Column Generation for Mining Cut Definition with Geometallurgical Interactions

Gonzalo Nelis, Frédéric Meunier, Nelson Morales

COLUMN GENERATION FOR MINING CUT DEFINITION WITH GEOMETALLURGICAL INTERACTIONS

GONZALO NELIS, FRÉDÉRIC MEUNIER, AND NELSON MORALES

Abstract. In this work we propose a novel approach to solve the mining cut definition problem in open pit mines with geometallurgical interactions. Mining cut definition deals with the aggregation of blocks into clusters which are extracted and processed as a single unit. The aggregation must fulfill operational considerations given by the loading equipment selectivity and should maximize the objectives given by the mine operation.

The proposed approach utilizes mixed integer programming and a model inspired by column generation that, contrarily to previous works, has its decision variables defined directly on the set of all feasible cuts. The advantage of this is that the model does not require linear approximations of the geometallurgical behavior of the cuts based on the blocks it contains, and therefore, can utilize any non-linear function.

We apply our methodology to a real-size case study to show that the model can be solved in reasonable time, but also that non-linear recovery functions influence the destination policy and expected profit, i.e., that following the traditional free selection policy (based on cut-off grade) is not the best strategy when realistic geometallurgical interactions are considered.

1. Introduction

Mine planning is the procedure of defining the best extraction strategy subject to physical, geological and operational constraints [Johnson, 1969]. This strategy includes selecting the material to extract, deciding the processing route for each unit (usually referred to as a *block*), and constructing a feasible extraction schedule in the planning horizon. Some common constraints are upper and lower limits on total material extracted in the mine and processed by the facilities, and geometrical restrictions for stability and operational reasons. Given the decision-centered nature of mine planning, operations research has been widely used in the extraction and development stages of mine operations [Newman et al., 2010].

Short-term mine planners deal with several problems related to the materialization of long-term schedules: ore-waste discrimination based on the blocks properties and the loading equipment size, the definition of block aggregates to organize the extraction in each period, and the optimal scheduling of these units to maximize profit or metal recovered in the short-term. A review of techniques to address these and other issues on short-term mine planning can be found in [Blom et al., 2017].

A common assumption for applying traditional mine planning techniques is the linearity and additivity of the block properties. Linearity is the assumption that the response variables in the processing facilities can be modeled as a linear combination of the blocks attributes. Additivity is the assumption that the properties of a blend of blocks can be calculated by the sum or average over the individual properties of each block. Under these assumptions, the response obtained by processing a single block is not affected by other blocks extracted in

the same period. Therefore, the objective function and constraints are linear expressions of the individual block values calculated beforehand.

Geometallurgy is the field that studies the interaction between rock properties and mining and processing outcomes [Coward et al., 2009]. A key area of the field is related to characterize response variables for a given set of rock attributes. Indeed, research in this area has proven that the assumptions of linearity and additivity do not hold true for some critical block attributes, such as metal recovery [Van Tonder et al., 2010, Nwaila et al., 2020] and grindability [Yan and Eaton, 1994, Tavares and Kallemback, 2013]. A better assessment of the plan feasibility, therefore, should incorporate the interaction given by the block blends. Accounting for this behavior in the traditional mine planning framework, unfortunately, results in a nonlinear program, rendering the usual models extremely hard to solve using exact techniques.

The issue of blending properties is specially relevant in short-term mine planning. Within this planning horizon, engineers face the problem of *selectivity*: loading equipment cannot extract blocks individually given its physical limitations. Therefore, the planners define *clusters* or *mining cuts* (also referred to as *polygons* or *mining blocks*) which are the combination of adjacent blocks which are extracted together and processed in the same facility. This is often a manual procedure of trial and error, where the planner must define a destination policy and extraction sequence that maximize profit or recovered metal from the cuts.

Clusters must also be operationally compatible with the loading equipment. The definition of what makes a cluster 'operationally feasible' is not clear, and we are not sure such a mathematical rigorous definition is possible. It depends on the operator expertise and the critical eye of the mine planner. In the literature, operational feasibility is defined with different metrics (mining width, minimum size, basic shapes), and additional steps to fix problematic locations are common.

Without a formal definition of a feasible cluster, we propose addressing the mining cut definition problem by brute-force enumeration of potential candidates. Instead of imposing an operational constraint in the model to group blocks, we generate a large set of feasible clusters as an input (and not output) of an optimization model. The advantages of this approach are directly related to both issues mentioned previously. Operational feasibility is achieved by construction, i.e., the clusters shapes used as an input already fulfill operational requirements. The problems of nonlinearity and nonadditivity on the mixing are also addressed: the cluster properties are calculated before the optimization process, so any function of the blocks properties is possible. Once this large set of feasible clusters is defined, the modeling of the problem as an integer program is straightforward. We propose then an efficient and simple column generation approach to solve the resulting integer program, which has a huge number of variables (much larger than usual integer programs deciding how to group blocks may have).

This forms a complete methodology to solve the mining cut definition problem in short-term mine planning. We emphasize that it requires no assumptions on the functions used to obtain cluster profits or properties, which contrasts with previous works that are either (i) linear, in which case have to approximate the characteristics of the clusters; or (ii) heuristic nonlinear, because the techniques used or the modeling cannot guarantee an optimal solution. Any geometallurgical model can be used within our methodology as long as a single value can be assigned for a cluster and facility combination. This flexibility shows potential to include complex interactions not yet studied in the short-term mine planning literature. In terms of

efficiency, the overall algorithm finds nearly optimal solutions of the problem in computation time that is compatible with short-term applications. We illustrate this by applying our methodology to a real-scale case study with up-to 912 blocks, three processing facilities, and over 19 million clusters, which was solved in 23 minutes. We also show that using linear models may lead to wrong destination policies when geometallurgical interactions are introduced.

A second contribution is a simple method to generate a set of possible shapes that are relevant from a practical point of view. It also illustrates a possible way to generate the clusters used as input of the optimization model we propose to solve the mining cut definition problem. This method can address different operational rules based on minimum and maximum mining cut dimensions depending on the loading equipment selectivity and the planners criteria. It has been used in the experiments mentioned above and the resulting clusters are compatible with our selectivity considerations.

The rest of the paper is organized as follows. A review of the literature on short-term mine planning and geometallurgy is presented in Section 2. Section 3 formalizes the mining cut definition problem addressed in the paper. The method we propose to solve the mining cut definition problem is described in Section 4. Section 5 proposes a method for generating practically relevant clusters. Section 6 shows the results in a real case study, and final conclusions are discussed in Section 7.

2. Literature Review

2.1. Short-term scheduling and selectivity.

A usual approach to incorporate selectivity into short-term mine planning is the definition of boundaries between blocks according to their processing destination. Such problems are referred to as *dig-limit* definition or optimization, and are based on a metric of operational space defined as the number of blocks that need to be extracted together and sent to the same destination. The goal is to make such a destination definition while maximizing a profit function and fulfilling operational requirements.

Isaaks et al. [2014] use a mining width as the base unit to differentiate between destinations. A heuristic approach is used to define the best dig-limit between materials while minimizing a loss-function related to the equipment selectivity. Sari and Kumral [2017] propose a mixed-integer program to solve the dig-limit optimization problem based on valid frames: each block must belong to a pre-defined valid frame, and all the blocks belonging to the same frame must be sent to the same destination.

Ruiseco et al. [2016] also propose a heuristic approach to the dig-limit problem. The authors penalize sectors that do not comply with operational constraints. The penalization function is nonlinear, so they rely on a genetic algorithm to solve the model. This tool was later used to evaluate the relationship between selectivity, equipment size and dig-limits definitions [Ruiseco and Kumral, 2017]. Williams et al. [2021] propose a neural network to evaluate the dig-limits definition made by the genetic algorithm in order to improve the efficency of this approach.

Vasylchuk and Deutsch [2019] tackle this problem with an iterative heuristic based on an initial classification using a fixed grid to maximize the expected profit, and additional steps to solve problematic locations related to operational restrictions.

A common downside of the dig-limit definition problem is the absence of a mining cut definition. While a boundary between materials is well-defined, additional steps are required

to organize the extraction of the blocks and fulfill capacity constraints. Moreover, local mixing and nonlinear properties of blocks are usually ignored in the literature.

Other works have dealt with the mining cut definition problem in short-term mine planning. Tabesh and Askari-Nasab [2011, 2013] propose a clustering procedure with shape control based on hierarchical clustering. This technique aggregates blocks based on several similarity indices such as rock type, grade, destination, closeness, among others. The authors note that some problematic shapes can be generated, so a post-processing step is needed to obtain mineable clusters. This work was extended by Tabesh and Askari-Nasab [2019] to introduce geological uncertainty.

The algorithm generates a feasible mining cut definition, but scheduling constraints are introduced as a posterior step [Tabesh et al., 2014]. The clustering process is based solely on similarity indices and therefore geometallurgical interactions of the blocks attributes in the processing facility are ignored.

Nelis and Morales [2021] propose an optimization model based on *representative SMUs*. These SMUs are used as anchor points where precedence arcs are defined to obtain connected shapes. However, imposing the shape feasibility through precedence constraints could still produce problematic shapes. The model allows multiple processing facilities, but the cluster properties are assumed to be linear and therefore gerometallurgical interactions are ignored.

2.2. **Mine planning and geometallurgy.** Literature dealing with mine planning and geometallurgy is limited. While some works have dealt with geometallurgical variables, they tend to omit nonlinearity and nonadditivity interactions. A common approach is estimating some geometallurgical attributes at the block level, but assuming there is no complex interaction between blocks [Morales et al., 2019, Maleki et al., 2020]. Another common approach is considering a processing facility flexibility to change the operational mode to deal with different geometallurgical properties [Navarra et al., 2018, Gerald Van Den Boogaart et al., 2011]. But works that deal directly with the nonlinearity and nonadditivity of geometallurgical variables are scarce and mostly focused on the long-term mine planning horizon. We briefly discuss these works to present current approaches for complex geometallurgy variables.

Goodfellow and Dimitrakopoulos [2016] declare the importance and challenging nature of incorporating nonlinear and nonadditive geometallurgical interactions in the optimization process. The authors propose a metaheuristic approach to obtain a mine plan for a mining complex considering nonlinear recovery curves. Since the planning horizon is long-term, they do not deal with local mixing and operational space needed in the short term. Zhang and Dimitrakopoulos [2018] presents similar shortcomings, as they propose an iterative heuristic to incorporate nonlinear recovery in a mineral value chain optimization in the long-term mine planning.

An application of the heuristic used in Goodfellow and Dimitrakopoulos [2016] is presented in Kumar and Dimitrakopoulos [2019]. However, instead of dealing with a nonlinear recovery curve, they introduce nonadditive attributes related with grindability. To deal with the nonadditive nature of these attributes, the authors impose blending restrictions over blocks based on "soft" and "hard" categories according to their grindability. The use of categories simplifies the blend properties and allows the authors to use linear blending constraints in the model.

Del Castillo and Dimitrakopoulos [2016] also deal with mining complexes and geometallurgy, but their focus is on the destination decision for a given mining sequence. They proposed

a *coalition formation clustering* procedure to define processing destinations of groups of similar blocks. The definition is made by considering a cooperative group value in order to cluster blocks that yield a high value blended together. The coalition process does not account for spatial considerations and, as the authors note, works under the assumption that all blocks sent to a given processing facility are blended in a yearly time-span, which is an oversimplification. They declare that adapting this approach to deal with local blending phenomena could provide more realistic results.

As noted in this review, there are no works dealing with the mining cut definition problem (or its coarse version formed by the dig-limit definition problem) and geometallurgical attributes in short-term mine planning simultaneously. Works that incorporate geometallurgy tend to rely on heuristics to deal with the nonlinear or nonadditive behavior of such variables, or they ignore mixing interactions altogether. In operational aspects, the definition of 'feasible shape' varies between works, and there is not a clear consensus in the literature. Our approach to tackle these issues is described in the next section.

3. Problem formulation

The problem we address consists in partitioning the blocks in a bench in an open pit mine into *clusters*, i.e., each block must be assigned to one and only one cluster. Also, blocks belonging to the same cluster have to be sent to the same processing facility (or to a stockpile, a waste dump, etc.).

The selection of the clusters and their destination depend on the profit of processing a cluster at a given facility. We assume that extracting a cluster and processing its blocks in a facility has a profit that may depend in a nonlinear and nonadditive way on the blocks in the cluster and on the facility.

Each facility has a processing capacity which limits the number of clusters that can be processed. This capacity is imposed over a given attribute of the clusters (weight, throughput, etc.). As in the case of the profit, we assume this attribute can be evaluated at the cluster level and does not need to be linear on the block's attributes.

We assume there is a procedure to generate a feasible cluster set. We provide an example on how this set might be generated in Section 5, but any procedure that fits with the operational requirements on the mine operation is allowed.

Formally, we are given a set B of blocks, a set $\mathcal{F}$ of *facilities*, a *mine capacity* $K \in \mathbb{R}_+$, and a *facility capacity* $Q_f \in \mathbb{R}_+$ for each $f \in \mathcal{F}$. The collection of all possible *clusters* is denoted as $\mathcal{C} \subseteq 2^B$. We are also given a weight function $w\colon \mathcal{C} \to \mathbb{R}_+$ that models the utilization of the mining capacity and a function $q\colon \mathcal{C} \times \mathcal{F} \to \mathbb{R}_+$ such that $q(c, f)$ is the utilization of capacity Q_f by cluster c. The *profit* of processing cluster c at facility f is given by a known function $v\colon \mathcal{C} \times \mathcal{F} \to \mathbb{R}$.

The problem consists in selecting a subset $\mathcal{S}$ of $\mathcal{C}$ forming a partition of B, and a map $\phi\colon \mathcal{S} \to \mathcal{F}$, assigning the selected clusters to the facilities, which maximize:

$$\sum_{c \in \mathcal{S}} v(c, \phi(c))$$

and which satisfy the following constraints:

$$\sum_{c \in \mathcal{C}} w(c) \leqslant K \qquad \text{and} \qquad \sum_{c \in \mathcal{C} \cap \phi^{-1}(f)} q(c, f) \leqslant Q_f, \, \forall f \in \mathcal{F}\,. \tag{1}$$

We call this problem the *mining cut definition problem*. If the constraints of Eq. (1) are discarded, solving this model provides a solution for the dig-limit definition problem.

The following mathematical program models the mining cut definition problem. We denote by $\mathcal{C}_b$ the set of all clusters $c \in \mathcal{C}$ such that $b \in c$. Binary variable $x_{c,f}$ equals 1 if cluster c is sent to facility f, and 0 otherwise.

$$
\begin{aligned}
\text{(MCDP)} \qquad \max \quad & \sum_{c\in\mathcal{C}} \sum_{f\in\mathcal{F}} v(c,f) x_{c,f} & \\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}_b} \sum_{f\in\mathcal{F}} x_{c,f} = 1 & \quad b \in B \\
& \sum_{c\in\mathcal{C}} \sum_{f\in F} w(c) x_{c,f} \leqslant K & \\
& \sum_{c\in\mathcal{C}} q(c,f) x_{c,f} \leqslant Q_f & \quad f \in \mathcal{F} \\
& x_{c,f} \in \{0,1\} & \quad c \in \mathcal{C},\ f \in \mathcal{F}\,.
\end{aligned}
$$

4. Method for solving the problem

We propose to address Problem (MCDP) with an approach based on column generation. Column generation is a technique used to solve a continuous linear problem with a large number of variables. The variables, or their indices, are usually called *columns* in this context (actually referring to the columns of the constraint matrix). The problem is solved iteratively, first restricted to a small subset of variables. This restricted version is the *master problem*. In each iteration, an auxiliary problem—the *pricing subproblem*—is solved to find new columns to add to the master problem. The pricing subproblem uses the reduced costs of the columns to make its decision. A detailed explanation of this method can be found in Lübbecke and Desrosiers [2005].

We use a special version of column generation, called *sifting* or *sprint* and introduced by Forrest [1989] to solve the linear relaxation of Problem (MCDP). While the pricing subproblem usually generates columns on-the-fly, the sifting method generates the whole set of columns beforehand and vectorization is usually used to calculate the reduced cost of all columns simultaneously [Bixby et al., 1992]. Finally, as in any column generation method, the reduced cost is used to select the best columns to add to the master problem. This technique has been applied successfully to very large-scale programs, specially to solve crew-pairing problems [Bixby et al., 1992, Chu et al., 1997, Anbil et al., 1998].

In order to solve the original integer problem, column generation has to be combined with other techniques from operations research, e.g., branch-and-bound. Here, we use another approach, which is quite useful when the optimal value of the problem is close to that of its linear relaxation. This approach uses an idea of Nemhauser and Wolsey [1988, Proposition 2.1, p. 389] for checking if any column not added during the algorithm may improve a given integer solution. If the number of such columns together with those generated when the linear relaxation is solved is not too big, then an off-the-shelf solver can be used. This strategy has been recently used by Cacchiani and Salazar-González [2017] and Parmentier and Meunier [2020] for airline scheduling and crew pairing problems.

The overall method can be summarized as follows. Generate the collection $\mathcal{C}$ of all possible clusters. Solve the linear relaxation of Problem (MCDP) with the sifting approach, where the columns are the pairs (c, f), with $c \in \mathcal{C}$ and $f \in \mathcal{F}$. Solve Problem (MCDP) restricted to the columns generated during the previous step with an off-the-shelf solver. Generate all columns that may improve the current integer solution. Solve Problem (MCDP) restricted to this extended set of columns, again with a solver.

Complementary details are given hereafter.

4.1. **Column generation approach.** We explain how we solve the linear relaxation of Problem (MCDP).

4.1.1. *Pricing subproblem.* The dual problem of the linear relaxation is the following optimization problem:

$$\min \quad \sum_{b \in B} \lambda_b + K\mu + \sum_{f \in F} Q_f \omega_f \tag{2}$$

$$\text{s.t.} \quad w(c)\mu + q(c,f)\omega_f + \sum_{b \in c} \lambda_b \geqslant v(c,f) \qquad c \in \mathcal{C},\ f \in \mathcal{F} \tag{3}$$

$$\mu,\ \omega_f \geqslant 0 \qquad f \in \mathcal{F}\,. \tag{4}$$

Given values of the dual variables λ_b, μ, and ω_f, the reduced cost of a column (c, f) is

$$p(c,f) = v(c,f) - w(c)\mu - q(c,f)\omega_f - \lambda(c)\,, \tag{5}$$

where $\lambda(c)$ is defined as $\sum_{b \in c} \lambda_b$. A column (c, f) may improve the current optimal value of the master problem only if $p(c, f) > 0$. (This fact is the classical result on which any column generation is based.) The pricing subproblem consists thus in finding such columns.

As the efficiency of this step is critical, we opt for a vectorized approach: we calculate the reduced cost (Eq. (5)) for all columns simultaneously. Then, the columns with the highest reduced costs are added to the master problem in each iteration.

4.1.2. *Column generation algorithm.* Algorithm 1 describes our column generation approach (the sifting or sprint version we mentioned above). Three auxiliary functions are used. Clusters(B) generates the cluster set for a given block model B. InitialFeasibleSet(B) generates a feasible solution for Problem (MCDP), since it is required for the initial iteration of the algorithm. In this work, we get an initial feasible solution by partitioning the bench with regular squares or rectangles. This initial solution is easy to generate and in our case studies provides a good starting point for the algorithm. Finally, MCDP($\mathcal{C}^{init}, \mathcal{F}$) builds problem (MCDP) using the initial feasible set $\mathcal{C}^{init}$ and the set of facilities $\mathcal{F}$.

We use $Cols$ as the set of columns represented by the pairs (c, f) obtained from the cluster set $\mathcal{C}$ and the set of facilities $\mathcal{F}$. The set $\widehat{\mathcal{C}}^{+}$ is used to store the best columns found in each iteration based on their reduced cost.

The main control variable of the algorithm is $N_{\max}$, the upper limit on the number of columns added to the master problem in each iteration. While adding a large number of columns could reduce the number of iterations needed to find the optimal solution, it could also increase the runtime of the master problem. For this reason, we test different strategies for the $N_{\max}$ parameter in Section 6.5.

Algorithm 1 Column Generation Algorithm

initialize $\mathcal{F}, v, K, Q_f, N_{\max}$
$\mathcal{C} \leftarrow$ Clusters(B)
$Cols \leftarrow \mathcal{C} \times \mathcal{F}$
$\mathcal{C}^{init} \leftarrow$ InitialFeasibleSet(B)
$m \leftarrow$ MCDP$(\mathcal{C}^{init}, \mathcal{F})$
while $Cols \neq \varnothing$ **do**
    solve linear relaxation of m
    $\widehat{\mathcal{C}}^+ \leftarrow \varnothing$
    for $f \in \mathcal{F}$ **do**
        for $c \in \mathcal{C}$ **do**
            $p(c, f) \leftarrow v(c, f) - w(c)\mu - q(c, f)\omega_f) - \lambda(c)$
        end for
        add $\{(c, f) \in Cols\}$ to $\widehat{\mathcal{C}}^+$ if $p(c, f) \geqslant 0$
        if $|\widehat{\mathcal{C}}^+| \geqslant N_{\max}$ **then**
            sort $\widehat{\mathcal{C}}^+$ by $p(c, f)$
            $\widehat{\mathcal{C}}^+ \leftarrow$ first $N_{\max}$ elements of $\widehat{\mathcal{C}}^+$
            break for
        end if
    end for
    if $\widehat{\mathcal{C}}^+ = \varnothing$ **then**
        No column with positive reduced cost found
        break while
    end if
    $\widehat{\mathcal{C}} \leftarrow \widehat{\mathcal{C}} \cup \widehat{\mathcal{C}}^+$
    remove $(c, f) \in \widehat{\mathcal{C}}^+$ from $Cols$
    update m with columns in $\widehat{\mathcal{C}}^+$
end while
return $m.solution$

4.2. **Obtaining an integer solution.** Column generation only permits solving to optimality the linear relaxation of (MCDP). An additional step is thus required to find an optimal—or at least good—integer solution. A simple approach to achieve this is performing classical branch-and-bound; yet we follow the other method mentioned above.

A feasible solution of the original integer problem can be found by imposing the variables to be binary in the master problem with the columns added during the column generation algorithm (referred to as *Restricted IP*). Since we start the column generation with an initial feasible solution of the mining cut definition problem, this method always provide a feasible integer solution. This integer solution may not be optimal for the complete master problem (MCDP). However, it provides a lower bound for the optimal value of (MCDP).

We perform one last iteration through the cluster set. We look for columns with reduced cost larger than the gap between our lower bound (restricted IP solution) and the linear relaxation solution of problem (MCDP). These columns are added to the master problem and it is finally solved to optimality (referred to as *Final IP*). Nemhauser and Wolsey noted

that any optimal solution of Final IP is an optimal solution of Problem (MCDP), and the method is thus exact.

Algorithm 2 shows these steps, and it is performed after solving the linear relaxation through Algorithm 1.

Algorithm 2 Integer Solution

Require: $m.solution$: Optimal solution from Algorithm 1
Require: $Cols$: Unused columns from Algorithm 1
$UB \leftarrow$ Optimal value of the linear relaxation of m
impose: $m.variables \in \{0, 1\}$
solve m (Restricted IP)
$LB \leftarrow$ Optimal value of m
$\widehat{\mathcal{C}}^+ \leftarrow \varnothing$
for $f \in \mathcal{F}$ **do**
    for $c \in \mathcal{C}$ **do**
        $p(c, f) \leftarrow v(c, f) - w(c)\mu - q(c, f)\omega_f - \lambda(c)$
    end for
    add $\{(c, f) \in Cols\}$ to $\widehat{\mathcal{C}}^+$ if $p(c, f) \geqslant LB - UB$
end for
update m with columns in $\widehat{\mathcal{C}}^+$
solve m (Final IP)
return $m.solution$

The overall method is thus Algorithm 1 followed by Algorithm 2.

5. Generating a collection of clusters

In this section, we propose a procedure to generate a possible collection $\mathcal{C}$, which is relevant from a practical point of view. We remind the reader that this procedure is used as input of the mining cut definition problem. The method of Section 4 is in no way restricted to this procedure of generating clusters; in principle, any procedure of generating them is a valid input.

The main idea for the generation of clusters is to consider a set of rectangles of predefined dimensions and then considering all their possible locations. To enrich the set, we also consider extensions of the base rectangles which are obtained by others, smaller rectangles that are adjacent to each side.

An example of this procedure is shown in Figure 1. The design parameters are highlighted in the figure, and must be defined by the mining engineer. The minimum size of the base and sides rectangles are given by the equipment selectivity to ensure there is enough operational space to extract the material. Smaller minimum sizes lead to less regular mining shapes, which tends to be more suitable to smaller loading equipment. A maximum cluster size is also given by the engineers depending on the planning horizon and the mixing span in the processing facility.

The parameters for the cluster generation algorithm are:

- b_x: Base rectangle size along the X coordinate. Limits are $[b_x^{\min}, b_x^{\max}]$
- b_y: Base rectangle size along the Y coordinate. Limits are $[b_y^{\min}, b_y^{\max}]$

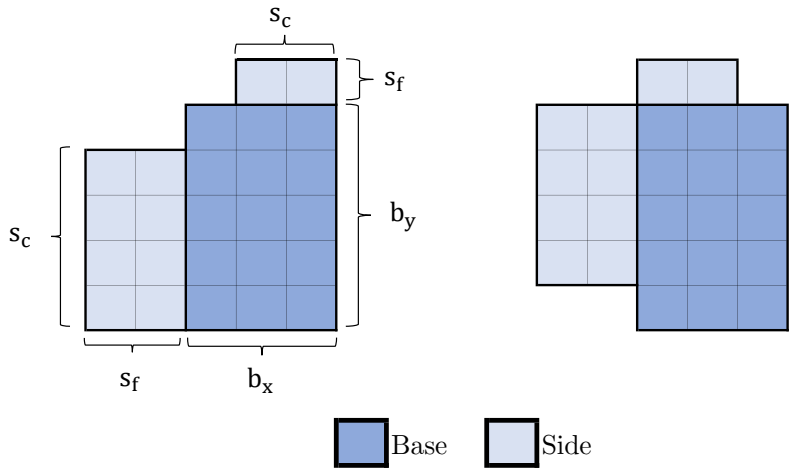

FIGURE 1. Shape generation example. Left side shows the control parameters and right side shows a possible permutation with the same side shapes but different location

- s_c: Side rectangle size along the axis parallel to the base side. Limits are $[s_c^{\min}, s_c^{\max}]$
- s_f: Side rectangle size along the axis perpendicular to the base side. Limits are $[s_f^{\min}, s_f^{\max}]$

After generating the set of shapes, they are applied through the bench to define the cluster set. Therefore, in our column generation algorithm a cluster is defined by a shape applied to a specific location of the bench. The total number of clusters is roughly the number of shapes times the number of blocks (some clusters are not created since they exceed the borders of the bench). Using this procedure, we are able to produce realistic cluster definitions for our case study.

All the properties needed for each cluster (value, weight, throughput, hardness, etc.) are calculated before the optimization process, and therefore, any complex relationship can be incorporated. The cluster set is then used as an input in the method described in Section 4.

6. EXPERIMENTS

6.1. **Case Study.** The case study corresponds to a real copper mine. Drillhole data is used to simulate short-term blasthole information. Four benches with different sizes (216, 432, 720, and 912 blocks respectively) are selected to test the algorithms. A large set of 32,764 shapes is generated using the approach described in Section 5, with a minimum operational size of 2×3 blocks and maximum size of 40 blocks.

Using the largest set of shapes would be recommended to obtain the best feasible schedule. However, we select smaller subsets to show the algorithm behavior at different scales. The sizes of these subsamples are 8,000, 15,000, and 24,000 shapes, all drawn randomnly from the largest set of 32,764 shapes.

We denote each instance by "Number of blocks-Number of shapes." Therefore, case "432-24" corresponds to 432 blocks and 24,000 shapes. A comprehensive list of these sets can be found in Nelis [2021].

The number of processing facilities is three. The cluster tonnage is used as the weight parameter for the mining and processing capacities (i.e., $q(c, f) = w(c)$). Table 1 describes the economic parameters used in all cases. Facility 3 represents the waste dump, where non-profitable material is discarded.

Nonlinear recovery curves $R(g, f)$, depending on the copper content, are used for Facilities 1 and 2, while no material is recovered in Facility 3 (Figure 2).

To incorporate mixing effects on the profit valuation, metallurgical recovery is assigned based on the mean copper content for each cluster, $g(c)$. Eq. (6) shows the cluster profit calculation. Note that the same expression can be used to compute block profit ($v(b, f)$), replacing $w(c)$ by $w(b)$ (block weight) and $g(c)$ by $g(b)$ (block grade).

$$v(c, f) = w(c)\big((P - C_s(f))\, g(c) R(g(c), f) - C_p(f) - C_m \big) . \tag{6}$$

TABLE 1. Economic parameters

Parameter	Facility 1	Facility 2	Facility 3
Processing Cost ($C_p(f)$) (\$/t)	7.0	11.0	0
Selling Cost ($C_s(f)$) (\$/lb)	0.5	0.5	0
Facility Capacity (Q_f)[1](kt)			
216	49.6	49.6	179.9
432	101.2	101.2	349.9
720	170.1	170.1	538.2
912	226.8	226.8	738.7
Copper Price (P)(\$/lb)		2.0	
Mining Cost (C_m) (\$/t)		1.5	
Mining Capacity (K)[1](kt)			
216		174.9	
432		349.9	
720		538.2	
912		738.7	

[1] Capacities vary according to the block model size.

The algorithm has been implemented using Python 3.8.6 and the Numpy library, version 1.19.2. Optimization problems are solved using Gurobi 9.1.0 on an AMD Ryzen 5 3600 processor with 16 GB of RAM. MIP Gap parameter has been fixed at 0.01% for all integer problems. $N_{\max}$ parameter is also fixed at 1,000 columns for all instances. A detailed analysis of the effect of this parameter is shown in Section 6.5.

6.2. **Mining cut and destination policy.** Figures 3.b and 3.c show a mining cut definition and destination policy obtained by the proposed approach. As a reference, the free selection destination policy with capacity constraints is also shown (Figure 3.a), where selectivity restrictions are ignored (the loading equipment is unable to extract blocks individually).

The mining cut definition (Figure 3.b) contains 34 clusters selected from a pool of 7.29 million candidates and three possible destinations for each one of them. The resulting

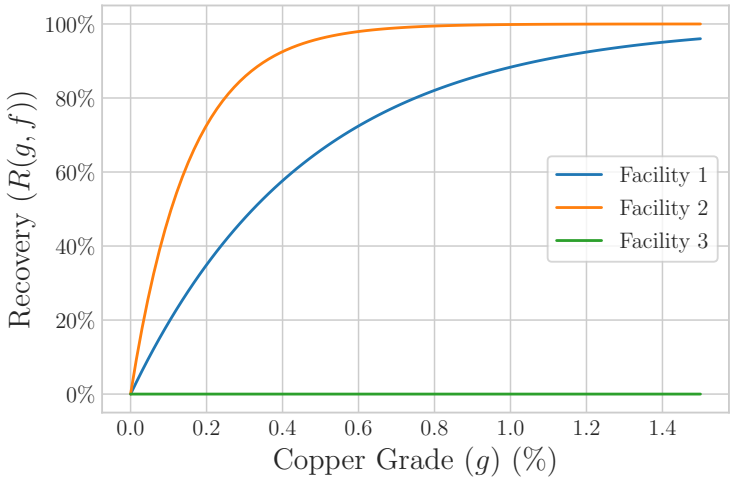

FIGURE 2. Recovery Curves

mining cut definition only contains clusters from the collection of feasible shapes described in Section 5. Other mining operations might define different feasible shapes and the main method presented in Section 4 would still provide the optimal solution.

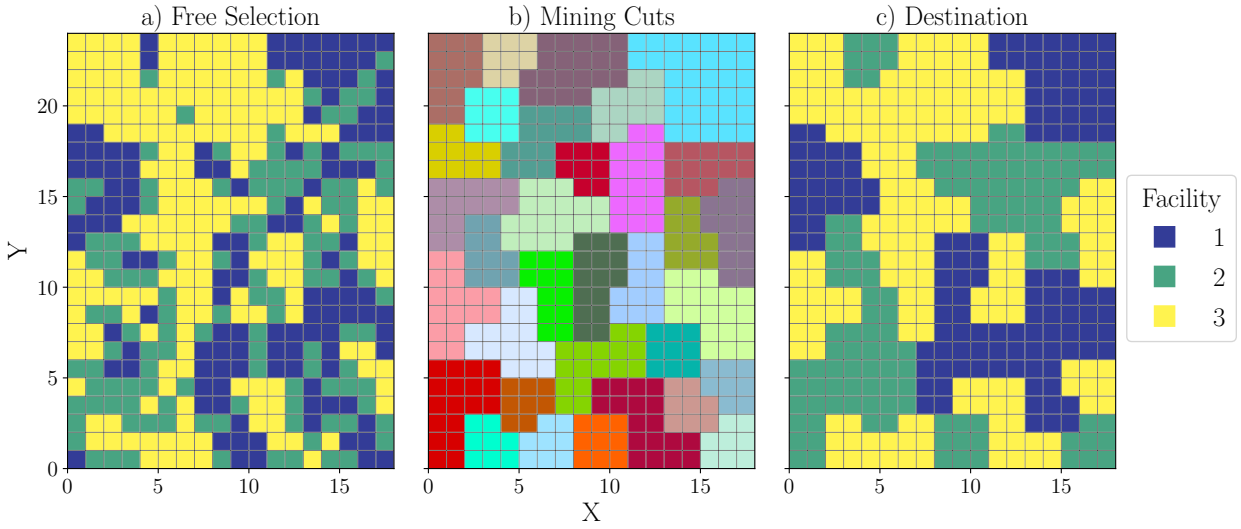

FIGURE 3. Mining cut definition for 432-33

Table 2 shows the profit achieved by our algorithm for all the block models and set of shapes tested. Columns shows the profit achieved for the different sets of shapes ("8" refers to the sample with 8,000 shapes, etc. and "33" refers to the full sample). As a reference value, the free selection policy is reported as well. This value was computed as the sum of the individual profit for each block, $v(b, f)$, for each block model. Since this policy ignores selectivity restrictions, its profit is often considered as an upper bound for the mining cut definition problem.

TABLE 2. Profit

		Set of Shapes			
Block Model	Free Selection (k$)	"8" (k$)	"15" (k$)	"24" (k$)	"33" (k$)
216	5,343.9	5,014.8	5,042.0	5,078.3	5,102.9
432	12,079.4	11,675.0	11,707.4	11,748.8	11,759.0
720	19,397.4	18,838.3	18,876.2	18,917.6	18,939.7
912	22,749.9	22,239.8	22,285.6	22,331.8	22,359.1

In this case, the profit depends on the number of shapes used by our approach. The larger the set, the higher the total profit because a richer set of shapes extends the possibilities of the algorithm. In all cases, a lower profit is achieved when selectivity is introduced. The difference compared to the free selection policy, however, is small. For the largest set of shapes, it ranges between 4.51% lower for the smaller block model, to 1.72% lower for the largest, which is in line with other dig-limits or mining cut definition algorithms.

While a decrease in profit compared to the free selection policy is expected when selectivity is considered, geometallurgical interactions among blocks play a critical role in the short-term. Next section shows a case where the nonlinear recovery curve introduces unexpected behaviors in terms of profit and destination policy.

6.3. **Nonlinear recovery effect.** We show an example on the effect of mixing and nonlinear recovery for the dig-limits definition problem. We remind that Problem (MCDP) provides a solution for the dig-limit definition problem if capacity constraints are ignored.

Recovery is a well-known nonadditive attribute in the block valuation. In the literature, the assumption of independency between adjacent block recoveries is common [Morales et al., 2019, Maleki et al., 2020], and any mixing interaction is usually ignored. For this case study, we incorporate geometallurgical interactions assuming a perfect mixing of the blocks inside each cluster, with a nonlinear recovery curve.

Figure 4 shows the destination policy obtained by solving Problem (MCDP) for instance 720-33 with and without mixing interactions.

Figure 4.a depicts the free selection policy. This definition, however, does not fulfill selectivity restrictions. A typical dig-limit algorithm would ignore mixing interactions of blocks extracted together. For our optimization problem, this is equivalent to define the cluster profit as the sum of the individual blocks profit, i.e., $v(c, f) = \sum_{b \in \mathcal{C}} v(b, f)$. Under this assumption, and ignoring capacity constraints, our method results in the solution shown in Figure 4.b.

The introduction of selectivity considerations (given by the clusters shapes) regularizes the free selection policy. In broad terms, however, both the free selection and the dig-limit definition without mixing interactions follow the same structures. However, if we introduce mixing effects on the cluster profit, the dig-limit definition changes. Figure 4.c depicts the results of our method and using Eq. (6) to introduce mixing effects into the profit. Facility 1 becomes much more prevalent than Facility 2, while the waste remains relatively unchanged.

Table 3 shows the profit for the destination policies. For this case, the free selection policy achieves the highest profit as expected if we ignore mixing interactions. But this result changes if we introduce mixing interactions: the profit obtained by the clusters is higher than the free selection policy profit. This is a counter-intuitive result due to the recovery function

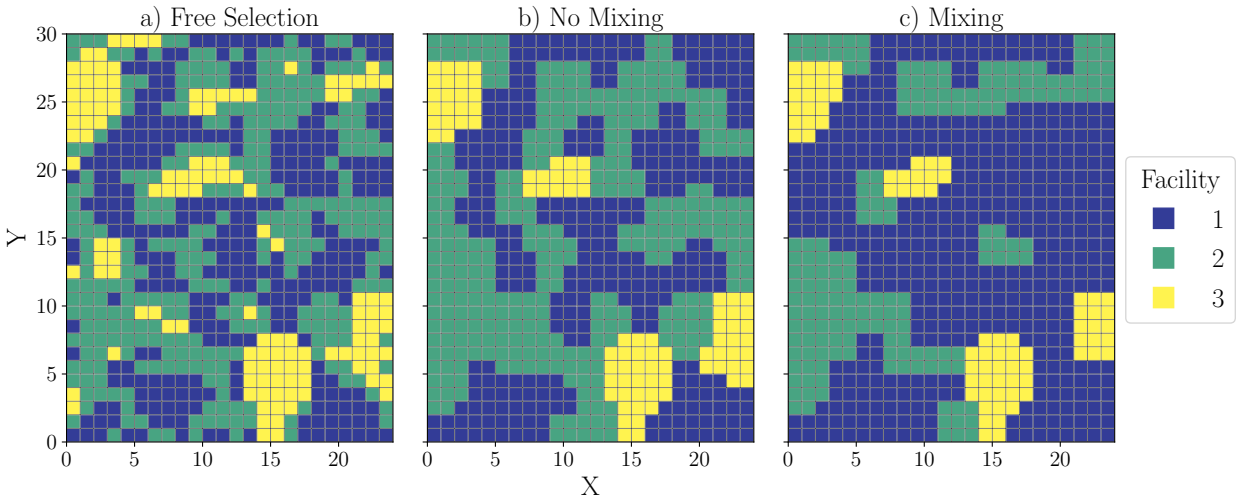

FIGURE 4. Dig-limit definition comparison - 720-33

TABLE 3. Profit and destination policies with and without mixing interactions.

Destination Policy	Profit without mixing (k$)	Profit with Mixing (k$)
Free Selection (Figure 4.a)	20,815.5	-
Clusters without mixing (Figure 4.b)	20,570.9	20,817.7
Clusters with mixing (Figure 4.c)	20,433.4	20,976.8

shape (Figure 2), which favors mixing blocks of low and high grade to yield a higher metal recovered content compared to processing these blocks independently. Profit with mixing is not calculated for the free selection policy since it is unclear how blocks are mixed locally without a mining cut definition.

While the value gap is rather small, obtaining a dig-limit definition with higher estimated profit than the free selection policy is a notable result. This is a clear example of how incorporating mixing effects and nonlinear recovery functions produces counter-intuitive results. More importantly, it shows that basing the dig-limit definition on the free selection policy is not the best methodology when nonadditive mixing effects are expected. The recovery curve used in this work is rather simple, and we expect seeing more sizable differences with more complex geometallurgical interactions.

6.4. **Algorithmic performance.** Table 4 summarizes our column generation algorithmic performance solving the linear relaxation of Problem (MCDP). We compare runtimes and objective function values against solving the full model with Gurobi, if possible. Reported fields are described next:

- Clusters:
  - **Time:** Time to generate the cluster set (given by function CLUSTERS(B)).
  - **N:** Number of clusters generated.
- Column Generation:

  - **Overhead:** Time to calculate the membership matrix, and value and weight vectors described in Section 4.1. It also includes the time to build the optimization model with the initial cluster set.
  - # **Cols.:** Number of columns added to the master problem.
  - # **Iters.:** Iterations of the algorithm.
  - **Time:** Time to obtain the linear relaxation solution.
  - **Value:** Optimal value of the objective function.
- Full Model:
  - **Overhead:** Time needed to build the optimization model.
  - **Time:** Time to obtain the linear relaxation solution.
  - **Value:** Optimal value of the objective function.

TABLE 4. Numerical results for the linear relaxation

Id	Clusters		Column Generation					Full Model		
	N	Time (s)	Overhead (s)	# Cols.	# Iters.	Time (s)	Value (k$)	Overhead (s)	Time (s)	Value (k$)
216-8	600,017	19	3	9,098	13	2	5,014.8	55	248	5,014.8
216-15	1,120,597	35	5	14,437	19	5	5,042.0	107	351	5,042.0
216-24	1,790,731	57	8	19,190	24	6	5,078.3	177	571	5,078.3
216-33	2,447,373	78	11	18,313	23	6	5,102.9	235	630	5,102.9
432-8	1,785,611	58	8	31,100	30	29	11,675.0	182	791	11,675.0
432-15	3,341,299	104	16	42,959	40	42	11,707.4	349	1,980	11,707.4
432-24	5,342,551	173	26	54,374	50	53	11,748.8	618	-	-
432-33	7,297,725	236	42	72,170	59	66	11,759.0	1,105	-	-
720-8	3,547,277	113	17	49,995	55	119	18,383.3	366	8,137	23,257
720-15	6,642,073	218	34	67,919	71	172	18,876.2	864	-	-
720-24	10,622,443	338	83	88,358	90	204	18,917.6	-	-	-
720-33	14,507,157	464	136	101,899	103	211	18,939.7	-		-
912-8	4,806,006	153	23	59,288	61	155	22,239.8	539	-	-
912-15	9,002,674	290	65	83,389	85	229	22,285.6	1,525	-	-
912-24	14,399,046	491	144	102,912	104	278	22,331.8	-	-	-
912-33	19,662,940	780	375	119,128	121	295	22,359.1	-	-	-

Table 5 shows the integer results for both approaches. As seen in Section 4.2, the column generation algorithm requires solving two integer problems to get the optimal solution: the reduced problem to get a feasible lower bound for the optimal value, and the final problem that adds several columns (reported under the 'Columns' field) to find the optimal solution of Problem (MCDP).

The optimization software does not allow starting the branch-and-bound algorithm using a pre-calculated linear relaxation solution. For this reason, the master problem is solved again with the subset of columns added during the Column Generation process before starting the branch-and-bound algorithm. Runtimes reported in Table 5 include this re-solving time in the Restricted IP and Final IP fields.

In contrast, solving the full model only requires solving a single integer problem dealing with all possible columns.

TABLE 5. Comparison of solutions obtained by the column generation approach and full model

Id	Restricted IP		Final IP			Full Model	
	Time[1] (s)	Value (k$)	# ColS.	Runtime[1] (s)	Value (k$)	Time (s)	Value (k$)
216-8	3	4,962.1	49,761	6	4,972.5	104	4,972.5
216-15	5	4,992.6	102,209	31	5,018.9	205	5,018.9
216-24	2	5,068.0	25,231	3	5,068.9	189	5,068.9
216-33	1	5,100.2	2,497	1	5,100.2	264	5,100.2
432-8	21	11,643.1	26,551	70	11,654.5	728	11,654.5
432-15	15	11,699.8	101,887	40	11,701.6	508[2]	11,699.7[2]
432-24	19	11,739.0	226,441	39	11,739.0	-	-
432-33	16	11,750.8	213,956	63	11,750.8	-	-
720-8	289	18,796.8	943,768	582	18,818.7	2,986[2]	18,672.9[2]
720-15	27	18,862.8	282,256	139	18,865.7	-	-
720-24	30	18,910.3	296,589	100	18,912.6	-	-
720-33	30	18,935.0	237,250	90	18,937.9	-	-
912-8	1,192	22,203.6	1,886,927	2,721	22,224.9	-	-
912-15	286	22,276.6	647,034	805	22,279.0	-	-
912-24	99	22,324.0	925,938	543	22,325.7	-	-
912-33	120	22,353.8	884,984	541	22,353.8	-	-

[1] Includes linear relaxation re-solving time.
[2] Last runtime and value reported before "out of memory" error.
"-" denotes instances where an "out of memory" error was raised before a feasible solution was found.

As a final comparison, Figure 5 shows the overall method runtime, from building the initial model to finding the optimal integer solution. Overhead times differ between algorithms, so they are included as well.

According to the results shown in Table 4, our method outperforms the full model approach in every instance. The full model runtimes are between 27 and 111 times longer than our approach. In fact, setting up the full optimization model (Overhead Time) takes more time than getting the optimal solution with column generation in all instances tested. The structure of Problem (MCDP), where the optimal solution contains just a small subset of all columns, favors the column generation approach as shown in these instances.

Moreover, our algorithm is not only faster but also memory-efficient. Dealing with the complete set of columns in the full model severely limits the instance size we are able to solve. In our tests, the largest instance the full model approach is able to solve consisted on 3.55 million clusters (10.64 million variables) and is solved in 8,137 seconds. In comparison, our column generation algorithm is able to solve an instance of 19.7 million columns (59

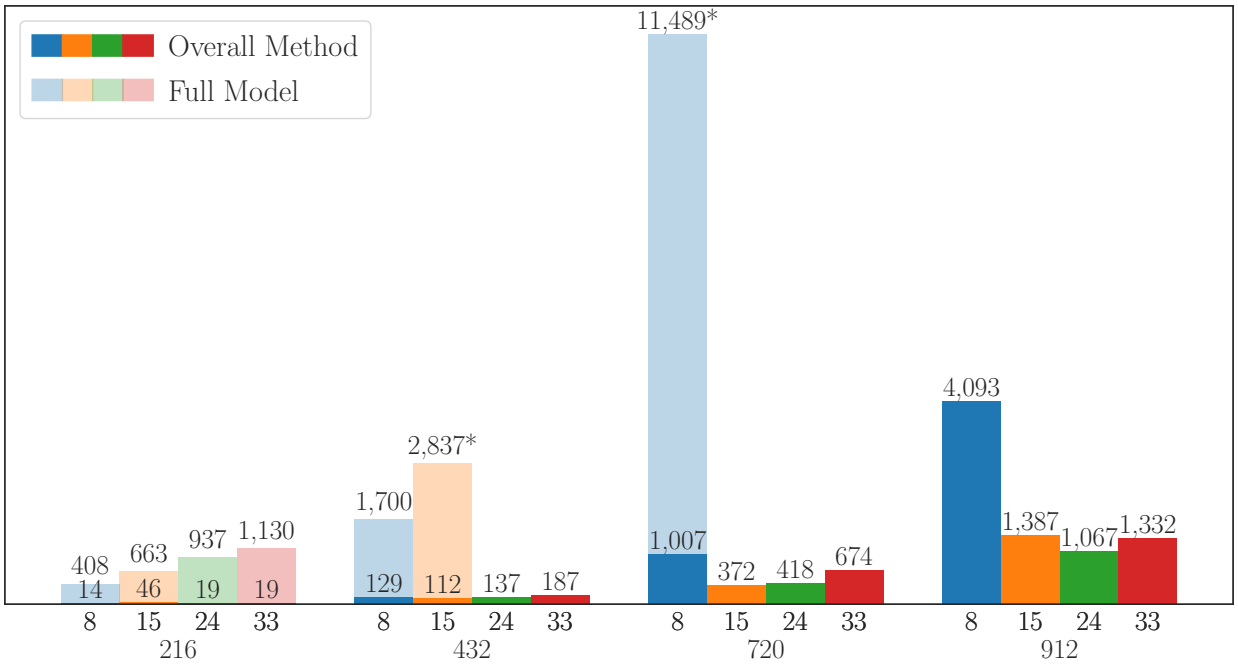

FIGURE 5. Total runtimes. Opaque bars represent the overall algorithm runtime, while transparent bars represent full model runtime. Runtime (in seconds) is displayed at the top of each bar. '*' denotes instances not solved to optimality. Absence of transparent bars denotes instances where an "out of memory" error was raised before a feasible solution was found.

million variables) in less than 5 minutes. On average, the column generation algorithm finds the optimal solution adding just 0.36% of the total columns to the master problem, which explains the large difference in performance for all instances.

In terms of objective function value, the differences between sets of shapes are small. This might be related to the profit structure used for this case study, where Facilities 1 and 2 are similar. Therefore, changes in destinations do not impact the objective function value in a significant way. More complex profit structures are expected to show larger differences.

In terms of the integer solution (Table 5), our method outperforms the full model approach in every instance as well. Full model runtimes are between 5.7 to 120 times larger than our method's. On average, however, the difference is not as wide as in the linear relaxation case. This is mostly because, in some instances, the lower bound provided by the Restricted IP is not strong enough to limit the number of columns added to find the optimal solution. For all instances, however, our algorithm was able to provide the optimal solution, which was not the case for the full model approach.

The difference in profit between the restricted and final IP solutions is also notably small, and ranges between 0% and 0.53% in every instance. The value provided by the restricted IP, therefore, might be enough for practical uses of this approach.

Interestingly, the smaller set of shapes poses the biggest challenge for the algorithm. With a restricted set of shapes, the runtimes for both integer problems increase compared to larger set of shapes. Limiting the number of shapes also limits the number of feasible cluster

combinations that deliver a feasible partition. A well populated set of shapes certainly increases the problem size, but simultaneously, presents more alternatives to get a feasible integer solution. For the current case study, results show that a larger set of shapes provides a meaningful performance advantage.

In terms of total runtime (Figure 5), our algorithm is between 13 and 59 times faster among instances solved to optimality. In practice, the short-term schedule is prepared on daily, weekly, or monthly basis. For all these instances, total runtimes are acceptable and could provide optimal solutions in real applications.

6.5. **Number of columns per iteration.** Finally, we discuss the influence of the control parameter $N_{\max}$ in our algorithm. An iteration in the column generation algorithm consists of three main tasks: finding columns with positive reduced cost, adding these columns to the master problem and then solving the master problem to get a new solution.

There is a direct relationship between $N_{\max}$ (the maximum number of columns to be added to the master problem at each iteration) and the algorithm runtime. This relationship is mainly driven by the solving time of the master problem. Adding a small set of columns keeps the problem size limited, and the warm start provided by the current solution is useful in each iteration. For larger values of $N_{\max}$, the problem size and runtimes grow rapidly, and the warm start is less useful. However, the number of iterations needed to reach the optimal solution also decreases with larger values of $N_{\max}$, which suggests the existence of a trade-off between the total number of iterations and each iteration runtime.

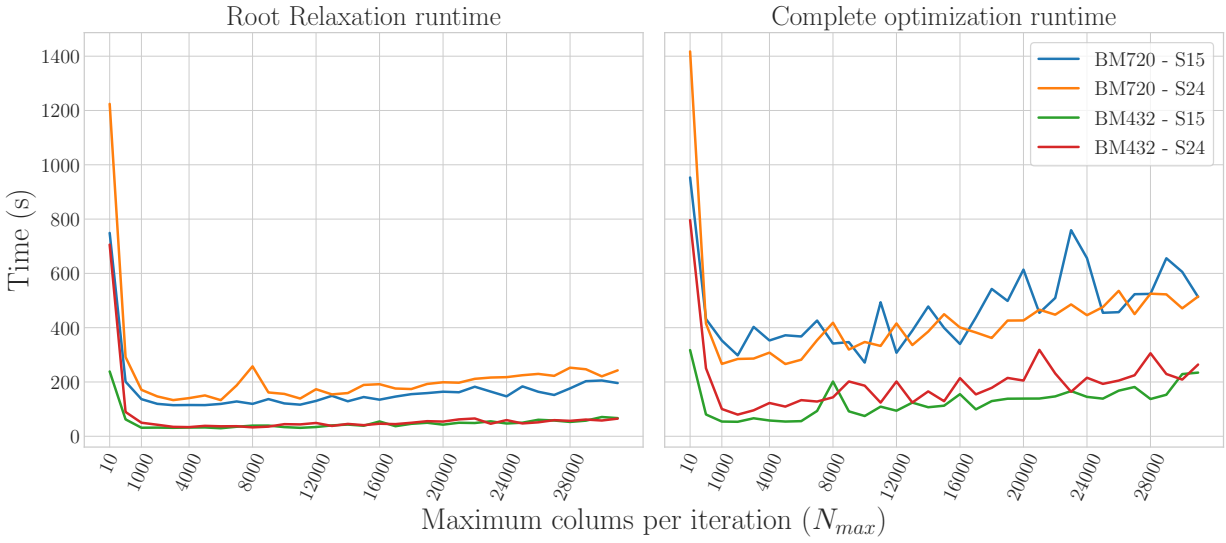

FIGURE 6. Algorithm runtime for different values of $N_{\max}$

Figure 6 shows the linear relaxation and total runtimes for four instances by different values of $N_{\max}$. For the linear relaxation runtimes, all instances show the worst performance for $N_{\max} = 10$. Having a faster solving time per iteration does not offset the large amount of iterations needed to reach the optimal solution. Runtimes decrease rapidly, though, and the best performance is around $N_{\max} = 1,000$. At higher values, a subtle trend towards higher

runtimes appears, indicating that increasing the problem size too fast is detrimental to the algorithm global performance.

The trend becomes much more evident when we analyze the total runtime (which includes the integer problems). Worst performance is still achieved by the lowest value of $N_{\max}$, and the best runtimes are still found around $N_{\max} = 1,000$. But the total runtime increases rapidly with higher values of $N_{\max}$ driven by the integer problems runtime. Higher values on $N_{\max}$ tends to reach the optimal solution with more columns added to the master problem. This makes both integer problems harder to solve due to the large number of variables. The number of columns added to solve the Final IP does not seem to decrease with larger values of $N_{\max}$. Therefore, a possible solution for this issue might be dropping unused columns from the master problem to reduce its size; see, e.g., Bixby et al. [1992]. Since we did not face this issue with lower values of $N_{\max}$, this step was not incorporated in the current implementation.

7. Conclusions

We present a novel approach to tackle the short-term scheduling problem. Our approach is able to obtain feasible mining cut definition for real case studies. Our resolution strategy based on column generation outperforms general solvers and allows solving real-size instances in reasonable runtimes for daily, weekly or monthly short-term horizons.

Also, the approach allows the incorporation of complex, nonlinear or nonadditive geometallurgical interaction in each cluster. We show how a simple interaction on the metallurgical recovery results in different dig-limits definitions compared to the traditional linear approach. Moreover, we show that following the free selection policy is not the best strategy to define operational mining cuts when nonlinear interactions are expected.

Several extensions of this approach are possible. The definition of operational shapes is performed by a simple heuristic, but other methodologies can be studied to incorporate different operational rules. Also, the proposed optimization model is currently limited to a single time period. Horizontal precedence constraints are needed to obtain multi-period schedules, and an efficient way to describe these arcs for a large set of clusters remains an open challenge.

Acknowledgments. This work was (partially) funded by the National Agency for Research and Development of Chile (ANID) through PIA Project AFB180004 and Scholarship Program Doctorado Nacional 2018 - 2118155.

References

R. Anbil, J. Forrest, and W. Pulleyblank. Column generation and the airline crew pairing problem. In *Documenta Mathematica J. DMV*, page 677–686, 1998.

R. Bixby, J. Gregory, I. Lustig, R. Marsten, and D. Shanno. Very large-scale linear programming: A case study in combining interior point and simplex methods. *Operations Research*, 40(5):85 – 897, 1992.

M.L. Blom, A.R. Pearce, and P.J. Stuckey. Short-term scheduling of an open-pit mine with multiple objectives. *Engineering Optimization*, 49(7):777–795, 2017.

V. Cacchiani and J. Salazar-González. Optimal solutions to a real-world integrated airline scheduling problem. *Transportation Science*, 51(1):250–268, 2017.

H. Chu, E. Gelman, and E. Johnson. Solving large scale crew scheduling problems. *European Journal of Operational Research*, 97(2):260–268, 1997.

S. Coward, J. Vann, S. Dunham, and M. Stewart. The primary-response framework for geometallurgical variables. In S. Dominy, editor, *Proceedings of the 7th International Mining Geology Conference*, pages 109 – 113, 2009.

M.F. Del Castillo and R. Dimitrakopoulos. A multivariate destination policy for geometallurgical variables in mineral value chains using coalition-formation clustering. *Resources Policy*, 50:322–332, 2016.

J. Forrest. Mathematical programming with a library of optimization subroutines. In *ORSA/TIMS Joint National Meeting, New York*, 1989.

K. Gerald Van Den Boogaart, C. Weibflog, and J. Gutzmer. The value of adaptive mineral processing based on spatially varying ore fabric parameters. In *Proceedings of IAMG 2011*, 2011.

R. Goodfellow and R. Dimitrakopoulos. Global optimization of open pit mining complexes with uncertainty. *Applied Soft Computing*, 40:292–304, 2016.

E. Isaaks, I. Treloar, and T. Elenbaas. Optimum dig lines for open pit grade control. In *Proceedings of the 9th International Mining Geology Conference*, pages 425 – 432. Australian Institute of Mining and Metallurgy, 2014.

T.B. Johnson. Optimum open-pit mine production scheduling. *A Decade of Digital Computing in the Mining Industry, Chapter 4*, page 539–562, 1969.

A. Kumar and R. Dimitrakopoulos. Application of simultaneous stochastic optimization with geometallurgical decisions at a copper-gold mining complex. *Mining Technology*, 128(2): 88–105, 2019.

M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

M. Maleki, E. Jélvez, X. Emery, and N. Morales. Stochastic open-pit mine production scheduling: A case study of an iron deposit. *Minerals*, 10(7):585, 2020.

N. Morales, S. Seguel, A. Cáceres, E. Jélvez, and M. Alarcón. Incorporation of geometallurgical attributes and geological uncertainty into long-term open-pit mine planning. *Minerals*, 9 (2):108, 2019.

A. Navarra, T. Grammatikopoulos, and K. Waters. Incorporation of geometallurgical modelling into long-term production planning. *Minerals Engineering*, 120:118–126, 2018.

G. Nelis. Sets of cluster shapes. https://github.com/gnelis/Clusters_shapes, 2021. Visited 05-05-2021.

G. Nelis and N. Morales. A mathematical model for the scheduling and definition of mining cuts in short-term mine planning. *Optimization and Engineering*, 2021.

G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization*. Wiley, New York, 1988.

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

G.T. Nwaila, Y. Ghorbani, M. Becker, H.E. Frimmel, J. Petersen, and S. Zhang. Geometallurgical approach for implications of ore blending on cyanide leaching and adsorption behavior of witwatersrand gold ores, south africa. *Natural Resources Research*, 20:1007–1030, 2020.

A. Parmentier and F. Meunier. Aircraft routing and crew pairing: Updated algorithms at air france. *Omega*, 93, 2020.

J. Ruiseco and M. Kumral. A practical approach to mine equipment sizing in relation to dig-limit optimization in complex orebodies: Multi-rock type, multi-process, and multi-metal case. *Natural Resources Research*, 26:23–35, 2017.

J.R. Ruiseco, J. Williams, and M. Kumral. Optimizing ore–waste dig-limits as part of operational mine planning through genetic algorithms. *Natural Resources Research*, 25(4): 99–121, 2016.

Y. A. Sari and M. Kumral. Dig-limits optimization through mixed-integer linear programming in open-pit mines. *Journal of the Operational Research Society*, 69(2):171–182, 2017.

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

M. Tabesh and H. Askari-Nasab. Automatic creation of mining polygons using hierarchical clustering techniques. *Journal of Mining Science*, 49(3):426 – 440, 2013.

M. Tabesh and H. Askari-Nasab. Clustering mining blocks in presence of geological uncertainty. *Mining Technology*, 2019.

M. Tabesh, C. Mieth, and H. Askari-Nasab. A multi-step approach to long-term open-pit production planning. *Int. J. Mining and Mineral Engineering*, 5(4):273 – 298, 2014.

L.M. Tavares and R.D.C. Kallemback. Grindability of binary ore blends in ball mills. *Minerals Engineering*, 41:115–120, 2013.

E. Van Tonder, D.A. Deglon, and T.J. Napier-Munn. The effect of ore blends on the mineral processing of platinum ores. *Minerals Engineering*, 23:621–626, 2010.

Y.V. Vasylchuk and C.V. Deutsch. Optimization of surface mining dig limits with a practical heuristic algorithm. *Mining, Metallurgy & Exploration*, 2019.

J. Williams, J. Singh, M. Kumral, and J. Ruiseco. Exploring deep learning for dig-limit optimization in open-pit mines. *Natural Resources Research*, 30:2085–2101, 2021.

D. Yan and R. Eaton. Breakage properties of ore blends. *Minerals Engineering*, 7(2/3): 185–199, 1994.

J. Zhang and R. Dimitrakopoulos. Stochastic optimization for a mineral value chain with nonlinear recovery and forward contracts. *Journal of the Operational Research Society*, 69 (6):864–875, 2018.

(Frédéric Meunier) CERMICS, École des Ponts ParisTech, 77455 Marne-la-Vallée, France
Email address: frederic.meunier@enpc.fr

(Gonzalo Nelis) Metallurgical Engineering Department, Universidad de Concepción, Concepción 4070386, Chile
Email address: gnelis@udec.cl

(Nelson Morales) Advanced Mining Technology Center & Mining Engineering Department, Universidad de Chile, Santiago 8370451, Chile
Email address: nelson.morales@amtc.cl