Cloud Application For Sheet Materials Cutting Optimization

Igor Ilin
Graduate school of Business and Management
Peter the Great St.Petersburg Polytechnic University
St. Petersburg Russia
ivi2475@gmail.com

Aleksander Kubarskii
Graduate school of Business and Management
Peter the Great St.Petersburg Polytechnic University
St. Petersburg Russia
sasha.kub95@mail.ru

Peter Cornelis Schuur
Industrial Engineering and Business Information Systems
University of Twente
Enschede Overijssel Netherlands
p.c.schuur@utwente.nl

Aleksander Lepekhin
Graduate school of Business and Management
Peter the Great St.Petersburg Polytechnic University
St. Petersburg Russia
lepekhinalexander@gmail.com

Alissa Dubgorn
Graduate school of Business and Management
Peter the Great St.Petersburg Polytechnic University
St. Petersburg Russia
alissa.dubgorn@gmail.com

ABSTRACT

The purpose of this study is to provide a working SaaS application that can be used for sheet materials cutting optimization using heuristics. For the research, literature and open sources in the internet were analyzed. The list of the most common and well-known applications for cutting optimization was formed. The cloud and on premises applications were briefly compared and cloud solutions advantages were highlighted. Few algorithms were chosen and implemented using JavaScript on server-side and deployed, using Amazon Web Services. However, there is a great room for improvement for methods used and implementation of algorithm for automatic cutting method choosing.

CCS CONCEPTS

• Networks → Network services → Cloud computing • Software and its engineering

KEYWORDS

Cloud computing, SaaS, Cutting optimization

ACM Reference format:

Igor Ilin, Aleksander Kubarskii, Peter Cornelis Cornelis Schuur, Aleksander Lepekhin and Alissa Dubgorn. 2020. Cloud Application For Sheet Materials Cutting Optimization. *In SPBPU DTMIS '20: Proceedings of Peter the Great St. Petersburg Polytechnic University International Scientific Conference "Digital Transformation on Manufacturing, Infrastructure and Service", November 18–19, 2020, Saint – Petersburg, Russia. ACM, New York, NY, USA, 5 pages.*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
DTMIS '20, November 18–19, 2020, Saint Petersburg, Russian Federation
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8890-0/20/11...$15.00
https://doi.org/10.1145/3446434.3446469

https://doi.org/10.1145/3446434.3446469

1 Introduction

For the modern economy it is usual to use various advanced technological processes and methods minimizing losses in production. Organizations are trying to gain competitive advantages in the competitive struggle by minimizing material costs and using new technologies. As a rule, this can be achieved through the implementation of mathematical methods in production and IT-technologies as well [1].

In one of the classes of combinatorial optimization problems, which is quite common in real production conditions, the tasks of cutting and packing are highlighted. They are united by the need to establish a certain correspondence between two groups, as a rule, large and small objects. This problem is discussed in [2].

The tasks of cutting packaging have a different applied interpretation. The most frequently encountered tasks are orthogonal packing and cutting, where small objects are rectangular billets - rectangles or boxes of various sizes, and large ones - material coming in the form of strips, rolls, rectangular sheets, rods or containers of various capacities. These tasks are a problem of both theoretical and practical terms, which has attracted attention of many researchers and manufacturers. The reason for the growing interest in the cutting-packing tasks is their diversity, complexity and wide applicability of the results.

The tasks of cutting-packaging belong to the class of NP-difficult combinatorial optimization problems [3] - there are no algorithms of polynomial complexity for finding the optimal solution, the exact result in the general case can be obtained only in exponential time.

Classical linear programming methods for solving packaging and cutting problems in single production conditions are difficult to apply. In addition, the tasks of cutting in the conditions of single production arise from individual production and, as a rule, from expensive materials.

Precise and pseudo-precise methods of solving, in some way or another, relate to the method of branches and boundaries. To use them, you need to know the lower bounds of the solution or have an algorithm for calculating them. However, the exact lower bound of the solution has not found yet in order to use both precise and approximate methods based on the method of branches and boundaries.

Considering the complexity of algorithms of linear programming methods and other exact methods, it should be mentioned that it is impractical to use them in production, particularly for the large amount of data, and there is no reason to use them in SaaS applications as it may cause an increment in processing time and costs as well.

The justification of development of cloud solution for cutting optimization refers to the increasing popularity of cloud computing [4] and the ability to provide multitenant architecture with advanced role system and flexibility of companies’ IT architecture [5].

2 Materials and Methods

Firstly, we investigated the market of the cutting optimization applications. We also analyzed methods for cutting optimization used in these applications. The pros and cons of the applications selection were highlighted for further analysis.

Next the advantages of the cloud computing comparing to on premises software were studied. The following criteria were considered: scalability, flexibility, tenants, payment model, reliability.

Then we made a research of optimization methods that can be applied in cutting optimization and chose methods that are applicable for usage in SaaS applications. When SaaS software is developed, it is necessary to take into account that used optimization algorithms should be fast and involve a minimum of computer time. Thus, a critical step in the development of such software, is the definition of optimization algorithm. Possible for usage algorithms were separated into the following categories: heuristic algorithms, metaheuristic, neural networks, linear programming, pseudo-exact algorithms.

3 Results

Currently there is a large amount of software that uses various methods to optimize cutting. The following are the most well-known software for cutting optimization.

PRO100 – simple and clear – the program can be used for the design of furniture with the possibility of its virtual placement in a particular interior. It works with complex parts: radial, asymmetrical, with oblique sections. It has clear interface, 3D visualization which is indispensable for both the designer and the sales manager, tools for building an interior, there is a function of calculating the estimated cost of future furniture.

Astra – furniture designer – program specifically designed for small and medium businesses. With its help, it is possible to design furniture complexes or individual parts, save projects created in a special library, have virtual furniture in a 3D interior. In fact, this is an analogue of the PRO100 program: a cheaper software with improved cutting and additive functions, but with much lower design possibilities.

KitchenDraw – specialized program for modeling and design of kitchen furniture, as well as – bathrooms. Allows you to create complex professional projects for non-standard layouts, “drawing” the interior from scratch to the smallest detail. A typical kitchen in it can be designed in a few minutes, using the extensive catalogs of libraries, adjusting the size and geometry of the furniture.

Master 2 provides users great opportunities not only in the drafting of cutting, but also in the conduct of business. Multi-user mode is supported, sorting and systematization of the entered information is present, data on materials and counterparties are saved.

The implementation of the warehouse will help to always be aware of the remaining amount of materials. There is a distribution to the tables where the active orders are located, scheduled and archive, all information is available to the administrator for viewing and editing.

Cutting 3 – has a huge selection of materials and parts. It is more suitable for individual use. The user will only need to enter the required dimensions, select materials and specify additional settings, if necessary.

The listed above programs use heuristics methods to build a cutting plan. They all have clear user interface and are built for furniture parts cutting optimization. There are other industry solutions not only for furniture but for textile, glass or metal. Only Master 2 provides ability to be deployed on the server to be accessed by several users at once. Others do not provide server deployment and integration with existing software. As a result, they are not suitable for production purposes. Moreover, this software was designed to process furniture and it can become a stumbling block for business to implement it.

In that case the usage of standard cheap solution with API for integration would be the best solution [6]. And this is the reason why the SaaS application for cutting optimization can be considered. At first it has a flexible payment model – pay per use. It is about purchasing, only necessary amount of resources, it becomes possible optimize the costs associated with the work of the organization of information systems. And in combination with multitenancy [7], sharing resources between different users reduces costs even further. Multitenancy is another pros. It implies not only multi-user access but deep role system development.

Trust in the provider of cloud services is the most important criterion for evaluating cloud technologies. Cloud service providers can guarantee better quality of services or at least the same that business had before.

Another thing to be mentioned is scalability [8]. With cloud, it is possible to manage the application easily, implement new instances, use load balancers, create snapshots or backups for restoring after unexpected situations. And as the application is provided with SaaS model, there is no need to examine it.

Heuristic algorithms that are mainly used in cutting optimization software – algorithms for solving the problem, which correctness for all possible cases is not proven, but about which we know that it gives fairly good solutions in most cases. In fact, it may even be known that the heuristic algorithm is formally incorrect. It still can be used, provided that it gives the wrong result in only a few, fairly rare and a well-defined case, or gives inaccurate, but still acceptable results [9][10].

Simply, heuristics – is not fully mathematically grounded (or even "not quite correct"), but it is almost a useful algorithm. Summarizing heuristic algorithms does not guarantee finding the best solutions, does not guarantee finding a solution, even if it is known to exist (you can "pass the target"), can give the wrong decision in some cases.

Metaheuristics are another way of solving NP – hard problems Metaheuristics is a high-level algorithm, independent of the task framework, which contains a set of guidelines or strategies for developing heuristic optimization algorithms. This term is also used to mean a specific implementation of a heuristic optimization algorithm in accordance with the guidelines set forth in such a framework. The best-known solutions for cutting-packing problems are the following metaheuristics: Ant Colony Optimization, AFR, Genetic Algorithms, GA, Simulated Annealing, SA, Tabu Search, TS.

Artificial neural network (ANN) can also be applied to solving packing problem– it is mathematical model as well as its software or hardware embodiment, based on the principle of the organization and functioning of biological neural networks – networks of nerve cells of a living organism. This concept originated in the study of the processes occurring in the brain, and when you try to simulate these processes. Following the development of learning algorithms derived model began to be used for practical purposes: to predict problems, pattern recognition, in control problems, optimization, and others.

ANN system is connected and interacting simple processors (artificial neurons). Such processors generally are quite simple (particularly in comparison with the CPUs used in personal computers). Each processor of such a network must deal only with signals that it regularly receives, and signals that he periodically sent to other processors. And, nevertheless, being connected by a large network with controlled interaction are individually simple processors together are able to perform quite complex tasks.

Last considered algorithms are pseudo exact. The most famous example pseudo exact optimization method is a method of branches and borders. The branch and bound method – total algorithmic method for finding optimal solutions of various optimization problems, especially discrete and combinatorial optimization. In essence, the method is a variation of the exhaustive search with classifying subsets feasible solutions containing known not optimal solutions.

For the development of the application JavaScript language was chosen, as it allows both client side and server-side development. For the deployment platform amazon EC2 was chosen. Amazon also provides application load balancers (ALB) and inbuild authentication and authorization with the help of ALB. Amazon also provides amazon CLI, which allows to use terraform software, which allows automatically create, destroy and update existing infrastructure.

To implement cutting optimization algorithms were chosen the following heuristics: Next Fit Decreasing High (NFDH), First Fit Decreasing High (FFDH), Best Fit Decreasing Height (BFDH) and Floor Ceiling No Rotation. These algorithms become more complex in order they are listed.

3.1 Next Fit Decreasing High (NFDH)

The rectangles are sorted by non-increasing height (Decreasing High hints), the highest is located in the lower left corner of the strip, thereby initializing the first level, equal in height to it. The remaining rectangles are arranged from left to right, as long as there is space at the current level. A rectangle that does not fit in the level is placed on top, forming the next level, and so on [11][12].

JavaScript NFDH pure function example:

```
nextFitDecreasingHeight(squares, stripConfig) {
  const {width} = stripConfig;
  const level = [
{
 height: 0,
 width: 0,
 squares: [],
}
  ];
  squares.map((square, index) => {
if (index === 0 && square.height > level[level.length - 1].height) level[level.length - 1].height = square.height;
if (level[level.length - 1].width + square.width <= width) {
 level[level.length - 1].width += square.width;
 level[level.length - 1].squares.push(square);
} else {
 level.push({
  height: 0,
  width: 0,
  squares: [],
 });
 if (level[level.length - 1].height === 0) level[level.length - 1].height = square.height;
 if (level[level.length - 1].width + square.width <= width) {
  level[level.length - 1].width += square.width;
  level[level.length - 1].squares.push(square);
 }
}
  });
  return level;
};
```

3.2 First Fit Decreasing High (FFDH)

It is similar to the previous algorithm, with the difference that for each next rectangle a place is searched for not only at the last level, but starting from the lowest one. From here and "first fit" - the rectangle is located on the first suitable level from below. Intuitively, the packaging will be better. Another significant difference is in performance, since in the worst case it is necessary to consider all levels from bottom to top at every step [13][14][15][16][17][18].

JavaScript BFDH pure function example:

```
firstFitDecreasingHeight(squares, stripConfig) {
  const {width} = stripConfig;
  const level = [
 {
  height: 0,
  width: 0,
  squares: [],
 }
  ];
  const usedIndexes = [];
  squares.map((square, index) => {
 if (index === 0 && square.height > level[level.length - 1].height) level[level.length - 1].height = square.height;

 if (level[level.length - 1].width + square.width <= width) {
  level[level.length - 1].width += square.width;
  if (!usedIndexes.includes(index)) level[level.length - 1].squares.push(square);
 } else {
  squares.map((square, k) => {
   if (level[level.length - 1].width + square.width <= width && k > index) {
 level[level.length - 1].width += square.width;
 if (!usedIndexes.includes(k)) level[level.length - 1].squares.push(square);
 usedIndexes.push(k);
   }
  });
  level.push({
   height: 0,
   width: 0,
   squares: [],
  });
  if (level[level.length - 1].height === 0) level[level.length - 1].height = square.height;
  level[level.length - 1].width += square.width;
  level[level.length - 1].squares.push(square);
 }
  });
  return level;
 };
```

3.3 Best Fit Decreasing High

It is modification of the previous algorithm. Its essence is that of the levels that are suitable for packing the next rectangle, not the first, but the best one is chosen. The best level is the one on which there will be a minimum of space after packing the current rectangle. Simply put, the minimum suitable space is chosen, which contributes to a better level filling [19][20].

JavaScript BFDH pure function example:

```
bestFitDecreasingHeight(squares, stripConfig) {
  const {width} = stripConfig;
  const level = [
 {
  height: 0,
  width: 0,
  squares: [],
 }
  ];
  squares.map((square, index) => {
 if (index === 0) level[level.length - 1].height = square.height;

 let bestFitLevelIndex = level.length - 1;
 let smallestSpace = width;
 level.map((lvl, index) => {
  const freeSpace = width - lvl.width;
  if (square.width <= freeSpace) {
   if (smallestSpace > freeSpace) {
 smallestSpace = freeSpace;
 bestFitLevelIndex = index;
   }
  }
 });
 if (bestFitLevelIndex !== level.length - 1) {
  level[bestFitLevelIndex].squares.push(square);
  level[bestFitLevelIndex].width += square.width;
  return;
 }
 if (level[level.length - 1].width + square.width <= width) {
  level[level.length - 1].width += square.width;
  level[level.length - 1].squares.push(square);
 } else {
  level.push({
   height: 0,
   width: 0,
   squares: [],
  });
  if (level[level.length - 1].height === 0) level[level.length - 1].height += square.height;
  level[level.length - 1].width += square.width;
  level[level.length - 1].squares.push(square);
 }
  });
  return level;
 }
```

3.4 Floor Ceiling No Rotation

The biggest issue of algorithms described above is the remaining space above the situated on the floor blocks. The rectangles are sorted by non-increasing height. And the BFDH algorithm is applied with some modifications. Each level has a "floor" and "ceiling". If possible, the rectangles are packed on the "floor" from left to right. When the place ends, an attempt is made to pack to the "ceiling" from right to left; if there is no space on the ceiling, then only a new level begins. In the best traditions of BFDH, at every step all levels are viewed – first the “floor”, then the “ceiling” – for the presence of the most suitable place. Packaging, as you can see, is not bad. The method successfully packs the smallest rectangles along the “ceilings” [21][22].

As the main idea of packing using BFDH was given below and the FCNR uses the same principles in this section pseudo code is presented.

```
Input: The number of rectangles to be packed n, the dimensions of the rectangles {w(Li); h(Li)} and the strip width W.
Output: The height of a packing obtained in the strip.
1: Sort the rectangles in order of non-increasing height such that h(L1) ≥ h(L2) ≥ ... ≥ h(Ln)
2: for i = 1..n do
3: if $L_i$ is ceiling feasible then
4: pack $L_i$ on ceiling with minimum residual space
5: else [$L_i$ is not ceiling feasible]
6: if $L_i$ is floor feasible then
7: pack $L_i$ on the floor with minimum residual space
8: else [$L_i$ is not floor feasible]
9: level++;
10: end if
11: end if
12: end for
13: Output the height H of the strip, found by adding the height of each level.
```

4 Discussion

It should be mentioned that developed software is far from production as it does not have inbuilt role system, integrated payment system and relies on only AWS services like ALB to achieve it. Moreover, far not all heuristics algorithms were implemented in the software. There were implemented only level algorithms. But with the help of current application architecture and tools that were used, as terraform it will take a little time to implement new methods for cutting optimization which is beneficial for business.

Another important thing to be mentioned is that end-user should manually choose algorithms according to production situations, materials used. In future researches the model that can do it automatically can be presented or a decision support module.

The research provides a brief and possibly blurry description of some areas, such as cloud computing and its advantages and disadvantages comparing to on premises software. But it should be said that the area of cutting optimization and NP-hard problems solving is extremely vast and it is complicated to cover even the top of the topic.

REFERENCES

[1] Lyovina, A., Kalyazina, S., Sinelnikov, M., Poljanskihh, A.: Conceptual model of IT-infrastructure for production company: Target vision and development approach. In: Proceedings of the 33rd International Business Information Management Association Conference, IBIMA 2019: Education Excellence and Innovation Management through Vision 2020. pp. 8728–8735 (2019).

[2] Knapsack Problem Algorithms and Computer Implementations University of Bologna // John Wiley & Sons 1990. 296 p.

[3] Lijun Wei, Wenbin Zhu, Andrew Lim.A., Block-based Layer Building Approach for the 2D Guillotine Strip Packing Problem // European Journal of Operational Research 2001

[4] ljashenko, O., Bagaeva, I., Levina, A.: Strategy for establishment of personnel KPI at health care organization digital transformation. In: IOP Conference Series: Materials Science and Engineering (2019). https://doi.org/10.1088/1757-899X/497/1/012029.

[5] Anisiforov, A., Dubgorn, A., Lepekhin, A. Organizational and economic changes in the development of enterprise architecture (2019) E3S Web of Conferences, 110, № 02051.

[6] Ilin, I., Levina, A., Abran, A., Iliashenko, O. (2017). Measurement of enterprise architecture (EA) from an IT perspective: Research gaps and measurement avenues. Paper presented at the ACM International Conference Proceeding Series, Part F131936, 232-243. doi:10.1145/3143434.3143457.

[7] Ilin, I. V., Iliashenko, O. Y., & Borremans, A. D. (2017). Analysis of cloud-based tools adaptation possibility within the software development projects. Paper presented at the Proceedings of the 30th International Business Information Management Association Conference, IBIMA 2017 - Vision 2020: Sustainable Economic Development, Innovation Management, and Global Growth, 2017-January, 2729-2739.

[8] Krasnov, S., Sergeev, S., Titov, A., Zotova, Y.: Modelling of digital communication surfaces for products and services promotion. In: IOP Conference Series: Materials Science and Engineering (2019). https://doi.org/10.1088/1757-899X/497/1/012032.

[9] Hopper, E., Two-dimensional packing utilizing evolutionary algorithms and other meta-heuristic methods. // Ph.D. thesis University of Wales 2000.

[10] A new placement heuristic for the orthogonal stock-cutting problem. Operations Research // Burke, E. K., Kendall, G., Whitwell, G. // 2004. 655-671 p.

[11] Pan W.T., A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example //Knowledge-Based Systems 2012. 69-74 p.

[12] Gent, I. 1998. Heuristic solution of open bin packing problems. Journal of Heuristics 3:299–304.

[13] Local Search in Combinatorial Optimization // E. Aarts and J. K. Lenstra // Wiley & Sons, Chichester 1997. 91–120 p.

[14] R. K. Ahuja, T. L. Magnanti, J. B. Orlin. Network Flows. // Prentice Hall, Englewood Cliffs, 1993.

[15] B.S. Baker, A new proof for the first-fit decreasing bin-packing algorithm, J. Algorithms 6 (1) (1985) 49–70.

[16] E.G.Coffmann Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: A survey, in: D. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing, Boston, 1997.

[17] M.Yue, A simple proof of the inequality FFD(L)11/9 OPT(L)+1,∀L, for the FFD bin-packing algorithm, Acta Math. Appl. Sin. 7 (4) (1991) 321–331.

[18] Gy. Dósa, The tight bound of first fit decreasing bin-packing algorithm isFFD(I)11/9OPT(I)+6/9, in: ESCAPE 2007, in: Lecture Notes in ComputerSciences, vol. 4614, 2007, pp. 1–11.

[19] W. Zhong, Gy. Dósa, Z. Tan, On the machine scheduling problem with job delivery coordination, Eur. J. Oper. Res. 182 (2007) 1057–1072.

[20] Johnson, D. 1973. Near-Optimal Bin Packing Algorithms. Ph.D. Dissertation, Dept. of Mathematics, M.I.T., Cambridge, MA.

[21] Bischoff E.E. and Wäscher G. Cutting and packing, European Journal of Operational Research, 1995, No. 84, pp. 503-505. 2.

[22] Ross P., Marin-Blazquez J.G., Schulenburg, S. and Hart E. Learning a Procedure That Can Solve Hard Bin-Packing Problems: A New GA-Based Approach to Hyper-heurstics, Proceeding of the Genetic and Evolutionary Computation Conference, GECCO 2003, Chicargo, Illinois, USA, 2003, pp. 1295-1306.