
Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

# Production Scheduling for Strategic Open Pit Mine Planning: A Mixed-Integer Programming Approach

Orlando Rivera Letelier, Daniel Espinoza, Marcos Goycoolea, Eduardo Moreno, Gonzalo Muñoz

To cite this article:
Orlando Rivera Letelier, Daniel Espinoza, Marcos Goycoolea, Eduardo Moreno, Gonzalo Muñoz (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

http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 68, No. 5, September–October 2020, pp. 1425–1444
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Crosscutting Areas

# Production Scheduling for Strategic Open Pit Mine Planning: A Mixed-Integer Programming Approach

**Orlando Rivera Letelier,[a] Daniel Espinoza,[b] Marcos Goycoolea,[c] Eduardo Moreno,[d] Gonzalo Muñoz[e]**

[a] Doctoral Program in Industrial Engineering and Operations Research, Universidad Adolfo Ibáñez, Santiago, Chile 7941169; [b] Google, Cambridge, Massachusetts 02142; [c] School of Business, Universidad Adolfo Ibáñez, Santiago, Chile 7941169; [d] Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Santiago, Chile 7941169; [e] Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile 2820000

**Contact:** orlando.rivera@uai.cl, https://orcid.org/0000-0003-1966-1565 (ORL); danielespinoza@google.com (DE); marcos.goycoolea@uai.cl, https://orcid.org/0000-0003-1904-7215 (MG); eduardo.moreno@uai.cl (EM); gonzalo.munoz@uoh.cl (GM)

**Received:** April 12, 2017
**Revised:** April 2, 2018; August 22, 2019
**Accepted:** September 30, 2019
**Published Online in Articles in Advance:** August 27, 2020

**Subject Classification:** industries: mining: metals
**Area of Review:** Environment, Energy and Sustainability

**https://doi.org/10.1287/opre.2019.1965**



**Abstract.** Given a discretized representation of an ore body known as a block model, the open pit mining production scheduling problem that we consider consists of defining which blocks to extract, when to extract them, and how or whether to process them, in such a way as to comply with operational constraints and maximize net present value. Although it has been established that this problem can be modeled with mixed-integer programming, the number of blocks used to represent real-world mines (millions) has made solving large instances nearly impossible in practice. In this article, we introduce a new methodology for tackling this problem and conduct computational tests using real problem sets ranging in size from 20,000 to 5,000,000 blocks and spanning 20 to 50 time periods. We consider both direct block scheduling and bench-phase scheduling problems, with capacity, blending, and minimum production constraints. Using new preprocessing and cutting planes techniques, we are able to reduce the linear programming relaxation value by up to 33%, depending on the instance. Then, using new heuristics, we are able to compute feasible solutions with an average gap of 1.52% relative to the previously computed bound. Moreover, after four hours of running a customized branch-and-bound algorithm on the problems with larger gaps, we are able to further reduce the average from 1.52% to 0.71%.

**Funding:** This work was supported by Comisión Nacional de Investigación Científica y Tecnológica, ACT 1407, BCH 72130388, Basal CMM Universidad de Chile, Fondo de Fomento al Desarrollo Científico y Tecnológico, 1130681, 1151098. This work was partially funded by the government of Chile through CONICYT [Grants FONDECYT 1151098 (M. Goycoolea, O. R. Letelier), FONDECYT 1161064 (E. Moreno), Basal CMM U. Chile, ANILLO ACT 1407 (M. Goycoolea, E. Moreno, O. R. Letelier), and BCH 72130388 (G. Muñoz)].
**Supplemental Material:** The e-companion is available at https://doi.org/10.1287/opre.2019.1965.

**Keywords:** open pit mining • production scheduling • column generation • heuristics • cutting planes • integer programming applications

## 1. Introduction

The central concern of strategic open pit mine planning is the construction of a tentative production schedule. This is a life-of-mine plan (20–50 years) specifying which part of a mineral deposit should be extracted, as well when and how, so as to maximize the value of the mining project while satisfying operational and economic constraints. In the early stages of a mining endeavor, a strategic production schedule provides insight into the cash flows of a project (e.g., capacity investments and production goals), and as such, it is critical to investors. Moreover, early strategic decisions in a mine planning project are binding for and critical to long-term profits. It is widely acknowledged that, amongst all stages in a mining project, strategic mine planning has the most significant impact on the overall costs (Hustrulid and Kuchta 2006).

Strategic mine planning begins with the construction of a discretized mathematical representation of the ore body, known as a *block model*. This is typically a three-dimensional array of units of equal size, called *blocks*. The horizontal levels of this array are referred to as *benches*. To each block, a number of geological attributes are assigned based on data retrieved from exploration drill holes, including mineral and contaminant grades, rock types, and rock density.

A strategic mine plan, or production schedule, specifies which blocks are extracted, their time of extraction, and their destination once they have been extracted. Destinations represent not only physical locations, but also block processing and treatment options. Examples of block destinations include mills, waste dumps, leach pads, and stockpiles.

Amongst others, production schedules must comply with capacity and spatial constraints. While the

former limit the amount of material which can be mined and sent to each destination, the latter ensure the accessibility of blocks and the safety of mining operations. The most common spatial constraints consist of imposing minimal wall-slope angles (to prevent cave-ins or wall-slope failures) and ensuring that extraction regions are wide and contiguous so as to accommodate ramps and roads for large equipment. Additional constraints are often imposed to address blending, stockpiling, and other concerns.

In practice, given the complexity of holistic approaches, the strategic open pit mine planning problem is subdivided into two more manageable problems. The first subproblem, which we term the open pit phase design problem (OPPDP), involves subdividing the deposit into large contiguous spatial units known as *phases* or *pushbacks*. Phases define the stages in which the deposit is to be excavated, and they are designed such that profitable parts of the mine are quickly reached while maintaining operational feasibility. The second subproblem, which we call the open pit production scheduling problem (OPPSP), consists of computing a production schedule detailing when and how the blocks in each phase should be extracted. Phases are extracted bench-by-bench, beginning with the blocks in Phase 1. The extraction of blocks in Phase 2 should begin during or after the extraction of the blocks in Phase 1, and so on. For a more detailed description of this scheme, see Hustrulid and Kuchta (2006).

Starting with the doctoral dissertation of Johnson (1968), a majority of academic papers regarding mathematical programming methods for strategic open pit mine planning have focused on the OPPDP. In addition, most of them have considered a simplification of the OPPDP known as the capacitated pit problem (C-PIT), which considers predefined destinations for each block and simple classes of constraints. Even these simplified problems, however, have proved very challenging considering the scale of practical problem instances and the difficulty of solving even their LP relaxations.

The works of Boland et al. (2009) and, shortly afterward, Bienstock and Zuckerberg (2009, 2010) mark a significant departure from previous results in two regards. First, they introduce a mathematical programming formulation that generalizes both the OPPDP and OPPSP, with block-destination selectivity, block clustering, and arbitrary constraints. Second, they present an effective decomposition method for solving the LP relaxation for problems of realistic size.

In this article, we build on the work of Bienstock and Zuckerberg (BZ) by proposing a framework for solving the problem with integral variables. This framework includes preprocessing, cutting plane, heuristic, and specialized branching techniques. A novel aspect of our work is that we discuss both the phase-design (OPPDP) and production scheduling (OPPSP) steps of the strategic mine planning problem and propose problem-specific techniques for each. Another novel aspect is that we consider realistic, but more complicated, constraints in our computations, including blending, flow-balance, and minimum production. Extensive computational tests show that we can solve both the OPPDP and OPPSP to near-optimality in a very reasonable amount of time. To our knowledge, this is the first time this is achieved.

## 2. Formulating the Phase Design and Production Scheduling Problems

In this section, we describe the precedence-constrained project scheduling problem with clusters (PCPSP-C), along with an integer programming formulation of it. In addition, we show how one can formulate both the OPPDP and OPPSP as a PCPSP-C and survey different methodologies proposed for solving this and similar problems.

### 2.1. Definitions and Notation

Let $\mathcal{B}$ represent the set of blocks of a discretized ore body, and let $\mathcal{D}$ be the set of possible destinations to which a block can be sent. For each $b \in \mathcal{B}$ and $d \in \mathcal{D}$, let $p_{b,d}$ represent the estimated value obtained by sending $b$ to $d$, and let $p_{b,*} = \max\{p_{b,d} : d \in \mathcal{D}\}$, bearing in mind that these values can be negative. Let $\mathcal{T} = \{1, \ldots, T\}$ represent the time periods in which decisions can be made. Let $p_{b,d,t}$ represent the value obtained from sending $b$ to $d$ within time period $t$. We assume that $p_{b,d,t}$ is obtained from $p_{b,d}$ by applying a discount factor $r > 0$ over time, that is, $p_{b,d,t} = p_{b,d}/(1+r)^t$.

Consider $b_1, b_2 \in \mathcal{B}$. Define $b_1$ to be a *precedence* of $b_2$—formally, $b_1 \prec b_2$—if $b_1$ must be extracted no later than $b_2$. Precedence relationships between blocks are imposed to ensure minimum wall-slope angles and, thus, avoid wall-slope failures or cave-ins. See Khalokakaie et al. (2000) for a description of how one determines these precedence relationships in practice.

Given sets $B_1, B_2 \subseteq \mathcal{B}$, we write $B_1 \prec B_2$ if there exist $b_1 \in B_1$ and $b_2 \in B_2$ such that $b_1 \prec b_2$.

Given a set $P \subseteq \mathcal{B}$, we say that $P$ is a *pit* in $\mathcal{B}$ if $b_1 \prec b_2 \ \wedge \ b_2 \in P \Rightarrow b_1 \in P$. Also, a sequence of pits $P_1, P_2, \ldots, P_n \subseteq \mathcal{B}$ is *nested* if $P_1 \subseteq P_2 \subseteq \ldots \subseteq P_n$.

Let $\mathcal{C}$ be a partition of the set of blocks $\mathcal{B}$. That is, the sets in $\mathcal{C}$ are pairwise disjoint, and they cover the entire set $\mathcal{B}$. We henceforth refer to the elements of $\mathcal{C}$ as *clusters*. Assume that $\prec$ induces a partial order over the elements of $\mathcal{C}$. Let $\mathcal{A} \subseteq \mathcal{C} \times \mathcal{C}$ be the arcs representing the directed acyclic graph (DAG) induced by $\prec$ in $\mathcal{C}$. That is, $(c_2, c_1) \in \mathcal{A}$ if and only if $c_1 \prec c_2$. Note that we chose the arc directions such that a cluster's *out-neighbors* are its precedents. For computational

purposes, we assume $\mathcal{A}$ to be the transitive reduction of this arc set. In other words, $\mathcal{A}$ is composed only of *immediate precedence relationships*: if $(c_1, c_2) \in \mathcal{A}$ and $(c_2, c_3) \in \mathcal{A}$, then $(c_1, c_3) \notin \mathcal{A}$.

Given $c \in \mathcal{C}$, we define the *closure* of $c$, or $cl(c)$, and the *reverse closure* of $c$, or $rcl(c)$, as follows:

$$cl(c) = \{c\} \cup \{c' \in \mathcal{C} : c' \prec c\},$$
$$rcl(c) = \{c\} \cup \{c' \in \mathcal{C} : c \prec c'\}.$$

Although each $c \in \mathcal{C}$ is a *set*, we use lowercase letters to represent these elements (as opposed to the usual uppercase notation). This is mainly because we define variables using each $c \in \mathcal{C}$ as a subscript. For each $b \in \mathcal{B}$, let $c(b)$ represent the unique cluster $c \in \mathcal{C}$ such that $b \in c$. We use clusters to indicate that sets of blocks should be extracted simultaneously in a mine-planning solution. That is, all the blocks in a single cluster are extracted in exactly the same proportion in each time period. These clusters only enforce an extraction-related constraint, as the destination of each block can be different from the destination of other blocks in the same cluster.

For each block $b \in \mathcal{B}$, let $q_b$ be its weight (or amount of material). For each $t \in \mathcal{T}$, let $U_t$ represent the maximum amount of material that can be extracted in time period $t$, and let $U_t^d$ represent the maximum amount of material that can be sent to destination $d \in \mathcal{D}$ in time period $t \in \mathcal{T}$.

Let variables $x_{c,t}$ indicate the proportion of cluster $c \in \mathcal{C}$ extracted in time $t \in \mathcal{T}$. For each $b \in \mathcal{B}$, $d \in \mathcal{D}$, and $t \in \mathcal{T}$, let the variable $y_{b,d,t}$ indicate the proportion of block $b$ sent to destination $d$ in time $t$. In (1)–(7), we describe a mixed-integer programming formulation proposed by Bienstock and Zuckerberg (2009), which generalizes both the OPPSP and OPPDP. Although these authors originally refer to this problem as the parcel assignment problem (PAP), we henceforth refer to it as the precedence-constrained production scheduling problem with clusters (PCPSP-C):

$$\max \sum_{b\in\mathcal{B}} \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} p_{b,d,t} y_{b,d,t} \tag{1}$$

$$\text{s.t. } x_{c,t} = \sum_{d\in\mathcal{D}} y_{b,d,t} \qquad \forall c \in \mathcal{C},\ b \in c,\ t \in \mathcal{T}, \tag{2}$$

$$\sum_{t\in\mathcal{T}} x_{c,t} \le 1 \qquad \forall c \in \mathcal{C}, \tag{3}$$

$$\sum_{t'=1}^{t} x_{c,t'} \le \sum_{t'=1}^{t} x_{c',t'} \qquad \forall (c,c') \in \mathcal{A},\ t \in \mathcal{T}, \tag{4}$$

$$Gy \le g, \tag{5}$$

$$y_{b,d,t} \ge 0 \qquad \forall b \in \mathcal{B}, d \in \mathcal{D}, t \in \mathcal{T}, \tag{6}$$

$$x_{c,t} \text{ satisfies an integrality condition} \qquad \forall c \in \mathcal{C}, t \in \mathcal{T}. \tag{7}$$

Here, the objective function (1) maximizes the net present value of the solution. Constraints (2) impose consistency between the $x$ and $y$ variables, ensuring that, in each period, all blocks within a cluster are extracted and processed in the same proportion. Constraints (3) impose that a cluster can be extracted at most once and (4) impose the precedence relationships. Constraints (5) represent general operational requirements, (6) bound the variables, and (7) impose integrality conditions on the $x$ variables. The exact nature of constraints (5) and (7) depends on specific considerations of the model, and we show some common choices below. Nevertheless, we should point out that, regardless of the application, constraints (5) typically include conditions of the form:

$$\sum_{b\in\mathcal{B}} \sum_{d\in\mathcal{D}} q_b y_{b,d,t} \le U_t \qquad \forall t \in \mathcal{T}, \tag{8}$$

$$\sum_{b\in\mathcal{B}} q_b y_{b,d,t} \le U_t^d \qquad \forall d \in \mathcal{D},\ t \in \mathcal{T}, \tag{9}$$

where (8) and (9) impose mining and destination capacity limits, respectively.

We denote by LP-PCPSP-C the linear programming (LP) relaxation of the PCPSP-C, that is, the relaxation obtained from (1)–(7) by removing constraints (7). Note that, due to the general constraints (5), the PCPSP-C, and even its LP relaxation, may be infeasible. However, if $G \ge 0$, then the null solution $(x, y) = (0, 0)$ is feasible.

In this work, we consider two different integrality conditions (7):

- The following conditions define *full integrality* constraints,

$$x_{c,t} \in \{0, 1\} \qquad \forall c \in \mathcal{C}, t \in \mathcal{T}. \tag{10}$$

These ensure that each block must be extracted entirely in a single time period.

- The following conditions define *partial integrality* constraints,

$$\sum_{t'=1}^{t} x_{c,t'} > 0 \text{ implies } \sum_{t'=1}^{t} x_{c',t'} = 1 \qquad \forall (c,c') \in \mathcal{A}, \forall t \in \mathcal{T}. \tag{11}$$

These constraints allow the duration of cluster extraction to span multiple time periods; however, they specify that a cluster must be entirely extracted before any of its successors can, in turn, be extracted.

Partial integrality conditions (11) are often imposed in commercial software packages, such as GEOVIA Whittle (Dassault Systèmes 2019), which, to our knowledge, were first identified in Gershon (1983). Note that, by using either (10) or (11), the PCPSP-C has a bounded feasible set. Thus, there exists an optimal solution whenever the set is nonempty.

From this point onward, we refer to PCPSP-C whenever a technique is independent of the exact nature of (7) (e.g., when the discussion concerns the LP relaxation); otherwise, we use PCPSP-F or PCPSP-P if the technique applies to the PCPSP-C under full integrality conditions (10) or partial integrality conditions (11), respectively. Similarly, the OPPDP and OPPSP can be formulated as special cases of the PCPSP-C.

### 2.2. The Phase Design Problem (OPPDP)

The objective of solving an open pit phase design problem (OPPDP) is to compute a sequence of nested pits, which a mining engineer can subsequently use to define phases. The OPPDP is the specific case of the PCPSP-F in which all the clusters are singletons (i.e., each cluster corresponds to a single block) and in which $G, g \geq 0$. Instances of the OPPDP with a single destination per block are sometimes known as C-PITs, as in Chicoisne et al. (2012), whereas those instances with multiple destinations per block are sometimes referred to as open pit mine production scheduling problems (OPMPSPs), as in Boland et al. (2009), or PCPSPs, as in Bienstock and Zuckerberg (2009). In general, instances of the OPPDP are often just referred to as *direct block scheduling* problems.

Observe that feasible solutions of the PCPSP-C always correspond to a sequence of nested pits, regardless of how the clusters are defined. In fact, for every $t \in \mathcal{T}$ and every feasible solution $x$ of the PCPSP-C, the set $\mathcal{B}(x;t) = \{b \in \mathcal{B} : \sum_{t'=1}^{t} x_{c(b),t'} > 0\}$ defines a pit. Furthermore, the sequence $\mathcal{B}(x;1), \ldots, \mathcal{B}(x;2), \ldots, \mathcal{B}(x;T)$ is nested. By defining clusters as singletons, it is possible to generate the best possible sequence of nested pits using the OPPDP in terms of net present value.

Traditionally, nested pits for phase design have heuristically been computed in the mining community using parameterized maximum closure algorithms. See Lerchs and Grossmann (1965), Hochbaum (2008), and Hustrulid and Kuchta (2006) for details.

Ideally, pits should be few in number, and the volumetric difference between consecutive pits should define wide contiguous areas, where one can easily accommodate ramps and where equipment (shovels, trucks) can work and move around easily; however, neither the pits generated by solving the OPPDP nor those obtained via parametric analysis necessarily meet these conditions. Moreover, it is not clear how one can modify these methodologies so that these conditions are met. In practice, this issue is addressed through the pits computed by the OPPDP or by parametric analysis as "guides" in a manual design process. This process begins with a sequence of pits $P_1, P_2, \ldots, P_n$, and by using specialized computer-aided design (CAD) software systems such as Deswik.CAD (2017) or Maptek Vulcan (2017), it constructs a new sequence $P'_1, P'_2, \ldots, P'_k$ such that the required operational conditions hold. These new pits, in turn, are used to compute a sequence of phases $\varphi_1, \varphi_2, \ldots, \varphi_n$, where $\varphi_1 = P'_1$ and $\varphi_j = P'_j \setminus P'_{j-1}$ for all $j > 1$.

Although ultimately both the OPPDP and parametric analysis are heuristic mechanisms by which to generate nested pits, the use of the OPPDP makes it easier to control the size and number of pits. See Meagher et al. (2014) for a discussion of the limitations of parametric analysis.

### 2.3. The Production Scheduling Problem (OPPSP)

The open pit production scheduling problem (OPPSP) is a specific case of the PCPSP-P in which all clusters correspond to bench-phases, defined as the intersection of a bench and a phase. As mentioned above, each bench corresponds to a horizontal level of the block model, and each phase is the output of the phase design process described in Section 2.2. The OPPSP is very closely related to the cutoff grade optimization problem, originally proposed by Lane (1964). See Bienstock and Zuckerberg (2009) for a discussion of how LP duality can be used to demonstrate the relationship between the OPPSP and cutoff grades.

As discussed in Section 2, given two clusters $c_1, c_2$, we have that $c_1 \prec c_2$ whenever there exist $b_1 \in c_1$ and $b_2 \in c_2$ such that $b_1 \prec b_2$. In practical applications, it is common to also use additional precedence relationships. For example, it is often required that clusters in the same bench be extracted in the order prescribed by the phases (e.g., Tolwinski 1998). This requirement is easy to model in the OPPSP, as it is simply a matter of adding arcs to set $\mathcal{A}$.

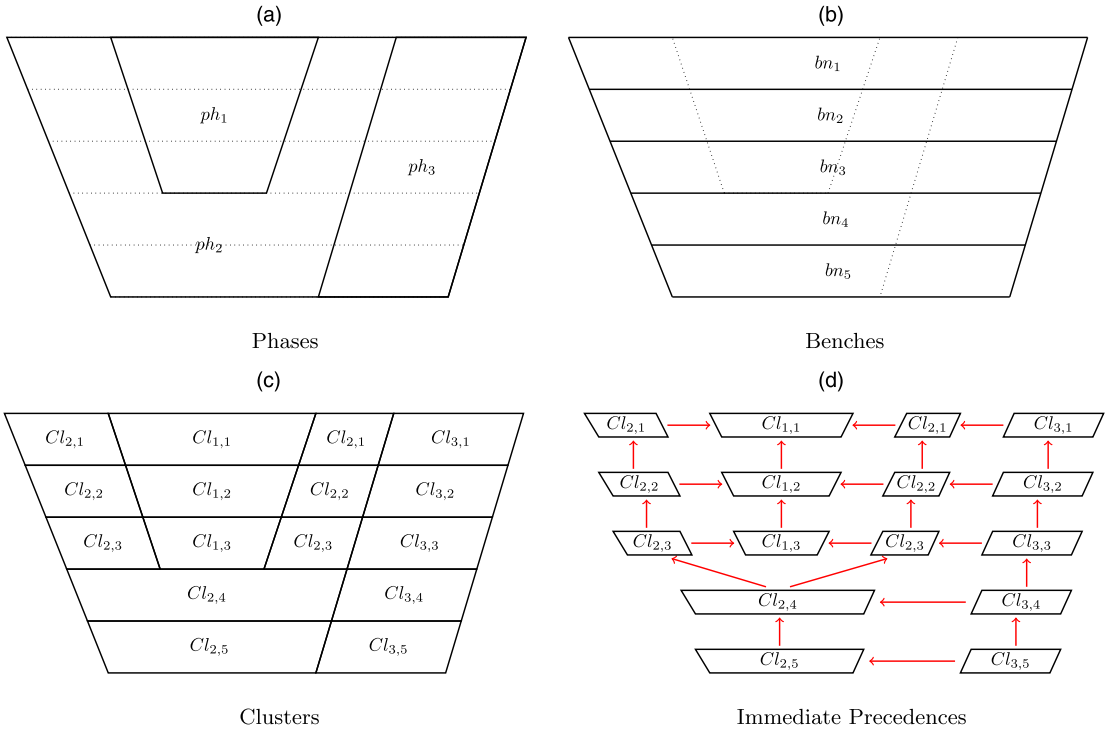
Figure 1 illustrates how a particular phase design could determine the definition of clusters used and the precedence relationships between them. In this figure, we let $ph_i$ be the $i$th phase and let $bn_i$ be the $i$th bench. For phase $ph_i$ and bench $bn_j$, let $Cl_{i,j}$ be the corresponding cluster obtained from the intersection of $ph_i$ and $bn_j$.

OPPSP formulations tend to be more detailed than OPPDP formulations. In addition to (9), inequalities (5) can include other mining constraints, such as those related to flow-balance, material blending, and stockpiles.

## 3. Previous Methodological Work and Contributions of This Article

A typical mine planning problem is composed of 1–5 million blocks, 20–50 time periods, and 2–3 destinations, which results in a huge number of decisions, leading to models that are impossible to solve with commercial integer programming solvers; even solving the LP-PCPSP-C has proved elusive. To date,

**Figure 1.** (Color online) Example Illustrating the Definition of Phases, Benches, Clusters, and the Precedences Between Clusters




*Notes.* Each arrow represents an immediate precedence relationship. The head of each arrow corresponds to the predecessor in the corresponding precedence relationship.

optimization approaches are varied and, to a large extent, work by exploiting problem-specific characteristics, such as the type of problem being solved (C-PIT, PCPSP, or PCPSP-C) and/or the characteristics of the matrix $G$, for example, $G \geq 0$, or $G$ general.

The earliest work attempting to solve PCPSP-C dates back to Johnson (1968), who proposed using the decomposition of Dantzig and Wolfe (1960) for the LP relaxation of the PCPSP but without any methods for computing integer-feasible solutions; he only managed to solve instances with up to 36 blocks. After Johnson's work, most optimization research regarding the PCPSP-C has focused on studying C-PIT, that is, the variant of the PCPSP-C in which each cluster is a single block with a single destination per block, full integrality conditions are assumed, and there are only positive (upper-bound) capacity constraints. Important milestones in this direction include the following:

- Dagdelen (1985) and Dagdelen and Johnson (1986) study problems with up to 5,400 blocks by solving the LP relaxation with Lagrangian relaxation. The authors argue that, in practical mining problems, primal feasibility is not so important, and they propose using the solution of the Lagrangian pricing problem as the primal output. Even though this solution is integer-feasible, it can violate capacity constraints.
- Caccetta and Hill (2003) prove that a common preprocessing technique based on computing ultimate pit limits is correct for C-PIT. They also claim to solve problems with up to 209,664 blocks to optimality with a customized branch-and-bound approach. Their methodology is not described, citing commercial interests.
- Gaupp (2008) addresses problems with up to 25,620 blocks by combining Lagrangian relaxation and customized rounding heuristics and by introducing the notion of "early starts" and "late starts" for preprocessing and LP bound strengthening.
- Cullenbine et al. (2011) and Lambert and Newman (2014) handle problems with up to 25,620 blocks by combining tailored Lagrangian relaxation and a sliding time window heuristic. Their results are the first to consider instances with lower-bound capacity constraints. Solutions within 4% of optimality are obtained.
- Espinoza et al. (2012) make a collection of OPPDP instances available by means of the MineLib website. These instances range in size from 1,060 to 2,140,342 blocks.
- Chicoisne et al. (2012) develop a customized decomposition algorithm for solving the LP relaxation of instances with only a single capacity constraint per period. To obtain integer-feasible solutions, they

propose a heuristic called *TopoSort* and a local search heuristic for further refinement. Solutions within 2% of optimality are reported for instances with up to 4 million blocks. By using the algorithm for variants of the instances with two capacity constraints, solutions with a provable gap of 4% are computed.

- Vossen et al. (2016) develop a customized branch-and-bound method using Benders' decomposition and customized heuristics. The approach considers lower-bound constraints and a novel type of integrality. Problems with up to 25,620 blocks are solved to 1% of optimality.
- To date, the best-known solutions for the C-PIT are those produced by three recent heuristics. Jelvez et al. (2016) propose a heuristic combining aggregation and disaggregation, Liu and Kozan (2016) propose a topological sorting heuristic which does not take as input an LP relaxation solution, and Samavati et al. (2018) extend TopoSort by introducing a diving component guided by genetic algorithms. All of these papers improve the best gaps reported for MineLib instances.

Complementary works related to optimization methodologies for C-PIT include the following:

- Fricke (2006), Bley et al. (2010), and Espinoza et al. (2015) study clique, cover, and other inequalities on precedence-constrained knapsack problems and derive cuts and separation algorithms.
- A number of articles have proposed aggregating variables to obtain smaller problems which are directly solvable with commercial integer programming solvers. These methods vary depending on how clusters are defined. Important examples encompass the *fundamental trees* of Ramazan et al. (2005), the *mining blocks* of Smith and Wicks (2013), the *mining cuts* of Askari-Nasab et al. (2010, 2011), and the *clumps* of Froyland and Menabde (2011).
- A number of articles have studied stochastic variants of C-PIT. For reference, see the work of Boland et al. (2008), Ramazan and Dimitrakopoulos (2013), and Rimélé (2016).

The work of Boland et al. (2009) represents a significant departure from earlier research, as they abandon the study of C-PIT and instead tackle the more general PCPSP (i.e., with multiple destinations and general classes of constraints). In their work, they develop a customized decomposition method that works by aggregating and disaggregating blocks. In computational experiments, they solve instances with up to 96,000 blocks to 1% of optimality.

- Bienstock and Zuckerberg (2009, 2010) generalize the formulation and decomposition method of Boland et al. (2009) to the more general PCPSP-C (which they call the PAP). Though their formulation generalizes both the OPPDP and OPPSP, their computational experiments focus on instances of the former, discussing neither model in detail and studying only the LP relaxation of the problem.
- Muñoz et al. (2018) build on the work of Bienstock and Zuckerberg (2009), extending its applicability to broader classes of scheduling problems, comparing the extension with other decomposition approaches, and introducing a number of computational enhancements. Similar to Bienstock and Zuckerberg (2009), this article only focuses on LP relaxations.
- Goycoolea et al. (2013, 2015) discuss how solutions generated by the PCPSP-C formulation compare with those generated by traditional commercial software, from both modeling and computational viewpoints.
- King et al. (2017) extend the PCPSP-C to model open-pit-to-underground transition problems, and Moreno et al. (2017) extend the PCPSP-C to model stockpiles.

Also, a different approach based on network flows and aimed at a simultaneous, aggregated analysis of multiple mine sites is that of Epstein et al. (2012).

In this article, we extend many of the aforementioned methodologies, originally developed in the context of the C-PIT, to the context of the PCPSP-C. We emphasize how these methodologies can be used to solve both the OPPDP and OPPSP, each of which requires some problem-specific techniques. Overall, our extensions mostly focus on the decisions which affect clusters of blocks rather than individual blocks, the multiplicity of destinations which a block can have, and the possibility of admitting partial integrality. In addition, we propose a number of new methodologies. We are especially concerned with techniques that are useful for the OPPSP, which has received much less attention in academic literature. Our work introduces new classes of cutting planes and heuristics designed to reduce the large integrality gaps observed in OPPSP instances. We also implement a specialized branch-and-bound algorithm.

Finally, we show that all of these methodologies work well when integrated. Most, if not all, academic papers have studied these techniques individually, rather than in concert. This integration is key to showing, for the first time, that it is possible to solve to near optimality instances of both the OPPDP and OPPSP on real mine planning data sets, such as those handled by commercial mine planning software systems.

## 4. Preprocessing

As discussed above, PCPSP-C can be very large in terms of the number of variables and constraints. In this section, we present techniques for transforming this problem into a mathematically equivalent one which possesses fewer variables. In other words, we reduce the dimension of the problem without affecting the optimal objective function value.

### 4.1. Ultimate Pit Limit Preprocessing

In this subsection, we describe a preprocessing scheme that can be used to eliminate variables from an instance of PCPSP-C by identifying clusters that provably do not appear in an optimal solution of the problem. The scheme is a generalization of one proposed by Caccetta and Hill (2003) in the context of C-PIT, which we extend to general instances of PCPSP-C, in which $G, g \geq 0$.That is, the generalization works for both partial and full integrality and also for instances with clusters which are not necessarily singleton blocks.

In general, given an optimization problem defined over a region $S \subseteq \mathbb{R}^n$, we say that an optimal solution $x \in S$ is *minimal* if every other optimal solution $x' \in S$ is such that $x' \leq x$ implies $x' = x$.

For each $c \in \mathscr{C}$, define $\bar{p}_c = \sum_{b \in c} p_{b,*}$, where $p_{b,*} = \max_{d \in \mathscr{D}} p_{b,d}$ (as defined in Section 2). We define the ultimate pit limit problem (U-PIT) of a PCPSP-C instance (see Formulation (1)–(7)) as

$$\max \sum_{c \in \mathscr{C}} \bar{p}_c x_c \tag{12}$$

$$\text{s.t. } x_c \leq x_{c'} \qquad \forall (c, c') \in \mathscr{A}, \tag{13}$$

$$x_c \in [0,1] \qquad \forall c \in \mathscr{C}. \tag{14}$$

Note that, because this problem does not consider limited resources, there is no need to define variables over multiple time periods.

U-PIT is a maximum closure problem. As such, (a) its Formulation (12)–(14) is totally unimodular, and hence, its optimal basic solutions are binary; (b) it admits a unique minimal optimal solution; and (c) it can be solved efficiently. For more information about the maximum closure problem, see Hochbaum (2008).

Given a vector $x \in [0,1]^{\mathscr{C}}$ that is feasible for U-PIT, we say that $P = \{c \in \mathscr{C} : x_c > 0\}$ is the *pit limit* associated with $x$. Let $(x^*, y^*)$ represent a minimal optimal solution to PCPSP-C. For each $c \in \mathscr{C}$, let $x^1_c = \sum_{t \in \mathscr{T}} x^*_{c,t}$. Since $x^*$ satisfies constraints (4), it follows that $x^1$ is feasible for U-PIT. Let us define $P^{OPT}$ as the pit limit associated with $x^1$. Slightly abusing notation, let $P^{OPT}$ be the pit limit associated with the minimal solution $(x^*, y^*)$.

The following theorem generalizes a result of Caccetta and Hill (2003). The proof is presented in the e-companion of this article.

**Theorem 1.** *Consider an instance of PCPSP-C, as described in Formulation (1)–(7), such that $G \geq 0$ and $g \geq 0$. Let $P^{OPT}$ and $P^{\text{U-PIT}}$ be the pit limits of the minimal optimal solutions to PCPSP-C and U-PIT, respectively. Then, $P^{OPT} \subseteq P^{\text{U-PIT}}$.*

Theorem 1 allows us to preprocess as follows when $G \geq 0$ and $g \geq 0$. For each $c \in \mathscr{C}$, compute $\bar{p}_c$, and solve U-PIT, as described in (12)–(14). We can eliminate all variables associated with clusters $c \notin P^{\text{U-PIT}}$ from the problem, as we know that there exists an optimal solution to PCPSP-C which does not use them. Note that the proof of Theorem 1 holds for both integrality conditions (10) and (11). Because of this, $P^{\text{U-PIT}}$ is typically referred to as the *ultimate pit limit* of PCPSP-C.

### 4.2. Dominated Triplet Elimination Preprocessing

In nearly all strategic mine planning problems, most blocks are typically "waste" blocks, that is, blocks whose grade is so low that the profit of selling the recovered metal does not cover the processing costs. In most, though not all, problems, the option of sending such blocks to the processor can be removed from the model (by eliminating a destination of these blocks), thus making the problem smaller. Below we present a new, yet simple, preprocessing algorithm that can be used to detect and eliminate destinations which are never used in an optimal solution for cases such as these.

For each triplet $(b, d, t) \in \mathscr{B} \times \mathscr{D} \times \mathscr{T}$, let $G_{b,d,t}$ be the column of matrix $G$ corresponding to variable $y_{b,d,t}$. We say that triplet $(b, d_1, t)$ dominates triplet $(b, d_2, t)$ if $G_{b,d_1,t} \leq G_{b,d_2,t}$ (componentwise) and $p_{b,d_1,t} \geq p_{b,d_2,t}$. Intuitively, $(b, d_1, t)$ dominates $(b, d_2, t)$ if the proportion of block $b$ sent to destination $d_1$ in time $t$ consumes the same or fewer resources and provides at least the same value as sending $b$ to $d_2$ in $t$. If this is the case, then there is no need to actually define variable $y_{b,d_2,t}$, as we may assume that there exists an optimal solution such that $y_{b,d_2,t} = 0$.

### 4.3. Aggregation Preprocessing

In many instances of PCPSP-C, blocks in close proximity to each other are identical. In such cases, it is possible to group these blocks to reduce the number of variables. We describe this new preprocessing idea below.

Consider an instance of PCPSP-C and two blocks $b_1$, $b_2$ within the same cluster such that, for some $\lambda \geq 0$, the following holds:

$$p_{b_1,d} = \lambda p_{b_2,d} \qquad \forall d \in \mathscr{D}, \tag{15}$$

$$G_{b_1,d,t} = \lambda G_{b_2,d,t} \qquad \forall d \in \mathscr{D},\ t \in \mathscr{T}. \tag{16}$$

In such a case, we can aggregate the two blocks into one block $\bar{b}$ with $G_{\bar{b},d,t} = G_{b_1,d,t} + G_{b_2,d,t}$ and $p_{\bar{b},d,t} = p_{b_1,d,t} + p_{b_2,d,t}$. The validity of this preprocessing comes from the fact that, given any feasible solution $(x^*, y^*)$ to the PCPSP-C satisfying (15) and (16), there exists another solution $(x, y)$ to the new aggregated problem with the same objective value as $(x^*, y^*)$. We can take $y_{\bar{b},d,t} = \frac{\lambda}{1+\lambda} y^*_{b_1,d,t} + \frac{1}{1+\lambda} y^*_{b_2,d,t}$ for each $d \in \mathscr{D}, t \in \mathscr{T}$, take $y_{b,d,t} = y^*_{b,d,t}$ for each $b \in \mathscr{B} \setminus \{b_1, b_2\}, d \in \mathscr{D}, t \in \mathscr{T}$, and take $x = x^*$. We can observe that $p_{\bar{b},d,t} y_{\bar{b},d} = p_{b_1,d,t} y_{b_1,d} + p_{b_2,d,t} y_{b_2,d}$ and $G_{\bar{b},d,t} y_{\bar{b},d,t} = G_{b_1,d,t} y_{b_1,d,t} + G_{b_2,d,t} y_{b_2,d,t}$ for each $d \in \mathscr{D}, t \in \mathscr{T}$.

Note that this scheme is not affected by precedence relationships given that only blocks within the same cluster are aggregated and precedences in PCPSP-C are only defined between clusters.

## 5. Cutting Planes

Bienstock and Zuckerberg (2010) claim that Formulation (1)–(7) of PCPSP-C empirically presents low integrality gaps. In Section 8, computational experiments show that this is largely true for OPPDP. However, we do not observe the same behavior in OPPSP. To obtain low integrality gaps, we require problem-specific cutting planes. In this section, we first extend clique and early start inequalities from the context of C-PIT to PCPCP-C. The first family of cuts that we obtain ignores the destination of each block, and we refer to them as *extraction cuts*. We also derive two new classes of cuts which exploit the problem structure resulting from block destination selectivity. We call these *production cuts*.

In what follows, we use a new set of variables describing the accumulated value of $x$ variables:

$$w_{c,t} = \sum_{t'=1}^{t} x_{c,t'} \qquad \forall d \in \mathcal{D}, t \in \mathcal{T}. \tag{17}$$

The value of $w_{c,t}$ in a feasible solution is the fraction of cluster $c$ extracted in time period $t$ or earlier. Observe that the precedence constraints (4) and mining capacity constraints (8) on the $x$ variables imply that

$$w_{c,t} \le w_{c,t+1}, \qquad \forall c \in \mathcal{C},\ \forall t \in 1,\dots,T-1, \tag{18}$$

$$w_{c,t} \le w_{c',t}, \qquad \forall (c,c') \in \mathcal{A},\ t \in \mathcal{T}, \tag{19}$$

$$\sum_{c\in\mathcal{C}} q_c w_{c,t} \le \sum_{t'=1}^{t} U_{t'}, \qquad \forall t \in \mathcal{T}, \tag{20}$$

where $q_c = \sum_{b\in c} q_b$ is the total weight of the cluster. The full integrality conditions (10) and partial integrality conditions (11) translate, respectively, as follows to the $w$ variables:

$$w_{c,t} \in \{0,1\}, \qquad \forall c \in \mathcal{C},\ \forall t \in \mathcal{T}, \tag{21}$$

$$w_{c,t} > 0 \text{ implies } w_{c',t} = 1, \quad \forall (c,c') \in \mathcal{A},\ \forall t \in \mathcal{T}. \tag{22}$$

All cuts are presented using the $w_{c,t}$ variables for simplicity; nonetheless, it is evident that they can be expressed using only variables $x_{c,t}$. Thus, slightly abusing notation, we discuss these *valid inequalities* for PCPSP-C using variables $w_{c,t}$.

For the reader's sake, let us recall relevant definitions and assumptions. We assume $\mathcal{C}$ to be such that $\prec$ defines a partial order, and therefore, $\mathcal{A}$ defines a DAG. For each cluster $c \in \mathcal{C}$, we define the *closure* of $c$ as $cl(c) = \{c\} \cup \{c' \in \mathcal{C} : c' \prec c\}$ and the *reverse closure* of $c$ as $rcl(c) = \{c\} \cup \{c' \in \mathcal{C} : c \prec c'\}$. For a block $b \in \mathcal{B}$, we denote by $c(b)$ the unique cluster containing it. Additionally, for a set of clusters $C \subseteq \mathcal{C}$, we denote by $b(C) := \{b \in \mathcal{B} : b \in c, c \in C\}$ the set of blocks on $C$. Given $S \subseteq \mathcal{C}$ and a vector $q \in \mathbb{R}^{\mathcal{B}}$, we define $q(S) = \sum_{c\in S} q_c = \sum_{c\in S}\sum_{b\in c} q_b$.

### 5.1. Extraction Cuts

We next describe a class of cuts which exploits extraction capacity limits (8). Since (18) and (19) are precedence constraints and (20) is a knapsack constraint (i.e., nonnegative coefficients and a positive right-hand side), (18)–(20) is a precedence-constrained knapsack relaxation of PCPSP-C.

For a discussion of cuts derived from the precedence-constrained knapsack problem, in addition to separation techniques, see Bley et al. (2010) and Espinoza et al. (2015). The precedence-constrained knapsack cuts that we analyze differ from those treated traditionally in academic literature in two aspects. First, the time indexing of the variables gives these relaxations a nested structure that can be exploited. Second, we not only consider the full integrality constraint (21), but also the partial integrality constraint (22).

We now move on to describing three families of extraction cuts: early start cuts, diamond cuts, and clique and lifted clique cuts.

#### 5.1.1. Early Start Cuts.

As noted by Gaupp (2008), Lambert et al. (2014), and others, one can use early start cuts to eliminate a number of variables from PCPSP. We extended this preprocessing technique to the more general PCPSP-C, including the case of partial integrality condition (11). This can reduce the size of the problem, possibly speeding up computations, and it can also improve the LP relaxation bound of the problem.

**Theorem 2.** *Let $t \in \mathcal{T}$ and $Q_t = \sum_{t'=1}^{t} U_{t'}$, and consider a cluster $c \in \mathcal{C}$ such that $q(cl(c)) > Q_t$. Then,*

- *the equality $w_{c,t} = 0$ is valid for the PCPSP-F; and*
- *whenever $q_c > 0$, the inequality $w_{c,t} \le \frac{(Q_t - q(cl(c)\setminus\{c\}))^+}{q_c}$ is valid for PCPSP-P.*

**Proof.** The result for the PCPSP-F is known; thus, we only prove this result for the PCPSP-P. We use integrality condition (22) obtained from (11). If $w_{c,t} > 0$, then $w_{c',t} = 1$ for all $c' \in cl(c) \setminus \{c\}$. From this and inequality (20), it follows that $q_c w_{c,t} \le Q_t - q(cl(c) \setminus \{c\})$. If $Q_t \le q(cl(c) \setminus \{c\})$, we obtain a contradiction with $w_{c,t} > 0$. Hence, $Q_t \le q(cl(c) \setminus \{c\})$ implies $w_{c,t} = 0$. From these conditions, we conclude that $w_{c,t} \le \frac{(Q_t - q(cl(c)\setminus\{c\}))^+}{q_c}$ is valid for PCPSP-P, where $(Q_t - q(cl(c) \setminus \{c\}))^+ = \max\{0, (Q_t - q(cl(c) \setminus \{c\}))\}$. □

Note that early start cuts can also be used as a preprocessing technique, eliminating variables in PCPSP-F or PCPSP-P when the cut sets them to 0.

**5.1.2. Diamond Cuts.** The following inequalities are similar to those used by Zhu et al. (2006) in the context of multimode resource-constrained project problems. These cuts use the set $cl(c_2) \cap rcl(c_1)$ for $c_1, c_2 \in \mathcal{C}$ such that $c_1 \prec c_2$, which intuitively can be pictured as the intersection of two cones facing each other.

**Theorem 3.** *Let $t_1, t_2 \in \mathcal{T}$, $t_1 \le t_2$, and $Q_{t_1,t_2} = \sum_{t'=t_1}^{t_2} U_{t'}$, and consider clusters $c_1, c_2 \in \mathcal{C}$ such that $c_1 \prec c_2$. Then,*

- *whenever $q(cl(c_2) \cap rcl(c_1)) > Q_{t_1,t_2}$, the inequality $w_{c_2,t_2} \le w_{c_1,t_1}$ is valid for PCPSP-F; and*
- *whenever $q(cl(c_2) \cap rcl(c_1) \setminus \{c_1, c_2\}) > Q_{t_1,t_2}$, the inequality $w_{c_2,t_2} \le w_{c_1,t_1}$ is valid for the PCPSP-P.*

**Proof.** First, consider full integrality condition (21) obtained from (10), and as a contradiction, suppose that $w_{c_2,t_2} > w_{c_1,t_1}$. This implies that $w_{c_2,t_2} = 1$ and $w_{c_1,t_1} = 0$. Consequently, all clusters in $cl(c_2) \cap rcl(c_1)$ are extracted in time periods $t_1$ through $t_2$. Nevertheless, this is impossible, as $q(cl(c_2) \cap rcl(c_1)) > Q_{t_1,t_2}$.

Now, assume partial integrality condition (22) obtained from (11) and that $w_{c_2,t_2} > w_{c_1,t_1}$. Consequently, $w_{c_1,t_1} < 1$ and $w_{c_2,t_2} > 0$. These conditions imply that every cluster in $cl(c_2) \cap rcl(c_1) \setminus \{c_1, c_2\}$ is fully extracted in time periods $t_1$ through $t_2$. Nevertheless, this is impossible, as $q(cl(c_2) \cap rcl(c_1) \setminus \{c_1, c_2\}) > Q_{t_1,t_2}$. □

**5.1.3. Clique and Lifted Clique Cuts.** Clique and lifted clique inequalities have been studied for precedence-constrained knapsack problems since Boyd (1993). We now introduce a natural generalization for the context of the PCPSP-C which can be used with partial integrality.

Consider two clusters $c_1, c_2 \in \mathcal{C}$ such that neither $c_1 \prec c_2$ nor $c_2 \prec c_1$. We define the *incompatibility* of $c_1$ and $c_2$ as follows:

- We say that $c_1$ and $c_2$ are *f-incompatible* in time $t$ if $q(cl(\{c_1, c_2\})) > Q_t$.
- We say that $c_1$ and $c_2$ are *p-incompatible* in time $t$ if $q(cl(\{c_1, c_2\}) \setminus \{c_1, c_2\}) > Q_t$.

Intuitively, incompatible clusters are such that neither can be extracted before a certain time period. Using this information, we obtain the following result.

**Theorem 4.** *Let $\{c_1, c_2, \ldots, c_k\}$ be a set of clusters, and consider the following inequality:*

$$\sum_{i=1}^{k} w_{c_i,t} \le 1. \tag{23}$$

*Then,*

- *if $\{c_1, c_2, \ldots, c_k\}$ are pairwise f-incompatible, then inequality (23) is valid for PCPSP-F; and*
- *if $\{c_1, c_2, \ldots, c_k\}$ are pairwise p-incompatible, then inequality (23) is valid for PCPSP-P.*

*In addition, if a cluster $c \in \mathcal{C}$ is such that $c \prec c_i$ for $i = 1, \ldots, k$, then constraint $\sum_{i=1}^{k} w_{c_i,t} \le w_{c,t}$ is also valid for PCPSP-F (PCPSP-P) if $\{c_1, c_2, \ldots, c_k\}$ are pairwise f-incompatible (p-incompatible).*

**Proof.** To prove that (23) is valid, suppose $\sum_{i=1}^{k} w_{c_i,t} > 1$. This assumption implies that there exist $i_1, i_2 \in 1, \ldots, k$ such that $w_{i_1,t} > 0$ and $w_{i_2,t} > 0$.

Assume that $\{c_1, c_2, \ldots, c_k\}$ are pairwise *f*-incompatible. Using integrality condition (21), we would have that $w_{i_1,t} = w_{i_2,t} = 1$. This would imply that all clusters in $cl(\{c_{i_1}, c_{i_2}\})$ were extracted at time $t$ or earlier, which would, in turn, contradict $q(cl(\{c_{i_1}, c_{i_2}\})) > Q_t$. This proves the result for the PCPSP-F.

Now, assume that $\{c_1, c_2, \ldots, c_k\}$ are pairwise p-incompatible. Using integrality condition (22), $w_{i_1,t} > 0$ and $w_{i_2,t} > 0$ would imply that all clusters in $cl(\{c_{i_1}, c_{i_2}\}) \setminus \{c_{i_1}, c_{i_2}\}$ were extracted at time $t$ or earlier, which would, in turn, contradict $q(cl(\{c_{i_1}, c_{i_2}\}) \setminus \{c_{i_1}, c_{i_2}\}) > Q_t$. From here, we conclude the result for PCPSP-P. □

## 5.2. Production Cuts

In this subsection, we describe new classes of cuts for the PCPSP-C, which we term *production cuts*, because they exploit the capacity limits of each destination (9) that are typically used to impose limits on production.

**5.2.1. Production Cuts in a Simplified Case.** For the reader's sake, in this subsection, we present the *production cuts* in a simplified version of the problem, which considers two destinations: blocks are either processed ($P$) or wasted ($W$). We assume no clusters (or, equivalently, single-block clusters) and no time index (i.e., $T = 1$). Consider an DAG $G = (\mathcal{B}, \mathcal{A})$, a vector of positive coefficients $q \in \mathbb{R}_+^{\mathcal{B}}$, and a capacity $U > 0$. We define the *mode-knapsack mixed-integer set* ($MK$) as follows. A vector $(x, y^P, y^W) \in \mathbb{R}^{\mathcal{B}} \times \mathbb{R}^{\mathcal{B}} \times \mathbb{R}^{\mathcal{B}}$ belongs to the set $MK$ if and only if the following conditions are met:

$$\sum_{b \in \mathcal{B}} q_b y_b^P \le U, \tag{24}$$

$$y_b^P + y_b^W = x_b \qquad \forall b \in \mathcal{B}, \tag{25}$$

$$x_b \le 1 \qquad \forall b \in \mathcal{B}, \tag{26}$$

$$x_b > 0 \Longrightarrow x_{b'} = 1 \qquad \forall (b, b') \in \mathcal{A}, \tag{27}$$

$$y_b^P, y_b^W \ge 0 \qquad \forall b \in \mathcal{B}. \tag{28}$$

We can derive the following cuts for $MK$.

***Simplified Variable-Right-Hand Side (VRHS) Cuts.*** This family of cuts combines the precedence relationships in the extraction of blocks and the production

limit (24). In other words, they ensure that if a solution sends a large number of blocks to be processed, then all predecessors of those blocks must be fully extracted as well.

**Theorem 5.** *Let $b \in \mathcal{B}$ be such that $q_b \le U$. Then, the following inequality is valid for MK:*

$$\sum_{b' \in rcl(b)} q_{b'} y^P_{b'} \le U x_b.$$

**Proof.** If $x_b = 1$, then the inequality holds because of (24). If $x_b < 1$, then $x_{b'} = 0$ for each $b' \in rcl(b) \setminus \{b\}$, and then, the inequality holds because $y^P_b \le x_b$ and $q_b \le U$. □

Note that although we assume that $q_b \le U$, we can easily relax this assumption by changing the coefficient of the variable on the right-hand side.

***Simplified Hourglass Cuts.*** Contrary to the VRHS cuts, the *hourglass* cuts derive valid inequalities involving $W$, the *waste* destination in our interpretation. In other words, we exploit the fact that, for a block to be extracted, it is necessary to send blocks comprising an aggregate minimum weight to destination $W$.

**Theorem 6.** *Let $b \in \mathcal{B}$, and let $S \subseteq cl(b) \setminus \{b\}$ be such that $q(S) \ge U$. Then, the following inequality is valid for MK:*

$$\big(q(S) + q_b - U\big)x_b \le \sum_{b' \in S \cup \{b\}} q_{b'} y^W_{b'}.$$

**Proof.** The inequality is clearly valid when $x_b = 0$. Suppose that $x_b > 0$; then, for each $b' \in S$, we have $x_{b'} = y^P_{b'} + y^W_{b'} = 1$. Hence,

$$\begin{aligned}
\sum_{b' \in S \cup \{b\}} q_{b'} y^P_{b'} \le U &\Longrightarrow U - \sum_{b' \in S} q_{b'} y^P_{b'} \ge q_b y^P_b \\
&\Longrightarrow U - \sum_{b' \in S} q_{b'} \big(1 - y^W_{b'}\big) \ge q_b y^P_b \\
&\Longrightarrow \sum_{b' \in S} q_{b'} y^W_{b'} \ge q_b y^P_b + q(S) - U \\
&\Longrightarrow \sum_{b' \in S \cup \{b\}} q_{b'} y^W_{b'} \ge q_b \big(y^P_b + y^W_b\big) \\
&\qquad + q(S) - U \\
&\Longrightarrow \sum_{b' \in S \cup \{b\}} q_{b'} y^W_{b'} \ge q_b x_b + q(S) - U \\
&\qquad \ge \big(q_b + q(S) - U\big)x_b.
\end{aligned}$$

In the above, the last inequality holds because $q(S) \ge U$ and $x_b \le 1$. Therefore, whenever $x_b > 0$, we have

$$\big(q(S) + q_b - U\big)x_b \le \sum_{b' \in S \cup \{b\}} q_{b'} y^W_{b'}.$$

These cuts can be lifted using the extraction variables for the nodes in $rcl(b)$ to obtain the following valid inequality for $MK$:

$$\big(q(S) + q_b - U\big)x_b + \sum_{b' \in rcl(b) \setminus \{b\}} q_{b'} y^P_{b'} \le \sum_{b' \in S \cup \{b\}} q_{b'} y^W_{b'}.$$

The proof of the validity of this inequality for $MK$ is identical to the previous one. Because this last expression uses variables for blocks in $cl(b)$ and in $rcl(b)$, we call this family *hourglass cuts*. □

Now, we introduce both families in the general PCPSP-C context. The intuition is the same; however, additional analysis is necessary to include clusters, arbitrary numbers of destinations, and multiple time periods.

**5.2.2. VRHS Cuts.** First, we observe that the following conditions are valid for PCPSP-C for both full and partial integrality conditions:

$$\begin{aligned}
&w_{c,t} < 1 \Rightarrow y_{b,d,t} = 0, \\
&\qquad \forall c \in \mathcal{C}, \forall b \in \{b \in \mathcal{B} : c(b) \in rcl(c) \setminus \{c\}\}, \\
&\qquad\qquad \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}, \qquad (29)
\end{aligned}$$

$$\begin{aligned}
&y_{b,d,t} \le w_{c,t}, \qquad \forall c \in \mathcal{C},\ \forall b \in c, \\
&\qquad\qquad \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}, \qquad (30)
\end{aligned}$$

$$\begin{aligned}
&\sum_{b \in \mathcal{B}} q_b y_{b,d,t} \le U^d_t, \qquad \forall d \in \mathcal{D},\ \forall t \in \mathcal{T}, \qquad (31)
\end{aligned}$$

with $w_{c,t}$ defined in (17). These can be used to derive a new family of valid inequalities, generalizing the cuts discussed above.

**Theorem 7.** *Consider a destination $d \in \mathcal{D}$, a time period $t \in \mathcal{T}$, and a family of clusters $c_1, c_2, \ldots, c_n \in \mathcal{C}$ such that the following conditions hold:*

1. $c_1 \prec c_2 \prec \ldots \prec c_n$,
2. $q(rcl(c_1) \setminus rcl(c_n)) < U^d_t$,
3. $q(rcl(c_1)) > U^d_t$.

*Additionally, define $\Delta_k = rcl(c_k) \setminus rcl(c_{k+1})$ and $\delta_k = q(\Delta_k)$ for $k = 1, \ldots, n-1$, and $\Delta_n = rcl(c_n)$ and $\delta_n = U^d_t - q(rcl(c_1) \setminus rcl(c_n))$. Consider the inequality*

$$\begin{aligned}
&\sum_{k=1}^{n-1} \left( \sum_{c \in \Delta_k} \sum_{b \in c} q_b y_{b,d,t} \right) + \sum_{b \in c_n} \alpha_b q_b y_{b,d,t} \\
&\qquad + \sum_{c \in rcl(c_n) \setminus \{c_n\}} \sum_{b \in c} q_b y_{b,d,t} \le \sum_{k=1}^{n} \delta_k w_{c_k,t} \qquad (32)
\end{aligned}$$

*for some constants $0 \le \alpha_b \le 1$, $b \in c_n$. Then, (32) is valid for PCPSP-F. Additionally, if constants $\alpha_b$ satisfy*

$$\sum_{b \in c_n} \alpha_b q_b \le U^d_t - q(rcl(c_1) \setminus rcl(c_n)), \qquad (33)$$

*then (32) is valid for PCPSP-P.*

**Proof.** The proof is divided in two cases: (i) the case in which $\forall k \in \{1,\dots,n\}\ w_{c_k,t} = 1$, and (ii) the case in which $\exists k \in \{1,\dots,n\}\ w_{c_k,t} < 1$.

Case 1: $w_{c_k,t} = 1$ for $k = 1,\dots,n$. Note that $\{\Delta_1,\dots,\Delta_{n-1}\}$ is a partition of $rcl(c_1) \setminus rcl(c_n)$. Thus, $\sum_{k=1}^{n-1} \delta_k = q(rcl(c_1) \setminus rcl(c_n))$, which implies $\sum_{k=1}^{n} \delta_k = U_t^d$. Then

$$\begin{aligned}
&\sum_{k=1}^{n-1}\left(\sum_{c\in\Delta_k}\sum_{b\in c} q_b y_{b,d,t}\right) + \sum_{b\in c_n} \alpha_b q_b y_{b,d,t} \\
&\quad + \sum_{c\in rcl(c_n)\setminus\{c_n\}}\ \sum_{b\in c} q_b y_{b,d,t} \\
&\le \sum_{k=1}^{n-1}\left(\sum_{c\in\Delta_k}\sum_{b\in c} q_b y_{b,d,t}\right) + \sum_{b\in rcl(c_n)} q_b y_{b,d,t} \\
&\le \sum_{b\in\mathcal{B}} q_b y_{b,d,t} \le U_t^d = \sum_{k=1}^{n} \delta_k = \sum_{k=1}^{n} \delta_k w_{c_k,t},
\end{aligned}$$

where the first inequality comes from the fact that $\alpha_b \le 1$, and the second inequality ensues, as each block is being counted at most once on the left-hand side. Thus, in this case, (32) is valid.

Case 2: Suppose now that $\exists k$ such that $w_{c_k,t} < 1$, and let $k_o = \min\{k \in 1,\dots,n : w_{c_k,t} < 1\}$. Observe that $w_{c_k,t} = y_{b,d,t} = 0$ for $k > k_o$ and $b \in b(\Delta_k)$. Additionally, since this case implies $w_{c_n,t} < 1$, for each $c \in rcl(c_n) \setminus \{c_n\}$, we have $w_{c,t} = 0$. This makes inequality (32) equivalent to

$$\sum_{k=1}^{n-1}\left(\sum_{c\in\Delta_k}\sum_{b\in c} q_b y_{b,d,t}\right) + \sum_{b\in c_n} \alpha_b q_b y_{b,d,t} \le \sum_{k=1}^{n} \delta_k w_{c_k,t}. \tag{34}$$

We now analyze two subcases regarding $k_o$.

- If $k_o < n$, since $c_n \in \Delta_n$ and $rcl(c_n) \setminus \{c_n\} \subseteq \Delta_n$, the second term on the left-hand side of inequality (34) is 0. Moreover, this inequality simplifies to

$$\sum_{k=1}^{k_o}\left(\sum_{c\in\Delta_k}\sum_{b\in c} q_b y_{b,d,t}\right) \le \sum_{k=1}^{k_o} \delta_k w_{c_k,t}.$$

This inequality is valid since

$$\begin{aligned}
\sum_{c\in\Delta_k}\sum_{b\in c} q_b y_{b,d,t} &\le \sum_{c\in\Delta_k}\sum_{b\in c} q_b w_{c,t} \le \left(\sum_{c\in\Delta_k}\sum_{b\in c} q_b\right) w_{c_k,t} \\
&= \delta_k w_{c_k,t} \qquad \forall k \in \{1,\dots,n-1\}.
\end{aligned} \tag{35}$$

- If $k_o = n$ and we use full integrality conditions, then $w_{c_n,t} = y_{b,d,t} = 0$ for all $b \in c_n$. Thus, the validity of (34) for the PCPSP-F follows from (35).

On the other hand, if $k_o = n$ and we use the partial integrality condition, from (29) and (33), we obtain

$$\begin{aligned}
\sum_{b\in c_n} \alpha_b q_b y_{b,d,t} &\le \left(\sum_{b\in c_n} \alpha_b q_b\right) w_{c_n,t} \\
&\le \left(U_t^d - q(rcl(c_1) \setminus rcl(c_n))\right) w_{c_n,t} \\
&= \delta_n w_{c_n,t}.
\end{aligned} \tag{36}$$

Thus, the validity of (34) for the PCPSP-P derives from (35) and (36). □

#### 5.2.3. The Hourglass Cuts.

We now generalize the hourglass cuts for the PCPSP-C. These are a variant of the VRHS inequalities described above.

In its simplified form, each hourglass cut considers a cluster $\bar{c}$, a set of blocks $S$ contained in $b(cl(\bar{c}))$, and the set of blocks in $b(rcl(\bar{c}))$, as the name of the inequality suggests.

**Theorem 8.** *Let $1 \le t_1 \le t \le T$, let $\bar{c} \in \mathcal{C}$, and let $S \subseteq b(cl(\bar{c}) \setminus \{\bar{c}\})$ be such that*

$$q(S) \ge \sum_{s=t_1}^{t} U_s^d.$$

*Let $R = rcl(\bar{c}) \setminus \{\bar{c}\}$. Then, the inequality*

$$\begin{aligned}
&\left(q(S \cup \{\bar{c}\}) - \sum_{s=t_1}^{t} U_s^d\right) w_{\bar{c},t} + \sum_{b\in b(R)} q_b \sum_{s=t_1}^{t} y_{b,d,s} \\
&\quad \le \sum_{b\in S\cup\{\bar{c}\}} q_b \left(w_{c(b),t} - \sum_{s=t_1}^{t} y_{b,d,s}\right)
\end{aligned} \tag{37}$$

*is valid for PCPSP-C.*

**Proof.** If $w_{\bar{c},t} = 0$, then the left-hand side is equal to 0, and the inequality holds true. Henceforth, assume $w_{\bar{c},t} > 0$. Then, for each $b \in S$, $w_{c(b),t} = 1$. We know that

$$\sum_{b\in S} q_b y_{b,d,s} + \sum_{b\in\bar{c}} q_b y_{b,d,s} + \sum_{b\in b(R)} q_b y_{b,d,s} \le U_s^d \quad \forall s \le t.$$

Combining these inequalities, we obtain

$$\sum_{b\in S} q_b \sum_{s=t_1}^{t} y_{b,d,s} + \sum_{b\in\bar{c}} q_b \sum_{s=t_1}^{t} y_{b,d,s} + \sum_{b\in b(R)} q_b \sum_{s=t_1}^{t} y_{b,d,s} \le \sum_{s=t_1}^{t} U_s^d.$$

Moving the leftmost term to the right-hand side and adding $\sum_{b\in S} q_b w_{c(b),t}$ on both sides yields

$$\begin{aligned}
&\sum_{b\in S} q_b \underbrace{w_{c(b),t}}_{1} + \sum_{b\in\bar{c}} q_b \sum_{s=t_1}^{t} y_{b,d,s} + \sum_{b\in b(R)} q_b \sum_{s=t_1}^{t} y_{b,d,s} \\
&\quad \le \sum_{s=t_1}^{t} U_s^d + \sum_{b\in S} q_b w_{c(b),t} - \sum_{b\in S} q_b \sum_{s=t_1}^{t} y_{b,d,s} \\
&\implies q(S) - \sum_{s=t_1}^{t} U_s^d + \underbrace{\sum_{b\in\bar{c}} q_b \sum_{s=t_1}^{t} y_{b,d,s}}_{\sum_{b\in\bar{c}} q_b \left(w_{\bar{c},t} - w_{\bar{c},t} + \sum_{s=t_1}^{t} y_{b,d,s}\right)} + \sum_{b\in b(R)} q_b \\
&\quad \times \sum_{s=t_1}^{t} y_{b,d,s} \le \sum_{b\in S} q_b \left(w_{c(b),t} - \sum_{s=t_1}^{t} y_{b,d,s}\right)
\end{aligned}$$

$$\Longrightarrow q(S) - \sum_{s=t_1}^{t} U_s^d + \underbrace{\sum_{b\in\bar{c}} q_b w_{\bar{c},t}}_{q(\bar{c})w_{\bar{c},t}} + \sum_{b\in b(R)} q_b \sum_{s=t_1}^{t} y_{b,d,s}$$
$$\leq \sum_{b\in S\cup\{\bar{c}\}} q_b \left( w_{c(b),t} - \sum_{s=t_1}^{t} y_{b,d,s} \right)$$
$$\Longrightarrow \left( q(S) - \sum_{s=t_1}^{t} U_s^d + q(\bar{c}) \right) w_{\bar{c},t} + \sum_{b\in b(R)} q_b \sum_{s=t_1}^{t} y_{b,d,s}$$
$$\leq \sum_{b\in S\cup\{\bar{c}\}} q_b \left( w_{c(b),t} - \sum_{s=t_1}^{t} y_{b,d,s} \right).$$

This proves the validity of (37). □

Given the flexibility in the choice of the set $S$ above, we are also interested in finding *the best* possible $S$. It turns out that there is a simple expression for the set $S$ with maximum violation. Specifically, given a solution $(w^*, y^*)$ of the relaxation of the PCPSP-C, a cluster $\bar{c} \in \mathscr{C}$, and time periods $1 \leq t_1 \leq t \leq T$, the set $S^*$ which maximizes the violation in (37) is

$$S^* = \left\{ b \in \mathscr{B} : c(b) \prec \bar{c},\ w^*_{\bar{c},t} > w^*_{c(b),t} - \sum_{s=t_1}^{t} y^*_{b,d,s} \right\}. \quad (38)$$

If a block $\bar{b}$ such that $c(\bar{b}) \prec \bar{c}$ satisfies $w^*_{\bar{c},t} > w^*_{c(b),t} - \sum_{s=t_1}^{t} y^*_{b,d,s}$ and it is not in $S^*$, then adding it to $S^*$ would increase the violation of the inequality. If a block $\bar{b} \in S^*$ is such that $w^*_{\bar{c},t} < w^*_{c(b),t} - \sum_{s=t_1}^{t} y^*_{b,d,s}$, then removing it from $S^*$ would decrease it. Expression (38) allows us to quickly compute the most-violated hourglass cut, thus efficiently finding a deep cut in this family.

## 6. Heuristics

To quickly compute good feasible solutions to instances of PCPSP-C (and, consequently, strong lower bounds), we rely on a combination of heuristic techniques and a local search method. The first technique is a generalization of the TopoSort heuristic, originally described by Chicoisne et al. (2012) for C-PIT, to PCPSP-C. That is, this generalization considers arbitrary clusters and both full and partial integrality. The heuristic requires all the coefficients in constraint matrix $G$ to be nonnegative; thus, it is well suited for all instances of the OPPDP and for instances of OPPSP with $G \geq 0$. The second technique that we consider is based on converting an instance of PCPSP-C to an instance of PCPSP by fixing the destination of each block before the optimization. This technique potentially allows us to use the many different algorithms that have been proposed for C-PIT, although it is more general since it does not require $G$ to be nonnegative. Furthermore, if the number of clusters that define the problem is not too large, as is often the case, one can solve the resulting PCPSP directly with a mixed-integer programming solver. This makes the technique well suited for the OPPSP but not very effective for the OPPDP. We call this second technique the *1-DEST heuristic*. Finally, given an integer-feasible solution of PCPSP-C, it is possible to perform a simple local search procedure to improve the destination assignment of the blocks. One can apply this to any integer-feasible solution of PCPSP-C simply by solving an LP problem, which we call the *optimize-destinations* heuristic. We now describe these in more detail.

### 6.1. The Generalized TopoSort Heuristic [TopoSort]

The generalized TopoSort heuristic, presented in Algorithm 1, is a modified version of the expected-time TopoSort heuristic of Chicoisne et al. (2012) that expands the functionality of the algorithm from instances of the C-PIT to instances of the PCPSP-C. It takes as input a solution $(x^*, y^*)$ obtained by solving the LP-PCPSP-C and outputs a feasible solution $(\bar{x}, \bar{y})$ of the PCPSP-P satisfying the partial integrality constraint, as defined in (11). The heuristic assumes that matrix $G$, appearing in constraints (23), is such that $G \geq 0$ (i.e., all its components are nonnegative). It is straightforward to modify this heuristic such that its output satisfies full integrality (10).

**Algorithm 1** (The TopoSort Heuristic)

**Input:** An instance of the PCPSP-P, as defined in (1)–(7) with integrality constraints (11), such that $G \geq 0$ and $g \geq 0$. A solution $(x^*, y^*)$ of the LP relaxation.

**Output:** A solution $(\bar{x}, \bar{y})$ of a PCPSP-P instance.

**1** Let $\mathscr{T}^* = \mathscr{T} \cup \{T+1\}$.

**2** For each $c \in \mathscr{C}$, define:

$$\delta(c) = |\{c' \in \mathscr{C} : (c, c') \in \mathscr{A}\}|,$$
$$S_c = \{c' \in \mathscr{C} : (c', c) \in \mathscr{A}\},$$
$$x^*_{c,T+1} = 1 - \sum_{t\in\mathscr{T}} x^*_{c,t},$$
$$es(c) = \min\{t \in \mathscr{T}^* : x^*_{c,t} > 0\},$$
$$E(c) = \sum_{t\in\mathscr{T}^*} t x^*_{c,t}.$$

**3** $(\bar{x}, \bar{y}) \leftarrow (0, 0)$.

**4** **while** $\mathscr{C} \neq \emptyset$ **do**

**5** Choose $c \in \mathscr{C}$ that solves $\min\{E(c) : \delta(c) = 0,\ es(c) \leq T\}$.

**6** **for** $b \in c$ **do**

**7** $d(b) \leftarrow \operatorname{argmax}\{\sum_{t\in\mathscr{T}} y^*_{b,d,t} : d \in \mathscr{D}\}$.

**8** $t \leftarrow es(c)$.

**9** $w \leftarrow 0$.

**10** **while** $w < 1$ *and* $t \leq T$ **do**

**11** $\bar{\alpha} \leftarrow \max\{\alpha : G\bar{y} \leq g,\ w + \alpha \leq 1, \bar{y}_{b,d(b),t} = \alpha,\ \forall b \in c\}$.

**12** $\bar{y}_{b,d(b),t} \leftarrow \bar{\alpha}$ for all $b \in c$.

**13** $\bar{x}_{c,t} \leftarrow \bar{\alpha}$.

**14** $w \leftarrow (w + \bar{\alpha})$.

15 $\quad t \leftarrow (t+1)$.
16 **for** $a \in S_c$ **do**
17 $\quad \delta(a) \leftarrow \delta(a) - 1$.
18 $\quad es(a) \leftarrow \max\{es(a), t\}$.
19 $\mathscr{C} \leftarrow \mathscr{C} \setminus \{c\}$.

Given a DAG, a topological ordering is such that if a node $u$ comes before a node $v$, then there cannot exist an arc going from $u$ to $v$. It is well known that every DAG admits a topological ordering and that computing it can be done in linear time (see Cook et al. 1998). In terms of the open pit production scheduling problem, a topological ordering corresponds to an extractable ordering of the blocks. That is, the position of every block $b$ in the ordering is such that all its predecessors appear before it.

The TopoSort heuristic presented in Chicoisne et al. (2012) works by first sorting all blocks in topological order. Following this order, each block is scheduled as early as possible, that is, in the first time period with sufficient resources and no earlier than its predecessors. Once a block is scheduled, the available resources are updated, and the algorithm continues by scheduling the next block. Because many topological orderings are available, the heuristic uses the optimal solution of the LP relaxation to guide the sorting process. See Chicoisne et al. (2012) for details.

The generalized TopoSort algorithm that we present in Algorithm 1 exhibits several important differences from the TopoSort heuristic described by Chicoisne et al. (2012). First, it schedules clusters rather than blocks. Second, it can handle partial integrality condition (11). Third, it forces clusters to be extracted no earlier than the first time period in which the corresponding variables in the LP relaxation have nonzero values. This last modification, implemented in Step 8 of Algorithm 1, prevents negative-valued extractions from occurring too early in the schedule. Fourth, we break ties with the coefficients in the objective function.

### 6.2. The 1-DEST Heuristic

This heuristic proceeds in three steps. First, it uses the optimal solution $(x^*, y^*)$ of LP-PCPSP-C to assign a fixed destination to each block for each period. Second, it constructs a small instance of PCPSP with a single destination (and no clusters). This is done by substituting out all of the $y$ variables and obtaining a problem only in terms of the $x$ variables. Third, the resulting problem is solved directly by using a mixed-integer programming solver.

Specifically, for each $b \in \mathscr{B}$, $d \in \mathscr{D}$, let us define

$$\bar{y}_{b,d} = \frac{\sum_{t\in\mathscr{T}} y^*_{b,d,t}}{\sum_{d'\in\mathscr{D}} \sum_{t\in\mathscr{T}} y^*_{b,d',t}}.$$

Given these values, we impose the following condition for each $c \in \mathscr{C}$, $b \in c, d \in \mathscr{D}$, and $t \in \mathscr{T}$:

$$y_{b,d,t} = x_{c,t} \cdot \bar{y}_{b,d}. \tag{37}$$

Substitute Equations (37) in constraints (3)–(6) of the PCPSP-C formulation to obtain

$$\begin{aligned}
\max \ & \sum_{c\in\mathscr{C}} \sum_{t\in\mathscr{T}} \bar{p}_{c,t} x_{c,t} \\
\text{s.t.} \ & \sum_{t\in\mathscr{T}} x_{c,t} \le 1 && \forall c \in \mathscr{C}, \\
& \sum_{t'=1}^{t} x_{c,t'} \le \sum_{t'=1}^{t} x_{c',t'} && \forall (c,c') \in \mathscr{A}, t \in \mathscr{T}, \\
& \bar{G}x \le g \\
& x_{c,t} \ge 0 && \forall c \in \mathscr{C}, t \in \mathscr{T}, \\
& x_{c,t} \text{ satisfies an integrality condition} && \forall c \in \mathscr{C}, t \in \mathscr{T},
\end{aligned}$$

where $\bar{p}$ and $\bar{G}$ are naturally defined from the substitution.

Solve this problem to optimality using a mixed-integer programming solver, and use (37) to determine the value of the associated $y$ variables. Depending on the integrality conditions used in the last optimization problem, we construct a solution to either PCPSP-F or PCPSP-P.

### 6.3. The Optimize-Destinations Heuristic

Consider a feasible mixed-integer solution $(x, y)$ of PCPSP-C. A simple method to obtain an improved solution $(x, y')$ is to fix the $x$ variables and reoptimize the $y$ variables with an LP solver (e.g., with the BZ algorithm proposed by Bienstock and Zuckerberg (2009, 2010)). This is a natural process after running the generalized TopoSort or 1-DEST heuristics, which can only improve the solution.

## 7. Branch-and-Bound

We now introduce a simple branch-and-bound algorithm to solve PCPSP-C to optimality. In our description of the algorithm, we assume that we are interested in computing a solution which satisfies the partial integrality constraint (11). Imposing integrality constraint (10) instead is analogous. This algorithm does not make any assumptions about the matrix $G$ appearing in the constraints (5). For background about the branch-and-bound algorithm, see Nemhauser and Wolsey (1989). The main characteristics of our branch-and-bound algorithm are the following:

- The nodes of the branch-and-bound tree can be explored in a depth-first search or a best-bound-first search. In our computational runs, we used the best-bound-first-search rule. We use strong branching at every node of the tree to select variables.

**Table 1.** Description of Instances Used for Computational Tests

| | | | | OPPDP | | OPPSP base | | | OPPSP extended | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $\lvert\mathcal{D}\rvert$ | $\lvert\mathcal{T}\rvert$ | mcap | $\lvert\mathcal{B}\rvert$ | nvars | $\lvert\mathcal{C}\rvert$ | $\lvert\mathcal{B}\rvert$ | nvars | $\lvert\mathcal{C}\rvert$ | $\lvert\mathcal{B}\rvert$ | nvars |
| *calbuco* | 3 | 21 | n | 5,016,971 | 316,069,173 | 324 | 200,241 | 8,416,926 | 548 | 200,241 | 12,615,183 |
| *chaiten* | 2 | 25 | y | 339,199 | 16,959,950 | 273 | 288,073 | 7,208,650 | 273 | 288,073 | 7,208,650 |
| *guallatari* | 3 | 21 | y | 1,672,198 | 105,348,474 | 272 | 57,527 | 2,421,846 | 272 | 57,527 | 2,421,846 |
| *kd* | 2 | 12 | n | 14,153 | 339,672 | 53 | 10,128 | 122,172 | 93 | 10,128 | 243,072 |
| *marvin* | 2 | 20 | y | 53,271 | 2,130,840 | 56 | 8,515 | 171,420 | 56 | 8,515 | 171,420 |
| *mclaughlin* | 2 | 20 | n | 2,140,342 | 85,613,680 | 173 | 180,749 | 3,618,440 | 173 | 180,749 | 3,618,440 |
| *mclaughlin_limit* | 2 | 15 | n | 112,687 | 3,380,610 | 166 | 110,768 | 1,664,010 | 166 | 110,768 | 1,664,010 |
| *palomo* | 2 | 40 | y | 772,800 | 61,824,000 | 74 | 190,319 | 7,615,720 | 123 | 190,319 | 15,225,520 |
| *ranokau* | 2 | 81 | y | 1,873,035 | 303,431,670 | 186 | 317,907 | 25,765,533 | 186 | 317,907 | 25,765,533 |
| *tronador* | 2 | 20 | y | 329,859 | 18,801,963 | 220 | 30,099 | 1,805,940 | 220 | 30,099 | 1,805,940 |

*Notes.* mcap indicates the presence of mining capacity constraints (8). All instances have destination capacity constraints (9), and nvars indicates the number of variables in the problem before preprocessing.

- The LP at each node of the tree is solved with the BZ algorithm, using the features described by Muñoz et al. (2018).
- When using the partial integrality condition (11), a solution $x^*$ is considered "fractional" if there exist an arc $(c^1, c^2) \in \mathcal{A}$ and a time period $t \in \mathcal{T}$ such that

$$\sum_{t'=1}^{t} x^*_{c^1,t'} > 0 \text{ and } \sum_{t'=1}^{t} x^*_{c^2,t'} < 1.$$

In this case, $(c^1, c^2) \in \mathcal{A}$ and $t \in \mathcal{T}$ define a possible branching point. The two branches to consider for this point are

$$\sum_{t'=1}^{t} x_{c^1,t'} = 0 \quad \text{(the down-branch)}, \qquad \text{and}$$

$$\sum_{t'=1}^{t} x_{c^2,t'} = 1 \quad \text{(the up-branch)}. \tag{40}$$

If, after branching on $(c^1, c^2)$ and $t$, we find that one of the branches is pruned, the parent node can be strengthened as follows.

—If the down-branch is pruned, all integral solutions at the parent node must satisfy $\sum_{t'=1}^{t} x_{c^1,t'} > 0$. This, in turn, implies that all integral solutions at the parent node must satisfy the following constraint, which can be added to the node:

$$\sum_{t'=1}^{t} x_{c,t'} = 1 \qquad \text{for all } c \text{ such that } (c^1, c) \in \mathcal{A}. \tag{41}$$

—If the up-branch is pruned, integral solutions at the parent node must satisfy $\sum_{t'=1}^{t} x_{c^2,t'} < 1$. This, in turn, implies that all integral solutions at the parent node must satisfy the following constraint, which can be added to the node:

$$\sum_{t'=1}^{t} x_{c,t'} = 0 \qquad \text{for all } c \text{ such that } (c, c^2) \in \mathcal{A}. \tag{42}$$

- The solver only adds cutting planes to the root node.

## 8. Computational Study

The primary goal of our computational study is to determine if the proposed methodology can be used to solve real problem instances of OPPDP and OPPSP. The secondary goal of our study is to determine the contribution of each feature to the overall effectiveness of the methodology. To conduct our study, we implemented the methodology described in this paper, including all the features covered in Sections 4–7, and collected ten mine planning instances which could be used to solve instances of both OPPDP and OPPSP.

Table 1 summarizes the numerical characteristics of our test instances. These instances all correspond to real mines, with the exception of *marvin*. Moreover, four of these instances (*kd*, *marvin*, *mclaughlin*, and *mclaughlin_limit*) are publicly available in the MineLib repository (see Espinoza et al. 2012). The remaining instances were provided by industry partners, whose names we have changed for confidentiality reasons.

Using each of these datasets, we created one instance of OPPDP and two instances of OPPSP. We refer to the latter two as the *OPPSP base instance* and the *OPPSP extended instance*.

To build each OPPSP base instance, we computed clusters for the corresponding problems as follows. A mining engineer used the GEOVIA Whittle (Dassault Systèmes 2019) strategic mine planning software to compute four phases for each instance using the "Automatic Pushback Generator" feature (Milawa Net Present Value (NPV) algorithm). The clusters used for our models consist of the bench-phases obtained by using these phases. Each instance has between 50 and 300 clusters. To build each OPPSP extended instance, each OPPSP base instance was modified in one of four ways:

- For instances *calbuco*, *kd*, and *palomo*, the mining engineer computed clusters using seven phases, rather than four.

**Table 2.** Percentage of Variables in OPPDP Instances Remaining After Preprocessing

| Instance | Default w/o ES | Default w/o DTE | Default w/o UPIT | Default |
|---|---|---|---|---|
| *calbuco* | 3.19 | 3.95 | 68.46 | 3.19 |
| *chaiten* | 53.73 | 84.06 | 61.04 | 53.16 |
| *guallatari* | 2.35 | 3.43 | 43.46 | 2.32 |
| *kd* | 85.88 | 85.88 | 100.00 | 85.88 |
| *marvin* | 12.07 | 15.88 | 53.67 | 11.96 |
| *mclaughlin* | 6.24 | 8.44 | 52.22 | 6.24 |
| *mclaughlin_limit* | 77.76 | 98.30 | 78.62 | 77.76 |
| *palomo* | 7.46 | 12.39 | 46.51 | 7.31 |
| *ranokau* | 11.89 | 15.31 | 46.33 | 11.01 |
| *tronador* | 29.21 | 32.24 | 70.33 | 28.75 |

- For instances *chaiten*, *marvin*, and *mclaughlin*, we added minimum processing constraints,

$$\sum_{b\in\mathcal{B}} q_b y_{b,d,t} \geq L_t^d \qquad \forall t \in \mathcal{T},$$

where $d \in \mathcal{D}$ is the mill or the concentrator destination. These constraints are fairly common and seek to ensure a constant yearly production and usage of the concentrator.

- For instances *guallatari*, *mclaughlin_limit*, and *ranokau*, we imposed what is known as a flow-balance constraint, that is, a constraint of the form

$$\sum_{b\in\mathcal{B}}\sum_{d\in\mathcal{D}} q_b y_{b,d,t+1} \leq (1+\alpha)\sum_{b\in\mathcal{B}}\sum_{d\in\mathcal{D}} q_b y_{b,d,t} \qquad \forall t \in \{1,\ldots,T-1\}. \tag{43}$$

These constraints are fairly common in industry since, in practice, it is not feasible to radically change the size of the workforce and fleet of trucks from one year to another. For example, Dassault Whittle offers a heuristic called Milawa balance which imposes this condition. In our runs, we used $\alpha = 0.0$, which produces solutions with characteristics similar to those produced by the default Milawa balance algorithm.

**Table 3.** Objective Values of OPPDP Instances as a Percentage of TopoSort Objective Value

| Instance | LP relaxation | Total time |
|---|---|---|
| *calbuco* | 102.06 | 13m 5s |
| *chaiten* | 100.33 | 26m 55s |
| *guallatari* | 101.22 | 2m 16s |
| *kd* | 100.87 | 2.8s |
| *marvin* | 102.49 | 4.5s |
| *mclaughlin* | 100.21 | 4m 55s |
| *mclaughlin_limit* | 100.16 | 1m 36s |
| *palomo* | 101.10 | 12m 12s |
| *ranokau* | 102.22 | 9h 39m 13s |
| *tronador* | 102.47 | 3m 13s |
| Geometric mean | 101.31 | — |

- In the case of instance *tronador*, we imposed a blending constraint. These constraints are typically used to limit the weighted average of a given contaminant sent to a fixed destination $d \in \mathcal{D}$. For each $b \in B$, we let $q'_b$ represent the tons of prohibited material in $b$, and since we wish to impose that the weighted average grade of this material is less than $\beta_t$ in each period $t \in \mathcal{T}$, we add the following constraint:

$$\frac{\sum_{b\in\mathcal{B}} q'_b y_{b,d,t}}{\sum_{b\in\mathcal{B}} q_b y_{b,d,t}} \leq \beta_t \qquad \forall t \in \mathcal{T} \setminus \{1\}.$$

By multiplying the denominator, this constraint becomes linear and can be added to the model. In the specific case of *tronador*, the constraint is used to limit the weighted average amount of arsenic sent to the mill.

The constraints added to the OPPSP extended instances made the OPPSP base instance solutions infeasible. Moreover, with the exception of the "seven phases" cases, the TopoSort and UPIT methods cannot be used in any of the OPPSP extended instances.

All our algorithms were implemented using the C programming language, with CPLEX® 12.6 as the optimization solver. The computer servers employed for computations use Linux 2.6.32 on x86_64 architecture, with four eight-core Intel® Xeon® E5-2670 processors and 128 GB of RAM. Results analyze the performance with different sets of activated features and are given as normalized geometric means.

### 8.1. Effectiveness of the Methodology on Instances of the OPPDP

We began by testing the effectiveness of our three proposed preprocessing techniques: ultimate pit limit preprocessing (UPIT), dominated triplet elimination (DTE), and early start cuts (ES), as described in Sections 4.1, 4.2, and 5.1, respectively. We ran the preprocessing algorithm four times on each instance: one with all our preprocessing techniques (we call this our default preprocessing methodology), and then we preprocessed each instance by using our default preprocessing with exactly one of the features disabled. Table 2 describes the results of these experiments. To facilitate comparisons, we present the resulting number of variables as a percentage of the original number of variables (originally given in Table 1). As can be appreciated from the fifth column, the effect of preprocessing can be significant. This is illustrated by the *calbuco* instance, where only 3.19% of the variables remained after preprocessing. However, in the other instances, the effect is only moderate. By observing columns two through four, it is noticeable that all three techniques (UPIT, DTE, and ES) are effective. However, disabling UPIT has the

**Table 4.** Percentage of Variables in OPPSP Instances Remaining After Preprocessing

| Instance | Default w/o ES | Default w/o DTE | Default w/o UPIT | Default w/o AGG | Default |
|---|---|---|---|---|---|
| *calbuco* | 30.56 | 50.25 | 30.56 | 70.94 | 30.56 |
| *chaiten* | 23.21 | 24.09 | 15.96 | 17.96 | 15.96 |
| *guallatari* | 51.09 | 80.58 | 41.61 | 42.20 | 41.61 |
| *kd* | 42.97 | 42.97 | 42.97 | 100.00 | 42.97 |
| *marvin* | 51.29 | 54.93 | 43.41 | 43.41 | 43.41 |
| *mclaughlin* | 8.45 | 10.59 | 8.45 | 47.94 | 8.45 |
| *mclaughlin_limit* | 12.06 | 14.96 | 12.06 | 58.28 | 12.06 |
| *palomo* | 1.36 | 1.44 | 1.46 | 6.04 | 1.26 |
| *ranokau* | 45.10 | 33.26 | 21.46 | 21.47 | 21.46 |
| *tronador* | 67.04 | 74.43 | 50.55 | 68.84 | 50.55 |

most detrimental effect. After preprocessing these instances, we solved the LP relaxation of the corresponding PCPSP-C instance using the BZ algorithm, as described in Muñoz et al. (2018). Then, we used the TopoSort heuristic, as described in Section 6.1, to compute integer-feasible solutions for each instance and the optimize-destinations heuristic, as described in Section 6.3, to improve them.

The results are presented in Table 3. To facilitate comparisons, in the second column, we show the objective function value of the LP relaxation as a percentage of the solution obtained via the TopoSort heuristic. The total time taken by the algorithm, including the time to solve the LP relaxation, is presented in column three. It should be further noted that the time required for the TopoSort and optimize-destination heuristics is negligible (less than one second). In Table 3, we observe that, with the exception of the *ranokau* instance, it is possible to compute very high-quality solutions to these problems in just minutes. For example, consider the *guallatari* instance. In just over two minutes (2 m 16 s), it is possible to obtain an integer-feasible solution within 1.3% of optimality. The LP relaxation of *ranokau* is difficult to solve owing to the number of time periods it contains. However, the quality of the solution

**Table 5.** Effect of Cutting Planes on Upper Bound Obtained from LP Relaxation in OPPSP Instances

| Instance | No cuts | No early start | No lifted clique | No VRHS | No hourglass | Ext. cuts | Pro. cuts | All cuts |
|---|---|---|---|---|---|---|---|---|
| Base instances | | | | | | | | |
| *calbuco* | 108.28 | 102.42 | 102.42 | 102.86 | 104.83 | 108.28 | 102.42 | 102.42 |
| *chaiten* | 117.26 | 102.01 | 101.34 | 100.03 | 100.29 | 100.88 | 109.23 | 100.00 |
| *guallatari* | 102.02 | 101.09 | 100.54 | 100.54 | 100.71 | 100.87 | 101.09 | 100.54 |
| *kd* | 101.75 | 100.21 | 100.21 | 100.22 | 101.10 | 101.75 | 100.21 | 100.21 |
| *marvinml* | 105.75 | 101.10 | 100.61 | 100.63 | 101.46 | 103.06 | 101.10 | 100.61 |
| *mclaughlin* | 102.52 | 100.34 | 100.34 | 100.37 | 101.37 | 102.52 | 100.34 | 100.34 |
| *mclaughlinlimit* | 102.39 | 100.25 | 100.25 | 100.18 | 101.43 | 102.39 | 100.25 | 100.25 |
| *palomo* | 114.87 | 103.62 | 101.26 | 101.33 | 103.55 | 111.37 | 103.62 | 101.26 |
| *ranokau* | 131.48 | 104.76 | 101.82 | 101.87 | 102.10 | 104.96 | 105.20 | 101.82 |
| *tronador* | 108.84 | 101.94 | 100.83 | 100.80 | 100.83 | 100.90 | 104.00 | 100.80 |
| Geometric mean | 109.17 | 101.76 | 100.96 | 100.88 | 101.76 | 103.65 | 102.71 | 100.82 |
| Extended instances | | | | | | | | |
| *calbuco* | 105.31 | 101.87 | 101.87 | 102.20 | 103.31 | 105.31 | 101.87 | 101.87 |
| *chaiten* | 117.86 | 101.73 | 101.68 | 100.14 | 100.48 | 100.90 | 109.90 | 100.12 |
| *guallatari* | 102.47 | 102.01 | 101.34 | 101.36 | 101.43 | 101.54 | 102.01 | 101.34 |
| *kd* | 100.70 | 100.25 | 100.25 | 100.28 | 100.66 | 100.70 | 100.25 | 100.25 |
| *marvinml* | 110.56 | 104.62 | 103.78 | 103.79 | 104.70 | 107.20 | 104.62 | 103.78 |
| *mclaughlin* | 103.24 | 100.85 | 100.85 | 100.90 | 101.95 | 103.24 | 100.85 | 100.85 |
| *mclaughlinlimit* | 103.16 | 100.75 | 100.75 | 100.77 | 102.09 | 103.16 | 100.75 | 100.75 |
| *palomo* | 111.53 | 103.68 | 101.75 | 101.88 | 103.04 | 108.07 | 103.68 | 101.75 |
| *ranokau* | 131.10 | 104.92 | 101.82 | 100.33 | 100.31 | 104.86 | 105.17 | 100.15 |
| *tronador* | 117.07 | 105.51 | 104.11 | 103.39 | 103.59 | 103.60 | 109.25 | 103.39 |
| Geometric mean | 109.95 | 102.60 | 101.81 | 101.50 | 102.15 | 103.83 | 103.78 | 101.42 |

**Table 6.** Effect of Branch-and-Bound on the Upper Bound After Four Hours of Branching in OPPSP Instances

| | Base instances | | | | Extended instances | | | |
|---|---|---|---|---|---|---|---|---|
| | No cuts | | All cuts | | No cuts | | All cuts | |
| Instance | Root | BB4 | Root | BB4 | Root | BB4 | Root | BB4 |
| *calbuco* | 108.28 | 103.76 | 102.42 | 102.12 | 105.31 | 103.03 | 101.87 | 101.61 |
| *chaiten* | 117.26 | 103.83 | 100.00 | 100.00 | 117.86 | 109.77 | 100.12 | 100.04 |
| *guallatari* | 102.02 | 101.18 | 100.54 | 100.26 | 102.47 | 101.92 | 101.34 | 100.67 |
| *kd* | 101.75 | 100.00 | 100.21 | 100.00 | 100.70 | 100.00 | 100.25 | 100.00 |
| *marvinml* | 105.75 | 100.01 | 100.61 | 100.00 | 110.56 | 101.07 | 103.78 | 100.01 |
| *mclaughlin* | 102.52 | 100.27 | 100.34 | 100.08 | 103.24 | 100.88 | 100.85 | 100.45 |
| *mclaughlinlimit* | 102.39 | 100.00 | 100.25 | 100.01 | 103.16 | 100.62 | 100.75 | 100.29 |
| *palomo* | 114.87 | 102.75 | 101.26 | 100.14 | 111.53 | 106.52 | 101.75 | 100.76 |
| *ranokau* | 131.48 | 126.43 | 101.82 | 101.82 | 131.10 | 123.41 | 100.15 | 100.15 |
| *tronador* | 108.84 | 101.41 | 100.80 | 100.32 | 117.07 | 110.02 | 103.39 | 102.44 |
| Geometric mean | 109.17 | 103.71 | 100.82 | 100.47 | 109.95 | 105.51 | 101.42 | 100.64 |

obtained by the heuristics is still very good, showing a gap of 2.22%. Overall, the geometric mean of gaps is only 1.25%, confirming that the bounds provided by LP relaxation of OPPDP instances tend to be very tight (as observed by Bienstock and Zuckerberg (2009)).

## 8.2. Effectiveness of the Methodology on Instances of OPPSP

In addition to testing the UPIT, DTE, and ES techniques, as we did in Section 8.1, we also tested the aggregation (AGG) technique, as described in Section 4.3. Note that the ES technique is considered for both the preprocessing and cutting planes analysis. Our default preprocessing option consists of applying all four techniques. The results of our experiments are summarized in Table 4. In the sixth column, we can observe that the effect of preprocessing is also significant for the OPPSP; however, the most efficient techniques change with respect to OPPDP.

Next, we analyze the tightness of the LP relaxation bounds for these instances and the effectiveness of the cutting planes described in Section 5 for improving these bounds. To do so, we compare the value of the LP relaxation to the value of the best-known integer-feasible solution of each instance by taking the ratio of these two values. This ratio is computed in six different manners for each instance: once with no cutting planes, once with our default options (early start, lifted clique, hourglass, and VRHS cuts), and once with our default options minus one of the cutting plane classes. Our cutting plane approach is simple. In each round, we generate all possible cuts but only add the 1,000 most violated ones. We run no more than ten rounds of cuts. Cuts are eliminated in

**Table 7.** Effectiveness of Heuristics and Branch-and-Bound Computing Feasible Integer Solutions in OPPSP Instances

| | Base instances | | | | | | Extended instances | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | No cuts | | | All cuts | | | No cuts | | All cuts | |
| Instance | Topo | 1-DEST | BB4 | Topo | 1-DEST | BB4 | 1-DEST | BB4 | 1-DEST | BB4 |
| *calbuco* | 97.68 | 97.68 | 97.68 | 94.65 | 98.07 | 98.07 | 97.92 | 97.92 | 98.38 | 98.38 |
| *chaiten* | 95.67 | 99.53 | 99.53 | 99.15 | 100.00 | 100.00 | 98.18 | 98.18 | 99.96 | 99.96 |
| *guallatari* | 99.66 | 99.66 | 99.66 | 99.66 | 99.66 | 99.66 | 99.24 | 99.24 | 99.33 | 99.33 |
| *kd* | 99.86 | 99.97 | 100.00 | 99.95 | 99.96 | 100.00 | 99.98 | 100.00 | 99.97 | 100.00 |
| *marvinml* | 98.00 | 99.63 | 100.00 | 99.37 | 99.89 | 100.00 | 99.03 | 99.03 | 99.46 | 99.99 |
| *mclaughlin* | 98.75 | 98.75 | 99.34 | 99.62 | 99.62 | 99.62 | 98.63 | 98.63 | 99.40 | 99.40 |
| *mclaughlinlimit* | 99.47 | 99.47 | 100.00 | 99.88 | 99.88 | 99.99 | 99.30 | 99.30 | 99.56 | 99.56 |
| *palomo* | 96.55 | 98.61 | 98.61 | 93.77 | 98.67 | 98.67 | 99.24 | 99.24 | 99.01 | 99.01 |
| *ranokau* | 92.63 | 98.21 | 98.21 | 95.30 | 97.94 | 97.94 | 99.78 | 99.78 | 99.69 | 99.69 |
| *tronador* | 98.12 | 99.69 | 99.69 | 99.45 | 99.79 | 99.79 | 94.93 | 94.93 | 99.94 | 99.94 |
| Geometric mean | 97.62 | 99.12 | 99.27 | 98.05 | 99.35 | 99.37 | 98.61 | 98.62 | 99.47 | 99.52 |

*Note.* Topo corresponds to the value obtained from using the TopoSort heuristic, 1-DEST to the value of the 1-DEST heuristic, and BB4 correspond to the values obtained by using the branch-and-bound algorithm with a time limit of four hours.

**Table 8.** Effect of Branch-and-Bound on the Proved Gap in OPPSP Instances

| | Base instances | | Extended instances | |
|---|---|---|---|---|
| Instance | Before BB4 | After BB4 | Before BB4 | After BB4 |
| *calbuco* | 2.70% | 2.37% | 1.87% | 1.63% |
| *chaiten* | 0.00% | 0.00% | 0.13% | 0.04% |
| *guallatari* | 0.63% | 0.36% | 1.33% | 0.67% |
| *kd* | 0.26% | 0.00% | 0.31% | 0.00% |
| *marvinml* | 0.71% | 0.00% | 4.16% | 0.01% |
| *mclaughlin* | 0.66% | 0.41% | 0.99% | 0.60% |
| *mclaughlinlimit* | 0.37% | 0.01% | 0.90% | 0.44% |
| *palomo* | 2.43% | 1.33% | 1.95% | 0.99% |
| *ranokau* | 2.06% | 2.06% | 0.31% | 0.31% |
| *tronador* | 0.80% | 0.32% | 3.34% | 2.44% |
| Geometric mean | 1.06% | 0.68% | 1.52% | 0.71% |

subsequent rounds if the corresponding dual variables are null. As is disclosed later, the objective function value of these integer-feasible solutions is within 1% of optimality for all instances. Thus, these ratios provide a very good estimate of the tightness of the bounds and the effectiveness of the cutting planes in improving these bounds (see Table 5).

Note that the ratios have been multiplied by 100 to facilitate readability. Thus, if the value of a ratio is 100, the LP relaxation has the same objective function value as the best-known integer-feasible solution (i.e., the bound is tight). The values in the second column of Table 5 provide evidence regarding how weak the LP relaxation bounds computed without cutting planes can be. On average, this bound is within 10% of optimality, but in some instances, it can be significantly worse (more than 30% in both *ranokau* instances and almost 20% in the *chaiten* instances), which stands in contrast to what we observe for the OPPDP instances. Nonetheless, the seventh column shows a significant tightening of the bounds after adding the cutting planes. In general, all cutting planes help improve the bounds (columns three through six); however, it seems that the hourglass cuts have the most significant impact. It is also worth pointing out that the behaviors of the base and extended instances are very similar. As would be expected, the gaps of the extended instances are marginally worse, but the cuts seem to help in both types of problems.

Table 6 demonstrates the impact which branching for four hours has upon the upper bounds provided by the LP relaxation as computed with and without cutting planes. We do not add cuts at nodes of the branch-and-bound tree in any of the instances. When comparing the third and fourth columns, in addition to the seventh and eighth columns, we can see that in most instances, branching for four hours is less effective than adding cuts. On the other hand, by comparing columns four and five, in addition to columns eight and nine, we can see that branching does improve the bound computed by the cutting planes. In fact, branching reduces the average ratio to 100.47 for the extended instances and 100.64 for the base ones.

Table 7 depicts the effectiveness of our different techniques for computing integer-feasible solutions to OPPSP (i.e., lower bounds). Specifically, we analyze the performance of the TopoSort heuristic (Topo), running the 1-DEST reduction heuristic, and branching with a time limit of four hours (BB4). All values are normalized relative to the best upper bound computed for each instance (which would have a value of 100) using branch-and-bound with cutting planes. Note that we cannot use the TopoSort heuristic on the extended problem sets (since the matrix $G$ appearing in the PCPSP-C has negative coefficients).

From the second column, as in the OPPDP instances, we observe that the TopoSort heuristic is very effective. However, the 1-DEST heuristic seems to compute slightly better solutions and has the advantage of being able to run for both the base and extended problem instances. Table 7 also shows that using the solutions computed from the LP relaxation with cutting planes results in better feasible solutions for all methods. Finally, the table shows that the best

**Table 9.** Relevant Times for Algorithms in OPPSP Instances

| | Base instances | | | | Extended instances | | | |
|---|---|---|---|---|---|---|---|---|
| Instance | LP no cuts | LP + cuts | 1-DEST | BB | LP no cuts | LP + cuts | 1-DEST | BB |
| *calbuco* | 10s | 4m 42.9s | 1m 3.9s | > 4h | 8.5s | 4m 4.6s | 7m 51.2s | > 4h |
| *chaiten* | 9.9s | 1m 26.4s | 5m 41.4s | 8.1s | 17s | 2m 23.7s | 1m 28.2s | > 4h |
| *guallatari* | 3.5s | 23.4s | 5m 26.7s | > 4h | 6.4s | 38.6s | 7m 13.9s | > 4h |
| *kd* | 0.2s | 0.9s | 0.7s | 38.5s | 0.1s | 0.7s | 1.3s | 51.5s |
| *marvinml* | 0.4s | 2s | 2.4s | 15m 54.1s | 0.4s | 3.5s | 1.6s | 3h 50m 50s |
| *mclaughlin* | 2.1s | 12.4s | 6.3s | > 4h | 3.4s | 22.2s | 7.7s | > 4h |
| *mclaughlinlimit* | 1.1s | 5.2s | 5.9s | 2h 19m 44.3s | 1.5s | 5.6s | 26.6s | > 4h |
| *palomo* | 3.4s | 29.6s | 21.7s | > 4h | 3.7s | 31.7s | 1m 19s | > 4h |
| *ranokau* | 9m 19.8s | 6m 12.6s | 13m 9.3s | > 4h | 10m 36.2s | 14m 55.5s | 41m 12.2s | > 4h |
| *tronador* | 2.9s | 9.8s | 17.5s | > 4h | 33.8s | 1m 45.4s | 6m 58.2s | > 4h |

solutions are typically computed with the 1-DEST heuristic, although branching sometimes yields slightly better solutions. This is significant because it shows that provably near-optimal solutions can be computed without the need for customized branch-and-bound algorithms, which are difficult to implement.

A summary of the performance of our default features on instances of OPPSP is presented in Table 8. Here, the gaps are computed as $gap = \frac{ub-lb}{ub}$, where $lb$ is the best bound computed by the branch-and-bound algorithm and $ub$ is the value of the best integer-feasible solution amongst those computed by the 1-DEST and BB4 algorithms. We refer to this as the *proved gap*, since it is the best gap that can be computed using the information obtained from the algorithm execution and not using the best-known feasible solution to the problem (possibly computed using some other method). As Table 8 indicates, the algorithms perform very well, obtaining integer-feasible solutions that, on average, are below 1%. Although we failed to compute solutions within 1% of optimality for some instances, the results were still within 4% throughout.

Finally, a summary of the times required to compute these solutions is presented in Table 9. In practice, 1-DEST often takes significantly less than four hours to compute high-quality feasible solutions. In fact, in most problems, solutions can be computed in just minutes. Table 9 also shows that, in just under a third of the instances, we were able to terminate the branch-and-bound run within the four hours. Though this suggests that there is room to improve these results through further research on the branch-and-bound methodology, by combining these overall results with those presented in Table 8, it is possible to see that our methodology is effective in addressing the proposed problem and efficient in terms of the overall time required.

## References

Askari-Nasab H, Awuah-Offei K, Eivazy H (2010) Large-scale open pit production scheduling using mixed integer linear programming. *Internat. J. Mining Mineral Engrg.* 2(3):185–214.

Askari-Nasab H, Pourrahimian Y, Ben-Awuah E, Kalantari S (2011) Mixed integer linear programming formulations for open pit production scheduling. *J. Mining Sci.* 47(3):338–359.

Bienstock D, Zuckerberg M (2009) A new LP algorithm for precedence constrained production scheduling. Working paper, Columbia University, New York.

Bienstock D, Zuckerberg M (2010) Solving LP relaxations of large-scale precedence constrained problems. *Proc. 14th Conf. Integer Programming Combinatorial Optim. (IPCO)* (Mathematical Optimization Society, Philadelphia), 1–14.

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

Boland N, Dumitrescu I, Froyland G (2008) A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology. *Math. Optim. Soc.* (2008):1–33.

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

Boyd EA (1993) Polyhedral results for the precedence-constrained knapsack problem. *Discrete Appl. Math.* 41(3):185–201.

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

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

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

Cullenbine C, Wood K, Newman AM (2011) A sliding time window heuristic for open pit mine block sequencing. *Optim. Lett.* 5(3): 365–377.

Dagdelen K (1985) Optimum multi-period open pit mine production scheduling. Unpublished doctoral dissertation, Colorado School of Mines, Golden.

Dagdelen K, Johnson TB (1986) Optimum open pit mine production scheduling by Lagrangian parameterization. Ramani RV, ed. *19th APCOM Sympos. Soc. Mining Engineers* (Society for Mining, Metallurgy, and Exploration, Englewood, CO), 127–142.

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Oper. Res.* 8(1):101–111.

Dassault Systèmes (2019) GEOVIA Whittle. Last accessed January 3, 2020, https://www.3ds.com/products-services/geovia/products/whittle/.

Deswik.CAD (2017) Deswik.CAD. Accessed December 21, 2017, https://www.deswik.com/product-detail/deswik-cad/.

Epstein R, Goic M, Weintraub A, Catalán J, Santibáñez P, Urrutia R, Cancino R, Gaete S, Aguayo A, Caro F (2012) Optimizing long-term production plans in underground and open-pit copper mines. *Oper. Res.* 60(1):4–17.

Espinoza D, Goycoolea M, Moreno E (2015) The precedence constrained knapsack problem: Separating maximally violated inequalities. *Discrete Appl. Math.* 194(1):65–80.

Espinoza D, Goycoolea M, Moreno E, Newman AM (2012) Minelib: A library of open pit mining problems. *Ann. Oper. Res.* 206(1): 1–22.

Fricke C (2006) Applications of integer programming in open pit mining. Unpublished doctoral dissertation, University of Melbourne, Melbourne, Australia.

Froyland GA, Menabde M (2011) System and method(s) of mine planning, design and processing. U.S. Patent 7,957,941.

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

Gershon ME (1983) Mine scheduling optimization with mixed integer programming. *Mining Engrg.* 35(4):351–354.

Goycoolea M, Moreno E, Rivera O (2013) Direct optimization of an open cut scheduling policy. *Proc. APCOM, Porto Alegre, Brazil* (Fundação Luiz Englert, Porto Alegre, Brazil), 424–432.

Goycoolea M, Moreno E, Rivera O (2015) Comparing new and traditional methodologies for production scheduling in open pit mining. *Proc. APCOM, Fairbanks, Alaska* (Society of Mining, Metallurgy, and Exploration, Englewood, CO), 352–359.

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

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

Jelvez E, Morales N, Nancel-Penard P, Peypouquet J, Reyes P (2016) Aggregation heuristic for the open-pit block scheduling problem. *Eur. J. Oper. Res.* 249(3):1169–1177.

Johnson TB (1968) Optimum open pit mine production scheduling. Unpublished doctoral dissertation, Operations Research Department, University of California, Berkeley, Berkeley.

Khalokakaie R, Dowd PA, Fowell RJ (2000) Lerchs–Grossmann algorithm with variable slope angles. *Mining Tech.* 109(2):77–85.

King B, Goycoolea M, Newman A (2017) Optimizing the open pit-to-underground mining transition. *Eur. J. Oper. Res.* 257(1):297–309.

Lambert WB, Newman AM (2014) Tailored Lagrangian relaxation for the open pit block sequencing problem. *Ann. Oper. Res.* 222(1): 419–438.

Lambert WM, Brickey A, Newman AM, Eurek K (2014) Open pit block sequencing formulations: a tutorial. *Interfaces* 44(2):127–142.

Lane KF (1964) Choosing the optimum cutoff grade. *Colorado School Mines Quart.* 59(4):811–829.

Lerchs H, Grossmann IF (1965) Optimum design of open-pit mines. *Trans. Canadian Mining Inst.* 58(1):47–54.

Liu SQ, Kozan E (2016) New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems. *Expert Systems Appl.* 43(1):59–65.

Maptek Vulcan (2017) Maptek Vulcan. Accessed December 21, 2017, http://www.maptek.com/products/vulcan/.

Meagher C, Dimitrakopoulos R, Avis D (2014) Optimized open pit mine design, pushbacks and the gap problem—A review. *J. Mining Sci.* 50(3):508–526.

Moreno E, Rezakhah M, Newman A, Ferreira F (2017) Linear models for stockpiling in open-pit mine production scheduling problems. *Eur. J. Oper. Res.* 260(1):212–221.

Muñoz G, Espinoza D, Goycoolea M, Moreno E, Queyranne M, Rivera O (2018) A study of the Bienstock-Zuckerberg algorithm, Applications in mining and resource constrained project scheduling. *Comput. Optim. Appl.* 69(2):501–534.

Nemhauser GL, Wolsey LA (1989) Integer programming. Nemhauser GL, Rinnooy Kan AHG, Todd MJ, eds. *Optimization*, Handbook in Operations Research and Management Science, vol. 1 (North-Holland, Amsterdam), 447–527.

Ramazan S, Dimitrakopoulos R (2013) Production scheduling with uncertain supply: a new solution to the open pit mining problem. *Optim. Engrg.* 14(2):361–380.

Ramazan S, Dagdelen K, Johnson TB (2005) Fundamental tree algorithm in optimising production scheduling for open pit mine design. *Mining Tech. (Trans. Inst. Mining Metallurgy A)* 114(1):45–54.

Rimélé A (2016) Méthode heuristique d'optimisation stochastique de la planification minière et positionnement des résidus miniers dans la fosse. Unpublished doctoral dissertation, École Polytechnique de Montréal, Montréal.

Samavati M, Essam D, Nehring M, Sarker R (2018) A new methodology for the open-pit mine production scheduling problem. *Omega* 81(December):169–182.

Smith ML, Wicks MJ (2013) Medium-term production scheduling of the Lumwana mining complex using MIP with a rolling horizon. *Interfaces* 44(2):176–194.

Tolwinski B (1998) Scheduling production for open pit mines. *Proc. 28th Internat. Sympos. Appl. Comput. Math. (APCOM) Mineral Indust.* (Institution of Mining and Metallurgy, London), 651–662.

Vossen TWM, Wood KRK, Newman AM (2016) Hierarchical benders decomposition for open-pit mine block sequencing. *Oper. Res.* 64(4):771–793.

Zhu G, Bard J, Yu G (2006) A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS J. Comput.* 18(3):377–390.

**Orlando Rivera Letelier** is a PhD student in Industrial Engineering and Operations Research at Universidad Adolfo Ibáñez in Chile and a PhD student in Applied Math at Université de Bordeaux, France. His research focuses on computational methods and applications of mixed-integer programming and large-scale optimization.

**Daniel Espinoza** is a senior researcher at Google. His areas of interest include mixed-integer optimization and stochastic programming, with applications in energy and natural resource management. He received his PhD in Industrial Engineering from the Georgia Institute of Technology, Atlanta, Georgia in 2006.

**Marcos Goycoolea** is professor of operations research in the School of Business at Universidad Adolfo Ibáñez, in Santiago, Chile. His primary research is in mixed-integer optimization, with applications in natural resource management.

**Eduardo Moreno** is vice-dean at the Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Santiago, Chile. His research focus is on combinatorial optimization, algorithms, and computational methods, with applications in network design, telecommunications, transportation, and mine planning.

**Gonzalo Muñoz** is an assistant professor at Universidad de O'Higgins, Chile. He was an IVADO Post-Doctoral Fellow at Polytechnique Montréal, Canada. His research focus is on computational methods for nonconvex optimization, with applications to the mining and energy industries.