2016

# MATH3202 – Assignment 2

## BIGALCO – MINING BLOCKS OF ORE UNDER DIFFERENT RULES

SIMON MCMAHON (43222322)

# Problem Formulation:

**Note that for the purposes of this report:**

**Method 1 =** Original Access rules for block mining

(i.e. Only able to be mined if block have had been completely mined out).

**Method 2 =** Original Method with additional access rules

(i.e. Only able to be mined if all upward adjacent blocks completely mined out

## Initial Discussion:

### What are we optimizing?

Mining Costs which we want to minimise

### What do we decide?

How much bauxite ore to mine from which block each month and which customers receive amounts of ore from which mined block.

### Constraints:

- Each customer requires:
  - Bauxite Demand Each month
  - Min Silicon (%) of combined delivered ore each month
  - Min Aluminium (%) of combined delivered ore each month
- Rainy Season
  - Column 0 or 1 cannot be mined in November and December
- Upper bound on the amount in total mined from each deposit equal to its contained ore
- Logical Block Selection Constraints:
  - Initial Access Rule
    - If the cumulative amount of ore mined from a singular block $(i, j)$ is equal to its contained ore, then it is mined out.
    - Once a block is mined out (exhausted), it cannot be mined from in successive months
    - A block $(i, j)$ in the current month may not be mined unless the block in the row above $(i - 1, j)$ it was completely mined out in the same month
  - Updated (Additional) Access Rules
    - A block $(i, j)$ in the current month may not be mined unless the block in the row above to the left $(i - 1, j - 1)$ it was completely mined out in the same month
      - **Note that on the left column this constraint is not required**
    - A block $(i, j)$ in the current month may not be mined unless the block in the row above, to the right $(i - 1, j + 1)$ it was completely mined out in the same month
      - **Note that on the right column this constraint is not required**

# Formulation:

## Sets:

$Row_i$ = Rows of mining face blocks. Indexed over $i \in Row$

$Col_j$ = Columns of mining face blocks. Indexed over $j \in Col$

$T_t$ = Time in Months. Indexed over $t \in T$

$C_c$ = Customers of the bauxite mine. Indexed over $c \in C$

## Data:

$Demand_{ct}$ = Demand for each customer $c$ in month $t$. $\forall c \in C, \forall t \in T$.

$minAl_c$=minimum (%) of Aluminium for each customer $c$. $\forall c \in C$

$minSi_c$=minimum (%) of Silicon for each customer $c$. $\forall c \in C$

$Tonnage_{ij}$ = Amount of ore in block $(i, j)$. $\forall i \in Row, \forall j \in Col$

$Al_{ij}$ = % of aluminium inside the ore of block $(i, j)$. $\forall i \in Row, \forall j \in Col$

$Si_{ij}$= % of silicon inside the ore of block $(i, j)$. $\forall i \in Row, \forall j \in Col$

$Cost_{ij}$= Cost of mining (per tonne) of block $(i, j)$. $\forall i \in Row, \forall j \in Col$

## Variables:

$X_{ijtc}$ = amount of ore mined from block $(i, j)$ in month $t$, delivered to customer $c$.
$\forall i \in Row, \forall j \in Col, \forall t \in T, \forall c \in C$

$Z_{ijt} \in \{0,1\}$ = 1 if block $(i, j)$ is exhausted at time $t$. $\forall i \in Row, \forall j \in Col, \forall t \in T$

## Objective Value

**Mining Cost = amount mined X Cost of mining**

$$\min\left(\sum_{i \in Row} \sum_{j \in Col} \sum_{t \in T} \sum_{c \in C} X_{ijtc} \times Cost_{ij}\right)$$

## Constraints:

### Trivial Constraints

No negative mining assumes:

$$(X_{ijtc} \geq 0) \forall i \in Row\,, \forall j \in Col\,, \forall t \in T\,, \forall c \in C$$

Note that these do not have to be added into the code as Gurobi can only handle positive variables.

Binary definition

$$Z_{ijt} \in \{0,1\}\ \forall i \in Row\,, \forall j \in Col\,, \forall t \in T$$

Note that this is accounted for using the vtype=GRB.BINARY definition when defining Z.

### Bauxite Delivery to each Customer

$$\left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \right) \geq Demand_{ct} \right) \forall t\ in\ T\,, \forall c \in C$$

Note that the greater than or equal to sign is to help prevent rounding errors in Gurobi as the model will not exceed customers' demands intentionally as this costs more money.

### Minimum Aluminium (%)

The Aluminium percentage is a weighted average of the Aluminium (%) of each deposit weighted against ore mined from a specific deposit as a proportion of total ore mined from all deposits in a given month. Hence,

$$\therefore \left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times Al_{ij} \right) \div \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \right) \geq minAl_c \right) \forall t \in T, \forall c \in C$$

$$\therefore \left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times Al_{ij} \right) \geq \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times minAl_c \right) \right) \forall t \in T, \forall c \in C$$

$$\therefore \left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times (Al_{ij} - minAl_c) \right) \geq 0 \right) \forall t \in T, \forall c \in C$$

### Minimum Silicon (%)

The Silicon percentage is a weighted average of the Silicon (%) of each deposit weighted against ore mined from a specific deposit as a proportion of total ore mined from all deposits in a given month. Hence,

$$\therefore \left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times Si_{ij} \right) \div \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \right) \geq minSi_c \right) \forall t \in T, \forall c \in C$$

$$\therefore \left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times Si_{ij} \right) \geq \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times minSi_c \right) \right) \forall t \in T, \forall c \in C$$

$$\therefore \left( \sum_{i \in Row} \left( \sum_{j \in Col} X_{ijtc} \times (Si_{ij} - minSi_c) \right) \geq 0 \right) \forall t \in T, \forall c \in C$$

### Maximum Mining from each deposit:

Total amount mined from a deposit, for all customers and time must be less than the total amount of ore in a deposit.

$$\left( \sum_{c \in C} \left( \sum_{t \in T} X_{ijtc} \right) \leq Tonnage_{ij} \right) \forall i \in Row, \forall j \in Col$$

## Logical Constraints (Access Rules):

### Make a mine exhausted if it has been mined out

$$if\ \left(cumulative\ mined < Tonnage_{ij}\right)\ then\ (Z_{ijt} == 0)$$

As this is a cumulative sum, define $T_{cumul} = \{0,1,2, \dots, t\}$

$$\left( \sum_{c \in C} \left( \sum_{t_{new} \in T_{cumul}} X_{ijtc} \right) \geq Z_{ijt} \times Tonnage_{ij} \right) \forall t \in T, \forall i \in Row, \forall j \in Col$$

Note here that the converse condition $if\ \left(cumulative\ mined == Tonnage_{ij}\right)\ then\ (Z_{ijt} == 1)$ is not needed as gurobi will set the $Z_{ijt}$ variable to 1 in order to access lower deposits via the block access rule below.

### Preserve the exhaustion of a mined site forward in time

$$if\ \left(Z_{ijt} == 1\right)\ then\ (Z_{ij(t+1)} == 1)$$

To account for boundary conditions, this constraint is not applied in the last month.

$$\left(Z_{ij(t+1)} \geq Z_{ijt}\right) \forall i \in Row, \forall j \in Col, \forall t \in T \backslash \{t_{end}\}$$

### Method 1:

### Can only mine if row above is exhausted in current month

$$if\ \left(Z_{(i-1)jt} == 0\right)\ then\ \left(X_{ijtc} == 0\right)$$

Note that with this constraint, we do not condition the first row as these blocks are always able to be mined from unless exhausted.

$$\left(\sum_{c \in C} X_{ijtc} \leq M \times Z_{(i-1)jt}\right) \forall t \in T, \forall j \in Col, \forall i \in Row \backslash \{i_{start}\}$$

Note the $M$ is a large number much larger than the maximum value of $X_{ijtc}$ and is given the value $Tonnage_{ij} + 1000000$ in the code. The large value of $M$ is used to avoid rounding errors in the binary variable $Z_{(i-1)jt}$. Necessary then is the additional constraint:

$$(X_{ijtc} \leq Tonnage_{ij}) \forall i \in Row, \forall j \in Col, \forall t \in T, \forall c \in C$$

Within the code, this condition is handled with an upper bound on the variable during definition.

## Method 2:

The additional access rules are used in addition to the logical constraints of Method 1.

### Can only mine if up left is exhausted in current month

$$if \ (Z_{(i-1)(j-1)t} == 0) \ then \ (X_{ijtc} == 0)$$

Note that with this constraint, we do not condition the first row as these blocks are always able to be mined from unless exhausted. To account for boundary conditions, we also do not condition the first column as an "up-left" block does not exist for those blocks.

$$\left(\sum_{c \in C} X_{ijtc} \leq M \times Z_{(i-1)(j-1)t}\right) \forall t \in T, \forall j \in Col \backslash \{j_{start}\}, \forall i \in Row \backslash \{i_{start}\}$$

Note the $M$ is a large number much larger than the maximum value of $X_{ijtc}$ and is given the value $Tonnage_{ij} + 1000000$ in the code. The large value of $M$ is used to avoid rounding errors in the binary variable $Z_{(i-1)(j-1)t}$.

### Can only mine if up right is exhausted in current month

$$if \ (Z_{(i-1)(j+1)t} == 0) \ then \ (X_{ijtc} == 0)$$

Note that with this constraint, we do not condition the first row as these blocks are always able to be mined from unless exhausted. To account for boundary conditions, we also do not condition the last column as an "up-right" block does not exist for those blocks.

$$\left(\sum_{c \in C} X_{ijtc} \leq M \times Z_{(i-1)(j+1)t}\right) \forall t \in T, \forall j \in Col \backslash \{j_{end}\}, \forall i \in Row \backslash \{i_{start}\}$$

Note the $M$ is a large number much larger than the maximum value of $X_{ijtc}$ and is given the value $Tonnage_{ij} + 1000000$ in the code. The large value of $M$ is used to avoid rounding errors in the binary variable $Z_{(i-1)(j+1)t}$.

**Applied together, the combination of these constraints yields the correct block selection algorithm in both Methods.**

# Impact of the Additional Access Rules:

## Analysis of Mining Costs (Objective Values) and Strategy

### Minimum Objective Value (Total Mining Cost)

| **Method 1:** | **Method 2:** |
|---|---|
| $21, 457, 317.70 | $24, 461, 548.76 |

The additional access rules increase the cost of mining. The makes intuitive sense as the linear program has more requirements for block selection and thus cannot select the cheapest block to mine from as often as in Method 1.

### Mining Strategy each month (aka Notable Events)

#### Method 1:

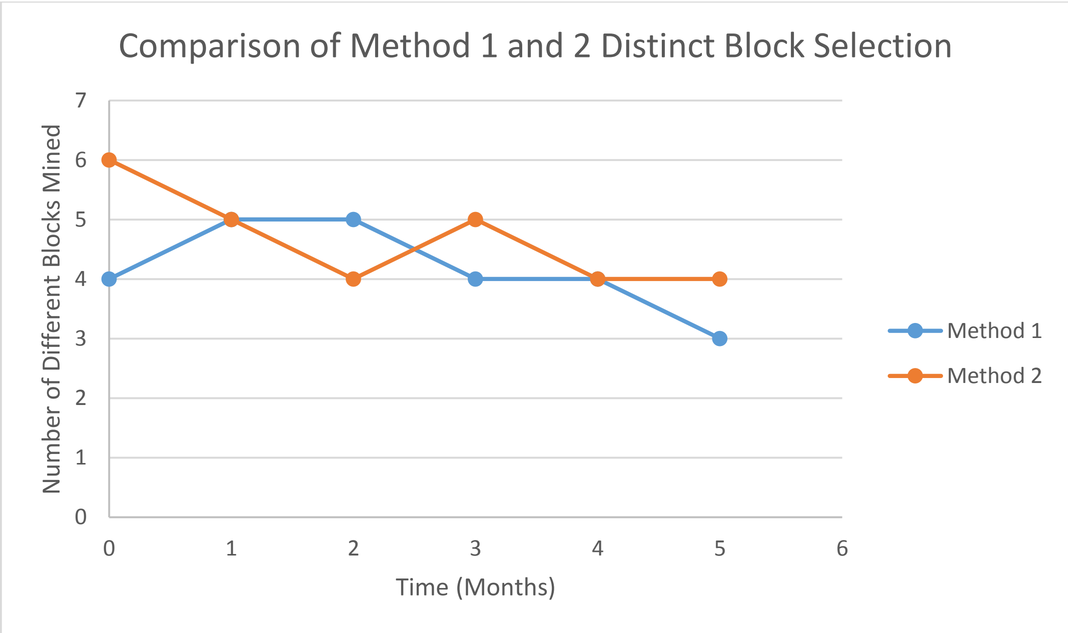
Time 0
55027.9705882 Tonnes from Block (0, 3)
121261.729655 Tonnes from Block (0, 5)
221880.0 Tonnes from Block (0, 6)
94364.2997572 Tonnes from Block (1, 6)
----------------------
Time 1
52507.6393941 Tonnes from Block (0, 2)
45362.9702603 Tonnes from Block (0, 3)
54176.2703454 Tonnes from Block (0, 5)
244317.0 Tonnes from Block (1, 5)
102805.12 Tonnes from Block (1, 6)
----------------------
Time 2
14044.5272727 Tonnes from Block (0, 2)
139910.059151 Tonnes from Block (0, 3)
222204.0 Tonnes from Block (1, 3)
18393.5802428 Tonnes from Block (1, 6)
104283.833333 Tonnes from Block (2, 3)
----------------------
Time 3
153075.833333 Tonnes from Block (0, 2)
136016.0 Tonnes from Block (1, 2)
106447.166667 Tonnes from Block (2, 3)
110605.0 Tonnes from Block (2, 5)
----------------------
Time 4
121363.0 Tonnes from Block (2, 5)
75954.0 Tonnes from Block (2, 6)
101973.0 Tonnes from Block (3, 3)
195872.0 Tonnes from Block (3, 5)
----------------------
Time 5
146338.0 Tonnes from Block (2, 6)
128163.0 Tonnes from Block (3, 3)
211021.0 Tonnes from Block (3, 6)

#### Method 2:

Time 0
50001.0 Tonnes from Block (0, 1)
30360.140671 Tonnes from Block (0, 2)
60973.8593289 Tonnes from Block (0, 3)
54149.0 Tonnes from Block (0, 4)
175438.0 Tonnes from Block (0, 5)
121612.0 Tonnes from Block (0, 6)
----------------------
Time 1
15016.364613 Tonnes from Block (0, 1)
162392.740671 Tonnes from Block (0, 3)
170569.0 Tonnes from Block (0, 4)

100268.0 Tonnes from Block (0, 6)
50922.8947159 Tonnes from Block (1, 6)
----------------------
Time 2
16934.4 Tonnes from Block (0, 3)
130221.333333 Tonnes from Block (1, 4)
187040.161383 Tonnes from Block (1, 5)
164640.105284 Tonnes from Block (1, 6)
----------------------
Time 3
29996.7524082 Tonnes from Block (0, 1)
115790.666667 Tonnes from Block (1, 4)
57276.8386174 Tonnes from Block (1, 5)
125077.384615 Tonnes from Block (2, 5)
178002.357692 Tonnes from Block (2, 6)
----------------------
Time 4
132960.742308 Tonnes from Block (0, 2)
106890.615385 Tonnes from Block (2, 5)
44289.6423077 Tonnes from Block (2, 6)
211021.0 Tonnes from Block (3, 6)
----------------------
Time 5
56307.1170213 Tonnes from Block (0, 2)
222204.0 Tonnes from Block (1, 3)
174923.0 Tonnes from Block (2, 4)
100854.225338 Tonnes from Block (3, 5)

## Number of Distinct Blocks Mined




Interestingly enough, Method 2 prioritises a higher number of distinct blocks mined earlier in the strategy in order to give it access to more preferable blocks in lower rows at the beginning. This is contrasted with Method 1 which uses more tonnage the fewer, most favourable mining sites. The Method 2 selection number decreases over time due to it having access to fewer valid, yet optimal, blocks.

## Analysis of the Block Selection

Table 1 (below) represents the mining selection over time. Entries corresponding to "1", coloured green, represent the blocks which have had any bauxite removed from them at all ($\geq 0.01 tonnes$, in actuality to account for Gurobi's rounding errors). "2", coloured red indicates an exhausted or mined out deposit. The matrix itself corresponds to the rows and columns of the mine face.

| **Time** | | Method 1 | | | | | | | | Method 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **July** | | 0 | 0 | 0 | 1 | 0 | 1 | 2 | | 0 | 1 | 1 | 1 | 1 | 2 | 1 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | | |
| **August** | | 0 | 0 | 1 | 1 | 0 | 2 | 2 | | 0 | 1 | 1 | 1 | 1 | 2 | 2 |
| | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | | |
| **September** | | 0 | 0 | 1 | 1 | 0 | 2 | 2 | | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| | | 0 | 0 | 0 | 2 | 0 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 2 |
| | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | | |
| **October** | | 0 | 0 | 2 | 2 | 0 | 2 | 2 | | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| | | 0 | 0 | 1 | 2 | 0 | 2 | 1 | | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| | | 0 | 0 | 0 | 1 | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | | |
| **November** | | 0 | 0 | 2 | 2 | 0 | 2 | 2 | | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| | | 0 | 0 | 1 | 2 | 0 | 2 | 2 | | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| | | 0 | 0 | 0 | 2 | 0 | 2 | 1 | | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| | | 0 | 0 | 0 | 1 | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | | | | | | | | | | | | | | | | |
| **December** | | 0 | 0 | 2 | 2 | 0 | 2 | 2 | | 0 | 1 | 2 | 2 | 2 | 2 | 2 |
| | | 0 | 0 | 1 | 2 | 0 | 2 | 2 | | 0 | 0 | 0 | 2 | 2 | 2 | 2 |
| | | 0 | 0 | 0 | 2 | 0 | 2 | 2 | | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Table 1 – Cumulative Display of the blocks which are mined from and which are exhausted for both access rule cases

### Accuracy of the Mining Algorithm

Accuracy of the block algorithm may be determined in Table 1 below as a red two should come above every green 1 in method 1. Additionally, a red 2 should come at all upward adjacent blocks to a green 1 in method 2.

Note that all the entries in the matrix appear to correspond to the correct block choosing procedure except for the entry at September, Method 1. Block (0,3). It should be a 2 but it is not. This is due to rounding error on the part of Gurobi.

The cumulative amount mined up to September for Block (0,3) is slightly less than the required value of Tonnage to set $Z[i, j, t] = 1$. And thus the code to generate the matrix displayed it as a 1 instead of a 2 due to rounding errors introduced by Gurobi.

$$\left( \sum_{t \in [0,1,2]} \left( \sum_{c \in C} X_{0,3,tc} \right) = 240300.99999999322 < 240301 = Tonnage[0][3] \right)$$

Hence, the program successfully applied the access rules.

## Impact of the additional access rule

The tendency of Method 1 is to create, as feared, an unstable mine face by digging deep into an individual column where the mining cost is cheaper relative to the percentage of available Aluminium and Silicon.

In December, in the mine face's final state it leaves an untouched column 4 surrounded by removed columns either side, leaving an unsupported structure composed of Blocks$(0,4), (1,4), \& \ (2,4)$. This column is at high risk of collapse (outward left and right) as there is little support of the ore from either side. Method 1 then poses a safety risk to workers at the bauxite mine and would not be recommended.

Ethically it makes sense to keep workers safe. Further, economically death or severe injury at a worksite would encounter legal fees larger than the additional cost ($\$3,004,231.06$) to implement the safer mining strategy with the additional access rules.

The impact of the additional access is the generation of a mine face where each unmined or partially mined block is supported by 2 or more adjacent unmined blocks. These unmined blocks are anchored to the rock face and so the mining pattern preserves stability through time, improving safety. Method 2 also eliminates the production of deep channels, algorithmically ensuring they cannot occur.

The Method 2 digging algorithm tends to favour mining the edge (in this case right edge) of the mining site as there are less required conditions to fill (i.e. rightmost column has no "up-right" necessary block to consider). The assumption required is that outside of the mined area, there is sufficient rock face support to always support the left-most and right-most columns.