Abstracts of Dissertations

Institute of Information and Communication Technologies

BULGARIAN ACADEMY OF SCIENCES

8 / 2022

Metaheuristic Methods for Reducing Cutting Tasks

Georgi Evtimov

Метаевристични методи за решаване на задачи за разкрояване

Георги Евтимов

Автореферати на дисертации

Институт по информационни и комуникационни технологии

БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

ISSN: 1314-6351

Поредицата ***„Автореферати на дисертации на Института по информационни и комуникационни технологии при Българската академия на науките“*** *представя в електронен формат автореферати на дисертации за получаване на научната степен „Доктор на науките” или на образователната и научната степен „Доктор”, защитени в Института по информационни и комуникационни технологии при Българската академия на науките. Представените трудове отразяват нови научни и научно-приложни приноси в редица области на информационните и комуникационните технологии като Компютърни мрежи и архитектури, Паралелни алгоритми, Научни пресмятания, Лингвистично моделиране, Математически методи за обработка на сензорна информация, Информационни технологии в сигурността, Технологии за управление и обработка на знания, Грид-технологии и приложения, Оптимизация и вземане на решения, Обработка на сигнали и разпознаване на образи, Интелигентни системи, Информационни процеси и системи, Вградени интелигентни технологии, Йерархични системи, Комуникационни системи и услуги и др.*

The series ***Abstracts of Dissertations of the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences*** *presents in an electronic format the abstracts of Doctor of Sciences and PhD dissertations defended in the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences. The studies provide new original results in such areas of Information and Communication Technologies as Computer Networks and Architectures, Parallel Algorithms, Scientific Computations, Linguistic Modelling, Mathematical Methods for Sensor Data Processing, Information Technologies for Security, Technologies for Knowledge management and processing, Grid Technologies and Applications, Optimization and Decision Making, Signal Processing and Pattern Recognition, Information Processing and Systems, Intelligent Systems, Embedded Intelligent Technologies, Hierarchical Systems, Communication Systems and Services, etc.*

Редактори

Геннадий Агре
Институт по информационни и комуникационни технологии, Българска академия на науките
E-mail: agre@iinf.bas.bg

Райна Георгиева
Институт по информационни и комуникационни технологии, Българска академия на
E-mail: rayna@parallel.bas.bg

Даниела Борисова
Институт по информационни и комуникационни технологии, Българска академия на науките
E-mail: dborissova@iit.bas.bg

Editors

Gennady Agre
Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
E-mail: agre@iinf.bas.bg

Rayna Georgieva
Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
E-mail: rayna@parallel.bas.bg

Daniela Borissova
Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
E-mail: dborissova@iit.bas.bg

e-ISSN: 1314-6351

BULGARIAN ACADEMY OF SCIENCES

Abstract of PhD Thesis

METAHEURISTIC METHODS FOR REDUCING CUTTING TASKS

Georgi Evtimov Evtimov

Supervisor: Prof. Stefka Fidanova

Approved by Supervising Committee:

Prof. Todor Atanasov
Prof. Olimpia Roeva
Assoc. Prof. Desislava Ivanova
Prof. Ivan Dimov
Assoc. Prof. Leonid Kirilov

IICT

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Department of Parallel Algorithms

Chapter 1

Introduction

The need for optimization of human labor leads to mass distribution of computing electronic devices that in most cases they replace the human presence. That's where it starts the extreme development of information technology (IT) as a tool to control computing devices. Significant reduction in the cost of electronics further develops this process. More and more production machines are operated by computers, whether used by a small, medium or small enterprise great. A natural continuation of this process is the development of a network to integrate electronic devices called the Internet.

There is a current trend to redirect all services from reality to virtual reality. High-tech industries in the manufacturing sector are massively building systems for planning and managing the resources of production. Information systems allow optimizing resources at all levels of the organizational hierarchy. This optimization in most cases has a positive impact on both the competitiveness of the company and contributes for more flexible and faster finding new markets. Use of information systems by companies makes it possible to their customers to solve better, faster and more efficiently subject-specific problems. This approach is especially effective in the field of heavy industry and construction.

The widespread application of IT in Europe and Bulgaria has created a dynamic and highly competitive environment in which a company without Implementing IT solutions is often doomed to bankruptcy. The need to cut production costs is vital important for the survival of the enterprise. On the other hand, customer requirements are growing, which further fuels the need for rapid solutions to optimize material and human resources, which is achieved effectively using software. This market situation reveals new and almost unlimited possibilities for the application of application software in solving a variety of tasks. Any software in which knowledge is applied can be seen as a strategic source of innovation. Another flexibility of IT is the ability for the technology to be developed by a third party that does not belong to the company, but to be effectively applied by many other companies.

In short, the focus of research combined with technological innovation is one of the most dynamic areas of development of modern industry. The present dissertation is an effort in this direction. The motivation and object of application of this work comes from the construction industry and in particular from the production of steel structures.

One of the most important and widely practiced activities there is the following: for the needs of a construction site it is necessary to cut a certain number of details (often reaching thousands) with different sizes, shapes, thicknesses, and in some cases from different material. The material is delivered in the form of metal sheets or remnants of sheets from which details have been previously cut. An example of such a steel structure is shown in Figure 1.1. It is necessary to cut out the necessary details while

minimizing the cost of material.

This statement is a special case of the general mathematical problem for optimal cutting. The practical task for optimal cutting lies in the following simple formulation: set is defined material (for example, in the textile industry it is fabric, in building constructions it is metal sheets) and a large amount of often different, details. It is necessary to cut out the necessary details while minimizing material consumption. In practice, this means minimizing the material that remains after cutting and cannot be recovered except to be recycled. This unused material is often called waste. This task is mathematically formulated more than 80 years ago in connection with the industrialization of garment production. Similar types of problems arise in many other industrial productions and the use of optimization solutions can lead to significant material savings.

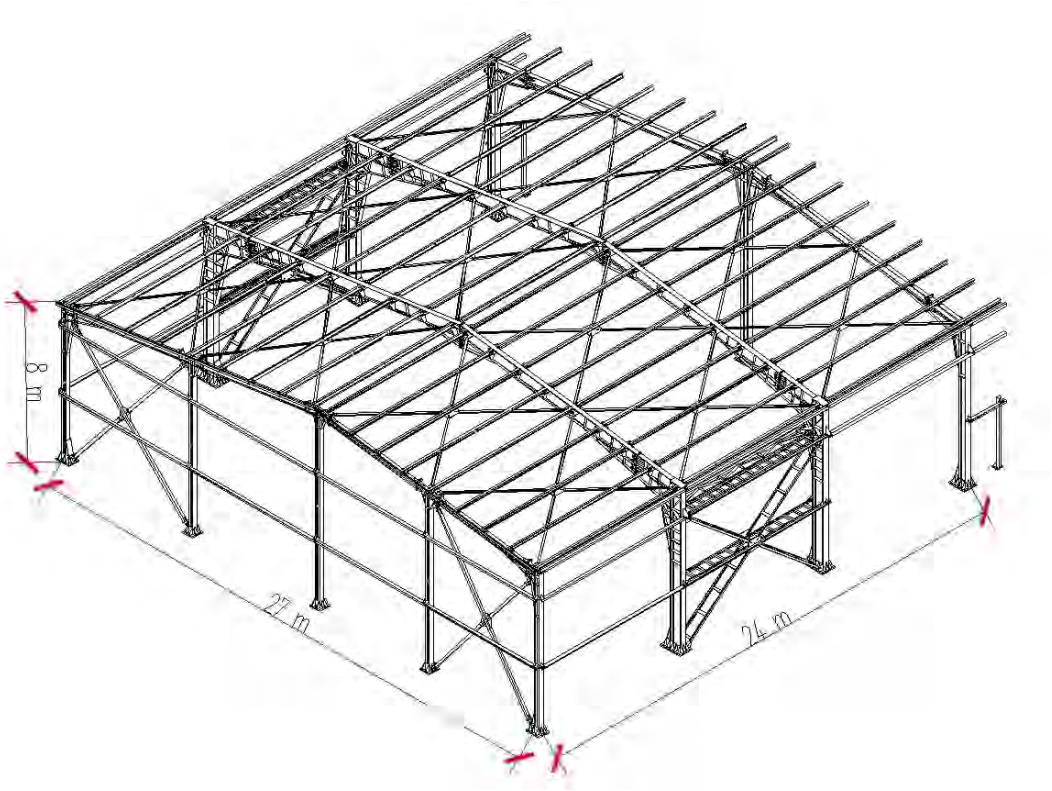


Figure 1.1: Steel Construction

The scientific methods proposed in this paper will be useful and most suitable for the needs of start-ups, including those in the software field. The methods presented here are based on mathematics and logic and do not use external libraries, can be written in any programming language and can help the development of any company.

1.1 Actuality of thesis

The topic of optimal cutting becomes even more relevant over the last two decades, finding a variety of applications in many industrial industries. The task is especially important now that the market is open and companies have to compete with a large number of competitors with modern equipment and low labor costs.

On the other hand, the mass consumption of goods leads to the need to optimize the production of these goods. This includes both minimizing the consumption of energy and raw materials and reducing the use of human labor. The problem is very acute in heavy industries, especially when it is necessary to make a large number of elements from expensive material. These two features are present, for example, in the construction industry. Therefore, the problem of optimal cutting there is on the agenda with special relevance.

An overview of the existing methods and their implementation in application software is made below in the Section 1.2. Here we will only note that the software market for the needs of optimal cutting has basically two types of implementation approaches. In one approach, the plates are approximated to rectangles that are optimally placed on the steel sheet, and in the second they have their exact geometry. In methods that apply approximation to a rectangle the disadvantage is that in figures other than a rectangle the waste is large. In different industries, a large number can be understood as a variety of numbers. In the present paper we will understand 25-50 % for large waste, 10-25 % for medium large waste, 0-10 % for small waste. When working with triangular shapes formed by straight lines on all three sides, the waste is 50 %. Big waste. This approach has limited application, but is used quite widely and gives good results in glassmaking and the paper industry. In both types of software, the input of the cutting objects is done manually. The polygons are entered by coordinates of the vertices or by segments of the sides. It takes a lot time and it is possible to make inaccuracies and / or errors in data entry.

In the last three decades, CAD systems have been widely used for the design of construction sites. In the presented dissertation the problem of optimal disclosure of building elements (or plates) is solved under the assumption that the polygons (plates) are generated and provided to the builder by a CAD system. Then the data is preprocessed and finally the plates are cut with their exact geometry.

Preparation of plates for the needs of steel structures represent a certain class subtasks for optimal cutting, which is characterized by a number of features that lead to simplification and to complicate the task of cutting. The most important features are:

1. often the planks have complex shapes, the boundaries of which are arbitrary non-self-intersecting polygons (in rare cases, making elements with elliptical contours is reduced to the above case by approximation with polygons with sufficient accuracy for practice);

2. very often the plate does not have a "face" and a "back", which allows a mirror search of its location in the cutting process; this feature can lead to material savings, but increases the complexity of the problem.

3. in the set of slats to be produced, there is often a considerable variety of sizes, areas and shapes.

The paper focuses on methods for solving the problem for cutting into slats in $2D$. The two-dimensional cutting task is more difficult than the one-dimensional, especially

when the cut-out figures are not convex and have an irregular shape. Both tasks are NP -complete combinatorial optimization tasks [12], [13].

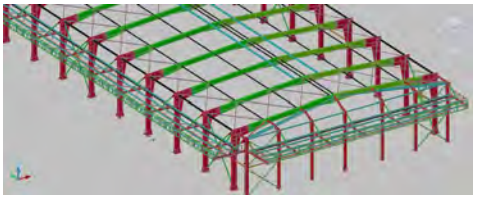


Figure 1.2: Steel Structure 1.

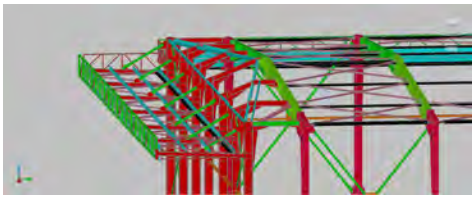


Figure 1.3: Steel Structure 2.

As an illustration, some examples of the case of steel structures will be presented. In the steel hall shown in figure ref fig: 3DView1 there may be about 2000 - 3000 steel plates for cutting. They are of different thickness, in practice we often have to work with 6 different thicknesses. Therefore, from a metal sheet of a given thickness should be cut about 500 pieces of plates with quite complex shape. In the present case rotation and mirror image of the plate are recommended in the optimization of the cut. Moreover, there are slats in which the ratio of length to width (of course as the plate is placed in a rectangle with minimum dimensions) is more than 100.

1.2 Overview of the main results in the field

The problem of optimal cutting (CSP) occurs in many industrial areas [61]. Most authors solve 2D cuts by approximating the input polygons (figures) to rectangles. These solutions are also applicable in many industries. For example, the production of paper and glass [24], container loading, multi-scale integration (VLSI) design, and various scheduling tasks [55].

The more complex version of the problem is when the input polygons (figures) are not approximated to rectangles. This problem arises in building structures in the manufacture of steel products, the manufacture of clothing, the manufacture of footwear, etc. In [24] the main theme is a two-dimensional orthogonal packaging problem in which a fixed group of small rectangles should be mounted in a large rectangle and the unused area of large rectangles should be minimized. The algorithm combines a substitution method with a genetic algorithm. In [45] a *Greedy* (greedy) procedure of random (randomized) adaptive search has been developed. In this study, there is a large primary stock that needs to be cut into smaller pieces to maximize the value of the pieces. Cintra [49] offers a precise algorithm based on dynamic programming that is suitable for small problems, since the problem is NP-difficult. Dusberger and Raidl [62], [63] offer two meta-heuristic algorithms based on searching for variable neighborhoods. The above works solve the simplified problem with rectangular elements. In the building industry, slabs are polygons that can be irregular in shape and can be convex or concave, but not just intersecting. Such a variety of forms significantly increases the severity of the problem. As the slats or entrance polygons can be applied in a mirror, since the steel sheets are homogeneous on both sides. The complexity of the task is also increased by the fact that a triangular plate can be described with more than three points.

1.3 Aims and objectives of the dissertation

The main aims set for the doctoral student are of scientifically applied and applied nature. They can be summarized as follows.
Dissertation objectives:

1. Optimal cutting of linear elements with minimal waste;
2. Optimal cutting of two-dimensional elements with an irregular shape with minimal waste.

To achieve these goals, the following tasks were formulated:

- Task 1. Development of an algorithm for solving the problem of one-dimensional (linear) cutting;
- Task 2. Development of an algorithm for solving the problem of cutting two-dimensional elements;
- Task 3. To make a program implementation of the developed algorithms and to be implemented the comparison of real construction sites with existing in practice methods of cutting.

The basis of development is a CAD environment for obtaining graphic information from a given construction site. After optimization, information is generated in terms of the same CAD environment. For this purpose, a numerical algorithm for cutting (placement) of arbitrary non-intersecting polygons has been developed. (called planks and generated by the CAD system) from a user-specified polygon (steel sheet). The focus of the dissertation are planar elements (sheet material) $2D$ figures (called plates here). Geometrically, this means that a certain number of figures in the plane are arranged in an area with a given user closed loop. This is a fairly general mathematical problem that can be applied in a variety of industries. The algorithm allows and additional settings and various principles in the optimization when arranging the figures. These tasks are based entirely on examples from practice, and the input data are from actually designed and executed construction sites.

1.4 Research approach

This dissertation deals with the solution of two optimization problems: (1) cutting of linear profiles, steel bars, T - and Π -shaped profiles, etc.) or $1D$ cutting and (2) cutting of two-dimensional (flat) plates of steel sheets.

The first task is one-dimensional (linear), $1D$ *cutting*. No optimizations are introduced for $1D$ optimization special definitions, as it works with one parameter, the length of the element. The task of minimal waste comes down to finding the minimum number of profiles used. Although it is easier than the two-dimensional task, it is also NP complex. The approach uses the ant method.

The second task is it $2D$ cutting. The data includes an incoming list of n per number of laths (called input polygons) that need to be arrange as tightly as possible in a polygon, called the main. In search of a possible placement of incoming polygons can be applied rotation and mirror image. Once the location of the input polygon is selected it is necessary to apply the algorithm for "subtraction" ("cutting") of two polygons. This is done in order for the next incoming polygon to look for a location

in the rest of the main polygon. This (cut) strip is then removed from the list with input strips. This is repeated until all the bars in the input list are exhausted. Then the overall solution (the set of planks) is evaluated by metaheuristics. See point **??**.

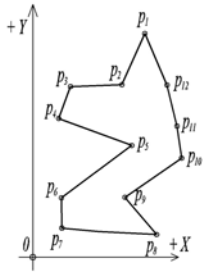


Figure 1.4: Initial main polygon (to be filled).

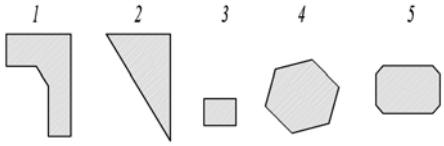


Figure 1.5: Exemplary input polygons (dimensions are significantly increased)

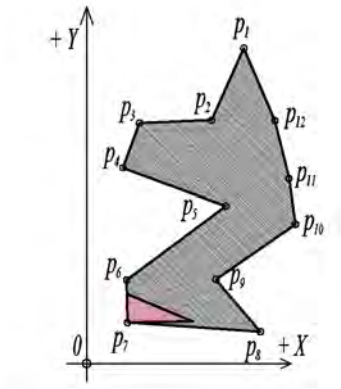


Figure 1.6: Main polygon before cutting.

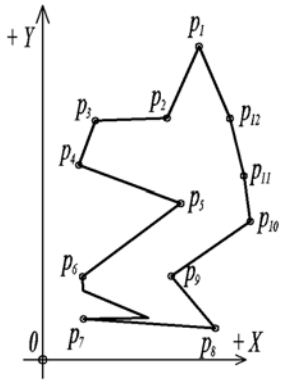


Figure 1.7: Main polygon after cutting.

A wide range of literature has been used in the process of studying the problem. Created new methods and algorithms are published in the author's articles, [12, 13, 15, 14]. Three methods have been developed. The first is to evaluate the input polygons (sets). To their angles and lengths. This method gives a score from 0. to 1. for the highest probability that a polygon will be placed in the fill polygon at a given peak. The second method gives an estimate of the largest contact length (area) between two polygons. The third method is hybrid metaheuristics. Gives an assessment of all admissible solutions for a given peak. The score for each decision is between 0. and 1. The one with the highest score is chosen. If there is more than one solution with a maximum score, one of them is chosen at random. The combination of the three methods allows us not to look for complete exhaustion of the possible combinations for placing the input polygons Π_i in the polygon of filling Δ. Two new algorithms have been developed, which are an improvement on two existing algorithms. One is the *Ray* method [40]. The addition is that before applying the *Ray* method, it is checked whether the given point is in the *box* of the polygon, if so then it is checked for the whole polygon. For a definition of *box* on the polygon, see point 2.5. The *box* of the polygon will be used if the number of vertices of the given polygon is greater than 4. The other method is "Bentley-Ottman " [39]. The addition is that not all segments are crawled, and the algorithm stops the first time the two segments intersect.

As a final result of the research of the problem, software was developed that successfully solves both tasks. A comparison of the obtained results with the results from

the use of commercial software is made. The advantages of the created program over the tested commercial software are given in 4.6 and in 5.

Chapter 2

Computational Geometry: Basic Definitions

In this section the geometric objects point, linear segment and polygon in the two-dimensional plane are used. All points will be represented as a list of ordered numbers (coordinates), [9], in two-dimensional cases these are pairs of numbers $P = (x, y)$. Here we also discuss the concepts important for the construction of algorithms when a point is inside a polygon, intersection and subtraction of two polygons, etc.

2.1 Definition of List

The list is strictly ordered items. Each element can be a number, a string or another list. Examples:

1. $list(X, Y)$ - point given by its Cartesian coordinates;
2. $list(pt_0, pt_1, \ldots, pt_i)$ - a list of points, where pt_i is a list representing a point with index i;
3. $list\ (e_0, e_1, \ldots, e_{n-1})$ - a list of segments where e_i is a linear segment with index i, see below.

2.2 Definition of point, segment and polygon

Definition of point.

The points in the d -dimensional space are represented as an ordered list of d numbers called coordinates, [9]. Since we consider the problem in a plane, in this work the point pt_i is defined as $pt_i = list\ (x_i, y_i)$, where the coordinates x_i and y_i are real numbers. When working with real numbers using computer arithmetic raises the question of the rounding error. The error of rounding real numbers is an important and extensive field in mathematics. The following rules have been adopted:

1. We work with an accuracy of four characters after the decimal point .0001;
2. We accept a deviation of $fuzz$, which is a real number greater than 0.

These rules are dictated by the need to work with data produced by CAD systems. In the field of design of objects made of steel structures, the dimensions of the structures are give in millimeters, so all numerical data (coordinates of points, segments, etc.)

are in millimeters. For the needs of the construction industry (steel structures), the difference in the lengths of the sides of the plates of 0.5mm does not make the detail indistinguishable. That is why we accept them as the same.

We assume that two points coincide $P_i \equiv P_q$, ако:

$$(x_i - fuzz) \leq x_q \leq (x_i + fuzz) \wedge (y_i - fuzz) \leq y_q \leq (y_i + fuzz) \tag{2.1}$$

или

$$max\{|x_i - x_q|, |y_i - y_q|\} \leq fuzz \tag{2.2}$$

Definition of linear segment.
We will use two types of segments:

(1) CAD linear segment. $CADe_i = list(pt_0, pt_1, \ dots, pt_i)$ is set with a list of points that lie on one line; such segments are obtained from the operation of the CAD system, which generates all input data used in this work.
(2) Linear segment $e_i = (pt_i, pt_{i+1})$ a closed set of points lying on a line between two points pt_i and pt_{i+1}, called endpoints, [9]. The items in the e_i list are sorted. The first is the initial and the second the final. In our work, the linear segments are obtained after removing the inner points of the CAD segment.

Definition of polygon.

Polygon is closed area of the plane surrounded by n linear segments forming a closed curve [9] Note that we are using a linear segment here, not a CAD linear segment. Let $pt_0, pt_1, \ldots, pt_n$ be n points on a plane such that $pt_0 = pt_n$. The points $pt_0, pt_1, \ldots, pt_n$ form a cyclic list. While pt_0 is followed by pt_1, pt_{n-1} is followed by $pt_0 = pt_n$. The polygon is also described by its vertices, the endpoints of its segments, so that equivalently, $\Pi = list\ (pt_0, pt_1, ..., pt_n)$. We say that two segments are adjacent when they have only one common endpoint.

Linear segments form a polygon if and only if:

1. The intersection point between each pair of adjacent segments in the cyclic list is: $e_i \cap e_{i+1} = pt_{i+1}$, for all $i = 0, ..., n-1$;

2. Non-adjacent segments do not intersect.

We will call the points pt_i *vertices* of the polygon, and a segment of the polygon will we call it a linear segment. Note that a polygon with n vertices has a n segment.

2.3 Rotation of point

Let's look at two different points $pt_A = list(x_a, y_a)$ и $pt_{base} = list(x_{base}, y_{base})$ in the coordinate system XOY. We want to turn the point pt_A arround the base point pt_{base} of given angle β. If the angle β is positive number, then the rotation is counterclockwise $anti - CW$, otherwise the rotation is clockwise CW. After the rotation we will get a new point $pt_B = list(x_b, y_b)$ in the coordinate system XOY.

To make the necessary calculations and get the calculation formulas for the coordinates of the point obtained after the rotation, we will introduce a new coordinate system $X'O'Y'$, whose coordinate origin matches pt_{base}. The axes of the new coordinate system $X'O'Y'$ are translate parallel to the axes of the XOY coordinate system. For the new coordinate system we get the coordinates of the point pt_A, $x'_a = (x_a - x_{base})$

and $y'_a = (y_a - y_{base})$ or $pt_A = list(x'_a, y'_a)$ in the new coordinate system $X'O'Y'$. The turning radius R is found by the formula:

$$R = \sqrt{{x'_a}^2 + {y'_a}^2} \tag{2.3}$$

The angle of rotation α is receiving by formula:

$$\alpha = arccos\frac{x'_a}{R} \tag{2.4}$$

Then the coordinates of the point pt_B in the coordinate system XOY are:

$$x_b = x_{base} + \frac{R}{\cos(\alpha + \beta)} \tag{2.5}$$

$$y_b = y_{base} + \frac{R}{\sin(\alpha + \beta)} \tag{2.6}$$

2.4 Perpendicular from a point to a line

The line is set with two points $A = list(x_a, y_a)$ $B = list(x_b, y_b)$ and the test point is $T = list(x_t, y_t)$. We want to find a point $C = list(x_c, y_c)$ from the line $list(A, B)$ such that the vector defined by the points T and C are perpendicular to the line.

Before we start looking for the point C, we need to check if the points $A = list(x_A, y_A)$, $B = list(x_B, y_B)$ and $T = list(x_t, y_T)$ do not lie on the same line. This is done by finding the face of the triangle $F = list(A, B, T)$. The result we will get from this algorithm is the oriented face of the triangle $list(A, B, T)$. We are only interested in whether the person F is zero or not. If the person $F = 0$, then the points lie on one line and we do not need to look for the perpendicular vector $\vec{TC}$ to the line $list(A, B)$. If the person $F \neq 0$. Then the algorithm proceeds in the following steps, according to [41]:

1. If $x_A = x_B$, then the line is vertical and the search point is $pt_C = list(x_A, y_T)$.;
2. If $y_A = y_B$, then the line is vertical and the point sought is $pt_C = list(x_T, y_A)$;
3. If the above conditions are not met, then we look for the slope m of the line $list(A, B)$;

$$m = \frac{y_B - y_A}{x_B - x_A} \tag{2.7}$$

$$x_C = \frac{(\frac{x_T}{m} + y_T + m.x_A - y_A)}{m + \frac{1}{m}} \tag{2.8}$$

$$y_C = y_A + m(x_C - x_A) \tag{2.9}$$

Or the coordinates of the desired point is: $C = list(x_C, y_C)$.

2.5 Finding of *box* of polygon

By *box* of polygon $\Pi_i = list(pt_0, pt_1 \dots pt_n)$ we will understand the rectangular shell of the given polygon Π_i. Look figure 2.1.

To find *box* on the polygon Π_i we have to go through the list of points $\Pi_i = list(pt_0, pt_1\ dotspt_n)$ and for each point we take its coordinates on $X.pt$ and on $Y.pt$.

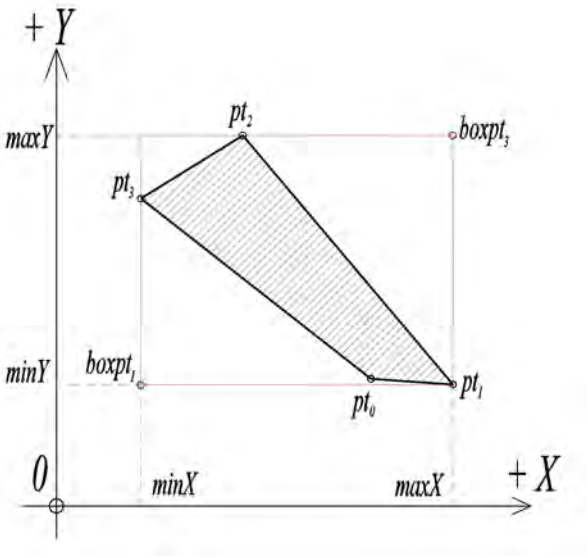


Figure 2.1: *box* of polygon

Then sort the two new lists in descending order, $listX = (x_0, x_1 \dots x_n)$ and $listY = (y_0, y_1 \dots y_n)$. The first value of $listX$ will give us $maxX$ the last $minX$. We do the same for the $listY$ list. Thus we get the coordinates of the points $boxpt_1 = list(minX, minY)$ and $boxpt_3 = list(maxX, maxY)$. Point $boxpt_1$ is always at the bottom, on the left. Point $boxpt_3$ is always at the top, on the right.

2.6 Mirror image of polygon

By mirror image of a polygon we will mean a mirror image on all sides of the polygon, which are not parallel to each other.

The polygon $Mirror1 = list(mpt_0, mpt_1, mpt_2, mpt_3)$ is obtained from the shaded polygon $\Pi_i = list(pt_0, pt_1, pt_2, pt_3)$. To find the mirror image of the polygon P_i uses the following sequence:

1. We take the first pt_0 and the second pt_1 point from the polygon P_i.

2. We form the rights $L1 - L1$. The line $L1 - L1$ is formed by two points. The first point is $pt_{L1} = (polar(pt_0; (angle = pt_1, pt_0); 10e10))$. The second point is $pt_{L2} = (polar(pt_1; (angle = pt_0, pt_1)); 10e10)$. We find the polar coordinates of the points pt_{L1} and pt_{L2} - base point, angle and length. In this case, the length is chosen large enough to be acceptable for the CAD system.

3. For each vertex of the polygon P_i we find the heel of the perpendicular to the line $L1 - L1$ and denote it by pt_{Perp_i}. The mirror point is obtained: $mpt_i = (polar(pt_{Perp_i}; (angle = pt_i, pt_{Perp_i}); distance(pt_i, pt_{Perp_i}))$. To find the heel of perpendicular to the line $L1 - L1$ see Subsection **??**.

The line $L1 - L1$ will be collinear with the segment $list(pt_0, pt_1)$. Then the distance $distance(pt_0, pt_{Perp_i})$ will be zero and the points pt_0 and mpt_i will coincide. Any pair of consecutive vertices $list = (pt_i, pt_{i+1})$ on the polygon P_i can be used to find the line $L1 - L1$.

2.7 Finding the direction of a polygon (CW or $anti-CW$)

Here we will discuss ways to determine the direction, (clockwise (CW) or counterclockwise ($anti-CW$)), at the boundary of polygon $\Pi = list\ (pt_0, pt_1, ..., pt_n)$. From the basics known in the literature methods for landmark orientation, here we will present one of the fastest methods for calculating the direction of traversal of the peaks along the border at the Π [28] polygon

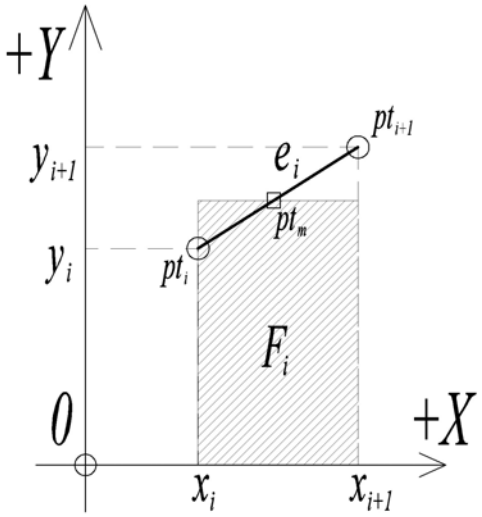


Figure 2.2: Area of Trapeze

Let $e_i \in \Pi$ is an arbitrary segment and let its midpoint P_m has coordinates:

$$P_m = \left(\frac{x_i + x_{i+1}}{2}, \frac{y_i + y_{i+1}}{2}\right). \tag{2.10}$$

Area of the figure between segment $e_i \in \Pi$ and the coordinate axis X is:

$$F_i = \frac{(x_{i+1} - x_i)(y_{i+1} + y_i)}{2} \tag{2.11}$$

Note that the person F_i can be positive, negative or zero. The face sign depends on the order of the points in the list defining P since the layout can be $list\ (pt_0, pt_1, ..., pt_{n-1})$ or $list\ (pt_{n-1}, pt_0, \ ... \ , pt_{n-1})$. We will call this person an oriented person. This procedure applies to all e_i segments. To save CPU time, it makes no sense to divide each person by 2. Therefore, the sum of the oriented persons can be recorded as follows:

$$2F = \sum_{i=0}^{n-1} (x_{i+1} - x_i)(y_{i+1} + y_i) \tag{2.12}$$

If the coordinates of the points on vertices $list(pt_0, pt_1,\ dots, pt_5)$ satisfy conditions $x_5 > x_4 > x_3 > x_2 > x_1 > x_0$, then the corresponding areas are positive and $F > 0$. If the coordinates of the points on vertices $list(pt_0, pt_1,\ dots, pt_5)$ satisfy the conditions $x_5 > x_6 > x_7 > x_8 > x_9$, then $F < 0$

2.8 Angle between two vectors. Internal angle of a polygon.

In order to get a better characteristic for a given landfill, we will need the internal ones its angles. First we will find the angle between two vectors $\vec{a} = list(T, pt_i)$ and $\vec{b} = list(T, pt_{i+1})$, defined in the XOY coordinate system.

We first determine the lengths of the vectors $\vec{a}$ and $\vec{b}$, and then their scalar product. To find the length of the vector $\vec{a}$, we will translate the point A_i to zero, $T_i = (list0, 0)$. Then the vectors $\vec{a}$ and $\vec{b}$ will have coordinates (x_a, y_a) and (x_b, y_b), respectively.

$$\|\vec{a}\| = \sqrt{x_a^2 + y_a^2} \tag{2.13}$$

$$\|\vec{b}\| = \sqrt{x_b^2 + y_b^2} \tag{2.14}$$

the scalar product is:

$$\vec{a} \cdot \vec{b} = x_a x_b + y_a y_b \tag{2.15}$$

and so we get

$$\cos\alpha = \frac{\vec{a} \cdot \vec{b}}{\|a\|\|b\|} \tag{2.16}$$

To find the interior angles of a polygon, we will need to check whether the polygon in the vicinity of a vertex is convex. or not protruding. In order to check this we need to find the orientation of the polygon. To find the interior angles the direction of the polygon must be counterclockwise $anti - CW$. Then we start checking everyone three points from the polygon $list(pt_i, pt_{i+1}, pt_{i+2})$. We find the inner angle α, which is at the vertex pt_{i+1}. We check the orientation of the three points. If they are clockwise CW, then from 2π we need to subtract the angle α. If the orientation of the vertices $list(pt_i, pt_{i+1}, pt_{i+2})$ is the opposite of clockwise $(anti - CW)$, then we record the angle α without correction.

Algorithm 1 InsidePolyAngle

/*Function for finding internal corners of a polygon*/
procedure INSIDEPOLYANGLE($pt_0, pt_1, \ldots, pt_n$)

if isClockWise $pt_0, pt_1, \ldots, pt_n$ **then return** ptList = reverse $pt_0, pt_1, \ldots, pt_n$
i = 0
L = length of $pt_0, pt_1, \ldots, pt_n$
Repeat L
for pt_i, pt_{i+1}, pt_{i+2} **do** α = getInsideAngle pt_i, pt_{i+1}, pt_{i+2}
if isClockWise pt_i, pt_{i+1}, pt_{i+2} **then return** α = (2π - α)
else α
i = i + 1
End Repeat

In this way, it will be possible to write information about the internal angles and lengths of its sides to each polygon.

2.9 Crossing of two segments

Finding an intersection between two lines is an "expensive" operation in terms of CPU time and should be used in the "extreme" case, so here we will look at two functions. The first is to find the coordinates of the intersection point pt_x of two given segments e_1 and e_2, and the second is to check whether two given segments e_1 and e_2 intersect without looking for the intersection point itself. The first function will return the intersection point $list = (x_i, y_i)$ as a value, and the second - $true$ or $false$.

Find the coordinates of the intersection point pt_x.

According to [31] and [30] the intersection point $ptX = (X, Y)$ of two given segments $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ has coordinates:

$$X = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}, Y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}. \tag{2.17}$$

By calculating the determinants in (2.17), we obtain the following expressions for X and Y:

$$X = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(x_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{2.18}$$

$$Y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(x_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}. \tag{2.19}$$

2.10 Point in polygon

We will consider the following problem in the XOY plane. For a given random point $T = list(x, y)$ (called test point) and polygon $\Pi = list(pt_0, pt_1, \ldots, pt_n)$ to determine whether the point is inside the polygon or not. The vertices of the polygon are given by $pt_i = list(x_i, y_i)$.

Here will be considered two methods for solving this problem - Ray crossing method (intersecting beam), [32], and Balanced sum of angles. In the software developed for the dissertation the following approach is used before applying the intersecting beam method to all segments. It is checked by the method of the intersecting beam whether the given point T is in box of the polygon Π_i. For more finding box on a polygon, see ??.

and if it is in box then the Ray method is applied to all segments of the Π polygon. This approach requires crawl the entire list of points $(pt_0, pt_1, ..., pt_n)$ and compare to find the minimum and maximum of each coordinate. This check is done quickly, as it comes down to comparing two numbers. This approach is justified because the number of segments in a polygon is growing very fast. Depending on the complexity of the input polygons and the allowed angles of rotation, the polygon for which we check Π can reaches 300-400 segments. As this check is repeated n a number of times. The

complexity of the algorithm is $\mathcal{O}(n^2)$, but if the method set out in **??** is applied, then the complexity can be reduced to $\mathcal{O}(n)$.

Ray crossing method

Check if a point T is in a given polygon Π. Recall that the area surrounded by the landfill is closed, i.e. including the border. For this purpose we build a horizontal half-line

with the beginning the given point T and the end point T_∞, which we will call the intersecting ray $Ray = list(T, T_\infty)$. Under a point in infinity we will understand the largest number that can be generated from a given CAD system. In most cases, 10^{20} is sufficient as an approximation to infinity. The idea of the intersecting beam method is based on the number of intersections with the polygon Π. If the bore of the intersection points is even, then the point T is outside the polygon, otherwise it is inside the polygon.

In special cases, when the intersections of the ray Ray with the segments of polygon Π coincide with a given vertex of the polygon Π inaccurate results are obtained. To solve the problem of coincidence of the transverse current with a given vertex of the polygon, a criterion for "Ascending" and "Descending" segment is introduced.

Balanced sum of angles

The task is to find the oriented internal angle between the vectors. $\vec{a} = list(T, pt_i)$ и $\vec{b} = list(T, pt_{i+1})$.

We will now give criteria for when a point T is inside or outside a polygon $\Pi = list(pt_0, \ldots, pt_n)$. We construct the vectors $\vec{a}_i$ from the point T to pt_i, $i = 0, \ldots, n-1$ and find the oriented angles α_i between $\vec{a}_i$ and $\vec{a}_{i+1}$, $i = 0, \ldots, n-1$. Then we calculate

$$Sum_\alpha = \sum_{i=1}^{n} arccos\alpha_i \tag{2.20}$$

If the point T is interior for the polygon Π, then the sum of all interior angles $Sum_{\ alpha}$ is equal to $\pm 2\pi$. With a calculation of all angles (crawling), this method gives us two important characteristics for a given polygon Π:

1. If $Sum_\alpha = 2\pi$ polygon Π is oriented CW;
2. If $Sum_\alpha = -2\pi$ polygon Π is oriented $anti - CW$.

This is a very reliable method, but not fast enough according to cite BGSIAM 2017. Also from critical the error that accumulates when adding the corners is also important. Usually the values of the angles are small with a large number of segments of polygon and then a large enough error can accumulate, which in turn, it will be difficult to judge whether the point T is in the polygon Π.

2.11 Add points in a linear segment

Adding points in a linear segment is necessary to find more possible valid ones locations of the polygons $P_i, i = 1, dots$, in the polygon Δ. See point 4.4. Now let's take a certain polygon $P = list(e_1, e_2, \ldots e_n)$ from the incoming list of polygons. Each segment e_i of the polygon $P = list(e_1, e_2, \ldots e_n)$ is divided into three subsegments. Detailed points are added for each sub-segment. The principle of placing detailed points.

Obtaining detailed points for the segment $e_i = list(pt_i, pt_{i+1})$ is done by polar coordinates (base point, angle and length). We define the function *polar* which returns

a point at a given base point, angle and length. The base point is pt_i. The angle of the segment e_i with respect to the abscissa axis OX is $angle(pt_i, pt_{i+1})$. Finding the length of each segment is as follows:

1. Divide the segment e_i into the ratio $L_1 = 0.1(distance = pt_i, pt_{i+1})$.
2. L_1 is divided by the corresponding number of segments we want to achieve. 3 or 5 divisions can be used.
3. The first detailed point will be $pt_{L1,i} = (polar(pt_0, ang_e 1, \frac{L_1}{5})$.

The calculation of the other detailed points for the segment e_i is done in the same logic as for the point $pt_{L1,i}$. The addition of these detailed points along the boundary of the landfill in some cases increases the quality of valid solutions. When testing, if I place the polygon Π_i at the vertex pt_0, then we will not get a valid solution (placement of the plate). However, if the polygon Π_i is placed in any of the detailed points, there will be a valid solution.

2.12 Remove redundant points from a linear segment

The inspection is performed before the arrangement of the polygons begins. When representing polygons as a list of vertices $list(pt_0, pt_1, \ldots, pt_n)$ is it is possible to describe a triangle with more than three points. So the number of the vertices (points) in a list do not determine the type of figure.

2.13 Crossing of two polygons

The function for finding the section (intersection) of two polygons A and B is one of the main operations with sets and is often used in cutting algorithms. Unfortunately, the implementation of this operation requires a lot of CPU time. The classic approach is to check whether each segment of the B boundary intersects segment of the boundary of A. This method is easy to implement, but as slow as possible, there is a complexity $\mathcal{O}(n^2)$, [10], p. 21. This approach is not used in the software developed by the author.

Faster methods for checking whether two polygons intersect use finding the points of intersection of their boundaries with their coordinates or verification for logical intersection. In most cases we will use logical intersection of the two polygons, then it is not necessary to know the number or coordinates of the intersection points, suffice it to know that one of the two polygons has at least one vertex that is inside the other landfill. This is also used when searching for a possible position of a polygon Π_i relative to the main polygon Δ. In the section 4.4 for 2D cutting we will give more information.

One of the possible cross-checks is by starting to emit rays from each vertex of the B polygon. And if a vertex from the polygon B is in the polygon A, then the two polygons A and B intersect. Not in this approach it is mandatory to check all points of the polygon B whether they are internal to the polygon A. The method is reliable, but its complexity is almost $\mathcal{O}(n^2)$, since each ray is actually a segment with initial vertex B_i. This approach is relatively faster than the classical method.

tab [0.5 cm] One of the fastest algorithms is that of the "Bentley – Ottmann algorithm", [39], which is concludes with the introduction of a vertical or horizontal "scanning" line passing through all segments. When reaching the beginning of a segment, we report "event" and when we reach the end of the segment we also have an

"event" the algorithm uses a vertical scan line. According to [43] the complexity of this algorithm for all K intersections between the N segment is $\mathcal{O}((N+K)\log N)$.

2.14 Reduction of polygon vertices

In the process of searching for a valid solution along the contour of the polygon, detailed points are "inserted", which increase significantly the likelihood of finding a valid solution. In the algorithm developed by the author [13], 15 detailed points are placed on each segment. Three intervals of five points. This area of search for solution-these detailed points. For each of them, the input polygon is translated and rotated. until a valid solution is found. A valid solution is that the input polygon is in the fill polygon. Without crossing the two polygons.

Reducing the polygon will significantly increase the speed of the algorithm and save accordingly computational time. To reduce the polygon, we must determine its direction of construction. Assume that the direction of construction of the polygon Δ is CW. We open a new blank list and start traversing the polygon Δ with every three points $list(pt_{i-1}, pt_i, pt_{i+1})$, $i = 1 \ldots n$. We calculate the direction of rotation of the points $list(pt_{i-1}, pt_i, pt_{i+1})$. If their direction coincides with the direction of rotation on the polygon Δ, we check whether $(getPolyArea(pt_{i-1}, pt_i, pt_{i+1}) < minArea$ and if so it we do not write a point pt_i in the empty list, otherwise we write pt_i in the empty list. If the direction does not match on rotation of $list(pt_{i-1}, pt_i, pt_{i+1})$ with Δ we write pt_i in the list.

Chapter 3

Task for $1D$ Cutting Stock Problem.

3.1 Formulation of the task.

The problem of optimal cutting of elements of a given polygons (plates) dates back to the beginning of the industrial revolution, the second half of the 18th century and the beginning of the 19th century. It is typical for this period of time the exponential development of the productive forces. The industrial revolution is connected not only with the beginning of the mass use of machines, but also with the sharp rise of labor productivity. High labor productivity is directly and directly proportional depending on the consumption of raw materials. Hence the need for optimal use of resources in production. The task of optimal linear cutting mainly affects industry. Industry is a sector that includes the extraction of minerals and the processing of raw materials in intermediate or final products. Conditionally, we can divide the industry into two extractive sectors and processing. The secondary sector also includes construction. Our task stems from the secondary sector - construction. Several types of materials are mainly used in construction: Reinforced concrete, steel, wood and others. In this case, we will focus on steel and wood constructions. These constructions allow to be produced in a workshop and to be installed on the construction site. The production of both types of materials (steel and wood) allows cutting. Let's take for example, the steel structure shown in figure 1.1.

In this construction, cross-sections of various types are used for the rod elements. Double "" T "profiles," L "profiles," C "profiles.

Sections of steel profiles often reach 100 kg/m. At a price of one kilogram of steel of the order of BGN 3.5 lv./ kg. (as of 2021) makes BGN 350 per linear meter. And when we can make savings that are repeated for each account, then the benefit of optimal cutting is obvious.

A list of input profile lengths $L = l_i$ is given. The solution of the problem for $1D$ will be reduced to finding a solution for Bar_i. Or this is the linear arrangement of part of profiles l_i in a given length Bar_i. The next steps are until all accounts in the L list are exhausted. Optimization involves locking the profiles so that get the smallest possible remainder for each given length Bar_i, see figure 3.1.

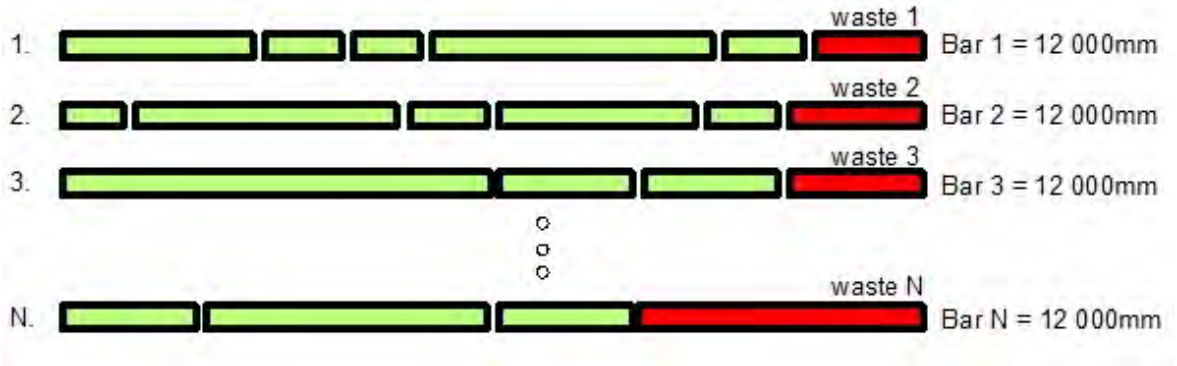


Figure 3.1: Defining a linear cut.

Where Bar_i, $i = 1 \ldots n$, see Figure 3.1, the target fill length for finding one profile layout is displayed. The problem will be solved with the method of ants ACO. The ant method is a metaheuristic method for solving computational problems [16]. This algorithm is part of the algorithms of Social Intelligence (SWARM models of the evolution of culture).

In this case, it is 12,000 units. It cannot be less than the smallest length of the input profiles. To find a valid solution, we take the one with the smallest value of the remainder $waste1$ of them among all the solutions found. We must note that $waste1$ consists of two wastes. One waste is real - that which remains as material stored in the variable $wasteReal$. The other is technological, written in the variable $wasteCut$. This is the width of the cutting tool. Therefore, the total waste is $waste_i = wasteReal + wasteCut$. Then the profiles included in the selected solution (green color of figure ref fig: 1D-Waste) exclude them from the input list. The task is repeated until as the incoming list becomes empty. Then we write in a variable $sumWaste = (waste_1 + waste_2 + \cdots + wasten)$ scrap from all profiles and save the solution. Once we get the next solution we compare them by $sumWaste$. We choose this with the smaller $sumWaste$. The decisions are repeated or until we get a waste less than 5 % or up to a given calculation time.

3.2 Finding a complete solution for $1D$ cutting.

Let the following incoming list of cutting profiles be given. The incoming list is obtained directly from the CAD system. One such list is shown in table ref tbl: ProfileInput.

Input data for $1D$ cutting:

Table 3.1: Input data for profile cutting.

n	Section	Count	Length
1	2	3	4
1	Profile 1	36	320
2	Profile 12	54	330
3	Profile 8	4	330
4	Profile 15	18	334
5	Profile 31	54	340
6	Profile 25	54	350
7	Profile 19	360	365

With n we will mark the "collapsed" list with 18 items. With N we will mark the developed list with 580 items. Three stacking methods were tested. The first is combinatorial optimization by the method of ants ACO, [53] and [54]. An ant was used to find a solution for a Bar_i.

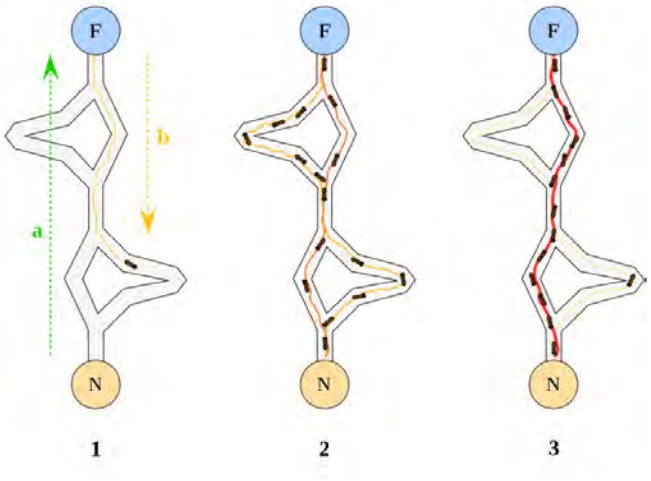


Figure 3.2: Illustration of the ant method.

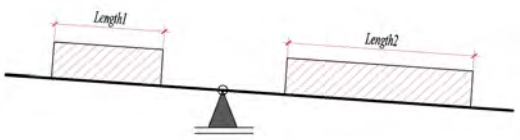


Figure 3.3: Value.

The Ant Colony Optimization method is part of the population methods. Population methods are part of Metaheuristics. See figure 4.1. In general, this is a probabilistic approach to solving a variety of computational problems. The ACO method

solves problems according to given parameters. This method was first proposed by Prof. Marco Dorigo in 1992 in his dissertation.

From then until today, the method proposed by Prof. Dorigo has been extended and modified to be able to solve a wider range of tasks. The idea is taken from nature, so here we will use terms like "ant" and "pheromone". A pheromone is a chemical that is released from the ants in the process of passing a road. This pheromone serves to communicate with other ants. In the beginning, the ants move at random. When they find food, they return to their nest. All the way back and forth they release a pheromone. This pheromone is deposited in their path. The pheromone attracts ants. The more pheromones there are in a given road, the more likely the ants are to move on it. In this way, the amount of pheromone increases and the path to the food source becomes more attractive to other ants. A different number of ants can be used to solve each task. The fewer ants are used to find the global optimum less current computing resources (CPU time) will be required. In the present case, an ant was used. The ant selects any valid element and places it in the valid solutions. For the next solution, it uses a function called transition probability. This feature is a product of the amount of pheromone and heuristic information. The amount of pheromone represents the experience of previous iterations of ants. The heuristic function is information representing prior knowledge of the task. The ant chooses this transition, which has a great deal of pheromone and heuristic information. This is the highest probability of a correct decision. If there is more than one solution with equal probability, then one of them is chosen at random. Once all the ants have found their solutions, the pheromone should be renewed. First, the pheromone is reduced to reduce the impact of previous decisions. A new pheromone proportional to the value of the target function is then added. The logic is that solutions with more pheromone are better than those with less pheromone and so they will become more desirable in the next iteration. In the specific task, the pheromone is placed on the transitions.

Probability of transition.

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum (\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}, \tag{3.1}$$

where:

- $\tau_{i,j}$ is the amount of pheromone corresponding to the transition from vertex i to vertex j;
- α is a parameter to control the influence of $\tau_{i,j}$;
- $\eta_{i,j}$ is heuristic information. Combination of the parameters of the objective function and the constraints;
- β is a parameter to control the influence of $\eta_{i,j}$.

Pheromone Update.

$$X = \begin{cases} L_k, & \text{if an ant } K \text{ passes through the edge } i,j \\ 0, & \text{else} \end{cases} \tag{3.2}$$

Algorithm of the *ACO* ant method, according to cite ESGI'113.

Algorithm 2 ACOAlgorithm

```
/*Algorithm of ants*/
  procedure ACOAlgorithm
  Begin
      Placing an initial pheromone
      While while the criterion is true do
          Place each ant at the top vertex
          Repeat
              For each , apply for each ant
                  Choose next vertex
              End Foreach , end of apply for each ant
          Until , each ant is construct a solution
                  Pheromone renewal
      End While , end of while
  End
```

Since the creation of the algorithm in 1992 by Prof. Dorigo until today, various modifications of the ACO algorithm have been developed. Some of the most popular variants of the ant optimization algorithm are presented in [17], [18], [19], [20], [21], [22], [23], [53] , [54],

In the current $1D$ task, the largest pheromone is given to those profiles that give the smallest remainder of Bar_i. The length of the profile is its weight because the cross section is the same. It is a constant. See figure 3.3. The $Length2$ profile will receive a higher pheromone (rating) than the $Length1$ profile. Or these are longer profiles will have a higher rating. The sum of the pheromone for Bar_i is larger for longer profiles. In this case, the ant method exhibits a *Greedy* character.

The second method is dynamic optimization. This approach does not make combinations between profiles, but records the sum of the accounts due to an index. In the next search, the sum of the lengths is not passed through the entire list to the given position, but their sum from previous calculations is used. Give a list of profiles with their lengths $L_p = list(p_0, p_1 \dots p_n)$. The sum of the lengths of the first 5 profiles will be:

1. $sum_1 = p_0 + p_1$;
2. $sum_2 = sum_1 + p_2$;
3. $sum_3 = sum_2 + p_3$;
4. $sum_4 = sum_3 + p_4$;
5. ...
6. $sum_i = sum_{i-1} + p_i$;

The third method is a combination of *Greedy* and a combinatorial method with complete depletion of the n-element, k-th class. In *Greedy*. method profiles are sorted by length. From largest to smallest. The arrangement of the profiles starts with the largest lengths being placed first. The next placement is the combinatorial method. Method the largest length of the sum of the lengths of each of the three elements in the list L_p. The combination class is limited to three elements. Each combination differs from each other by at least 1 element.

$$C_n^k = \frac{V_n^k}{P_n^k} = \frac{n!}{k!(n-k)!} \tag{3.3}$$

Or for ten profiles we have the following number of combinations with three elements:

$$\frac{10!}{3!(10-3)!} = \frac{3628800}{6.5040} = 120 \tag{3.4}$$

The lengths of profiles with the same cross section do not vary widely. In the comparisons made between the three methods, for the purposes of steel structures the hybrid approach *Greedy* + n^3 the algorithm gives the best results in terms of density and time. The combinatorial method works with the "collapsed" list, we will mark with the n element = 18 pieces. The maximum number of iterations is $7^3 = 343$. It is not necessary to reduce the number of items n by one after each profile placement. In the developed list we will mark with $N = 580$ pieces. The maximum number of iterations should be $580^3 = 19,511,000$.

Cutting lengths 12000 mm, cutting knife width 0. mm. The number of lengths (profiles) for cutting is unlimited. Number of profiles for cutting - 580. The total length of the profiles for cutting is 205 332 mm. Therefore, the global minimum will be around $\frac{205332}{12,000} = 17.11$, or up to 18 12,000m profiles.

3.3 Results at $1D$ cutting. Examples.

We will use the following commercial product:

Figure 3.4: Comemercial product for $1D$ CSP.

The Results of calculation:

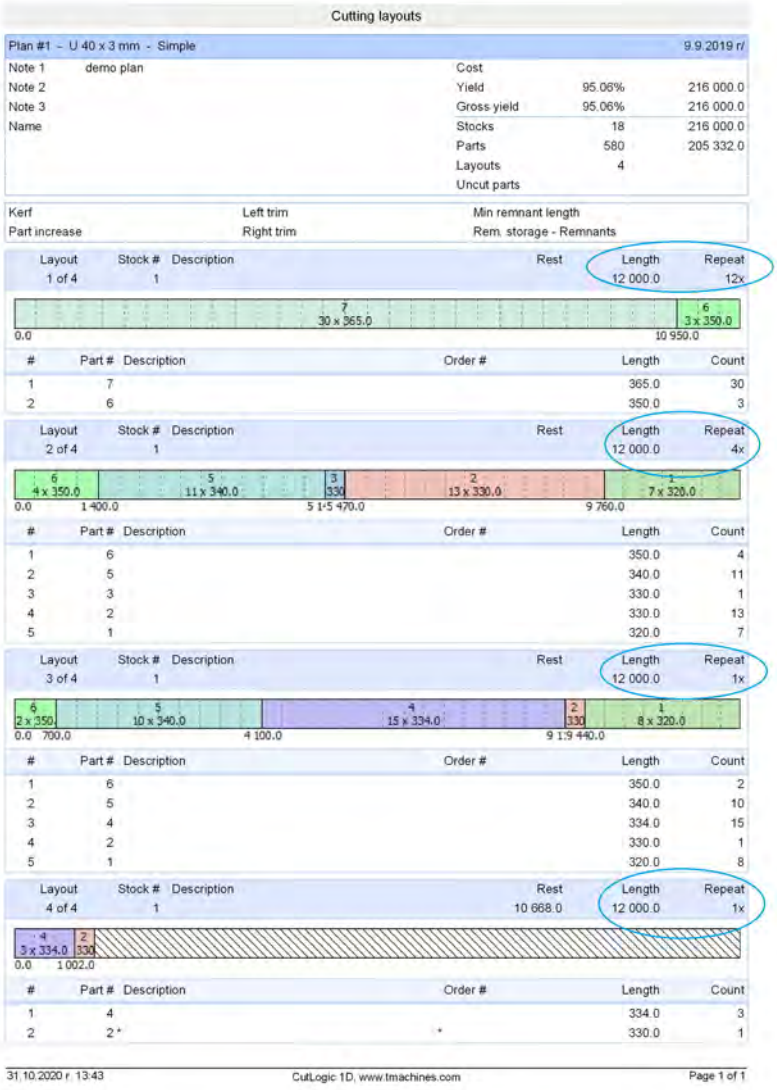


Figure 3.5: Solution of $1D$ cutting with a commercial product.

The blue ellipses in figure 3.5 indicate the number of required profiles. They are 17 whole lengths of 12,000 mm + 1002 mm of the 18th length.

Results in this method:

Table 3.2: Results with Ants for $1D$ cutting.

N	Дължини	Остатък	Профили
1	2	3	4
1	12000	120	profile 19 (360x33),
2	12000	120	profile 19 (360x33),
3	12000	120	profile 19 (360x33),
4	12000	120	profile 19 (360x33),
5	12000	120	profile 19 (360x33),
6	12000	120	profile 19 (360x33),
7	12000	120	profile 19 (360x33),
8	12000	120	profile 19 (360x33),
9	12000	120	profile 19 (360x33),
10	12000	120	profile 19 (360x33),
11	12000	120	profile 12 (330x36),
12	12000	120	profile 31 (330x36),
13	12000	100	profile 25 (350x34),
14	12000	120	profile 1 (320x36), profile 19 (360x1),
15	12000	160	profile 19 (360x29), profile 25 (350x4),
16	12000	120	profile 31 (330x18), profile 12 (330x18),
17	12000	130	profile 15 (330x18), profile 25 (350x16), profile 8 (330x1),
18	12000	11010	profile 8 (330x3),

Table 3.3: Резултати с *Greedy* + n^3 method for 1*D* CSP.

N	Length	Waste	Profile
1	2	3	4
1	12000	150	profile 1 (320x36), profile 12 (330x1),
2	12000	120	profile 12 (330x36),
3	12000	120	profile 12 (330x17), profile 8 (330x4), profile 15 (330x15),
4	12000	120	profile 15 (330x3), profile 31 (330x33),
5	12000	170	profile 31 (330x21), profile 25 (350x14),
6	12000	100	profile 25 (350x34),
7	12000	180	profile 25 (350x6), profile 19 (360x27),
8	12000	120	profile 19 (360x33),
9	12000	120	profile 19 (360x33),
10	12000	120	profile 19 (360x33),
11	12000	120	profile 19 (360x33),
12	12000	120	profile 19 (360x33),
13	12000	120	profile 19 (360x33),
14	12000	120	profile 19 (360x33),
15	12000	120	profile 19 (360x33),
16	12000	120	profile 19 (360x33),
17	12000	120	profile 19 (360x33),
18	12000	10920	profile 19 (360x3),

Table 3.4: Comparison of results for 1*D* cutting.

Software	Number of Accounts Used	Time [s]
1	2	3
Commercial product	17x12000 + 1002 mm	7
Greedy + n^3	17x12000 + 1080 мм	<1
Ants *ACO*	17x12000 + 990mm	1

Obviously, the commercial product uses very complex heuristics or other optimization methods. The commercial product is the slowest compared to the other two methods *Greedy* + n^3 and *ACO*. As can be seen from the comparison table ref tbl: Compare1D, the ant method *ACO* gives the best result in terms of least waste, which is our main goal. As a computation time *Greedy* + n^3 is slightly faster than *ACO* at the expense of the worse solution. The commercial product is slow and gives a worse solution than the *ACO* ant method. Therefore, we can conclude that the algorithm proposed by the author of the dissertation *ACO* is superior to the other two.

Chapter 4

Task for $2D$ Cutting Stock Problem.

4.1 Formulation of the task.

The industry sets a variety of optimization tasks to solve. These tasks can be classified on many grounds depending on:

1. the nature of the problem to be solved;
2. task structure;
3. the number of control parameters;
4. the nature of the dependence of the criterion and the constraints of the parameters;
5. the presence of various restrictions;
6. the nature of the required minimum;
7. number of criteria;
8. and others.

In the production of steel structures it is necessary to cut plates from a given steel sheet. The plates come as polygons from a CAD system. This requires arranging the input plates on the sheet so as to obtain minimal waste. This is the cutting of a certain number of figures from a given material, which in the general case will be a polygon. We will call this polygon a polygon to fill. See figure 1.4.

This task is also known as Cutting Stock Problem or (CSP), [69]. This problem is an NP-complex combinatorial task [88]. The literature gives exact solutions to the problem for figures (planks) that are rectangles. It will be given below an algorithm for finding a solution to the problem of arranging a given number of arbitrary geometric figures (plates described by polygons) inscribed in any contour (polygon to fill). The method allows use of rotation and a mirror image of the figure. CPU computing time increases significantly with increasing the number of figures and their complexity as geometry. Finding a solution by exhausting all possible combinations is unacceptable as too large a calculation. In the modern development of computer technology it is possible to solve a complex task on a super computer that will find all possible solutions, but in most cases this is not justified. Of course, the cost of a significant computing resource depends on the importance of the task. The algorithm presented below has the possibility to parallelize the computational processes. But the mass tasks will

be realized on a desktop or personal computer. The aim is to create an algorithm that gives in a short time an acceptable solution to complex combinatorial tasks using mobile computing devices. We will introduce all the mathematical concepts that are used to describe the mathematical model and algorithm for solving the optimization problem.

The CAD system produces polygons, which are lists of points in the coordinate of the XOY system. In the general case, these polygons also contain points that are not vertices of the polygon, ie. points lying on one right. We will call these points redundant. Therefore, all polygons generated by the CAD system will we apply the function for clearing the excess points. After removing the excess points we get input allowable polygon described by a list of points

$$P = list(pt_0, pt_1, ...pt_n), \tag{4.1}$$

where pt_i are the vertices of the polygon. This list has the following properties: 1. The list is ordered; 2. The list is cyclic $pt_n = pt_0$; 3. There is no self-crossing.

In the process of working on the algorithm we will need the concept of a segment, which is a segment between two consecutive tops. In this way we generate the segments $e_1 = list(pt_0, pt_1), \ldots, e_n = list(pt_{n-1}, pt_n)$ and we get another characterization of the polygon $\Pi = list(e_1, e_2, \ldots, e_n)$. Plates with curve boundaries are approximated by polygons with a sufficient number of vertices. Examples of input valid polygons are shown in Figure 1.5.

A polygon to fill Δ to be filled will also be described with a list of points:

$$\Delta = list(p_0, p_1, \ldots, p_m) \tag{4.2}$$

The filling contour must not be self-intersecting.

There are two criteria for task optimization:

1. Minimum fill height optimization - $minY$;
2. Optimization of the number of vertices of the residual polygon after cutting of the input polygon from the fill polygon - $minVertex$.

Let's consider optimization of the minimum height.

Of all the dispositions, we will consider as the best the one with the smallest ordinate in the coordinate XOY system. If more than one solution with the same ordinates $minY$ is obtained, then we take this with best fill factor. If there is more than one solution with a maximum coefficient of fill we choose those solutions that cut the fill polygon with the smallest number of vertices. This means that the cut achieves "correct" shapes. If there is more than one solution with the smallest number of vertices, then select the first in the list.

To find the face of a polygon, see the formula 2.12. This formula gives the double face of the polygon. Then we define the fill factor $ratio$ as:

$$ratio = \frac{\sum_{i=1}^{n} F_i}{A_P} \tag{4.3}$$

Therefore, the minimum value $ratio = 0$ of the coefficient is when the polygon A_P is not filled. The maximum value $ratio = 1.0$ is at maximum fill of A_P.

Of course, the definition of $minY$ leads to the situation of how the fill polygon P is introduced. If we want to avoid this problem, then the solution of the problem will

have to take several different angles of the filling polygon P. The angles of rotation will be the angles that each segment concludes with the abscissa axis. The number of rotations is equal to the number of segments of the respective polygon.

Consider optimization of the number of vertices of the residual polygon.

Of all the layouts, the best will be considered what when cutting the input polygon in the fill polygon has at least the vertices of the residual polygon. In the present dissertation, this is the criterion for a polygon to be "smoother". Finding a suitable polygon from the input polygons will be done by comparing the sides of the two polygons - the input and the filling polygon. If we have a complete match of the lengths of the sides of the two polygons, then we will choose this input polygon. If there is no coincidence of the sides of the input polygon to the filling polygon, then we will use the coincidence of the angles of the two polygons.

If more than one solution with the same number of vertices of the residual polygons $minVertex$ is obtained, then we take the first decision in the list of valid decisions. The difference between $minY$ and $minVertex$ is that with $minVertex$ large (long) polygons are allowed to "enter" the filling polygon first, because they are more likely to produce a smooth residue. The minimum smoothness of the residual polygon that can be obtained is a polygon with three vertices $list = (pt_0, pt_1, pt_2)$ with an area other than zero.

4.2 Strategy for selection of an input polygon. Metaheuristic methods.

Metaheuristics is a powerful tool for finding the optimal or suboptimal solution of complex combinatorial problems. A key role in the development of metaheuristic methods is the need to find an acceptable solution for an acceptable time with limited hardware resources. According to cite Wiki: Meta, metaheuristic algorithms (metaheuristic algorithms, in short: metaheuristics, metaheuristics) in computer science are algorithms for mathematical optimization, which solve combinatorial optimization problems. These tasks are generally complex, represented by sampling the input data. Such tasks are usually characterized by strong nonlinearity, many parameters, various complex constraints for satisfaction and many - often contradictory - optimization criteria.

Even with one optimization criterion, there may not be a single valid solution. Only then there is no optimal solution. If there is even just one acceptable solution, then there must be an optimal solution.

In general, finding the optimal or even close to the optimal solution is difficult to achieve. The term "metaheuristics" was introduced by Fred Glover in his founding article in 1986 as an upgrade of the term "heuristic" algorithm, which in the broadest sense means a trial-and-error solution-finding algorithm. "Means" beyond "," super "," at a higher level "and the metaheuristic algorithm means a" higher "strategy that guides and modifies other heuristic algorithms to achieve solutions better than those that would normally be obtained when searching for a local optimum [35], [36]. In addition, all metaheuristic algorithms balance between global and local search. Qualitative solutions to difficult optimization problems can be achieved in a reasonable (ie polynomial) time, but without a guarantee that (global) optimal solutions will be achieved. The two main components of any metaheuristic algorithm are: intensification and diversification, or exploration and exploitation. Diversification means generating a variety of solutions so that the search space can be explored over a wide range, while intensification means focusing demand on a local region, knowing that the current best solution is in that region. When selecting the best solutions, a good balance must be found between intensification and diversification in order to improve the rate of convergence of an algorithm. The choice of the best current solution ensures that the solutions will converge to the optimum, while the diversification by choosing random values of variables allowing to avoid falling into a local extremum and the same time to increase the diversity of the solution. A good combination of these two main components usually leads to finding a global optimum. According to [27], the basic properties of metaheuristics can be summarized as follows:

1. Metaheuristics provides strategies to guide the search process;
2. Our goal is to effectively explore the search space to find optimal or suboptimal solutions;
3. Metaheuristic techniques cover a wide range of procedures - from local search procedures to complex machine learning procedures;
4. Metaheuristic algorithms are approximate and usually nondeterministic;
5. Metaheuristic algorithms generally provide mechanisms to avoid focusing demand only in limited areas of space;
6. The basic concepts of metaheuristics can be described on an abstract level;

7. Metaheuristic algorithms are universal;
8. The metaheurist can use knowledge specific to the field in the form of heuristics, which is governed by a high-level strategy;
9. To guide the search, modern metaheuristics uses the experience gained in the search;
10. Metaheuristics is a high-level strategy for exploring the search space using different methods;
11. Requires a dynamic balance between the use of two fundamental concepts: diversification and intensification.

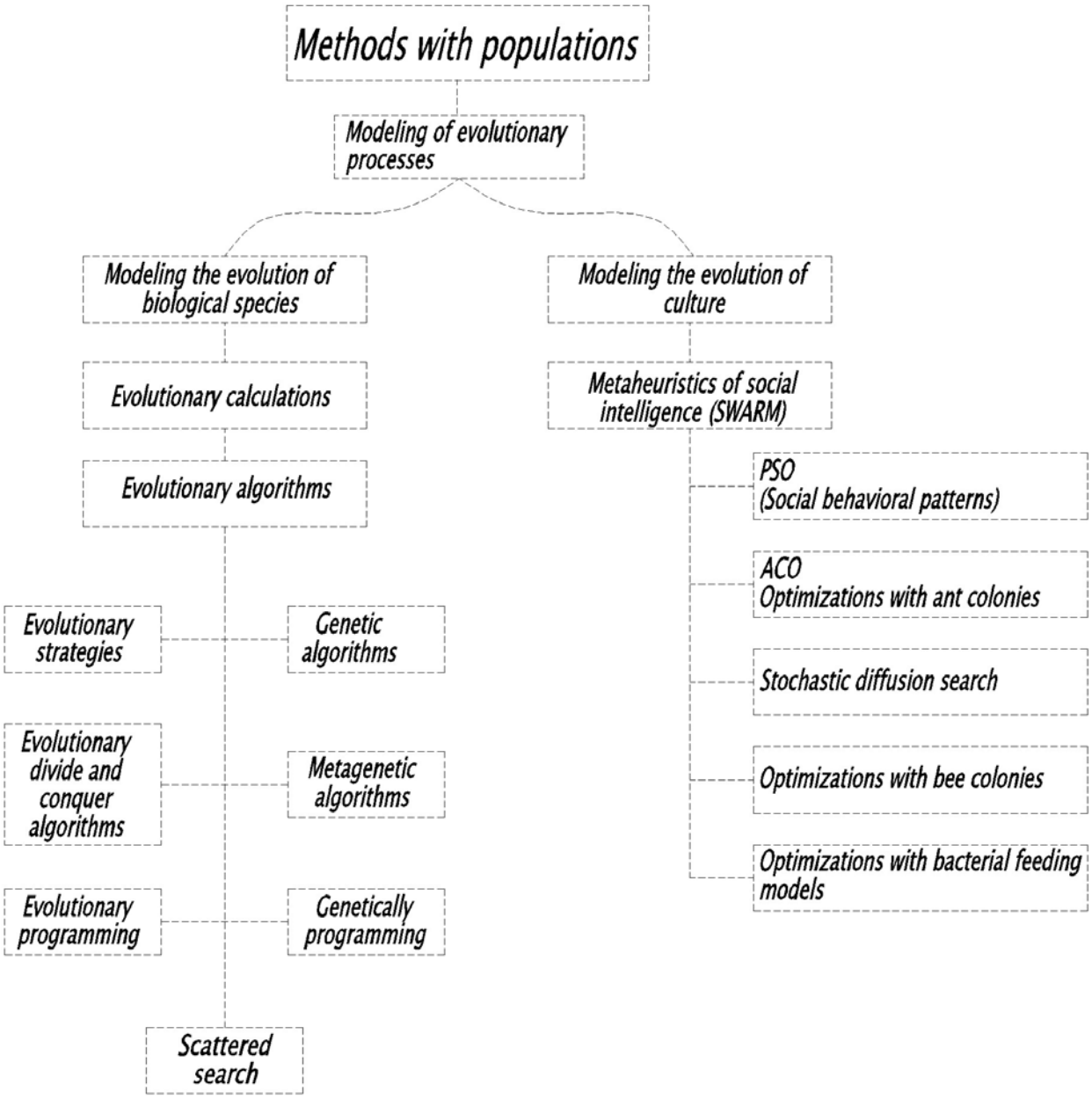


Figure 4.1: Classification of metaheuristics with populations.

Hybrid metaheuristics provides opportunities to increase search efficiency by combining different metaheuristic algorithms. Hybrid metaheuristics is used in the present dissertation. The following strategy was used:

1. "Scattered Search" from the Evolutionary Algorithms of the Population Method See figure 4.1;

2. Probabilistic prediction of the selection of an element;

3. Hierarchical evaluation of decisions.

We have a list of polygons

$$\Pi_L = (list_1(pt_0, pt_1, ...pt_n), (list_2(pt_0, pt_1, ...pt_n)), \dots (list_n), pt_1, ...pt_n)), \tag{4.4}$$

where $list_i$ are polygons describing through their vertices. The vertices are described by a list of two real numbers $list(X, Y)$, see 2.1. We will call the list Π_L a list of input polygons.

The polygon Δ to be filled is described by a list of points:

$$\Delta = list(p_0, p_1, \dots, p_m) \tag{4.5}$$

The filling contour must not be self-intersecting. In the given task the polygon for filling Δ is one in number. If we have more than one polygon, then the solution of the problem is repeated for each of them.

Before selecting an input set, it is necessary to estimate the coincidence of the angles and sides of the current input polygon.

$$\Pi_i = (list(pt_0, pt_1, ...pt_n), i = 1, \dots, n$$

to it the angles and sides of the filling polygon $\Delta = (list_0, pt_1, ...pt_n)$. For this purpose, two new derivative lists of Π_i and Δ are compiled. Accordingly, they contain sequentially arranged lists $list(previosLength, Angle, NextLength)$. It is good for the elements of the two lists to be composed of constructive pairs. The constructive pair is composed of two elements - the first with a name, the second with a variable. It can be written as $cons("name".AnyValue)$. In "name" we write "angle" for angle or "length" for length In AnyValue - any value which can be Integer, Real, String or List.

Or the new derivative lists are:

$\Pi_i = list((list(cons"previosLength"(distance_{pt_n,pt_0}))$
$(cons"angle"pt_n, pt_0, pt_1)$
$(cons"NextLength"(distance_{pt_0,pt_1})))$
...
$(list(cons"previosLength"(distance_p t_n pt_{n-1}))$
$(cons"angle"pt_0, pt_n, pt_{n-1})$
$(cons"NextLength"(distance_{pt_n,pt_0}))))$

Where n the number of vertices of the polygon Π_i. If $(i + 1) > n$ then $i = 0$. The Π_i list is cyclic.

A similar list is compiled on the polygon to fill Δ. The Δ list is cyclic.

Each item in the Π_i List is compared to the corresponding item in the Δ list. The largest number of consecutive matching elements from the two lists will give us the highest probability that the polygon Π_i coincides with the polygon Δ. In order to make a correct comparison of the two lists, we will have to choose which list will be the main one. Below the main list we will understand the one that has a greater length (more elements). In the list $list(A)$ can be either Π_i or Δ. Not necessarily the number of vertices of Δ to be greater than the number of vertices of Π_i. Crawling both lists

will be done on base indices. The first iteration is when the index 0 from the list $listB$ coincides with the index 0 on the list $listA$. Now the first index of $listA$ increases by 1. A second iteration follows when the index 0 from the list B coincides with the index 1 on the list A. As an index 0 from $listA$ goes last in $listA$. Or the $listA$ list has a cyclical behavior. This is repeated as we go through all indexes of $listA$ or the loop is repeated as many times as the length of the list $listA$. For each check we record the number of consecutive matches. The evaluation of the list $listB$ is given by the following formula:

$$k = \sum Ratio_{angle_i} + \sum Ratio_{Length_i}, \tag{4.6}$$

, where

$$Ratio_{angle_i} = (if|(angle_{Ai} - angle_{Bi})| < fuzz\ toreturn 1., else 0.).$$

At the corners we look for a complete match. As Ai is an angle from the list $listA$, and Bi is the corresponding angle from the list $listB$.

Once we have a complete coincidence of the corresponding angles $Ratio_{angle_i} = 1.$, then we proceed to estimate their respective lengths.

$$Ratio_{Length_i} = (if(length_{Ai} > length_{Bi}) \rightarrow return(\frac{length_{Bi}}{length_{Ai}}), else(\frac{length_{Ai})}{length_{Bi}}) \tag{4.7}$$

Condition 4.7 gives coefficients close to or equal to 1., no matter which length is greater $lengthA_i$ or $lengthB_i$. This is because we are looking for polygons whose sides almost coincide.

As can be seen from the formula (4.6) polygons with a larger number of vertices will be more likely for higher matching scores. This is good, because after subtracting the two polygons $A \setminus B$ we get polygon with fewer vertices. The coefficient of the formula (4.6) can be used as an estimate for the similarity or similarity of figures.

4.3 Strategy for choosing a solution from valid placements.

Let's look at two polygons. Fill polygon $\Delta = list(pt_1, pt_2, pt_3, pt_4)$ and input polygon $\Pi_i = list(pt_1, pt_2, pt_3, pt_4, pt_5, pt_6)$. The right Π_i polygon is the input polygon. The left polygon Δ is the fill polygon. The polygon Π_i is not a triangle! It is necessary to add detailed points along the boundary of the polygon to fill Δ. These points will be areas of placement of the polygon Π_i and check whether the polygon Π_i is in the polygon Δ. If the check is satisfied, we record this decision as possible. Rotation of the landfill is allowed. Mirror image not attached. From these valid solutions we estimate those that have the greatest contact length with the polygon Δ and the distance $LengthA = distance(pt_0, pt_M)$. There is a relationship between the length of the support and the length $LengthA$. This dependence is accepted by the author. Of course, other dependencies can be written. The evaluation of each solution is $eval = LengthOfTouch + (2.0 * LengthA)$. It is given the weight of the distance from the center of gravity of the polygon to a selected point of the polygon to fill. In this case, this is the point at the bottom, at the left. Another point can be selected regardless of whether it is from the many points of the fill polygon. After evaluating the solutions, we choose the solution with the green outline of the polygon. We subtract $\Delta \setminus \Pi_i$.

Once we have chosen the first solution, we proceed to choose the second. The selection of an input polygon is done according to the procedure described in point 4.2. Since our goal is the maximum consolidation of the solutions, the choice of the second solution should be sought along the contour of the already found solutions. At this stage, a hierarchical evaluation of decisions is applied. We build a rectangular contour around the selected polygons. We will call it *box*. We will first look for solutions that go into the *box* of valid solutions, if we do not find such we use all valid solutions for evaluation. The evaluation of the solutions inside *box* is also done according to the formula: $eval = LengthOfTouch + (2.0 * LengthA)$. It should be noted here that this procedure for selecting solutions after the first must bypass all incoming polygons and then take this with the highest score. In the software developed by the author, this crawl is not done due to the lack of computational resources, but the proposed method is not limited in this direction. If there is enough powerful hardware plus GPUs, the algorithm will give very close results to the global optimum.

After cutting the two polygons, the new contour is obtained. This loop will serve as a fill loop for subsequent polygons. As you can see new contour does not follow completely old. The new contour is purposefully reduced. For more information on how to reduce the contour, see 2.14.

$$ratio_{Global} = \frac{A_{box}}{A_{\Pi_L}} \leq 1.0 \tag{4.8}$$

4.4 Finding a possible location of the plate in the filling polygon.

Finding a possible arrangement of the plates (represented as polygons) is done by placing an input polygon in the polygon for filling and applying the function of subtracting two polygons. The function is described below. Depending on the requirement of the particular case we can build derived polygons from the input polygon often with permission to apply rotation and mirror symmetry.

The number of rotations on which we can rotate a given polygon is arbitrary. The more angles we have in the rotation list, the more likely we are to get a possible solution. In order to limit the arbitrary rotation of the polygon without a quality solution (the polygon should be in the filling polygon) it is necessary to choose appropriate rotation angles. The angles of the segments of the input polygon with the abscissa axis X can be taken as a starting point. The exact angles $(0; 0.5\pi; \pi; 1.5\pi)$ can then be added. If it is necessary to consider the application of the input polygon, then the number of additional polygons will increase. Some states of rotation will be completely identical in geometry.

Checking whether one landfill is contained in another can be done in two ways:

1. Check for each node whether it is in the fill polygon. This check should be done with a while loop, if the current test point is outside the loop, the solution is dropped without continuing;

2. Check if there is a trivial intersection of the two polygons. If there is an intersection, then the decision is dropped, otherwise - is accepted.

If the first criterion for an optimal solution is $minY$, then only the top of the polygon P can be checked with the smallest ordinate (in this case p_7). But it is best to do for all vertices of the polygon to fill P. Another criterion for choosing a solution is

when a smooth shape is obtained when cutting the two polygons. The criterion for a smooth figure will be the minimum number of vertices after cutting the given polygon from the filling polygon. A combination of criteria is also possible. On the selected valid solutions for $minY$, the criterion of a minimum number of vertices after cutting them with the fill polygon will be applied.

In the presence of a multi-core processor and the language in which the application is written, parallel calculations are allowed. Parallel calculations can be made for the other possible states of the input polygon.

The algorithm is repeated for the generated mirror image of the input polygon. All generated rotation angles are valid for the mirror polygon.

4.5 Elimination of the real waste in 2D cutting.

A cutting tool with a certain cutting characteristic is used for cutting the figures. For this reason, a distance between the polygons must be provided. We enter the parameter $cutW$ for the width of the joint (knife) between the polygons.

This problem is solved as the input polygons are expanded with a strip with a width of $0.5cutW$ and each segment is moved off the range by $0.5cutW$.

4.6 Results at $2D$ cutting. Examples.

In the following pages we will illustrate the cutting of the following input polygons (plates). The examples are taken from a real construction site.

The landfill will be a standard rectangular sheet with a width of 1500 mm. For the input polygons we will show results with the contour line of the polygon shifted by the width of the knife. Width of the knife 5 mm. The dimensions of the plates are in millimeters. Before we start calculating the input polygons, they go through preprocessor processing, which includes:

1. Sorting the slats by thickness;
2. Read their number;
3. Find the contour of the plates.

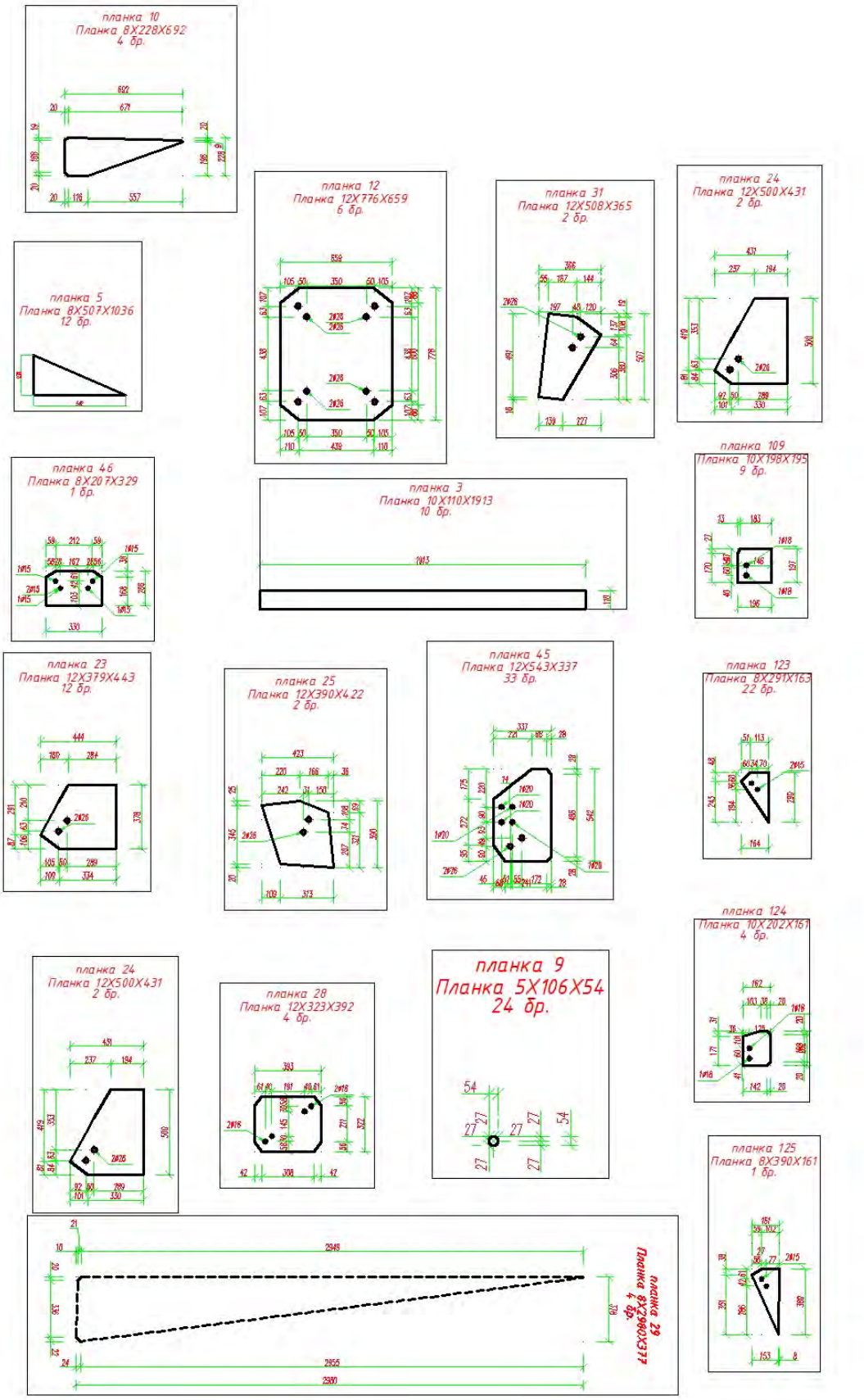


Figure 4.2: Input polygons.

Comparison of the current algorithm with a commercial product.

It takes about three months to make a steel structure from figure 1.1. The number of slats in such a construction is about 1000. The number of unique slats is about 100. The number of iterations is relative. It depends on whether the material is expensive or not. Several iterations can be run on different computers and the best of them can be taken. This is a matter of consumer decision. If hardware is available, several iterations can be made until acceptable waste is obtained.

In the present comparison we will use the bars given in figure 4.4. The total number is 106. With these plates a comparison is made between the commercial product and the developed method in the present dissertation. Finding the final solution with the presented method is for one iteration. More iterations can be made with more homogeneous planks.

Used commercial product.

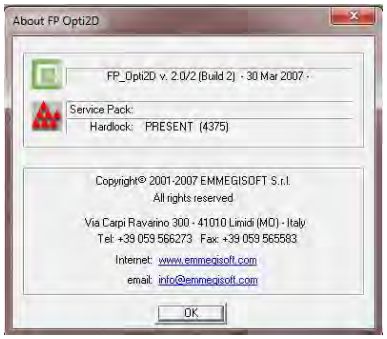


Figure 4.3: commercial product FP Opti2D.

According to the manufacturer's website, the product offers the following functionalities:

1. User friendly graphical interface.
2. Handles panels, metal sheets and glazing.
3. Specific functions for aluminum composite panels.
4. Defines the parts to be optimized.
5. Directly uses the panels and glazing lists generated by FP PRO.
6. Imports work lists from Excel and from external calculation programs.
7. Manages the stock of full sheets and short bars.
8. Graphic display and printing of the optimized sheets, with clear indication of the cuts to be made and layout of the workpieces.
9. Provides statistical information on use of the sheets.

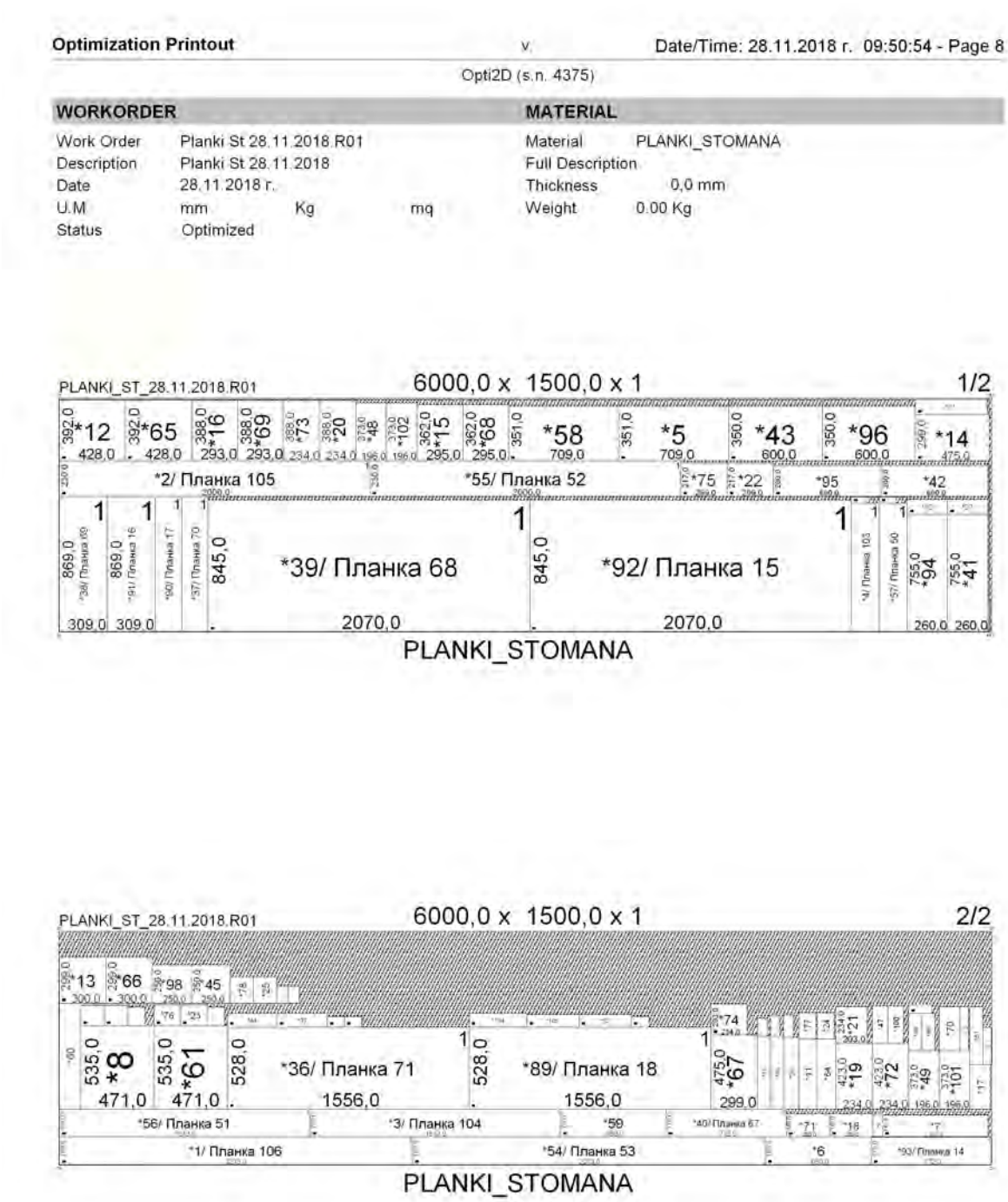


Figure 4.4: Sheet from the printout of the commercial product.

As can be seen from the printout, the commercial product version V.2.0 / 2 (Build 2) only works with rectangular plates. It is allowed to rotate the plates. According to

the operator, the optimization lasted about 20 minutes or 1200 seconds. Dimensions of the filling sheet 1500mm / 6000mm. Operating time is not specified in the printout. Waste 18.2 %. All plates have an offset contour of 5mm. We will work with this contour in the calculations. Actual waste can be calculated as follows:

1. Total used area - 2 sheets x 1500mm x 6000mm = 18,000,000 mm^2;
2. Real area of all 106 plates - 9,859,421 mm^2 ;

The real waste is 8,140,579 mm^2. Or 45 % waste. Actual occupancy 54 %.

Results of the presented method in the present paper. Number of plates are 106. All plates have an offset contour of 5 mm from their actual contour. Permitted rotation of the plates: Yes. Outline: $offsetPtL$.

Table 4.1: Comparison of a commercial product and the presented algorithm

Turn On parameters	Contour	Count plates	Ratio	Time [s]
1	2	3	4	5
(a) Mirror:Yes, Rortate:Yes, Intervals:No	*box*	106	0.72	18 776
(b) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.70	109 519
(c) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.71	227 846
(d) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.69	7 555
(e) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.76	41 031
(f.1) Mirror:Yes, Rortate:Yes, Intervals:No	*box*	51	0.71(0.67)	2 194
(f.2) Mirror:Yes, Rortate:Yes, Intervals:No	*box*	55	0.57 (0.44)	2 454
(Commercial product) Mirror:N/A, Rortate:Yes, Intervals:N/A	*box*	106	0.54	1200

For the case $f.1$ and $f.2$ in brackets are given the fillings of the plates relative to the base sheet 1500mm x 6000mm. The same parameters were used as for the commercial product. The commercial product has a number of limitations. Some of them are that the figures are approximated to a rectangle, a mirror image of the figures is not used. The angles of rotation are reduced to two: 0 ° and 90 °. In terms of the fill ratio *Ratio*, the presented algorithm is much better. see table 4.1. The odds are 0.71 for the current algorithm compared to 0.67 for the commercial product. The second comparison is 0.57 for the current algorithm compared to 0.44 for the commercial product. With large volumes of work or expensive material from which it will be cut the difference increases even more in favor of the presented algorithm. The algorithm presented in this dissertation is better than the commercial product because it gives a higher percentage of compaction of the figures. Another advantage is the very good one its suitability for parallelization of calculations.

Chapter 5

Conclusion

1*D* Cutting Stock Problem.

As can be seen from the comparison table 3.4 the ant method (*ACO*) gives the best result in a very short time. In this case, the *ACO* method exhibits the character of a *Greedy* algorithm. The *ACO* algorithm is better than the commercial product both in time and in optimization. For large decoupling volumes and computers with weaker processors, the *ACO* method is very suitable.

2*D* Cutting Stock Problem.

After the tests of different types of boards, the conclusion is that a larger number of iterations are needed for an acceptable solution of a given problem. The results in the present dissertation are in 3 iterations. Three iterations are accepted because finding a solution takes considerable time. For some types of plates this number is insufficient. The tests were performed on a Windows ®10 Pro, x64 desktop computer. Intel textregistered Core (TM) i5-9500@3.0 GHz processor. Used processors one. CPU CPU type. Although the processor is one of the last generations at the time of writing this paper is proving weak for a higher degree of compaction of the plates. But if you are looking for a relatively fast arrangement and a small number of boards, the desktop computer can handle it. The presented approach to solving the problem can be applied in 90 % of the cases in practice. It should be noted that a slightly higher density of the solution requires significantly more calculation time. Whether time will be sacrificed at the expense of material depends on how expensive the material from which the figures will be cut is expensive. A further development of the problem will be its development for hardware with sufficient computing resources based on GPU processors. The results of this dissertation have been reported at various national and international conferences.

5.1 List of publications

1. Evtimov G. Fidanova S. **"Subtraction of Two 2D Polygons with Some Matching Vertices"**, pp. 80-87, *Numerical Methods and Applications, 9th International Conference NM&A'18, Borovets, Bulgaria, 2018*, Geno Nikolov, Natalia Kolkovska, Krassimir Georgiev (eds.), ISBN 978-3-030-10691-1, doi:10.1007/978-3-030-10692-8_9

2. Evtimov G. Fidanova S. **"Heuristic Algorithm for 2D Cutting Stock Problem"**, pp.350-357 *Large-Scale Scientific Computing, 11th International Conference, LSSC 2017, Sozopol, Bulgaria*, Ivan Lirkov, Svetozar Margenov (eds.), Springer, Vol. 793, ISBN 978-3-319-73440-8, doi:10.1007/978-3-319-73441-5_37

3. Evtimov G. Fidanova S. **"Analyses and Boolean Operation of 2D Polygons"**, pp. 107-118, *Advanced Computing in Industrial Mathematics, BGSIAM 2017*,K. Georgiev, I. Georgiev (eds.), Springer, Vol. 793, ISBN 978-3-319-97276-3, doi:10.1007/978-3-319-97277-0_9

4. Evtimov G. Fidanova S. **"2D Optimal Cutting Problem"**, pp. 33-40, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*, K. Georgiev, I. Georgiev (eds.), Springer, Vol. 728, ISBN 978-3-319-65529-1, doi:10.1007/978-3-319-65530-7_4

5. Ana Avdzhieva, Todor Balabanov, Georgi Evtimov, Ivan Jordanov, Nikolai Kitanov, Nadia Zlateva, **Two Dimensional Optimal Cutting Problem, 120th European Study Group with Industry (ESGI'120)**, Problems & Final Reports

6. V. Bodurov, D. Dimov, G. Evtimov, I. Georgiev, S. Harizanov, G. Nikolov, V. Pirinski, **The 2D/3D Best-Fit Problem, 113th European Study Group with Industry (ESGI'113)**, Problems & Final Reports, pp. 62-73, 2015. ISBN:978-619-7223-12-5

7. A. Avdzhieva, T. Balabanov, G. Evtimov, D. Kirova, H. Kostadinov, T. Tsachev, S. Zhelezova, N. Zlateva, **Optimal Cutting Problem, 113th European Study Group with Industry (ESGI'113)**, Problems & Final Reports, pp. 49-61, 2015. ISBN:978-619-7223-12-5

5.2 Approbation of the results

The results in the present dissertation have been reported to different events of the section "Parallel Algorithms" at IICT-BAS such as:

1. 113th European Study Group with Industry (BGSIAM - 2015);
2. 11th Annual Meeting of the Bulgarian Section of SIAM (BGSIAM - 2016);
3. 120th European Study Group with Industry (ESGI'120 - 2016);
4. 12th Annual Meeting of the Bulgarian Section of SIAM (BGSIAM - 2017) ;
5. 13th Annual Meeting of the Bulgarian Section of SIAM (BGSIAM - 2018);
6. Conference on Large-Scale Scientific Computations LSSC'17, Sozopol, 2017;
7. Ninth International Conference on Numerical Methods and Applications NM&A'18, Borovets.

5.3 Yields

Contributions to this dissertation can be divided into scientific and applied science, as scientific contributions concern the development of methods and algorithms for 1D and 2D cutting, and the scientific-applied ones refer to their program realization.

The scientific contributions are:

- An algorithm for optimal cutting in one-dimensional space has been developed;
- An algorithm for optimal cutting in two-dimensional space has been developed;
- A method for two-dimensional cutting based on hybrid optimization has been developed;

The scientific and applied contributions are:

- A program implementation of the algorithm for one-dimensional cutting has been made;
- A program implementation of the two-dimensional cutting algorithm has been made;

The results of this dissertation can be used in various fields of science and engineering practice:

- The design of buildings and facilities;
- The design of the wear of parts in machines as well as the design of mechanisms;
- Earth mechanics - soil consolidation;
- Aviation equipment - finding the optimal path in an environment with obstacles;
- And in many other areas where CAD systems are used.

Applied contributions can also be developed in companies that produce steel structures. The application software can be implemented in other industries that are not related to the construction of buildings and facilities. Another very great application advantage is that the input data is taken directly from the database of the CAD system with which the facility was designed. This repeatedly increases the speed of receiving and accuracy of the data with which the program works. With a few clicks, thousands of polygons can be selected and the cutting program can be started. The software can automatically remove or correct "incorrect" polygons to avoid inaccuracies in the initial results. The solution takes a few minutes depending from the performance of the computer system on which the software is used. The algorithm developed in the presented dissertation, allows the use of a mirror image of the polygons, rotation and other operations, which can lead to a significant improvement of the obtained approximate solution. Of course, for the purposes of large-scale research, the algorithm can be implemented of a supercomputer as it allows significant parallelization of calculations.

5.4 Declaration of originality

Declaration of originality of resultse

I declare that the present dissertation contains original results obtained at research conducted by me with the support and assistance of my supervisor. The results obtained, described and / or published by other scientists, are duly and in detail cited in the bibliography.

This dissertation is not applied for the acquisition of a scientific one degree at another university, university or research institute.

Signature:

5.5 Thanks

I am very grateful to my supervisor Prof. Stefka Fidanova for her help valuable advice, for the interpretation of the results of the decisions and for the overall guidance during the work on the dissertation and especially for the introduction of theof metaheuristic methods in the subject of automatic cutting.

I express special gratitude to Prof. Raycho Lazarov for his help in my work on the article cite NMA'18. The missing link was made in this work in the technology for approximate solution of this complex optimization problem. Without this article, solving the problem would be very difficult.

Thanks to Assoc. Prof. Ivan Georgiev and Assoc. Prof. Stanislav Harizanov for their help in interpreting some mathematical models. Thanks to Dr. Todor Balabanov for the advice during the writing of this dissertation.

Thanks to Prof. Stefka Dimova, who introduced me to this topic through the organization of the 113th and 120th European Study Group with Industry in Sofia;

Thanks to Art. cor. Svetoslav Margenov and the whole team from the Institute of Information and Communication Technologies in Bulgaria Academy of Sciences for the trust and support during my work on the dissertation.

Bibliography

[1] Valerio de Carvalho J.M. **"Lp models for bin packing and cutting stock problems"**, European Journal of Operational Research 141:253–273,(2002).

[2] Chen C.L.S., Hart S.M., Tham W.M. **"A simulated annealing heuristic for theone-dimensional cutting stock problem"**, European Journal of Operational Research93:522–535,(1996).

[3] Foerster H., Wscher G. **"Pattern reduction in one-dimensional cutting stock problems"**, In: Proceedings of the 15th Trienal Conference of the International Federationof Operational Research Societies, (1999).

[4] Falkenauer E., **"A hybrid grouping genetic algorithm for bin packing"**, Journal of Heuristics Vol. 2, 1996

[5] Song X., Chu C.B., Nie Y.Y., Bennell J.A. **"An iterative sequential heuristic procedureto a real-life 1.5-dimensional cutting stock problem"**. European Journal of OperationalResearch 175:1870–1889, (2006).

[6] Suliman S.M.A. **"Pattern generating procedure for the cutting stock problem"**, IntJ Production Economics 74:293–301, (2001).

[7] Vahrenkamp R. **"Random search in the one-dimensional cutting stock problem"**, European Journal of Operational Research 95:191–200, (1996).

[8] Vanderbeck F. **"Exact algorithm for minimizing the number of setups in theone-dimensional cutting stock problem"**, Operations Research 48:915–926, (2000).

[9] O'Rourke J., **"Computational Geometry in C second edition"**, ISBN 0 521 64976 5, http://www.cambridge.org

[10] de Berg M., van Kreveld M., Overmars M., Schwarzkopf O.C. **"Computational Geometry: Algorithms and Applications, Second Edition"**, ISBN 3-540-65620-0

[11] Graham, R.L. (1972). **"An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set"**, Information Processing Letters. 1 (4): 132–133. doi:10.1016/0020-0190(72)90045-2 , https://en.wikipedia.org/wiki/Graham_scan

[12] Evtimov G., Fidanova S. **"Subtraction of Two 2D Polygons with Some Matching Vertices"**, pp. 80-87, Numerical Methods and Applications, 9th International Conference NM&A'18, Borovets, Bulgaria, 2018, ISBN 978-3-030-10691-1, doi:10.1007/978-3-030-10692-8_9

[13] Evtimov G., Fidanova S. **"Heuristic Algorithm for 2D Cutting Stock Problem"**, pp.350-357, Large-Scale Scientific Computing, 11th International Conference, LSSC 2017, Sozopol, Bulgaria, Springer, Vol. 793, ISBN 978-3-319-73440-8, doi:10.1007/978-3-319-73441-5_37

[14] Evtimov G., Fidanova S. **"Analyses and Boolean Operation of 2D Polygons"**, pp. 107-118, Advanced Computing in Industrial Mathematics, BGSIAM 2017, Springer, Vol. 793, ISBN 978-3-319-97276-3, doi:10.1007/978-3-319-97277-0_9

[15] Evtimov G. Fidanova S. **"2D Optimal Cutting Problem"**, pp. 33-40, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*, Springer, Vol. 728, ISBN 978-3-319-65529-1, doi:10.1007/978-3-319-65530-7_4

[16] Evtimov G., Fidanova S. **"Ant Colony optimization algorithm for 1D Cutting Stock Problem"**, pp. 24, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*

[17] Fidanova S. **"ACO Algorithm with Eddditional Reinforcement, From Ant Colonies to Artificial Ants"**, *Lecture Notes in Computer Science, N2463, Spinger, Germany, 2002,292-293*

[18] Fidanova S., Marinov P., Alba E. **"ACO for Optimal Sensor Layout, In Proc. of Int. Conf. on Evolutionary Computing, Valencia, Spain, Joaquim Filipe and Janus Kacprzyk eds."**, *SciTePress-science and Tchnology Publications Portugal, ISBN 978-989-8425-31-7, 2010, 5-9*

[19] Fidanova S., Marinov P., Alba E. **"Wireless Sensor Network layout, In Monte Carlo Methods and aplications"**, *K. Sabelfeld, I. Dimov eds., Chapter 9, De Gruyter Pub, Berlin, germany, 2012, 39-46*

[20] Fidanova S., Shindarov M., Marinov P. **"Mono-objective algorithm for Wireless Sensor Network layout, In Proc of OMKO-NET Int."**, *Conference, Southampton, UK, 2012a, 57-63*

[21] Fidanova S., Shindarov M., Marinov P. **"Optimal Sensor Layou Using Multi-objective Metaheuristic, In Proc of OMKO-NET Int."**, *Of Int. Conference of information systems and Grid Technologies, Sofia, Bulgaria, 2011, 114-122*

[22] Fidanova S., Shindarov M., Marinov P. **"Multi-Objective ant algorithm for Wireless Sensor Network Positioning."**, *Proceedings of Bulgarian academy of Science, Vol 66(3), 2013, 353-360.*

[23] Fidanova S., Shindarov M., Marinov P. **"Wireless Sensor Positioning ACO Algorithm."**, *Studies of Computational intelligence, J.Kacprzyk and K. atanassov eds., Spinger, Germany*

[24] Gonalves J.F., **"A hybrid genetic algorithm-heuristic for a two-dimensional or-thogonal packing problem"** Eur J Oper Res, Vol 183, No 3, 2007, 1212-1229.

[25] V. Bodurov, D. Dimov, G. Evtimov, I. Georgiev, S. Harizanov, G. Nikolov, V. Pirinski, **The 2D/3D Best-Fit Problem, 113th European Study Group with Industry (ESGI'113)**, Problems & Final Reports, pp. 62-73, 2015. ISBN:978-619-7223-12-5

[26] A. Avdzhieva, T. Balabanov, G. Evtimov, D. Kirova, H. Kostadinov, T. Tsachev, S. Zhelezova, N. Zlateva, **"Optimal Cutting Problem, 113th European Study Group with Industry (ESGI'113)"**, Problems & Final Reports, pp. 49-61, 2015. ISBN:978-619-7223-12-5

[27] Боровска П., **"Синтез и анализ на паралелни алгоритми"**, ISBN:978-954-438-764-4

[28] https://www.sit.ac.jp/user/konishi/JPN/Tech_inform/Pdf/GeometricalMoment.pdf

[29] Р. Даскалов, Е. Даскалова, **"Висша математика, част I, Аналитична геометрия"**, Габрово, 2012

[30] Antonio, Franklin **"Chapter IV.6: Faster Line Segment Intersection"**, In Kirk, David (ed.). Graphics Gems III. Academic Press, Inc. pp. 199–202. ISBN 0-12-059756-X, (1992).

[31] "Weisstein, Eric W. **"Line-Line Intersection. From MathWorld"**, A Wolfram Web Resource. Retrieved 2008-01-10.

[32] Shimrat, M., **"Algorithm 112: Position of point relative to polygon"**, 1962, Communications of the ACM Volume 5 Issue 8, Aug. 1962

[33] Eric Haines, **"Point in Polygon Strategies" in Graphics Gems IV"**, (1994) http://geomalgorithms.com/a03-_inclusion.html

[34] https://bg.wikipedia.org/wiki/РЁРхСЪРөРхРўСЃРчСҒСЪРчСЊР,,Рч_РөРњРуР«СЃРчСЪРёРч

[35] Glover F., **"Future paths for integer programming and links to artficial intelligence"**, Computers and Operations Research,13,533-549 (1986).

[36] Glover F. and Laguna M., **"Tabu Search"**, Kluwer, Boston, (1997).

[37] Balaban, I. J. **"An optimal algorithm for finding segments intersections"**, Proc. 11th ACM Symp. Computational Geometry, pp. 211–219, doi:10.1145/220279.220302, (1995).

[38] Bartuschka, U.; Mehlhorn, K.; Näher, S. **"A robust and efficient implementation of a sweep line algorithm for the straight line segment intersection problem"**, in Italiano, G. F.; Orlando, S. (eds.), Proc. Worksh. Algorithm Engineering, (1997).

[39] Bentley, J. L.; Ottmann, T. A. **"Algorithms for reporting and counting geometric intersections"**, IEEE Transactions on Computers, C-28 (9): 643–647, doi:10.1109/TC.1979.1675432, (1979).

[40] **"Point in Polygon, One More Time..."**, Archived 2018-05-24 at the Wayback Machine, Ray Tracing News, vol. 3 no. 4, October 1, 1990.

[41] http://www.theswamp.org/index.php?topic=1891.0

[42] Boissonat, J.-D.; Preparata, F. P. **"Robust plane sweep for intersecting segments"**, (PDF), SIAM Journal on Computing, 29 (5): 1401–1421, doi:10.1137/S0097539797329373, (2000).

[43] Brown, K.Q. **"Comments on "Algorithms for Reporting and Counting Geometric Intersections"**, IEEE Transactions on Computers, C-30 (2): 147, doi:10.1109/tc.1981.6312179, (1981).

[44] Chazelle, Bernard; Edelsbrunner, Herbert (1992), **"An optimal algorithm for intersecting line segments in the plane"**, Journal of the ACM, 39 (1): 1–54, doi:10.1145/147508.147511

[45] Alvarez-Valdes R, Parajon A, Tamarit J. M, **"A computational study of heuristicalgorithms for two-dimensional cutting stock problems"**, 4th metaheuristics inter-national conference (MIC2001), 2001, 16-20.

[46] Chen, E.Y.; Chan, T.M. **"A space-efficient algorithm for segment intersection"**, Proc. 15th Canadian Conference on Computational Geometry (PDF), (2003).

[47] Clarkson, K.L. **"Applications of random sampling in computational geometry"**, II", Proc. 4th ACM Symp. Computational Geometry, pp. 1–11, doi:10.1145/73393.73394, (1988).

[48] Eppstein, D.; Goodrich, M.; Strash, D. **"Linear-time algorithms for geometric graphs with sublinearly many crossings"**, Proc. 20th ACM-SIAM Symp. Discrete Algorithms (SODA 2009), pp. 150–159, (2009).

[49] Cintra G., Miyazawa F., Wakabayashi Y., Xavier E., **"Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation"**, European Journal of Operational Research, European Journal of Operational Research, Vol. 191, 2008,61-85.

[50] Mulmuley, K. **"A fast planar partition algorithm"**, I", Proc. 29th IEEE Symp. Foundations of Computer Science (FOCS 1988), pp. 580–589, doi:10.1109/SFCS.1988.2197, (1988).

[51] Preparata, F. P.; Shamos, M. I. **"Section 7.2.3: Intersection of line segments"**, Computational Geometry: An Introduction, Springer-Verlag, pp. 278–287, (1985).

[52] Shamos, M. I.; Hoey, Dan **"Geometric intersection problems"**, 17th IEEE Conf. Foundations of Computer Science (FOCS 1976), pp. 208–215, doi:10.1109/SFCS.1976.16, (1976)

[53] Dorigo M., Manieezz M., mColorni A., **"The ant syste: Optimization by a colony of cooperating agents"**, IEEE Transactions on Systems, Man and Cybernetics B, Vol 26(1), 1996, pp 29-41

[54] Dorigo M., **"Optimization, Learning and Natural Algorithms"**, PhD thesis, Politecnico di Milano, Italie, 1992.

[55] Lodi, A., Martello S., Vigo D., **"Recent advances on two-dimensional bin packingproblems"** , Discrete Applied Mathematics, Vol. 123, 2002, 379-396.

[56] Falkenauer E., **"A hybrid grouping genetic algorithm for bin packing"**, Journal of Heuristics, Vol. 2, 1996, pp. 5-30

[57] Hinterding R., Khan L., **"Genetic algorithms for cutting stock problems: with and without countiguity"**, InX. Yao, editor, Progress in Evolutionary Computation, Berlin, Germany, Springer, 1995, pp. 166-186

[58] Jaromi M.H., Tavakkoli-Moghaddam, R., Makui, A. Shamsi A., **"Solving an one-dimensional cutting stock problem by simulated and tabu search"**, J. of Industrial Engineering International, Vol. 8 (1), Spinger, 2012, pp. 24

[59] Reeves C., **"Hybrid genetic algorithms for bin-packing and related problems"**, Annals of Opera Research 63, 1996, pp.371 396

[60] Vink M., **Solving combinatorial problems using evolutionary algorithms**, 1997 http://citeseer.nj.nec.com/vink97solving.html

[61] Parmar K., Prajapati H., Dabhi V., **"Cutting stock problem: A survey of evolutionary computing based solution"**, In Proc. of Green Computing Communication and Electrical Engineering, 2014, doi:10.1109/ICGCCEE.2014.6921411.

[62] Dusberger, F., Raidl, G.R., **"A variable neighborhood search using very large neighborhood structures for the 3-staged 2-dimensional cutting stock problem"**, In: Blesa, M.J., Blum, C., Voß, S. (eds.) HM 2014. LNCS, vol. 8457, pp. 85–99. Springer, Cham (2014). doi:10.1007/978-3-319-07644-7_7

[63] Dusberger, F., Raidl, G.R., **"Solving the 3-staged 2-dimensional cutting stock problem by dynamic programming and variable neighborhood search."**, Electron. Notes Discret. Math. 47, 133–140 (2015)MathSciNetCrossRefzbMATH Google Scholar

[64] Lin Z., Li Y., **"An Efficient Algorithm for Intersection, Union and Difference between Two Polygons"**, *In proc. of Computer Network and Multimedia Technology, Wuhan, China, 2009, 1-4*

[65] Gonçalves J.F., **"A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem"**, *European Journal of Operational Research*, Vol 183, No 3, 2007, 1212-1229.

[66] Alvarez-Valdes R., Parreno F., Tamarit J.M., **"A Tabu Search algorithm for two dimensional non-guillotine cutting problems"**, European Journal of Operational Research , Vol.183(3), 2007, 1167-1182.

[67] Alvarez-Valdes R., Parajon, A., Tamarit, J.M. **"A computational study of heuristic algorithms for two-dimensional cutting stock problems"**, In: 4th Metaheuristics International Conference (MIC 2001), pp. 16–20 (2001)

[68] Lodi A., Martello S., Vigo D., **"Recent advances on two-dimensional bin packing problems"**, Discrete Applied Mathematics, Vol. 123, 2002, 379-396.

[69] Delorme M., M. Iori, S. Martello, **"Bin packing and Cutting Stock Problems: Mathematical models and exact algorithms"**, European Journal of Operational Research 2016, 255, 1–20, doi:10.1016/j.ejor.2016.04.030

[70] Wäscher, G.; Haußner, H.; Schumann, H. **"An Improved Typology of Cutting and Packing Problems"**, European Journal of Operational Research Volume 183, Issue 3, 1109-1130

[71] M.P. Johnson, C. Rennick and E. Zak **"Skiving addition to the cutting stock problem in the paper industry"**, SIAM Review, 472-483, (1997).

[72] Raffensperger, J. F. **"The generalized assortment and best cutting stock length problems"**, International Transactions in Operational Research. 17: 35–49, doi:10.1111/j.1475-3995.2009.00724.x., (2010).

[73] Kantorovich, V.L., **"Mathematical methods of organizing and planning production"**, Leningrad State University. 1939

[74] Kantorovich L.V. and Zalgaller V.A. **"Calculation of Rational Cutting of Stock"**, Lenizdat, Leningrad, (1951)

[75] Gilmore P.C., R.E. Gomory **"A linear programming approach to the cutting-stock problem"**, Operations Research 9: 849-859, (1961).

[76] Gilmore P.C., R.E. Gomory **"A linear programming approach to the cutting-stock problem - Part II"**, Operations Research 11: 863-888, (1963).

[77] Gradiar M., Kljaji M., Resinovi, G., Jesenko J. **"A sequential heuristic procedure for one-dimensional cutting"**, European Journal of Operational Research. 114. 3, 557-568, (1999).

[78] Gradisar M., Resinovic G., Kljajic, M. , **"A hybrid approach for optimization of one-dimensional cutting"**, European Journal of Operational Research. 119. 3, 719-728, (1999).

[79] Gradisar M., Resinovic G., Kljajic M. (2002) **"Evaluation of algorithms for one-dimensional cutting"**, Computers and Operations Research 29:1207–1220

[80] Dyckhoff H., **"A typology of cutting and packing problems"**, European Journal of Operational Research. 44, 145-159,(1990).

[81] Vance P., Barnhart, C., Johnson, E.L., Nemhauser, G.L. **"Solving binary cutting stock problems by column generation and branch-and-bound"**, Computational Optimization and Applications. 3. 111-130,(1994).

[82] Vance P., **"Branch-and-price algorithms for the one-dimensional cutting stock problem"**, Computational Optimization and Applications. 9, 211-228,(1998).

[83] Goulimis C. **"Optimal solutions for the cutting stock problem"**, European Journal of Operational Research 44: 197-208, (1990)

[84] de Carvalho V. **"Exact solution of cutting stock problems using column generation and branch-and-bound"**, International Transactions in Operational Research 5: 35–44, (1998)

[85] Berberler M.E., Nuriyev U.G., **"A New Heuristic Algorithm for the One-Dimensional Cutting Stock Problem"**. Appl. Compuy. Math. 9.1, 19-30,(2010).

[86] Diegel A., E. Montocchio, E. Walters, S. van Schalkwyk and S. Naidoo (1996), **"Setup minimizing conditions in the trim loss problem"**, European Journal of Operational Research 95:631-640

[87] McDiarmid C., **"Pattern Minimisation in Cutting Stock Problems"**, Discrete Applied Mathematics, 121-130, (1999)

[88] Garey M.R., Johnson, D.S., **"Computers and Intractability: A Guide to the Theory of NP-Completeness"**, WH Freemann, New York (1979).

[89] Kantorovich L.V., **"Mathematical methods of organizing and planning production"**, Management Science 6.4, 366-422,(1960).

[90] Jahromi, M.H., Tavakkoli-Moghaddam, R., Makui, A. Shamsi A. **"Solving an one-dimensional cutting stock problem by simulated annealing and tabu search"**, Journal of Industrial Engineering International, Vol. 8(1), Springer, 2012, paper 24.

[91] Jahromi, M.H., Tavakkoli-Moghaddam, R., Makui, A., Shamsi, A., **"Solving an one dimensionalcutting stock problem by simulated annealing and tabu search"**, Journal of Industrial Engineering International. (2012).

[92] **"Constantine Goulimis. Counterexamples in the CSP"**, arXiv:2004.01937

[93] Maria Garcia de la Banda, P. J. Stuckey., **"Dynamic Programming to Minimize the Maximum Number of Open Stacks"**, INFORMS Journal on Computing, Vol. 19, No. 4, Fall 2007, 607-617.

[94] Chvátal, V. **"Linear Programming"**, W.H. Freeman. ISBN 978-0-7167-1587-0, (1983).

[95] Cherri, A.C., Arenales, M.N., Yanasse, H.H., **"The one-dimensional cutting stock problems with usable leftover: a heuristic approach"**, European Journal of Operational Research. 196.3, 897-908,(2009).

[96] Cherri, A.C., Arenales, M.N., Yanasse, H.H., **"The usable leftover one-dimensional cutting stock problem-a priority-in-use heuristic"**, Intl. Trans. in Op. Res. 20, 189-199,(2013).

[97] Valerio de Carvalho, J.M., **"Exact solution of bin-packing problems using column generation and branch-and-bound"**, Annals of Operation Research. 86, 629-659, (1999).

[98] Hatem Ben Amor, J.M. Valério de Carvalho, **"Cutting Stock Problems in Column Generation"**, edited by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, Springer, 2005, XVI, ISBN 0-387-25485-4.

[99] Waescher, G., Gau, T., **"Heuristics for the Integer one-dimensional Cutting Stock Problem"**, A Computational Study. OR Spektrum18. 3, 131-144,(1996).

[100] Vanderbeck, F., **"Computational study of a column generation algorithm for binpacking and cutting stock problems"**, Mathematical Programming A. 86, 565-594, (1999).

[101] Vanderbeck, F., **"On DantzigWolfe decomposition in integer programming andways to perform branching in a branch-and-price algorithm"**, Operations Research. 48, 111-128,(2000).

[102] Mobasher, A., Ekici A., **"Solution approaches for the cutting stock problem with setup cost"**, Computers and Operations Research. 40, 225-235,(2013).

[103] Johnston, R.E., Sadinlija, E., **"A new model for complete solutions toone-dimensional stock problems"**, European Journal of Operational Research. 153, 176-183,(2004).

[104] Johnston, R.E. **"Rounding algorithm for cutting stock problems"**, Journal of Asian-Pacific Operations Research societies 3:166–171, (1986).

[105] Umetani, S., Yagiura, M., Ibaraki, T., **"One-dimensional cutting stock problem to minimize the number of different patterns"**, European Journal of Operational Research. 146, 388-402,(2003).

[106] S. Umetani, M. Yagiura, and T. Ibaraki **"One dimensional cutting stock problem to minimize the number of different patterns"**, European Journal of Operational Research 146, 388–402, (2003)

[107] Scheithauer, G., Terno, J., Muller, A., Belov, G., **"Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm"**, Journal of the Operational Research Society. 52, 1390-1401,(2001).

[108] Belov, G., Scheithauer, G., **"A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting"**, European Journal of Operational Research. 171, 85-106,(2006).

[109] Belov, G., Scheithauer, G., **"A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths"**, European Journal of OperationalResearch. 141, 274-294,(2002).

[110] Mukhacheva, E.A., Belov, G., Kartak, V., Mukhacheva, A. S., **"One-dimensional cutting stock problem: Numerical experiments with the sequential value correction method and a modified branch-and-bound method"**, Pesquisa Operacional. 2000.2, 153-168,(2001).

[111] Belov G., Scheithauer G. **"Setup and open stacks minimization in one-dimensional stock cutting"**, INFORMS Journal of Computing 19(1):27–35, (2007).

[112] Belov G., Scheithauer G. **"The number of setups (different patterns) in one-dimensional stock cutting"**, Technical Report, Desden University, (2003).

[113] Belov G., Scheithauer G. **"A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting"**, European Journalof Operational Research 171:85–106, (2006).

[114] Dikili A.C., Sarz E., PekN. A., **"A successive elimination method for one-dimensional stock cutting problems in ship production"**, Ocean engineering. 34.13, 1841-1849, (2007).

[115] Reinertsen H., Vossen Thomas W.M., **"The one-dimensional cutting stock problem with due dates"**, European Journal of Operational Research. 201.3, 701-711, (2010).

БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

АВТОРЕФЕРАТ НА ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен "доктор" по научна специалност "Информатика и компютърни науки"

МЕТАЕВРИСТИЧНИ МЕТОДИ ЗА РЕШАВАНЕ НА ЗАДАЧИ ЗА РАЗКРОЯВАНЕ

Георги Евтимов Евтимов

Ръководител: Проф. Стефка Фиданова

Научно жури:

Проф. Красимир Атанасов
Проф. Олимпия Роева
Доц. Десислава Иванова
Проф. Иван Димов
Доц. Леонид Кирилов

Институт по информационни и комуникационни технологии

Секция „Паралелни алгоритми"

Глава 1

Увод

Необходимостта от оптимизация на човешкия труд води до масовото разпространение на изчислителни електронни устройства, които в повечето случаи заместват човешкото присъствие. От там започва и екстремното развитие на информационните технологии (ИТ) като средство за управление на изчислителните устройства. Значителното поевтиняване на електрониката допълнително развива този процес. Все повече машини за производство се управляват от компютри, без значение дали се ползуват от предприятие, което е малко, средно или голямо. Естествено продължение на този процес е развитието на мрежа за обединяване на електронните устройства наречена Интернет.

Съвременна тенденция е всички услуги да се пренасочват от реалността във виртуалната реалност. Високо технологичните индустрии в производствения сектор масово изграждат системи за планиране и управление на ресурсите на производството. Информационните системи позволяват оптимизиране на ресурсите на всички нива на организационната йерархия. Тази оптимизация в повечето случаи оказва положително влияние както върху конкурентноспособността на фирмата, така и способства за по-гъвкаво и по-бързо намиране на нови пазари. Използване на информационните системи от фирмите дава възможност на клиентите им да решават по-добре, по-бързо и по-ефективно специфични проблеми в дадена предметна област. Особено ефективен е този подход в сферата на тежката индустрия и строителството.

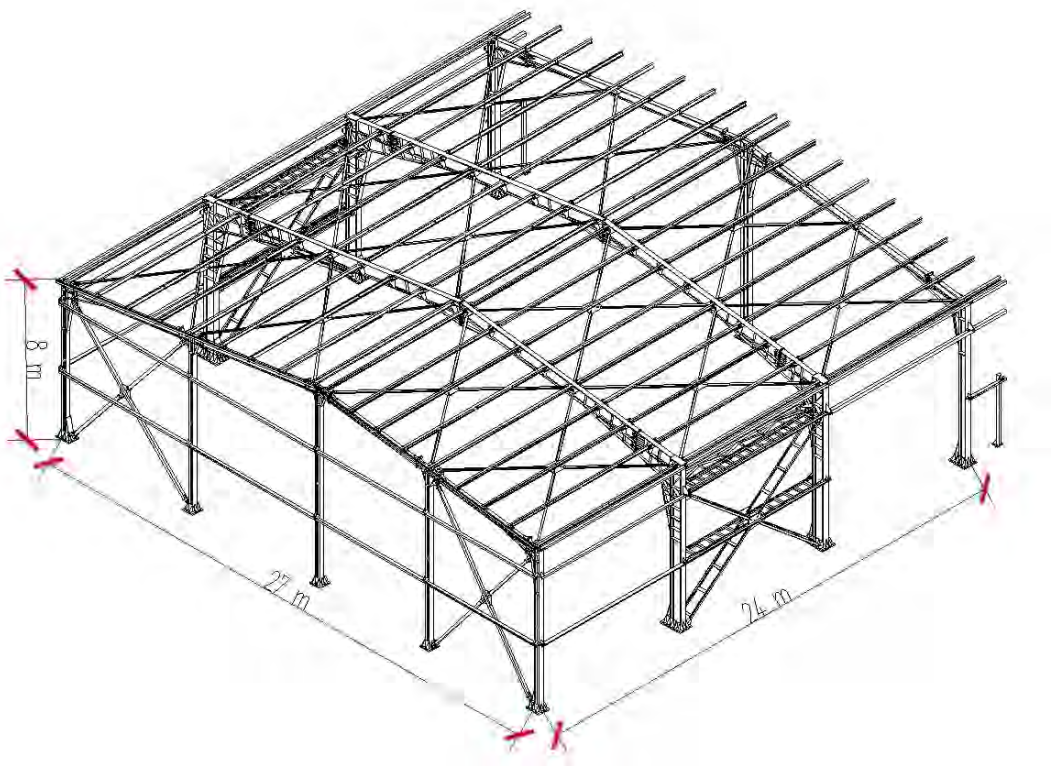
Широкото прилагане на ИТ в Европа и България създаде една динамична и високо конкурентна среда, в която фирма без внедряване на ИТ решения е често обречена на фалит. Необходимостта от съкращаване на производствени разходи е жизнено важна за оцеляването на предприятието. От друга страна изискванията на клиентите нараства, което допълнително подхранва необходимостта от бързи решения за оптимизация на материалните и човешки ресурси, което се постига ефективно с използуване на софтуер. Тази ситуация на пазара открива нови и почти неограничени възможности за прилагането на приложен софтуер при решаването на най-различни задачи. Всеки софтуер, в който е приложено знание, може да се разглежда като стратегически източник на иновация. Друга гъвкавост на ИТ е възможността технологията да бъде разработена от трета страна, непринадлежаща на фирмата, но да се прилага ефективно от много други фирми.

Накратко, фокусът на научно-изследователската работа в съчетание с технологичните иновации е една от най-динамичните области на развитите на съвременната индустрия. Настоящата дисертацията представлява усилие в тази посока. Мотивацията и обектът на приложение на тази работа произлиза от строителната индустрия и по специално от производството на стоманени конструкции.

Една от най-важните и широко практикувани дейности там е следната: за нуж-

дите на строителен обект е необходимо да се разкроят определено число детайли (много често достигащ хиляди) с различни размери, форми, дебелини, а в някой случай и от различен материал. Материалът е доставен под формата на метални листове или остатъци от листове, от които преди това са изрязвани детайли. Пример на такава стоманена конструкция е показан на Фигура 1.1. Необходимо е да се разкроят нужните детайли като се минимизира разхода на материал.

Тази постановка е частен случай на общата математическа задача за оптимален разкрой. Практическата задача за оптимален разкрой се заключава в следната лесна формулировка: зададен е определен материал (например, в текстилната индустрия това е плат, в строителните конструкции това са метални листове) и голямо количество често различни, детайли. Необходимо е да се разкроят нуждите детайли като се минимизира разходът на материал. Практически, това означава да се минимизира материалът, който остава след разкроя и не може да се оползотвори освен да се предаде за вторична преработка или рециклиране. Този неизползуван материал се нарича често фира. Тази задача математически е формулирана преди повече от 80 години във връзка с индустриализацията на шивашкото производство. Подобен тип задачи възникват в много други индустриални производства и използуването на оптимизация на решенията може да доведе до съществени икономии на материал.



Фигура 1.1: Стоманена конструкция

Предложените в тази работа научни методи ще бъдат полезни и най-подходящи за нуждите на стартиращи компании, включително и такива от сферата на софтуера. Методите, които са представени тук, са базирани на математика и логика и не се използват външни библиотеки, могат да се напишат, на които и да е език за програмиране и могат да помогнат развитието на която и да е компания.

1.1 Актуалност на темата

Темата за оптимален разкрой придобива още по-голяма актуалност през последните две десетилетия, намирайки разнообразни приложения в много индустриални производства. Особено важна е задачата сега, когато пазарът е отворен и фирмите трябва да се състезават с голям брой конкуренти с модерно оборудване и ниска цена на труда.

От друга страна масовото потребление на стоки води до необходимостта от оптимизиране на производството на тези стоки. Това включва както минимизиране на потреблението на енергия и суровини, така и намаляване на използуване на човешки труд. Много остър е проблемът при тежките индустрии, особено когато е необходимо изработването на голям брой елементи от скъпо струващ материал. Тези две особености са налице, например, в строителната индустрия. Затова проблемът за оптималния разкрой там стои на дневен ред с особена актуалност.

Обзор на съществуващите методи и тяхната реализация в приложен софтуер е направено по-долу в Секция 1.2. Тук ще отбележим само, че на пазара на софтуер за нуждите на оптималния разкрой има основно два вида подходи на реализация. В единия подход планките се апроксимират до правоъгълници, които се разполагат оптимално върху листа стомана, а при втория те се разполагат с точната им геометрия. При методите, които прилагат апроксимиране до правоъгълник недостатъкът е, че при фигури различни от правоъгълник отпадъкът е голям. В различните индустрии под голям може да се разбират най-различни числа. В настоящия труд 25-50% ще разбираме за голям отпадък, 10-25% за средно голям отпадък, 0-10% малък отпадък. При работа с триъгълни фигури образувани от прави по трите си страни отпадъка е 50%. Голям отпадък. Този подход има ограничено приложение, но се използува доста масово и дава добри резултати в стъкларската и хартиената промишленост. И в двата вида софтуер въвеждането на обектите за разкрой става ръчно. Полигоните се въвеждат по координати на върховете или по сегменти на страните. Това отнема много време и е възможно да се допуснат неточности и/или грешки при въвеждането на данните.

През последните три десетилетия за проектиране на строителни обекти се използуват масово CAD системи. В представената дисертация проблемът за оптимален разкрий на строителни елементи (или планки) е решен при предположението, че полигоните (планките) се генерират и предоставят на строителя от CAD система. След това се прави предварителна обработка на данните и накрая планките се разкрояват с точната им геометрия.

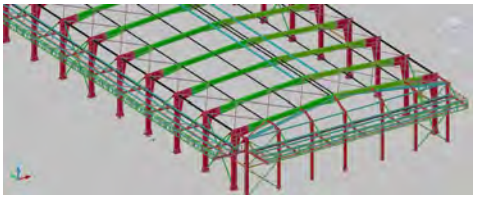
Заготовка на планки за нуждите на стоманените конструкции, представляват определен клас подзадачи за оптимален разкрой, който се характеризира с редица особености, които водят както до опростяване така и до усложняване на задачата за разкроя. Най-важните особености са:

1. често планките са със сложни форми, чиито граници са произволни несамопресичащи се полигони (в редки случаи, направата на елементи с елипсовидни контури се свежда към горния случай чрез апроксимация с полигони с достатъчна за практиката точност);

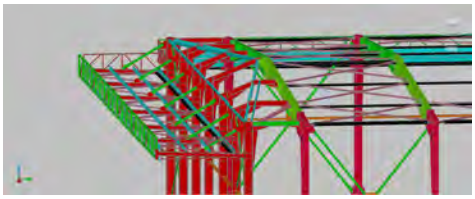
2. много често планката няма "лице" и "гръб", което позволява огледално търсене на местоположението ѝ в процеса на разкроя; тази особеност може да доведе до икономия на материал, но увеличава сложността на проблема.

3. в набора от планки, които трябва да бъдат произведени, много често има значително разнообразие на размерите, площите и формата.

В работата ударението е поставено върху методи за решаване на задачата за разкрой на планки в $2D$. Двумерната задача за разкрой е по-трудна от едномерната, особено когато изрязваните фигури не са изпъкнали и са с неправилна форма. И двете задачи са NP-пълни комбинаторни оптимизационни задачи [12], [13].



Фигура 1.2: Стоманена конструкция 1.



Фигура 1.3: Стоманена конструкция 2.

Като илюстрация ще се представят някои примери за случая на стоманени конструкции. В стоманено хале показано на фигура 1.2 може да има около 2000 - 3000 стоманени планки за разкрой. Те са с различна дебелина, в практиката често се налага да работим с 6 различни дебелини. Следователно от метален лист с дадена дебелина трябва да бъдат разкроени около 500 броя планки с доста сложна форма. В разглеждания случай ротация и огледален образ на планката са препоръчителни при оптимизацията на разкроя. Нещо повече, има планки, при които съотношението дължина към широчина (това се разбира като планката се постави в правоъгълник с минимални размери) е повече от 100.

1.2 Обзор на основните резултати в областта

Проблемът с оптималния разкрой (CSP) възниква в много индустриални области [61]. Повечето автори решават 2D разкроя, както апроксимират входящите полигони (фигурите) до правоъгълници. Тези решения също са приложими в много индустрии. Например производството на хартия и стъкло [24], зареждане на контейнери, дизайн с много мащабна интеграция (VLSI) и различни задачи за планиране [55].

По-сложната версия на проблема е, когато входящите полигони (фигурите) не са апроксимирани до правоъгълници. Този проблем възниква при строителните конструкции в производството на стоманени изделия, производството на дрехи, производството на обувки и т.н. В [24] основната тема е двуизмерен ортогонален проблем с опаковането, при който фиксирана група от малки правоъгълници трябва да се монтира в голям правоъгълник и неизползваната площ от големи правоъгълници да се сведе до минимум. Алгоритъмът комбинира метод на заместване с генетичен алгоритъм. В [45] е разработена *Greedy* (алчна) процедура на произволно (рандомизирано) адаптивно търсене. В това проучване има голям първичен запас, който трябва да бъде нарязан на по-малки парчета, за да се увеличи максимално стойността на парчетата. Cintra [49] предлага точен алгоритъм, базиран на динамично програмиране, който е подходящ за малки проблеми, тъй като проблемът е NP-труден. Dusberger и Raidl [62], [63] предлагат два мета-евристични алгоритъма, базирани на търсене на променливи квартали. Гореспоменатите работи решават опростения проблем с правоъгълни елементи. В индустрията на строителните конструкции планките са полигони, които могат да имат неправилна форма и могат да бъдат изпъкнали или вдлъбнати, но не и само пресичащи се. Подобно разнообразие от форми значително увеличава трудността на проблема. Като планките или входящите полигони могат да прилагат и огледално, тъй като стоманените листа са хомогенни от двете си страни. Също сложността на задачата се увеличава и от това, че триъгълна планка може да бъде описана с повече от три точки.

1.3 Цели и задачи на дисертацията

Основните цели поставени пред докторанта са от научно-приложен и приложен характер. Те могат да бъдат обобщени по следния начин.
Цели на дисертацията:

1. Оптимален разкрой на линейни елементи при минимален отпадък;
2. Оптимален разкрой на двумерни елементи с неправилна форма при минимален отпадък.

За постигане на тези цели бяха формулирани следните задачи:

Задача 1. Разработване на алгоритъм за решаване на задачата за едномертен (линеен) разкрой;

Задача 2. Разработване на алгоритъм за решаване на задачата за разкрой на двумерни елементи;

Задача 3. Да се направи програмна реализация на разработените алгоритми и да бъдат проведени сравнвния на реални строителни обекти със съществуващи в практиката методи за разкрой.

База на разработката е CAD среда за получаване на графична информация от даден строителен обект. След оптимизацията се генерира информация в термините на същата CAD среда. За целта е разработен числен алгоритъм за разкрояване (разполагане) на произволни не самопресичащи се полигони (наричани планки и генерирани от CAD система) от зададен от потребителя полигон (стоманен лист). Фокус на дисертацията са равнинни елементи (листов материал) $2D$ фигури (наричани тук планки). Геометрически това означава определен брой фигури в равнината да се подреди в площ със зададен от потребителя затворен контур. Това е доста обща математическа задача, която може да се приложи в най-различни индустрии. Алгоритъмът позволява и допълнителни настройки и различни принципи в оптимизацията при подреждане на фигурите. Тези задачи са базирани изцяло на примери от практиката, а входните данни са от реално проектирани и изпълнени строителни обекти.

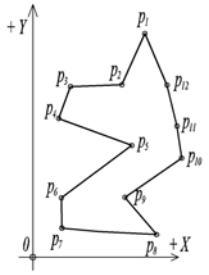
1.4 Подход на изследването

Тази дисертация се занимава с решаването на две оптимизационни задачи: (1) разкрой на линейни профили (стоманени пръти, T- и П-образни профили, и т.н.) или $1D$ разкрой и (2) разкрой на двумерни (плоски) планки от стоманени листове.

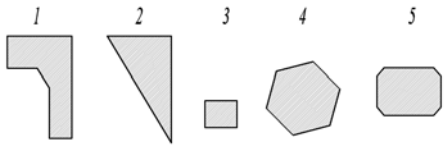
Първата задача е едномерен (линеен), $1D$ *разкрой*. За $1D$ оптимизацията не се въвеждат някакви специални дефиниции, тъй като се работи с един параметър, дължината на елемента. Задачата за минимален отпадък се свежда до намиране на минимален брой използвани профили. Въпреки, че е по-лесна от двумерната задача, тя също е NP сложна. Подходът използува метода на мравките.

Втората задача е $2D$ *разкрой*. Данните включват даден входящ списък от n на брой планки (наречени входящи полигони), които трябва да се подредят възможно най-плътно в даден полигон, наречем основен. При търсенето на едно възможно разполагане на входящите полигони може да се приложи ротация и огледален образ. След като е избрано местоположение на входящия полигон е необходимо

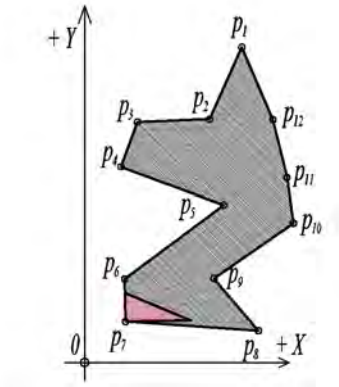
да се приложи алгоритъма за "изваждане" ("изрязване") на два полигона. Това се прави с цел за следващия входящ полигон да се търси местоположение в остатъка от основния полигон. След това тази (изрязана) планка се премахва от списъка с входящи планки. Това се повтаря докато се изчерпят всички планки от входящия списък. След това цялостното решение (съвкупността от планки) се оценява от метаевристиката. Виж точка 4.2.



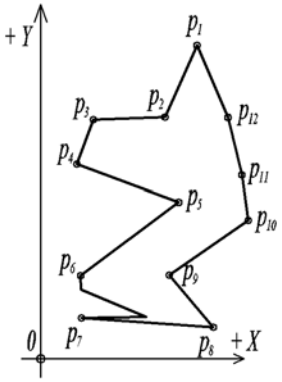
Фигура 1.4: Първоначален основен полигон (за запълване).



Фигура 1.5: Примерни входящи полигони (размерите са съществено увеличени)



Фигура 1.6: Основен полигон преди изрязването.



Фигура 1.7: Основен полигон след изрязването.

В процеса на изучаване на проблема е използвана широк кръг от литература. Създадените нови методи и алгоритми са публикувани в статии на автора, [12, 13, 15, 14]. Създадени са три метода. Първия е за оценка на входящите полигони (множества). Спрямо ъглите им и дължините им. Този метод дава оценката от 0. до 1. за най-голямата вероятност даден полигон да се постави в полигона на запълване при даден връх. Втория метод дава оценка за най-голяма допирна дължина (площ) между два полигона. Третия метод е хибридната метаевристика. Дава оценка на всички допустими решения за даден връх. Оценката за всяко едно решение е между 0. и 1. Избира се това с най-голяма оценка. При положени, че има повече от едно решение с максимална оценка се избира произволно едно от тях. Комбинацията е от трите метода ни дава възможност да не търсим пълно изчерпване на възможните комбинации за поставяне на входящите полигони Π_i в полигона на запълване Δ. Разработените са два нови алгоритъма, които са подобрение на два съществуващи алгоритъма. Единия е *Ray* метода [40]. Добавката е, че преди да се приложи *Ray* метода се проверява дали дадената точка е в *box* на полигона, ако е така тогава се проверява за целия полигон. За дефиниция на *box*

на полигона виж точка 2.2. *box* на полигона ще се използва, ако броя на върховете на дадения полигон е по-голям от 4. Другият метод е "Bentley-Ottman" [39]. Добавката е, че не се обхождат всички сегменти, а алгоритъма спира при първото пресичане на двата сегмента.

Като краен резултат от изследването на проблема е разработен софтуер, който успешно решава двете задачи. Направено е сравнение на получените резултати с резултатите от използуване на комерсиален софтуер. Предимствата на създадената програма пред тествания комерсиален софтуер са дадени в 4.6 и в 5.

Глава 2

Изчислителна геометрия.

В тази секция се използуват дефиниции от изчислителната геометрия - точка, линеен сегмент и полигон в двумерното пространство. Всички точки ще бъдат представени като списък от наредени числа (координати), [9], в двумерния случаи това са двойки числа $P = (x, y)$.

2.1 Основни дефиниции

Дефиниция на списък

Списъка представлява строго подредени елементи. Всеки един елемент може да бъде число, стринг или друг списък. Примери:

1. $list(X, Y)$ - точка зададена с нейните декартови координати;
2. $list(pt_0, pt_1, \ldots, pt_i)$ - списък от точки, където pt_i е списък, представящ точка с индекс i;
3. $list\ (e_0, e_1, \ldots, e_{n-1})$ - списък от сегменти, където e_i е линеен сегмент с индекс i, виж по-долу.

Дефиниция на точка.

Точките в d-мерното пространство са представени като нареден списък от d числа, наречени координати, [9]. Тъй като разглеждаме задачата в равнина, то в тази работа точка pt_i е дефинирана като $pt_i = list\ (x_i, y_i)$, където координатите x_i и y_i са реални числа. При работа с реални числа използуващи компютърна аритметика се поставя въпросът за грешката при закръгление. Грешката от закръгление на реалните числа е важна и обширна област в математиката. Тук са приети следните правила:

1. Работим с точност четири знака след десетичната запетая .0001;
2. Приемаме отклонение $fuzz$, което е реално число, по-голямо от 0.

Тези правила са продиктувани от необходимостта да работим с данни произведени от CAD системи. В областта на проектиране на обекти изпълнени със стоманени конструкции, размерите на конструкциите се дават в милиметри, така че всички числени данни (координати на точки, сегменти и т.н.) са в милиметри. За нуждите на строителната индустрия (стоманени конструкции), разликата в дължините на страните на планките от 0.5мм на дава детайла ги прави неразличими. Затова ги приемаме, че са еднакви.

Приемаме, че две точки съвпадат $P_i \equiv P_q$, ако:

$$(x_i - fuzz) \leq x_q \leq (x_i + fuzz) \wedge (y_i - fuzz) \leq y_q \leq (y_i + fuzz) \tag{2.1}$$

или

$$max\{|x_i - x_q|, |y_i - y_q|\} \leq fuzz \tag{2.2}$$

Дефиниция на линеен сегмент.

Ще използуваме два вида сегменти:
(1) CAD линеен сегмент $CADe_i = list(pt_0, pt_1, \ldots, pt_i)$ се задава със списък от точки, които лежат на една права; такива сегменти се получават от работата на CAD системата, която генерира всички входни данни използувани в тази работа.
(2) Линеен сегмент $e_i = (pt_i, pt_{i+1})$ е затворено множество от точки лежащи на една права между две точки pt_i и pt_{i+1}, наречени крайни точки, [9]. Точките в списъка e_i са подредени. Като първата е начална, а втората крайна. В нашата работа линейните сегменти се получават след премахване на вътрешните точки на CAD сегмента.

Дефиницията на полигон.

Полигон $\Pi = list\ (e_0, e_1, \ldots, e_{n-1})$ е затворена област от равнината оградена от n линейни сегменти образуващи затворена крива, [9]. Обръщаме внимание, че тук използваме линеен сегмент, а не CAD линеен сегмент. Нека $pt_0, pt_1, \ldots, pt_n$ са n точки от дадена равнина, такива, че $pt_0 = pt_n$. Точките $pt_0, pt_1, \ldots, pt_n$ образуват цикличен списък. Докато pt_0 е последвана от pt_1, то pt_{n-1} е последвана от $pt_0 = pt_n$. Полигонът се описва също и от неговите върхове, крайните точки на сегментите му, така че еквивалентно, $\Pi = list\ (pt_0, pt_1, ..., pt_n)$. Казваме, че два сегмента са съседни когато имат само една обща крайна точка.

Линейни сегменти образуват полигон тогава и само тогава, когато:

1. Пресечната точка между всяка двойка съседни сегменти в цикличния списък, е : $e_i \cap e_{i+1} = pt_{i+1}$, за всички $i = 0, ..., n-1$;

2. Несъседни сегменти не се пресичат.

Точките pt_i ще ги наричаме *върхове* на полигона, а сегмент от полигона ще наричаме линеен сегмент. Нека отбележим, че полигон с n върхове има n сегмента.

Ротация на точка

Да разгледаме две различни точки $pt_A = list(x_a, y_a)$ и $pt_{base} = list(x_{base}, y_{base})$ в координатна система XOY. Желаем да завъртим точката pt_A около базовата точка pt_{base} на даден ъгъл β. Ако ъгълът β е положително число тогава въртенето се разбира обратно на часовниковата стрелка $anti - CW$, в противен случай въртенето е по часовниковата стрелка CW. След завъртането ще получим нова точка $pt_B = list(x_b, y_b)$ в координатната система XOY. За да направим нужните пресмятания и получим изчислителните формули за координатите на точката получена след ротацията, ще въведем нова координатна система $X'O'Y'$, чието координатно начало съвпада с pt_{base}. Осите на новата координатна система $X'O'Y'$ се транслират успоредно на осите на координатната система XOY. Спрямо новата координатна система получаваме координатите на точка pt_A, $x'_a = (x_a - x_{base})$ и $y'_a = (y_a - y_{base})$ или $pt_A = list(x'_a, y'_a)$ в новата координатна система $X'O'Y'$. Радиуса на завъртане R се намира по формулата:

$$R = \sqrt{{x'_a}^2 + {y'_a}^2} \tag{2.3}$$

а ъгълът на завъртане α се получава от формулата:

$$\alpha = arccos\frac{x'_a}{R} \tag{2.4}$$

Тогава координатите на точка pt_B в координатната система XOY са:

$$x_b = x_{base} + \frac{R}{\cos(\alpha+\beta)} \tag{2.5}$$

$$y_b = y_{base} + \frac{R}{\sin(\alpha+\beta)} \tag{2.6}$$

Перпендикуляр от точка към дадена права.

Правата е зададена с две точки $A = list(x_a, y_a)$ $B = list(x_b, y_b)$ и тестовата точка е $T = list(x_t, y_t)$. Искаме да намерим точка $C = list(x_c, y_c)$ от правата $list(A, B)$ такава, че векторът определен от точките T и C е перпендикулярен на правата. Преди да започнем търсенето на точката C, трябва да проверим дали точките $A = list(x_A, y_A)$, $B = list(x_B, y_B)$ и $T = list(x_t, y_T)$ не лежат на една права. Това става чрез намиране на лицето на триъгълника $F = list(A, B, T)$. Резултатът, които ще получим от този алгоритъм, е ориентираното лице на триъгълника $list(A, B, T)$. Нас ни интересува само дали лицето F е нула или не. Ако лицето $F = 0$, то точките лежат на една права и не е нужно да търсим перпендикулярния вектор $\vec{TC}$ към правата $list(A, B)$. Ако лицето $F \neq 0$. то алгоритъма протича в следните стъпки, съгласно [41]:

1. Ако $x_A = x_B$, тогава правата е вертикална и търсената точка е $pt_C = list(x_A, y_T)$.;
2. Ако $y_A = y_B$, тогава правата е вертикална и търсената точка е $pt_C = list(x_T, y_A)$;
3. Ако не са изпълнени горните условия, тогава търсим наклона m на правата $list(A, B)$:

$$m = \frac{y_B - y_A}{x_B - x_A} \tag{2.7}$$

$$x_C = \frac{(\frac{x_T}{m} + y_T + m.x_A - y_A)}{m + \frac{1}{m}} \tag{2.8}$$

$$y_C = y_A + m(x_C - x_A) \tag{2.9}$$

Или координатите на желаната точка $C = list(x_C, y_C)$.

Огледален образ на полигон.

Под огледален образ на полигон ще разбираме огледален образ по всички страни от полигона, които не са успоредни една на друга.

Полигона $Mirror1 = list(mpt_0, mpt_1, mpt_2, mpt_3)$ е получен от защрихования полигон $\Pi_i = list(pt_0, pt_1, pt_2, pt_3)$. За намирането на огледалния образ на полигона Π_i се използва следната последователност:

1. Взимаме първата pt_0 и втора pt_1 точка от полигона Π_i.
2. Образуваме правата $L1 - L1$. Правата $L1 - L1$ се образува по две точки. Първата точка е $pt_{L1} = (polar(pt_0; (angle = pt_1, pt_0); 10e10))$. Втората точка е $pt_{L2} = (polar(pt_1; (angle = pt_0, pt_1)); 10e10)$. Намираме полярните координати на точките pt_{L1} и pt_{L2} - базова точка, ъгъл и дължина. В случая дължината е избрана достатъчно голяма така, че да бъде допустима за CAD системата.

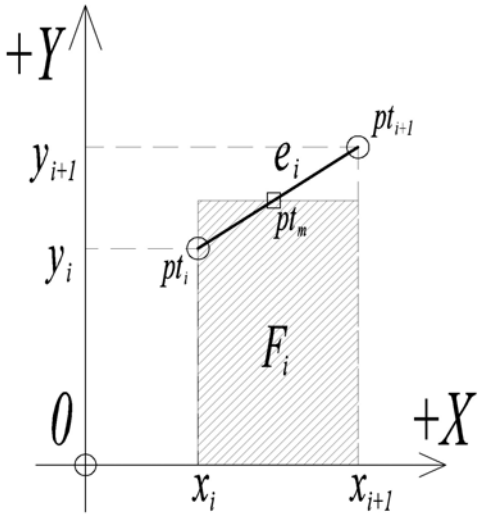
3. За всеки връх на полигона Π_i намираме петата на перпендикуляра към правата $L1 - L1$ и я означаваме с pt_{Perp_i}.

   Огледалната точка се получава: $mpt_i = (polar(pt_{Perp_i}; (angle = pt_i, pt_{Perp_i}); distance(pt_i, pt_{Perp_i}))$. За намиране на петата на перпендикуляра върху правата $L1 - L1$ виж Подсекция 2.1.

Правата $L1 - L1$ ще бъде колинеарна със сегмента $list(pt_0, pt_1)$. Тогава и разстоянието $distance(pt_0, pt_{Perp_i})$ ще бъде нула и точките pt_0 и mpt_i ще съвпадат. За намиране на правата $L1 - L1$ може да се използва всяка една двойка последователни върхове $list = (pt_i, pt_{i+1})$ на полигона Π_i.

Намиране на посока на полигон $(clock - wise, CW$ **или** $anti - CW)$**.**

Тук ще обсъдим начини за определяне за посоката, (по часовата стрелка, CW, или обратна на часовата стрелка, $anti - CW$), на границата на полигон $\Pi = list\ (pt_0,\ pt_1,...,\ pt_n)$. От основите известни в литературата методи за ориентация на полигон, тук ще представим един от най-бързите методи за пресмятане на посоката на обхождане на върховете по границата на полигона Π [28].



Фигура 2.1: Лице на трапец

Нека $e_i \in \Pi$ е произволен сегмент и нека средната му точка P_m има кординати:

$$P_m = \left(\frac{x_i + x_{i+1}}{2}, \frac{y_i + y_{i+1}}{2}\right). \tag{2.10}$$

Площта на фигурата между сегмент $e_i \in \Pi$ и координатна ос X е:

$$F_i = \frac{(x_{i+1} - x_i)(y_{i+1} + y_i)}{2} \tag{2.11}$$

Да отбележим, че лицето F_i може да бъде положително, отрицателно или нула. Знакът на лицето зависи от подредбата на точките в списъка определящи Π, тъй като подредбата може да бъде $list\ (pt_0,\ pt_1,...,\ pt_{n-1})$ или $list\ (pt_{n-1},\ pt_0,...,\ pt_{n-1})$. Това лице ще наречем ориентирано лице. Тази процедура се прилага за всички e_i

сегменти. За да се пести процесорно време няма смисъл всяко лице да се дели на 2. Затова сборът от ориентираните лица можем да запишем както следва:

$$2F = \sum_{i=0}^{n-1} (x_{i+1} - x_i)(y_{i+1} + y_i) \tag{2.12}$$

Ако координатите на точките на върхове $list(pt_0, pt_1, \ldots, pt_5)$ удовлетворяват условията $x_5 > x_4 > x_3 > x_2 > x_1 > x_0$, то съответните площи са положителни и $F > 0$. Ако коррдинатите на точките на върхове $list(pt_0, pt_1, \ldots, pt_5)$ удовлетворяват условията $x_5 > x_6 > x_7 > x_8 > x_9$, тогава $F < 0$

Ъгъл между два вектора. Вътрешен ъгъл на полигон.

С цел да получим по-добра характеристика за даден полигон ще са ни нужни вътрешните му ъгли. Първо ще намерим ъгъла между два вектора $\vec{a} = list(T, pt_i)$ и $\vec{b} = list(T, pt_{i+1})$, дефинирани в XY координатна система.

Първо определяме дължината на векторите $\vec{a}$ и $\vec{b}$, а след това и тяхното скаларно произведение. За да намерим дължината на вектора $\vec{a}$, ще транслираме точка A_i до нулата, $T_i = (list0, 0)$. Тогава векторите $\vec{a}$ и $\vec{b}$ ще имат координати съответно (x_a, y_a) и (x_b, y_b).

$$\|\vec{a}\| = \sqrt{x_a^2 + y_a^2} \tag{2.13}$$

$$\|\vec{b}\| = \sqrt{x_b^2 + y_b^2} \tag{2.14}$$

скаларното произведение е:

$$\vec{a} \cdot \vec{b} = x_a x_b + y_a y_b \tag{2.15}$$

и така получаваме

$$\cos\alpha = \frac{\vec{a} \cdot \vec{b}}{\|a\|\|b\|} \tag{2.16}$$

За намирането на вътрешните ъгли на даден полигон ще трябва да проверим дали полигонът в околност на даден връх е изпъкнал или не изпъкнал. За да може да проверим това ние трябва да намерим ориентацията на полигона. За да намерим вътрешните ъгли посоката на полигона трябва да бъде обратно на часовниковата стрелка $anti - CW$. След това започваме да проверяваме всеки три точки от полигона $list(pt_i, pt_{i+1}, pt_{i+2})$. Намираме вътрешния ъгъл α, който е при връх pt_{i+1}. Правим проверка за ориентацията на трите точки. Ако са ориентирани по часовниковата стрелка CW, то от 2π трябва да извадим ъгъла α. Ако ориентацията на върховете $list(pt_i, pt_{i+1}, pt_{i+2})$ е обратно на часовниковата стрелка $(anti - CW)$, то записваме ъгъла α без корекция.

Algorithm 1 InsidePolyAngle

/*Функция за намиране на вътрешни ъгли на полигон*/
procedure INSIDEPOLYANGLE($pt_0, pt_1, \ldots, pt_n$)

if isClockWise $pt_0, pt_1, \ldots, pt_n$ **then return** ptList = reverse $pt_0, pt_1, \ldots, pt_n$
i = 0
L = length of $pt_0, pt_1, \ldots, pt_n$
Repeat L
for pt_i, pt_{i+1}, pt_{i+2} **do** α = getInsideAngle pt_i, pt_{i+1}, pt_{i+2}
if isClockWise pt_i, pt_{i+1}, pt_{i+2} **then return** $\alpha = (2\pi$ - $\alpha)$
else α
i = i + 1
End Repeat

По този начин ще може към всеки един полигон да се запише информация за вътрешните ъгли и дължините на страните му.

Добавяне на точки в линеен сегмент

Добавянето на точки в линеен сегмент е необходимо за намирането на повече възможни валидни разполагания на полигоните Π_i, $i = 1, \ldots,$ в полигона Δ. Виж точка 4.4. Сега да вземем един определен полигон $\Pi = list(e_1, e_2, \ldots e_n)$ от входящия списък от полигони. Всеки сегмент e_i от полигона $\Pi = list(e_1, e_2, \ldots e_n)$ се разделя на три подсегмента. За всеки подсегмент се добавят подробни точки. Принципа за поставяне на подробни точки.

Получаването на подробни точки за сегмент $e_i = list(pt_i, pt_{i+1})$ става чрез полярни координати (базова точка, ъгъл и дължина). Дефинираме фунцията *polar* която връща точка по зададена базова точка, ъгъл и дължина. Базовата точка е pt_i. Ъгъла на сегмента e_i спрямо абцисната ос OX е $angle(pt_i, pt_{i+1})$. Намирането на дължината на всеки сегмент е както следва:

1. Разделяме сегмента e_i в съотношение $L_1 = 0.1(distance = pt_i, pt_{i+1})$.

2. L_1 се разделя на съответния брой сегменти, които искаме да постигнем. Може да се използват 3 или 5 деления.

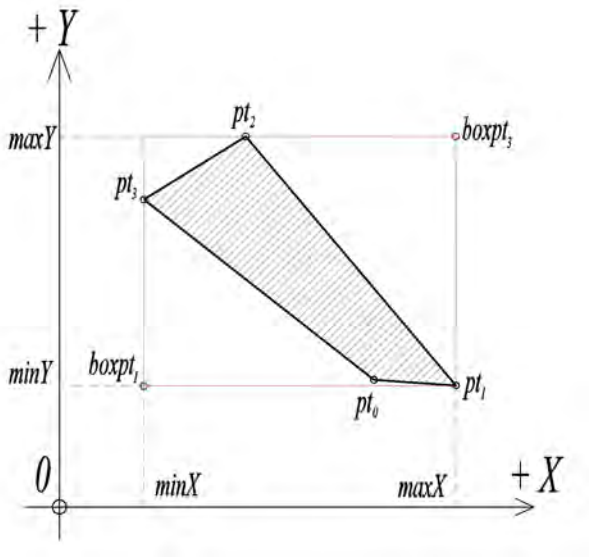
3. Първата подробна точка ще бъде $pt_{L1,i} = (polar(pt_0, ang_e 1, \frac{L_1}{5})$.

Изчисляването на другите подробни точки за сегмента e_i става в същата логика както за точка $pt_{L1,i}$. Добавянето на тези подробни точки по границата на полигона в някои случаи повишава качеството на валидните решения. При тестване, ако поставям полигона Π_i във връх pt_0, тогава няма да получим валидно решение (разполагане на планката). Но, ако полигона Π_i се постави в някоя от подробните точки валидно решение ще има.

Премахване на излишни точки от линеен сегмент Проверката се изпълнява преди да започне самото подреждане на полигоните. При представянето на полигоните като списък от върхове $list(pt_0, pt_1, \ldots, pt_n)$ е възможно триъгълник да се опише с повече от три точки. Така, че броя на върховете (точките) в един списък не определя вида на фигурата.

2.2 Намиране на *box* на полигон.

Под *box* на полигон $\Pi_i = list(pt_0, pt_1 \dots pt_n)$ ще разбираме правоъгълната обвивка на дадения полигон Π_i. Виж фигура 2.2.



Фигура 2.2: *box* на полигон

За да намерим *box* на полигона Π_i трябва да преминем през списъка от точки $\Pi_i = list(pt_0, pt_1 \dots pt_n)$ и за всяка една точка да вземем координатите ѝ по $X.pt$ и по $Y.pt$. След това да сортираме двата нови списъка по низходящ ред, $listX = (x_0, x_1 \dots x_n)$ и $listY = (y_0, y_1 \dots y_n)$. Първата стойност от $listX$ ще ни даде $maxX$ последната $minX$. Същото правим и за списъка $listY$. Така получаваме координатите на точките $boxpt_1 = list(minX, minY)$ и $boxpt_3 = list(maxX, maxY)$. Точка $boxpt_1$ е винаги най-долу, най-вляво. Точка $boxpt_3$ е винаги най-горе, най-вдясно.

2.3 Пресичане на две прави

Намирането на пресечна точка между две прави е "скъпо" струваща операция от гледна точка на процесорно време и би следвало да се използва в "краен" случай. Затова тук ще разгледаме две функции. Първата е намиране на координатите на пресечната точка pt_x на два дадени сегмента e_1 и e_2, а втората е проверка дали два дадени сегмента e_1 и e_2 се пресичат, без да се търси самата пресечна точка. Първата функция ще връща като стойност пресечната точка $list = (x_i, y_i)$, а втората – $true$ или $false$.

Намиране на координати на пресечна точка pt_x.

Съгласно [31] и [30] пресечната точка $ptX = (X, Y)$ на два дадени сегмента

$e_1 = (x_1, y_1)$ и $e_2 = (x_2, y_2)$ има координати:

$$X = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}, Y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}. \tag{2.17}$$

Като пресметнем детерминантите в (2.17), получаваме следните изрази за X и Y:

$$X = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(x_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{2.18}$$

$$Y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(x_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}. \tag{2.19}$$

2.4 Точка в полигон

Ще разгледаме следната задача в равнината XOY. За дадена произволна точка $T = list(x, y)$ (наречена тествана точка) и полигон $\Pi = list(pt_0, pt_1, \ldots, pt_n)$ да се определи дали точката е вътре в полигона или не е. Върховете на полигона са зададени с $pt_i = list(x_i, y_i)$.

Тук ще бъдат разгледани два метода за решавана на тази задача - Ray crossing method (пресичащ лъч), [32], и Balanced sum of angles (Балансирана сума на ъглите). В разработения към дисертационния труд софтуер се използва следният подход преди да се приложи метода на пресичащия лъч за всички сегменти. Прави се проверка чрез метода на пресичащия се лъч дали дадената точка T е в box на полигона Π_i. За повече намиране на box на полигон виж 2.2.

и ако е в box тогава се прилага Ray метода за всички сегменти на полигона Π. Този подход изисква обхождане на целия списък с точки $(pt_0, pt_1, ..., pt_n)$ и сравнение за намиране на минимум и максимум на всяка координата. Тази проверка се прави бързо, тъй като се свежда до сравняване на две числа. Този подход е оправдан тъй като броя на сегментите в един полигон много бързо нараства. В зависимост от сложността на входните полигони и разрешените ъгли на ротация, полигона за който проверяваме Π може да достигне 300-400 сегмента. Като тази проверка се повтаря n на брой пъти. Сложността на алгоритъма е $\mathcal{O}(n^2)$, но ако се приложи методът изложен в точка 2.5 то сложността може да се намали до $\mathcal{O}(n)$.

Ray crossing method (метод на пресичащия лъч)
Проверка дали дадена точка T е в даден полигон Π. Да припомним, че областта оградена от полигона е затворена, т.е. включваща и границата. За целта построяваме хоризонтална полуправа с начало дадената точка T и крайна точка T_∞, който ще наречем пресичащ лъч $Ray = list(T, T_\infty)$. Под точка в безкрайността ще разбираме най-голямото число което може да се генерира от дадената CAD система. В повечето случаи 10^{20} е достатъчно като приближение до безкрайност. Идеята на метода на пресичащия лъч се базира на броя пресечни точки с полигона Π. Ако боря на пресечните точки е четен, то точка T е извън полигона, иначе тя е вътре в полигона.

В частните случаи, когато пресечните точки на лъча Ray със сегментите на полигона П съвпадат с даден връх полигона П се получават неточни резултати. За решаване на проблема със съвпадение на пречната тока с даден връх от полигона се въвежда критерий за "Възходящ" и "Низходящ" сегмент.

Balanced sum of angles (Балансирана сума на ъглите)

Задачата се свежда до намирането на ориентирания вътрешен ъгъл между векторите. $\vec{a} = list(T, pt_i)$ и $\vec{b} = list(T, pt_{i+1})$.

Сега ще дадем критерии кога дадена точка T се намира вътре или вън от даден полигон $\Pi = list(pt_0, \ldots, pt_n)$. Построяваме векторите $\vec{a}_i$ от точката T до pt_i, $i = 0, \ldots, n-1$ и намираме ориентираните ъгли α_i между $\vec{a}_i$ и $\vec{a}_{i+1}$, $i = 0, \ldots, n-1$. След това пресмятаме

$$Sum_\alpha = \sum_{i=1}^{n} arccos\alpha_i \tag{2.20}$$

Ако точката T е вътрешна за полигона П, то сборът на всички вътрешни ъгли Sum_α е равен на $\pm 2\pi$. С едно изчисление на всички ъгли (обхождане) този метод ни дава две важни характеристики за дадения полигон П:

1. Ако $Sum_\alpha = 2\pi$ полигона П е ориентиран CW;
2. Ако $Sum_\alpha = -2\pi$ полигона П е ориентиран $anti - CW$.

Това е много надежден метод, но не достатъчно бърз съгласно [14]. Също така от критично значение е и грешката, която се акумулира при събирането на ъглите. Обикновено стойностите на ъглите са малки при голям брой на сегменти на полигона и тогава може да се натрупа достатъчно голяма грешка, което от своя страна ще затрудни преценката дали точката T е в полигона П.

2.5 Пресичане на два полигона

Функцията за намиране на сечението (пресичане) на два полигона A и B е една от основните операции с множества и често използувана в алгоритмите за разкрой. За съжаление реализацията на тази операция изисква много процесорно време. Класическият подход се заключава в проверка дали всеки сегмент от границата на B пресича сегмент от границата на A. Този метод е лесен за реализация, но възможно най-бавен, има сложност $\mathcal{O}(n^2)$, [10], стр. 21. Този подход не се използува в разработения от автора софтуер.

По-бързи методи за проверка дали два полигона се пресичат използуват намирането на точките на пресичане на границите им с техните координати или проверка за логическо пресичане. В повечето случаи ще използуваме логическото пресичане на двата полигона, тогава не е необходимо да знаем броя или координатите на пресечните точки, достатъчно е да знаем, че един от двата полигона имат поне един връх, който е във вътрешността на другия полигон. Това се използува също при търсенето на възможно положение на даден полигон Π_i спрямо основния полигон Δ. В секция 4.4 при 2D разкроя ще дадем повече информация.

Една от възможните проверки за пресичане е като започнат да се пускат лъчи от всеки връх от полигон B. И ако даден връх от полигона B е в полигона A, то двата полигона A и B се пресичат. В този подход не е задължително да се проверяват всички точки от полигон B дали са вътрешни за полигон A. Методът е

надежден, но сложността му е почти $\mathcal{O}(n^2)$, тъй като всеки един лъч реално е сегмент с начален връх B_i. Този подход е сравнително по-бърз от класическия метод.

Един от бързите алгоритми е този на "Bentley–Ottmann algorithm", [39], който се заключава във въвеждане на вертикално или хоризонтално "сканираща" линия преминаваща през всички сегменти. При достигане началото на даден сегмент отчитаме "събитие" и при достигане края на сегмента имаме също "събитие". За разработването на алгоритъма се използва вертикална сканираща линия. Съгласно [43] сложността на този алгоритъм за всички K пресичания между N сегмента е $\mathcal{O}((N+K)\log N)$.

2.6 Редуциране на върховете на полигон.

В процеса на търсене на валидно решение по контура на полигона се "вмъкват" подробни точки, които повишават съществено вероятността да се намери валидно решение. В алгоритъма разработен от автора [13] на всеки сегмент се поставят по 15 подробни точки. Три интервала по пет точки. Това е областта на търсене на решения - тези подробни точки. За всяка една от тях входящия полигон се транслира и завърта до намиране на валидно решение. Валидно решение е това което входящия полигона е в полигона на запълване. Без пресичане на двата полигона.

Редукцията на полигона ще повиши значително скоростта на алгоритъма и съответно ще спести изчислително време. За редуцирането на полигона трябва да определим неговата посока на построение. Да приемем, че посоката на построение на полигона Δ е CW. Отваряме нов празен списък и започваме обхождане на полигона Δ със всеки три точки $list(pt_{i-1}, pt_i, pt_{i+1})$, $i = 1 \dots n$. Изчисляваме посоката на въртене на точките $list(pt_{i-1}, pt_i, pt_{i+1})$. Ако посоката им съвпада с посоката на въртене на полигона Δ, правим проверка дали $(getPolyArea(pt_{i-1}, pt_i, pt_{i+1}) < minArea$ и ако това е така то не записваме точка pt_i в празния списък, иначе записваме pt_i в празния списък. Ако не съвпада посоката на въртене на $list(pt_{i-1}, pt_i, pt_{i+1})$ с Δ то записваме pt_i в списъка.

Глава 3

Задача за $1D$ разкрой.

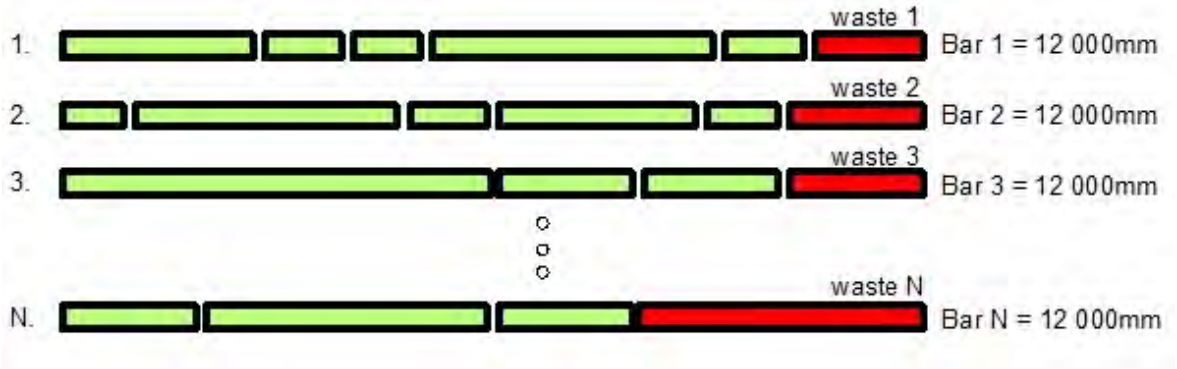
3.1 Формулировка на задачата.

Проблемът за оптимално разкрояване на елементи от зададена заготовка датира от началото на индустриалната революция, втората половина на 18-ти век и началото на 19-ти век. Характерно за този период от време е експоненциалното развитие на производствените сили. Индустриалната революция е свързана не само с началото на масовото използване на машини, но и с рязкото повишаване на производителността на труда. Високата производителност на труда е в пряка и правопропорционална зависимост от разхода на суровини. От тук се ражда необходимостта от оптимално използване на ресурсите в производството. Задачата за оптимален линеен разкрой засяга предимно промишлеността. Индустрията е сектор, който включва добива на полезни изкопаеми и преработката на суровини в междинни или крайни продукти. Условно може да разделим индустрията на два сектора добивна и преработваща. Във вторичния сектор се включва и строителството. Нашата задача произлиза от вторичния сектор - строителството. В строителството се използват основно няколко вида материали: Стоманобетон, стомана, дърво и други. В случая ще се фокусираме върху стоманените и дървените конструкции. Тези конструкции позволяват да се произведат в цех и да се монтират на строежа. Производството и на двата вида материали (стомана и дърво) позволяват разкрояване. Да вземем за пример стоманената конструкция показана на фигура 1.1.

В тази конструкция за прътовите елементи се използват сечения от най-различен вид. Двойно Т" профили, "L" профили, "C" профили.

Сеченията стоманените профили често достигат 100 кг/м. При цена на един килограм стомана от порядъка на 3,5 лв./кг. (към 2021 година) прави 350 лв. на линеен метър. И когато можем да направим икономия, която се повтори за всеки профил, то ползата от оптимален разкрой е очевидна.

Даден е списък от дължини на входящи профили $L = l_i$. Решението на задачата за $1D$ ще бъде сведено до намирането на решение за един Bar_i. Или това е линейното разполагане на част от профилите l_i в дадена дължина Bar_i. Следващите етапи са до изчерпването на всички профили от списъка L. Оптимизацията се заключава в такова разполагане на профилите, че да се получи възможно най-малък остатък за всяка една дадена дължина Bar_i, виж фигура 3.1.



Фигура 3.1: Дефиниране на линеен разкрой.

Където Bar_i, $i = 1 \ldots n$, виж фигура 3.1 е изобразена целевата дължина на запълване за намиране на едно разполагане на профилите. Проблема ще бъде решен с метода на мравките ACO. Метода на мравките е метаевристичен метод за решаване на изчислителни задачи [16]. Този алгоритъм е част от алгоритмите на Социалния интелект (SWARM модели на еволюцията на културата).

В дадения случай тя е 12 000 единици. Тя не може да бъде по-малка от най-малката дължина на входящите профили. За намирането на едно валидно решение взимаме това с най-малка стойност на остатъка $waste1$ им между всичките намерени решения. Трябва да обърнем внимание, че $waste1$ е съставен от два отпадъка. Единия отпадък е реалния - този, които остава като материал, записан в променлива $wasteReal$. Другия е технологичен, записан в променлива $wasteCut$. Това е ширината на режещия инструмент. Следователно общия отпадък е $waste_i = wasteReal + wasteCut$. След това профилите включени в избраното решение (зеления цвят на фигура 3.1) ги изключваме от входящия списък. Задачата се повтаря до като входящия списък стане празен. След това записваме в една променлива $sumWaste = (waste_1 + waste_2 + \cdots + wasten)$ фирата от всички профили и запазваме решението. След като получим следващото решение ги сравняваме по $sumWaste$. Избираме това с по-малкия $sumWaste$. Решенията се повтарят или докато получим фира под 5% или до дадено време за изчисление.

3.2 Намиране на цялостно решение за $1D$ разкрой.

Нека е даден следния входящ списък с профили за разкрояване. Входящият списък се получава директно от CAD системата. Един такъв списък е показан в таблица 3.1.

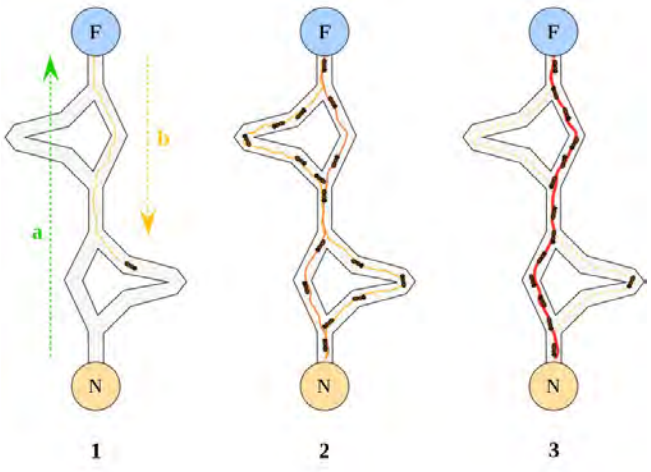
Входни данни за $1D$ разкрой:

Таблица 3.1: Входни данни за разкрой на профили.

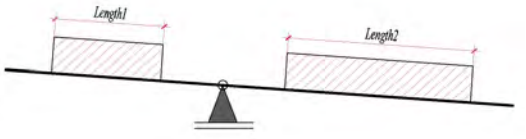
n	Сечение	Броя	Дължина
1	2	3	4
1	Profile 1	36	320
2	Profile 12	54	330
3	Profile 8	4	330
4	Profile 15	18	334
5	Profile 31	54	340
6	Profile 25	54	350
7	Profile 19	360	365

С n ще отбележим "свития"списък с 18 броя елемента. С N ще отбележим развития списък с 580 броя елементи.

Изпробвани са три метода на подреждане. Първият е комбинаторна оптимизация по метода на мравките ACO, [53] и [54]. Използвана е една мравка за намиране на решение за един Bar_i.



Фигура 3.2: Илюстрация на метода мравките.



Фигура 3.3: Оценка.

Метода на мравките (Ant Colony Optimization) е част от методите с популации. Методите с популации са част от Метаевристиката. Виж фигура 4.1. Най-общо това е вероятностен подход за решаване на най-различни изчислителни задачи. ACO метода решава задачи по зададени параметри. За първи път този метод е предложен от проф. Марко Дориго през 1992 в неговата дисертация.

От тогава до днес методът предложен от проф. Дориго е разширяване и модифицирана за да може да решава по-широк кръг от задачи. Идеята е взета от природата, затова тук ще използваме термини като "мравка" и "феромон". Феромона е химическо вещество което се отделя от мравките в процеса на изминаване на даден път. Този феромон служи за комуникация с други мравки. В началото мравките се движат на случаен принцип. При откриване на храна се завръщат в гнездото си. През целия път на отиване и връщане те отделят феромон. Този феромон се отлага по пътя им. Феромона привлича мравките. Колкото повече феромон има на даден път толкова повече вероятността мравките да се движат по него е по-голяма. По този начин количеството феромон се увеличава и пътя до източника на храна става по привлекателен за другите мравки. За решаването на всяка една задача могат да се използват различен брой мравки. Колкото помалко мравки се използват за намирането на глобалния оптимум токова по-малко изчислителни ресурси (процесорно време) ще са необходими. В настоящия случай е използвана една мравка. Мравката избира един произволен валиден елемент и го поставя във валидните решения. За следващо решение използва функция наречена вероятност на прехода. Тази функция е произведение на количеството феромон и евристична информация. Количеството феромон представлява опита на предходните итерации на мравките. Евристичната функция е информация представляваща предварително познание за задачата. Мравката избира този преход, които има на голямо произведение на феромона и евристичната информация. Това е най-голямата вероятност на вярно решение. Ако има повече от едно решение с еднаква вероятност, то се избира едно от тях на произволен принцип. След като всички мравки намерят решенията си, следва да се обнови феромона. Първо феромона се намалява за да се намали влиянието от предишни решения. След това се добавя нов феромон пропорционален на стойността на целевата функция. Като логиката е, че решенията с повече феромон са по-добри от тези с по-малко феромон и така те ще станат по-желани на следващата итерация. В конкретната задача феромонът се поставя на преходите.

Вероятност на прехода.

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum (\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}, \tag{3.1}$$

където:

- $\tau_{i,j}$ е количеството феромон, съответстващо на прехода от връх i във връх j;
- α е параметър за контрол на влиянието на $\tau_{i,j}$;
- $\eta_{i,j}$ е евристична информация. Комбинация от параметрите на целевата функция и ограниченията;
- β е параметър за контрол на влиянието на $\eta_{i,j}$.

Обновяване на феромона.

$$X = \begin{cases} L_k, & \text{ако мравка } K \text{ премине през реброто } i, j \\ 0, & \text{иначе} \end{cases} \tag{3.2}$$

Алгоритъм на метода на мравките ACO, съгласно [26].

Algorithm 2 ACOAlgorithm

/*Алгоритъм на мравките*/

procedure ACOAlgorithm
Begin
Поставяне на начален феромон
While докато критерия е истина **do**
Поставяне на всяка мравка в начален връх
Repeat
For each , приложи за всяка мравка
Избор на следващ връх
End Foreach , край на приложи за всяка мравка
Until , всяка мравка е построила решение
Обновяване на феромона
End While , край на while
End

От създаването на алгоритъма през 1992 год. от проф. Дориго до днес са разработени различни модификации на ACO алгоритъма. Едни от най-популярните варианти на алгоритъма за оптимизация по метода на мравките са изложени в [17], [18], [19], [20], [21], [22], [23], [53], [54],

В настоящата задача за $1D$ разкороя най-голям феромон получават тези профили, които дават най-малък остатък от Bar_i. Дължината на профила е неговата тежест, защото сечението е едно и също. То е константа. Виж фигура 3.3. Профил $Length2$ ще получи по-голям феромон (оценка) спрямо профил $Length1$. Или това са по-дългите профили ще имат по-висока оценка. Сбора на феромона за Bar_i е по-голям при по-дългите профили. В този случай метода на мравките проявява $Greedy$ характер.

Вторият метод е динамично оптимиране. При този подход не се прави комбинации между профилите, а записва сбора на дължите на профилите до даден индекс. При следващо търсене на сбора на дължините не се преминава през целия списък до дадената позиция, а се използва сбора им от предишни изчисления. Нека е даден списък с профили с техните дължини $L_p = list(p_0, p_1 \dots p_n)$. Сбора от дължините на първите 5 профила ще бъде:

1. $sum_1 = p_0 + p_1$;
2. $sum_2 = sum_1 + p_2$;
3. $sum_3 = sum_2 + p_3$;
4. $sum_4 = sum_3 + p_4$;
5. ...
6. $sum_i = sum_{i-1} + p_i$;

Третият метод е комбинация *Greedy* и комбинаторен метод с пълно изчерпване на n-елемента, k-ти клас. В *Greedy*. метода профилите се сортират по дължина. От най-голям към най-малък. Подреждането на профилите започва като първи се поставят най-големите дължини. Следващото поставяне е комбинаторния метод. Метода най-голямата дължина от сбора на дължините от всеки до три елемента в списъка L_p. Класа на комбинацията е ограничен до три елемента. Всяка комбинация се различава една от друга с поне 1 елемент.

$$C_n^k = \frac{V_n^k}{P_n^k} = \frac{n!}{k!(n-k)!} \tag{3.3}$$

Или при десет профила имаме следния брой комбинации с три елемента:

$$\frac{10!}{3!(10-3)!} = \frac{3628800}{6.5040} = 120 \tag{3.4}$$

Дължините на профилите с еднакво сечение не варират в големи граници. При направените сравнения между трите метода, за целите на стоманените конструкции хибридния подход $Greedy + n^3$ алгоритъма дава най-добри резултати по отношение на плътност и време. Комбинаторният метод работи със "свития"списък ще бележим с n елемента = 18 броя. Максималния брой итерации е $7^3 = 343$. Не е задължително след всяко поставяне на профил броя на елементите n да намалее с едно. При развития списък ще бележим с $N = 580$ броя. Максималният брой итерации би следвало да бъдат е $580^3 = 195112000$.

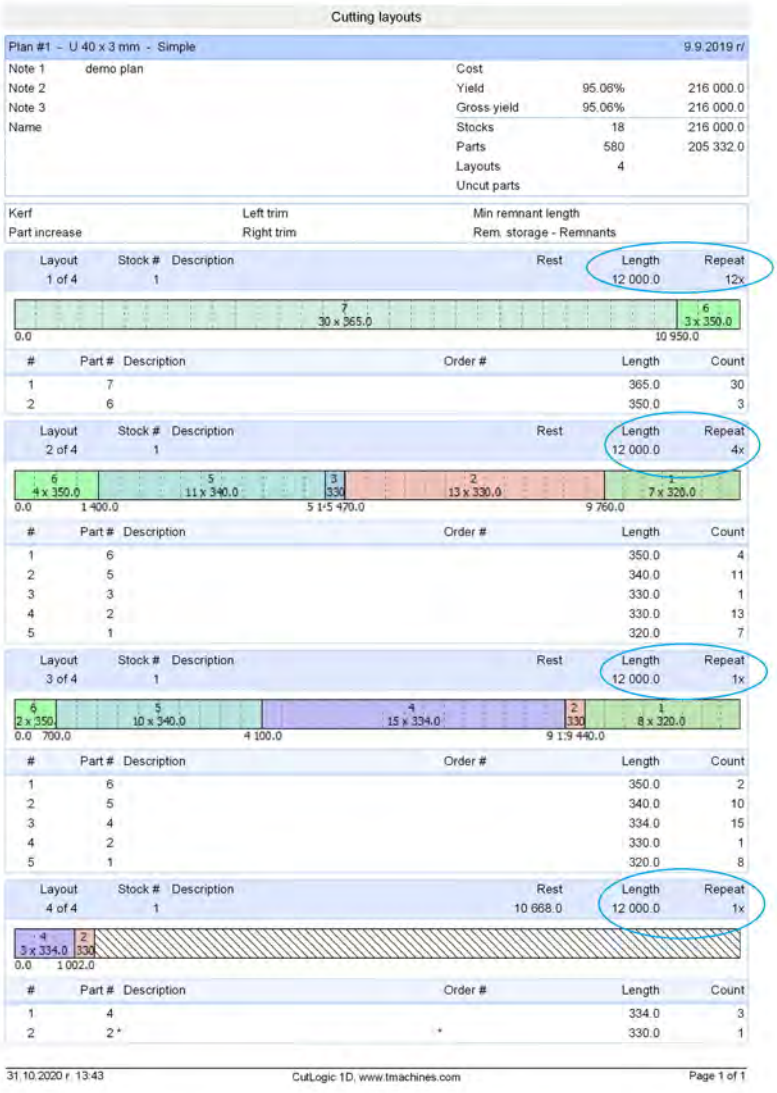
Дължини за разкрояване 12000 мм , ширина режещия нож 0. мм. Броят на дължините (профилите) за разкрояване е неограничен. Брой профили за разкрой - 580. Обща дължина на профилите за разкрояване е 205 332 мм. Следователно глобалният минимум ще бъде около $\frac{205332}{12000} = 17.11$, или до 18 броя профили по 12 000м.

3.3 Резултати при $1D$ разкрой. Примери.

Ще използваме следния комерсиалния продукт:

Фигура 3.4: Комерсиален продукт за $1D$ разкрой.

Резултати от изчисленията:



Фигура 3.5: Решение на $1D$ разкрой с комерсиален продукт.

В сините елипси на фигура 3.5 са посочени броя на необходимите профили. Те са 17 броя цели дължини по 12 000мм + 1002 мм от 18-та дължина.

Резултати в настоящия метод:

Таблица 3.2: Резултати с Мравки за $1D$ разкроя.

N	Дължини	Остатък	Профили
1	2	3	4
1	12000	120	profile 19 (360x33),
2	12000	120	profile 19 (360x33),
3	12000	120	profile 19 (360x33),
4	12000	120	profile 19 (360x33),
5	12000	120	profile 19 (360x33),
6	12000	120	profile 19 (360x33),
7	12000	120	profile 19 (360x33),
8	12000	120	profile 19 (360x33),
9	12000	120	profile 19 (360x33),
10	12000	120	profile 19 (360x33),
11	12000	120	profile 12 (330x36),
12	12000	120	profile 31 (330x36),
13	12000	100	profile 25 (350x34),
14	12000	120	profile 1 (320x36), profile 19 (360x1),
15	12000	160	profile 19 (360x29), profile 25 (350x4),
16	12000	120	profile 31 (330x18), profile 12 (330x18),
17	12000	130	profile 15 (330x18), profile 25 (350x16), profile 8 (330x1),
18	12000	11010	profile 8 (330x3),

Таблица 3.3: Резултати с $Greedy + n^3$ метод за $1D$ разкроя.

N	Дължини	Остатък	Профили
1	2	3	4
1	12000	150	profile 1 (320x36), profile 12 (330x1),
2	12000	120	profile 12 (330x36),
3	12000	120	profile 12 (330x17), profile 8 (330x4), profile 15 (330x15),
4	12000	120	profile 15 (330x3), profile 31 (330x33),
5	12000	170	profile 31 (330x21), profile 25 (350x14),
6	12000	100	profile 25 (350x34),
7	12000	180	profile 25 (350x6), profile 19 (360x27),
8	12000	120	profile 19 (360x33),
9	12000	120	profile 19 (360x33),
10	12000	120	profile 19 (360x33),
11	12000	120	profile 19 (360x33),
12	12000	120	profile 19 (360x33),
13	12000	120	profile 19 (360x33),
14	12000	120	profile 19 (360x33),
15	12000	120	profile 19 (360x33),
16	12000	120	profile 19 (360x33),
17	12000	120	profile 19 (360x33),
18	12000	10920	profile 19 (360x3),

Таблица 3.4: Сравнение на резултатите за $1D$ разкроя.

Софтуер	Броя използвани профили	Време[s]
1	2	3
Комерсиален продукт	17x12000 + 1002 мм	7
$Greedy + n^3$	17x12000 + 1080 мм	< 1
Мравки ACO	17x12000 + 990мм	1

Явно комерсиалния продукт ползва много сложна евристика или други методи за оптимизиране. Комерсиалният продукт е най-бавен спрямо другите два метода $Greedy + n^3$ и ACO. Както се вижда от сравнителната таблица 3.4, методът на мравките ACO дава най-добър резултат от гледна точка на най-малко отпадък, което е главната ни цел. Като време за пресмятане $Greedy + n^3$ е малко по-бърз от ACO за сметка на по-лошото решение. Комерсиалният продукт е бавен и дава по-лошо решение от метода на мравките ACO. Затова може да заключим, че предложеният от автора на дисертацията ACO алгоритъм превъзхожда другите два.

Глава 4

Задача за $2D$ разкрой.

4.1 Формулировка на задачата.

Индустрията поставя за решаване най-различни оптимизационни задачи. Тези задачи могат да се класифицират по много признаци зависещи от:

1. характера на решавания проблем;
2. структура на задачата;
3. броя на управляващите параметри;
4. характера на зависимостта на критерия и ограниченията на параметрите;
5. наличието на различни ограничения;
6. характера на търсения минимум;
7. броя на критериите;
8. и други.

При производството на стоманени конструкции се налага да се изрязват планки от даден стоманен лист. Планките идват като полигони от дадена CAD система. Това налага подреждане на входящите планки върху листа така, че да се получи минимална фира. Това е изрязването на определен брой фигури от даден материал, които в общия случай ще бъде полигон. Този полигон ще наричаме полигон за запълване. Виж фигура 1.4.

Тази задача е известна е още като Cutting Stock Problem или (CSP), [69]. Този проблем е NP- сложна комбинаторна задача [88]. В литературата се дават точни решения на задачата за фигури (планки), които са правоъгълници. По долу ще бъде даден алгоритъм за намиране на едно решение на задачата за разполагане на даден брой произволни геометрични фигури (планки описани чрез полигони) вписани в произволен контур (полигон за запълване). Методът позволява използуване на въртене и огледален образ на фигурата. Процесорното изчислителното време се увеличава съществено с увеличаване на броя на фигурите и на тяхната сложност като геометрия. Намирането на решение чрез изчерпване на всички възможни комбинации е неприемливо като твърде голям обем изчисления. При съвременното развитие на изчислителната техника е възможно да се реши сложна задача на супер компютър, който да намери всички възможни решения, но в повечето случаи това не е оправдано. Разбира се, разход на значителен изчислителен

ресурс зависи от важността на задачата. Представеният по-долу алгоритъм има възможност за паралелизация на изчислителните процеси. Но масовите задачи ще се реализират на настолен или персонален компютър. Целта е да се създаде алгоритъм, който дава за кратко време приемливо решение на сложни комбинаторни задачи използвайки мобилни изчислителни устройства. Ще въведем всички математически понятия, които се използват за описание на математическия модел и алгоритъм за решаването на оптимизационната задача.

От CAD системата се получават полигони, които са списъци от точки в координата на система XOY. В общия случай тези полигони съдържат и точки, които не са върхове на полигона, т.е. точки лежащи на една права. Тези точки ще наричаме излишни. Затова на всички полигони генерирани от CAD системата ще приложим функцията за изчистване на излишните точки. След премахване на излишните точки получаваме входящ допустим полигон описан чрез списък от точки

$$\Pi = list(pt_0, pt_1, ...pt_n), \tag{4.1}$$

където pt_i са върхове на полигона. Този списък има следните свойства: 1. Списъкът е подреден; 2. Списъкът е цикличен $pt_n = pt_0$; 3. Няма самопресичане.

В процеса на работа на алгоритъма ще ни бъде необходимо понятието сегмент, което е отсечка между два последователни върха. По този начин генерираме сегментите $e_1 = list(pt_0, pt_1), \ldots, e_n = list(pt_{n-1}, pt_n)$ и получаваме друга характеризация на полигона $\Pi = list(e_1, e_2, \ldots, e_n)$. Планки с криволинейни граници се апроксимират с полигони с достатъчен брой върхове. Примери за входящи допустими полигони са показани на фигура 1.5.

Полигон за запълване Δ, които трябва да се запълни също ще се описва със списък от точки:

$$\Delta = list(p_0, p_1, \ldots, p_m) \tag{4.2}$$

Контурът за запълване не трябва да бъде самопресичащ се.

Има два критерия за оптимизация на задачата:

1. Оптимизация на минимална височина на запълване - $minY$;

2. Оптимизация на броя на върховете на остатъчния полигон след изрязването на входящия полигон от полигона на запълване - $minVertex$.

Да разгледаме оптимизация на минималната височина.

Из между всички разполагания за най-добро ще смятаме онова което с най-малка ордината в координата система XOY. Ако се получат повече от едно решение с еднакви ординати $minY$ то взимаме това с най-добър коефициент на запълване. Ако има повече от едно решение с максимален коефициент на запълване избираме тези решения, които изрязват полигона за запълване с най-малък брой върхове. Това означава, че изрязването постига "правилни"фигури. Ако има повече от едно решение с най-малък брой върхове, тогава избираме първото в списъка.

За намиране на лице на полигон виж формула 2.12. Тази формула дава удвоеното лице на полигона.

Тогава определяме коефициента на запълване $ratio$ като:

$$ratio = \frac{\sum_{i=1}^{n} F_i}{A_P} \tag{4.3}$$

Следователно минималната стойност $ratio = 0$ на коефициента е при не запълване на полигона A_P. Максималната стойност $ratio = 1.0$ е при максимално запълване на A_P.

Разбира се, дефиницията за $minY$ води до положението как е въведен полигона за запълване P. Ако искаме да избегнем този проблем то решението на задачата ще трябва да се проведе за няколко различни ъгли на полигона за запълване P. Ъглите на ротация ще бъдат ъглите, които всеки сегмент сключва с абцисната ос. Броя на ротациите е равен на броя на сегментите на съответния полигон.

Да разгледаме оптимизация на броя на върховете на остатъчния полигон.

Из между всички разполагания за най-добро ще се смята онова което при изрязването на входящия полигон в полигона на запълване има най-малко върхове на остатъчния полигон. В настоящия дисертационен труд това е критерият един полигон да бъде "по-гладък". Намирането на подходящ полигон от входящите полигони ще става чрез сравнение на страните на двата полигона - входящият и полигона за запълване. Ако имаме пълно съвпадение на дължините на страните на двата полигона, то ще изберем този входящ полигон. Ако няма никакво съвпадение на страните на входящия полигон към то полигона на запълване, то ще използваме съвпадение на ъглите на двата полигона.

Ако се получат повече от едно решение с еднакъв брой върхове на остатъчните полигони $minVertex$, то взимаме първото решение от списъка с валидни решения. Разликата между $minY$ и $minVertex$ е в това, че при $minVertex$ се дава възможност на големи (дълги) полигони да "влезнат"първи в полигона на запълване, защото вероятността да произведат гладък остатък е по-голяма. Минималната гладкост на остатъчния полигон която може да се получи е полигон с три върха $list = (pt_0, pt_1, pt_2)$ с площ различна от нула.

4.2 Стратегия за избор на входящ полигон. Метаевристични методи.

Метаевристиката е мощен инструмент за намирането на оптимално или субоптимално решение на сложни комбинаторни задачи. Основна роля за развитието на метаевристичните методи е необходимостта да се намери приемливо решение за приемливо време при ограничени хардуерни ресурси. Съгласно [34] метаевристичните алгоритми (на английски: metaheuristic algorithms, накратко: метаевристики, metaheuristics) в компютърните науки са алгоритми за математическа оптимизация, с които се решават комбинаторни оптимизационни задачи. Тези задачи в общия случай са сложни, представяни чрез дискретизация на входящите данни. Такива задачи обикновено се характеризират със силна нелинейност, множество параметри, разнообразни сложни ограничения за удовлетворяване и множество – често противоречащи си – оптимизационни критерии. Дори и при един оптимизационен критерий може да не съществува нито едно допустимо решение. Само тогава не съществува оптимално решение. Ако има дори само едно допустимо решение, то тогава задължително съществува и оптимално решение.

Като цяло откриването на оптимално или дори близко до оптималното решение е трудно постижимо.

Терминът "метаевристика“ е въведен от Фред Глоувър в основополагащата му статия от 1986 г. като надграждане на термина "евристичен"алгоритъм, с който в най-общ смисъл се разбира алгоритъм за търсене на решение, базиран на пробата и грешката. Частицата "мета“ означава "отвъд”, "свръх”, "на по-високо ниво” и с метаевристичния алгоритъм се означава "по-висша” стратегия, която направлява и модифицира други евристични алгоритми, за да постигне решения по-добри от тези, които нормално биха се получили при търсене на локален оптимум [35], [36]. В допълнение всички метаевристични алгоритми балансират между глобално и локално търсене. Качествените решения на трудни оптимизационни задачи могат да се постигнат в разумно (т.е. полиноминално) време, но без гаранция, че ще бъдат постигнати (глобалните) оптималните решения. Двата основни компонента на всеки метаевристичен алгоритъм са: интензификация и диверсификация (intensification and diversification), или още изследване и експлоатация (exploration and exploitation). Диверсификацията означава да се генерират разнообразни решения, така че пространството на търсене да може да бъде проучвано в широк диапазон, докато интензификацията означава да се фокусира търсенето в локален регион, знаейки, че текущото най-добро решение се намира в този регион. При подбора на най-добрите решения трябва да се открие добър баланс между интензификацията и диверсификацията с цел да се подобри скоростта на сходимост на алгоритъм. Изборът на най-доброто текущо решение осигурява, че решенията ще схождат към оптимум, докато диверсификацията посредством избор на случайни стойности на променливи позволяващи да се избегне попадането в локален екстремум и в същото време да се повиши разнообразието на решението. Добрата комбинация от тези два основни компонента обичайно води до намиране на глобален оптимум. Съгласно [27] основните свойства на метаевристиката могат да бъдат обобщени както следва:

1. Метаевристиката осигурява стратегии за направляване на процеса на търсене;

2. Целта ни е ефективно да се изследва пространството на търсене, за да се открият оптимални или субоптимални решения;

3. Метаевристичните техники обхващат широк спектър процедури - от процедури за локално търсене до сложни процедури с машинно обучение;

4. Метаевристичните алгоритми са приближени и обикновено недетермистични;

5. Метаевристичните алгоритми в общия случай осигуряват механизми за избягване на съсредоточаване на търсенето само в ограничени области на пространството;

6. Основните концепции на метаевристиката могат да бъдат описани на абстрактно ниво;

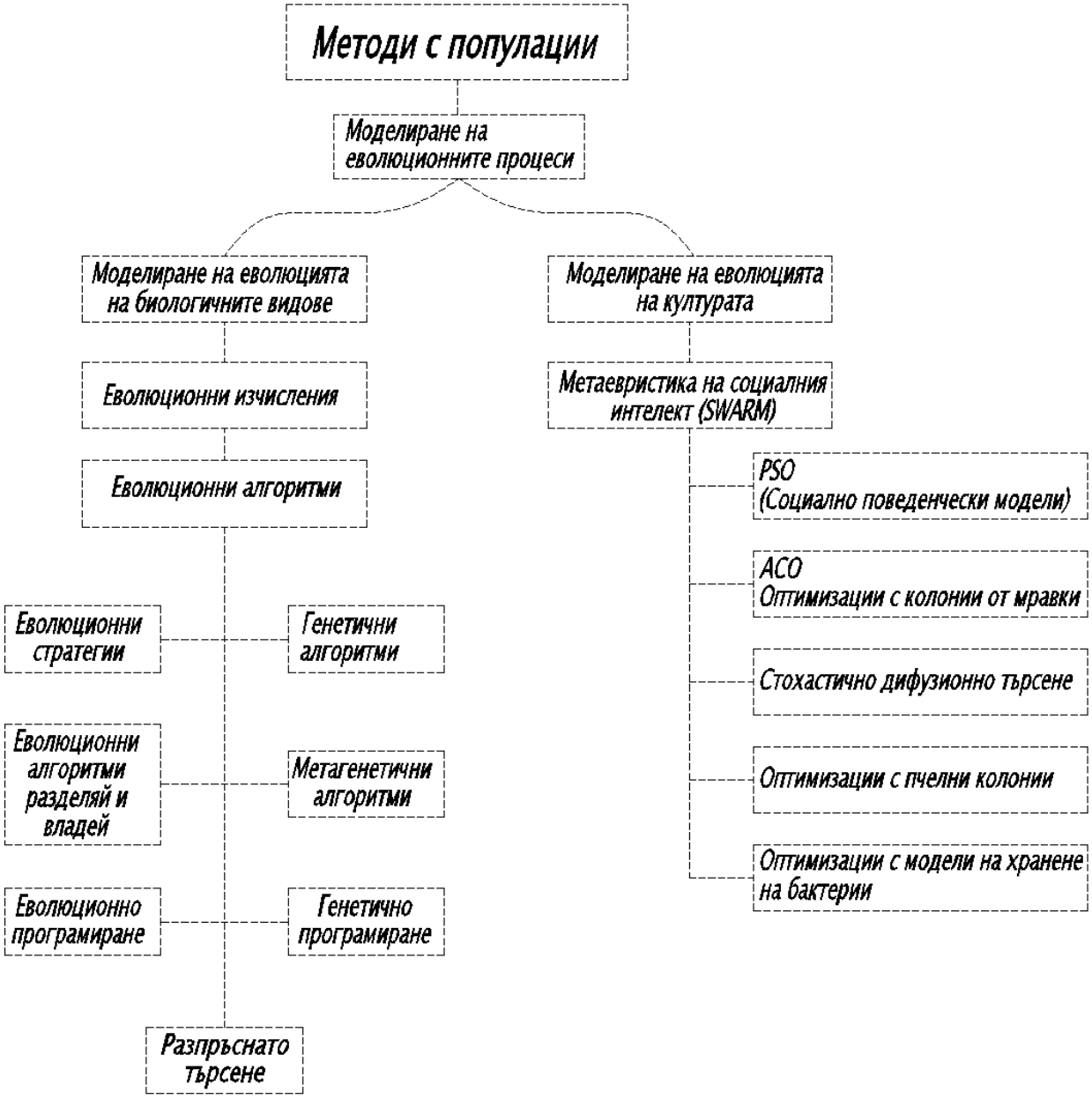
7. Метаевристичните алгоритми са универсални;

8. Метаевристката може да използва знание, специфично за областта под формата на евристика, която се управлява от стратегия на високо ниво;

9. За направляване на търсенето съвременната метаевристика използва натрупания опит при търсенето;

10. Метаевристиката представлява стратегии от високо ниво на изследване на пространството на търсене чрез използване на различни методи;

11. Изисква динамичен баланс между използването на две фундаментални концепции: диверсификация и интензификация.



Фигура 4.1: Класификация на метаевристики с популации.

Хибридната метаевристика осигурява възможности за повишаване на ефективността на търсене като комбинира различните метаевристични алгоритми. В настоящата дисертация е използвана хибридна метаевристика. Използвана е следната стратегия :

1. "Разпръснато търсене" от Еволюционните алгоритми от Метода с популации. Виж фигура 4.1;

2. Вероятностно предвиждане на избора на даден елемент;

3. Йерархично оценяване на решенията.

Имаме даден списък от полигони

$$\Pi_L = (list(list_1(pt_0, pt_1, ...pt_n), (list_2(pt_0, pt_1, ...pt_n), \dots (list_n(pt_0, pt_1, ...pt_n)), \quad (4.4)$$

където $list_i$ са полигони описание чрез върховете им. Върховете са описани чрез списък от две реални числа $list(X, Y)$, виж 2.1. Списъкът Π_L ще наричаме списък от входящи полигони.

Полигона Δ, които трябва да се запълни се описва със списък от точки:

$$\Delta = list(p_0, p_1, \ldots, p_m) \tag{4.5}$$

Контурът за запълване не трябва да бъде самопресичащ се. В дадената задача полигона за запълване Δ е един на брой. Ако имаме повече от един полигон то решението на задачата се повтаря за всеки един от тях.

Преди да се избере входящо множество е необходимо да се направи оценка за съвпадението на ъглите и страните от текущия входящ полигон

$$\Pi_i = (list(pt_0, pt_1, ...pt_n), i = 1, \ldots, n$$

към то ъглите и страните на полигона за запълване $\Delta = (list(pt_0, pt_1, ...pt_n)$. За целта се съставят два нови производни списъка на Π_i и Δ. Те съответно съдържат последователно подредени списъци $list(previosLength, Angle, NextLength)$. Добре е елементите на двата списъка да се съставят от конструктивни двойки. Конструктивната двойка е съставена от два елемента - първият с име, вторият с променлива. Може да се запише така $cons("name".AnyValue)$. В "name" записваме "angle" за ъгъл или "length" за дължина. В AnyValue - произволна стойност която може да бъде Integer, Real, String или List.

Или новите производни списъци са:

$\Pi_i = list((list(cons"previosLength"(distance_{pt_n,pt_0}))$
$(cons"angle"pt_n, pt_0, pt_1)$
$(cons"NextLength"(distance_{pt_0,pt_1})))$
...
$(list(cons"previosLength"(distance_p t_n pt_{n-1}))$
$(cons"angle"pt_0, pt_n, pt_{n-1})$
$(cons"NextLength"(distance_{pt_n,pt_0}))))$

Където n броя на върховете на полигона Π_i. Ако $(i + 1) > n$ тогава $i = 0$. Списъка Π_i е цикличен.

Съставя се и подобен списък и на полигона за запълване Δ. Списъка Δ е цикличен.

Всеки елемент от Списъка Π_i се сравнява със съответния елемент от списъка Δ. Най-големият брой на последователно съвпадащи се елементи от двата списъка ще ни даде най-голямата вероятност полигона Π_i да съвпадне с полигона Δ. За да направим коректно сравнение на двата списъка ще трябва да изберем кой списък ще бъде основен. Под основен списък ще разбираме този, които има по-голяма дължина (по-голям брой елементи). В списъка $list(A)$ може да бъде или Π_i или Δ. Не е задължително броя на върховете на Δ да бъде по-голям от броя на върховете на Π_i. Обхождането на двата списъка ще става на базата индекси. Първата итерация е когато индекс 0 от списъка $listB$ съвпада с индекс 0 на списъка $listA$. Сега първия индекс от $listA$ се увеличава с 1. Следва втора итерация когато индекс 0 от списъка B съвпада с индекс 1 на списъка A. Като индекс 0 от $listA$ отива като последен в $listA$. Или списъка $listA$ има циклично поведение. Това се повтаря докато преминем през всички индекси на $listA$ или цикълът се повтаря толкова пъти колкото е дължината на списъка $listA$. За всяка проверка

записваме броя на последователните съвпадения. Оценката на списъка $listB$ се дава със следната формула:

$$k = \sum Ratio_{angle_i} + \sum Ratio_{Length_i}, \tag{4.6}$$

където

$$Ratio_{angle_i} = (if|(angle_i - angle_{Bi})| < fuzz \rightarrow return1., else0.).$$

При ъглите търсим пълно съвпадение. Като Ai е ъгъл от списъка $listA$, а Bi е съответния ъгъл от списъка $listB$.

След като имаме пълно съвпадение на съответните ъгли $Ratio_{angle_i} = 1.$, тогава пристъпваме към оценка на съответните им дължини.

$$Ratio_{Length_i} = (if(length_{Ai} > length_{Bi}) \rightarrow return(\frac{length_{Bi}}{length_{Ai}}), else(\frac{length_{Ai}}{length_{Bi}}) \tag{4.7}$$

Условие 4.7 дава коефициенти близки или равни на 1., без значение коя дължина е по-голяма $lengthA_i$ или $lengthB_i$. Това е така, защото търсим полигони, на които страните им почти съвпадат.

Както се вижда от формула (4.6) полигоните с по-голям брой върхове ще имат по-голяма вероятност за по-високи оценки на съвпадение. Това е добре, защото след изваждането на двата полигона $A \backslash B$ ще получим полигон с по-малко върхове. Коефициентът от формула (4.6) може да се използва като оценка за подобие или еднаквост на фигури.

4.3 Стратегия за избор на едно решение от валидни разположения.

Нека разгледаме два полигона. Полигона за запълване $\Delta = list(pt_1, pt_2, pt_3, pt_4)$ и входящ полигон $\Pi_i = list(pt_1, pt_2, pt_3, pt_4, pt_5, pt_6)$. Десният Π_i полигон е входящия. Левият полигон Δ е полигона за запълване. Полигона Π_i не е триъгълник! Необходимо е добавянето на подробни точки по границата на полигона за запълване Δ. Тези точки ще бъдат области на поставяне на полигона Π_i и проверката дали полигона Π_i е в полигона Δ. Ако проверката е удовлетворена, то записваме това решение като възможно. Разрешена е ротация на полигона. Огледален образ не е приложен. От тези валидни решения оценяваме тези, които имат най-голяма допирна дължина с полигона Δ и разстоянието $LengthA = distance(pt_0, pt_M)$. Между дължината на опиране и дължината $LengthA$ има зависимост. Тази зависимост е приета от автора. Разбира се могат да се напишат и други зависимости. Оценката на всяко решение е $eval = LengthOfTouch + (2.0 * LengthA)$. Тя дава се тежест на разстоянието от центъра на тежестта на полигона до избрана точка от полигона на запълване. В случая това е точката най-долу, най-вляво. Може да се избере друга точка без значение дали е от множеството точки от полигона за запълване. След като сме оценили решенията, избираме решението със зеления контур на полигона. Изваждаме $\Delta \setminus \Pi_i$.

След като сме избрали първо решение пристъпваме към избор на второ. Избора на входящ полигон става по процедурата описна в точка 4.2. Тъй като целта ни е максимално уплътняване на решенията то избора на второто решение трябва да се търси по контура на вече намерените решения. На този етап се прилага

йерархично оценяване на решенията. Около избраните полигони построяваме правоъгълен контур. Ще го наречем box. Първо ще търсим решения, които влизат в box от валидните решения, ако не намерим такива използваме за оценка всички валидни решения. Оценяването на решенията вътре в box също става по формулата: $eval = LengthOfTouch + (2.0 * LengthA)$. Тук трябва да се отбележи, че тази процедура за избор на решения след първото трябва да обходи всички входящи полигони и тогава да се вземе това с най-висока оценка. В софтуера разработен от автора това обхождане не се прави поради липсата на изчислителен ресурс, но предложения метод не е ограничен в тази посока. При наличие на достатъчно мощен хардуер плюс графични процесори алгоритъма ще даде много близки резултати до глобалния оптимум.

След изрязването на двата полигона се получава новия контур. Този контур ще послужи като контур на запълване за следващите полигони. Както се забелязва нови контур не следва напълно стария. Новия контур целенасочено е редуциран. За повече информация за начина на редуциране на контура виж точка 2.6.

$$ratio_{Global} = \frac{A_{box}}{A_{\Pi_L}} \leq 1.0 \tag{4.8}$$

4.4 Намиране на едно възможно разположение на планка в полигона за запълване.

Намирането на едно възможно разполагане на планките (представяни като полигони) става чрез поставяне на входящ полигон в полигона за запълване и прилагане на функцията изваждане на два полигона. Функцията е описана по-долу. В зависимост от изискването на конкретния случай може да построим производни полигони от входящия полигон често с разрешение за прилагане на ротация и огледална симетрия.

Броят на ротациите, на които можем да ротираме дадения полигон е произволен. Колкото повече ъгли имаме в списъка за ротиране толкова е по-голяма вероятността да получим възможно решение. С цел да ограничим произволното ротиране на полигона без да има качествено решение (полигона да бъде в полигона на запълване) е необходимо да изберем подходящи ъгли на ротиране. Като начало могат да се вземат ъглите на сегментите на входящия полигон с абсцисната ос X. След това могат да се добавят точните ъгли $(0; 0.5\pi; \pi; 1.5\pi)$. Ако е необходимо и огледано прилагане на входящия полигон, то броят на допълнителните полигони ще се увеличи. Някои състояния на ротация ще бъдат напълно идентични като геометрия.

Проверката дали даден полигон се съдържа в друг може да стане по два начина:

1. Проверка за всеки един възел дали е в полигона за запълване. Тази проверка е добре да стане с цикъл while, ако текущата тествана точка е извън контура решението отпада, без да продължава нататък;

2. Проверка дали има тривиално пресичане на двата полигона. Ако има пресичане, то решението отпада, иначе – се приема.

Ако първият критерий за оптимално решение е $minY$, то може да се провери само върха на полигона P с най-малка ординатата (в случая p_7). Но най-добре е да се направи за всички върхове на полигона за запълване P. Друг критерий за избор

на решение е когато при изрязването на двата полигона се получи гладка фигура. Критерият за гладка фигура ще бъде минимален брои на върховете след изрязване на дадения полигон от полигона за запълване. Възможна е и комбинация за критериите. На избраните валидни решения за $minY$, ще се приложи критерия за минимален брой на върховете след изрязването им с полигона за запълване.

При наличие на многоядрен процесор и езика на които се пише приложението позволяват паралелни изчисления. Паралелните изчисления могат да се направят и за останалите възможни състояния на входящия полигон.

Алгоритъмът се повтаря за генерирания огледален образ на входящия полигон. За огледалния полигон са валидни всички генерирани ъгли на ротация.

4.5 Премахване на реалната фира при 2D разкроя.

За изрязването на фигурите се използва режещ инструмент с определена характеристика на рязането. По тази причина трябва да се осигури разстояние между полигоните. Въвеждаме параметъра $cutW$ за ширина на фугата (ножа) между полигоните.

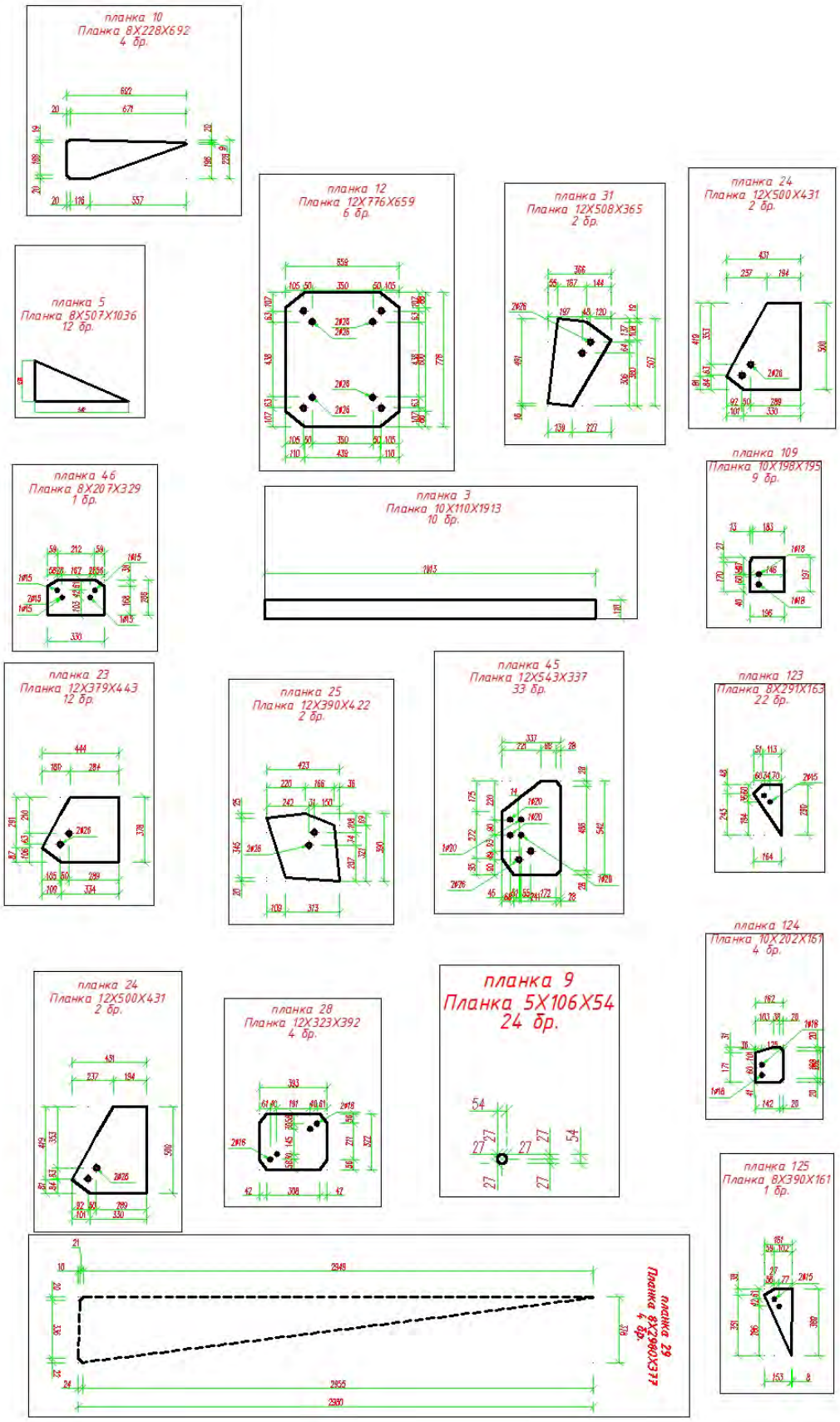
Този проблем е решен като входящите полигони са разширени с ивица с ширина $0.5cutW$ и всеки един сегмент е отместен извън полигона с $0.5cutW$.

4.6 Резултати при $2D$ разкрой. Примери.

В следващите страници ще илюстрираме разкрояване на следните входящи полигони (планки). Примерите са взети от реален строителен обект.

Полигонът за запълване ще бъде стандартен правоъгълен лист със широчина 1500мм. За входящите полигони ще покажем резултати с контурната линия на полигона отместена със широчината на ножа. Ширина на ножа 5мм. Размерите на планките са в милиметри. Преди да започнем изчисление на входящите полигони те минават през препроцесорна обработка, която включва:

1. Сортиране на планките по дебелина;
2. Прочитане на бройката им;
3. Намиране на контура на планките.



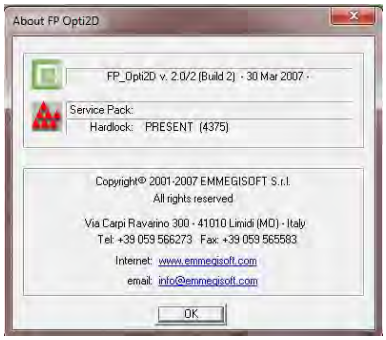
Фигура 4.2: Входящи полигони.

Сравняване на настоящия алгоритъм с комерсиален продукт.

За изработването на една стоманена конструкция от фигура 1.1 отнема около три месеца. Брои планки в една такава конструкция е около 1000. Брои уникални планки около 100. Броя итерации е относителен. Той зависи от това дали материалът е скъп или не е. Могат да се пуснат няколко итерации на различни компютри и да се вземе най-доброто от тях. Това е въпрос на решение на потребителя. При наличието на хардуер могат да се направят няколко итерации докато се получи приемлив отпадък.

В настоящето сравнение ще използваме планките дадени на фигура 4.4. Общия брои е 106. С тези планки е направено сравнение между комерсиалния продукт и разработения метод в настоящата дисертация. Намирането на крайното решение с представения метод е за една итерация. При по еднородни планки може да се направят повече итерации.

Използван комерсиален продукт.



Фигура 4.3: Комерсиален продукт FP Opti2D.

Съгласно сайта на производителя, продукта предлага следните функционалности:

1. User friendly graphical interface.
2. Handles panels, metal sheets and glazing.
3. Specific functions for aluminum composite panels.
4. Defines the parts to be optimized.
5. Directly uses the panels and glazing lists generated by FP PRO.
6. Imports work lists from Excel and from external calculation programs.
7. Manages the stock of full sheets and short bars.
8. Graphic display and printing of the optimized sheets, with clear indication of the cuts to be made and layout of the workpieces.
9. Provides statistical information on use of the sheets.

Optimization Printout v. Date/Time: 28.11.2018 г. 09:50:54 - Page 8

Opti2D (s.n. 4375)

WORKORDER				MATERIAL	
Work Order	Planki St 28.11.2018.R01			Material	PLANKI_STOMANA
Description	Planki St 28.11.2018			Full Description	
Date	28.11.2018 г.			Thickness	0,0 mm
U.M	mm	Kg	mq	Weight	0.00 Kg
Status	Optimized				

Фигура 4.4: Лист от разпечатката на комерсиалния продукт.

Както се вижда от разпечатката комерсиалния продукт версия V.2.0/2 (Build 2) работи само с правоъгълни планки. Разрешено е завъртане на планките. По

данни на оператора оптимизацията е продължила около 20 минути или 1200 сек. Размери на листа за запълване 1500мм / 6000мм. Време за работа не е посочено в разпечатката. Отпадък 18.2%. Всички планки са с отместен контур на 5mm. С този контур ще работим в изчисленията. Реалният отпадък може да се изчисли както следва:

1. Обща използвана площ - 2 листа x 1500mm x 6000mm = 18 000 000 ^{2};
2. Реална площ на всички 106 планки - 9 859 421 ^{2};

Реалната фира е 8 140 579 ^{2}. Или 45% фира. Реално запълване 54%.

Резултати от представения метод в настоящия труд. Брой планки 106. Всички планки са с отместен контур на 5мм от действителния им контур. Разрешено завъртане на планките : Да. Контур : $offsetPtL$.

Таблица 4.1: Сравнение на комерсиален продукт и представения алгоритъм

Включени параметри	Контур	Брой планки	Ratio	Време [s]
1	2	3	4	5
(a) Mirror:Yes, Rortate:Yes, Intervals:No	*box*	106	0.72	18 776
(b) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.70	109 519
(c) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.71	227 846
(d) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.69	7 555
(e) Mirror:Yes, Rortate:Yes, Intervals:No	$Offset$	106	0.76	41 031
(f.1) Mirror:Yes, Rortate:Yes, Intervals:No	*box*	51	0.71(0.67)	2 194
(f.2) Mirror:Yes, Rortate:Yes, Intervals:No	*box*	55	0.57 (0.44)	2 454
(Комерсиален продукт) Mirror:N/A, Rortate:Yes, Intervals:N/A	*box*	106	0.54	1200

За случай $f.1$ и $f.2$ в скоби са дадени запълванията на планките спрямо основния лист 1500мм x 6000мм. Използвани са същите параметри както при комерсиалния продукт. Комерсиалният продукт има редица ограничения. Някои от тях са, че фигурите се апроксимират до правоъгълник, огледален образ на фигурите не се използва. Ъглите на завъртане са сведени до два: 0° и 90 °. По отношение на коефициента на запълване *Ratio*, представения алгоритъм е много по-добър. виж таблица 4.1. Коефициентите са 0.71 за настоящия алгоритъм съпоставен с 0,67 за комерсиалния продукт. Второто сравнение е 0,57 за настоящия алгоритъм съпоставен с 0,44 за комерсиалния продукт. При големи обеми от работа или скъп материал, от които ще се изрязва разликата нараства още повече в полза на представения алгоритъм. Представеният алгоритъм в настоящата дисертация е по-добър от комерсиалния продукт, защото дава по-голям процент на уплътнение на фигурите. Друго негово предимство е много добрата му пригодност към паралелизация на изчисленията.

Глава 5

Заключение

1D разкрой.
Както се вижда от сравнителната таблица 3.4 метода на мравките (ACO) дава най-добър резултат за много кратко време. В случая ACO метода проявя характер на *Greedy* алгоритъм. ACO алгоритъма е по-добър от комерсиалния продукт и по време и по оптимизиране. За големи обеми на разкрояване и компютри с по-слаби процесори ACO метода е много подходящ.

2D разкрой.
След проведените тестове на различни видове планки заключението е, че за приемливо решение на дадена задача са необходими по-голям на брой итерации. Резултатите в настоящата дисертацията са при 3 броя итерации. Приети са три броя итерации, защото намирането на едно решение отнема значително време. За някои видове планки този брой се оказва недостатъчен. Тестовете бяха направени на настолен компютър с операционна система Windows ®10 Pro, x64. Процесор Intel ®Core (TM) i5-9500@3.0 GHz. Използвани процесори един. Тип на процесора CPU. Въпреки, че процесорът е едно от последните поколения към дата на писане на настоящия труд се оказва слаб за по-висока степен на уплътняване на планките. Но ако се търси сравнително бързо подреждане и неголям брой планки настолният компютър може да справи. Представеният подход за решаване на проблема може да се приложи в 90% от случаите в практиката. Трябва да се отбележи, че за малко по-голяма плътност на решението е необходимо значително по-голямо време за изчисление. Дали ще се жертва време за сметка на материал зависи от това доколко е скъп даденият материал, от които ще се изрязват фигурите. Като следващо развитие на проблема ще бъде разработването му за хардуер с достатъчни изчислителни ресурси базиран на GPU процесори. Резултатите от настоящата дисертация са докладвани на различни национални и международни конференции.

5.1 Списък на публикациите

1. Evtimov G. Fidanova S. **"Ant Colony Optimization algorithm for 1D Cutting Stock Problem"**, pp. 25-31, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*, K. Georgiev, I. Georgiev (eds.), Springer, Vol. 728, ISBN 978-3-319-65529-1, doi:10.1007/978-3-319-65530-7_3

2. Evtimov G. Fidanova S. **"Subtraction of Two 2D Polygons with Some Matching Vertices"**, pp. 80-87, *Numerical Methods and Applications, 9th International Conference NM&A'18, Borovets, Bulgaria, 2018*, Geno Nikolov, Natalia Kolkovska, Krassimir Georgiev (eds.), ISBN 978-3-030-10691-1, doi:10.1007/978-3-030-10692-8_9

3. Evtimov G. Fidanova S. **"Heuristic Algorithm for 2D Cutting Stock Problem"**, pp.350-357 *Large-Scale Scientific Computing, 11th International Conference, LSSC 2017, Sozopol, Bulgaria*, Ivan Lirkov, Svetozar Margenov (eds.), Springer, Vol. 793, ISBN 978-3-319-73440-8, doi:10.1007/978-3-319-73441-5_37

4. Evtimov G. Fidanova S. **"Analyses and Boolean Operation of 2D Polygons"**, pp. 107-118, *Advanced Computing in Industrial Mathematics, BGSIAM 2017*,K. Georgiev, I. Georgiev (eds.), Springer, Vol. 793, ISBN 978-3-319-97276-3, doi:10.1007/978-3-319-97277-0_9

5. Evtimov G. Fidanova S. **"2D Optimal Cutting Problem"**, pp. 33-40, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*, K. Georgiev, I. Georgiev (eds.), Springer, Vol. 728, ISBN 978-3-319-65529-1, doi:10.1007/978-3-319-65530-7_4

6. Ana Avdzhieva, Todor Balabanov, Georgi Evtimov, Ivan Jordanov, Nikolai Kitanov, Nadia Zlateva, **Two Dimensional Optimal Cutting Problem, 120th European Study Group with Industry (ESGI'120)**, Problems & Final Reports

7. V. Bodurov, D. Dimov, G. Evtimov, I. Georgiev, S. Harizanov, G. Nikolov, V. Pirinski, **The 2D/3D Best-Fit Problem, 113th European Study Group with Industry (ESGI'113)**, Problems & Final Reports, pp. 62-73, 2015. ISBN:978-619-7223-12-5

8. A. Avdzhieva, T. Balabanov, G. Evtimov, D. Kirova, H. Kostadinov, T. Tsachev, S. Zhelezova, N. Zlateva, **Optimal Cutting Problem, 113th European Study Group with Industry (ESGI'113)**, Problems & Final Reports, pp. 49-61, 2015. ISBN:978-619-7223-12-5

5.2 Апробация на резултатите

Резултатите в настоящия дисертационен труд са докладвани на различни мероприятия на секция "Паралелни алгоритми" към ИИКТ-БАН като:

1. 113th European Study Group with Industry (BGSIAM - 2015);
2. 11th Annual Meeting of the Bulgarian Section of SIAM (BGSIAM - 2016);
3. 120th European Study Group with Industry (ESGI'120 - 2016);
4. 12th Annual Meeting of the Bulgarian Section of SIAM (BGSIAM - 2017) ;
5. 13th Annual Meeting of the Bulgarian Section of SIAM (BGSIAM - 2018);
6. Conference on Large-Scale Scientific Computations LSSC'17, Sozopol, 2017;
7. Ninth International Conference on Numerical Methods and Applications NM&A'18, Borovets.

5.3 Приноси

Приносите в тази дисертация могатда бъдат разделени на научни и научно-приложни, като научните приноси касят разработването на методи и алгоритми за 1D и 2D разкрой, а научно-приложните се отнасят към тяхната програмна реализация.

Научните приноси са:

- Разработен е алгоритъм за оптимален разкрой в едномерното пространство;
- Разработен е алгоритъм за оптимален разкрой в двумерното пространство;
- Разработен е метод за двумерен разкрой на базата на хибридна оптимизация;

Научно-приложните приноси са:

- Направена е програмна реализация на алгоритъма за едномерен разкрой;
- Направена е програмна реализация на алгоритъма за двумерен разкрой;

Резултатите на настоящия дисертационен труд могат да се използуват в най-различни области от науката и инженерната практика:

- Проектирането на сгради и съоръжения;
- Проектирането на износването на детайли при машините както и проектирането на механизми;
- Земна механика - консолидация на почвите;
- Авиационната техника - намиране на оптимален път в среда с препятствия;
- И в много други области, където се използуват CAD-системи.

Приложните приноси могат да се развият и във фирми които произвеждат стоманени конструкции. Приложния софтуер може да се внедри и в други отрасли които не са свързани със строителството на сгради и съоръжения. Друго много голямо приложно предимство е, че входните данни се вземат директно от базата данни на CAD системата, с която е проектирано съоръжението. Това многократно повишава скоростта на получаване и точността на данните, с които работи програмата. С няколко кликвания могат да се изберат хиляди полигони и да стартира програмата за разкрой. Софтуерът може сам да отстрани или коригира "неправилните" полигони за да се избегнат неточности в изходните резултати. Решението протича в рамките на няколко минути в зависимост от производителността на компютърната система, на която софтуерът се използува. Алгоритъмът, разработен в представеният дисертационен труд, позволява използуване на огледален образ на полигоните, ротация и други операции, които могат да доведат до съществено подобрение на полученото приближено решение. Разбира се, за целите на широкомащабни научни изследвания алгоритъмът може да се внедри на супер-компютър тъй като позволява значителна паралелизация на изчисленията.

Библиография

[1] Valerio de Carvalho J.M. **"Lp models for bin packing and cutting stock problems"**, European Journal of Operational Research 141:253–273,(2002).

[2] Chen C.L.S., Hart S.M., Tham W.M. **"A simulated annealing heuristic for theone-dimensional cutting stock problem"**, European Journal of Operational Research93:522–535,(1996).

[3] Foerster H., Wscher G. **"Pattern reduction in one-dimensional cutting stock problems"**, In: Proceedings of the 15th Trienal Conference of the International Federationof Operational Research Societies, (1999).

[4] Falkenauer E., **"A hybrid grouping genetic algorithm for bin packing"**, Journal of Heuristics Vol. 2, 1996

[5] Song X., Chu C.B., Nie Y.Y., Bennell J.A. **"An iterative sequential heuristic procedureto a real-life 1.5-dimensional cutting stock problem"**. European Journal of OperationalResearch 175:1870–1889, (2006).

[6] Suliman S.M.A. **"Pattern generating procedure for the cutting stock problem"**, IntJ Production Economics 74:293–301, (2001).

[7] Vahrenkamp R. **"Random search in the one-dimensional cutting stock problem"**, European Journal of Operational Research 95:191–200, (1996).

[8] Vanderbeck F. **"Exact algorithm for minimizing the number of setups in theone-dimensional cutting stock problem"**, Operations Research 48:915–926, (2000).

[9] O'Rourke J., **"Computational Geometry in C second edition"**, ISBN 0 521 64976 5, http://www.cambridge.org

[10] de Berg M., van Kreveld M., Overmars M., Schwarzkopf O.C. **"Computational Geometry: Algorithms and Applications, Second Edition"**, ISBN 3-540-65620-0

[11] Graham, R.L. (1972). **"An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set"**, Information Processing Letters. 1 (4): 132–133. doi:10.1016/0020-0190(72)90045-2 , https://en.wikipedia.org/wiki/Graham_scan

[12] Evtimov G., Fidanova S. **"Subtraction of Two 2D Polygons with Some Matching Vertices"**, pp. 80-87, Numerical Methods and Applications, 9th International Conference NM&A'18, Borovets, Bulgaria, 2018, ISBN 978-3-030-10691-1, doi:10.1007/978-3-030-10692-8_9

[13] Evtimov G., Fidanova S. **"Heuristic Algorithm for 2D Cutting Stock Problem"**, pp.350-357, Large-Scale Scientific Computing, 11th International Conference, LSSC 2017, Sozopol, Bulgaria, Springer, Vol. 793, ISBN 978-3-319-73440-8, doi:10.1007/978-3-319-73441-5_37

[14] Evtimov G., Fidanova S. **"Analyses and Boolean Operation of 2D Polygons"**, pp. 107-118, Advanced Computing in Industrial Mathematics, BGSIAM 2017, Springer, Vol. 793, ISBN 978-3-319-97276-3, doi:10.1007/978-3-319-97277-0_9

[15] Evtimov G. Fidanova S. **"2D Optimal Cutting Problem"**, pp. 33-40, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*, Springer, Vol. 728, ISBN 978-3-319-65529-1, doi:10.1007/978-3-319-65530-7_4

[16] Evtimov G., Fidanova S. **"Ant Colony optimization algorithm for 1D Cutting Stock Problem"**, pp. 24, *Advanced Computing in Industrial Mathematic, BGSIAM 2016*

[17] Fidanova S. **"ACO Algorithm with Eddditional Reinforcement, From Ant Colonies to Artificial Ants"**, *Lecture Notes in Computer Science, N2463, Spinger, Germany, 2002,292-293*

[18] Fidanova S., Marinov P., Alba E. **"ACO for Optimal Sensor Layout, In Proc. of Int. Conf. on Evolutionary Computing, Valencia, Spain, Joaquim Filipe and Janus Kacprzyk eds."**, *SciTePress-science and Tchnology Publications Portugal, ISBN 978-989-8425-31-7, 2010, 5-9*

[19] Fidanova S., Marinov P., Alba E. **"Wireless Sensor Network layout, In Monte Carlo Methods and aplications"**, *K. Sabelfeld, I. Dimov eds., Chapter 9, De Gruyter Pub, Berlin, germany, 2012, 39-46*

[20] Fidanova S., Shindarov M., Marinov P. **"Mono-objective algorithm for Wireless Sensor Network layout, In Proc of OMKO-NET Int."**, *Conference, Southampton, UK, 2012a, 57-63*

[21] Fidanova S., Shindarov M., Marinov P. **"Optimal Sensor Layou Using Multi-objective Metaheuristic, In Proc of OMKO-NET Int."**, *Of Int. Conference of information systems and Grid Technologies, Sofia, Bulgaria, 2011, 114-122*

[22] Fidanova S., Shindarov M., Marinov P. **"Multi-Objective ant algorithm for Wireless Sensor Network Positioning."**, *Proceedings of Bulgarian academy of Science, Vol 66(3), 2013, 353-360.*

[23] Fidanova S., Shindarov M., Marinov P. **"Wireless Sensor Positioning ACO Algorithm."**, *Studies of Computational intelligence, J.Kacprzyk and K. atanassov eds., Spinger, Germany*

[24] Gonalves J.F., **"A hybrid genetic algorithm-heuristic for a two-dimensional or-thogonal packing problem"** Eur J Oper Res, Vol 183, No 3, 2007, 1212-1229.

[25] V. Bodurov, D. Dimov, G. Evtimov, I. Georgiev, S. Harizanov, G. Nikolov, V. Pirinski, **The 2D/3D Best-Fit Problem, 113th European Study Group with Industry (ESGI'113)**, Problems & Final Reports, pp. 62-73, 2015. ISBN:978-619-7223-12-5

[26] A. Avdzhieva, T. Balabanov, G. Evtimov, D. Kirova, H. Kostadinov, T. Tsachev, S. Zhelezova, N. Zlateva, **"Optimal Cutting Problem, 113th European Study Group with Industry (ESGI'113)"**, Problems & Final Reports, pp. 49-61, 2015. ISBN:978-619-7223-12-5

[27] Боровска П., **"Синтез и анализ на паралелни алгоритми"**, ISBN:978-954-438-764-4

[28] https://www.sit.ac.jp/user/konishi/JPN/Tech_inform/Pdf/GeometricalMoment.pdf

[29] Р. Даскалов, Е. Даскалова, **"Висша математика, част I, Аналитична геометрия"**, Габрово, 2012

[30] Antonio, Franklin **"Chapter IV.6: Faster Line Segment Intersection"**, In Kirk, David (ed.). Graphics Gems III. Academic Press, Inc. pp. 199–202. ISBN 0-12-059756-X, (1992).

[31] "Weisstein, Eric W. **"Line-Line Intersection. From MathWorld"**, A Wolfram Web Resource. Retrieved 2008-01-10.

[32] Shimrat, M., **"Algorithm 112: Position of point relative to polygon"**, 1962, Communications of the ACM Volume 5 Issue 8, Aug. 1962

[33] Eric Haines, **"Point in Polygon Strategies"in Graphics Gems IV"**, (1994) http://geomalgorithms.com/a03-_inclusion.html

[34] https://bg.wikipedia.org/wiki/РЁРҳСЪРөРҳРўСЃРчСҒСЪРчСЊР„Рч_РөРњРуР«СЃРчСЪРёРч

[35] Glover F., **"Future paths for integer programming and links to artficial intelligence"**, Computers and Operations Research,13,533-549 (1986).

[36] Glover F. and Laguna M., **"Tabu Search"**, Kluwer, Boston, (1997).

[37] Balaban, I. J. **"An optimal algorithm for finding segments intersections"**, Proc. 11th ACM Symp. Computational Geometry, pp. 211–219, doi:10.1145/220279.220302, (1995).

[38] Bartuschka, U.; Mehlhorn, K.; Näher, S. **"A robust and efficient implementation of a sweep line algorithm for the straight line segment intersection problem"**, in Italiano, G. F.; Orlando, S. (eds.), Proc. Worksh. Algorithm Engineering, (1997).

[39] Bentley, J. L.; Ottmann, T. A. **"Algorithms for reporting and counting geometric intersections"**, IEEE Transactions on Computers, C-28 (9): 643–647, doi:10.1109/TC.1979.1675432, (1979).

[40] **"Point in Polygon, One More Time..."**, Archived 2018-05-24 at the Wayback Machine, Ray Tracing News, vol. 3 no. 4, October 1, 1990.

[41] http://www.theswamp.org/index.php?topic=1891.0

[42] Boissonat, J.-D.; Preparata, F. P. **"Robust plane sweep for intersecting segments"**, (PDF), SIAM Journal on Computing, 29 (5): 1401–1421, doi:10.1137/S0097539797329373, (2000).

[43] Brown, K.Q. **"Comments on "Algorithms for Reporting and Counting Geometric Intersections"**, IEEE Transactions on Computers, C-30 (2): 147, doi:10.1109/tc.1981.6312179, (1981).

[44] Chazelle, Bernard; Edelsbrunner, Herbert (1992), **"An optimal algorithm for intersecting line segments in the plane"**, Journal of the ACM, 39 (1): 1–54, doi:10.1145/147508.147511

[45] Alvarez-Valdes R, Parajon A, Tamarit J. M, **"A computational study of heuristicalgorithms for two-dimensional cutting stock problems"**, 4th metaheuristics inter-national conference (MIC2001), 2001, 16-20.

[46] Chen, E.Y.; Chan, T.M. **"A space-efficient algorithm for segment intersection"**, Proc. 15th Canadian Conference on Computational Geometry (PDF), (2003).

[47] Clarkson, K.L. **"Applications of random sampling in computational geometry"**, II Proc. 4th ACM Symp. Computational Geometry, pp. 1–11, doi:10.1145/73393.73394, (1988).

[48] Eppstein, D.; Goodrich, M.; Strash, D. **"Linear-time algorithms for geometric graphs with sublinearly many crossings"**, Proc. 20th ACM-SIAM Symp. Discrete Algorithms (SODA 2009), pp. 150–159, (2009).

[49] Cintra G., Miyazawa F., Wakabayashi Y., Xavier E., **"Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation"**, European Journal of Operational Research, European Journal of Operational Research, Vol. 191, 2008,61-85.

[50] Mulmuley, K. **"A fast planar partition algorithm"**, I Proc. 29th IEEE Symp. Foundations of Computer Science (FOCS 1988), pp. 580–589, doi:10.1109/SFCS.1988.2197, (1988).

[51] Preparata, F. P.; Shamos, M. I. **"Section 7.2.3: Intersection of line segments"**, Computational Geometry: An Introduction, Springer-Verlag, pp. 278–287, (1985).

[52] Shamos, M. I.; Hoey, Dan **"Geometric intersection problems"**, 17th IEEE Conf. Foundations of Computer Science (FOCS 1976), pp. 208–215, doi:10.1109/SFCS.1976.16, (1976)

[53] Dorigo M., Manieezz M., mColorni A., **"The ant syste: Optimization by a colony of cooperating agents"**, IEEE Transactions on Systems, Man and Cybernetics B, Vol 26(1), 1996, pp 29-41

[54] Dorigo M., **"Optimization, Learning and Natural Algorithms"**, PhD thesis, Politecnico di Milano, Italie, 1992.

[55] Lodi, A., Martello S., Vigo D., **"Recent advances on two-dimensional bin packingproblems"** , Discrete Applied Mathematics, Vol. 123, 2002, 379-396.

[56] Falkenauer E., **"A hybrid grouping genetic algorithm for bin packing"**, Journal of Heuristics, Vol. 2, 1996, pp. 5-30

[57] Hinterding R., Khan L., **"Genetic algorithms for cutting stock problems: with and without countiguity"**, InX. Yao, editor, Progress in Evolutionary Computation, Berlin, Germany, Springer, 1995, pp. 166-186

[58] Jaromi M.H., Tavakkoli-Moghaddam, R., Makui, A. Shamsi A., **"Solving an one-dimensional cutting stock problem by simulated and tabu search"**, J. of Industrial Engineering International, Vol. 8 (1), Spinger, 2012, pp. 24

[59] Reeves C., **"Hybrid genetic algorithms for bin-packing and related problems"**, Annals of Opera Research 63, 1996, pp.371 396

[60] Vink M., **Solving combinatorial problems using evolutionary algorithms**, 1997 http://citeseer.nj.nec.com/vink97solving.html

[61] Parmar K., Prajapati H., Dabhi V., **"Cutting stock problem: A survey of evolutionary computing based solution"**, In Proc. of Green Computing Communication and Electrical Engineering, 2014, doi:10.1109/ICGCCEE.2014.6921411.

[62] Dusberger, F., Raidl, G.R., **"A variable neighborhood search using very large neighborhood structures for the 3-staged 2-dimensional cutting stock problem"**, In: Blesa, M.J., Blum, C., Voß, S. (eds.) HM 2014. LNCS, vol. 8457, pp. 85–99. Springer, Cham (2014). doi:10.1007/978-3-319-07644-7_7

[63] Dusberger, F., Raidl, G.R., **"Solving the 3-staged 2-dimensional cutting stock problem by dynamic programming and variable neighborhood search."**, Electron. Notes Discret. Math. 47, 133–140 (2015)MathSciNetCrossRefzbMATH Google Scholar

[64] Lin Z., Li Y., **"An Efficient Algorithm for Intersection, Union and Difference between Two Polygons"**, *In proc. of Computer Network and Multimedia Technology, Wuhan, China, 2009, 1-4*

[65] Gonçalves J.F., **"A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem"**, *European Journal of Operational Research*, Vol 183, No 3, 2007, 1212-1229.

[66] Alvarez-Valdes R., Parreno F., Tamarit J.M., **"A Tabu Search algorithm for two dimensional non-guillotine cutting problems"**, European Journal of Operational Research , Vol.183(3), 2007, 1167-1182.

[67] Alvarez-Valdes R., Parajon, A., Tamarit, J.M. **"A computational study of heuristic algorithms for two-dimensional cutting stock problems"**, In: 4th Metaheuristics International Conference (MIC 2001), pp. 16–20 (2001)

[68] Lodi A., Martello S., Vigo D., **"Recent advances on two-dimensional bin packing problems"**, Discrete Applied Mathematics, Vol. 123, 2002, 379-396.

[69] Delorme M., M. Iori, S. Martello, **"Bin packing and Cutting Stock Problems: Mathematical models and exact algorithms"**, European Journal of Operational Research 2016, 255, 1–20, doi:10.1016/j.ejor.2016.04.030

[70] Wäscher, G.; Haußner, H.; Schumann, H. **"An Improved Typology of Cutting and Packing Problems"**, European Journal of Operational Research Volume 183, Issue 3, 1109-1130

[71] M.P. Johnson, C. Rennick and E. Zak **"Skiving addition to the cutting stock problem in the paper industry"**, SIAM Review, 472-483, (1997).

[72] Raffensperger, J. F. **"The generalized assortment and best cutting stock length problems"**, International Transactions in Operational Research. 17: 35–49, doi:10.1111/j.1475-3995.2009.00724.x., (2010).

[73] Kantorovich, V.L., **"Mathematical methods of organizing and planning production"**, Leningrad State University. 1939

[74] Kantorovich L.V. and Zalgaller V.A. **"Calculation of Rational Cutting of Stock"**, Lenizdat, Leningrad, (1951)

[75] Gilmore P.C., R.E. Gomory **"A linear programming approach to the cutting-stock problem"**, Operations Research 9: 849-859, (1961).

[76] Gilmore P.C., R.E. Gomory **"A linear programming approach to the cutting-stock problem - Part II"**, Operations Research 11: 863-888, (1963).

[77] Gradiar M., Kljaji M., Resinovi, G., Jesenko J. **"A sequential heuristic procedure for one-dimensional cutting"**, European Journal of Operational Research. 114. 3, 557-568, (1999).

[78] Gradisar M., Resinovic G., Kljajic, M. , **"A hybrid approach for optimization of one-dimensional cutting"**, European Journal of Operational Research. 119. 3, 719-728, (1999).

[79] Gradisar M., Resinovic G., Kljajic M. (2002) **"Evaluation of algorithms for one-dimensional cutting"**, Computers and Operations Research 29:1207–1220

[80] Dyckhoff H., **"A typology of cutting and packing problems"**, European Journal of Operational Research. 44, 145-159,(1990).

[81] Vance P., Barnhart, C., Johnson, E.L., Nemhauser, G.L. **"Solving binary cutting stock problems by column generation and branch-and-bound"**, Computational Optimization and Applications. 3. 111-130,(1994).

[82] Vance P., **"Branch-and-price algorithms for the one-dimensional cutting stock problem"**, Computational Optimization and Applications. 9, 211-228,(1998).

[83] Goulimis C. **"Optimal solutions for the cutting stock problem"**, European Journal of Operational Research 44: 197-208, (1990)

[84] de Carvalho V. **"Exact solution of cutting stock problems using column generation and branch-and-bound"**, International Transactions in Operational Research 5: 35–44, (1998)

[85] Berberler M.E., Nuriyev U.G., **"A New Heuristic Algorithm for the One-Dimensional Cutting Stock Problem"**. Appl. Compuy. Math. 9.1, 19-30,(2010).

[86] Diegel A., E. Montocchio, E. Walters, S. van Schalkwyk and S. Naidoo (1996), **"Setup minimizing conditions in the trim loss problem"**, European Journal of Operational Research 95:631-640

[87] McDiarmid C., **"Pattern Minimisation in Cutting Stock Problems"**, Discrete Applied Mathematics, 121-130, (1999)

[88] Garey M.R., Johnson, D.S., **"Computers and Intractability: A Guide to the Theory of NP-Completeness"**, WH Freemann, New York (1979).

[89] Kantorovich L.V., **"Mathematical methods of organizing and planning production"**, Management Science 6.4, 366-422,(1960).

[90] Jahromi, M.H., Tavakkoli-Moghaddam, R., Makui, A. Shamsi A. **"Solving an one-dimensional cutting stock problem by simulated annealing and tabu search"**, Journal of Industrial Engineering International, Vol. 8(1), Springer, 2012, paper 24.

[91] Jahromi, M.H., Tavakkoli-Moghaddam, R., Makui, A., Shamsi, A., **"Solving an one dimensionalcutting stock problem by simulated annealing and tabu search"**, Journal of Industrial Engineering International. (2012).

[92] **"Constantine Goulimis. Counterexamples in the CSP"**, arXiv:2004.01937

[93] Maria Garcia de la Banda, P. J. Stuckey., **"Dynamic Programming to Minimize the Maximum Number of Open Stacks"**, INFORMS Journal on Computing, Vol. 19, No. 4, Fall 2007, 607-617.

[94] Chvátal, V. **"Linear Programming"**, W.H. Freeman. ISBN 978-0-7167-1587-0, (1983).

[95] Cherri, A.C., Arenales, M.N., Yanasse, H.H., **"The one-dimensional cutting stock problems with usable leftover: a heuristic approach"**, European Journal of Operational Research. 196.3, 897-908,(2009).

[96] Cherri, A.C., Arenales, M.N., Yanasse, H.H., **"The usable leftover one-dimensional cutting stock problem-a priority-in-use heuristic"**, Intl. Trans. in Op. Res. 20, 189-199,(2013).

[97] Valerio de Carvalho, J.M., **"Exact solution of bin-packing problems using column generation and branch-and-bound"**, Annals of Operation Research. 86, 629-659, (1999).

[98] Hatem Ben Amor, J.M. Valério de Carvalho, **"Cutting Stock Problems in Column Generation"**, edited by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, Springer, 2005, XVI, ISBN 0-387-25485-4.

[99] Waescher, G., Gau, T., **"Heuristics for the Integer one-dimensional Cutting Stock Problem"**, A Computational Study. OR Spektrum18. 3, 131-144,(1996).

[100] Vanderbeck, F., **"Computational study of a column generation algorithm for binpacking and cutting stock problems"**, Mathematical Programming A. 86, 565-594, (1999).

[101] Vanderbeck, F., **"On DantzigWolfe decomposition in integer programming andways to perform branching in a branch-and-price algorithm"**, Operations Research. 48, 111-128,(2000).

[102] Mobasher, A., Ekici A., **"Solution approaches for the cutting stock problem with setup cost"**, Computers and Operations Research. 40, 225-235,(2013).

[103] Johnston, R.E., Sadinlija, E., **"A new model for complete solutions toone-dimensional stock problems"**, European Journal of Operational Research. 153, 176-183,(2004).

[104] Johnston, R.E. **"Rounding algorithm for cutting stock problems"**, Journal of Asian-Pacific Operations Research societies 3:166–171, (1986).

[105] Umetani, S., Yagiura, M., Ibaraki, T., **"One-dimensional cutting stock problem to minimize the number of different patterns"**, European Journal of Operational Research. 146, 388-402,(2003).

[106] S. Umetani, M. Yagiura, and T. Ibaraki **"One dimensional cutting stock problem to minimize the number of different patterns"**, European Journal of Operational Research 146, 388–402, (2003)

[107] Scheithauer, G., Terno, J., Muller, A., Belov, G., **"Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm"**, Journal of the Operational Research Society. 52, 1390-1401,(2001).

[108] Belov, G., Scheithauer, G., **"A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting"**, European Journal of Operational Research. 171, 85-106,(2006).

[109] Belov, G., Scheithauer, G., **"A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths"**, European Journal of OperationalResearch. 141, 274-294,(2002).

[110] Mukhacheva, E.A., Belov, G., Kartak, V., Mukhacheva, A. S., **"One-dimensional cutting stock problem: Numerical experiments with the sequential value correction method and a modified branch-and-bound method"**, Pesquisa Operacional. 2000.2, 153-168,(2001).

[111] Belov G., Scheithauer G. **"Setup and open stacks minimization in one-dimensional stock cutting"**, INFORMS Journal of Computing 19(1):27–35, (2007).

[112] Belov G., Scheithauer G. **"The number of setups (different patterns) in one-dimensional stock cutting"**, Technical Report, Desden University, (2003).

[113] Belov G., Scheithauer G. **"A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting"**, European Journalof Operational Research 171:85–106, (2006).

[114] Dikili A.C., Sarz E., PekN. A., **"A successive elimination method for one-dimensional stock cutting problems in ship production"**, Ocean engineering. 34.13, 1841-1849, (2007).

[115] Reinertsen H., Vossen Thomas W.M., **"The one-dimensional cutting stock problem with due dates"**, European Journal of Operational Research. 201.3, 701-711, (2010).

Abstracts of Dissertations

Number 8, 2022

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES

БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ
ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Брой 8, 2022

Автореферати на дисертации