

A Programming Approach to Solving the Blast Scheduling problem for Sub – Level Stoping

Based on Advanced Quantitative Decision Making Course

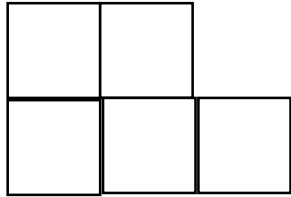
Taken by: Prof. Ashis Bhattacharjee

Prepared by: Arnab Moitra

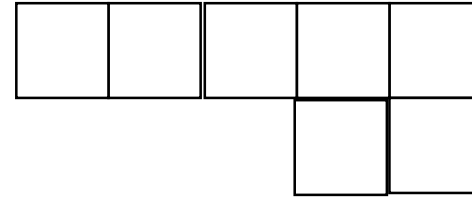
Constraints and Objective Function

We shall go through couple of constraints for this problem, and also would discuss the overall objective function, that we have to optimize.

Pictorial representation of the constraints:



This is not possible.
Since, I have to mine the lower block first in order to mine the one below it



This is not possible.
Due to support constraints, i.e. we can't keep more than 2 blocks without support except if it's not the lowest level.

Additional Considerations:

- a. Every round of blasting, we shall EXACTLY 2 blocks.
- b. The Direction of Blasting shall be mentioned.

Objective Function:

Minimize Cumulative sum of the absolute difference b/w average grade of the blasted ore and overall mean grade, after each round of blasting.

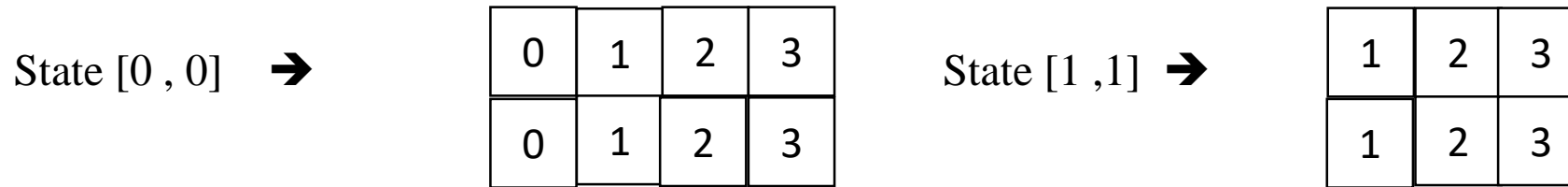
So, we essentially need to find the sequence of the blasting, keeping in mind all the constraints at the same time.

Algorithm Design and Modelling the Problem

Representation of State:

It is very important here to understand how we represent the state. **We here deal with 2 levels.**

The State is represented by 1 dimensional array containing 2 elements [i , j], where this means that upto i-th index 1st level has been mined already, and up to j-th index the 2nd level has been mined.



Here, we used 2 well known techniques for modelling this problem.

- A Simple Recursive Solution [based on it's recursive structure].
- A much more efficient Memoized Solution. [Memoization].

Recursive Solution:

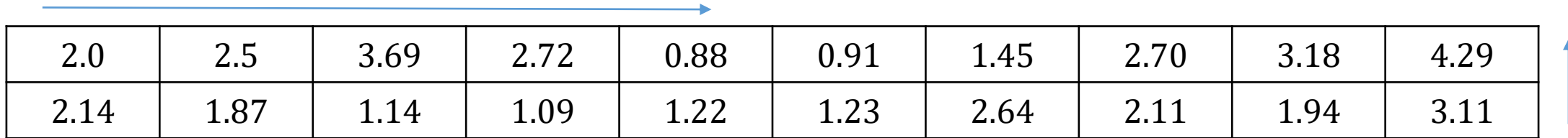
The Logic for the Code is very easy to understand, but a particular state is to be evaluated multiple times.

Memoization Method:

This creates an n-dimensional array, and stores the result [i.e. the Minimum Cumulative Deviation] reaching to state (i,j) from state (0,0).

In the next 3 slides, we shall go through some examples of some problems, which have been solved using computer.

Example - 1



2.0	2.5	3.69	2.72	0.88	0.91	1.45	2.70	3.18	4.29
2.14	1.87	1.14	1.09	1.22	1.23	2.64	2.11	1.94	3.11

Direction of Extraction: Left to Right

Result from our code:

- Mean Grade: 2.14 %
- Minimum Cumulative Deviation :4.58
- Mining Sequence:

((0, 0), (1, 0))
((1, 1), (1, 2))
((0, 1), (1, 3))
((0, 2), (1, 4))
((0, 3), (0, 4))
((1, 5), (1, 6))
((0, 5), (1, 7))
((0, 6), (0, 7))
((0, 8), (1, 8))
((0, 9), (1, 9))

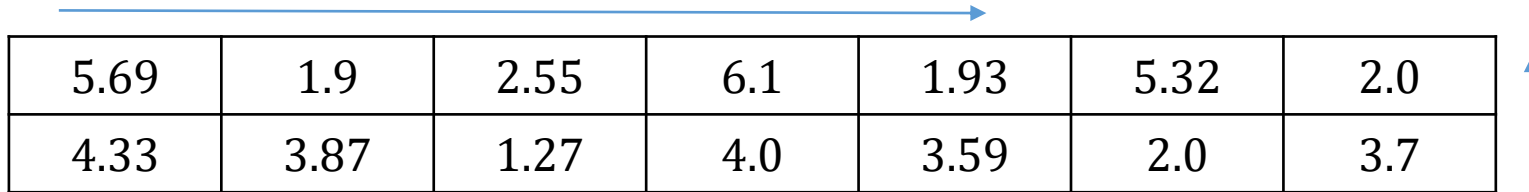
Indexing of the Array [0 based indexing]

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	(0,8)	(0,9)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)

In the Sequence of Mining , the indices are mentioned. During each blasting round 2 blocks are blasted, without violating any constraint.

The one shown is the most optimal policy.

Example - 2



5.69	1.9	2.55	6.1	1.93	5.32	2.0
4.33	3.87	1.27	4.0	3.59	2.0	3.7

Direction of Extraction: Left to Right

Result from our code:

- Mean Grade: 3.45 %
- Minimum Cumulative Deviation :2.94
- Mining Sequence:

((1, 0), (1, 1))
((0, 0), (1, 2))
((0, 1), (1, 3))
((0, 2), (1, 4))
((0, 3), (0, 4))
((0, 5), (1, 5))
((0, 6), (1, 6))

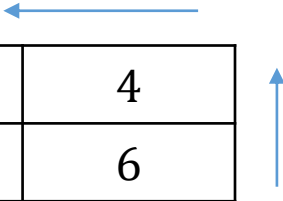
Indexing of the Array [0 based indexing]

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)

In the Sequence of Mining , the indices are mentioned. During each blasting round 2 blocks are blasted, without violating any constraint.
The one shown is the most optimal policy.

Example - 3

9	5.6	6	5	4
8	6	5	7	6



Direction of Extraction: Right to Left

Result from our code:

- Mean Grade: 3.45 %
- Minimum Cumulative Deviation :3.96
- Mining Sequence:
 - ((0, 0), (1, 0))
 - ((0, 1), (1, 1))
 - ((0, 2), (1, 2))
 - ((1, 3), (1, 4))
 - ((0, 3), (0, 4))

Indexing of the Array [0 based indexing]

(0,4)	(0,3)	(0,2)	(0,1)	(0,0)
(1,4)	(1,3)	(1,2)	(1,1)	(1,0)

In the Sequence of Mining , the indices are mentioned.

During each blasting round 2 blocks are blasted, without violating any constraint.

The one shown is the most optimal policy.

Further Steps

The algorithm we have designed here, is only applicable for situation where there are 2 levels. In terms of overall time and space efficiency, the Memoization Method essentially does better than the Simple Recursive method.

Below are the some of the points to ponder about:

- a. The overall algorithm design and overall performance essentially depends upon the way we choose to represent the state. With our method of representation, with increasing number of levels the complexity would increase exponentially.
So, the fundamental point to ponder about is: **Does there exist some superior representation of the states, that shall help us achieve computational efficiency ?**
- b. We haven't considered any **Probabilistic scenario**, for instance let's say the grade of a particular block is varying from a certain value to a certain value, and the corresponding distribution could be Exponential, Gaussian or even Uniform.
- c. We have taken, the entire block to be mined! But it's fundamentally not that much true, what if we can actually extract some portion of the block and not the block entirely. It can induce so much complexity.

Having said that, this particular Dynamic Programming technique is a very efficient method to solve any particular problem, that has certain number of stages and steps involved. This essentially can be improved so as to model, much more complex situations.