Column Generation for Mining Cut Definition with Geometallurgical Interactions

Gonzalo Nelis[1], Frédéric Meunier[2], and Nelson Morales[3]

[1]Metallurgical Engineering Department, Universidad de Concepción, Concepción 4070386, Chile. Email address: gnelis@udec.cl. (*corresponding author*)
[2]CERMICS, École des Ponts ParisTech, 77455 Marne-la-Vallée, France
[3]Département des génies civil, géologique et des mines, Polytechnique Montréal, 2500 Chem. de Polytechnique, Montréal, QC H3T 1J4, Canadá

Abstract

This work presents a novel approach to solve the mining cut definition problem with geometallurgical interactions in short-term mine planning for open-pit operations. 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, contrary 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 nonlinear function.

An industry-sized data set is used to show that the model can be solved in reasonable time. Also, the results show that nonlinear recovery functions influence the destination policy and expected profit. Therefore, following the traditional free selection policy (based on cut-off grade) is not the best strategy when geometallurgical interactions are considered.

Keywords: Dig-limit; Mining Cuts; Geometallurgy; Column Generation; Mine Planning; Short-term planning

Declarations

Funding. 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.
Conflicts of interest. The authors have no relevant financial or non-financial interests to disclose.
Availability of data and material. The datasets generated during and/or analyzed during the current study are not publicly available due to confidentiality issues but are available from the corresponding author on reasonable request.
Code availability. There is no public code available.

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 destination for each unit (usually referred to as a *block*), and constructing a feasible extraction schedule. 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: (i) material classification based on the block's properties and the loading equipment size, (ii) definition of mining cuts by time period, and (iii) scheduling optimally these cuts 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 plants can be modeled as a linear combination of the block's attributes. Additivity is the assumption that the properties of a combination 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 processed 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 aggregation of blocks. Accounting for this behavior results in nonlinear programs, which are extremely hard to solve using exact techniques.

The issue of blending properties is especially 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 extracted together and sent to the same destination. This definition 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, the critical eye of planners and the grade control procedures in the operation. In the literature, operational feasibility is captured using 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, a large set of feasible clusters is generated as an input of an optimization model. The advantages of this approach are related to both issues mentioned previously. Operational feasibility is achieved by construction, i.e., the cluster shapes already fulfill operational requirements. The problems of nonlinearity and nonadditivity on the blending 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. A column generation approach is proposed to solve the resulting integer program, since this technique is well suited to deal with the huge number of variables of the problem.

The proposed approach forms a complete methodology to solve the mining cut definition problem in short-term mine planning. It is worth emphasizing that it requires no assumptions on the functions used to obtain cluster profits or properties. Any geometallurgical model can be used within our methodology as long as a single value can be assigned for a cluster and destination 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.

A second contribution is a simple method to generate a set of possible shapes that are relevant from a practical point of view. 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 the next section. Section "Problem Formulation" formalizes the mining cut definition problem addressed in the paper. Section "Method for Solving the Problem" describes the proposed algorithm to solve the mining cut definition problem. Section "Generating a Collection of Clusters" proposes a method for generating practically relevant clusters. Section "Experiments" shows the results in a real case study and final conclusions and extensions are discussed in Section "Conclusions".

LITERATURE REVIEW

Geometallurgy and short-term selectivity are usually covered in different works and by different techniques. For this reason, this section is divided into two parts. A review on usual approaches to deal with selectivity and operational short-term plans is presented first. Techniques to incorporate geometallurgical variables in mine planning are introduced later.

Short-term planning 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.

An early approach to solve the dig-limit optimization problem is presented in Norrena and Deutsch (2002). The authors use a dig-limit polygon to define the boundaries between materials. Simulated annealing is used to modify the location of the vertices of the dig-limit polygon to maximize the expected profit. A penalty function is used to generate mineable shapes based on the angles formed by the polygon's vertices.

Richmond and Beasley (2004) propose a "floating circle algorithm" that defines dig-limits based on a moving circle in two dimensions. The minimum diameter of the circle accounts for the minimum mining width, and the destination of the material is based on the average grade inside the circle. Then, a local search heuristic improves the solution in terms of different financial-risk measures.

Wilde and Deutsch (2007) introduce a grade control algorithm into the feasibility planning stage. High-resolution simulated data is used to mimic short-term blasthole information, and an iterative algorithm is used to aggregate the high-resolution blocks into larger units based on their profit. The authors include a penalty in the objective function to avoid impractical shapes.

Isaaks et al. (2014) use a minimum 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 *frames*, which are block's aggregates of a given dimension to comply with the loading equipment selectivity. Each block must belong to at least one 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 dig-limits 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 efficiency 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.

There are commercial implementations of dig-limit optimization algorithms. Deutsch (2017) models the dig-limit optimization problem as a covering problem, and proposes a branch and bound algorithm to solve it. Simulated annealing is also used to improve solutions in reasonable times. A version of this algorithm is implemented in Maptek Vulcan$^{\text{TM}}$.

Another commercial implementation is found in OrePro® 3D (Poupeau et al., 2019; Hunt and La Rosa, 2019). The dig-limit definition is based on a fixed mining width size given by the equipment size and the mining direction. This implementation also includes a previous step where the blasting movement is modeled. Therefore, the dig-limit definition is performed on a post-blast bench to account for horizontal and vertical displacements of the materials.

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, in some cases this delimitation generates large areas that the planner must divide to find a feasible short-term schedule. Therefore, additional steps are required to organize the extraction of blocks in a smaller scale and to fulfill the capacity constraints. In terms of geometallurgy, these works ignore how the local aggregation of blocks might change properties related to profit, such as recovery or hardness. The total profit is usually modeled as the sum of the block's individual profit given by the dig-limit definition. Therefore, these models assume there is no geometallurgical interaction between adjacent blocks in the short-term.

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 plants 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 destinations, but the cluster properties are assumed to be linear and therefore complex geometallurgical interactions are ignored.

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 simulating some geometallurgical attributes at the block level, but assuming there is no complex interaction between blocks. Kumral (2011) uses this approach in a two-stage stochastic model to define a production schedule, where the destination decision is modeled as a recourse action that depends on geometallurgical variables. Morales et al. (2019) and Maleki et al. (2020) also propose a stochastic model with this approach that minimizes deviations from production targets. The fulfillment of these targets depends on geometallurgical variables such as throughput, metallurgical recovery and magnetic ratio.

Another common approach is considering a processing plant with the flexibility to change the operational mode to deal with different geometallurgical properties (Navarra et al., 2018; Van Den Boogaart et al., 2011). These works focus on how the processing plants can adapt to the ore feed to maximize metal recovered or profit. However, works that deal directly with the nonlinearity and nonadditivity of geometallurgical variables are scarce and mostly focused on long-term mine planning optimization. We briefly discuss these works to present current approaches for complex geometallurgical 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 blending and operational space needed in the short term. Zhang and Dimitrakopoulos (2018) present 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 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 plant are blended in a yearly timespan, 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 blending 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.

PROBLEM FORMULATION

The problem addressed in this work consists in partitioning the blocks in a bench of 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 must be sent to the same destination (stockpile, waste dump, mill, heap leach, etc.).

The selection of the clusters and their destination depend on the profit of sending a cluster at a given destination. We assume that extracting a cluster and processing its blocks in a plant has a profit that may depend in a nonlinear and nonadditive way on the blocks in the cluster and on the destination. A waste-dump is also considered, where clusters are discarded and no metal is recovered.

A maximum capacity limits the number of clusters that can be sent to each destination. This capacity is imposed over a given attribute of the clusters (weight, throughput, etc.). As in the case of the profit, 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. An example on how this set might be generated is presented afterwards, 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 $\mathcal{D}$ of *destinations*, a *mine capacity* $K \in \mathbb{R}_+$, and a *destination capacity* $Q_d \in \mathbb{R}_+$ for each $d \in \mathcal{D}$. 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 $q \colon \mathcal{C} \times \mathcal{D} \to \mathbb{R}_+$ such that $q(c, d)$ is the utilization of capacity Q_d by cluster c. The *profit* of sending cluster c to destination d is given by a known function $v \colon \mathcal{C} \times \mathcal{D} \to \mathbb{R}$.

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

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

and satisfying the following constraints:

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

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. $\mathcal{C}_b$ represents the set of all clusters $c \in \mathcal{C}$ such that $b \in c$. Binary variable $x_{c,d}$ equals 1 if cluster c is sent to destination d, and 0 otherwise.

$$
\begin{aligned}
\max \quad & \sum_{c\in\mathcal{C}} \sum_{d\in\mathcal{D}} v(c,d) x_{c,d} & & \text{(CUTOPT)}\\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}_b} \sum_{d\in\mathcal{D}} x_{c,d} = 1 & b \in B & \quad (2)\\
& \sum_{c\in\mathcal{C}} \sum_{d\in D} w(c) x_{c,d} \leqslant K & & \quad (3)\\
& \sum_{c\in\mathcal{C}} q(c,d) x_{c,d} \leqslant Q_d & d \in \mathcal{D} & \quad (4)\\
& x_{c,d} \in \{0,1\} & c \in \mathcal{C},\, d \in \mathcal{D}\,. & \quad (5)
\end{aligned}
$$

The objective function is the total profit obtained by the assignment of each cluster to a destination. Constraint (2) declares that each block must belong to a single cluster. Constraints (3) and (4) are the translation in the model of the constraints (1).

METHOD FOR SOLVING THE PROBLEM

We propose to address Problem (CUTOPT) 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 (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).

The proposed approach uses a special version of column generation, called *sifting* or *sprint* and introduced by Forrest (1989), to solve the linear relaxation of Problem (CUTOPT). 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).

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 (CUTOPT) with the sifting approach, where the columns are the pairs (c,d), with $c \in \mathcal{C}$ and $d \in \mathcal{D}$. Solve Problem (CUTOPT) restricted to the columns selected during the previous step with an off-the-shelf solver. Select all columns that may improve the current integer solution. Solve Problem (CUTOPT) restricted to this extended set of columns, again with a solver.

Complementary details are given hereafter.

Column generation approach

We explain how to solve the linear relaxation of Problem (CUTOPT). First, the pricing subproblem is introduced, and our approach to solve it is explained. Then, we describe the complete column generation algorithm in detail and how the optimal solution is obtained.

Pricing subproblem

The dual relaxation of Problem (CUTOPT) is the following optimization problem:

$$\min \quad \sum_{b\in B}\lambda_b + K\mu + \sum_{d\in\mathcal{D}} Q_d\omega_d \tag{6}$$

$$\text{s.t.} \quad w(c)\mu + q(c,d)\omega_d + \sum_{b\in c}\lambda_b \geqslant v(c,d) \qquad c\in\mathcal{C},\, d\in\mathcal{D} \tag{7}$$

$$\mu,\, \omega_d \geqslant 0 \qquad d\in\mathcal{D}\,. \tag{8}$$

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

$$p(c,d) = v(c,d) - w(c)\mu - q(c,d)\omega_d - \lambda(c)\,, \tag{9}$$

where $\lambda(c)$ is defined as $\sum_{b\in c}\lambda_b$. A column (c, d) may improve the current optimal value of the master problem only if $p(c, d) > 0$ (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, a vectorized approach is preferred: the reduced cost (Eq. (9)) is calculated for all columns. Then, the columns with the highest reduced costs are added to the master problem in each iteration.

Column generation algorithm

Algorithm 1 describes our column generation approach. Three auxiliary functions are used. CLUSTERS(B) generates the cluster set for a given block model B. INITIALFEASIBLESET(B) generates a feasible solution for (CUTOPT), since it is required for the initial iteration of the algorithm. In this work, an initial feasible solution is obtained 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, CUTOPT($\mathcal{C}^{init}$, $\mathcal{D}$) builds Problem (CUTOPT) using the initial feasible set $\mathcal{C}^{init}$ and the set of destinations $\mathcal{D}$.

We use $Cols$ as the set of columns represented by the pairs (c, d) obtained from the cluster set $\mathcal{C}$ and the set of destinations $\mathcal{D}$. 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 “Number of columns per iteration”.

Obtaining an integer solution

Column generation only permits solving to optimality the linear relaxation of (CUTOPT). 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 at the beginning of this section.

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 Integer Program or *Restricted IP*). Since column generation starts with an initial feasible solution of the

Algorithm 1 Column Generation Algorithm

```
1: initialize $\mathcal{D}, v, K, Q_d, N_{\max}$
2: $\mathcal{C} \leftarrow$ CLUSTERS($B$)
3: $Cols \leftarrow \mathcal{C} \times \mathcal{D}$
4: $\mathcal{C}^{init} \leftarrow$ INITIALFEASIBLESET($B$)
5: $m \leftarrow$ CUTOPT($\mathcal{C}^{init}, \mathcal{D}$)
6: while $Cols \neq \varnothing$ do
7:     solve linear relaxation of $m$
8:     $\widehat{\mathcal{C}}^{+} \leftarrow \varnothing$
9:     for $d \in \mathcal{D}$ do
10:        for $c \in \mathcal{C}$ do
11:            $p(c,d) \leftarrow v(c,d) - w(c)\mu - q(c,d)\omega_d - \lambda(c)$
12:        end for
13:        add $\{(c,d) \in Cols\}$ to $\widehat{\mathcal{C}}^{+}$ if $p(c,d) \geqslant 0$
14:        if $|\widehat{\mathcal{C}}^{+}| \geqslant N_{\max}$ then
15:            sort $\widehat{\mathcal{C}}^{+}$ by $p(c,d)$
16:            $\widehat{\mathcal{C}}^{+} \leftarrow$ first $N_{\max}$ elements of $\widehat{\mathcal{C}}^{+}$
17:            break for
18:        end if
19:    end for
20:    if $\widehat{\mathcal{C}}^{+} = \varnothing$ then
21:        No column with positive reduced cost found
22:        break while
23:    end if
24:    remove $(c,d) \in \widehat{\mathcal{C}}^{+}$ from $Cols$
25:    update $m$ with columns in $\widehat{\mathcal{C}}^{+}$
26: end while
27: return $m.solution$
```

mining cut definition problem, this method always provides a feasible integer solution. This integer solution may not be optimal for the complete master problem (CUTOPT). However, it provides a lower bound for the optimal value of (CUTOPT).

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 (CUTOPT). These columns are added to the master problem and it is finally solved to optimality (referred to as Final Integer Program or *Final IP*). Nemhauser and Wolsey noted that any optimal solution of Final IP is an optimal solution of Problem (CUTOPT), and the method is thus exact.

Algorithm 2 shows these steps, and it is performed after solving the linear relaxation through Algorithm 1. The overall method is thus Algorithm 1 followed by Algorithm 2.

GENERATING A COLLECTION OF CLUSTERS

In this section, a procedure to generate a possible collection $\mathcal{C}$ is proposed, which is relevant from a practical point of view. We remind the reader that this procedure is used as input for the mining cut definition problem. The method presented in the previous section 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, the method considers 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

Algorithm 2 Integer Solution

```
Require: $m.solution$: Optimal solution from Algorithm 1
Require: $Cols$: Unused columns from Algorithm 1
 1: $UB \leftarrow$ Optimal value of the linear relaxation of $m$
 2: impose: $m.variables \in \{0, 1\}$
 3: solve $m$ (Restricted IP)
 4: $LB \leftarrow$ Optimal value of $m$
 5: $\widehat{\mathcal{C}}^+ \leftarrow \varnothing$
 6: for $d \in \mathcal{D}$ do
 7:     for $c \in \mathcal{C}$ do
 8:         $p(c, d) \leftarrow v(c, d) - w(c)\mu - q(c, d)\omega_d - \lambda(c)$
 9:     end for
10: end for
11: add $\{(c, d) \in Cols\}$ to $\widehat{\mathcal{C}}^+$ if $p(c, d) \geqslant LB - UB$
12: update $m$ with columns in $\widehat{\mathcal{C}}^+$
13: solve $m$ (Final IP)
14: return $m.solution$
```

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. A maximum cluster size is also given by the engineers depending on the planning horizon and the expected timespan of the blending effect in the processing plants.

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 $[b_y^{\min}, b_y^{\max}]$
- 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).

This procedure is able to produce realistic cluster definitions. For instance, mining shovel selectivity can be modeled as a square of a certain minimum size to simulate the movement pattern of the loading equipment (Sari and Kumral, 2017; Vasylchuk and Deutsch, 2019). Smaller minimum sizes lead to less regular mining shapes, which tends to be more suitable to smaller loading equipment such as front loaders.

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 the previous section.

EXPERIMENTS

In this section, the proposed method is tested over a real case study. The solution returned by our method is analyzed in terms of mining cut definition, destination policy and total profit. Then, nonlinear recovery effects are introduced to show the changes in the destination policy. Finally, the algorithmic performance of the proposed approach is compared against the naive approach of solving Problem (CUTOPT) including all columns from the beginning (referred to as *Monolithic Model*).

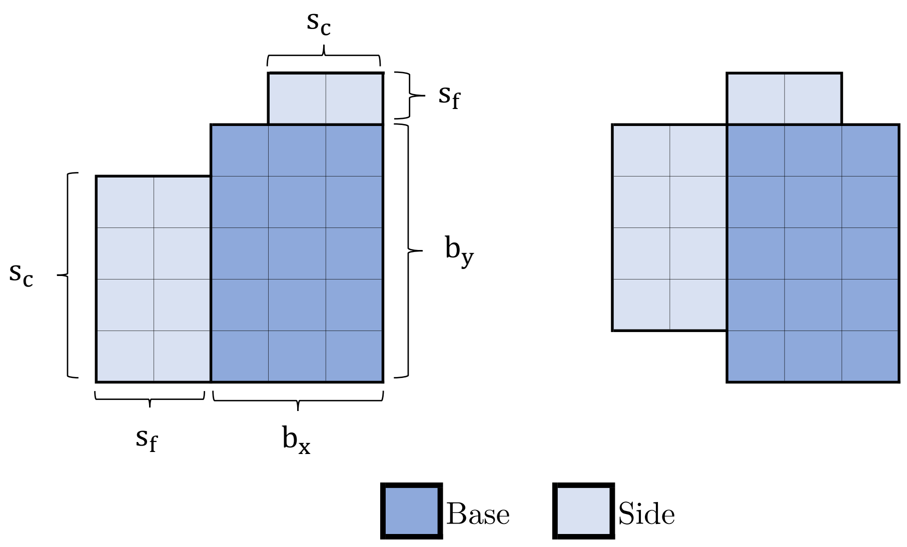

Fig. 1 Shape generation example. Left side shows the control parameters and right side shows a possible permutation with the same side shapes but in a different location

Case Study

The case study corresponds to a real copper mine. Blasthole data is used to simulate short-term grade information. Four benches with different sizes (216, 432, 720, and 912 blocks respectively) are selected to test the algorithms. Block size is 5m×5m×12m in all cases. A large set of 32,764 shapes is generated using the approach described in the previous section with the parameters shown in Table 1. A minimum base size of 2×3 blocks and maximum size of 40 blocks were considered for this case study.

Table 1 Parameters for the shape generation algorithm

	Min	Max
b_x	2	6
b_y	2	6
s_c	2	5
s_f	0	2

Using the largest set of shapes would be recommended to obtain the best feasible schedule. However, smaller subsets are also selected to show the algorithm behavior at different scales. The sizes of these subsamples are 8,000, 15,000, and 24,000 shapes, all drawn randomly from the largest set of 32,764 shapes. A comprehensive list of these sets can be found in Nelis (2021).

There are three destinations: two processing plants and one waste dump, where non-profitable material is discarded. The cluster tonnage is used as the weight parameter for the mining and processing capacities (i.e., $q(c, d) = w(c)$). Tables 2 and 3 describes the economic parameters used in all cases. Note that capacities vary according to the block model size.

Nonlinear recovery curves $R(g, d)$, depending on the copper content, are used for Plants 1 and 2, while no metal is recovered in the waste dump (Figure 2).

To incorporate blending effects on the profit valuation, metallurgical recovery is assigned based on the mean copper content for each cluster, $g(c)$. Eq. (10) 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).

$$v(c,d) = w(c)\Big(\left(P - C_s(d)\right) g(c) R(g(c),d) - C_p(d) - C_m\Big)\,. \tag{10}$$

Table 2 Common economic parameters

Parameter	Value
Copper Price (P)(\$/lb)	2.0
Mining Cost (C_m) (\$/t)	1.5
Mining Capacity (K) (kt)	
216 blocks	174.9
432 blocks	349.9
720 blocks	538.2
912 blocks	738.7

Table 3 Economic parameters for each destination

Parameter	Plant 1	Plant 2	Waste dump
Processing Cost ($C_p(d)$) (\$/t)	7.0	11.0	0.0
Selling Cost ($C_s(d)$) (\$/lb)	0.5	0.5	0.0
Destination Capacity (Q_d) (kt)			
216 blocks	49.6	49.6	179.9
432 blocks	101.2	101.2	349.9
720 blocks	170.1	170.1	538.2
912 blocks	226.8	226.8	738.7

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. This choice follows a detailed analysis of the effect of this parameter, which is shown afterwards.

Mining cut and destination policy

Figure 3.b shows the mining cut definition obtained by the proposed approach. It is an optimal solution. Each block is colored according to the cluster selected by the algorithm. Figure 3.c shows the destination policy induced by this optimal solution. 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 can 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 mining cut definition only contains clusters from the collection of feasible shapes described in the previous section. Other mining operations might define different feasible shapes and the main methods presented in Algorithms 1 and 2 would still provide the optimal solution.

Table 4 shows the profit achieved by our algorithm for all the block models and set of shapes tested. The columns show the profit achieved for the different sets of shapes. 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, the total profit is often considered as an upper bound for the mining cut definition problem.

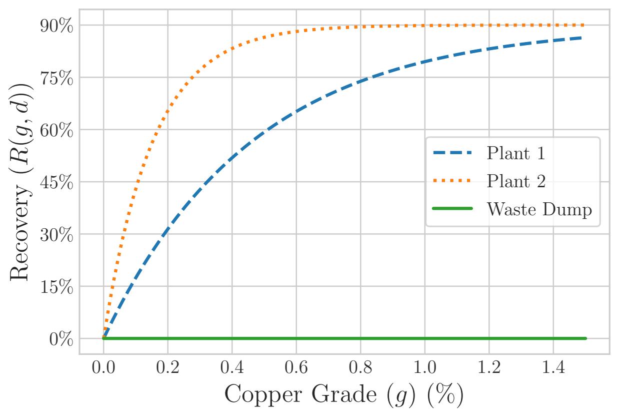

Fig. 2 Recovery Curves

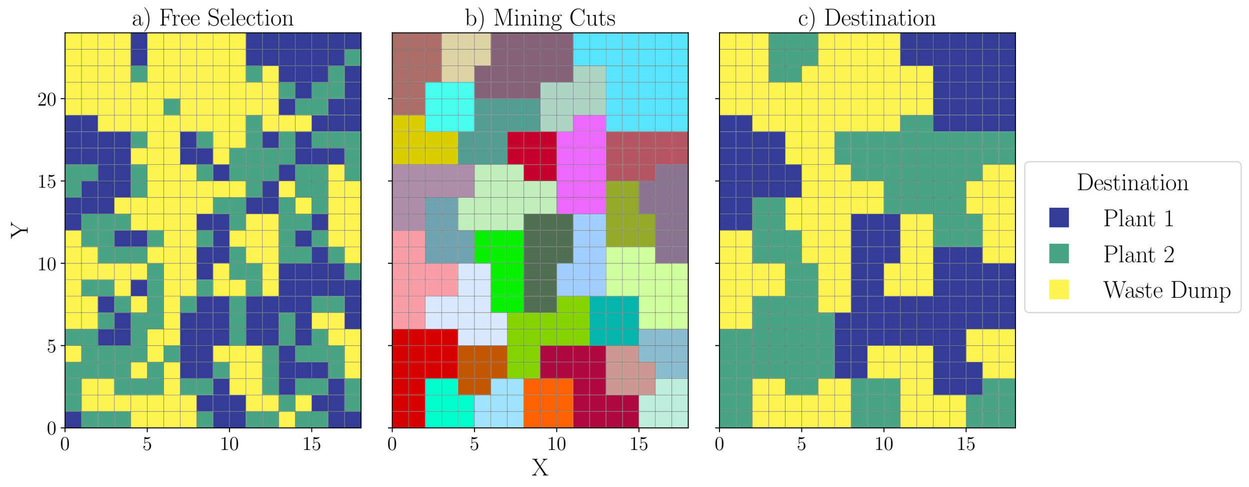

Fig. 3 Mining cut definition for 432 blocks and 32,764 shapes

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 larger 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, the difference 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.

Nonlinear recovery effect

This section presents an example on the effect of blending and nonlinear recovery for the dig-limits definition problem. We remind that Problem (CUTOPT) 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),

Table 4 Profit comparison

		Number of Shapes			
Number of Blocks	Free Selection (k$)	8,000 (k$)	15,000 (k$)	24,000 (k$)	32,764 (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

and any blending interaction is usually ignored. For this case study, geometallurgical interactions are incorporated assuming a perfect blending of the blocks inside each cluster, with a nonlinear recovery curve. A primary-response framework (Coward et al., 2009) is used to obtain the recovery of each cluster because using a direct spatial estimation must deal with its nonadditive behavior (Carrasco et al., 2008).

Figure 4 shows the destination policy obtained by solving Problem (CUTOPT) for the instance with 720 blocks and 32,764 shapes with and without blending interactions.

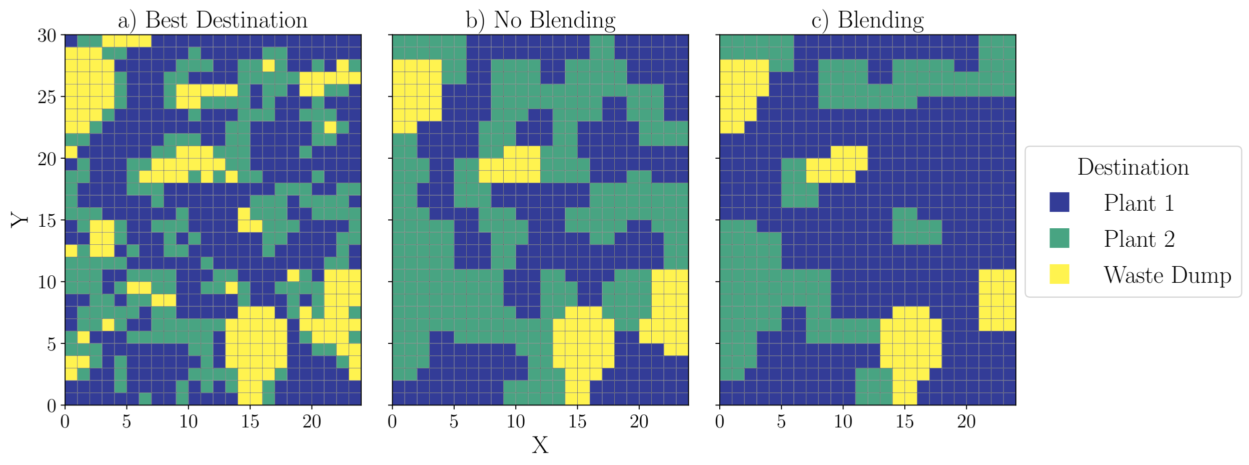

Fig. 4 Dig-limit definition comparison - 720 blocks and 32,764 shapes

Figure 4.a depicts the free selection policy. This definition, however, does not fulfill selectivity restrictions.

As discussed in the literature review, typical dig-limits algorithms ignore geometallurgical interactions of blocks extracted in the same mining cut. For our optimization problem, this is equivalent to define the cluster profit as the sum of the individual blocks profit, i.e., $v(c,d) = \sum_{b \in \mathcal{C}} v(b,d)$. 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 cluster shapes) regularizes the free selection policy. In broad terms, however, both the free selection and the dig-limit definition without blending interactions follow the same structures. However, if blending is introduced on the cluster profit, the dig-limit definition changes. Figure 4.c depicts the results of our method and using Eq. (10) to introduce blending effects into the profit. Plant 1 becomes much more prevalent than Plant 2, while the waste remains relatively unchanged.

Table 5 shows the profit for the destination policies. For this case, the free selection policy achieves the highest profit as expected if blending interactions are ignored. But this result changes with the introduction of blending: 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 shape (Figure 2), which favors blending blocks of low and high grade to yield a higher metal recovered content compared to processing these blocks independently.

Table 5 Profit and destination policies with and without blending.

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

Profit with blending is not calculated for the free selection policy since it is unclear how blocks are blended 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 blending 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 blending effects are expected. The recovery curve used in this work is rather simple, more sizable differences are expected with more complex geometallurgical interactions.

Algorithmic performance

In this section, we discuss the algorithmic performance of the proposed approach. Table 6 shows the results obtained by solving the linear relaxation of Problem (CUTOPT). Runtimes and objective function values are compared against solving the Monolithic Model with Gurobi, if possible. Reported fields are described next:

- Clusters:
  - **Time:** Time to generate the cluster set (given by function $\textsc{Clusters}(B)$).
  - **N:** Number of clusters generated.
- Column Generation:
  - **Overhead:** Time to build matrices and vectors used to perform the calculation described in Algorithm 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.
- Monolithic 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 7 shows the integer results for both approaches. As seen in Algorithm 2, the column generation approach requires solving two integer problems to get the optimal solution: the restricted problem to get a feasible lower bound for the optimal value, and the final problem that adds several columns (reported under the '# Cols.' field) to find the optimal solution of Problem (CUTOPT).

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 7 include this re-solving time in the Restricted IP and Final IP fields.

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

Table 6 Numerical results for the linear relaxation

		Clusters		Column Generation					Monolithic Model		
# Blocks	# Shapes	N	Time (s)	Overhead (s)	# Cols.	# Iters.	Time (s)	Value (k$)	Overhead (s)	Time (s)	Value (k$)
216	8,000	600,017	19	3	9,098	13	2	5,014.8	55	248	5,014.8
	15,000	1,120,597	35	5	14,437	19	5	5,042.0	107	351	5,042.0
	24,000	1,790,731	57	8	19,190	24	6	5,078.3	177	571	5,078.3
	32,764	2,447,373	78	11	18,313	23	6	5,102.9	235	630	5,102.9
432	8,000	1,785,611	58	8	31,100	30	29	11,675.0	182	791	11,675.0
	15,000	3,341,299	104	16	42,959	40	42	11,707.4	349	1,980	11,707.4
	24,000	5,342,551	173	26	54,374	50	53	11,748.8	618	-	-
	32,764	7,297,725	236	42	72,170	59	66	11,759.0	1,105	-	-
720	8,000	3,547,277	113	17	49,995	55	119	18,383.3	366	8,137	18,838.3
	15,000	6,642,073	218	34	67,919	71	172	18,876.2	864	-	-
	24,000	10,622,443	338	83	88,358	90	204	18,917.6	-	-	-
	32,764	14,507,157	464	136	101,899	103	211	18,939.7	-		-
912	8,000	4,806,006	153	23	59,288	61	155	22,239.8	539	-	-
	15,000	9,002,674	290	65	83,389	85	229	22,285.6	1,525	-	-
	24,000	14,399,046	491	144	102,912	104	278	22,331.8	-	-	-
	32,764	19,662,940	780	375	119,128	121	295	22,359.1	-	-	-

"-" denotes instances where an "out of memory" error occurred.

According to the results shown in Table 6, our method outperforms the Monolithic Model approach in every instance. The Monolithic 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 (CUTOPT), 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 Monolithic Model severely limits the instance size we are able to solve. In our tests, the largest instance the Monolithic 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 can solve an instance of 19.7 million columns (59 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 Plants 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 7), our method outperforms the Monolithic Model approach in every instance as well. Monolithic 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 Monolithic 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

Table 7 Numerical results for the integer solution

		Restricted IP		Final IP			Monolithic Model	
# Blocks	# Shapes	Time[a] (s)	Value (k$)	# Cols.	Time[a] (s)	Value (k$)	Time (s)	Value (k$)
216	8,000	3	4,962.1	49,761	6	4,972.5	104	4,972.5
	15,000	5	4,992.6	102,209	31	5,018.9	205	5,018.9
	24,000	2	5,068.0	25,231	3	5,068.9	189	5,068.9
	32,764	1	5,100.2	2,497	1	5,100.2	264	5,100.2
432	8,000	21	11,643.1	26,551	70	11,654.5	728	11,654.5
	15,000	15	11,699.8	101,887	40	11,701.6	508[b]	11,699.7[b]
	24,000	19	11,739.0	226,441	39	11,739.0	-	-
	32,764	16	11,750.8	213,956	63	11,750.8	-	-
720	8,000	289	18,796.8	943,768	582	18,818.7	2,986[b]	18,672.9[b]
	15,000	27	18,862.8	282,256	139	18,865.7	-	-
	24,000	30	18,910.3	296,589	100	18,912.6	-	-
	32,764	30	18,935.0	237,250	90	18,937.9	-	-
912	8,000	1,192	22,203.6	1,886,927	2,721	22,224.9	-	-
	15,000	286	22,276.6	647,034	805	22,279.0	-	-
	24,000	99	22,324.0	925,938	543	22,325.7	-	-
	32.764	120	22,353.8	884,984	541	22,353.8	-	-

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

for practical uses of this approach. This is also relevant for larger block models, where the number of columns added to find the optimal integer solution tends to be much larger compared to smaller block models.

On the same line, the block model size has a meaningful impact on all runtimes. The number of clusters increases with the block model size, and the constraints of the model follow the same pattern. This is particularly relevant for the integer solution, where the increase in block model size for the same set of shapes shows a notable increase in runtime. While all instances were solved in reasonable runtimes, other techniques to solve the integer problem could be explored for larger block models.

Interestingly, for a given block model, 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 for a given block model.

While our column generation algorithm is notably efficient in solving the linear relaxation of problem (CUTOPT), memory utilization seems the main limiting factor in the instance size we are able to solve. This is a disadvantage of the sifting approach: the algorithm must always keep the full list of clusters in memory. If this list is small enough to fit in memory, calculations are notably fast, which was the case for all instances shown in this case study. But if the list is too large, the decrease in performance could be meaningful. A different strategy might be needed to avoid this step with larger block models.

In terms of total runtime, our method 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-world applications.

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.

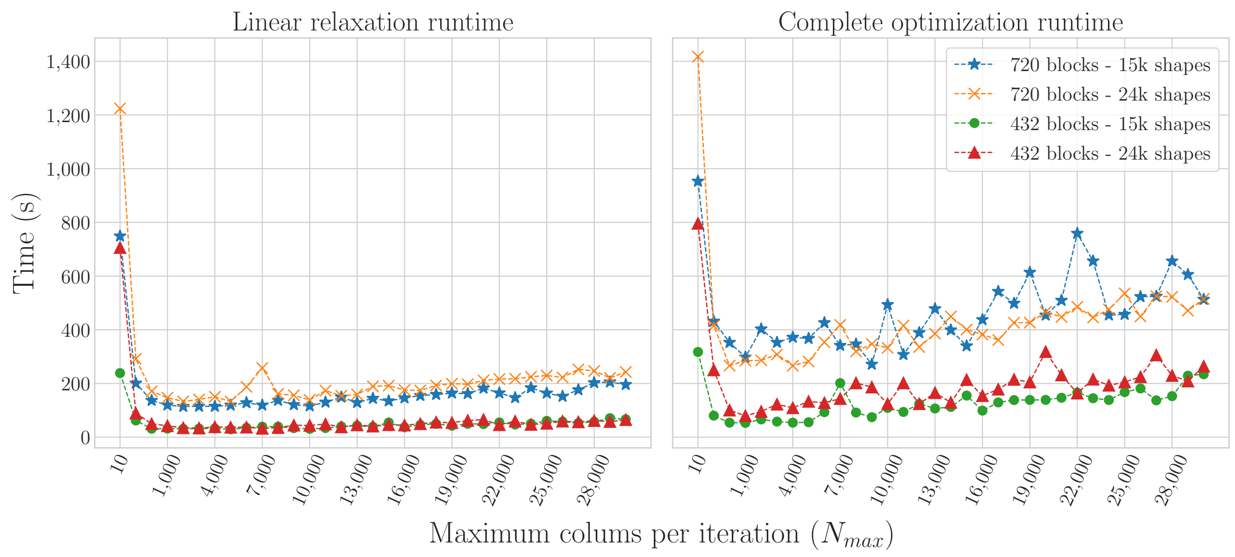

Fig. 5 Algorithm runtime for different values of $N_{\max}$

Figure 5 shows the linear relaxation and complete optimization runtimes (without overhead times) 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 number 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 in 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 this issue was not present with lower values of $N_{\max}$, this step was not incorporated in the current implementation.

CONCLUSIONS

This work presents a novel approach to tackle the mining cut definition problem. This 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 industry-size instances in reasonable runtimes for daily, weekly or monthly short-term horizons. Moreover, the proposed strategy is optimal for a given set of mining cut shapes.

The proposed approach allows the incorporation of complex, nonlinear or nonadditive geometallurgical interaction in each cluster. Including a simple interaction on the metallurgical recovery results in different dig-limits definitions compared to the traditional linear approach. Moreover, results show that following the free selection policy is not the best strategy to define operational mining cuts when nonlinear interactions are expected. Other geometallurgical attributes, such as grindability, throughput or acid consumption, could be incorporated to test their nonlinear or nonadditive behavior in the optimal destination policy.

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.

While the performance of the sifting approach was notably better than solving the Monolithic Model, our approach relies on always keeping the cluster set in memory. This would be problematic for larger block models, since the size of the cluster set increases rapidly with the number of blocks. Developing an algorithm that avoids the brute-force enumeration of the current approach in favor of generating, in each iteration, the best possible set of columns for the current solution would be a meaningful extension of this work.

Finally, a holistic approach must consider blending interactions from different benches and phases in each processing plant. Extending this work to tackle that problem would provide a better mining cut definition in all benches simultaneously.

REFERENCES

Anbil, R., Forrest, J., & Pulleyblank, W. (1998). Column generation and the airline crew pairing problem. *Documenta Mathematica*, 677–686.

Bixby, R., Gregory, J., Lustig, I., Marsten, R., & Shanno, D. (1992). Very large-scale linear programming: A case study in combining interior point and simplex methods. *Operations Research* 40(5), 85 – 897. http://doi.org/10.1287/opre.40.5.885

Blom, M., Pearce, A., & Stuckey, P. (2017). Short-term scheduling of an open-pit mine with multiple objectives. *Engineering Optimization* 49(7), 777–795. http://doi.org/10.1080/0305215X.2016.1218002

Cacchiani, V., & Salazar-González, J. (2017). Optimal solutions to a real-world integrated airline scheduling problem. *Transportation Science* 51(1), 250–268. http://doi.org/10.1287/trsc.2015.0655

Carrasco, P., Chilès, J., & Séguret, S. (2008). Additivity, metallurgical recovery, and grade. In Ortiz, J., & Emery, X. (Eds.), *Proceedings of the 8th international Geostatistics Congress* (pp. 109 – 113).

Chu, H., Gelman, E., & Johnson, E. (1997). Solving large scale crew scheduling problems. *European Journal of Operational Research* 97(2), 260–268. http://doi.org/10.1016/s0377-2217(96)00196-8

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

Del Castillo, M., & Dimitrakopoulos, R. (2016). A multivariate destination policy for geometallurgical variables in mineral value chains using coalition-formation clustering. *Resources Policy* 50, 322–332. http://doi.org/10.1016/j.resourpol.2016.10.003

Deutsch, M. (2017). A branch and bound algorithm for open pit grade control polygon optimization. In *Proceedings of the 38th International Symposium on the Applications of Computers and Operations Research in the Mineral Industry (APCOM)* (pp 3.1–3.8).

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

Van Den Boogaart, K., Weibflog, C., & Gutzmer, J. (2011). The value of adaptive mineral processing based on spatially varying ore fabric parameters. In Marschallinger, R., & Zobl, F. (Eds.), *Proceedings of IAMG 2011*.

Goodfellow, R., & Dimitrakopoulos, R. (2016). Global optimization of open pit mining complexes with uncertainty. *Applied Soft Computing* 40, 292–304. http://doi.org/10.1016/j.asoc.2015.11.038

Hunt, W., & La Rosa, D. (2019). How to use heave, movement, and ore block optimisation to increase grade and decrease dilution. In *SME Annual Conference*.

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

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

Kumar, A., & Dimitrakopoulos, R. (2019). Application of simultaneous stochastic optimization with geometallurgical decisions at a copper-gold mining complex. *Mining Technology* 128(2), 88–105. http://doi.org/10.1080/25726668.2019.1575053

Kumral, M. (2011). Incorporating geo-metallurgical information into mine production scheduling. *Journal of the Operational Research Society* 62, 60–68. http://doi.org/10.1057/jors.2009.174

Lübbecke, M., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research* 53(6), 1007-–1023. http://doi.org/10.1287/opre.1050.0234

Maleki, M., Jélvez, E., Emery, X., & Morales, N. (2020). Stochastic open-pit mine production scheduling: A case study of an iron deposit. *Minerals* 10(7), 585. http://doi.org/10.3390/min10070585

Morales, N., Seguel, S., Cáceres, A., Jélvez, E., & Alarcón, M. (2019). Incorporation of geometallurgical attributes and geological uncertainty into long-term open-pit mine planning. *Minerals* 9(2), 108. http://doi.org/10.3390/min9020108

Navarra, A., Grammatikopoulos, T., & Waters, K. (2018). Incorporation of geometallurgical modelling into long-term production planning. *Minerals Engineering* 120, 118–126. http://doi.org/10.1016/j.mineng.2018.02.010

Nelis, G. (2021). Sets of cluster shapes. https://github.com/gnelis/Clusters_shapes. Visited 25-oct-2021.

Nelis, G., & Morales, N. (2021). A mathematical model for the scheduling and definition of mining cuts in short-term mine planning. *Optimization and Engineering*. Advance online publication. http://doi.org/10.1007/s11081-020-09580-1

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

Newman, A., Rubio, E., Caro, R., Weintraub, A., & Eurek, K. (2010). A review of operations research in mine planning. *Interfaces* 40(3), 222–245. http://doi.org/10.1287/inte.1090.0492

Norrena, K., & Deutsch, C. (2002). Optimal determination of dig limits for improved grade control. In *Proceedings of the 30th International Symposium on the Applications of Computers and Operations Research in the Mineral Industry (APCOM).*

Nwaila, G., Ghorbani, Y., Becker, M., Frimmel, H., Petersen, J., & Zhang, S. (2020). 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. http://doi.org/10.1007/s11053-019-09522-4

Parmentier, A., & Meunier, F., (2020). Aircraft routing and crew pairing: Updated algorithms at air france. *Omega* 93, 102073. http://doi.org/10.1016/j.omega.2019.05.009

Poupeau, B., Hunt, W., & La Rosa, D. (2019). Blast induced ore movement: The missing step in achieving realistic reconciliations. In *Proceedings of the 11th International Mining Geology Conference*. Australian Institute of Mining and Metallurgy.

Richmond, A., & Beasley, J. (2004). Financially efficient dig-line delineation incorporating equipment constraints and grade uncertainty. *International Journal of Surface Mining, Reclamation and Environment* 18(2), 99–121. http://doi.org/10.1080/13895260412331295376

Ruiseco, J., & Kumral, M. (2017). 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. http://doi.org/10.1007/s11053-016-9301-8

Ruiseco, J., Williams, J., & Kumral, M. (2016). Optimizing ore–waste dig-limits as part of operational mine planning through genetic algorithms. *Natural Resources Research* 25(4), 99–121. http://doi.org/10.1007/s11053-016-9296-1

Sari, Y., & Kumral, M. (2017). Dig-limits optimization through mixed-integer linear programming in open-pit mines. *Journal of the Operational Research Society* 69(2), 171–182. http://doi.org/10.1057/s41274-017-0201-z

Tabesh, M., & Askari-Nasab, H. (2011). Two-stage clustering algorithm for block aggregation in open pit mines. *Mining Technology* 120(3), 158 – 169. http://doi.org/10.1179/1743286311Y.0000000009

Tabesh, M., & Askari-Nasab, H. (2013). Automatic creation of mining polygons using hierarchical clustering techniques. *Journal of Mining Science* 49(3), 426 – 440. http://doi.org/10.1134/S1062739149030106

Tabesh, M., & Askari-Nasab, H. (2019). Clustering mining blocks in presence of geological uncertainty. *Mining Technology* 128, 162–176. http://doi.org/10.1080/25726668.2019.1596425

Tabesh, M., Mieth, C., & Askari-Nasab, H. (2014). A multi-step approach to long-term open-pit production planning. *Int J Mining and Mineral Engineering* 5(4), 273 – 298. http://doi.org/10.1504/IJMME.2014.066577

Tavares, L., Kallemback, R. (2013). Grindability of binary ore blends in ball mills. *Minerals Engineering* 41, 115–120. http://doi.org/10.1016/j.mineng.2012.11.001

Van Tonder, E., Deglon, D., & Napier-Munn, T. (2010). The effect of ore blends on the mineral processing of platinum ores. *Minerals Engineering* 23, 621–626. http://doi.org/10.1016/j.mineng.2010.02.008

Vasylchuk, Y., & Deutsch, C. (2019). Optimization of surface mining dig limits with a practical heuristic algorithm. *Mining, Metallurgy & Exploration* 36, 773–784. http://doi.org/10.1007/s42461-019-0072-8

Wilde, B., Deutsch, C. (2007). Feasibility grade control (fgc): Simulation of grade control on geostatistical realizations. In *Centre for Computational Geostatistics Report 9 - 301.*

Williams, J., Singh, J., Kumral, M., & Ruiseco, J. (2021). Exploring deep learning for dig-limit optimization in open-pit mines. *Natural Resources Research* 30, 2085–2101. http://doi.org/10.1007/s11053-021-09864-y

Yan, D., & Eaton, R. (1994). Breakage properties of ore blends. *Minerals Engineering* 7(2/3), 185–199. http://doi.org/10.1016/0892-6875(94)90063-9

Zhang, J., & Dimitrakopoulos, R. (2018). Stochastic optimization for a mineral value chain with nonlinear recovery and forward contracts. *Journal of the Operational Research Society* 69(6), 864–875. http://doi.org/10.1057/s41274-017-0269-5