Software Impacts 22 (2024) 100704

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

SlabCutOpt: A code for ornamental stone slab cut optimization

S. Bonduà [a,*], S. Focaccia [a], M. Elkarmoty [b]

[a] DICAM – University of Bologna, via Terracini, 28 40131 Bologna, Italy
[b] Cairo University, Faculty of Engineering, Department of Mining, Petroleum, and Metallurgical Engineering, 1 Gamaa Street, 12613, Giza, Egypt

ARTICLE INFO

Keywords:
Slabs
Optimization algorithm
Rock
Ornamental stone
Triangle intersection

ABSTRACT

Rock masses are naturally affected by discontinuities, joints and fractures that affect their exploitation. After block extraction, different cutting pattern can produce different recovery ratio of the block. The optimization of the cutting pattern can be computed if the discontinuities are mapped by the use of non-destructive methods. We propose a software code able to compute the number of intersected slabs by different cuttings scenarios. The algorithm adopts a brute force computation of several scenarios as specified by the user. The software uses the Open MP library in order to reduce computation time.

Code metadata

Current code version	V1.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2024-149
Permanent link to reproducible capsule	
Legal code license	GNU GPLv3
Code versioning system used	none
Software code languages, tools and services used	C++, OpenMP.
Compilation requirements, operating environments and dependencies	
If available, link to developer documentation/manual	https://github.com/stebond/SlabCutOpt/blob/main/README.md
Support email for questions	stefano.bondua@unibo.it

1. Motivation and significance

Ornamental stones production needs high-quality rock mass for block production. Among several possible uses of the extracted rock blocks, one of the most common is to cut them to obtain slabs. The natural fractures and discontinuities, usually present in rock masses, lower the recovery ratio of blocks and the number of slabs produced. At quarry scale, several researches and algorithms have been presented in literature. These algorithms are mainly focused on finding an optimum pattern for block extraction, in order to minimize waste and increase the recovery of the final products. Furthermore, the dimension stone market offers a variety of slabs dimensions with a correlated selling price. As shown in the work of [1], a linear correlation between the selling price and the slab dimension (surface area) was found for a limestone product.

Related works in literature about cutting pattern optimization are mainly developed at the quarry scale. In [2], the maximum block dimension is evaluated on the basis of a fracture model based on discontinuities' set, aiming at finding the theoretical maximum number of blocks. In [3], the discontinuities are considered as cutting planes and the volume of interest is divided on the basis of these fractures. The total amount of recoverable blocks is determined by a graphical procedure. In [4], the fracture and discontinuities are modeled as planes and the blocks' production is optimized by choosing the block dimension and determining the intersected blocks, considered as waste. In [5], the maximum cuboid volume of the block is computed by modeling the fractures as planes and using a Genetic Algorithm approach for finding the best solution. In [6], fractures and discontinuities are used as input and the algorithm classifies the resulting blocks on the basis of their

DOI of original article: https://doi.org/10.1016/j.resourpol.2019.101533.
* Corresponding author.
E-mail address: stefano.bondua@unibo.it (S. Bonduà).

https://doi.org/10.1016/j.simpa.2024.100704
Received 18 June 2024; Received in revised form 21 August 2024; Accepted 28 August 2024

volume. In this approach, it is considered that an intersected block can still have residual utilization.

A detailed literature review about algorithms and codes related to stone cutting optimization can be found in [6].

The SlabCutOpt software can compute the number of slabs that can be obtained from a dimension block when a fracture model is available. A detailed overview about the fracture model and how to build it can be found in [7–15]. SlabCutOpt generates several cutting pattern scenarios that can be used to cut a block by several virtual displacements and rotations or by using several slabs dimensions. It computes the number of intersected/not intersected slabs for all the generated scenarios. The generation of the scenario often implies the generation of a high number of solutions, but the use of practical displacement, rotation steps and the use of a parallelization method, allows to evaluate the cutting grid in a reasonable computing time. The particular nature of the problem (discrete fracture) does not fit the classical approach of optimization of multi-variable problems algorithms (linear or non-linear optimization).

The software needs a fracture model, consisting in a set of surfaces representing the fractures, as input. Users must define the dimensions of the slabs, or a set of possible dimensions, in order to obtain a complete overview of the recovery ratio that can be obtained from the block under examination.

2. Software description

SlabCutOpt is a command line software program of high-level functions and data structure, written in C++ language, for generating, visualizing and analyzing block cutting grid scenarios. Cutting grid scenarios are automatically generated by SlabCutOpt through cutting grid parameters as defined by the user. All generated scenarios are computed for fracture detection and results are given in ASCII text files and vtu (VTK unstructured grid) file format for visualization in Paraview, outside the functional core provided by the SlabCutOpt software.

2.1. Software architecture

The software is composed of several modules: (1) input module; (2) solution generator; (3) intersection computation; (4) output module. Each module is composed by several functions.

2.1.1. Input module

The input module read from external files: (i) the parameters of the cutting grid; (ii) the fracture model. The input of SlabCutOpt is composed of several ASCII files:

- SlabCutOpt.par: the parameters file. Parameters are defined by keywords. This file contains the cutting grid size, the step and the maximum displacement for each direction, the step and the maximum rotation angles, thickness of the saw, etc. (Appendix A);
- SlabDimension.dat: optionally, the solution generator can consider a variety of commercial slab dimensions as specified by the user. An example can be found in Appendix D;
- PlyFileList.dat: it contains the list of the ply file representing the fracture model. Each file can contain one or more surfaces. Surfaces have to be digitalized by a triangular mesh. An example can be found in Appendix C.

2.1.2. Solution generator

The solution generator module generates, on the basis of the parameters of the cutting grid, the whole set of scenarios. The scenarios set store, in a structured array, all the parameters of the generated cutting grid and the results as computed by the intersection computation algorithm. The array containing the scenarios set is generated before running the intersection algorithm, to take advantage of the intersection's computation algorithm.

2.1.3. Intersections computation algorithm

Each cutting grid scenario will hypothetically generate slabs. Each slab is tested against the fracture model intersection, in order to evaluate if it can be considered as an exploitable slab or waste. The algorithm assumes that the slabs are orthogonal prisms, with 6 faces and 12 edges each. As the surfaces fractures are represented by triangles, a fast intersect segment/triangle algorithm is used [16]. If one of the edges of the slab is intersecting a triangle of the fracture model, then the slab is marked as "intersected" and considered as waste.

2.1.4. Output module

There are two kinds of output files produced by SlabCutOpt. The first one contains the quantitative evaluation of the number of intersected slabs (see Appendix E). The second one, optional, is used for the graphical representation of the cutting pattern using the open-source software Paraview. The visualization of the output files can be limited to the best solution found, or a complete file set can be generated for all the computed scenarios and for all slabs dimensions.

2.2. Software functionalities

Once the input file has been read, the solution generator will create the whole set of solutions by iterating all the displacements, rotations and slab dimensions. This array of scenarios is then processed by using the OpenMP library, to allow a fast computation through the directive *#pragma omp parallel for*, by distributing the computation in several threads, depending on the hardware architecture.

When the array of the scenarios results is completely filled by the intersection computation algorithm, the computation is completed and the results are written to a file. Optionally, it is possible to generate a vtu file, for results visualization by the open-source software Paraview.

3. Illustrative example

We present a dimension block with a fracture model obtained as described in [17]. The dimensions of the block are 0.90 × 2.00 x 1.10 m^3. The block is affected by several fractures, as shown in Fig. 1.

For this limestone rock type, typical slab commercial size dimensions vary in range between 0.20 m up to 0.80 m, while the thickness range can vary between 0.02 m to 0.04 m.

The surfaces representing the discontinuities have been coded as ply files, and each ply file contains triangle elements representing the whole surface. An extract of a ply file is given in Appendix B.

The algorithm also takes into account the dimension of the saw, which in this example is assumed to be 0.01 m. For this example, we considered a set of 37 slab dimensions. The dimensions are defined in a file named BlockDimension.dat. See Appendix D for an example.

For this rock type, there are not preferred cutting orientations, so the block can be cut in any orthogonal direction. The displacement step has been set to 0.05 m, from a minimum of $-\frac{l_i}{2}$ up to $+\frac{l_i}{2}$, where l is the slab dimension and i=x, y, z, referring to the 3 dimensions.

The material considered is isotropic, so slabs can be cut in every direction.

The solution with a high number of recoverable slabs is shown in Fig. 2.a, while the solution with the lower number of recoverable slabs is shown in Fig. 2.b. The slabs are marked with a rock type identifier: "0" for non-intersected slabs (no intersections with fractures model), "1" if there is an intersection with the fracture model. In the latter case the slab is considered as waste. Obviously, higher slab dimensions increase the probability of an intersection with the fractures, while smaller slabs increase the number of recoverable slabs, but also the cost for cutting, dust emissions, water consumption etc. For a comparison of these aspects, please refer to [8].

A graphical representation of the recovery ratio by different slab surface is presented in Fig. 3. The recovery ratio, computed as the ratio between volume of recoverable slabs and the total volume of the

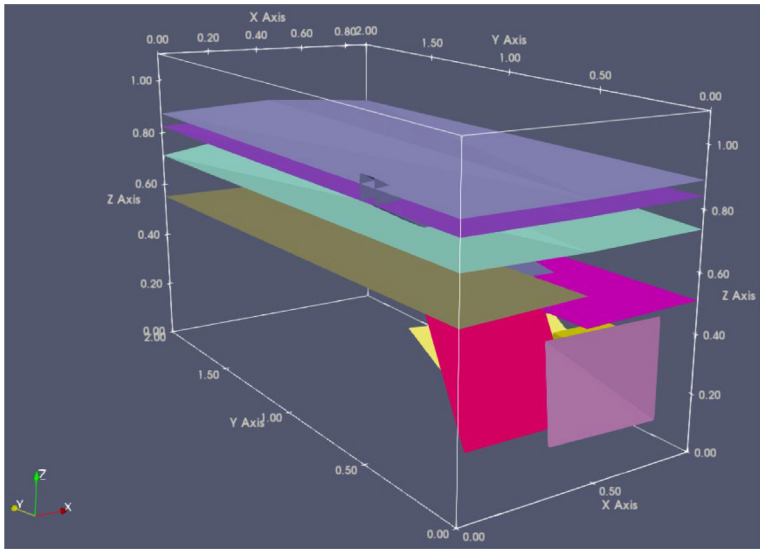

Fig. 1. 3D Fracture model inside a limestone block.

block, is higher for limited slab surface (small slabs). The recovery ratio strongly decreases for slabs with higher surface area, as, from a probabilistic point of view, the slabs can have higher probabilities of intersecting the fracture network.

From an economic point of view, higher recovery ratio does not imply directly a higher revenue for the production. In fact, other parameters, function of the slab size, are involved, such as power and water consumption, dust generation and so on. The evaluation of these parameters is out of the scope of this paper. An evaluation of the economic and environmental issue related to the optimization cut of dimensional stone can be found in [8].

4. Impact

Ornamental stones blocks contain natural discontinuities and fractures. The definition of the cutting pattern of these blocks is a crucial phase of slabs production. Up to date, the cutting pattern is operated by quarry operators based on empirical method and experience. SlabCutOpt represents a possible solution for improving and speeding up the definition of the cutting pattern, thus improving the operators' overall experience.

SlabCutOpt has been proven to be an effective software that can assist operators in assessing the number of slabs that can be produced by several cuttings scenarios. Additionally, it is a helpful tool in evaluating the best economic revenue and the lowest environmental impact of the cutting operations, by associating its computational results with other parameters, such as the selling price for each size, the cost of the produced waste, the time needed for the cut, the total amount of dust and the electricity consumption, among others.

The impact of the proposed software extends beyond the industrial setting to encompass the academia and research spheres. While ornamental stone industry can use the software in processing plants, researchers studying ornamental block cutting optimization can easily make reference to SlabCutOpt to solve their problems. Furthermore, the code can be easily adapted to other cutting optimization problems, as well as in numerical modeling problems where surface/grid blocks intersection detection is needed. This adaptability positions the software as a valuable asset with implications across a spectrum of domains where cutting optimization play a crucial role.

Moreover, the run of SlabCutOpt can positively affect on societies nearby quarries by means of waste reduction and maximization of profit. Thus, SlabCutOpt software considers sustainability in environment, economy, and society cores.

5. Conclusions

The software package *SlabCutOpt* provides a set of functions and data structure for parsing ply files and detected slabs/surfaces detection, as well as for the generation, visualization and analysis of the cutting grids scenarios. The use of C++ and of the OpenMP library allows computing an elevated number of scenarios in a reasonable computational time. The software is released under the GPL2.0 license, allowing for the integration of the SlabCutOpt in commercial software.

CRediT authorship contribution statement

S. Bonduà: Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **S. Focaccia:** Writing – review & editing, Visualization, Validation, Data curation. **M. Elkarmoty:** Writing – review & editing, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Funding source

This work has been developed under the framework ERA-MIN3, financed by the Italian Ministry of Instruction and Research (MIUR), CUP: J35F21004410006.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

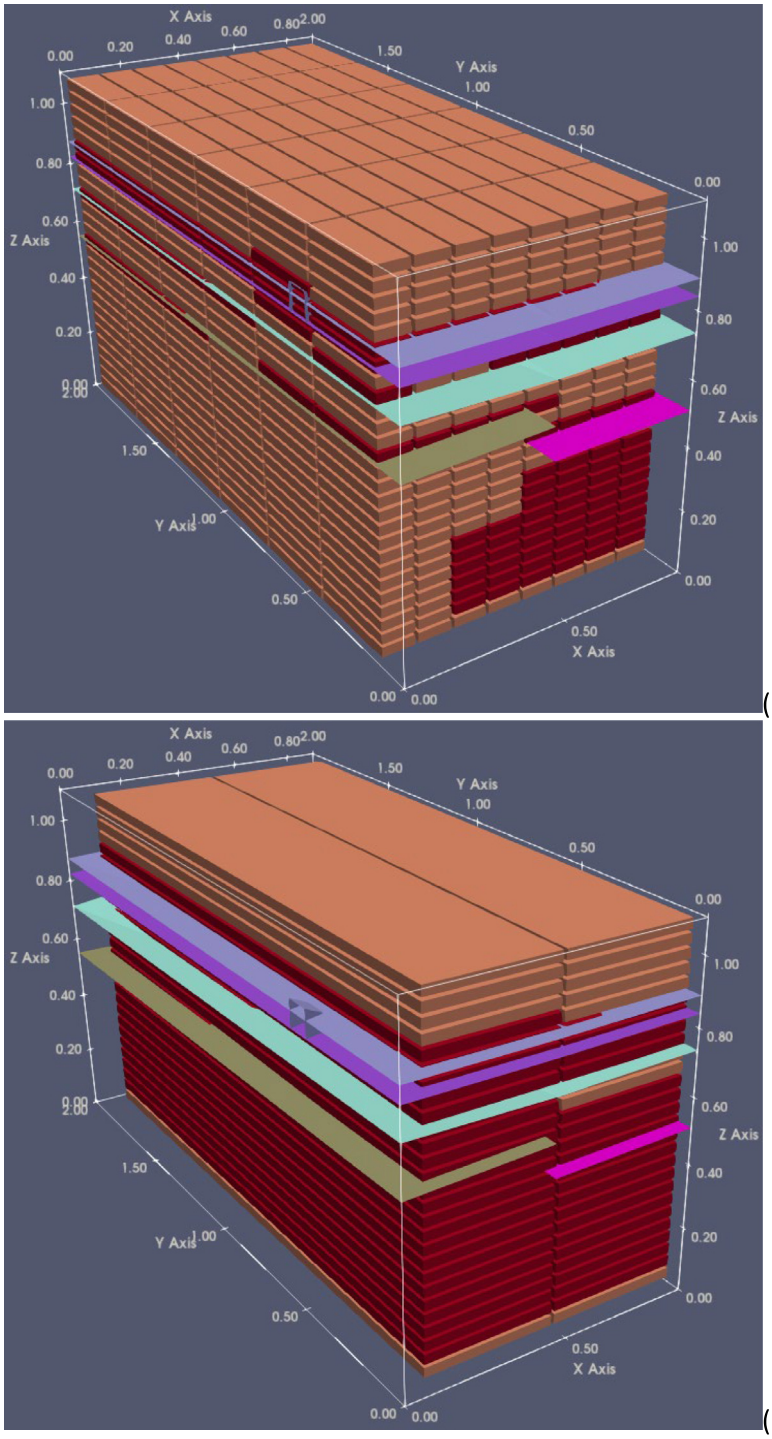

Fig. 2. Visualization of the results of SlabCutOpt: (a) Solution n.22, (slab size 0.30 × 0.10 x 0.03 m^3); (b) solution n.28 (slab size 0.40 × 1.90 x 0.03 m^3).

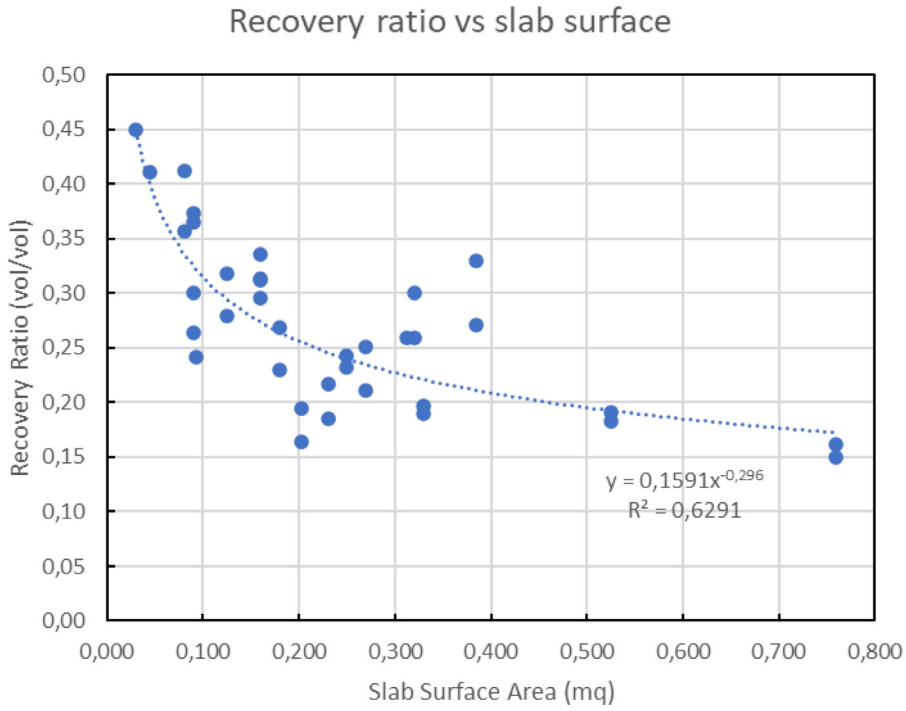

Fig. 3. Recovery ratio by slab surface. As expected, the recovery ratio is higher for small slab surface.

Appendix A. Parameter file

```
************************************************
*         DICAM - University of Bologna        *
*          SlabCutOpt - ver.1.0                *
************************************************
*A computer code for Slabs Cutting Optimization *
************************************************
*Authors:                                      *
*Stefano Bondua, Sara Focaccia, Mohamed Elkarmoty*
************************************************
*For any information please contact:           *
*                 stefano.bondua@unibo.it       *
************************************************
*Parameters                                     *
************************************************
x_max=0.900000
x_min=0.000000
y_max=2.000000
y_min=0.000000
z_max=1.100000
z_min=0.000000
tetha_step=1.570796
tetha_max=1.570796
phi_max=1.570796
phi_step=1.570796
csi_max=1.570796
csi_step=1.570796
n_x_division=1
n_y_division=1
n_z_division=1
dx_step=0.050000
dy_step=0.050000
dz_step=0.050000
dx_max=0.200000
dy_max=0.200000
dz_max=0.200000
dim_block_x=0.410000
dim_block_y=0.210000
dim_block_z=0.040000
read_block_dimension=1
read_bound=0
BiDimensional=0
write_vtu=2
read_PLY_FileList=1
n_x_division=1
n_y_division=1
n_z_division=1
rotation_method=1
cut_saw_thickness=0.010000
angle_type=0
bound_tolerance=0.00100000000000000
end=end
************************************************
*               END PARAMETERS                 *
************************************************
```

Appendix B. PLY file format

The PLY file format readable by the SlabCutOpt software is a simplified version of the more general PLY file format, such as the example below. Each surface element must have 3 vertices (a triangle in the 3D space). The fixed characters are presented in italic font, the geometrical parameters for a fracture description are presented in bold. After the header, the number specified in [*element vertex*] indicates the number of vertices that are stored in the ply file. The [*property float*] specify the format of the coordinates of vertices, then the following [*element face*] lines define the number surface elements and the vertices indexes of each triangle. The [*property list*] define the format of the indexes of the triangle vertexes. The keyword [*end_header*] close the header section. In the presented example, after the header there are 4 lines containing the x, y, z coordinates of each vertex. Following the 4 vertexes coordinates, there are 2 lines, with the first number indicating "3"; so the number of indexes is 3 (a triangle) and the indexes refer to the vertex coordinate index of the previous lines.

Is worth mentioning that, in the more general PLY file format, the number of vertices can be higher for polygonal elements.

```
ply
format ascii 1.0
comment example of a fracture represented by 2 triangles
element vertex 4
property float x
property float y
property float z
element face 2
property list uchar int vertex_index
end_header
0.000   2.000   0.553
0.450   2.000   0.553
0.000   0.000   0.583
0.450   0.000   0.599
3 0 1 2
3 1 2 3
```

Appendix C. PLY FileList

The PLY_FileList.dat input file must contain the list of the ply files of the modeled discontinuities. No header is allowed. Below is an example of PLY_FileList.dat.

```
block_fracturepink_g.ply
block_fracturepink_h.ply
block_fracturepink_k.ply
brown_frac.ply
extended_void_blue.ply
purple_frac.ply
red_frac.ply
single_void.ply
turquoise_frac.ply
yellow_frac.ply
```

Appendix D

The slab_dimensions.dat input file must contain the list of the different sizes of slabs that the algorithm must test. Below an example of slab_dimensions.dat is shown. The order of the columns, from left to right, is dim_x, dim_y, dim_z. The dimensions of the slabs must be increased by the "cut_saw_thickness" parameters for geometrical considerations in the algorithm.

```
0.41    0.21    0.025
0.51    0.26    0.025
0.41    0.21    0.03
0.51    0.26    0.03
0.61    0.31    0.025
0.31    0.31    0.025
0.41    0.41    0.025
0.61    0.46    0.025
0.46    0.46    0.025
0.49    0.49    0.025
0.61    0.31    0.03
0.31    0.31    0.03
0.41    0.41    0.03
0.61    0.46    0.03
0.46    0.46    0.03
0.49    0.49    0.03
```

Appendix E. Output file – results

The results file is composed of several sections:

- Resume of the input parameters used;
- List of the input dimensions of the slabs;
- Results for each scenario of each dimension;
- Resume of the best solution for each dimension.

Example:

Resume of the input parameters used:

```
************************************************
*          DICAM - University of Bologna       *
*           SlabCutOpt - ver.1.0               *
************************************************
*A computer code for Slabs Cutting Optimization *
************************************************
*Authors:                                       *
*Stefano Bondua, Sara Focaccia, Mohamed Elkarmoty*
************************************************
*For any information please contact:            *
*                  stefano.bondua@unibo.it       *
************************************************
*Parameters                                      *
************************************************
x_max=0.900000
x_min=0.000000
y_max=2.000000
y_min=0.000000
z_max=1.100000
.....
```

List of the input dimensions of the slabs

```
Number of dimensions to test: 37
[0] dim_block_x= 0.410000 dim_block_y= 0.210000 dim_block_z= 0.025000
[1] dim_block_x= 0.510000 dim_block_y= 0.260000 dim_block_z= 0.025000
[2] dim_block_x= 0.410000 dim_block_y= 0.210000 dim_block_z= 0.030000
.....
N_iter i_x_zone i_y_zone i_k_zone dim_block_x dim_block_y dim_block_z
Tetha Phi Csi dx dy dz n_block_inside n_block_no_intersect
```

Results for each scenario of each dimension

```
Results_for_slab_[0] (0.410000 0.210000 0.025000)
 0 0 0 0 0.410000 0.210000 0.025000 0.000000 0.000000 0.000000 -0.205000 -0.105000 -0.012500 387 275
 1 0 0 0 0.410000 0.210000 0.025000 0.000000 0.000000 0.000000 -0.205000 -0.055000 -0.012500 387 277
.....
```

Resume of the best solution for each dimension

```
Optimum solution for slab_[0] is solution[24], no_intersected_block=589
Optimum solution for slab_[1] is solution[440], no_intersected_block=295
Optimum solution for slab_[2] is solution[912], no_intersected_block=510
.....
Elapsed time 135817 (ms)
```

References

[1] M. Elkarmoty, S. Bonduà, R. Bruno, A 3D optimization algorithm for sustainable cutting of slabs from ornamental stone blocks, Resour. Policy 65 (2020) 101533, http://dx.doi.org/10.1016/J.RESOURPOL.2019.101533.

[2] M. Fernández-de Arriba, M.E. Díaz-Fernández, C. González-Nicieza, et al., A computational algorithm for rock cutting optimisation from primary blocks, Comput. Geotech. 50 (2013) 29–40, http://dx.doi.org/10.1016/j.compgeo.2012.11.010.

[3] M. Mutlutürk, Determining the amount of marketable blocks of dimensional stone before actual extraction, J. Min. Sci. 43 (2007) 75–80, http://dx.doi.org/10.1007/s10913-007-0008-4.

[4] S. Mosch, D. Nikolayew, O. Ewiak, et al., Optimized extraction of dimension stone blocks, Environ. Earth Sci. 63 (2011) 1911–1924, http://dx.doi.org/10.1007/s12665-010-0825-7.

[5] E. Ülker, A. Turanboy, Maximum volume cuboids for arbitrarily shaped in-situ rock blocks as determined by discontinuity analysis—A genetic algorithm approach, Comput. Geosci. 35 (2009) 1470–1480, http://dx.doi.org/10.1016/j.cageo.2008.08.017.

[6] R. Yarahmadi, R. Bagherpour, S.G. Taherian, et al., Discontinuity modelling and rock block geometry identification to optimize production in dimension stone quarries, Eng Geol 232 (2018) 22–33, http://dx.doi.org/10.1016/j.enggeo.2017.11.006.

[7] M. Elkarmoty, S. Bonduà, R. Bruno, A 3D optimization algorithm for sustainable cutting of slabs from ornamental stone blocks, Resour. Policy (2020;) 65, http://dx.doi.org/10.1016/j.resourpol.2019.101533.

[8] E. Şirin, S. Bonduà, M. Elkarmoty, Environmental and economic optimization for block cutting of dimension stones in a limestone quarry, Resour. Policy (2021) 74, http://dx.doi.org/10.1016/j.resourpol.2021.102396.

[9] M. Elkarmoty, C. Colla, E. Gabrielli, et al., Deterministic three-dimensional rock mass fracture modeling from geo-radar survey: A case study in a sandstone quarry in Italy, Environ. Eng. Geosci. 23 (2017) 313–330, http://dx.doi.org/10.2113/gseegeosci.23.4.314.

[10] M. Elkarmoty, F. Tinti, S. Kasmaeeyazdi, et al., 3D modeling of discontinuities using GPR in a commercial size ornamental limestone block, Constr. Build. Mater. 166 (2018) 81–86, http://dx.doi.org/10.1016/j.conbuildmat.2018.01.091.

[11] M. Elkarmoty, C. Colla, E. Gabrielli, et al., A combination of GPR survey and laboratory rock tests for evaluating an ornamental stone deposit in a quarry

bench, in: Procedia Engineering, 2017, http://dx.doi.org/10.1016/j.proeng.2017.05.272.

[12] M. Elkarmoty, F. Tinti, S. Kasmaeeyazdi, et al., Implementation of a fracture modeling strategy based on georadar survey in a large area of limestone quarry bench, Geosciences (2018) 8, http://dx.doi.org/10.3390/geosciences8120481.

[13] M. Elkarmoty, C. Colla, E. Gabrielli, et al., Mapping and modelling fractures using ground penetrating radar for ornamental stone assessment and recovery optimization: Two case studies, Rudarsko Geolosko Naftni Zbornik (2017) 32, http://dx.doi.org/10.17794/rgn.2017.4.7.

[14] A. Turanboy, E. Ülker, LIP-RM: An attempt at 3D visualization of in situ rock mass structures, Comput. Geosci. 12 (2008) 181–192, http://dx.doi.org/10.1007/s10596-007-9077-3.

[15] A. Turanboy, E. Ulker, LIP-RM: An attempt at 3D visualization of in situ rock mass structures, Comput. Geosci. 12 (2008) 181–192, http://dx.doi.org/10.1007/s10596-007-9077-3.

[16] P. Guigue, O. Devillers, Fast and robust triangle-triangle overlap test using orientation predicates, J. Graph. Tools 8 (2003) 25–32, http://dx.doi.org/10.1080/10867651.2003.10487580.

[17] M. Elkarmoty, F. Tinti, S. Kasmaeeyazdi, et al., 3D modeling of discontinuities using GPR in a commercial size ornamental limestone block, Constr. Build. Mater. (2018) ,166, http://dx.doi.org/10.1016/j.conbuildmat.2018.01.091.