Stope Boundary Optimization Using A 3D Approximate Hybrid Algorithm

Vahid Nikbin, Majid Ataee-pour, Kourosh Shahriar and Yashar Pourrahimian
Mining Optimization Laboratory (MOL)
University of Alberta, Edmonton, Canada
Amirkabir University of Technology, Tehran, Iran

ABSTRACT

Determining stope boundaries is one of the critical steps to be taken when an underground mining method is selected; because of their significant impact on the profitability of the mining project, the stope boundaries have to be optimum to achieve maximum profits. This paper introduces a new hybrid algorithm that is a combination of dynamic programming and greedy algorithm. Although this proposed algorithm may fail to provide a true optimum solution, it generates better solutions than existing algorithms do. The new proposed algorithm and three existing algorithms are used to find the optimal stope boundaries on a real case ore body. The results demonstrate that the proposed algorithm can improve the profit by 117.78%, 16.86% and 0.42% compared to Floating Stope, Maximum Value Neighborhood (MVN), and Greedy algorithm solutions, respectively, on a real case study at a reasonable CPU time.

1. Introduction

Due to the increase in stripping ratios and haulage distances in open pit mines, mining costs increase. As a result, switching from open pit mining to underground mining may be more beneficial. Also, underground mining can be the only alternative for deposits with large overburdens.

When the underground mining method is selected, it is necessary to determine the optimal workable layout as stope boundaries. The main purpose of optimizing stope boundaries is to determine an appropriate plan to select the best combination of blocks in the block model. This plan should maximize the overall profit subject to geometrical, geotechnical, and safety constraints. Obviously, this plan should be optimal because even a minor deviation from the exact plan's value may result in wasting millions of dollars of mining capitals.

The existing algorithms are either heuristic and therefore do not guarantee the optimal solution or else they suffer from over simplifications that restrict their applications. Despite its importance, because of the inherent complexities of the problem, a comprehensive algorithm has not been reported. Most of the presented algorithms are heuristic, and true optimality is not guaranteed. Also, other algorithms with solutions that are assumed to be optimal, either fail to run on 3D problems or their applications are limited to a specific method. In sum, no efficient algorithm has yet been reported.

Most of the mining problems are large scale while exact optimization methods are usually not applicable to solve these problems. Inevitably, they should be solved by low computational time (meta) heuristic algorithms. There exist a large number of strong heuristic and metaheuristic algorithms in the literature to solve a variety of mining problems such as prediction of hangingwall stability, slope stability, waste tailing recycling potential and so on (Qi et al.,

[1]This paper, in its entirety, is available online from 14 May 2018 in Computer and Operations Research. https://doi.org/10.1016/j.cor.2018.05.012

2017, Qi et al., 2018a, Qi et al., 2018b, Qi and Tang, 2018). Before describing next sections, stope boundary optimization problem is defined as follows:

Let P an economic block model and $p_{i,j,\ k}$ be the obtained value from extraction of block $B_{i,j,\ k}$. Each stope is a subset of P that has a cube shape and its sizes along the main coordination axis are determined based on the minimum space required for labors and operating machines in each production cycle. Each combination of these stopes creates a feasible boundary which is denoted by q. Boundary value is equal to summation value of those blocks which are inside of the set q i.e., $value(q) = \sum_{(i,j,k);B_{i,j,k} \in q \cap p} p_{i,j,k}$. Our goal is maximizing boundary value among the all possible boundaries i.e., $\max\{\text{value}(q) : q \in Q\}$. Where, Q is a set which consists of all feasible boundaries.

The next section of this paper reviews relevant literature related to stope boundary optimization. Section 3 highlights the development of the algorithm for stope boundary optimization. Section 4 focuses on the application of the presented algorithm, and compares the results with other existing methods. The paper concludes in Section 5.

2. Methodology

Even four decades after the presentation of the first algorithm for optimizing stope boundaries, the growth rate of these algorithms has been slow. The reported works on the stope boundaries optimization problem are either sub-optimal or over simplified. For instance, although the Riddle's (1977) algorithm was designed based on the powerful dynamic programming approach, it does not consider the constraints simultaneously; this weakness may lead to a sub-optimal solution. Also, it is a two-dimensional algorithm that can only be used for the block caving method. The mixed-integer programming model that was presented by Ovanic and Young, 1995, Ovanic and Young, 1999) guarantees the optimal starting and ending points for stopes in block model rows. It is a one-dimension model and it is not clear how the model can be used to find the optimal stope boundaries in 2D or 3D ore bodies.

Floating Stope is a famous stope boundary algorithm that was proposed by Alford (1995) and implemented on the Datamine software package. In this algorithm, two envelopes (the inner and outer envelopes) are introduced as the output, and the final solution is located between the two. The final solution is determined by users, which means it is greatly dependent on their experience. Although, this algorithm is simple to understand and implement, its solution remains sub-optimal due to its heuristic logic.

The MVN algorithm, another heuristic algorithm, was proposed by Ataee-pour, 2000, Ataee-pour, 2005). It investigates all stopes that cover a block with a positive value and selects the stope with the highest value. It then updates the solution by adding this stope to the stope boundaries; this is continued until all positive blocks are checked. MVN is a fast algorithm and easy to understand; however, it is also a heuristic algorithm (similar to the Floating Stope), and the optimality of the solution is not guaranteed. The Optimum Limit Integrated Probable Stope (OLIPS), another dynamic programming algorithm, was designed by Jalali et al. (2004). Despite its strong mathematical background, the method uses complex logic and its application has been limited to inclined vein deposits. Also, no analysis has been reported for its solution time, but because it is complex it is expected to have a significant solution time. Therefore, applying this algorithm to large-sized problems may lead to restrictions due to hardware limitations.

Topal and Sens (2010) presented a heuristic algorithm that can use different strategies to optimize stope boundaries. At first, location, size and profit of all possible stopes are listed in a table if they have a positive value. In the next step, the maximum value stope is selected from this table and it is flagged as a minable stope. Then found stope and its neighboring overlapped stopes which have common blocks with this selected stope are eliminated from the aforementioned table. This procedure continues while all stopes are removed from the table. Low computation time and application on 3D problems are the main advantages of this algorithm. However, by removing overlapping stopes without performing any more accurate analysis on them some feasible combinations of stopes are rejected from the problem which may have good potentials to yield a higher stope layout (Sandanayake, 2014, Nhleko et al., 2017). Therefore, due to simplification of the problem by eliminating the overlapping stopes, its optimality is not guaranteed.

Bai et al., 2014, Bai et al., 2013 have used an intelligent method to find the optimal boundaries around the vertical raises based on the network flow algorithm. By converting the conventional orthogonal block model to a cylindrical block model with this method, the complicated underground mining problem is converted to an analogous, simpler, and well known open pit problem. However, the raise locations are determined by heuristic methods and the application of this technique is restricted to the sub-level stoping method.

Sandanayake et al., 2015a, Sandanayake et al., 2015b) developed two analogous heuristic algorithms which obtained a better solution than MVN. Both these reported works track the following five main steps sequentially in order to find the highest value stope layout. These steps are: (1) standardize the irregular block model (2) create all possible stopes (3) assign a value to each stope (4) define non-overlapping stopes set (5) find the maximum value combination among the mentioned set. These methods are 3D and their validation was successful compared to other alternatives (Erdogan et al., 2017). However, similar to Topal and Sens approach, these algorithms do not consider the overlapping assumptions and because of this simplification, they fail to guarantee the optimum solution.

Nikbin et al. (2017) introduced a greedy algorithm to find the highest value stope boundary subject the minimum stope sizes. At first, the algorithm finds the most valuable probable stope between the non-investigated stopes set. Then all of the blocks located in this stope are added to the optimal stope boundaries set if the stope value is positive. The found stope is removed from the non-investigated set. This procedure will iterate untill the stope value is positive. The validation was successful compared to MVN on a 2D hypothetical economic block model. However, this algorithm is not only heuristic but also it has not been employed on a real large-size ore body.

Advantages, application domains and some other related features of these explained algorithms have been summerized in Table 1.

Although the heuristic algorithms for the stope boundary problem mainly generate sub-optimal solutions, they usually have low processing time and include fewer simplifications. Consequently, they are more applicable than rigorous alternatives for solving the real world problems. The heuristic algorithms' solutions differ from each other. Any modification to these algorithms to decrease the optimality gap is very valuable. The contribution of this paper is to propose a new hybrid algorithm for stope boundaries optimization to achieve better results than what is possible with current algorithms. A new polynomial-time dynamic programming algorithm is proposed in this paper that guarantees the optimal solution for a given row or column. It is a 1D algorithm and in order to make it capable of solving 3D problems, it is combined by a greedy algorithm as a hybrid approach. Since dynamic programming optimality has a direct effect on the hybrid algorithm, highest value results was recorded by this proposed approach and its validation was completely successful on both hypothetical and real cases.

3. Methodology

In this paper, a new hybrid algorithm is proposed to optimize stope boundaries. This combines greedy and dynamic programming algorithms. The main problem is solved using the proposed greedy algorithm. In the greedy algorithm, by converting specific two-dimensional subsets of the original economic block model to single blocks, the initial complicated 3D problem is converted to a simple secondary one-dimensional sub-problem. Then, a new dynamic programming algorithm is applied to find the optimal boundaries on the aforementioned secondary sub-problem. Of the solutions to these simple sub-problems, the best is chosen based on greedy approach criterion. Then corresponding blocks to this solution are added to the initial problem solution.

Table 1. Comparison of common existing algorithms to solve stope boundary optimization problem

Algorithm	Model type	Mining method	Dimension	Mathematical formulation	Partial blocks	True optimality
Dynamic Programming	Fixed blocks	Block caving	2D	Yes	No	No
MIP model	Irregular blocks	All	1D	Yes	Yes	Yes
Floating stope	Fixed blocks	All	3D	No	Yes	No
MVN	Fixed blocks	All	3D	No	No	No
OLIPS	Fixed blocks	All	2D	Yes	No	Yes
Topal and Sens	Irregular blocks	All	3D	No	No	No
Network Flow	Cylindrical	Sublevel stoping	3D	No	No	No
Sandanayake et al.	Irregular blocks	All	3D	No	No	No
Greedy	Fixed blocks	All	3D	No	No	No

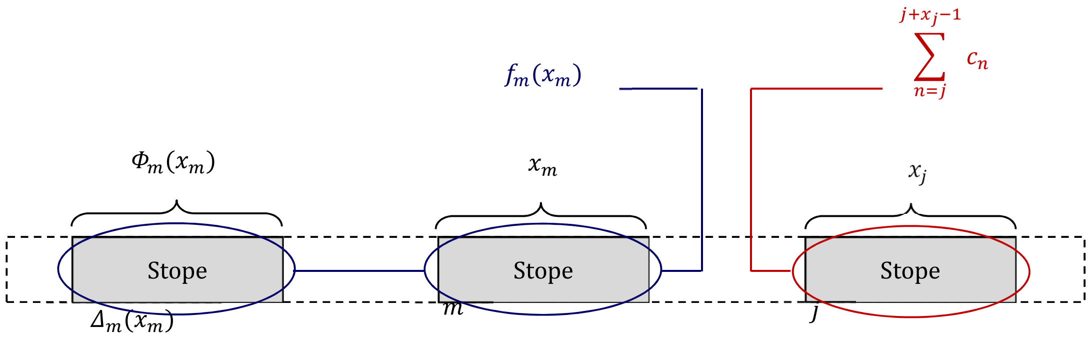


Fig 1. Relative position of algorithm's variables to each other.

3.1. One-dimensional exact dynamic programming algorithm (ODEDPA)

Dynamic programming is a well-known technique that can be used to solve a large number of optimization problems. Using this technique, a complicated problem is broken into smaller sub-problems and instead of solving the complex original problem, the sub-problems are solved and their solutions stored. After solving all of the sub-problems, the algorithm investigates the sub-problems solutions and combines the solutions to solve the original problem.

The new proposed algorithm is designed based on the dynamic programming procedure, and is suitable for finding the optimal stope boundaries in a given row or column of a block model. It is a one-dimensional polynomial-time algorithm and guarantees the optimal solution.

All dynamic programming algorithms almost use the same approach. They calculate the cumulative value for each state at each stage; find the highest cumulative value and track it to find other states located in the optimal solution. The algorithms' steps are:

1. Calculate the cumulative value for all states and store their best previous states.
2. Find the highest cumulative value at the final stage; its value presents the optimal stope boundaries value.
3. Track this highest value state at the final stage; dimensions, positions, and the number of the stopes that make the optimal boundaries.

The algorithm's variables and parameters are as follows:

d_j	Minimum stope size along j axis in terms of blocks
C	A 1D economic block model consists of J blocks along the j axis; input of dynamic programming algorithm
c_n	Value obtained by extracting n^{th} block of C
j	Current stage number
m	Previous stage number
x_j	Discrete state variable at the j^{th} stage; length of a stope that its left corner is located at j
x_m	Discrete state variable at the m^{th} stage; length of a stope that its left corner is located at m
$f_j(x_j)$	Cumulative value of x_j
$f_m(x_m)$	Cumulative value of x_m
$F_j(x_j)$	Cumulative blocks set of x_j; a subset of C that makes $f_j(x_j)$
Γ	State set; set of all feasible stope sizes along the j axis
$H_{j,m}(x_j, x_m)$	Overlap function; a function to recognize overlapping of two neighborhood stopes
$\Delta_j(x_j)$	The best previous stage ($\overline{m}$) that maximizes the cumulative value; previous stope position
$\Phi_j(x_j)$	The best previous state ($\bar{x}_m$) that maximizes the cumulative value; length of previous stope
$\pi_j(x_j)$	Optimal path of x_j; the union of all states that are created $F_j(x_j)$
$\lvert\pi_j(x_j)\rvert$	Optimal path's size, the number of states that are created $F_j(x_j)$
z^{dp}	Dynamic programming value
J^{dp}	Left corner position of stopes that are located at the dynamic programming solution

X^{dp}	Length of stopes that are located at the dynamic programming solution
$g(a,b)$	A function for comparing two numbers, 1 if a is greater than b, otherwise 0
$ge(a,b)$	A function for comparing two numbers, 1 if a is greater than or equal to b, otherwise 0
M^-	A large negative number
C^*	A 1D binary matrix consists of J blocks along j axis; output of dynamic programming algorithm
c_j^*	Value of j^{th} block of C^*, 1 if the block is selected, otherwise 0

Input and output matrices are denoted by capital letters and their elements are defined by lowercase letters.

The stage and state are the position (left corner) and dimension (length along j axis) of stopes respectively (Fig. 1). Since, each stope should has a minimum size, the state set (Γ) consists of all sizes equal to zero or greater than d$_j$ (Eq. (1)).

$$\Gamma = \{0\} \cup \{d_j, \dots, 2d_j - 1\} \tag{1}$$

If x_j is zero the state set is empty and no block is located in the stope. Therefore, the cumulative value is equal to the highest cumulative value among the previous states. This is expressed by Eq. (2).

$$\begin{aligned} f_j(x_j) &= \max\{f_m(x_m): m < j\} \\ x_j &= 0 \end{aligned} \tag{2}$$

In other situations (i.e., x_j is not zero), the right corner of stope is located at $j + x_j - 1$. Similar to the left corner, the right corner should not be located out of the block model boundaries. The right corner is forced to be located in C (i.e., it should be less than or equal to J). Therefore, the cumulative value is expressed by the following two-conditional equation (Eq. (3)):

$$\begin{array}{l} f_j(x_j) = \\ x_j \neq 0 \end{array} \begin{cases} -\infty & j + x_j - 1 > J \\ \max\{f_m(x_m) + M^-.H_{j,m}(x_j, x_m): m < j\} + \sum_{n=j}^{j+x_j-1} c_n & j + x_j - 1 \leq J \end{cases} \tag{3}$$

The current cumulative value ($f_j(x_j)$) is maximized by a specific combination of m and x_m. This combination is denoted by $\bar{m}$ and $\bar{x}_m$ symbols and they are stored in two independent functions (EQs. (4, 5)).

$$\Delta_j(x_j) = \bar{m} \tag{4}$$

$$\Phi_j(x_j) = \bar{x}_m \tag{5}$$

Also, the cumulative block set is defined by recursive Eq. (6). It is the union of the cumulative block set of $\bar{x}_m$ and the current state set.

$$F_j(x_j) = \begin{cases} F_{\bar{m}}(\bar{x}_m) \cup \emptyset & x_j = 0 \\ F_{\bar{m}}(\bar{x}_m) \cup \{B_n \in C: n \in \{j, \dots, j + x_j - 1\}\} & x_j \neq 0 \end{cases} \tag{6}$$

The union of all the state values from 1^{th} stage to j^{th} stage that create the $F_j(x_j)$, define the optimal path of x_j which is denoted by $\pi_j(x_j)$. The mathematical representation of $\pi_j(x_j)$ is illustrated in Eq. (7). This path consists of all C blocks that create $F_j(x_j)$. In order to track these paths, it is necessary to determine the path size. Therefore, after solving each sub-problem and finding the best value of previous stage and state, the current path size is updated by Eq. (8). The only difference between $\pi_j(x_j)$ and $\pi_{\overline{m}}(\bar{x}_m)$ is the combination of (j, x_j). Therefore, the relationship between the sizes of these two path sets is represented by Eq. (8).

$$\pi_j(x_j) = \left\{ \begin{pmatrix} \Delta_{\ddots_{\Delta_{\Delta_{\overline{m}}(\bar{x}_m)}(\Phi_{\overline{m}}(\bar{x}_m))}}\left(\ldots \Phi_{\Delta_{\overline{m}}(\bar{x}_m)}(\Phi_{\overline{m}}(\bar{x}_m))\right) \\ \Phi_{\ddots_{\Delta_{\Delta_{\overline{m}}(\bar{x}_m)}(\Phi_{\overline{m}}(\bar{x}_m))}}\left(\ldots \Phi_{\Delta_{\overline{m}}(\bar{x}_m)}(\Phi_{\overline{m}}(\bar{x}_m))\right) \end{pmatrix}, \ldots, \begin{pmatrix} \Delta_{\Delta_{\overline{m}}(\bar{x}_m)}(\Phi_{\overline{m}}(\bar{x}_m)) \\ \Phi_{\Delta_{\overline{m}}(\bar{x}_m)}(\Phi_{\overline{m}}(\bar{x}_m)) \end{pmatrix}, \begin{pmatrix} \Delta_{\overline{m}}(\bar{x}_m) \\ \Phi_{\overline{m}}(\bar{x}_m) \end{pmatrix}, \begin{pmatrix} \overline{m} \\ \bar{x}_m \end{pmatrix}, \begin{pmatrix} j \\ x_j \end{pmatrix} \right\} \tag{7}$$

$$|\pi_j(x_j)| = 1 + |\pi_{\overline{m}}(\bar{x}_m)| \tag{8}$$

This algorithm does not allow the two adjacent states to overlap, without any negative effects on the global optimality.

After identifying the overlap between the current state value and previous state path $\pi_m(x_m)$), selection of x_m as the best previous state of x_j is prevented by penalizing the recursive cumulative value equation, $((\overline{m}, \bar{x}_m) \neq (m, x_m))$. Overlapping is recognized by using a two-conditional equation (Eq. (9)) which is denoted by $H_{j,m}(x_j, x_m)$ and its output is binary; if two states overlap then the output is one; otherwise it is zero. If its output is one, the previous state has overlapped with the current state. By adding a large negative number to the cumulative value equation, the previous state has no chance of being selected as the best previous state of the current state. When x_j is zero, the state set is empty and obviously the overlap set is also empty. In the other situation, when x_j is not zero, overlapping is determined by comparing the right corner of previous stopes on the $\pi_m(x_m)$ to the left corner of current stope (j), the overlapping is determined.

$$H_{j,m}(x_j, x_m) = \begin{cases} 0 & x_j = 0 \\ \sum_{n=1}^{|\pi_m(x_m)|} g(\lambda_n + \varphi_n - 1, j) & x_j \neq 0 \end{cases} \tag{9}$$

Where λ_n and φ_n are respectively the left corner and the length of stopes that are located on $\pi_m(x_m)$. By knowing the size and the final element of this path, all other elements are determined by backward tracking Eqs. (10) and (11).

$$\lambda_n = \begin{cases} m & n = |\pi_m(x_m)| \\ \Delta_{\lambda_{n+1}}(\varphi_{n+1}) & n < |\pi_m(x_m)| \end{cases} \tag{10}$$

$$\varphi_n = \begin{cases} x_m & n = |\pi_m(x_m)| \\ \Phi_{\lambda_{n+1}}(\varphi_{n+1}) & n < |\pi_m(x_m)| \end{cases} \tag{11}$$

After calculating all the cumulative values, it is necessary to determine the optimal path $(\pi_J(\tilde{x}_J))$ and its value $(f_J(\tilde{x}_J))$. This optimal path is represented the dynamic programming solution. Therefore, the value of the optimal path is the same as the dynamic programming value (z^{dp}). This value is calculated by Eq. (12).

$$z^{dp} = \max\{f_J(x_J) : x_J \in \Gamma\} = f_J(\tilde{x}_J) \tag{12}$$

Also, the positions and dimensions of all the stopes that are located on the $\pi_J(\tilde{x}_J)$, are determined by backward tracking Eqs. (13) and (14).

$$J_n^* = \begin{cases} J & n = |\pi_J(\tilde{x}_J)| \\ \Delta_{J_{n+1}^*}(X_{n+1}^*) & n < |\pi_J(\tilde{x}_J)| \end{cases} \tag{13}$$

$$X_n^* = \begin{cases} \tilde{x}_J & n = |\pi_J(\tilde{x}_J)| \\ \Phi_{J_{n+1}^*}(X_{n+1}^*) & n < |\pi_J(\tilde{x}_J)| \end{cases} \tag{14}$$

To determine C blocks that are located on the $\pi_J(\tilde{x}_J)$, each block index is compared to the right and left corners of stopes on the $\pi_J(\tilde{x}_J)$. If this index becomes greater than or equal to the left stope's corner and also less than or equal to the right stop's corner simultaneously, Eq. (15) returns one; otherwise c_j^* is zero.

$$c_j^* = \sum_{\substack{n=1 \\ X_n^* \neq 0}}^{|\pi_J(\tilde{x}_J)|} ge(j, J_n^*) ge(J_n^* + X_n^* - 1, j) \tag{15}$$

Algorithm 1 Pseudo-code of the One-Dimensional Exact Dynamic Programming Algorithm

```
ODEDPA($C, d_j$)
1  $\Gamma = \{0\} \cup \{d_j, \ldots, 2d_j - 1\}$
2  for ($j = 1$ to $J$){
3      for ($x_j \in \Gamma$){
4          if ($x_j = 0$) {
5              $f_j(x_j) = \max\{f_m(x_m) : m < j\}$
6          }
7          else{
8              if ($j + x_{j-1} > J$){
9                  $f_j(x_j) = -\infty$
10             }
11             else ($j + x_{j-1} \leq J$){
12                 $f_j(x_j) = \max\{f_m(x_m) + M^-.H_{j,m}(x_j, x_m): m < j\} + \sum_{n=j}^{j+x_j-1} c_n$
13             }
14         }
15         $H_{j,m}(x_j, x_m)$ in the above equations is calculated using Eq. (9)
16         Store the best previous stage and state: $\Delta_j(x_j) = \bar{m}$ and $\Phi_j(x_j) = \bar{x}_m$
17         Update the path size: $|\pi_j(x_j)| = 1 + |\pi_{\bar{m}}(\bar{x}_m)|$
18     }
19 }
20 Find the highest cumulative value ($z^{dp}$) and its corresponding state ($\tilde{x}_J$):
   $z^{dp} = \max\{f_J(x_J)\} = f_J(\tilde{x}_J)$
21 Track the highest valued path ($J^{dp}, X^{dp}$) using Eqs. (13, 14)
22 Calculate values of $C^*$elements using Eq. (15)
23 Return $C^*$and $z^{dp}$
```

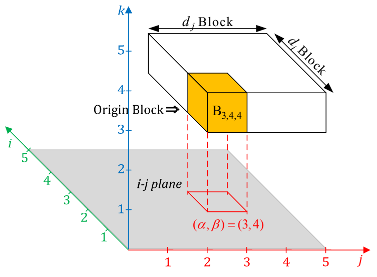
3.2. Three-dimensional approximate hybrid algorithm

The proposed hybrid algorithm uses a combination of greedy and dynamic programming approaches to optimize stope boundaries. The dynamic programming is optimal but the greedy approach does not consider its previous and future decisions in order to obtain the current decision: therefore its optimality is not guaranteed. Consequently, a combination of these two approaches as a hybrid algorithm remains sub-optimal. The symbols that are used in this hybrid algorithm are as follows:

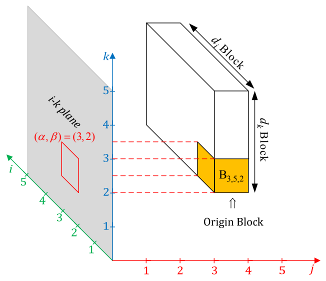
d_i	Minimum stope size along i axis; in terms of blocks
d_j	Minimum stope size along j axis; in terms of blocks

d_k	Minimum stope size along k axis; in terms of blocks
P	A 3D economic block model; Input of Hybrid algorithm
I	The number of P blocks along i axis
J	The number of P blocks along j axis
K	The number of P blocks along k axis
$\mathrm{B}_{m,n,o}$	A P block; m, n, o present its position along i, j, k axes, respectively
$p_{i,j,k}$	Net value obtained by extracting $\mathrm{B}_{i,j,k}$
z^h	Value of hybrid algorithm
P^*	Solution of hybrid algorithm; a subset of P that presents the optimal or near-optimal stope boundaries as the output of Hybrid algorithm
s	Scenario number
$\bar{I}$	The maximum allowable index for origin blocks along i axis
$\bar{J}$	The maximum allowable index for origin blocks along j axis
$\bar{K}$	The maximum allowable index for origin blocks along k axis
Ω	Non-investigated set
C^*	solution of dynamic programming

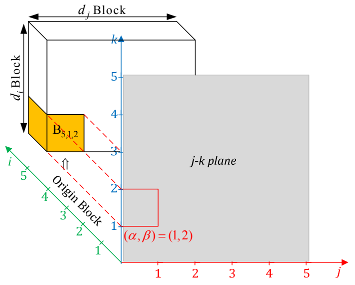
In this algorithm, by converting a subset of P blocks to a single block, a 1D matrix is built and is denoted by C. These subsets are 2D slices from P that own specific attributes which are defined as follows: the slices are fixed-size rectangles that are located in planes parallel to one of the $i - j$, $i - k$, or $j - k$ planes. They are identified by their origin blocks. Among the blocks of each slice, a unique block exists that has a minimum index on the both two slice directions which is named as origin block. Origin blocks have two indices on their parallel planes. The first index is α and the second one is denoted by β. Parallel slices with the same α and β are set behind each other in a specific direction. Three possible scenarios are defined to identify these directions and parallel planes. In the scenario 1, slices are parallel to the $i - j$ plane and their direction is along k axis (Fig 2.a). In the two other scenarios, No. 2 and No 3, slices are located on the planes parallel to $i - k$ and $j - k$ planes (Fig 2.a and Fig 2.b).



(a) Scenario 1(s=1)



(b) Scenario 2 (s=2)



(C) Scenario 3 (s=3)

Fig 2. Three possible scenarios for slice positions

In these scenarios, slices are arranged in the j and i directions, respectively. Dimensions of these slices depend on the minimum stope size at each plane. For instance, in scenario 1, the length and width of the slice is equal to d_i and d_j. The total value of $P\backslash P^*$ blocks which are located inside γ^{th} slice is denoted by c_γ. For each (α,β,s) combination located inside Ω, C is built and then Algorithm 1 is called. Algorithm 1 returns a binary 1-D matrix named as C^*. Based on the dynamic programming solution for each combination, the best combination is selected and denoted as $(\bar{\alpha},\bar{\beta},\bar{s})$. Our criteria for selecting the best combination is the absolute ratio of positive to negative blocks in the dynamic programming solution, i.e. $\max\{\sum_{\gamma=1}^{\Gamma} c_\gamma^* c_\gamma^+ / |\sum_{\gamma=1}^{\Gamma} c_\gamma^* c_\gamma^-|\}$. The objective function value at each iteration is called the best ratio. The dynamic programming solution that is obtained from the $(\bar{\alpha},\bar{\beta},\bar{s})$ combination is denoted as $\bar{C}^*$. In the next step, the $(\bar{\alpha},\bar{\beta},\bar{s})$ combination is removed from Ω. Then all of the blocks located in the γ^{th} slice are added to P^* if $\bar{c}_\gamma{}^*$ is equal to one. This procedure is iterated until the dynamic programming solution becomes positive. The pseudo-code of this algorithm has been illustrated as follows:

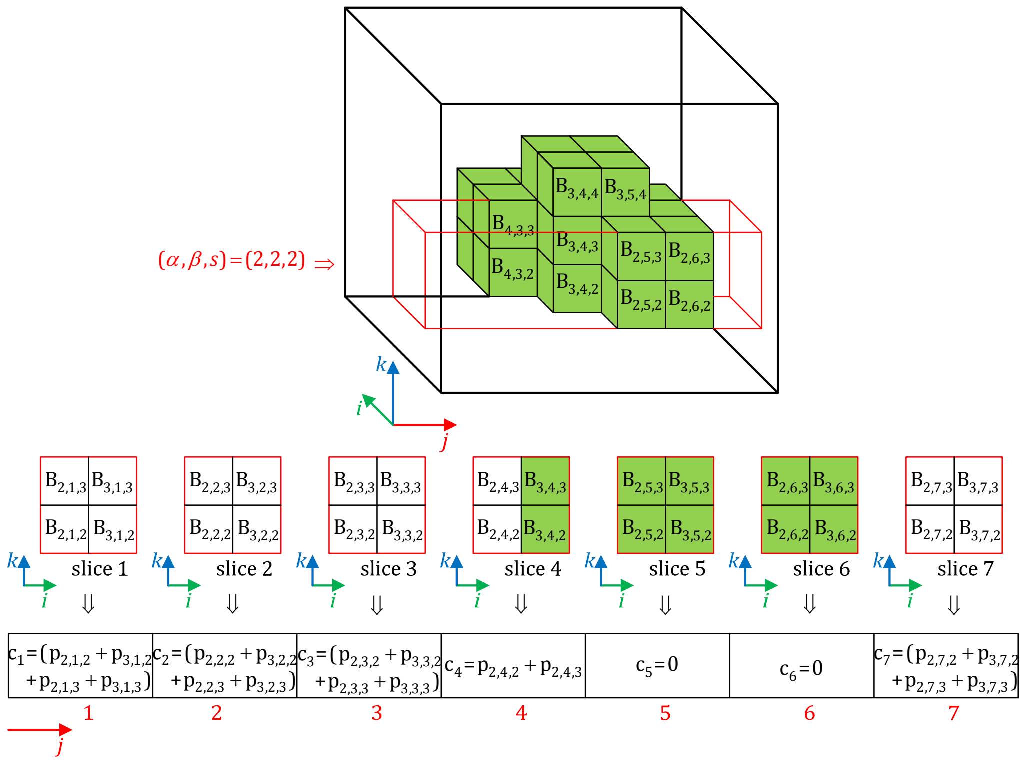
Algorithm 2 Pseudo-code of the Three-Dimensional Approximate Hybrid Algorithm

```
TDAHA $(P, d_i, d_j, d_k)$

1   $z^h = 0$
2   $P^* = \emptyset$
3   $(\bar{I},\bar{J},\bar{K}) = (I - d_i + 1, J - d_j + 1, K - d_k + 1)$
4   $\Omega = \{(i,j,1) \cup (i,k,2) \cup (j,k,3) : i \in \{1,\dots,\bar{I}\}, j \in \{1,\dots,\bar{J}\}, k \in \{1,\dots,\bar{K}\}\}$
5   $Best\ Ratio = 0$
6   $(\bar{\alpha},\bar{\beta},\bar{s},\bar{C}_{1\times\Gamma}) = Null$
7   for (s = 1 to 3){
8       $(A,B,\Gamma,l) = \begin{cases} (\bar{I},\bar{J},K,d_k) & s=1 \\ (\bar{I},\bar{K},J,d_j) & s=2 \\ (\bar{J},\bar{K},I,d_i) & s=3 \end{cases}$
9       for ($\alpha = 1$ to $A$){
10          for ($\beta = 1$ to $B$){
11              if (($\alpha,\beta,s) \in \Omega$){
12                  $C_{1\times\Gamma} = Null$
13                  for ($\gamma = 1$ to $\Gamma$){
14                      $c_\gamma = \begin{cases} \sum_{m=\alpha}^{\alpha+d_i-1} \sum_{n=\beta}^{\beta+d_j-1} p_{m,n,\gamma} : B_{m,n,\gamma} \notin P^* & s=1 \\ \sum_{m=\alpha}^{\alpha+d_i-1} \sum_{n=\beta}^{\beta+d_k-1} p_{m,\gamma,n} : B_{m,\gamma,n} \notin P^* & s=2 \\ \sum_{m=\alpha}^{\alpha+d_j-1} \sum_{n=\beta}^{\beta+d_k-1} p_{\gamma,m,n} : B_{\gamma,m,n} \notin P^* & s=3 \end{cases}$
15                  }
16                  Call Algorithm 1: ODEDPA($C_{1\times\Gamma}, l$)
17                  if ($\frac{\sum_{\gamma=1}^{\Gamma} c_\gamma^* c_\gamma^+}{|\sum_{\gamma=1}^{\Gamma} c_\gamma^* c_\gamma^-|} > Best\ Ratio$){
18                      $Best\ Ratio = \sum_{\gamma=1}^{\Gamma} c_\gamma^* c_\gamma^+ / |\sum_{\gamma=1}^{\Gamma} c_\gamma^* c_\gamma^-|$
19                      $(\bar{\alpha},\bar{\beta},\bar{s},\bar{\Gamma},\bar{C}_{1\times\Gamma},\bar{C}^*_{1\times\Gamma}) = (\alpha,\beta,s,\Gamma,C_{1\times\Gamma},C^*_{1\times\Gamma})$
20                  }
21              }
22          }
23      }
24  }
25  Remove $(\bar{\alpha},\bar{\beta},\bar{s})$ from non-investigated set: $\Omega = \Omega \backslash (\bar{\alpha},\bar{\beta},\bar{s})$
26  while ($\sum_{\gamma=1}^{\bar{\Gamma}} \bar{c}_\gamma^* \bar{c}_\gamma > 0$){
27          Update TDAHA value: $z^h = z^h + \sum_{\gamma=1}^{\bar{\Gamma}} \bar{c}_\gamma^* \bar{c}_\gamma$
28          Update TDAHA solution:
```

```
29        $P^* =$
      $P^* \cup \begin{cases} \{B_{m,n,\gamma}: m \in \{\{\bar{\alpha},\dots,\bar{\alpha}+d_i-1\}\}, n \in \left\{\{\bar{\beta},\dots,\bar{\beta}+d_j-1\}\right\}, \gamma \in \{1,\dots,\bar{\Gamma}\}\} & \bar{s}=1, \bar{c}^*_\gamma \\ \{B_{m,\gamma,n}: m \in \{\{\bar{\alpha},\dots,\bar{\alpha}+d_i-1\}\}, n \in \left\{\{\bar{\beta},\dots,\bar{\beta}+d_k-1\}\right\}, \gamma \in \{1,\dots,\bar{\Gamma}\}\} & \bar{s}=2, \bar{c}^*_\gamma \\ \{B_{\gamma,m,n}: m \in \left\{\{\bar{\alpha},\dots,\bar{\alpha}+d_j-1\}\right\}, n \in \left\{\{\bar{\beta},\dots,\bar{\beta}+d_k-1\}\right\}, \gamma \in \{1,\dots,\bar{\Gamma}\}\} & \bar{s}=3, \bar{c}^*_\gamma \end{cases}$
30
31        Go to Line 5
32  }
33  **Output** $z^h$ and $P^*$
```

A graphical visualization for converting 2D slices to 1D matrix (C) is illustrated in Fig 3. The green blocks are located in the P^* set and they are not considered in the calculation of C elements (Algorithm 2).

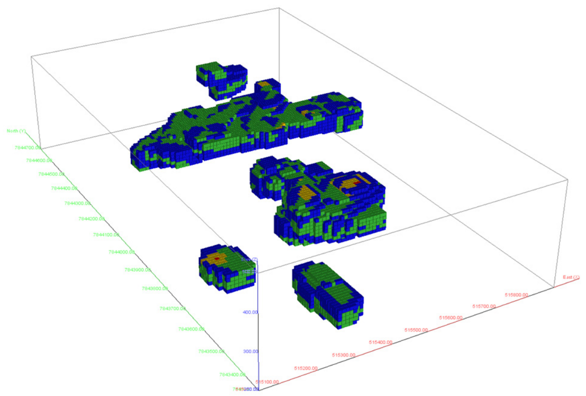


$(\alpha,\beta,s)=(2,2,2) \Rightarrow$
$B_{3,4,4}$ $B_{3,5,4}$ $B_{4,3,3}$ $B_{4,3,2}$ $B_{3,4,3}$ $B_{3,4,2}$ $B_{2,5,3}$ $B_{2,6,3}$ $B_{2,5,2}$ $B_{2,6,2}$
k i j
$B_{2,1,3}$ $B_{3,1,3}$ $B_{2,1,2}$ $B_{3,1,2}$ slice 1
$B_{2,2,3}$ $B_{3,2,3}$ $B_{2,2,2}$ $B_{3,2,2}$ slice 2
$B_{2,3,3}$ $B_{3,3,3}$ $B_{2,3,2}$ $B_{3,3,2}$ slice 3
$B_{2,4,3}$ $B_{3,4,3}$ $B_{2,4,2}$ $B_{3,4,2}$ slice 4
$B_{2,5,3}$ $B_{3,5,3}$ $B_{2,5,2}$ $B_{3,5,2}$ slice 5
$B_{2,6,3}$ $B_{3,6,3}$ $B_{2,6,2}$ $B_{3,6,2}$ slice 6
$B_{2,7,3}$ $B_{3,7,3}$ $B_{2,7,2}$ $B_{3,7,2}$ slice 7
$c_1=(p_{2,1,2}+p_{3,1,2}+p_{2,1,3}+p_{3,1,3})$
$c_2=(p_{2,2,2}+p_{3,2,2}+p_{2,2,3}+p_{3,2,3})$
$c_3=(p_{2,3,2}+p_{3,3,2}+p_{2,3,3}+p_{3,3,3})$
$c_4=p_{2,4,2}+p_{2,4,3}$
$c_5=0$
$c_6=0$
$c_7=(p_{2,7,2}+p_{3,7,2}+p_{2,7,3}+p_{3,7,3})$
1 2 3 4 5 6 7
j

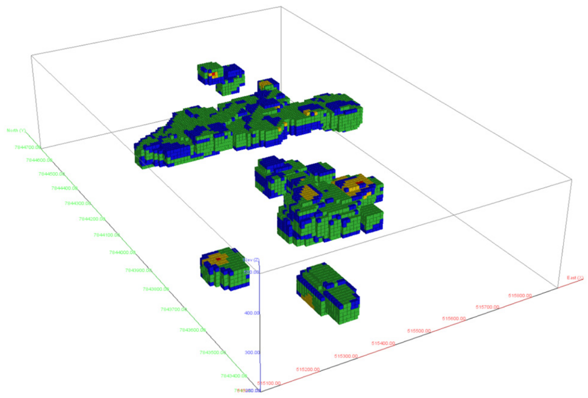
Fig 3. Converting 2D slices of P blocks to a single block in C

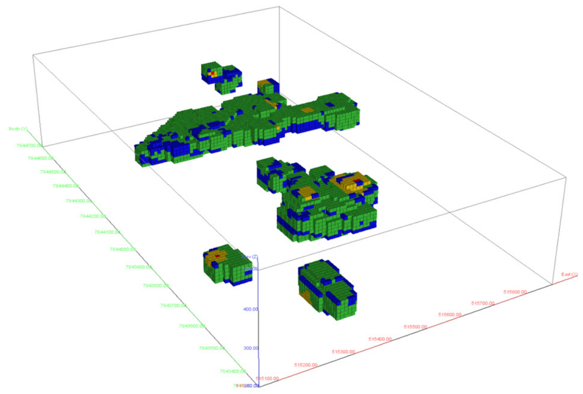
4. Application of algorithm in a case study

A poly-metal silver-zinc-lead deposit was modeled using the proposed algorithm. This deposit consists of 1,036,800 blocks. Block dimensions along the all main coordinate axes are 10 m and the average grade of silver, zinc and lead is 139.21 gr/ton, %3.21 and %0.26, respectively. Due to the high volume of the deposit's overburden, underground mining methods generate more profit than open pit method. Also, feasibility studies show that the sublevel stoping method is more compatible than other underground mining methods for extracting this deposit. The minimum length, width and height of the stopes are assumed to be 50 m. Fig. 4 shows the stope layouts generated by the Floating Stope, MVN, Greedy and proposed hybrid algorithms. The codes are written in C# programming language and the algorithms are run on a personal computer with an Intel(R) Core(TM) i5-2430 M CPU @ 2.40 GHz and 4.00GB of RAM. Table 2 shows the net profits and solution times of these algorithms.


(a) A 3D view of stopes obtained by Floating Stope algorithm (Inner Envelope)

BEV(10,000$)	
🟦	-30 to 0
🟩	0 to 15
🟨	15 to 30
🟥	30 to 45


(b) A 3D view of stopes obtained by MVN algorithm


(c) A 3D view of stopes obtained by greedy algorithm

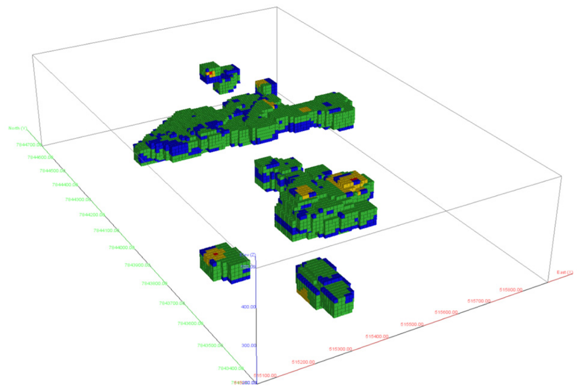

(d) A 3D view of stopes obtained by new proposed hybrid algorithm

Fig 4. Visualization of stope layouts from (a) Floating Stope, (b) MVN, (c) greedy, (d) proposed hybrid algorithm

Table 2. Results from the Floating Stope, MVN, greedy and new hybrid algorithms

Algorithm	Net profit ($)	Solution times	Number of minable blocks
Floating Stope	268,525,412	00:00:03	16,925
MVN	500,423,614	00:00:01	12,898
Greedy	582,352,644	00:02:08	11,678
Hybrid Algorithm	584,809,384	00:22:54	11,541

The new hybrid algorithm improves the profit by 117.87%, 16.86% and 0.42% compared to the Floating Stope, MVN and Greedy algorithms, respectively. Although the new algorithm requires more run time compared to its alternatives, its solution time still falls within a reasonable range given the size of the problem.

5. Conclusion

Undoubtedly, stope boundary optimization is one of the main mining problems which plays an important role in the economy of the mining industry. However, four decades after the first algorithm was presented for this problem, there is yet to be a comprehensive algorithm. Most of the algorithms are heuristic and their optimality is not guaranteed. Also, due to many simplifications in designing rigorous algorithms, they are not appropriate to solve real case problems. In this research, a new hybrid algorithm was introduced for stope boundary optimization. Although it may provide a locally optimum solution, it can find a better solution compared to its alternative algorithms at a reasonable CPU time. This hybrid algorithm and three existing algorithms were employed to find optimal stope boundary in a silver-zinc-lead deposit. The minimum and maximum improvements by this algorithm were 0.42% and 117.87%, respectively. The obtained results confirm the introduced algorithm's ability to find a better solution to stope boundary optimization problems. Better solution and better boundaries mean less waste of mining project capital, while help mining companies to develop sustainable projects that boost their profit. At the end, it should be noted that substitution of the proposed greedy algorithm by one of its alternatives such as genetic algorithm (GA), particle swarm optimization (PSO), simulated annealing (SA) and so on may yield to better solutions for the stope boundary optimization problem. For further investigation, it is suggested to check these meta-heuristics instead of the new proposed greedy algorithm.

6. References

[1] Alford, C., 1995. Optimisation in underground mine design. *In: 25th Interna-tional Symposium on the Application of Computers and Operations Research in the Mineral Industry*. The Australasian Institute of Mining and Metallurgy, pp. 213–218.

[2] Ataee-pour, M., 2000. A Heuristic Algorithm to Optimize Stope Boundaries Ph.D Thesis. University of Wollongong, New South Wales.

[3] Ataee-pour, M., 2005. A critical survey of the existing stope layout optimization techniques. *J. Min. Sci.* 41 (5), 447–466.

[4] Bai, X., Marcotte, D., Simon, R., 2013. Underground stope optimization with network flow method. Comput. *Geosci.* 52, 361–371.

[5] Bai, X., Marcotte, D., Simon, R., 2014. A heuristic sublevel stope optimizer with mul-tiple raises. *South. Afr. Inst. Mining Metall.* 114, 427–434.

[6] Erdogan, G., Cigla, M., Topal, E., Yavuz, M., 2017. Implementation and comparison of four stope boundary optimization algorithms in an existing underground mine. *Int. J. Min. Reclam. Environ.* 31 (6), 389–403.

[7] Jalali, S.E., Ataee-pour, M., Shahriar, K., 2004. A 2D dynamic programming algo-rithm to optimize stope boundary. *In: Proceedings of the Thirteenth Interna-tional Symposium on Mine Planning and Equipment Selection*, Wroclaw, Poland, pp. 45–52.

[8] Ovanic, J., Young, D., 1995. Economic optimisation of stope geometry using separable programming with special branch and bound techniques. *In: Third Canadian Conference on Computer Applications in the Mineral Industry*. Quebec, pp. 129–135.

[9] Ovanic, J., Young, D., 1999. Economic optimisation of open stope geometry. *In: 28th International APCOM Symposium. Colorado school of Mines*, Colorado, pp. 855–862.

[10] Riddle, J., 1977. A dynamic programming solution of a block-caving mine layout. *In: 14th International Symposium on the Application of Computers and Opera-tions Research in the Mineral Industry*. Society for Mining, Metallurgy and Exploration, Colorado, pp. 767–780.

[11] Sandanayake, D.S.S., 2014. Stope Boundary Optimization in Underground Mining Based on a Heuristic Approach Ph.D Thesis. Curtin University, Western Australian School of Mines.

[12] Sandanayake, D.S.S., Topal, E., Asad, M.W.A., 2015a. A heuristic approach to optimal design of an underground mine stope layout. *Appl. Soft Comput*. 30, 595–603.

[13] Sandanayake, D.S.S., Topal, E., Asad, M.W.A., 2015b. Designing an optimal stope layout for underground mining based on a heuristic algorithm. *Int. J. Min. Sci. Technol*. 25, 767–772.

[14] Topal, E., Sens, J., 2010. A new algorithm for stope boundary optimization. *Coal Sci. Eng*. 16 (2), 113–119.

[15] Nhleko, AS., Tholana, T., Neingo, PN., 2017. A review of underground stope boundary algorithms. *Resour. Policy* doi: 10.1016/j.resourpol.2017.12.004.

[16] Nikbin, V., Ataee-pour, M., Shahriar, K., Pourrahimian, Y., 2017. A Greedy Algorithm for Stope Boundaries Optimization. *8th Annual Report Mining Optimization Laboratory (MOL)*. University of Alberta, pp. 246–252. Report Eight.

[17] Qi, C., Fourie, A., Ma, G., Tang, X., Du, X., 2017. Comparative study of hybrid artificial intelligence approaches for predicting hangingwall stability. *J. Comput. Civil Eng*. 32 (2), 04017086.

[18] Qi, C., Fourie, A., Chen, Q., Zhang, Q., 2018a. A strength prediction model using artificial intelligence for recycling waste tailings as cemented paste backfill. *J. Cleaner Prod*. 183, 566–578.

[19] Qi, C., Fourie, A., Chen, Q., 2018b. Neural network and particle swarm optimization for predicting the unconfined compressive strength of cemented paste backfill. *Constr. Build. Mater*. 159, 473–478.

[20] Qi, C., Tang, X., 2018. Slope stability prediction using integrated metaheuristic and machine learning approaches: a comparative study. *Comput. Ind. Eng*. 118, 112–122.