Canada Research Chair (Tier I) in
Sustainable Mineral Resource Development and Optimization under Uncertainty

COSMO – Stochastic Mine Planning Laboratory
Dept. of Mining and Materials Engineering

LUNCHTIME SEMINAR

“Metaheuristic Methods for Open Pit Mine Scheduling”

Jacques Ferland
Université de Montréal

Metaheuristic Methods
for
Open Pit Mine Scheduling

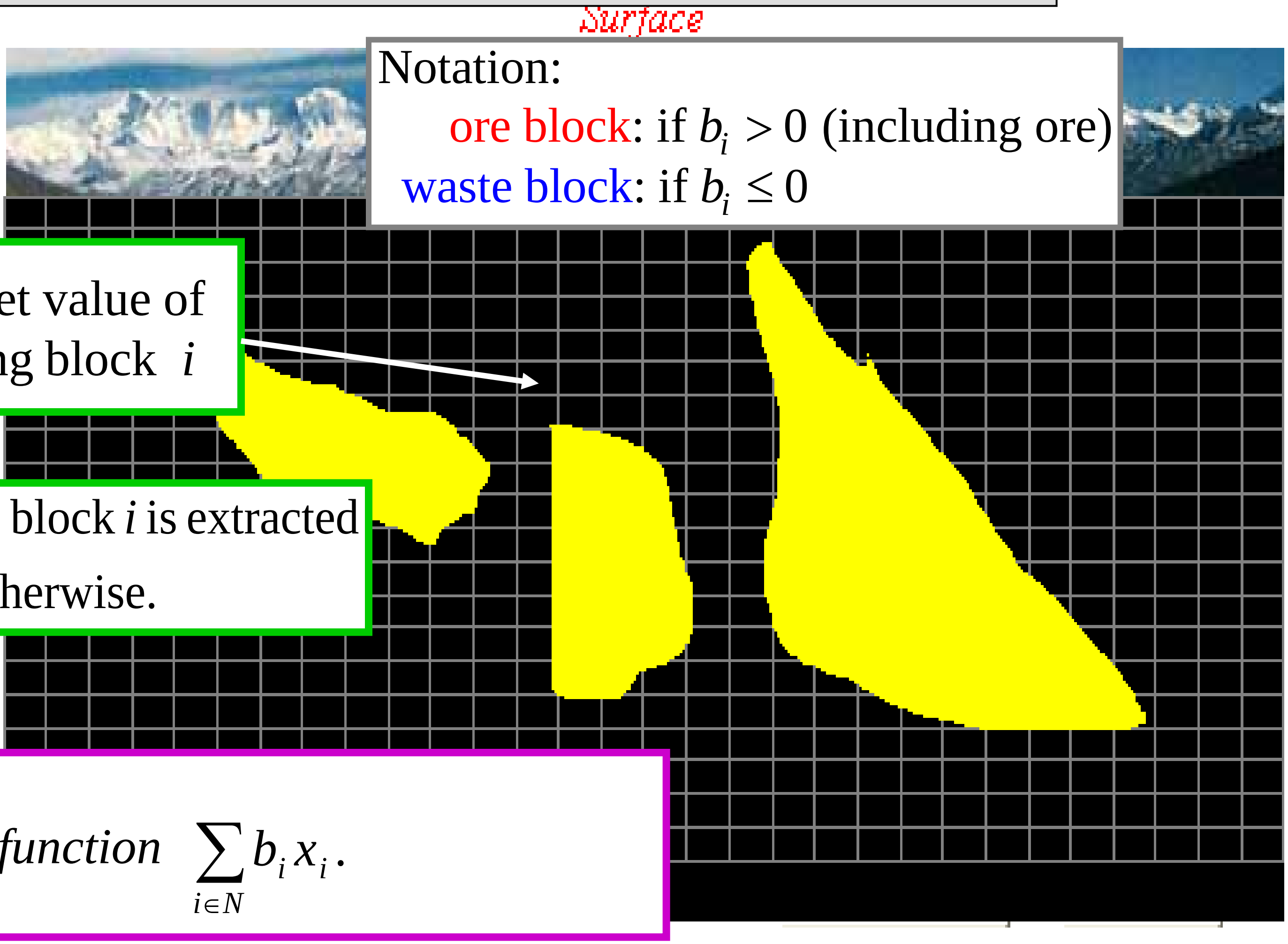
Jacques A. Ferland,
DIRO
CIRRELT
University of Montreal

Introduction

Mine à ciel ouvert
de Palabora en Afrique
du Sud

Mine à ciel ouvert en
Australie

RIOT Mining Problem web site:
http://riot.ieor.berkeley.edu/riot/Applications/OPM/OPMInteractive.html



Notation:

ore block: if $b_i > 0$ (including ore)

waste block: if $b_i \leq 0$

b_i the net value of extracting block i

$$x_i = \begin{cases} 1 & \text{if block } i \text{ is extracted} \\ 0 & \text{otherwise.} \end{cases}$$

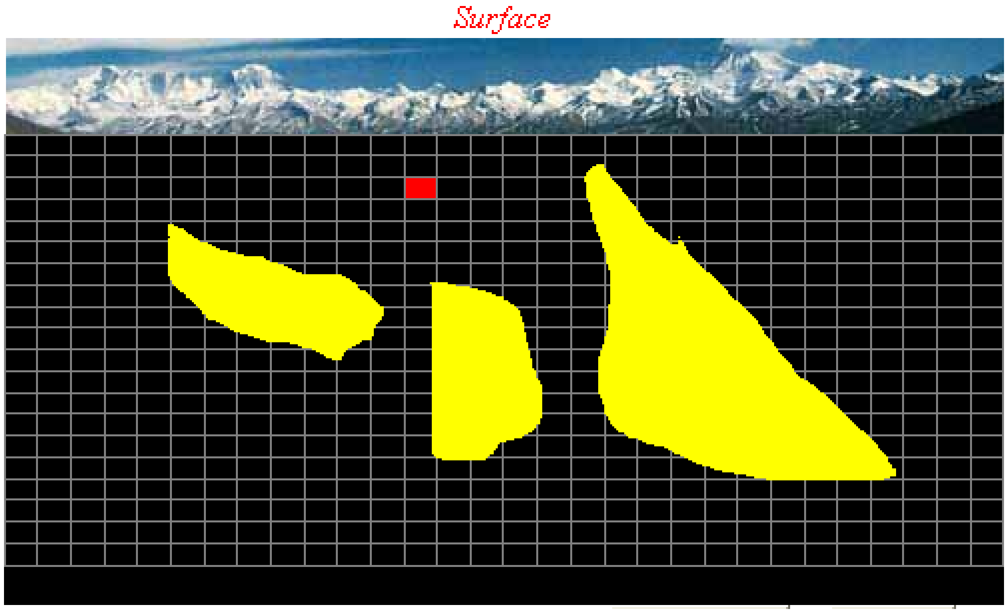
objective function $\sum_{i \in N} b_i x_i$.

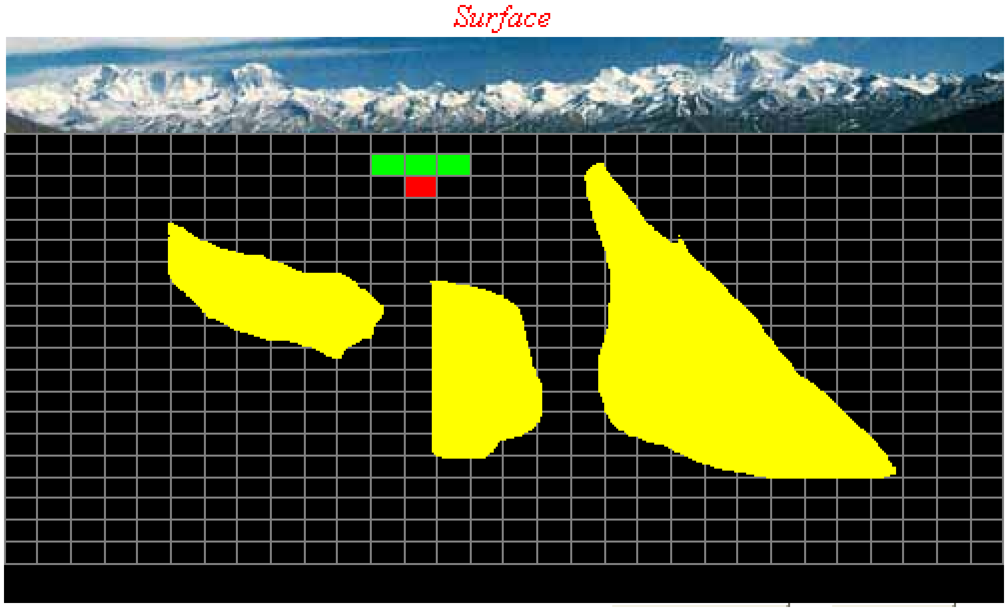
Maximal Open Pit problem: to determine the maximal gain expected from the extraction

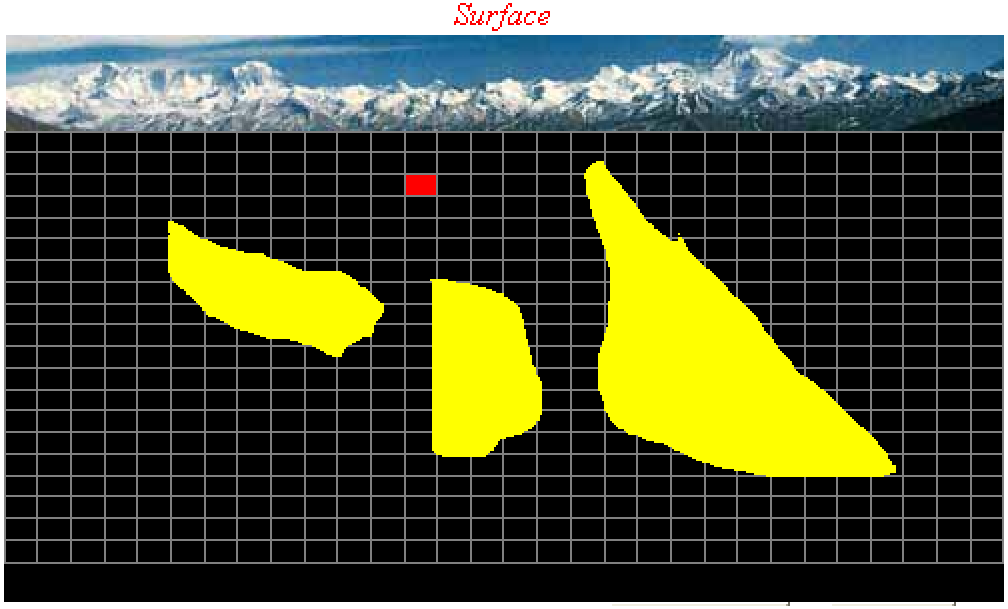
Introduction

Maximal pit slope constraints to identify the set B_i of predecessor blocks that have to be removed before block i

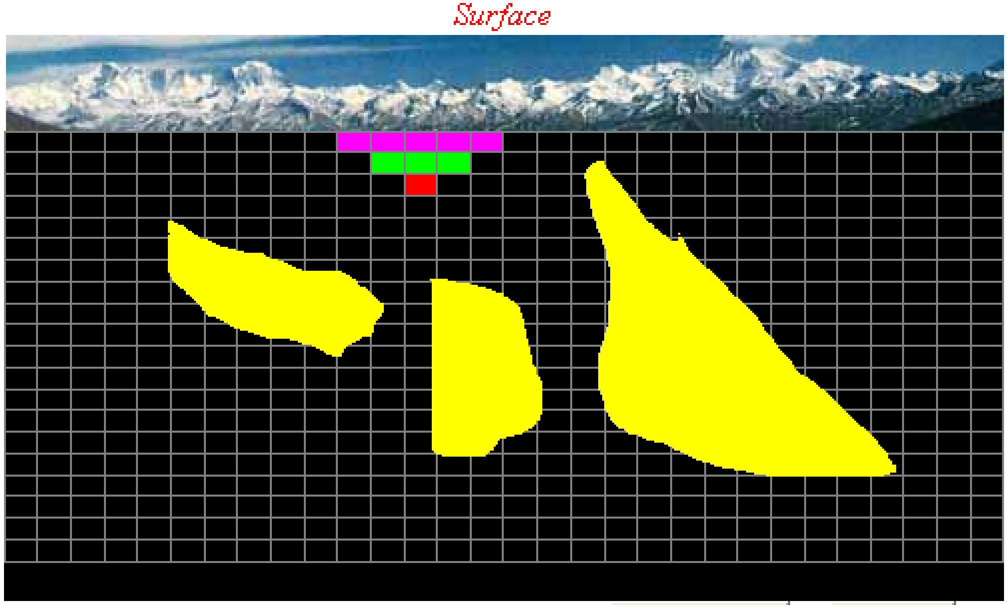
Maximal pit slope constraints to identify the set B_i of predecessor blocks that have to be removed before block *i*

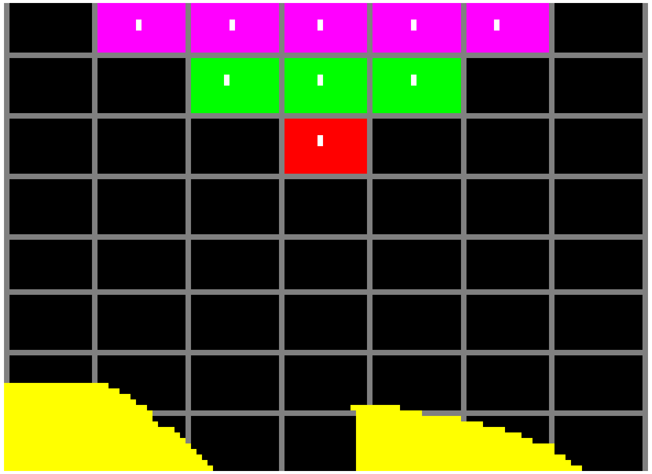






Maximal pit slope constraints to identify the set B_i of predecessor blocks that have to be removed before block *i*



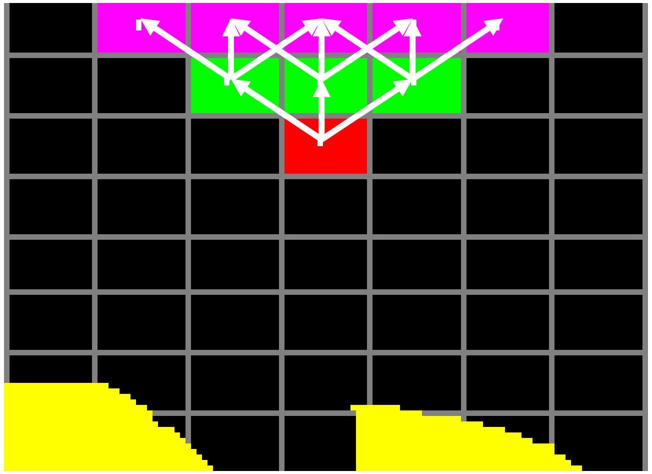


Use the *oper*
specify the *n*

(MOP) Max $\sum_{i \in N} b_i x_i$

Subject to $x_j - x_i \geq 0 \qquad (i, j) \in A \qquad (1)$

$x_i = 0 \text{ or } 1 \qquad i \in N. \qquad (2)$



$V = \{ i \in N : \text{node } i \text{ corresponds to block } i \}$

$A = \{ (i, j) : j \in B_i, i \in N \}$

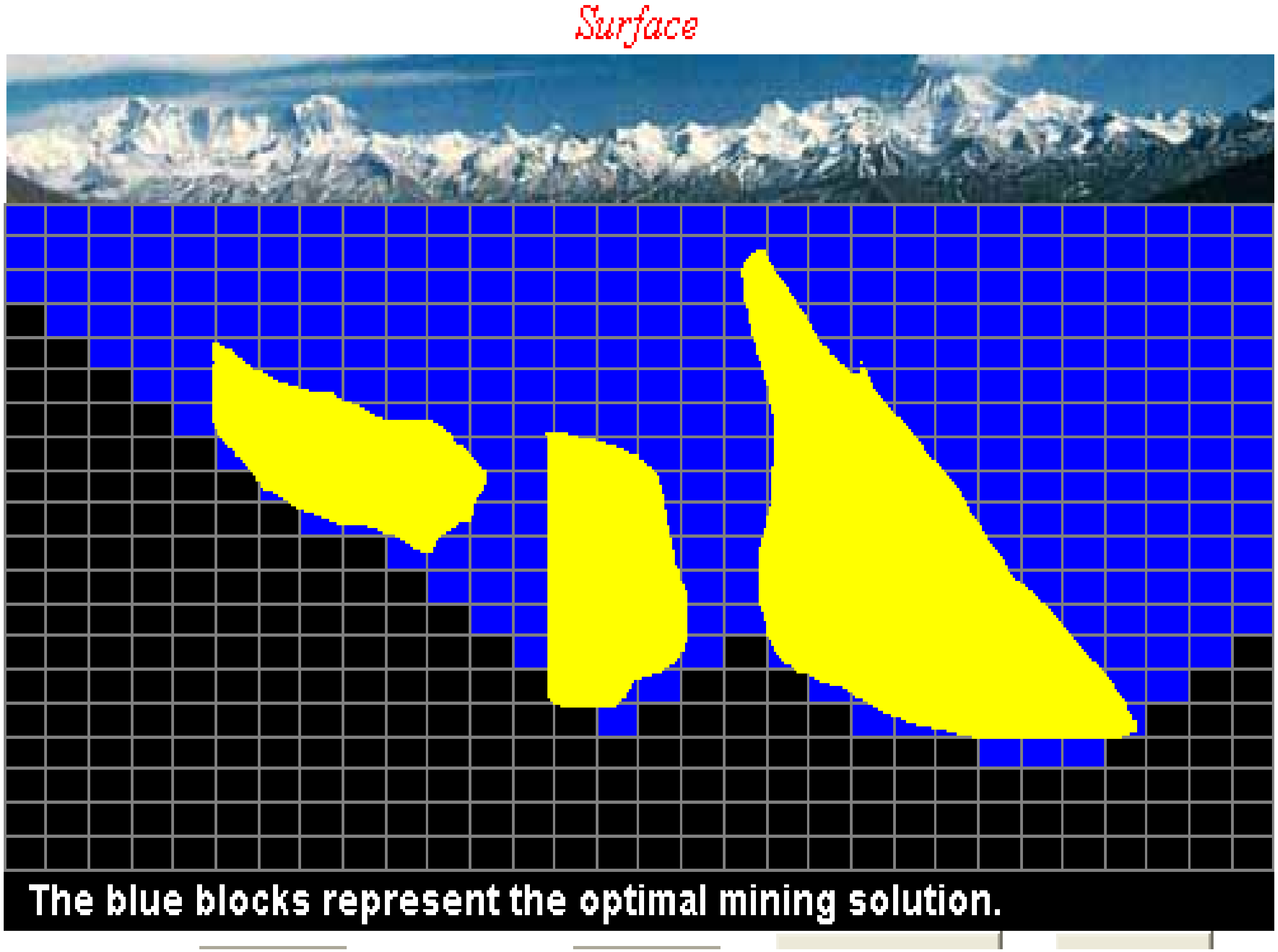
The *maximal pit slope constraints*:

$$x_j - x_i \geq 0 \qquad (i, j) \in A$$

$$
\begin{array}{llll}
\text{(MOP)} \quad \text{Max} & \sum_{i\in N} b_i x_i & & \\
\text{Subject to} & x_j - x_i \geq 0 & (i, j) \in A & (1) \\
 & x_i = 0 \text{ or } 1 & i \in N. & (2)
\end{array}
$$

- (MOP) equivalent to determine the maximal closure of $G = (V, A)$

- Equivalent to determine the *minimum cut* $\left(S, \overline{S}\right)$ of the associated Picard’s graph $\overline{G} = \left(\overline{V}, \overline{A}\right)$



The blue blocks represent the optimal mining solution.

Scheduling block extraction

Account for operational constraints:

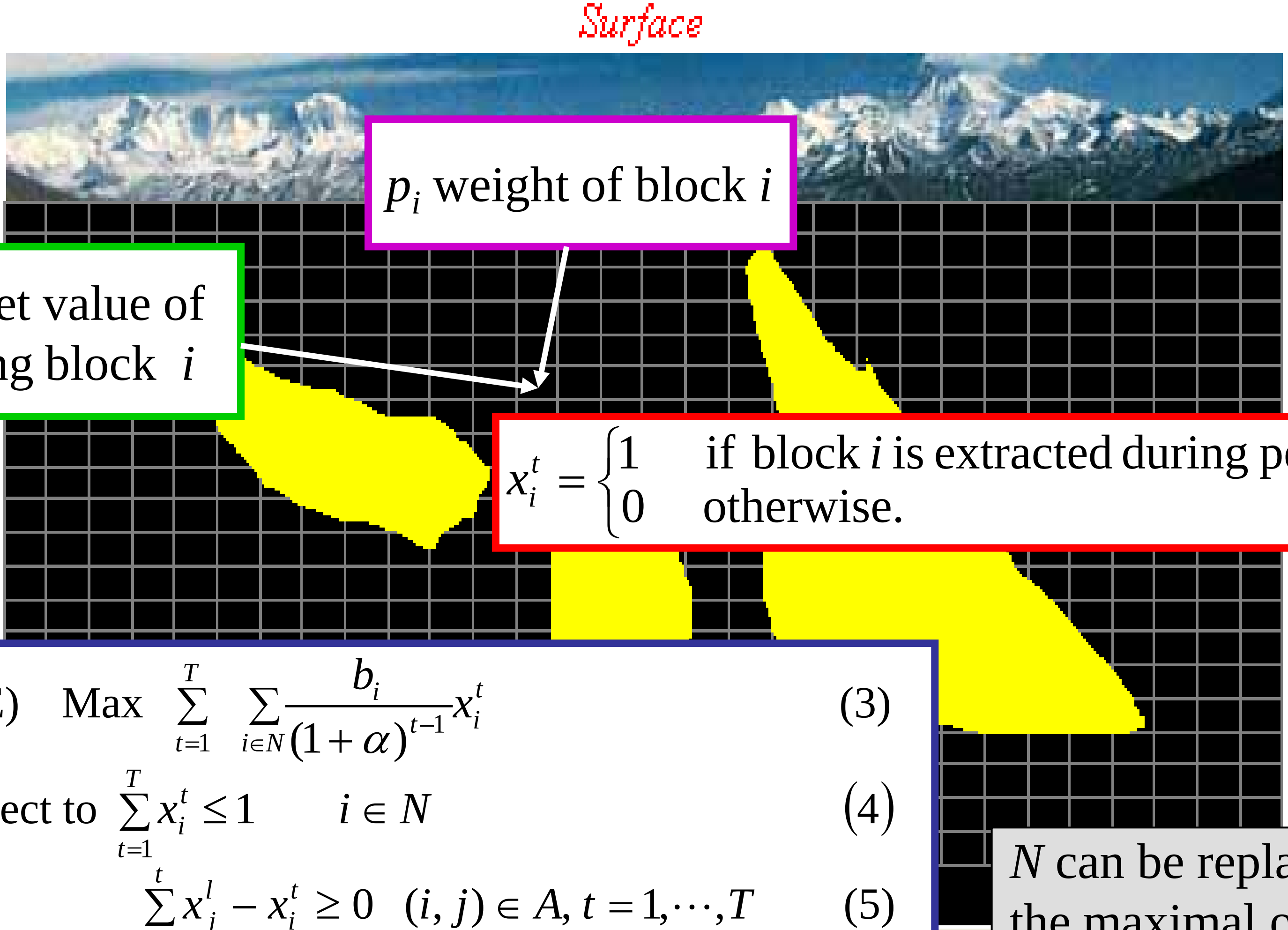
C_t the maximal weight that can be extracted during period t

and for the discount factor during the extracting horizon:

$\frac{1}{1+\alpha}$ discount rate per period

Reach for the best ore blocks $(b_i > 0)$ as early as possible
because the discount factor induces a reduction of the value over time

Current value of block i extracted at period t : $\frac{b_i}{(1+\alpha)^{t-1}}$



p_i weight of block i

b_i the net value of extracting block i

$$x_i^t = \begin{cases} 1 & \text{if block } i \text{ is extracted during period } t \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{(SBE)} \quad \text{Max} \sum_{t=1}^{T} \sum_{i \in N} \frac{b_i}{(1+\alpha)^{t-1}} x_i^t \qquad (3)$$

$$\text{Subject to} \sum_{t=1}^{T} x_i^t \le 1 \qquad i \in N \qquad (4)$$

$$\sum_{l=1}^{t} x_j^l - x_i^t \ge 0 \quad (i,j) \in A,\ t = 1,\cdots,T \qquad (5)$$

$$\sum_{i \in N} p_i x_i^t \le C_t \qquad t = 1,\cdots,T \qquad (6)$$

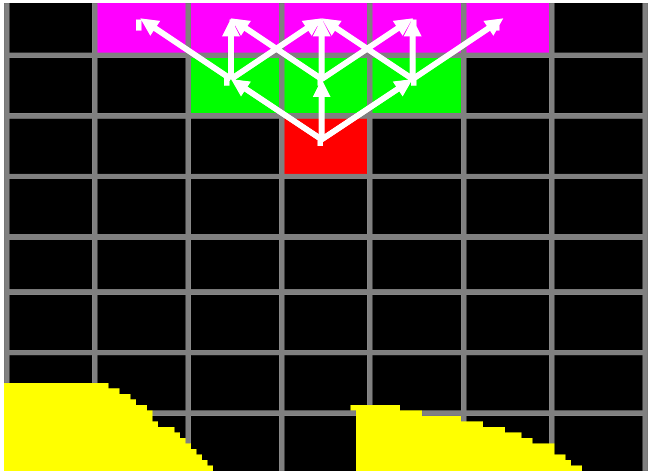
$$x_i^t = 0 \text{ or } 1 \qquad i \in N,\ t = 1,\cdots,T. \qquad (7)$$

N can be replaced by the maximal open pit $N^* = (S - \{s\})$

Scheduling block extraction ↔ RCPSP

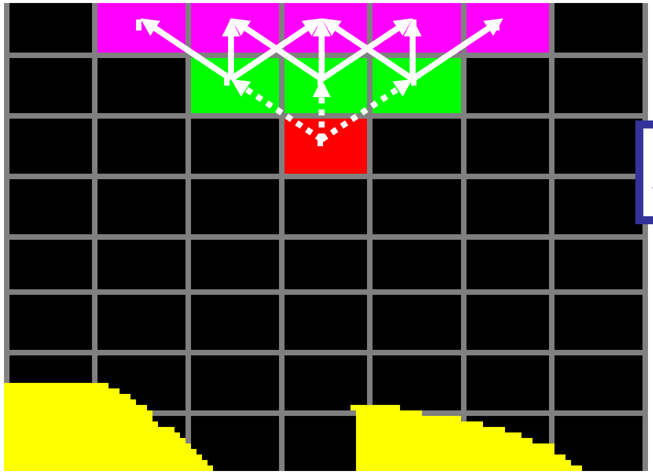
- Open pit extraction ↔ project
- Each block extraction ↔ activity
- Precedence relationship derived from the slope constraints

Scheduling block extraction ↔ RCPSP



$$A = \left\{ (i, j) : j \in B_i, i \in N \right\}$$

Scheduling block extraction ↔ RCPSP



$$A = \left\{ (i, j) : j \in B_i, i \in N \right\}$$

$$P_i = \left\{ \text{block } i \text{ predecessors} \right\} = \left\{ j \in N^* : (i, j) \in A \right\}$$

Scheduling block extraction ↔ RCPSP

- Open pit extraction ↔ project
- Each block extraction ↔ activity
- Precedence relationship derived from the open pit graph

$$\boxed{P_i = \left\{ \text{block } i \text{ predecessors} \right\} = \left\{ j \in N^* : (i, j) \in A \right\}}$$

- Reward associated with activity (block) *i* depends of the extraction period *t*

$$\boxed{\frac{b_i}{(1+\alpha)^{t-1}}}$$

Solution encoding and decoding

Encoding (Representation):
A solution is encoded as a block list indicating their order of extraction and where the precedence relationship is satisfied

Decoding :
Use a decoding procedure to generate a solution corresponding to a bloc list

Decoding a block list into a schedule

Serial decoding

- Initiate the first extraction period $t = 1$

- During any period t:

  - The next block to be extracted is the first block in the rest of the block list (including the blocks not extracted yet) having all their predecessors already extracted such that the capacity C_t is not exceeded by its extraction.
  - If no such block exists, then a new extraction period ($t + 1$) is initiated.

- When all blocs are extracted, the corresponding solution is obtained by stopping extraction when the discounted profit stops increasing.

Metaheuristic solution approach

1. Generate an initial block list
2. At each iteration, apply a local search procedure to generate a new block list in the neighborhood of the current block list.

   Two different neighborhoods are used in order to intensify or to diversify the search.

   moving one ore block
   moving several ore blocks

3. Decode each block list generated.

Outline of the solution approach

Now we introduce

- the procedure to generate the initial solution
- the two neighborhoods

before summarizing the details of the procedure.

Initial solution

Notation:

ore block: if $b_i > 0$ (including ore)

waste block: if $b_i \leq 0$

Reach for the best ore blocks ($b_i > 0$) as early as possible because the discount factor induces a reduction of the value over time

Basic process: include the ore blocks in the list after including waste blocks whenever necessary to satisfy precedence

Associate with each ore block i a depth value

$depth_i =$ number of blocks still to be included in the list before ore block i is scheduled

	0	1	2	3	4	5	6	7	8	9	l
0	-4	-4	-4	6	-4	-4	6	-4	-4	-4	
1	-4	-4	6	6	8	-4	16	4	-4	-4	
2	-4	-4	10	4	-4	-4	-4	16	-4	-4	
3	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	
h											

Associate with each ore block i a depth value

$depth_i =$ number of blocks still to be included in the list before ore block i is scheduled

Examples: $depth_{\begin{bmatrix}0\\6\end{bmatrix}} = 1$

$$depth_{\begin{bmatrix}2\\2\end{bmatrix}} = 9$$

At each iteration of the procedure

Determine the ore block i having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i

Hence to complete the iteration

a) Include in the list the waste blocks preceding the ore block selected

b) Include the ore block selected in the list

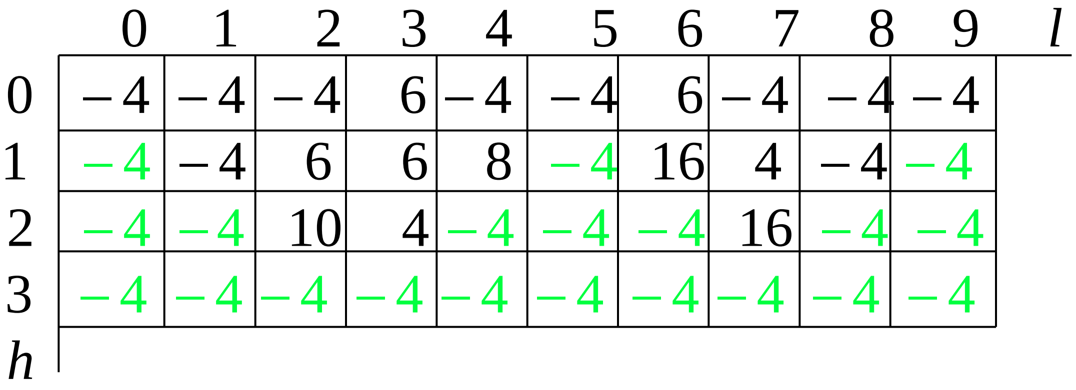
c) Update the *depth* of the ore blocks not scheduled yet

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

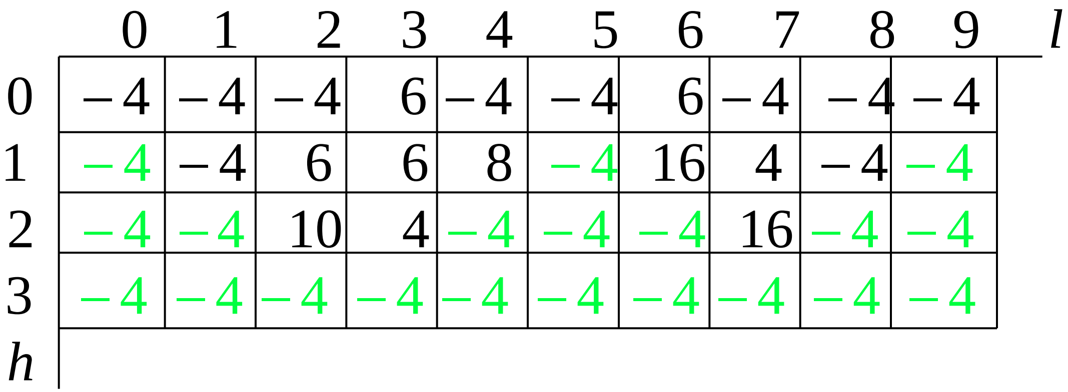
	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											



	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											



	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

Determine the ore block i having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i
Include in the list the waste blocks preceding the ore block selected
Include the ore block selected in the list
Update the *depth* of the ore blocks not scheduled yet

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	1	−4	−4	1	−4	−4	−4	
1	−4	−4	4	4	4	−4	4	4	−4	−4	
2	−4	−4	9	9	−4	−4	−4	9	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

Determine the ore block i having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i
Include in the list the waste blocks preceding the ore block selected
Include the ore block selected in the list
Update the *depth* of the ore blocks not scheduled yet

$$\begin{array}{c} 1 \\ \begin{bmatrix} 0 \\ 6 \end{bmatrix} \end{array}$$

	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	1	−4	−4	0	−4	−4	−4	
1	−4	−4	4	4	4	−4	3	3	−4	−4	
2	−4	−4	9	9	−4	−4	−4	8	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

$$\begin{array}{cc} 1 & 2 \\ \begin{bmatrix} 0 \\ 6 \end{bmatrix} & \begin{bmatrix} 0 \\ 3 \end{bmatrix} \end{array}$$

Determine the ore block *i* having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i
Include in the list the waste blocks preceding the ore block selected
Include the ore block selected in the list
Update the *depth* of the ore blocks not scheduled yet

	0	1	2	3	4	5	6	7	8	9	*l*
0	−4	−4	−4	0	−4	−4	0	−4	−4	−4	
1	−4	−4	3	3	3	−4	3	3	−4	−4	
2	−4	−4	8	8	−4	−4	−4	8	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

Determine the ore block i having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i
Include in the list the waste blocks preceding the ore block selected
Include the ore block selected in the list
Update the *depth* of the ore blocks not scheduled yet

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \begin{bmatrix} 0 \\ 6 \end{bmatrix} & \begin{bmatrix} 0 \\ 3 \end{bmatrix} & \begin{bmatrix} 0 \\ 7 \end{bmatrix} & \begin{bmatrix} 0 \\ 5 \end{bmatrix} & \begin{bmatrix} 1 \\ 6 \end{bmatrix} \end{array}$$

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	0	−4	−4	0	−4	−4	−4	
1	−4	−4	3	3	3	−4	3	3	−4	−4	
2	−4	−4	8	8	−4	−4	−4	8	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

Determine the ore block i having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i
Include in the list the waste blocks preceding the ore block selected
Include the ore block selected in the list
Update the *depth* of the ore blocks not scheduled yet

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \begin{bmatrix}0\\6\end{bmatrix} & \begin{bmatrix}0\\3\end{bmatrix} & \begin{bmatrix}0\\7\end{bmatrix} & \begin{bmatrix}0\\5\end{bmatrix} & \begin{bmatrix}1\\6\end{bmatrix} \end{array}$$

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	0	−4	−4	0	−4	−4	−4	
1	−4	−4	3	3	2	−4	0	2	−4	−4	
2	−4	−4	8	7	−4	−4	−4	5	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	6	−4	−4	6	−4	−4	−4	
1	−4	−4	6	6	8	−4	16	4	−4	−4	
2	−4	−4	10	4	−4	−4	−4	16	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

Determine the ore block i having the smallest $depth_i$
If more than one ore blocks have the smallest *depth* value,
then select one of those having the largest value b_i
Include in the list the waste bloc preceding the ore block selected
Include the ore block selected in the list
Update the *depth* of the ore blocks not scheduled yet

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	7	5	6	4	4	2	3	1	2	3	8	7	9	8	7	0	1	2

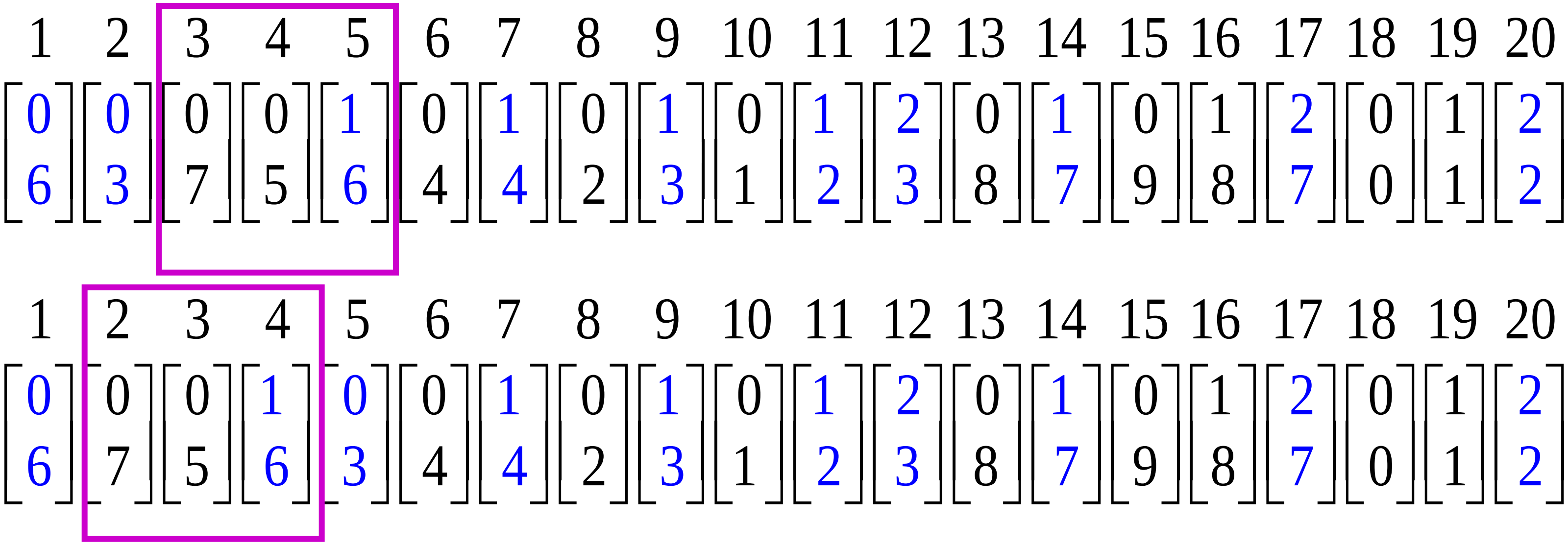
	0	1	2	3	4	5	6	7	8	9	l
0	−4	−4	−4	0	−4	−4	0	−4	−4	−4	
1	−4	−4	0	0	0	−4	0	0	−4	−4	
2	−4	−4	0	0	−4	−4	−4	0	−4	−4	
3	−4	−4	−4	−4	−4	−4	−4	−4	−4	−4	
h											

First neighborhood moving one ore block

Backward move

Move the selected ore block with its cluster of waste blocks to preceding positions in the list

Process can be repeated to generate different neighboor solutions until reaching the nearest predecessor of the ore block



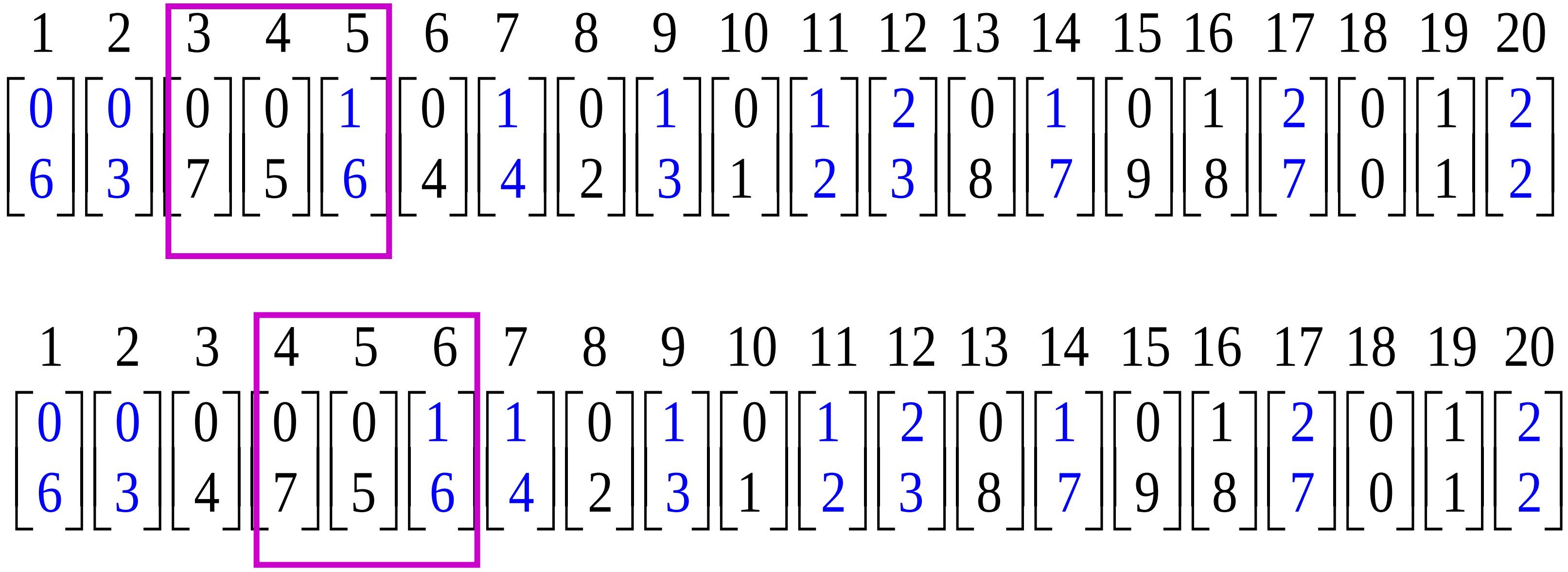
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	7	5	6	4	4	2	3	1	2	3	8	7	9	8	7	0	1	2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	1	0	0	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	7	5	6	3	4	4	2	3	1	2	3	8	7	9	8	7	0	1	2

Forward move

Move the selected ore block with its cluster of waste blocks to succeding positions in the list

Process can be repeated to generate different neighboor solutions until reaching the nearest successor of the ore block



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	7	5	6	4	4	2	3	1	2	3	8	7	9	8	7	0	1	2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	1	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	4	7	5	6	4	2	3	1	2	3	8	7	9	8	7	0	1	2

Purged forward move

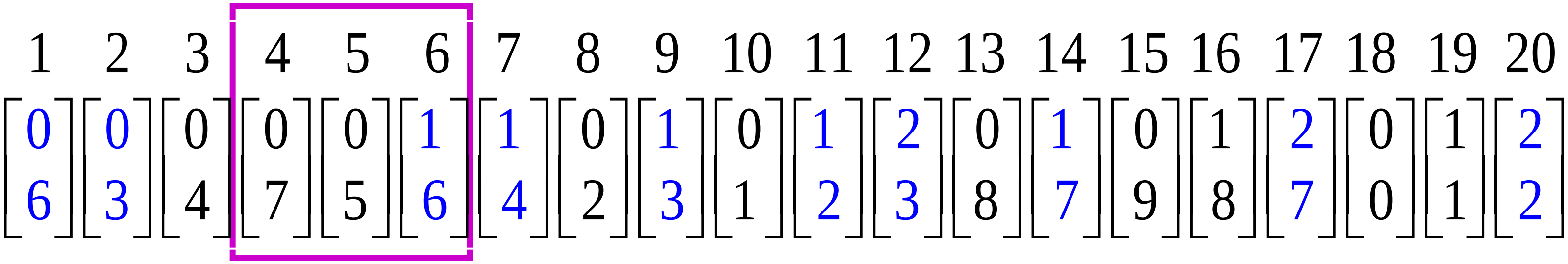
Move the selected ore block with its cluster of waste blocks to succeding positions in the list

Process can be repeated to generate different neighboor solutions until reaching the nearest successor of the ore block

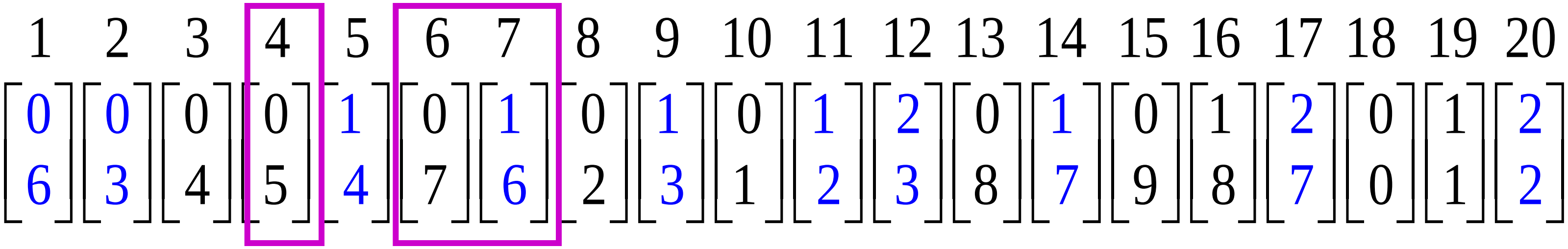
Purge some negative blocs of the cluster:
waste block cannot be moved any further because it is also the predecessor of another ore block now positioned before the selected moving cluster

Purged forward move

Move the selected ore block with its cluster of waste blocks to succeding positions in the list
Purge some negative blocs of the cluster:
waste block cannot be moved any further because it is also the predecessor of another ore block now positioned before the selected moving cluster



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	1	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	4	7	5	6	4	2	3	1	2	3	8	7	9	8	7	0	1	2



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	4	5	4	7	6	2	3	1	2	3	8	7	9	8	7	0	1	2

Unique ore block process on cluster *i*

Generate all the elements of the unique ore block neighborhood by moving (forwardly and backwardly) the cluster associated with the ore bloc *i*.

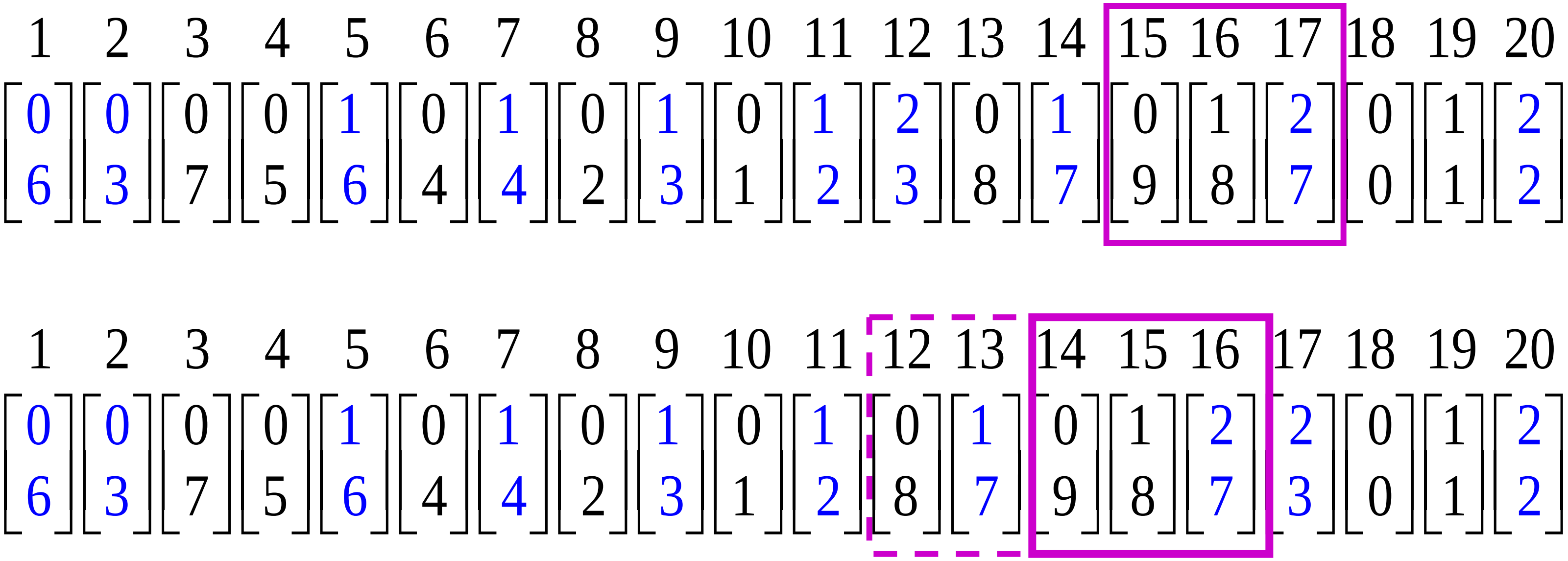
Determine the best valued decoded solution

Second neighborhood moving multiple ore blocks

Backward process where several ore blocks are moved

Move the selected ore block with its cluster of waste blocks to preceding positions in the list.

Furthermore, whenever a preceding ore block is reached, the backward process continues, but the cluster of this ore block is also moved backward to preserve precedence constraints.



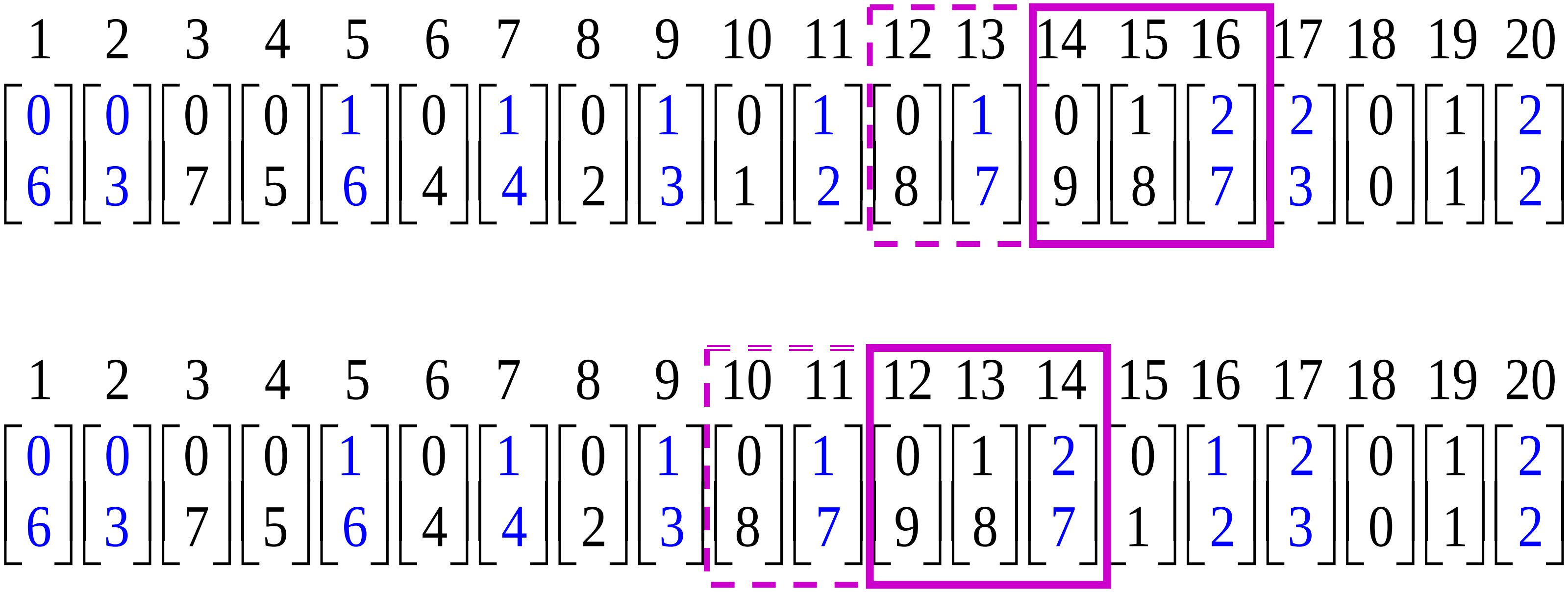
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	2	0	1	0	1	2	0	1	2
6	3	7	5	6	4	4	2	3	1	2	3	8	7	9	8	7	0	1	2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	2	2	0	1	2
6	3	7	5	6	4	4	2	3	1	2	8	7	9	8	7	3	0	1	2

Backward process where several ore blocks can move

Move the selected ore block with its cluster of waste blocks to preceding positions in the list

Furthermore, whenever a preceding ore block is reached, the backward process continues, but the cluster of this ore block is also moved backward to preserve precedence constraints.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	2	2	0	1	2
6	3	7	5	6	4	4	2	3	1	2	8	7	9	8	7	3	0	1	2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	1	0	1	0	1	0	1	0	1	2	0	1	2	0	1	2
6	3	7	5	6	4	4	2	3	8	7	9	8	7	1	2	3	0	1	2

Multiple ore block process on cluster *i*

Generate all the elements of the multiple ore block neighborhood by using the backward process of the cluster associated with the ore bloc *i*.

Determine the best valued decoded solution

Implementation of a metaheuristic procedure

$NbCl =$ number of ore blocs

```
Determine an initial solution
Set $iter := 1$
 while $iter < Nbiter$ do
     for $i = NbCl$ to 1 do
         Apply the Multiple ore blocks process on a randomly
         selected cluster
         Apply the Unique ore block process on cluster $i$
 1 $iter := iter + 1$
 if the evaluation improvement is $< \varepsilon$ do
     for $i = 1$ to $NbCl$ do
         Apply the Multiple ore blocks process on cluster $i$
     if there is no evaluation improvement STOP
     otherwise go to 1
```

Numerical experimentation

Preliminary experimentation with this implementation
is very encouraging.
Further tests need to be completed.

Other implementations can be obtained by combining
the unique ore bloc process and the multiple ore bloc process
with genetic algorithms used to solved the RCPSP

Second encoding of the solution
and
Evolutionary process

Genotype representation of solution

Similar to Hartman’s priority value encoding for RCPSP

$$\boxed{PR = [pr_1, \cdots, pr_{|N|}]}$$

$\boxed{pr_i \in [0,1]}$ priority of scheduling block *i* extraction

$$\boxed{\sum_{i=1}^{|N|} pr_i = 1}$$

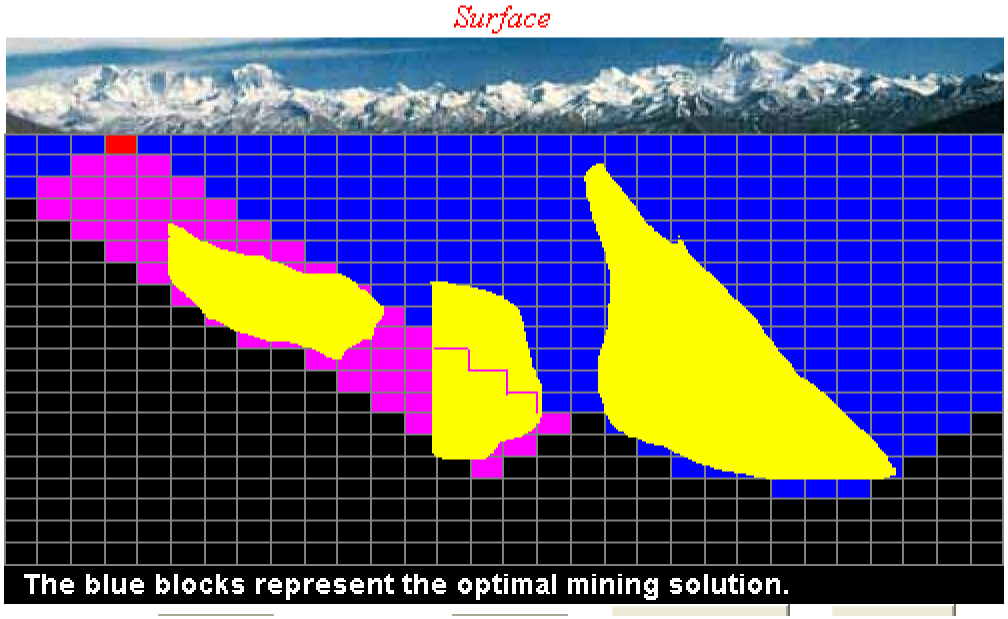
Priority of a block

- Consider its

  net value b_i and

  impact on the extraction of other blocks in future periods

- *Block lookahead value* $\bar{b}_i$ (Tolwinski and Underwood) determined by referring to the *spanning cone* SC_i of block i



$$\boxed{SC_i = \{j \in N^* : i \text{ must be extracted before } j\} \cup \{i\}.}$$

$$\boxed{\overline{b}_i = \sum_{j \in SC_i} b_j}$$

Genotype priority vector generation

- Several different genotype priority vectors can be randomly generated with a *GRASP procedure* biased to give higher priorities to blocks *i* having larger lookahead values $\overline{b}_i$

- Several feasible solutions can be obtained by decoding different genotype vectors generated with the *GRASP procedure*.

Decoding of a representation *PR* into a solution *x*

- Serial decoding to schedule blocks sequentially one by one to be extracted
- To initiate the first extraction period $t = 1$:

  remove the block among those having no predecessor (i.e., in the top layer) having the highest priority.
- During any period t, at any stage of the decoding scheme:

  the next block to be removed is one of those with the highest priority among those having all their predecessors already extracted such that the capacity C_t is not exceeded by its extraction.

  If no such block exists, then a new extraction period $(t + 1)$ is initiated.

Evolutionary process

- Evolutionary process evolving in the set of genotype vectors to converge to an improved feasible solution
- Initial population *P* of *M* genotype vectors (individuals) generated using GRASP

$$P = \left\{ PR^1, \cdots, PR^M \right\}.$$

Particle Swarm Procedure

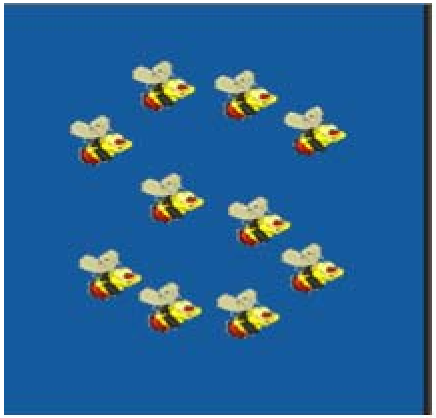
- Evolutionary process evolving in the set of genotype vectors to converge to an improved feasible solution
- Initial population *P* of *M* genotype vectors (individuals) generated using GRASP

$$P = \left\{ PR^1, \cdots, PR^M \right\}.$$

- Denote

$\overline{PR}^k$ the best achievement of the individual *k* up to the current iteration

PRb the best overall genotype vector achieved up to the current iteration



Particle Swarm Procedure

- Denote

  $\overline{PR}^{k}$ the best achievement of the individual k up to the current iteration

  PRb the best overall genotype vector achieved up to the current iteration

$$\left.\begin{array}{r} \left.\begin{array}{l}\text{current sol. } k\\ \text{best achiev. of } k\\ \text{best sol.}\\ \text{current velocity}\end{array}\right\} \rightarrow \left\{\begin{array}{l}\text{new current}\\ \text{velocity}\end{array}\right. \\ \text{current sol. } k \end{array}\right\} \rightarrow \left\{\begin{array}{l}\text{current sol. } k\\ \text{in next population}\end{array}\right.$$

Genetic Algorithm

- Evolutionary process evolving in the set of genotype vectors to converge to an improved feasible solution
- Initial population P of M genotype vectors (individuals) generated using GRASP

$$P = \left\{ PR^{1}, \cdots, PR^{M} \right\}.$$

At each iteration of this Steady-state population genetic algorithm:

- Two parent-solutions are selected

- A crossover operator is applied to the pair of parent-solutions to generate two offspring-solutions.

- The population of the next iteration includes the N best solutions among the current population and the offspring-solutions

Numerical experimentation

Preliminary experimentation indicates that this approach does not give results as good as those obtained with the first approach introduced before.

McGill

CIM
ICM

UPCOMING LUNCHTIME SEMINARS

April 13th	**Real Time Modeling on Mine Operations Data - Opportunities and Challenges**
12:15, Adams Building, Rm 126	*Dr. Rajive Ganguli, Professor of Mining Engineering, University of Alaska, Fairbanks, Alaska*