Vol. 1, No. 2, 2006, pp. 7-20

دوره اول، شماره ۲، سال ۱۳۸۵، صفحه ۷۳ تا ۸۶

Iranian Journal of Mining Engineering
(IRJME)

نشریه علمی-پژوهشی "مهندسی معدن"

THE *MVN* MULTIPLE PASS ALGORITHM FOR OPTIMISATION OF STOPE BOUNDARIES

Majid Ataee-pour[1]

[1]- Amirkabir University of Technology, Tehran, Iran
(map60@aut.ac.ir)

ABSTRACT

There are few algorithms, developed for optimisation of ultimate stope limits. These are either heuristic or rigorous. The rigorous algorithms such as the application of the branch and bound technique and dynamic programming approach do not apply on 3D problems. Heuristic algorithms such as floating stope of Datamine and the Maximum Value Neighbourhood (*MVN*) algorithm do not guarantee the true optimum solution; they only provide a solution, which is close to the optimum one. The *MVN* algorithm runs on a fixed 3D economic block model of the ore-body and forms the best neighbourhood of each block in the model. The algorithm starts from the first block to form its *MVN* and include it into the ultimate limit. Then it proceeds to the last block to build up the small *MVN* as largely as possible. However, since it is a heuristic approach, the final stope limit obtained may contain a number of unnecessary waste blocks, ie they may be excluded from the ultimate stope without violating the stope constraints. In addition, there may be a number of necessary ore blocks, excluded from the ultimate stope. This paper introduces the extension of the *MVN* algorithm to run a multiple pass and check if it is possible to remove waste blocks from the ultimate stope and add other ore blocks to it. Running the multiple pass makes the ultimate stope limits as close as possible to the optimum one.

Keywords: Optimisation, Stope boundaries, Algorithm, Neighbourhood, Multiple pass

1. INTRODUCTION

Generally, current algorithms, developed for optimisation of mining limits are implemented on a block model of the orebody. These algorithms are either rigorous or heuristic. The rigorous algorithms are supported by mathematical proof and hence, they guarantee the true optimum solution of the mining limits, for the level they are applied. In open pit cases, for example, the two dimensional dynamic programming (DP) algorithm [1] is a rigorous algorithm that guarantees the true optimum pit limits in two dimensions. In contrast, the moving cone (MC) technique [2] is a heuristic algorithm that provides a solution, which is not necessarily the optimum one, although it is very close to the optimum.

For underground cases, there are few algorithms available for optimisation of the stope layout. The rigorous algorithms include the application of dynamic programming technique in 2D problems [3] and the use of Branch and Bound technique in 1D problems [4]. These algorithms fail to provide 3D analysis and/or they are tailored for specific mining methods. Recently, a new application of the DP technique was suggested [5]; however, it is useful for vein type deposits and does not comply with 3D cases. Heuristic approaches, mainly, include the floating stope of Datamine [6] and the *Maximum Value Neighbourhood (MVN)* algorithm [7].

Algorithms, usually, apply a recursive operation over the blocks of the model to find the optimum limits. For heuristic algorithms, this process is a search technique, which strongly depends on the search direction. This paper examines the *MVN* algorithm, illustrates its failure to provide the true optimum solution, studies the influence of the search order, explores the main causes of the problem and suggests a second pass over the block model as a modification to improve the algorithm performance.

2. THE *MVN* ALGORITHM

The neighbourhood *(NB)* concept was first introduced as a basis for optimisation of the 3D stope layout [7]. It was then developed using the neighbourhood concept and economic factors, which determine the blocks economic values [8]. A non-commercial software tool, called *SLO*, has been developed for implementation of the algorithm [9].

The *MVN* algorithm uses a fixed economic block model of an ore-body and searches for the best combination of blocks to provide a maximum profit while imposing certain geo-technical and mining constraints, eg the minimum stope geometry. The minimum size of the stope must ensure that a sufficient space is provided for activities of drilling, blasting and loading equipment, as well as movement of personnel and machinery in the stope. The *NB* concept formulates a minimum stope size in terms of neighbourhood factor for each block. The set of sequential blocks that could be mined to satisfy mining constraints defines the neighbourhood for a given block. The size of this set is called the *"order of neighbourhood" (O_{nb})*. Fig. 1 shows examples of 1D neighbourhood. An example of 3D neighbourhoods is illustrated in Fig. 2.

In order to locate the optimum neighbourhood of a block, the economic value of each neighbourhood has to be calculated and compared with one another. The term neighbourhood value *(NBV)* represents the net value of the neighbourhood if all its blocks are extracted as a set. For each block, the NB with the highest value is considered the maximum NBV and included into the stope, ie its members are flagged "1". The process of locating and flagging the *MVN* for a block with a neighbourhood order of 2 is illustrated in Fig. 3.

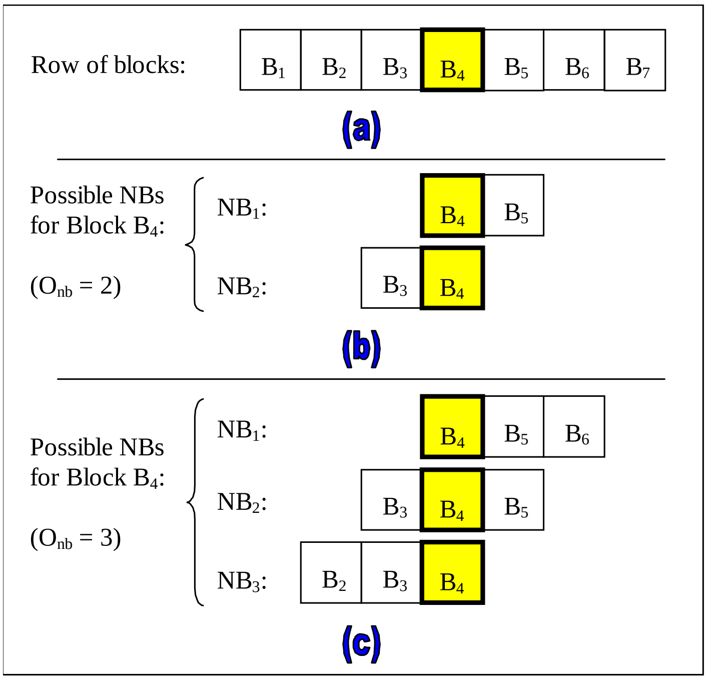

Row of blocks:
B_1 B_2 B_3 B_4 B_5 B_6 B_7
(a)
Possible NBs for Block B_4:
(O_{nb} = 2)
NB_1: B_4 B_5
NB_2: B_3 B_4
(b)
Possible NBs for Block B_4:
(O_{nb} = 3)
NB_1: B_4 B_5 B_6
NB_2: B_3 B_4 B_5
NB_3: B_2 B_3 B_4
(c)

Fig. 1: Possible neighbourhoods of the Block B_4, for *NB* orders of 2 and 3

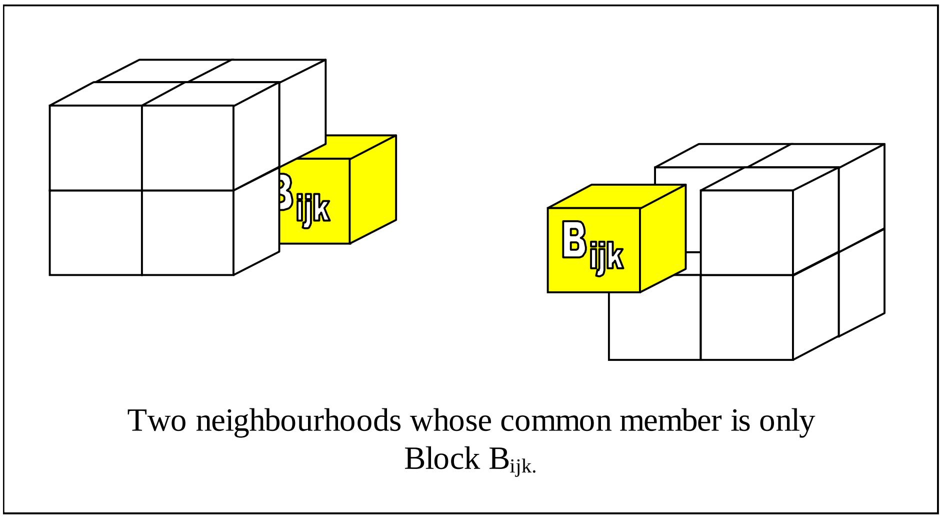


Fig. 2: An example of 3D neighbourhoods (O_{nb} = 2×2×2)

3. INFLUENCE OF SEARCH DIRECTION

Algorithms, usually, apply a recursive operation on blocks of the model to find the optimum limits. These blocks are normally taken into consideration in special order, eg from left to right or vice versa. Rigorous algorithms are independent of the direction of the search as they use mathematical formulations. So, the pit limit defined by using the DP algorithm is unique, regardless of applying the algorithm from left to right or from right to left. However, heuristic algorithms such as moving cone in open pit cases are search-based and hence, the block order or the search direction has a great impact on the results, ie the pit limit defined by applying the algorithm from left to right may differ from that of applying the algorithm from right to left.

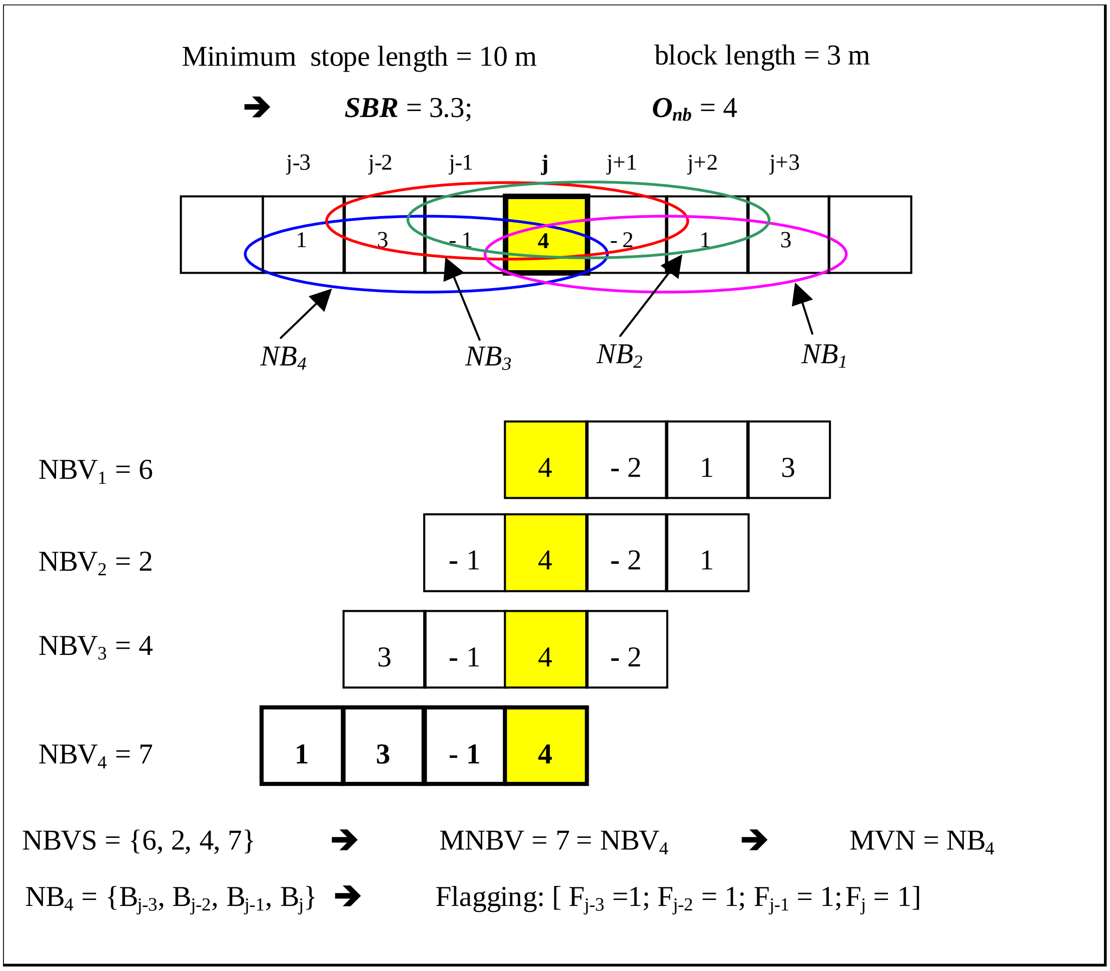

Minimum stope length = 10 m
block length = 3 m
➔ SBR = 3.3;
Onb = 4
j-3
j-2
j-1
j
j+1
j+2
j+3
1
3
- 1
4
- 2
1
3
NB4
NB3
NB2
NB1
NBV1 = 6
4
- 2
1
3
NBV2 = 2
- 1
4
- 2
1
NBV3 = 4
3
- 1
4
- 2
NBV4 = 7
1
3
- 1
4
NBVS = {6, 2, 4, 7} ➔ MNBV = 7 = NBV4 ➔ MVN = NB4
NB4 = {Bj-3, Bj-2, Bj-1, Bj} ➔ Flagging: [Fj-3 =1; Fj-2 = 1; Fj-1 = 1; Fj = 1]

Fig. 3: Locating the Maximum Value Neighbourhood

In underground cases, the floating stope technique and the *MVN* algorithm are based on a heuristic approach; therefore, the results are dependent on the order of blocks, on which these algorithm are implemented. The *MVN* algorithm is implemented on all non-negative blocks. Although it is not necessary to take blocks into consideration in any special order, for the sake of simplicity, the algorithm takes them in the order of rows, columns and sections, in their positive directions, as it provides the most convenient way to search for blocks. If the algorithm is applied through the positive direction, the optimised ultimate stope is not necessarily similar to that obtained when applying it through the negative direction. Consider a simple model section with five rows and 12 columns, as shown in Fig. 4. Assume a one dimensional constraint of three blocks as the minimum stope length. Applying the *MVN* algorithm once from left to right and then vice versa yield in similar results for rows 1, 3 and 4 but different results for rows 2 and 5.

As a result, both solutions are non-optimum. This situation does not necessarily occur all the time. In many cases, the results may coincide to the true optimum; however, the key point is that the true optimum solution is not guaranteed.

i \ j→	1	2	3	4	5	6	7	8	9	10	11	12
1	2	1	-1	0	3	2	-2	4	1	2	-2	-1
2	5	-1	-1	2	3	-2	1	0	1	3	-1	-1
3	3	0	4	1	-2	-1	0	-1	2	1	-1	2
4	-1	0	-2	1	2	0	4	-1	1	2	2	-1
5	6	-1	-3	1	0	-3	5	3	1	-4	2	0

(a)

i \ j→	1	2	3	4	5	6	7	8	9	10	11	12
1	2	1	-1	0	3	2	-2	4	1	2	-2	-1
2	5	-1	-1	2	3	-2	1	0	1	3	-1	-1
3	3	0	4	1	-2	-1	0	-1	2	1	-1	2
4	-1	0	-2	1	2	0	4	-1	1	2	2	-1
5	6	-1	-3	1	0	-3	5	3	1	-4	2	0

(b)

Fig. 4: Application of the *MVN* algorithm a) from left to right b) from right to left

3.1 Causes of the Problem

This problem may occur, at least, in two situations, ie tie cases and negative marginal values, as explained below.

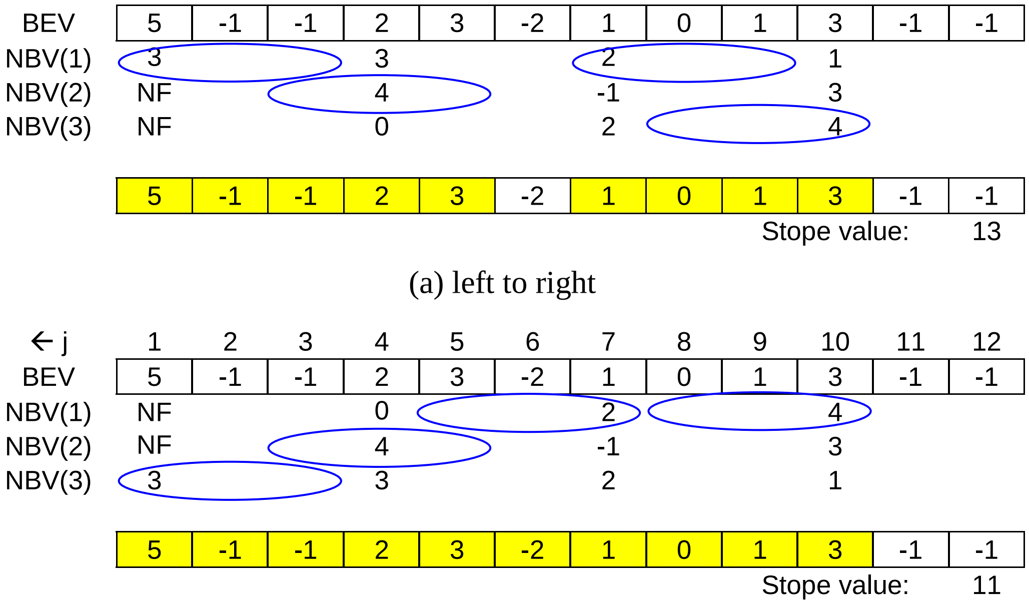
Tie cases: Through the process of defining the maximum neighbourhood value (MNBV), there may be two or more neighbourhoods with the same and maximum net values. So, there is a tie for the algorithm to decide. Normally, in tie cases, the first true condition (the maximum net value here) is selected and the procedure continues, ignoring the other true condition. However, defining the **first** true condition strongly depends on the direction of the search. In fact, what is the **first**, when applying the algorithm from left to right, is exactly the **last**, when applying the algorithm in the opposite direction. This will cause inclusion of different neighbourhoods (ie a set of blocks) to the ultimate limits. This difference (error) may or may not be covered by the *MVN* of the next blocks, as the algorithm proceeds. If the difference is not covered later, the final results will be different for two directions. One may decide to accept both (or all) true conditions in tie cases to solve the problem, but it should be noted that the economic constraint (maximum profit) may be violated due to the inclusion of less valuable blocks.

Negative marginal values: The marginal value of the maximum value neighbourhood *(MVN)* is evaluated to determine its contribution to the final stope. It is the real difference that the inclusion of the *MVN* will make in the stope value. The marginal value of an *MVN* is defined by the total value of those elements of the *MVN* that are new to the final stope, and contribute to the stope value when considering the current block. In essence, the marginal value is defined as elements of an *MVN* that are not flagged already. Negative marginal values, which cause a decrease in the stope value, may occur when valuable elements of the current *MVN* have been flagged earlier and the costly elements are new to the stope.

The above fact may influence the results of the algorithm, when applying it in different directions. In one direction, the *MVN* of a block may provide a negative marginal value. So, the algorithm proceeds to take the next block into consideration without including the elements of that *MVN* into the final stope. This may or may not be covered by considering *MVN*s of next blocks, as the algorithm proceeds. This situation may or may not happen when applying the algorithm in the opposite direction.

3.2 A One Dimensional Example

In order to shed light on the issue, Rows 2 and 5 of the example shown in Fig.1, which give different results in different search directions are examined here. Details of the application of the *MVN* algorithm on the second row are illustrated in Fig. 5, in both positive and negative directions. The block economic values (*BEV*) are shown inside each cell of the model. Since the stope length is limited to a minimum of three blocks, the order of neighbourhood is 3 and hence there are three possible neighbourhoods for each block to compare. Neighbourhood values (*NBV*) for each block is shown below the block in three lines. The *MVN* of each block is also shown with an ellipse drawn beneath the elements of that neighbourhood. An "NF" sign has been used for non-feasible neighbourhoods. If a block value is negative or the block is already flagged, the block is skipped.



BEV	5	-1	-1	2	3	-2	1	0	1	3	-1	-1
NBV(1)	3			3			2			1		
NBV(2)	NF			4			-1			3		
NBV(3)	NF			0			2			4		
	5	-1	-1	2	3	-2	1	0	1	3	-1	-1
										Stope value:		13

(a) left to right

← j	1	2	3	4	5	6	7	8	9	10	11	12
BEV	5	-1	-1	2	3	-2	1	0	1	3	-1	-1
NBV(1)	NF			0			2			4		
NBV(2)	NF			4			-1			3		
NBV(3)	3			3			2			1		
	5	-1	-1	2	3	-2	1	0	1	3	-1	-1
										Stope value:		11

(b) right to left

Fig. 5: Different results of the *MVN* algorithm due to a tie occurrence

Fig. 5a shows the application of the algorithm from left to right. As illustrated, the block B_7 with the net value of (1) has two equal neighbourhood values, *NBV(1)* and *NBV(3)*, which has the maximum net value among the three neighbourhoods.

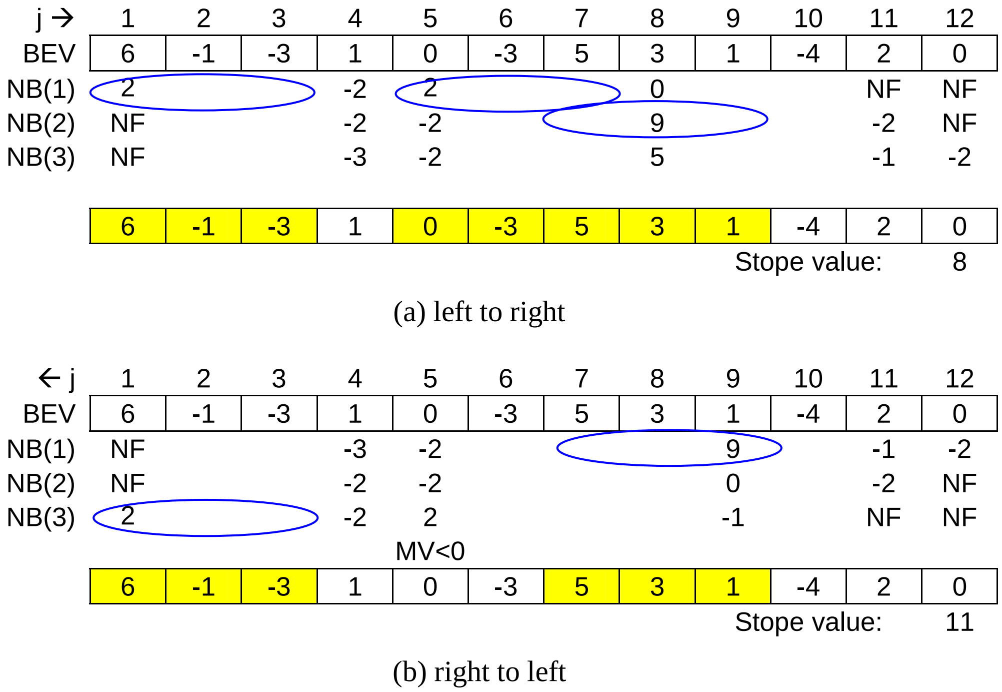
```
NB(1) = {(1), (0), (1)} ➔ NBV(1) = 2
NB(2) = {(-2), (1), (0)}➔ NBV(2) = -1
NB(3) = {(3), (-2), (1)}➔ NBV(3) = 2
```

The algorithm takes the first true condition and includes B_7, B_8 and B_9 to the final stope. In this case, B_6 with the value of (-2) is left un-mined. The final stope will then have two parts, one extended from B_1 to B_5 and the other extended from B_7 to B_{10}.

Fig. 5b shows the application of the algorithm from right to left. In this case, neighbourhoods of the same block, B_7, are located in the opposite direction, ie:

```
NB(1) = {(1), (-2), (3)}➔ NBV(1) = 2
NB(2) = {(0), (1), (-2)}➔ NBV(2) = -1
NB(3) = {(1), (0), (1)} ➔ NBV(3) = 2
```

The algorithm takes the first true condition and includes B_7, B_6 and B_5 to the final stope. The final stope will then be extended from B_1 to B_{10}. Details of the application of the *MVN* algorithm on Row 5 are illustrated in Fig. 6, for both positive and negative directions.



j →	1	2	3	4	5	6	7	8	9	10	11	12
BEV	6	-1	-3	1	0	-3	5	3	1	-4	2	0
NB(1)	2			-2	2			0			NF	NF
NB(2)	NF			-2	-2			9			-2	NF
NB(3)	NF			-3	-2			5			-1	-2
	6	-1	-3	1	0	-3	5	3	1	-4	2	0
									Stope value:			8

(a) left to right

← j	1	2	3	4	5	6	7	8	9	10	11	12
BEV	6	-1	-3	1	0	-3	5	3	1	-4	2	0
NB(1)	NF			-3	-2				9		-1	-2
NB(2)	NF			-2	-2				0		-2	NF
NB(3)	2			-2	2				-1		NF	NF
					MV<0							
	6	-1	-3	1	0	-3	5	3	1	-4	2	0
									Stope value:			11

(b) right to left

Fig. 6: Different results of the *MVN* algorithm due to the negative marginal value

Fig. 6.a shows the application of the algorithm from left to right. As illustrated, the block, B_5 with the net value of (0) has its first neighbourhood (ie {B_5, B_6, B_7}) with the value of (2) as its *MVN*. So the algorithm includes blocks B_5, B_6 and B_7 to the final stope. In other words, although B_5 is zero and B_6 is negative, they are included because B_7 pays for their cost. When applying the algorithm from right to left (Fig. 6.b), the valuable block, B_7, is flagged and included in the stope before processing B_5 since it is an element of the *MVN* of B_9, which was processed earlier. The *MVN* of B_5 is its last neighbourhood (ie {B_7, B_6, B_5}). The valuable block B_7 is already flagged and a care should be taken not to include it twice in the ultimate stope. Therefore, only costly blocks B_5 and B_6 are subject to flag and inclusion in the final stope. This means that the marginal value of the *MVN* of B_5 is the cumulative value of B_6 and B_7, which are new to the stope. Due to the negative marginal value, the algorithm rejects including the *MVN*. This difference between application of the algorithm on opposite directions yields to different ultimate stope.

3.3 A Two Dimensional Example

Now consider an example with a 2D neighbourhood, shown in Fig. 7. The model consists of six rows and eight columns. The neighbourhood is assumed to be 2 × 2, ie the stope should be at least two blocks long and two blocks wide. The blocks included into the final stope, as the optimised, are shaded.

As Fig. 7 shows the crucial blocks are B_{17}, B_{27}, B_{33} and B_{64}. The first three blocks have made problems due to various marginal values in different directions and the fourth block due to a tie case. Possible neighbourhoods for B_{33} are shown in Fig. 8, regardless of their numbers since the neighbourhood numbering depends on the search direction. As Fig. 8 shows, the neighbourhood with the value of (2) has the maximum neighbourhood value (MNBV). When applying the algorithm from left, it is found that the left column of the neighbourhood is flagged already, shown in Fig. 9 as shaded blocks. Therefore, only blocks located in the right column are new to the stope, which make a marginal value of (-1). Due to the negative marginal value (MV), the MVN of the block is ignored and hence, blocks of the right column (B_{33} and B_{43}) are not included into the stope, at this stage. The lower block of the right column, B_{43}, has been further included to satisfy the constraints of neighbourhoods of blocks located at next row.

	1	2	3	4	5	6	7	8
1	1	0	1	-2	3	1	0	2
2	2	-1	2	1	1	1	-2	-1
3	1	-1	0	-2	2	1	-1	1
4	2	4	-1	3	1	-1	2	3
5	-1	2	1	3	-1	2	0	1
6	2	1	0	-2	3	-1	4	1

a) Applying the algorithm from left to right

	1	2	3	4	5	6	7	8
1	1	0	1	-2	3	1	0	2
2	2	-1	2	1	1	1	-2	-1
3	1	-1	0	-2	2	1	-1	1
4	2	4	-1	3	1	-1	2	3
5	-1	2	1	3	-1	2	0	1
6	2	1	0	-2	3	-1	4	1

b) Applying the algorithm from right to left

Fig. 7: A 2D example with a neighbourhood of 2 × 2

0	-2
-1	3

NBV = 0

-1	0
4	-1

(Max) NBV = 2

2	1
0	-2

NBV = 1

-1	2
-1	0

NBV = 0

Fig. 8: Neighbourhoods of *B_{33}*

-1	0
4	-1

→→→

MNBV = 2

MV = -1

-1	0
4	-1

←←←

MNBV = 2

MV = 2

Fig. 9: Various marginal values for *B_{33}* in opposite directions

When applying the algorithm from right to left, all elements of the *MVN* of the block are new to the stope and hence, the marginal value would be (2), the same as its MNBV. Therefore, due to the non-negative marginal value, all elements of the *MVN* are included into the final stope and the stope value is updated.

Difference in inclusion or exclusion of B_{64} has been made by a tie occurrence when examining B_{65}. There are two feasible neighbourhoods for B_{65}, as shown in Fig. 10. So, the first true condition is different for opposite search directions, which has influenced on exclusion of the B_{64} element. The examining block is made bold and underlined to be distinguished.

3	-1
-2	**3**

NBV = 3

-1	2
3	-1

NBV = 3

Fig. 10: The tie occurrence for B_{65}

4. MULTIPLE PASS

It is known from the above that the optimised stope applying the *MVN* algorithm may include some negatively valued blocks, which are not necessary in the ultimate stope. That is, the mining constraints are not violated if these blocks are removed from the ultimate stope. In addition, some zero or positively valued blocks may be found excluded from the ultimate stope, which may be added without violation of constraints. A supplement to the *MVN* algorithm is introduced in this paper to improve the optimisation results and make the ultimate stope as close as possible to the true optimum.

After conducting the first pass on blocks applying the *MVN* algorithm, a second pass is run to check possibility of including those non-negatively valued blocks, which have not been included into the ultimate stope through the fisrt (general) pass. After completion of this stage, another pass is run to check possibility of excluding those negatively valued blocks from the final stope. The supplement is called the *MVN* Multiple Pass algorithm and is consisted of two parts, as described below.

4.1 Checking Inclusion of Non-negative blocks

A non-negative block, which is not included into the ultimate stope, may have a negative MNBV or a negative MV. In any case, after completion of the first pass, if it is re-examined, it might be included due to the possible changes made to the flags of its neighbours. The supplement algorithm suggests forming bridging blocks and checking the bridge for non-negativity. The *bridge* of the block is defined as the set of blocks required to join the re-examined block to the ultimate stope (including the block itself) while satisfying the stope size constraints. The bridge would, obviously, be a subset of one of the block's neighbourhoods. In other words, the bridge is consisted of the marginal blocks of a neighbourhood of the block. So, the proposed algorithm 1) constructs all neighbourhoods of the block, 2) computes the MV of each neighbourhood, 3) selects the one with the maximum MV and 4) adds all the marginal blocks to the ultimate stope if they contribute non-negatively to the stope (ie if MV ≥ 0). The flowchart for checking possibility of inclusion of non-negative blocks is illustrated in Fig. 11.

4.2 Checking Exclusion of Negative Blocks

The ultimate stope, normally, includes some negatively valued blocks. These have been included to satisfy the minimum stope size. Due to overlapping *MVN*s, some of these blocks may no longer be required if all *MVN*s are examined. In other words, overlapping *MVN*s may help each other to satisfy the constraints and hence, avoid inclusion of some waste blocks. This will be known after completion of the general (first) pass.

A waste block may be excluded from the stope if when excluded, the stope constraints are not violated. For example, in a one dimensional constraint, a waste block may be flagged off if leaving it un-mined, two separate stopes satisfying the minimum size are generated on both sides of the waste block. Therefore, if the order of neighbourhood is three blocks through the stope length, a waste block is taken out of the stope, if three consecutive blocks on the right and three consecutive blocks on the left of that waste block are all flagged. Generally speaking, the following set of conditions should be satisfied to exclude a waste block, B_{ijk}:

$$\begin{cases} a)\text{ To make left stope :} \\ F_{i-1,j,k} = F_{i-2,j,k} = \ldots = F_{i-\lambda,j,k} = 1 \\ b)\text{ To make right stope :} \\ F_{i+1,j,k} = F_{i+2,j,k} = \ldots = F_{i+\lambda,j,k} = 1 \end{cases} \qquad (1)$$

where λ is the order of neighbourhood in the specified direction and "F" is the flag of the block, indicating the block is included into the stope if it is "1" and excluded if it is "0". If the block is on the left border of the model, the first set of conditions is not applied, and if the block is on the right border of the model, the second set of conditions is not applied.

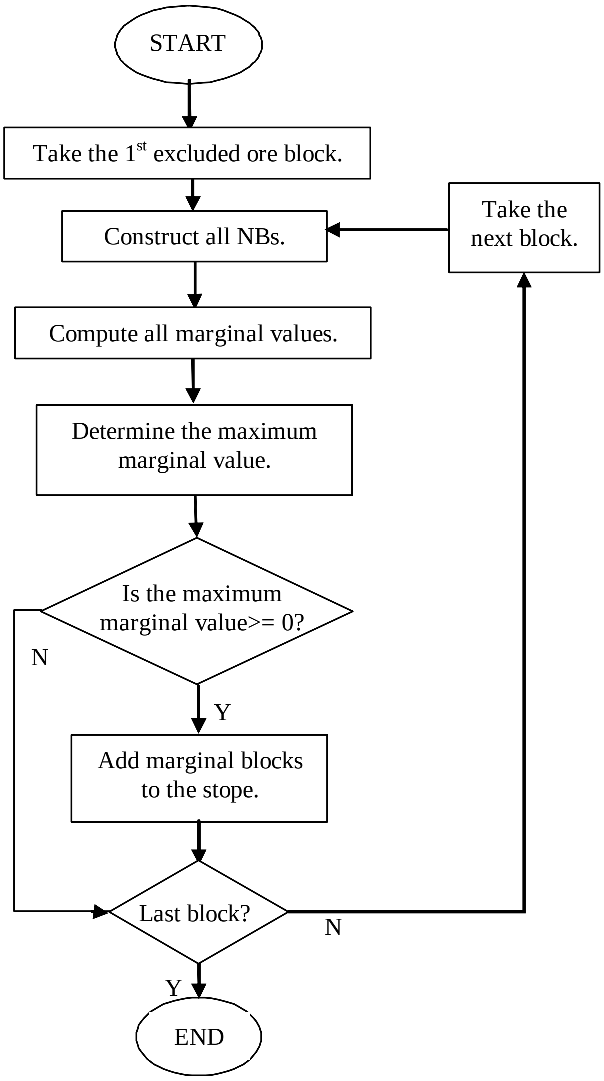

START
Take the 1st excluded ore block.
Construct all NBs.
Take the next block.
Compute all marginal values.
Determine the maximum marginal value.
Is the maximum marginal value>= 0?
N
Y
Add marginal blocks to the stope.
Last block?
N
Y
END

Fig. 11: The flowchart of the multiple pass algorithm for non-negative blocks

For 2D and 3D constraints, the situation is more complicated and the simple expansion of above conditions to two or three dimensions is not adequate. Generally speaking, exclusion of any block B_{ijk} may influence all blocks within its neighbourhood space. So, each block of the neighbourhood space should be checked to make sure that it forms at least one flagged neighbourhood that does not contain B_{ijk} to allow exclusion of B_{ijk}. Relations expressed in Equation (1) above are, in fact, the reduced form of these conditions to one dimension.

In order to run this pass, negative flagged blocks should be sorted first. If a waste block is removed from the ultimate stope, it may prevent exclusion of the next waste block, ie the chance of exclusion is reduced with the order of search. Therefore, the algorithm starts to examine the block with the minimum value (ie the most costly block) then the second and so on. The flowchart for checking possibility of exclusion of waste blocks from the ultimate stope is illustrated in Fig. 12.

5. NUMERICAL EXAMPLES

Consider the 1D example, shown in Fig.3a. Applying the first pass of the algorithm has resulted in two separate stopes; the first one is consisted of three blocks B_1 to B_3 with the net value of (2) and the second one is consisted of five blocks B_5 to B_9 valued at (6). Three non-negative blocks, B_4, B_{11} and B_{12} with a total value of (3) are not included into the final stope. Instead, three waste blocks, B_2, B_3 and B_6 with a total cost of (-7) have been included into the stope.

Non-negative blocks are checked firstly for possibility of inclusion. Clearly, B_4 may be added to the stope since the bridging blocks is consisted of the block itself with a net value of (1). Blocks B_{11} and B_{12} may not still be included into the stope due to their negative marginal value. Bridging blocks and the marginal values for these cases together with the updated stope are shown in Fig. 13. The updated stope value is increased by (1) to a total net value of (9).

Another pass is required to check waste blocks of the ultimate stope to see if they could be taken out. In the 1D example, there is only λ_x, which equals three blocks. Blocks B_2 and B_3 may not be excluded from the stope since their exclusion will make a violation to the minimum stope size. For B_2, the following conditions should be satisfied:

$F_{-1} = F_0 = F_1 = 1$; and $F_3 = F_4 = F_5 = 1$

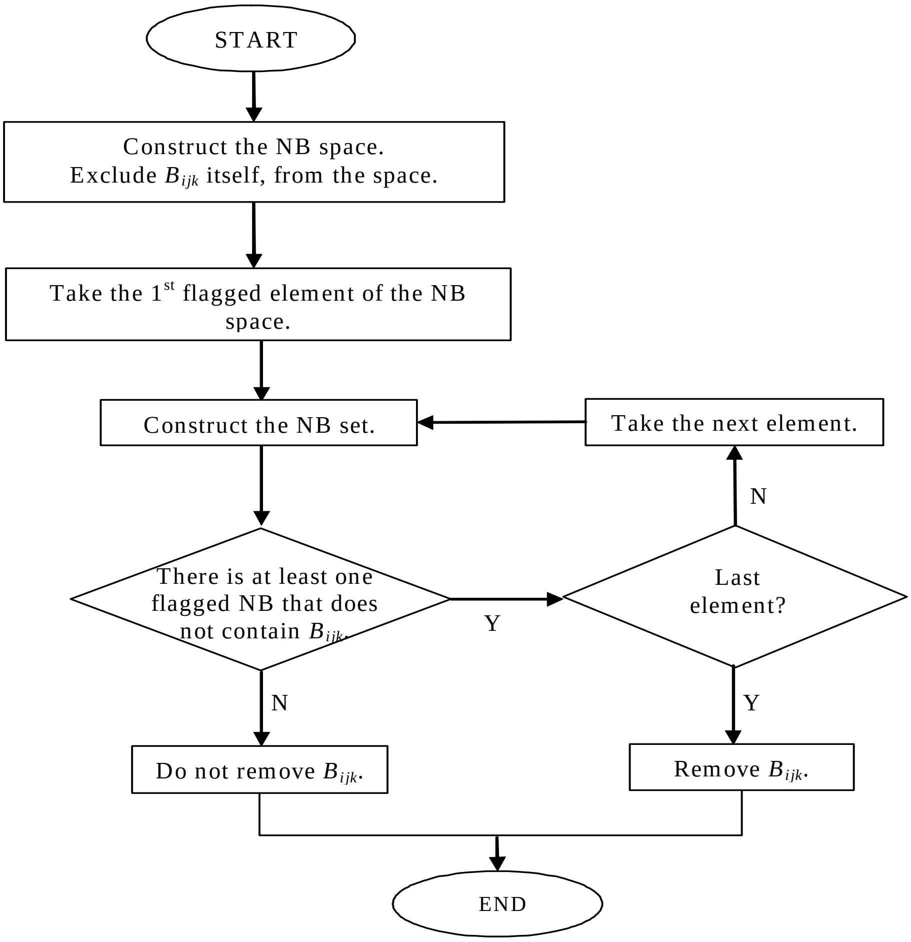

START
Construct the NB space.
Exclude B_{ijk} itself, from the space.
Take the 1st flagged element of the NB space.
Construct the NB set.
Take the next element.
N
There is at least one flagged NB that does not contain B_{ijk}.
Y
Last element?
N
Y
Do not remove B_{ijk}.
Remove B_{ijk}.
END

Fig. 12: The flowchart of the multiple pass algorithm for negative blocks

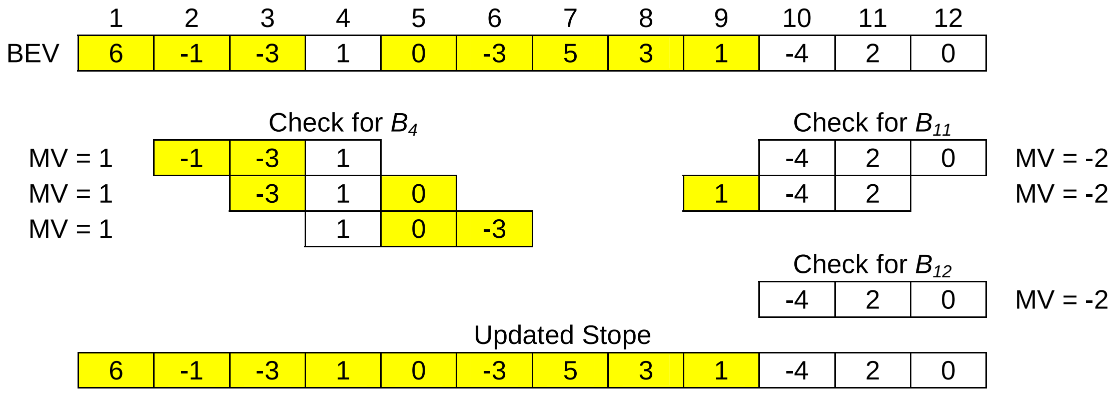


Fig. 13: A 1D example of applying the 2nd pass on non-negative non-flagged blocks

From the above F_0 and F_{-1} are undefined so the block is not excluded. Similarly, for B_3, the following conditions should be satisfied:

$$F_0 = F_1 = F_2 = 1; \text{ and } F_4 = F_5 = F_6 = 1$$

From the above, F_0 is undefined so the block is not excluded. However, the situation for B_6 is different after inclusion of B_4 to the stope. For this block all the following conditions are true:

$$F_3 = F_4 = F_5 = 1; \text{ and } F_7 = F_8 = F_9 = 1$$

and hence, it may be taken out to improve the ultimate stope as shown in Fig. 14. Through this pass, the stope is separated into two parts and the stope net value is increased by (3) to a total net value of (12).

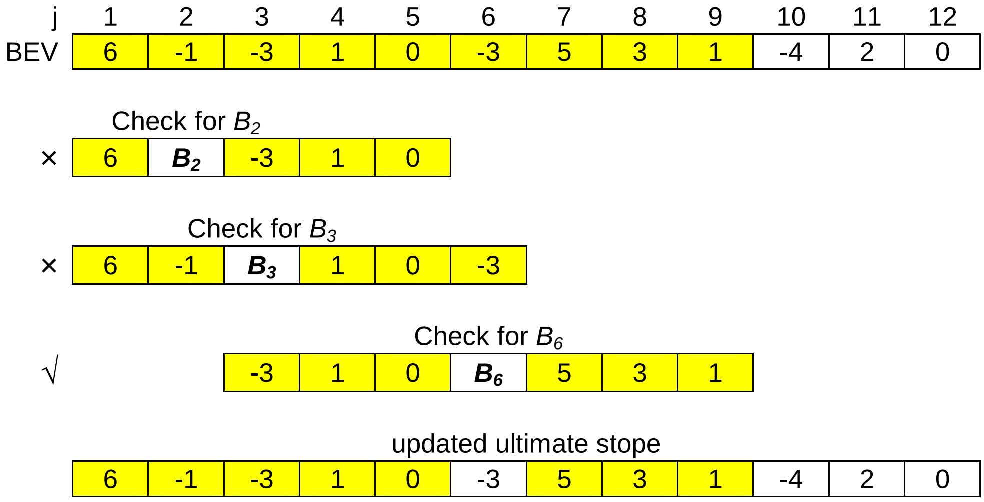


Fig. 14: A 1D example of applying the second pass on non-negative non-flagged blocks

As a 2D example, consider the model discussed above in Fig. 7a. Blocks B_{17}, B_{18} and B_{33} are considered ore but are not included in the ultimate stope. The maximum marginal value provided by both B_{17} and B_{18} is the net value of the neighbourhood {B_{17}, B_{18}, B_{27}, B_{28}}, which is valued at (-1); therefore, they may not be added to the final stope. The bridge for inclusion of B_{33} includes the block itself with the value of (0); so it is added to the final stope. This will make no change in the stope net value but it increases the ore content.

Waste blocks contained in the ultimate stope include B_{14}, valued at (-2) and B_{22}, B_{32}, B_{37}, B_{43}, B_{51}, B_{55} and B_{66}, each valued at (-1). Therefore, the algorithm is applied in the order those blocks are sorted.

1. Exclusion of B_{14} violates neighbourhood of B_{24}; so, it may not be removed.
2. Exclusion of B_{22} violates neighbourhoods of B_{11}, B_{12} and B_{21}; so, it may not be removed.

3. Exclusion of B_{32} violates neighbourhood of B_{31}; so, it may not be removed.
4. Exclusion of B_{37} violates neighbourhood of B_{38}; so, it may not be removed.
5. Exclusion of B_{43} does not violate any neighbourhoods; so, it ***may be*** removed.
6. Exclusion of B_{51} violates neighbourhoods of B_{61}; so, it may not be removed.
7. Exclusion of B_{55} violates neighbourhoods of B_{44}, B_{45}, B_{54} and B_{65}; so, it may not be removed.
8. Exclusion of B_{66} violates neighbourhoods of B_{65} and B_{56}; so, it may not be removed.

As a result, in this pass B_{43} is excluded from the ultimate stope, as shown in Fig. 15 and the total stope value is increased by one.

	1	2	3	4	5	6	7	8
1	1	0	1	-2	3	1	0	2
2	2	-1	2	1	1	1	-2	-1
3	1	-1	0	-2	2	1	-1	1
4	2	4	-1	3	1	-1	2	3
5	-1	2	1	3	-1	2	0	1
6	2	1	0	-2	3	-1	4	1

Fig. 15: A 2D example of applying the second pass

6. CONCLUSIONS

The *MVN* algorithm Developed for optimisation of stope boundaries uses a heuristic approach and hence, it may not guarantee the true optimum limits. The results of applying the algorithm is strongly dependent of the direction, in which it is applied. The main causes are tie cases and negative marginal values. A modification of the algorithm was introduced in this paper to run a second pass of applying the algorithm over the block model. The second pass could improve the obtained ultimate layout by incresing the total stope value or the ore content with no extra costs. The proposed modification provides a 3D analysis and attempts to add non-negative blocks to the stope, which were already excluded during the first pass. It also attempts to remove the negative blocks from the stope, which were already included into the stope during the first pass. Numerical examples were used to discuss the problem causes and to illustrate how the second pass of the algorithm is applied successfully over the block model. Heuristic algorithms, due to their nature, may not guarantee the true optimum. However, it was shown that the *MVN* Multiple Pass algorithm may improve the ultimate stope layout and makes it closer to the true optimum solution.

REFERENCES

1. Lerchs, H and Grossmann, I F, 1965, *"Optimum Design of Open-Pit Mines"*, The Transactions of CIM Bulletin, Volume LXVIII, pp. 17-24.

2. Pana, M.T., 1965, *"The Simulation Approach to Open Pit Design"*, Proceedings of the 5th APCOM Symposium, Tucson, Arizona, pp. zz1-zz24.

3. Riddle, J M, 1977, *"A dynamic programming solution of a block-caving mine layou,"*, Proceedings of the 14th International APCOM Symposium, SME, Colorado, pp. 767-780.

4. Ovanic, J and Young, D S, 1995, *"Economic Optimisation of Stope Geometry Using Separable Programming with Special Branch and Bound Techniques"*, Proceedings of the 3rd Canadian Conference on Computer Applications in the Mineral Industry, Montreal, Canada, pp. 129-135.

5. Jalali, S E and Ataee-pour, M, 2004, *"A 2D Dynamic Programming Algorithm to Optimise Stope Boundaries"*, Proceedings of the 13th International Symposium on Mine Planning and Equipment Selection – MPES'04, M Hardygora et al (eds.), Poland, pp. 45-52.

6. Alford, C, 1995, *"Optimisation in Underground Mine Design"*, Proceedings of 25th APCOM Symposium, The Australasian Institute of Mining and Metallurgy: Brisbane, pp. 213-218.

7. Ataee-pour, M, 1997, *"A New Heuristic Algorithm to Optimise Stope Boundaries"*, Proceedings of the Second Regional APCOM Symposium on Computer Applications and Operations Research in the Mineral

Industry, L A Puchkov (ed.), Moscow, Russia, 6 p.

8. Ataee-pour, M and Baafi E Y, (1999), *"Stope Optimisation Using the Maximum Value Neighbourhood (MVN) Concept"*, Proceedings of the 28th International Symposium on Computers Applications in the Minerals Industries – APCOM'99, K Dagdelen (ed.), Colorado, pp. 493-501.
9. Ataee-pour, M and Baafi, E Y, 2003, *"SLO – A Program for Stope Limit Optimisation Using A Heuristic Algorithm"*, Proceedings of the 18th International Mining Congress and Exibition of Turkey – IMCET'2003, G Ozbayoglu (ed.), Turkey, pp. 295-301.