Shahrood University of Technology

Iranian Society of Mining Engineering (IRSME)

Journal of Mining and Environment (JME)

journal homepage: www.jme.shahroodut.ac.ir

THOMSON REUTERS

Highest-Level Implementation of Push–Relabel Algorithm to Solve Ultimate Pit Limit Problem

Mahdi Talaie, Amin Mousavi* and Ahmad Reza Sayadi

Department of Mining Engineering, Faculty of Engineering, Tarbiat Modares University, Tehran, Iran

Article Info

Received 19 January 2021

Received in Revised form 27 February 2021

Accepted 15 March 2021

Published online 15 March 2021

DOI:10.22044/jme.2021.10481.1999

Keywords

Graph Theory

HI-PR

Push–Relabel

Maximum Flow

Open Pit

Abstract

Nowadays due to the existence of the economic and geological uncertainties and the increasing use of scenario-based project evaluation in the design of open-pit mines, it is necessary to find an exact algorithm that can determine the ultimate pit limit in a short period of time. Determining the ultimate pit limit is an important optimization problem that is solved to define what will be eventually extracted from the ground, and directly impacts the mining costs, revenue, choosing mining equipment, and approximation of surface infrastructures outside the pit. This problem is solved in order to maximize the non-discounted profit under the precedence relation (access) constraints. In this paper, the Highest-Level Push-Relabel (HI-PR) implementation of the push–relabel algorithm is discussed and applied in order to solve the ultimate pit limit optimization problem. HI-PR uses the highest-label selection rule, global update, and gap heuristics to reduce the computations. The proposed algorithm is implemented to solve the ultimate pit limit for the nine real-life benchmark case study publicly available on the Minelib website. The results obtained show that the HI-PR algorithm can reach the optimum solution in a less computational time than the currently implemented algorithms. For the largest dataset, which includes 112687 blocks and 3,035,483 constraints, the average solution time in 100 runs of the algorithm is 4 s, while IBM CPLEX, as an exact solver, could not find any feasible solution in 24 hours. This speeding-up capability can significantly improve the current challenges in the real-time mine planning and reconciliation, where fast and reliable solutions are required.

Abbreviation

2D	Two Dimensional
3D	Three Dimensional
DFS	Depth First Search
FIFO	First-in, First-out
HI-PR	Highest-Level Implementation of Push–Relabel
LG	Lerchs and Grossmann
Minelib	A library of Open-Pit Mining Problems
UPL	Ultimate Pit Limit

1. Introduction

Open-pit mine planning consists of a series of decision-making problems including defining the ultimate minable shape of a given mineral deposit and its overlying waste rocks. The optimum ultimate pit limit (UPL) defines a boundary that leads to the maximum non-discounted profit under the physical and pre-defined techno-economic configurations of the project. The pre-defined term states that any change in the input parameters such as the commodity price could potentially affect the whole profile of mining cash flow. An excellent mine planning tool should be able to respond to the input changes in a short period of time in order to prevent the production deficit. It is worth noting

Corresponding author: a_mousavi@modares.ac.ir (A. Mousavi).

that a mine plan that improves 1% in the net present value can represent millions of dollars in respect to the mining operation scale.

The UPL determination is the main core of a mine planning practice. From a discrete optimization viewpoint, UPL is formed by a set of rectangular blocks that should be extracted from the ground in the entire life of the mine. A block is a part of the mining area that represents a certain volume of rock with its estimated attributes. A set of such blocks is called a geological block model. The attributes such as the ore grade and density are estimated for each block using the geostatistical methods based on the exploration studies [1]. The economic values of blocks are calculated considering a cut-off grade, mining and processing costs, product price, and other technical parameters such as the mining recovery. The blocks with a positive value are defined as ore; otherwise, as waste. The optimum UPL is a boundary of mining that achieves the maximum positive value of extraction.

In order to model the UPL problem, it is crucial to obtain the block economic value from Equation (1), presented by Bakhtavar et al. [2], where v_i is the block economic value, P is the unit selling price of ore, C_s is the unit selling cost of ore, r is the total ore recovery, g_i is the block grade, T_o is the total amount of ore in block, C_o is the unit operational cost of ore extraction, C_w is the unit cost of waste removal, and T_w is the total amount of waste in each block.

$$v_i = (P - C_s).r.g_i.T_o - C_o.T_o - C_w.T_w \quad (1)$$

The UPL problem can be modeled as a binary programming model as Equation 2, where x is the decision variable for mining of block i, v is the economic value of block i, and B is the set of all blocks.

$$\text{Maximize} \quad \sum_{i \in B} v_i x_i \quad (2)$$

$$\text{Subject to} \quad x_i \le x_j \quad \forall i, j \in B \,\&\, j \in A_i \quad (3)$$

$$x_i \in \{0,1\} \quad i \in B \quad (4)$$

In the above model, A_i is a set of blocks that must be extracted before block i to physically and safely access this block. The number of predecessors in A_i may vary for different blocks depending on the location of block i, regional geological structure, and rock stability measures. A_i is called a predecessor set for block i, and it should be defined before running a solver for the UPL model. The model presented above is used for determination of UPL in all algorithms.

Solving the UPL problem has been an interesting optimization problem since the emergence of the personalized computer age in the 1960s. Surprisingly, in an early study, a polynomial graph-based algorithm has been presented by Lerchs and Grossmann [3] that is able to obtain the optimum solution for the UPL problem. This algorithm is known as the LG algorithm, and is widely used in the mine design packages [4]. However, the complexity of the algorithm and the low-capacity computers directed other researchers toward developing new solution techniques. Zhao and Kim [5] have appended some heuristics to the original LG algorithm in order to reduce the computation time while keeping the optimality of the solution. Khalokakaie and Dowd [6] have extended a modified version of the LG algorithm with the capability of considering variable wall slopes, which imply a variable predecessor set. Giannini [7] has applied a maximum flow algorithm in order to solve the UPL problem. Underwood and Tolwinski [8] have proposed a combined mathematical programming and graph theory approach and, employing a dual simplex algorithm to find the optimum solution of the UPL problem. Hochbaum and Chen [9] have discussed the LG algorithm's computational complexity, applying a maximum flow push–relabel algorithm to solve UPL.

Several studies have used dynamic programming for the UPL problem. In the study by Lerchs and Grossmann [3], a dynamic programming algorithm has been proposed for the 2D optimization of UPL. Later, Koenigsberg [10] presented a dynamic programming algorithm in order to solve the problem in a 3D approach. Erarslan and Celebi [11] have added production sequencing to the original UPL problem and have used a dynamic programming algorithm to optimize both problems simultaneously. Najafi *et al.* [12] have proposed the Dijkstra's algorithm in order to solve UPL and have compared it using the 2D dynamic programming.

Since 1965 and the emergence of the UPL problem, several heuristics approaches have been proposed in order to solve the problem in a shorter period of time. Pana [13] has proposed a floating cone algorithm, and Wright [14] has extended a modified version of this algorithm. The floating cone algorithm was widely used in the early days of computerized mine planning due to its simplicity and its low computational complexity. Achireko and Frimpong [15, 16] and Sayadi *et al.* [17] have used the neural network-based heuristics in order to solve the problem under uncertainty and deleterious element constraints. A brief summary of the UPL solution algorithms is shown in Table 1.

Although solving the generic form of the UPL problem is not a challenge, finding a computationally cheaper algorithm to solve the UPL problem is still of interest to the mine planners and managers for several reasons. First, revolutionizing technology in data collection creates new inputs for mine planning, which enforces extensive reconciliation efforts. Secondly, commodity price volatility and inherent grade uncertainty necessitate fast and reliable solution techniques to solve UPL for a few million simulation realizations. Thirdly, the UPL solution will still be the main pillar of automatic-oriented mine planning tools and practice such as automated road design [18] or open-pit to the underground transition optimization problem [19]. Finally, a more flexible algorithm is of interest to the variation of the UPL problem where an additional constraint may be added to the generic version of UPL. Consideration of mining royalty [20] or semi and full in-pit crushing [21, 22] are examples of the extended UPL problem. Therefore, a flexible algorithm is required to use in such more sophisticated instances.

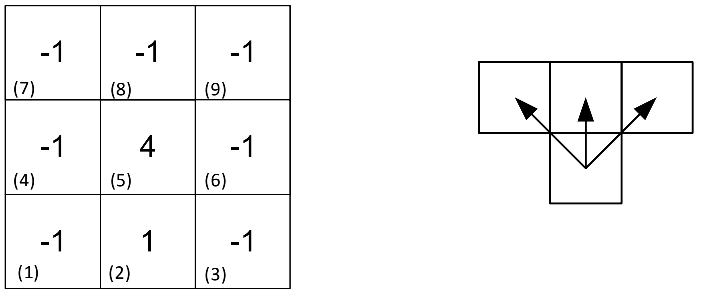
Table 1. A brief summary of previous UPL algorithms.

Researcher(s)	Year	Optimization method	Exact algorithm	3D model
Lerchs & Grossmann [3]	1965	Dynamic programing	Yes	No
Lerchs & Grossmann[3]	1965	Graph theory	Yes	Yes
Pana [13]	1965	Heuristic	No	No
Meyer [23]	1969	Linear programing	Yes	Yes
Janson & Sharp [24]	1971	Dynamic programing	No	Yes
Koenigsberg [10]	1982	Dynamic programing	Yes	Yes
Giannini [7]	1990	Graph theory	Yes	Yes
Zhao & Kim [5]	1992	Graph theory	No	Yes
Underwood & Tolwinski [8]	1998	Graph theory	Yes	Yes
Frimpong & Achireko [25]	1998	Heuristic	No	Yes
Wright [14]	1999	Heuristic	No	No
Khalokakaei & Dowd [6]	2000	Graph theory	Yes	Yes
Hochbaum & Chen [9]	2000	Graph theory	Yes	Yes
Erarsalan & Celebi [26]	2001	Dynamic programming	Yes	Yes
Frimpong [27]	2002	Heuristic	No	Yes
Sayyadi *et al.* [17]	2011	Heuristic	No	Yes
Elahi *et al.* [28]	2012	Heuristic	No	No
Hay *et al.* [22]	2019	Hybrid heuristic and graph theory	No	Yes

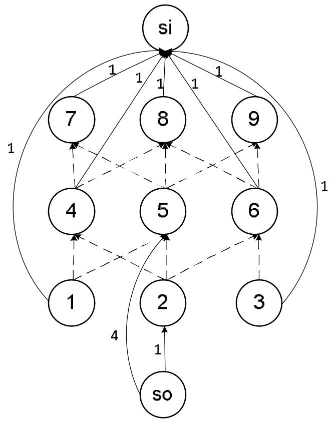
In this work, the so-called HI-PR implementation of the push–relabel algorithm was coded and used to solve the UPL problem. The implementation of this algorithm for the UPL problem is the novelty of this paper. Although several exact algorithms in the literature have been applied to optimally solve the UPL problem, the implemented HI-PR algorithm can significantly improve the solution time. A successful implementation of this algorithm will be a key pillar of the upcoming research projects for real-time capacitated UPL and scenario-based production planning and optimization problems, which are not in the scope of this paper. However, the main part of the solutions to those problems is the HI-PR algorithm, which is proposed in this paper. The computation time of the HI-PR algorithm was tested and validated on several benchmark real-life case studies.

2. Network modeling of mineral deposits

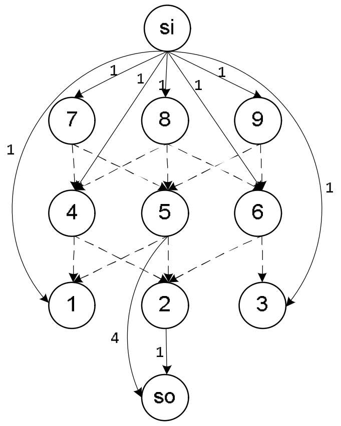
The first step in solving UPL using the network flow algorithms including HI-PR is to create a representative graph of the geological block model. In order to construct the corresponding graph, each block is considered as a vertex with the weight equal to the block economic value. Directed arc-connected vertices are defined according to the precedence relationships. The most common patterns used to construct the precedence relationships are the 1:5 pattern (to extract a given block, five overlying blocks should be extracted in advance), 1:9 pattern, and 1:5:8 (or knight move) pattern [29]. Figure 1 shows a 2D side-view of a block model with nine blocks and the corresponding network model for both the original and reverse graphs. In this example, the precedence relations are generated according to a 1:3 pattern, and the block economic values are tagged inside the blocks. The next step is to add two dummy vertices as the source and the sink vertices (so and si). An arc should connect the source to each positive vertex, and an arc should originate from each negative vertex to the sink. The capacity of an arc between the source/sink and a vertex is equal to the economic value of the corresponding block. The arcs that represent the precedence relationships have an infinite capacity.



(a)



(b)



(c)

Figure 1. a) A simple 2D block model and its 1:3 precedence relations pattern; b) and c) are the corresponding original and reverse networks.

3. Generic push–relabel algorithm

A push–relabel algorithm is a polynomial-time algorithm for solving the maximum flow problem. As Hochbaum and Chen [9] have stated, this algorithm has been first presented by Goldberg. Ahuja *et al.* [30] have discussed the details of this algorithm and the other techniques that can improve the time complexity of this algorithm. According to Ahuja *et al.*, the time complexity of the generic push–relabel algorithm is $O(nm + n^2 \log U)$, where n and m represent the number of vertices and arcs, respectively, and U is the maximum edge capacity.

In the push–relabel algorithm, a function $d: N \to Z^+ \cup \{0\}$ is defined and named the distance label for vertex i; d is a valid function if:

$$\begin{cases} d_{soi} = 0 \\ d_i \le d_j + 1 \text{ for } i, j \in N \,\&\, (i, j) \in A \end{cases} \tag{5}$$

In addition, the arc (i, j) is an admissible arc if $d_i = d_j + 1$; otherwise, it is an admissible arc. A path (a series of arcs) is called admissible if it only contains admissible arcs.

The generic push–relabel algorithm starts by pushing all flows of the source to the adjacent vertices. Based on the vertex balance constraint, the excess (e_i) flow of each vertex i is calculated as Equation (6).

$$e_i = \sum_{j:(j,i)\in A} x_{ji} - \sum_{l:(i,l)\in A} x_{il} \geq 0 \quad i \in N - \{So, Si\} \qquad (6)$$

$e_i \geq 0$ for all nodes except source;

$e_i < 0$ for source source;

If $e_i > 0$, the node i is called an active vertex (except sink); otherwise, it is called an inactive vertex.

In the next step, the algorithm selects one active vertex and tries to send the excess flows of this vertex to a sink or another node that is closer to the sink through admissible arcs. In order to find the admissible arcs, the distance label algorithm is applied. If all the outgoing arcs of node i are saturated and e_i is still non-zero, then vertex i will be relabeled and e_i will be returned to the source. This process terminates when all the excess flows are changed to zero. In order to show how the UPL problem is solved by the push–relabel algorithm, a 2D instance of this problem is solved and the corresponding push and relabeling iterations are shown in Figure 2.

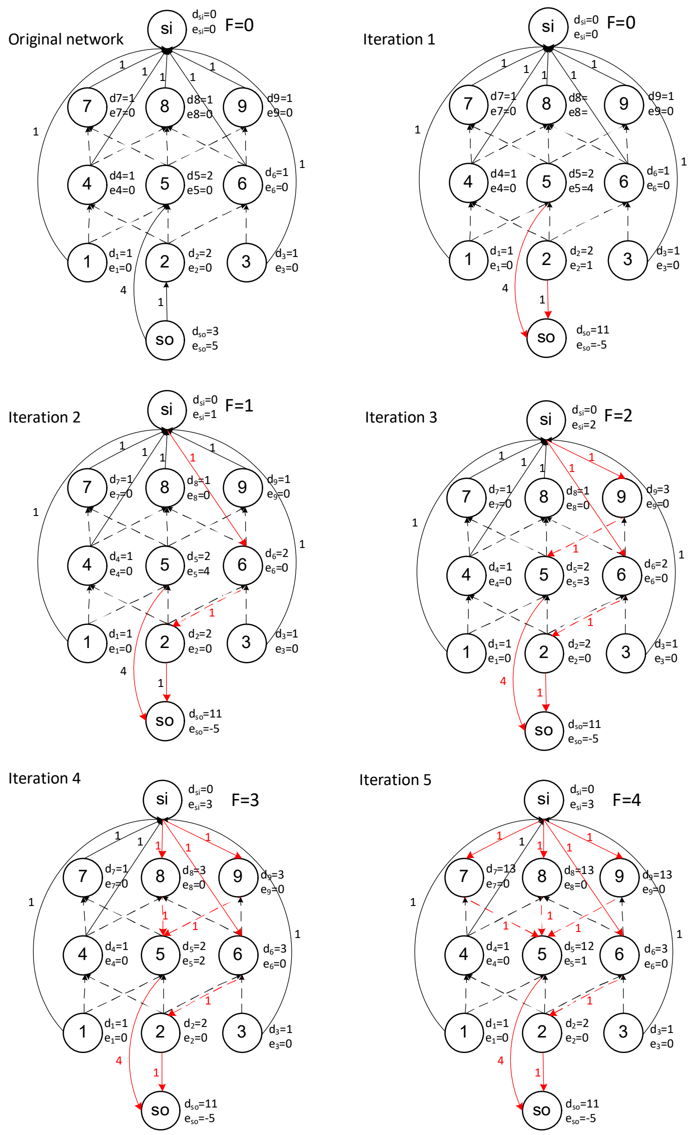


Figure 2. An example of determining the ultimate pit limit using the push–relabel algorithm.

4. HI-PR implementation of push–relabel algorithm

Although the generic push–relabel algorithm is an efficient algorithm, a major downside may affect its efficiency. This drawback can be explained by considering Figure 3. In this figure, after pushing flow from the source to vertices 1, 2, and 3, the set {1, 2, 3} is considered as a list of active vertices. According to the push–relabel algorithm with the first-in first-out (FIFO) policy, vertex 3 is selected as the first active vertex, and two units of flow are pushed to the sink (vertex s_i). As a result, vertex 3 is changed to an inactive vertex and omitted from the list of active vertices. Now, the set {1, 2} is a list of active vertices. Vertex 2 is selected, and two units of flow are pushed to vertex 3. Therefore, vertex 3 turns to an active vertex and is added to the back of the list. If the FIFO procedure continues for this example, it will be observed that vertex 3 is examined for three times, vertex 2 is examined twice, and therefore, the solution time increases.

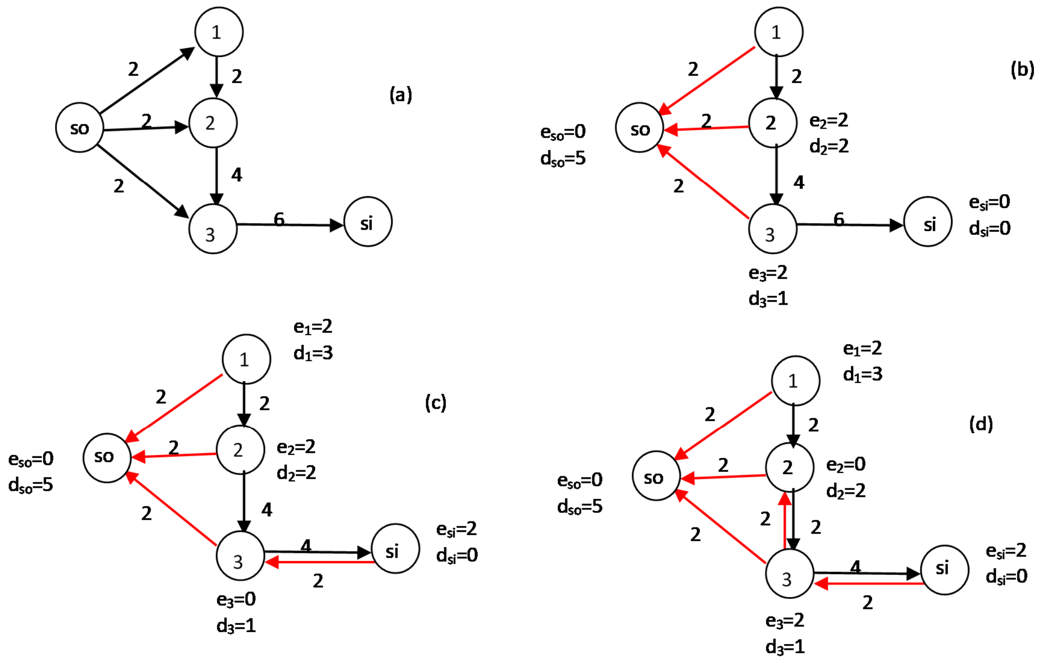


Figure 3. An example of a push–relabel algorithm and its computational inefficiency.

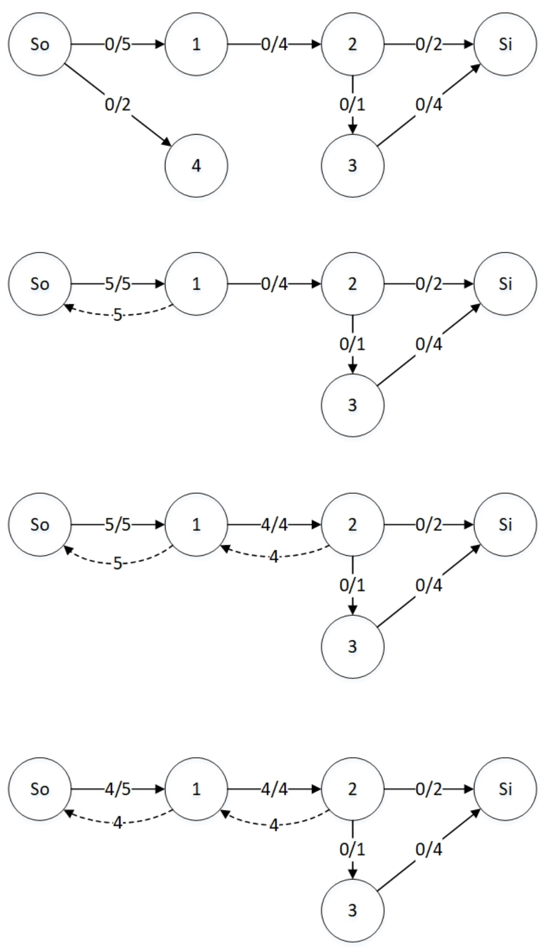
The HI-PR implementation of the push–relabel algorithm has been discussed by Cherkassky and Goldberg [31]. As reviewed by Goldberg [32], this algorithm takes advantage of the highest-label selection rule, global labeling, and gap heuristic benefit in order to speed-up the solution time. Local re-labeling labels each vertex locally, and it may cause the graph to lose its distance label picture. Global re-labeling calculates the exact vertex distance from the sink using a backward breath-first search. This could be done in a linear time, and compared with local re-labeling is computationally more expensive. Global relabeling should run periodically after k re-label operations. This operation strongly improves the running time.

Global re-labeling uses layers of the bucket data structure, which correspond to the vertex distance labels. Each layer contains the active and inactive buckets. A vertex v with $d(v) = i$ is in the active bucket α_i if $e_{f(v)} > 0$, and in bucket β_i for the inactive bucket if $e_{f(v)} \leq 0$. The highest-label selection rule requires to maintain the index μ of the highest layer with a non-empty active bucket. During a re-label operation, if an active vertex is inserted to layer i higher than the current value of μ, the index should increase to i [32].

At every step of the algorithm, α_μ must be examined. If it is empty, μ will be decreased, and if $\mu = 0$, the algorithm is terminated; otherwise, the first active vertex of α_μ extracts to v. If there is an admissible arc such as (v, w), the flow should be pushed along it. As a result of pushing the flow, $e_f(w)$ may be increased from zero, which makes w an active vertex. In this situation, w will be removed from $\beta_{d(w)}$ and inserted to $\alpha_{d(w)}$. In another situation, $e_f(w)$ may be decreased to zero, which makes v inactive. In this case, v will be removed from the head of $\alpha_{d(v)}$ and inserted to $\beta_{d(v)}$. If no admissible arc out of v exists, v will be re-labeled. This operation increases $d(v)$ and v will be extracted from the head of $\alpha_{d'(v)}$, where $d'(v)$ is the old distance label of v, and will be inserted to $\alpha_{d(v)}$. In this case, μ should be increased to $d(v)$. Then the algorithm will proceed to the next step [32].

The role of the gap heuristic is temporary to delete some of the vertices that cannot reach sink (si) in regard to the validity of d. Suppose for $0 < i < n$, there is no vertex with a distance label of i but some vertices w have distance labels $j: i < j < n$. The validity of d allows w to be temporarily deleted from the graph [32]. The layer modeling speeds-up the implementation of the gap heuristic. Global update places the remaining vertices in the appropriate buckets, and resets their current arcs to the corresponding first arcs.

Although it is difficult to perform the HI-PR implementation manually, a simple example is shown in **Error! Reference source not found.**.



BFS

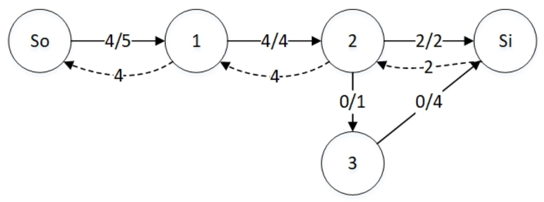
$$\alpha_0 = \{\} \quad \beta_0 = \{Si\}$$
$$\alpha_1 = \{\} \quad \beta_1 = \{2,3\}$$
$$\alpha_2 = \{\} \quad \beta_2 = \{1\}$$
$$\alpha_3 = \{\} \quad \beta_3 = \{So\}$$

$$\alpha_0 = \{\} \quad \beta_0 = \{Si\}$$
$$\alpha_1 = \{\} \quad \beta_1 = \{2,3\}$$
$$\alpha_2 = \{1\} \quad \beta_2 = \{\}$$
$$\alpha_3 = \{\} \quad \beta_3 = \{So\}$$
$$\mu = 2$$

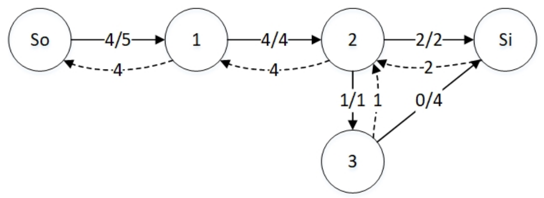
$$\alpha_0 = \{\} \quad \beta_0 = \{Si\}$$
$$\alpha_1 = \{2\} \quad \beta_1 = \{3\}$$
$$\alpha_2 = \{1\} \quad \beta_2 = \{\}$$
$$\alpha_3 = \{\} \quad \beta_3 = \{So\}$$
$$\mu = 2$$

$$\alpha_0 = \{\} \quad \beta_0 = \{Si\}$$
$$\alpha_1 = \{2\} \quad \beta_1 = \{3\}$$
$$\alpha_2 = \{\} \quad \beta_2 = \{\}$$
$$\alpha_3 = \{\} \quad \beta_3 = \{So\}$$
$$\alpha_4 = \{1\} \quad \beta_4 = \{\}$$
$$\mu = 4$$

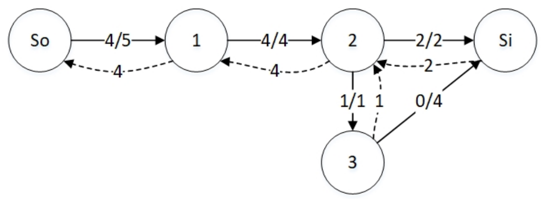
Figure 4. An example of an HI-PR implementation



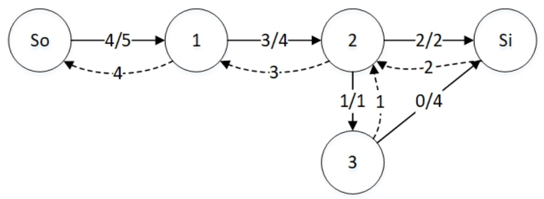
$\alpha_0 = \{\}$ $\beta_0 = \{Si\}$
$\alpha_1 = \{2\}$ $\beta_1 = \{3\}$
$\alpha_2 = \{\}$ $\beta_2 = \{\}$
$\alpha_3 = \{\}$ $\beta_3 = \{So\}$
$\alpha_4 = \{\}$ $\beta_4 = \{1\}$
$\mu = 1$



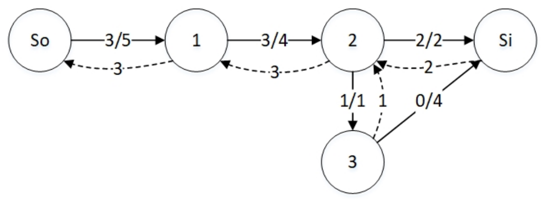
$\alpha_0 = \{\}$ $\beta_0 = \{Si\}$
$\alpha_1 = \{3\}$ $\beta_1 = \{\}$
$\alpha_2 = \{2\}$ $\beta_2 = \{\}$
$\alpha_3 = \{\}$ $\beta_3 = \{So\}$
$\alpha_4 = \{\}$ $\beta_4 = \{1\}$
$\mu = 2$

BFS


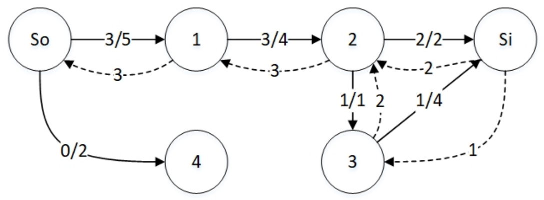
$\alpha_0 = \{\}$ $\beta_0 = \{Si\}$
$\alpha_1 = \{2,3\}$ $\beta_1 = \{\}$
$\alpha_2 = \{\}$ $\beta_2 = \{1\}$
$\alpha_3 = \{\}$ $\beta_3 = \{So\}$
$\mu = 1$



$\alpha_0 = \{\}$ $\beta_0 = \{Si\}$
$\alpha_1 = \{3\}$ $\beta_1 = \{\}$
$\alpha_2 = \{1\}$ $\beta_2 = \{\}$
$\alpha_3 = \{\}$ $\beta_3 = \{So,2\}$
$\mu = 2$



$\alpha_0 = \{\}$ $\beta_0 = \{Si\}$
$\alpha_1 = \{3\}$ $\beta_1 = \{\}$
$\alpha_2 = \{\}$ $\beta_2 = \{\}$
$\alpha_3 = \{\}$ $\beta_3 = \{So,2\}$
$\alpha_4 = \{\}$ $\beta_4 = \{1\}$
$\mu = 1$

BFS


$\alpha_0 = \{\}$ $\beta_0 = \{Si\}$
$\alpha_1 = \{\}$ $\beta_1 = \{2,3\}$
$\alpha_2 = \{\}$ $\beta_2 = \{1\}$
$\alpha_3 = \{\}$ $\beta_3 = \{So\}$
$\mu = 0$

Continues of Figure 4. An example of an HI-PR implementation.

According to the max-flow min-cut theorem, the maximum amount of flow passing from the source to the sink is equal to the total weight of the edges in the minimum cut; in other words, the smallest total weight of the edges that if removed, would disconnect the source from the sink. In order to determine the cut, source cut, and sink cut, after finalizing the max-flow algorithm, a depth first search (DFS) runs from the source in the residual graph and all vertices that can be reached from the source are source cut; otherwise, the sink cut and all edges that connect the source cut to the sink cut are then cut. The sink cut represents the ultimate pit limits.

5. Numerical experiments

In order to test the computational complexity of the HI-PR algorithm in solving the UPL problem, the proposed algorithm was applied to solve UPL for several real-life datasets. The datasets are available on the Minelib website [33], and they are publicly available as the test instances for three classical types of open-pit mine planning problems including the UPL problem and two other production scheduling problems. The specifications of the problem size including the number of decision variables (number of blocks) and constraints (number of precedence relationships) are presented in Table 1. These datasets are classified as small- to medium-size problems in the mining industry. Each dataset was solved using HI-PR as well as IBM ILOG CPLEX, version 12.6. CPLEX is a standard solver that employs a branch and bound algorithm in order to solve the integer programming problems. The computational times for both approaches are given in Table 2 and visualized in Figure 5. As it can be seen, the solution time of the CPLEX solver exponentially increases as the problem size increases. For the largest dataset with 112,687 blocks and 3,035,483 precedence relations, the CPLEX solver could not find a feasible solution in two hours, while using HI-PR, the optimum solution was achieved in less than four minutes. As expected, and seen in Figure 5, the solution time by CPLEX increases exponentially, while there is a slight linear incremental trend for HI-PR.

Table 2. Specifications of benchmark dataset and solution times.

Instance	Dataset name	Number of blocks	Number of precedences	CPLEX solution time (s)	HI-PR solution time (s)
1	Newman1	1060	3922	1.14	0.00
2	Zuck_small	9400	145640	11.98	0.11
3	Kd	14153	219778	18.80	0.18
4	Zuck_medium	29277	1271207	135.29	1.43
5	P4hd	40947	738609	65.67	0.61
6	Marvin	53271	650631	50.74	0.47
7	W23	74260	764786	75.87	0.71
8	Zuck_large	96821	1053105	145.95	0.97
9	Sm2	99014	96642	11.137	0.10
10	Mclaughlin_limit	112687	3035483	-	3.99

In addition, each case study was run 100 times to check the effect of the random element in the HI-PR algorithm on the solution times. The results obtained are summarized in Table 3. As seen, the maximum running time for the largest instance is 7.4 (worst case), which is still a reasonable time for such a medium-size problem.

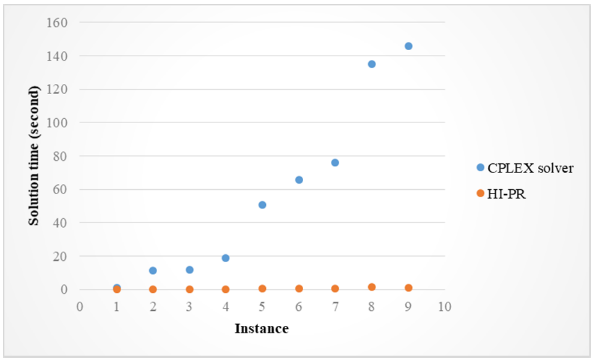


Figure 5. Solution time for HI-PR and CPLEX.

Table 3. Descriptive statistics of HI-PR solutions for 100 runs.

Instance	Block model	Average of run time (s)	Std (s)	Min (s)	Max (s)	#Blocks	#Precedence
1	Newman1	0.00	0.00	0.00	0.00	1060	3922
2	Zuck_small	0.12	0.03	0.08	0.22	9400	145640
3	Kd	0.19	0.04	0.13	0.29	14153	219778
4	Zuck_medium	1.43	0.30	1.03	2.27	29277	1271207
5	P4hd	0.61	0.08	0.48	0.93	40947	738609
6	Marvin	0.47	0.07	0.37	0.72	53271	650631
7	W23	0.72	0.09	0.55	1.05	74260	761453
8	Zuch_large	0.98	0.13	0.78	1.40	96821	1053105
9	Sm2	0.11	0.02	0.08	0.23	99014	96642
10	Mclaughlin_limit	3.99	0.72	2.51	7.41	112687	3035483

If an engineer wishes to design a mine according to a few possible scenarios, CPLEX or other polynomial-time algorithms such as a generic push–relabel algorithm can solve UPL for the optimum solution. However, regarding different sources of uncertainties in the mining industry, a common practice would be running multiple scenarios in order to evaluate the risk of the project, understanding best- and worst-case scenarios, and eventually, making the best decision. In a common case, one may generate 100 possible realizations of the orebody (due to grade uncertainty). In addition to grade, variation in the commodity prices may force the managers and designers to evaluate 100 simulations of the price. In such a situation, 10000 UPL problems must be solved. If the sensitivity analysis on other parameters such as unit cost of mining is also considered, then an even larger number of scenarios must be evaluated. For example, in another try-and-evaluate method, the mine production rate is also achieved by analyzing a range of production capacities to obtain the best production rate. Therefore, adding 10 more scenarios of production rate leads to 100000 possible scenarios. Solving UPL in such a situation and for a medium-size deposit (such as Zuck_large in Table 2 with 96821 blocks) by CPLEX takes 4054 h of computer running time, while the same situation takes only 27 h using HI-PR. This example clearly shows that the faster solution techniques are required, although UPL can be optimally solved by the other algorithms. Considering that in some mines the block models could be a few million blocks, a fast and reliable technique such as the HI-PR algorithm is a must in order to achieve a low-risk mine design and planning.

6. Conclusions

The UPL problem is a basic optimization problem in surface mining. UPL determines the final shape of the mine. Indeed, UPL selects the profitable area of mining, and all the other planning, scheduling, design, and equipment selection will be optimized based on the results of solving UPL. The UPL problem is a discrete optimization problem such that in large-size instances, a few million decision variables must be assigned values under several million constraints. In this work, the HI-PR algorithm was implemented in order to solve the UPL faster than the previous algorithms. The computational time was tested on several benchmark datasets and compared with the results of the exact solution by IMB ILOG CPLEX. The solution time for the largest dataset, named mclaughlin_limit with 112,267 decision variables and 3,035,483 constraints, is 4 s on average. In addition, in order to verify the robustness of the proposed algorithm, each one of the case studies was solved 100 times. The maximum solution time was 7.4 s for the most extensive dataset. Thus the results clearly demonstrate the strength of the HI-PR algorithm in achieving the optimum solution in a short period of time. This verifies that this algorithm would be very useful when the solution of hundreds of thousands of scenarios is required or when changes are frequently happening in the input and technical parameters. As the HI-PR algorithm is a polynomial algorithm, a solution can be found in a reasonable time period even if the size of the problem is large. This is the strength of this algorithm, which can be highly beneficial when the ore deposit is very large or the block size is small and sub-cells are included in the block model. However, the same as the other exact algorithms for the UPL problem, HI-PR is not flexible to include extra constraints (other than the precedence constraints). A future research work will focus on implementing this algorithm in order to solve the next steps of the long-term and short-term production scheduling in a real-time manner. Moreover, this algorithm can be implemented in the simultaneous optimization of open-pit and underground mine design and planning to optimize the open-pit to the underground transition time.

References

[1]. Mousavi, A., Sayadi, A.R., and Fathianpour, N. (2016). A comparative study of kriging and simulation-based methods in classifying ore and waste blocks. Arabian Journal of Geosciences, 9(17): p. 691-700.

[2]. Bakhtavar, E., Abdollahisharif, J., and Aminzadeh, A. (2017). A stochastic mathematical model for determination of transition time in the non-simultaneous case of surface and underground mining. Journal of the Southern African Institute of Mining and Metallurgy, 117 (12): p. 1145-1153.

[3]. Lerchs, H. and Grossmann, I. (1965). Optimum design of open pit mines. Transaction on CIM, LX VIII: p. 17-24.

[4]. Mousavi, A., (2015). Optimisation of open pit mine block sequencing. Queensland University of Technology: Australia.

[5]. Zhao, Y. and Kim, Y.C. (1992). A new optimum pit limit design algorithm, in Proceedings of 23rd APCOP. Society for Mining, Metallurgy and Exploration: Littleton, Colorado. p. 423-434.

[6]. Khalokakaie, R., Dowd, P.A., and Fowell, R.J. (2000). Lerchs–Grossmann algorithm with variable slope angles. Mining Technology, 109 (2): p. 77-85.

[7]. Giannini, L., (1990). Optimum design of open pit mines. Curtin University of Technology: Perth. p. 166.

[8]. Underwood, R. and Tolwinski, B. (1998). A mathematical programming viewpoint for solving the ultimate pit problem. European Journal of Operational Research, 107 (1): p. 96-107.

[9]. Hochbaum, D.S. and Chen, A. (2000). Performance analysis and best implementations of old and new algorithms for the open pit mining problem. Operation research, 48 (6): p. 894-914.

[10]. Koenigsberg, E., (1982). The Optimum Contours of an Open Pit Mine: an Application of Dynamic Programming, in Applications of Computers and Operations Research in the Mineral Industry (17th APCOM). New York. p. 274-287.

[11]. Erarslan, K. and Celebi, N. (2001). A simulative model for optimum open pit design. CIM BULLETIN, 94(1055): p. 59-68.

[12]. Najafi, M., Rafiee, R., and Jalali, S.M.E. (2020). Open pit limit optimization using dijkstra's algorithm. International Journal of Mining and Geo-Engineering, 54 (1): p. 39-43.

[13]. Pana, M.T., (1965). The simulation approach to open pit design, in Proceedings of the 5th APCOM Tucson, AZ. p. 139-144.

[14]. Wright, A. (1999). A simple algorithm for optimum pit limits design, in Proceedings of the 28rd APCOM, Dagdelen, K., et al., Editors. Colorado School of Mines: Golden, Colorado. p. 367-374.

[15]. Achireko, P., (1998). Application of modified conditional simulation and artificial neural networks to open pit optimization. Dalhousie University: Nova Scotia.

[16]. Frimpon, S. and Achireko, P. (1997). The MCS/MFNN algorithm for open pit optimization. International Journal of Surface Mining, Reclamation and Environment, 11 (1): p. 45-52.

[17]. Sayadi, A.R., Fathianpour, N., and Mousavi, A.A. (2011). Open pit optimization in 3D using a new artificial neural network. Archives of Mining Sciences, 56 (3): p. 389–403.

[18]. Espejo, N., Nancel-Penard, P., and Morales, N. (2020). A methodology for automatic ramp design in open pit mines. Journal of Mining Engineering and Research, 1 (2).

[19]. Bakhtavar, E., Shahriar, K., and Mirhassani, A. (2012). Optimization of the transition from open-pit to underground operation in combined mining using (0-1) integer programming. Journal of the Southern African Institute of Mining and Metallurgy, 112 (12): p. 1059-1064.

[20]. Mergani, H., Osanloo, M., and Parichehp, M., (2019). Ultimate Pit Limit Determination Considering Mining Royalty in Open-Pit Copper Mines, in International Symposium on Mine Planning & Equipment Selection. Springer. p. 346-358.

[21]. Hay, E., Nehring, M., Knights, P., and Kizil, M. (2019). Ultimate pit limit determination for semi mobile in-pit crushing and conveying system: a case study. International Journal of Mining, Reclamation and Environment: p. 1-21.

[22]. Hay, E., Nehring, M., Knights, P., and Kizil, M.S. (2019). Ultimate pit limit determination for fully mobile in-pit crushing and conveying systems. International Journal of Mining and Mineral Engineering, 10 (2-4): p. 111-130.

[23]. Meyer, M. (1969). Applying linear programming to the design of ultimate pit limits. Management Science, 16(2): p. B-121-B-135.

[24]. Johnson, T.B. and Sharp, W.R., (1971). A Three-dimensional dynamic programming method for optimal ultimate open pit design. Vol. 7553. Bureau of Mines, US Department of the Interior.

[25]. Frimpong, S. and Achireko, P.K. (1998). Conditional LAS stochastic simulation of regionalized variables in random fields. Computational Geosciences, 2 (1): p. 37-45.

[26]. Erarslan, K. and Celebi, N. (2001). A simulative model for optimum open pit design. CIM Bulletin, 94: p. 59–68.

[27]. Frimpong, S., Szymanski, J., and Narsing, A. (2002). A computational intelligent algorithm for surface mine layouts optimization. Simulation, 78(10): p. 600-611.

[28]. Elahi, E., Kakaie, R., and Yusefi, A. (2012). A new algorithm for optimum open pit design: Floating cone method III. Journal of Mining and environment, 2 (2): p. 118-125.

[29]. Wright, A., (1990). Open pit mine design model: introduction with Fortran 77 programs. Clausthal-Zellerfeld: Trans Tech Publications.

[30]. Ahuja, R.K., Magnanti, T.L., and Orlin, J.B. (1993). Network flows : theory, algorithms, and applications. Prentice Hall.

[31]. Cherkassky, B.V. and Goldberg, A.V. (1995). On implementing push-relabel method for the maximum flow problem, in International Conference on Integer Programming and Combinatorial Optimization. Springer. p. 157-171.

[32]. Goldberg, A.V. (2009). Two-level push-relabel algorithm for the maximum flow problem, in International Conference on Algorithmic Applications in Management. Springer. p. 212-225.

[33]. Espinoza, D., Goycoolea, M., Moreno, E., and Newman, A. (2013). MineLib: a library of open pit mining problems. Annals of operations research, 206 (1): p. 93-114.

پیاده سازی بالاترین-سطح الگوریتم ارسال-برچسب برای حل مسئله محدوده نهایی معدن

مهدی طلایی، امین اله موسوی[*] و احمد رضا صیادی

گروه مهندسی معدن، انشکده‌ی مهندسی، دانشگاه تربیت مدرس، تهران، ایران

ارسال ۲۰۲۱/۰۱/۱۹، پذیرش ۲۰۲۱/۰۳/۱۵

* نویسنده مسئول مکاتبات: a_mousavi@modares.ac.ir

چکیده:

امروزه و با در نظر گرفتن عدم قطعیت‌های اقتصادی و زمین شناسی و همچنین افزایش استفاده از روش ارزیابی سناریو-محور در طراحی معادن روباز، ضروری است تا الگوریتم دقیق و سریعی برای حل مسئله محدوده نهایی معدن استفاده شود. تعیین محدوده نهایی معادن روباز یک مسئله بهینه‌سازی مهم است که جواب آن تعیین کننده میزان نهایی استخراج سنگ از معدن است و به طور مستقیم بر هزینه‌های معدنکاری، درآمد، انتخاب ماشین آلات و موقعیت زیرساخت‌های معدن تاثیر می‌گذارد. این مسئله با هدف بیشینه‌سازی سود غیرتنزیلی و تحت محدودیت پیش نیازی (محدودیت دسترسی) حل می‌شود. در این مقاله، از پیاده سازی بالاترین-سطح الگوریتم ارسال-برچسب برای حل مسئله محدوده نهایی معدن استفاده می‌شود. این الگوریتم از قواعد انتخاب بالاترین برچسب، به روز رسانی کلی و الگوریتم‌های ابتکاری برای کاهش زمان حل مسئله استفاده می‌کند. الگوریتم پیشنهادی، برای حل ۹ مطالعه موردی که داده‌های آن‌ها به صورت عمومی در دسترس هستند، به کار گرفته شد. نتایج نشان داد که زمان حل این الگوریتم نسبت به الگوریتم‌های دیگر کمتر می‌باشد. برای پیچیده‌ترین مطالعه موردی با ۱۱۲۶۸۷ بلوک و ۳۰۳۵۴۸۳ محدودیت، متوسط زمان حل برای ۱۰۰ تکرار کمتر از ۴ ثانیه می‌باشد. این در حالی است که برای این مورد، نرم افزار IBM CPLEX جوابی را در مدت ۲۴ ساعت پیدا نکرد. این بالا بردن سرعت می‌تواند چالش زمان حل طولانی را در برنامه‌ریزی‌های برخط برطرف کند.

کلمات کلیدی: تئوری گراف، ارسال-برچسب، جریان بیشینه، معدن روباز، HI-PR.