3762	2967	5744
	2	0.5	1.56	19	288	1015	1728	3478	643	2354	3860	7933	989	3762	5934	12,477
	3	0.5	1.47	46	287	979	2583	5245	665	2423	5985	12,600	1059	3922	9531	20,250
	4	**0.5**	**1.27**	**68**	286	985	3432	7102	666	2397	7992	16,930	1059	3922	12,708	27,353
	6	0.5	1.32	92	1059	3922	19,062	41,559	1059	3922	19,062	41,559	1059	3922	19,062	41,559
Zuck small	1	3.5	12.18	140	689	2044	2067	3426	1197	7850	3592	10,249	3047	35,070	9141	41,168
	2	2.5	7.23	414	771	2449	4626	8761	1617	10,721	9700	29,532	3047	35,070	18,282	85,383
	3	1.5	4.85	871	780	2439	7020	13,569	2004	14,423	18,037	59,314	3047	35,070	27,423	129,598
	4	0.5	2.55	2194	631	1639	7572	13,513	2341	18,382	28,095	99,298	3319	35,070	39,828	173,813
	6	**0.5**	**1.00**	**17,078**	537	1383	9666	17,451	3342	31,970	60,162	248,666	4770	47,196	85,860	364,290
KD	1	2.5	8.24	113	943	3789	2829	5677	2387	24,501	7160	29,276	3980	56,709	11,940	64,671
	2	1.5	4.58	291	612	1907	3672	6878	2598	26,044	15,591	65,084	4265	60,102	25,590	141,533
	3	**0.5**	**0.68**	**2419**	573	1825	5157	10,065	3248	35,632	29,228	132,883	4883	70,325	43,947	250,045
Zuck medium	1	2.5	9.23	36,000	993	7819	2979	9809	4407	95,142	13,222	103,961	13,586	468,845	40,758	496,021
*Marvin	1	3.5	11.75	137	277	441	831	999	1099	5198	3298	7400	2723	22,126	8169	27,576
	2	2.5	7.94	375	273	431	1638	2235	1421	6754	8525	20,621	2723	22,126	16,338	57,875
	3	1.5	3.75	739	277	441	2493	3551	1839	9655	16,551	43,690	2723	22,126	24,507	88,174
	4	0.5	2.24	1330	276	439	3312	4808	2149	12,033	25,784	71,784	3217	22,126	38,604	118,473
	6	**0.5**	**0.77**	**9652**	277	441	4986	7379	3066	20,168	55,189	173,157	4541	31,194	81,738	264,385
W23	1	1.5	11.00	28,133	295	453	1475	1063	4290	30,002	21,452	38,597	16,601	149,678	83,005	182,894
	2	**0.5**	**0.74**	**33,001**	95	128	950	745	6934	52,676	69,336	140,045	22,281	199,195	222,810	509,823
Zuck large	1	2.5	2.67	36,000	2809	11,929	8427	17,551	7694	51,749	23,083	67,141	14,484	134,604	43,452	163,576
SM2	1	2.5	4.09	1848	384	363	1152	1135	2133	1992	6400	6263	2756	2573	8268	8026
	2	**1.5**	**0.09**	**1876**	1188	1107	7128	8162	1770	1607	10,623	12,074	2560	2224	15,360	17,256
	3	0.5	0.21	1977	1441	1353	17,292	21,279	3129	2874	37,548	45,930	4222	3815	50,664	61,718
McLaughlin lim	1	0.5	0.87	2536	846	2980	2538	4674	8521	124,831	25,562	141,875	15,861	371,169	47,583	402,893
	2	0.5	0.25	10,999	560	2029	3360	6862	15,460	242,441	92,760	562,185	28,880	681,500	173,280	1,507,404
McLaughlin	1	0.5	0.85	8374	3139	10,806	9417	17,086	11,297	156,372	33,891	178,968	19,047	378,204	57,141	416,300
	2	0.5	0.24	24,948	12,958	114,385	77,748	304,584	22,600	356,467	135,597	825,935	37,699	851,514	226,194	1,891,527

as the solution with parameters $w = 2,\ \Delta = 2$ (0.25%), but taking twice as long. Finally, for the instance `McLaughlin` the solution (gap 0.24%) is worse than the one obtained with parameters $w = 2$, $\Delta = 2$ (gap 0.07%) and takes almost twice as long.

References

[1] Lerchs H, Grossmann H. Optimal design of open-pit mines. Trans CIM 1965;58:47–54.
[2] Picard J-C. Maximal closure of a graph and applications to combinatorial problems. Manage Sci. 1976;22(11):1268–72.
[3] Ibarra O, Kim C. Approximation algorithms for certain scheduling problems. Math Oper Res 1978;3(3):197–204.
[4] Boyd E. Polyhedral results for the precedence-constrained knapsack problem. Discrete Appl Math 1993;41(3):185–201.
[5] Underwood R, Tolwinski B. A mathematical programming viewpoint for solving the ultimate pit problem. Eur J Oper Res 1998;107(1):96–107.
[6] Hochbaum D. The pseudoflow algorithm: a new algorithm for the maximum-flow problem. Oper Res 2008;56(4):992–1009.
[7] Caccetta L, Hill S. An application of branch and cut to open-pit mine scheduling. J Global Optim 2003;27:349–65.
[8] Moreno E, Espinoza D, Goycoolea M. Large-scale multi-period precedence constrained knapsack problem: a mining application. Electronic Notes in Discrete Mathematics 2010;36:407–14.
[9] Bley A, Boland N, Fricke C, Froyland G. A strengthened formulation and cutting planes for the open pit mine production scheduling problem. Comput Oper Res 2010;37(9):1641–7.
[10] Shishvan MS, Sattarvand J. Long term production planning of open pit mines by ant colony optimization. Eur J Oper Res 2015;240(3):825–36.
[11] Bienstock D, Zuckerberg M. Solving lp relaxations of large-scale precedence constrained problems. In: International Conference on Integer Programming and Combinatorial Optimization. Springer; 2010. p. 1–14.
[12] Chiles J, Delfiner P. Geostatistics: modeling spatial uncertainty, 497. John Wiley & Sons; 2009.
[13] Jélvez E, Morales N, Askari-Nasab H. A new model for automated pushback selection. Computers & Operations Research 2018.
[14] Johnson T. Optimum open-pit mine production scheduling. Operations Research Department, University of California, Berkeley; 1968.
[15] Ramazan S. The new fundamental tree algorithm for production scheduling of open pit mines. Eur J Oper Res 2007;177(2):1153–66.
[16] Cullenbine C, Wood R, Newman A. A sliding time window heuristic for open pit mine block sequencing. Optim Lett 2011;5(3):365–77.
[17] Chicoisne R, Espinoza D, Goycoolea M, Moreno E, Rubio E. A new algorithm for the open-pit mine production scheduling problem. Oper Res 2012;60(3):517–28.
[18] Jélvez E, Morales N, Nancel-Penard P, Peypouquet J, Reyes P. Aggregation heuristic for the open-pit block scheduling problem. Eur J Oper Res 2016;249(3):1169–77.
[19] Samavati M, Essam D, Nehring M, Sarker R. A new methodology for the open-pit mine production scheduling problem. Omega 2018;81:169–82.
[20] Espinoza D, Goycoolea M, Moreno E, Newman A. Minelib: a library of open pit mining problems. Ann Oper Res 2013;206(1):93–114.
[21] Lamghari A, Dimitrakopoulos R, Ferland J. A hybrid method based on linear programming and variable neighborhood descent for scheduling production in open-pit mines. J Global Optim 2015;63(3):555–82.
[22] Liu S, Kozan E. New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems. Expert Syst Appl 2016;43:59–65.
[23] Samavati M, Essam D, Nehring M, Sarker R. A methodology for the large-scale multi-period precedence-constrained knapsack problem: an application in the mining industry. Int J Prod Econ 2017;193:12–20.
[24] Kenny A, Li X, Ernst A, Thiruvady D. Towards solving large-scale precedence constrained production scheduling problems in mining. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM; 2017. p. 1137–1144.
[25] Lambert W, Brickey A, Newman A, Eurek K. Open-pit block-sequencing formulations: a tutorial. Interfaces 2014;44(2):127–42.
[26] Jélvez E, Morales N, Nancel-Penard P. Open-pit mine production scheduling: Improvements to minelib library problems. In: Proceedings of the 27th International Symposium on Mine Planning and Equipment Selection-MPES 2018. Springer; 2019. p. 223–32.
[27] Lambert W, Newman A. Tailored lagrangian relaxation for the open pit block sequencing problem. Ann Oper Res 2014;222(1):419–38.