CSC8499 Individual Project: System for area planning

Fedor Shmarov

MSc in Advanced Computer Science,
School of Computing Science, Newcastle University.
f.shmarov@ncl.ac.uk

Abstract. Nowadays there is a demand in the efficient use of territory. This dissertation discusses main issues connected to the problem of area planning. There are several techniques used for dealing with the different aspects of area planning. However, there is a lack of planning applications delivering the desired functionality. The main aim of the project is to develop such a system. The formal statement of the problem investigated in the project is presented and the reduction of one of the existing problems is performed. The motivation for representing the problem mathematically is stated and main steps and issues faced during the modeling process are discussed in details. The system for area planning meeting all specified requirements is implemented, tested and results are evaluated.

Declaration: I declare that this dissertation represents my own work except where otherwise explicitly stated.

1 Introduction

Due to the city growth there is a need to develop new territories rationally and make a more efficient reuse of the existing ones. Urban planning is a paradigm dealing with these issues. The process of urban planning has several aspects such as district planning, infrastructure and transportation networking [19]. The importance of optimal planning is very high because it can reduce the impact on the environment (for example, by efficient road planning) and increase the profit of the construction companies by allocating more objects in the area, thus utilizing space more efficiently. In the scope of the project, issues connected to area planning are considered.

Area planning can be defined as a process of finding optimal disposition of construction units (such as buildings and roads) inside the predefined area in order to achieve some of the objectives (for example, to maximize the number of housing units on a site or minimize the total road length). Objectives are defined depending on particular needs.

Currently, the problem seems to be very difficult as all the planning is done by human beings. The performance and accuracy of calculations made by people are low, and the result cannot be considered optimal. Therefore, the importance of using modern planning systems and techniques for solving optimization problems is very high. Area planning is one of the fields where computing capacities can be applied.

Nowadays in the industry there is a lack of software performing area planning. Hence, there is a demand in such systems in the market.

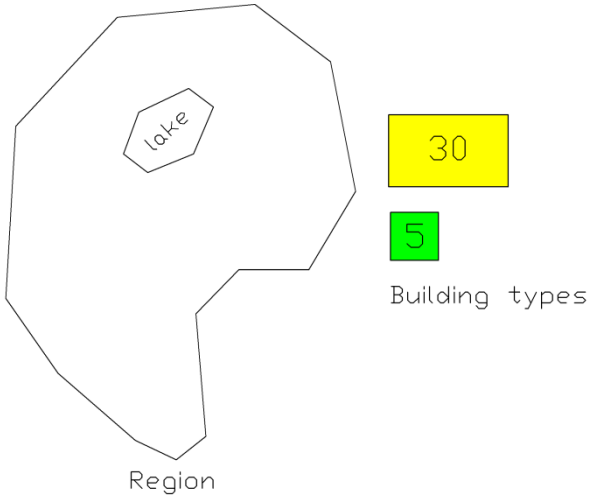

Figure 1. Predefined region and building types.

The problem investigated in the project is illustrated by the example in Figure 1. There is a land region defined geographically and building types. Each building type is characterized by its dimensions and a positive *benefit* (e.g. a sale price of the building). The total benefit of the construction is a sum of benefits of all the buildings in the construction site. The problem can be stated as computing the disposition of buildings in the given area maximizing total benefit.

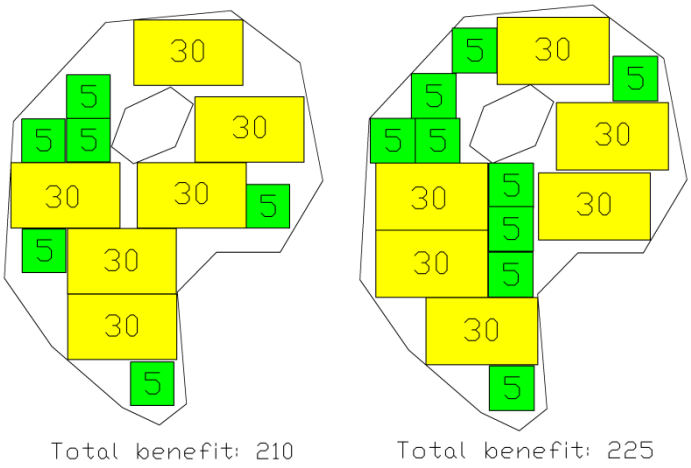

Figure 2. Examples of problem solution

The problem is rather complicated, and the optimal solution is often not obvious. Figure 2 demonstrates two solutions. Intuitive solution of the problem is presented on the left-hand side of the picture. It can be seen that there is no free space sufficient to fit a building, and the total benefit of the construction is 210. Nevertheless, there is solution with a higher total benefit which demonstrates that the previous one is not optimal. Therefore, more sophisticated techniques than just intuition should be applied to the problem.

It is easy to show that the decision version of this optimization problem (finding a solution with the benefit not less than a given number) is NP-complete (it will be demonstrated in Section 2.2).

Several approaches to solving the problem were considered (Section 2.3) and it was decided to present the problem as a Mixed Integer Programming problem [2] and use an *"off-the-shelf"* solver [9].

Now aim and objectives of the project can be stated.

1.1 Aim and objectives

The aim of the project is to develop a software performing maximization of total benefit having the predefined region and building types as input parameters and outputting graphical representation of the region with the buildings placed inside it.

Objectives:

- Evaluate existing software and modern techniques for the area planning problem
- Make a reduction of existing problem to the investigated problem
- Evaluate existing tools and methods for solving reduced problem
- Implement the system in Java
- Perform testing of the system

Personal objectives:

- Understand the paradigm of solving difficult problems by reducing them to a standard ones and using an off-the-shelf solver.

1.2 System requirements

This subsection states functional and non-functional requirements which were applied to the developing system.

Functional requirements give an answer for the question "what system should do" while non-functional requirements define characteristics of the whole system [17]. System requirements are described in Table 1.

Table 1. System requirements

Requirement	Description
F1	System should provide a graphical user interface implementing windows for inputting data and outputting the result
F1.1	Main window should contain a map allowing input of the region as a sequence of points in a counter clockwise direction and user should be able to specify the main region and excluded areas.
F1.2	The map created for an input should be suitable for displaying the result of optimization.
F1.3	Main window should contain a panel reflecting the progress of optimization, current value of objective function, number of solutions found and lower and upper bound for the binary search mode.
F1.4	Main window should contain a panel displaying building types involved into the optimization process.
F1.5	Main window must have buttons for manipulating the optimization process allowing user to start and terminate optimization process, save the result of optimization as an image. Also user should be allowed to reduce the upper bound of objective function without terminating the process.
F1.6	There should be a window for setting building types containing fields for inputting name, width, length, benefit and color of the building type
F1.7	There should be a window for settings allowing user to set a time limit for iteration in binary search, lower bound for the total benefit and checkbox enabling and disabling binary search mode.
F2	GUI should reply with an error message if any of input data is invalid:
F2.1	If entered polygon is not convex
F2.2	If user attempted to create two building types with the same name
F2.3	If width, length or benefit of the building less or equal to 0
F3	Application should not block during the optimization process, and intermediate solutions should be displayed on the map as they are found.
N1	System should not crash during the execution

1.3 Project plan

The work on the project started on 22.04.2013 and a plan containing several stages with the deadlines was developed (Figure 3).

During the first stage of the work research into existing software is performed. Upon the completion of this stage there should be made a conclusion about whether there exists a system meeting all the requirements of the project or not. If the evaluation shows the absence of such a system, an application for area planning should be developed.

The second stage of the project requires reduction of one of existing problems to the investigated problem. After that existing methods and techniques of solving the reduced problem should be evaluated. Upon the end of the stage a decision about the approaches to the problem should be made.

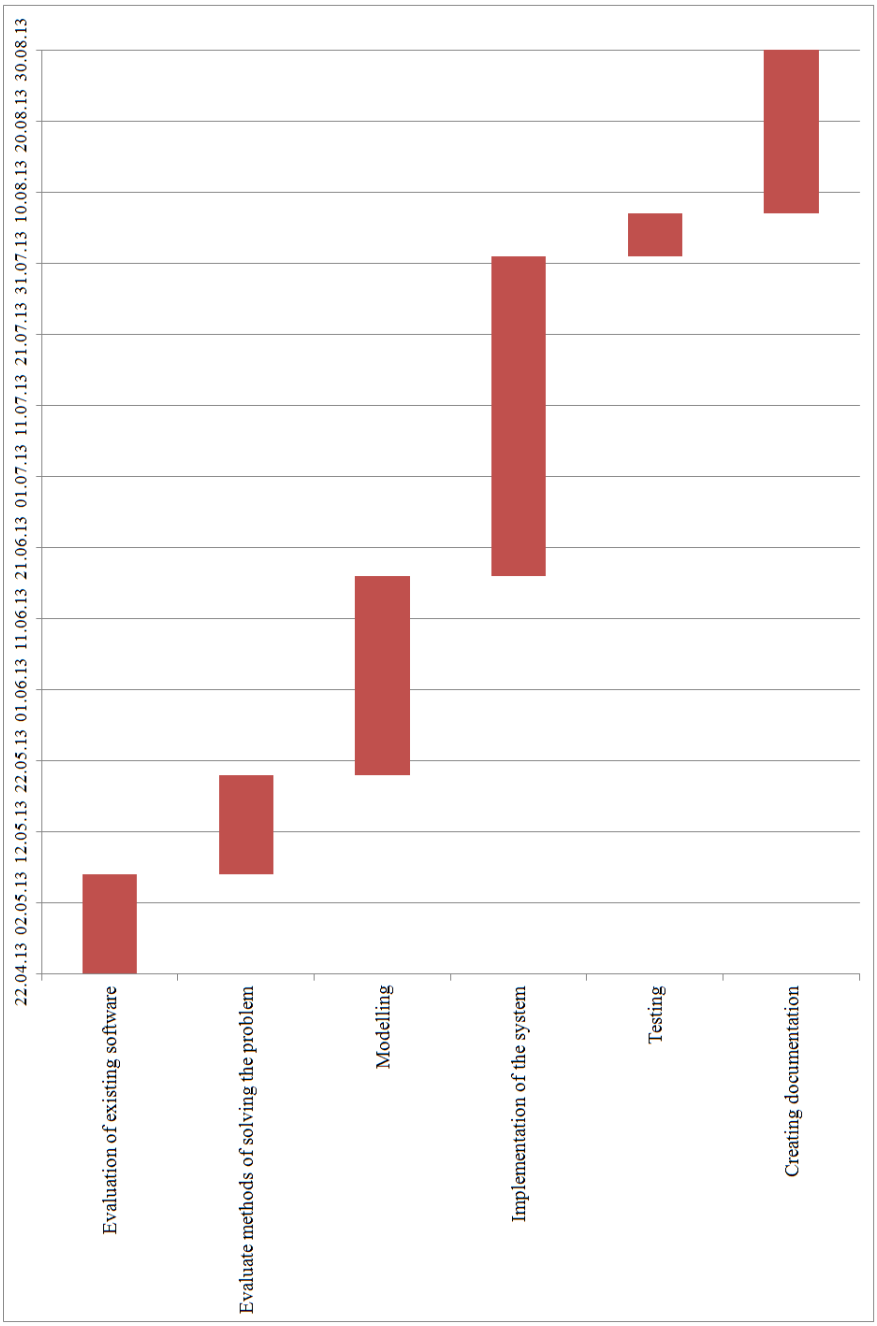

Figure 3. Project plan.

The aim of the third stage is to formulate the problem as Mixed Integer Programming problem [2]. The result of the stage is mathematical model with the core constraints. All the enhancements to the basic model will be provided during the implementation stage in accordance with the incremental software development model [17] (Section 2.5).

When the basic model is built and a method of optimizing the model is chosen the next stage can be started. According to the plan, it is the implementation of the system. The result of this stage is the system for area planning implemented in Java language. After that the system is built it is tested and evaluated.

The final stage of the project is creating documentation for the project and undertaken work.

1.4 Risk management

Risks through all the stages of the project were rated and defined as high, moderate and low risk tasks.

Modelling is the most research-intensive period and therefore it was considered as a high risk task. The main risk on this stage is a failure to create a basic model which can cause a failure of the planned process. In the case of failure, other approach to solving the problem should be chosen and project plan should be reevaluated.

Implementation of the system was marked as a low risk task. Design and implementation of the system is considered to be trivial and no major risks are associated with the process. As it was stated enhancements to the basic model should be introduced during this step. The main risk on this stage is a failure to provide enhancements to the model. Nevertheless, it is not crucial and in the case of failure system with core functionality will be developed.

All the other stages of the project were evaluated as low risk tasks as they do not make a big influence on the whole process and none the high risks can appear throughout these steps.

Also there are several general risks considered as moderate. For example, in the case of sickness only basic functionality will be implemented, or if a supervisor is on a vacation a flexible schedule of meetings will be created.

1.5 Dissertation structure

The whole article is divided into several sections. In section 2 (Background) background research and motivation for using different techniques is discussed. Section 3 (Modeling) contains detailed mathematical formulation of the problem. Section 4 (Optimization) introduces enhancements made to the model to improve performance of the system. Section 5 (Design) and Section 6 (Implementation) provides details about the design and implementation of the system. Section 7 (Testing) and Section 8 (Evaluation) contains results of system execution and evaluation of requirements of the system. Section 9 (Conclusions and Future Work) completes the article with a

conclusion and discussion about enhancement which can be applied to the system in future.

2 Background

This section presents background research performed before the development of the system started. It contains information about the evaluated software for area planning, brief introduction into NP-complete problems [7], methods of solving the problem, evaluation of existing *"off-the-shelf"* solvers and considered software development models [17].

2.1 Existing Software

Several systems for the area planning problem were evaluated.

SmartDraw [16] and Autodesk InfraWorks [1] are ones of the major systems in market. They are powerful design and visualization tools. These systems allow users to specify the region and define construction units (such as roads, buildings). Moreover, they provide three-dimensional representation of the construction site.

Another system which was evaluated is SITEOPS [15]. The system is more sophisticated than those mentioned previously. Apart from the functionality described it provides users with several optimization features (for example, creating optimal piping system or road layout for the existing infrastructure). Nevertheless, it is required that all prior area planning is performed beforehand by users.

It was identified that none of existing systems meet all the requirements and satisfy them only to some extent. Considered software does not deliver the desired optimization functionality. Therefore, the motivation for the development of the system for area planning is now clarified.

2.2 NP-completeness

The problem is NP-complete if it is in NP complexity class and any of the existing problems in NP can be reduced to the presented problem [7].

Problem belongs to the NP complexity class if it is possible to find a candidate to be a solution for the problem and this solution can be verified in polynomial time [11]. Investigated problem belongs to the NP complexity class because it is possible to find a set of buildings matching particular total benefit and it can be verified in polynomial time whether these buildings can be placed in the region satisfying all the constraints.

Decision version of the problem can be defined as finding solution for the particular benefit. Therefore, problem presented in the project requires solving a sequence of decision problems for all possible total benefits. The decision version of the problem can be defined as a problem of packing arbitrary rectangles inside arbitrary polygons without overlapping. Formulated problem can be obtained performing reduction of

the problem of two dimensional bin packing. The decision version of this problem belongs to the class of NP-complete [13].

It was shown that problem is in NP complexity class and NP-complete problem of two dimensional bin packing can be reduced to the presented problem. Hence, investigated problem is NP-complete.

2.3 Approaches to problem solution

Investigated problem was formulated as a packing problem. Several approaches to solving the problem were considered: packing algorithm, non-linear optimization and mixed integer optimization.

Packing algorithm. Packing algorithm proposes application of existing methods of solving packing problem [3]. This approach can be beneficial as the algorithm specialized for solving particular problem can be optimized and tuned to increase its accuracy and performance. Nevertheless, disadvantages of this approach outweigh its advantages. The main drawback of the algorithm is coming from its narrow specialization which makes the system built on the packing algorithm highly inflexible and inextensible. Moreover, any improvements and additions to the initial problem can cause reconstruction of the whole algorithm. Finally, there is a lack of existing implementations suitable for this problem. Considering all pros and contras the use of packing algorithm was rejected and further research was undertaken.

Another approach is to formulate the problem mathematically and use tools for solving mathematical problem.

Nonlinear optimization. The idea of the method is to represent the total benefit as a function and all the constraints in the form of linear and non-linear equations and inequalities [4]. Solving packing problem now can be reduced to finding solution for non-linear optimization problem. The approach is more beneficial in comparison to packing algorithms. Mainly, the model formulated mathematically is more flexible and extensible as any additions to the problem only requires changes into the model (adding or removing constraints from the model) without changing the technique of solving the problem. Moreover, it allows reusing of existing tools instead of implementing new ones, and code reusing was chosen as a fundamental approach because of the benefits it has. First of all, it allows focusing on high-level reduction rather than low-level improvements. Moreover, current solvers are more powerful than naive low-level implementations. Finally, solvers are well debugged and supported by developers and it is possible to benefit from updates to the solver.

Nevertheless, nonlinear approach was denied as there is a lack of nonlinear solvers capable of handling models with a big number of variables and constraints.

Mixed Integer optimization. The idea of the approach is similar to non-linear representation of the problem with the only difference that constraints and objective func-

tion are linear and some variables can be discrete [2]. Evaluation of tools for solving MIP problem demonstrated a presence of many efficient and scalable solvers in the industry. Having the motivation for using existing tools in mind, it was decided to use one of existing solvers, and analysis of modern mixed integer solvers was performed.

2.4 Evaluation of existing MIP solvers

The first solver which was observed was Microsoft Excel inbuilt solver. The performance and scalability of the solver is low and it was observed on the stage of model formulation in order to identify its correctness. According to the results of the testing, several corrections to the problem formulation were introduced.

A couple of non-commercial solvers such as COBYLA2 [5] and lp_solve [12] were tested. Both solvers are used for linear programming and mixed integer programming problems. The results of the testing were not satisfactory. In comparison to the Microsoft Excel solver they demonstrated much higher performance, but still suffered from a lack of scalability.

Also there were considered two commercial solvers: CPLEX [6] and Gurobi Optimizer [9] (both of them are currently leading in the industry). They demonstrated much higher performance and scalability (up to one million of variables and constraints) than described non-commercial solvers. Comparison of two solvers demonstrated the advantage of Gurobi Optimizer over CPLEX solver in performance (factor 1.5X). Therefore, Gurobi Optimizer was accepted to be used in the system.

2.5 Software development approach

Several software development approaches were considered. The most focus was concentrated on the *waterfall* [14] and *incremental* models [17].

Waterfall model. Waterfall software engineering approach represents the development of the application as a sequence of separate stages (requirements definition, system and software design, implementation and unit testing, integration and system testing, operation and maintenance) [14, 17]. This approach is highly beneficial when the requirements are strict and well understood. Essentially, upon the completion of one step there cannot be any significant changes introduced to the previous ones. This model is suitable to the systems which are straightforward and which implementation is trivial (for example, development of the web page).

Nevertheless, requirements for the developing system are not strict and there is only a core set of requirements, and extra requirements are added as the system is being developed. Therefore, it requires introducing changes to the model and system design. These are significant improvements and they cannot be made in the scope of waterfall model. Hence, waterfall model was rejected and another approach was considered.

Incremental model. The idea of incremental approach is to provide initial implementation and develop the system through the series of incremental steps [17]. This ap-

proach is more natural as it demonstrates the way all the problems are usually solved. As it was mentioned earlier, the system contains a set of the most urgent requirements which must be met in the first version. All the other requirements can be formulated as the system is being implemented and they should be met in later versions. As it will be demonstrated later incremental approach fits the system better than the waterfall model. It also will be shown that some significant changes to the model are performed (for example, Section 3.5 and Section 4) in the scope of this software engineering model. Hence, incremental software development model was chosen as the best option for the system.

3 Modelling

This section will describe main steps to present the investigated problem as a MIP problem and issues solved during building the model. It provides mathematical representation of non-overlapping and boundary constraints and extensions to the boundary constraints. Also objective function will be formulated and binding of the function to the constraints will be performed. Finally, formal statement of the problem will be presented.

In order to simplify the problem, some relaxations were imposed on the area and objects, such as area should be convex and objects should be rectangular and cannot be rotated.

3.1 Non-overlapping constraints.

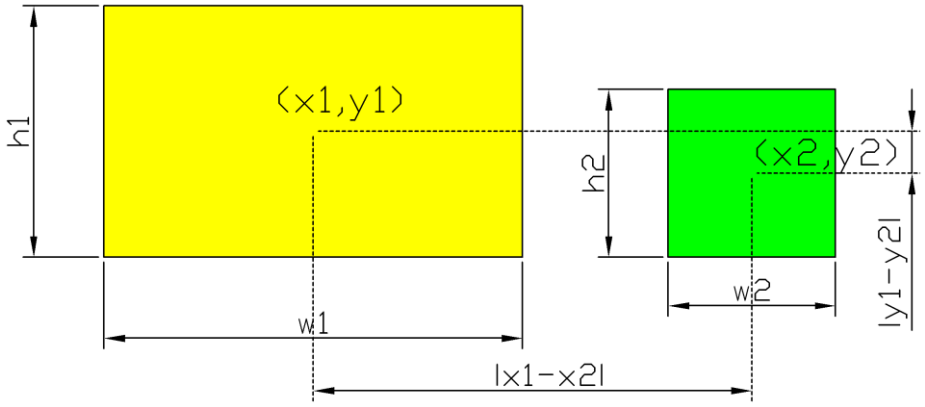

Figure 4. Example of disposition of two buildings.

Non-overlapping constraint states that each pair of buildings do not overlap. Having in mind that all the buildings have rectangular shape (Figure 4) non-overlapping constraint can be represented as a system:

$$\begin{bmatrix} |x_1 - x_2| \geq \frac{w_1+w_2}{2} \\ |y_1 - y_2| \geq \frac{h_1+h_2}{2} \end{bmatrix} \quad (1)$$

The system (1) is presented as a disjunction of two inequalities [4]. Therefore, it has a solution if at least one of them is satisfied.

According to the rules set by the solver provider on the model representation, all the constraints should be presented in a conjunctive form. Next subsection will describe main steps of transforming disjunction of inequalities into conjunction.

3.2 Eliminating disjunction

The first step is to eliminate modulus. It is a trivial procedure and it does not require any additional explanation. The system with removed modulus can be presented as following:

$$\begin{bmatrix} -x_1 + x_2 \leq -\frac{w_1+w_2}{2} \\ x_1 - x_2 \leq -\frac{w_1+w_2}{2} \\ -y_1 + y_2 \leq -\frac{h_1+h_2}{2} \\ y_1 - y_2 \leq -\frac{h_1+h_2}{2} \end{bmatrix} \quad (2)$$

The second step is to transform the system (2) to a conjunctive form. Special technique (also known as *big-M* method) was applied to the system of inequalities [8]. The idea of the method is to introduce new binary variables $z_i = \{0,1\}$ and multiply them by a big positive arbitrary number $M \gg \max(w_{max}, h_{max})$. Intuitively, if $z_i = 0$ then $Mz_i = 0$ then inequality is presented in the initial form and should be satisfied. In other words, there should be found a pair of continuous variables (x_1, x_2) or (y_1, y_2) satisfying the inequality. If $z_i = 1$ then $Mz_i = M$ and inequality is *"relaxed"* and it is easier to be satisfied using the pair of variables found for the inequality containing $z_i = 0$. Setting the restriction $z_1 + z_2 + z_3 + z_4 \leq 3$ it can be seen that there should be at least one variable $z_i = 0$. Hence, only one *"strict"* inequality should be satisfied and the rest of them will be also satisfied.

The non-overlapping constraint in conjunctive form is represented as a system:

$$\begin{cases} -x_1 + x_2 - \mathrm{M}z_1 \leq -\frac{w_1+w_2}{2} \\ x_1 - x_2 - \mathrm{M}z_2 \leq -\frac{w_1+w_2}{2} \\ -y_1 + y_2 - \mathrm{M}z_3 \leq -\frac{h_1+h_2}{2} \\ y_1 - y_2 - \mathrm{M}z_4 \leq -\frac{h_1+h_2}{2} \\ z_1 + z_2 + z_3 + z_4 \leq 3 \end{cases} \quad (3)$$

Now the constraint is in conjunctive form and can be used in the mathematical model and applied to each pair of buildings.

3.3 Region approximation

Any region can be approximated linearly with a high precision. Increasing a number of lines more accurate representation of the region can be achieved. Therefore, initial region can be presented as a polygon (Figure 5).

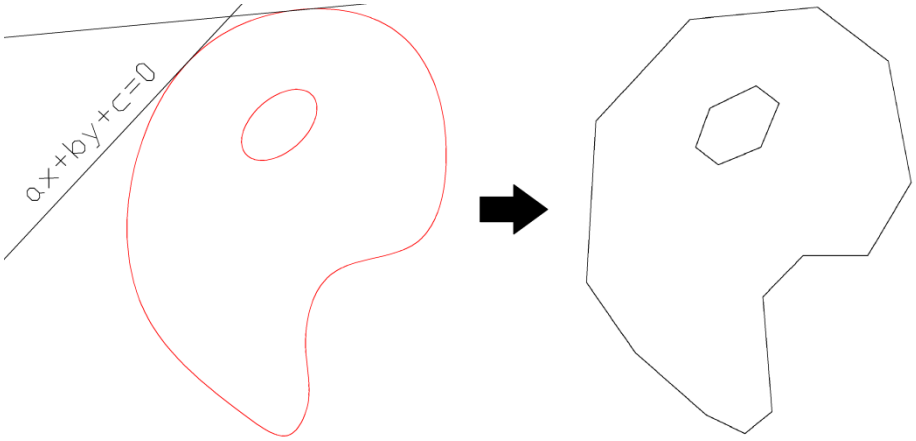

Figure 5. Linear approximation of region.

Each line approximating the region can be presented as an equation:

$$ax + by + c = 0 \tag{4}$$

In this section only convex polygons will be considered. It can be seen from Figure 5 that polygon approximating the primary region is not convex. Therefore, a simplification will be introduced to the initial polygon and it will be presented as a convex polygon with removed inner parts (Figure 6).

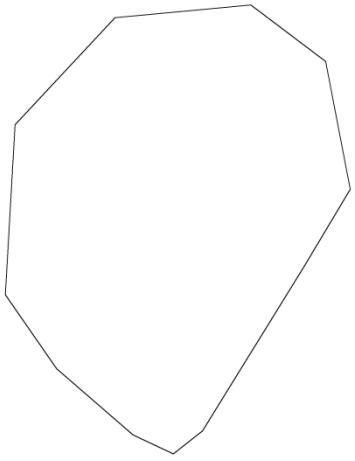

Figure 6. Convex polygon with removed concave parts.

This simplification will be disposed later and more complex case will be discussed in Section 3.5.

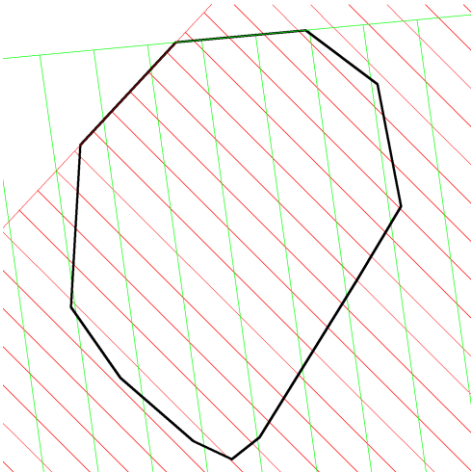

Figure 7. Polygon presented as an intersection of areas.

Space inside the convex polygon can be presented as an intersection of areas created by each line forming the polygon (Figure 7). Each of these areas can be represented as an inequality:

$$ax + by + c \leq 0. \tag{5}$$

Any inequality $ax + by + c \geq 0$ can be transformed to $-ax - by - c \leq 0$ and inequality (5) is used as a general form.

Convex polygon in Figure 7 can be represented mathematically as an intersection of solutions of inequalities (5):

$$\begin{cases} a_1 \mathrm{x} + b_1 \mathrm{y} + c_1 \leq 0 \\ \ldots \\ a_n \mathrm{x} + b_n \mathrm{y} + c_n \leq 0 \end{cases} \tag{6}$$

3.4 Boundary constraints

Boundary constraints state that all the buildings are placed inside the allowed region and no buildings lie in the restricted areas.

Keeping in mind that all the buildings are of rectangular shape, the first part of the boundary constraints describing the permitted area can be reformulated as: all vertices of the buildings should be inside the polygon approximating the area. Possessing information about the coordinates of the building's centre, its length and width, coordinates of all the vertices can be calculated using the formula [4]:

$$
\begin{aligned}
x_{lb} &= x - \frac{w}{2} \\
y_{lb} &= y - \frac{h}{2} \\
x_{lt} &= x - \frac{w}{2} \\
y_{lt} &= y + \frac{h}{2} \\
x_{rt} &= x + \frac{w}{2} \\
y_{rt} &= y + \frac{h}{2} \\
x_{rb} &= x + \frac{w}{2} \\
y_{rb} &= y - \frac{h}{2}
\end{aligned}
\tag{7}
$$

Indexes lb, lt, rt, rb in (5) stand for the left-bottom, left-top, right-top and right-bottom vertices of the building respectively (Figure 8).

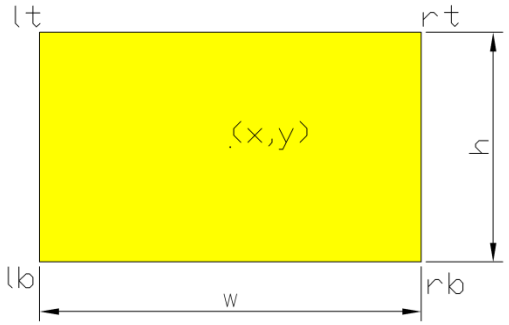

Figure 8. Labelling vertices of the buildings.

Hence, boundary constraints can be represented as a system of inequalities applied to all the vertices of each building:

$$
\begin{cases}
a_1 \mathrm{x}_{lb} + b_1 \mathrm{y}_{lb} + c_1 \le 0 \\
\dots \\
a_1 \mathrm{x}_{rb} + b_1 \mathrm{y}_{rb} + c_1 \le 0 \\
\dots \\
a_n \mathrm{x}_{lb} + b_n \mathrm{y}_{lb} + c_n \le 0 \\
\dots \\
a_n \mathrm{x}_{rb} + b_n \mathrm{y}_{rb} + c_n \le 0
\end{cases}
\tag{8}
$$

3.5 Extension

Model containing non-overlapping and boundary constraints can be considered as basic. In order to allow of non-convex and excluded polygons an extension to the initial model was introduced. In a simple case any non-convex polygon can be presented as a combination of convex polygons (Figure 9). It can be seen from the figure that approximated polygon with excluded areas can be found as a difference between initial and excluded polygons. In the same manner, each excluded area can be presented as a combination of convex polygons.

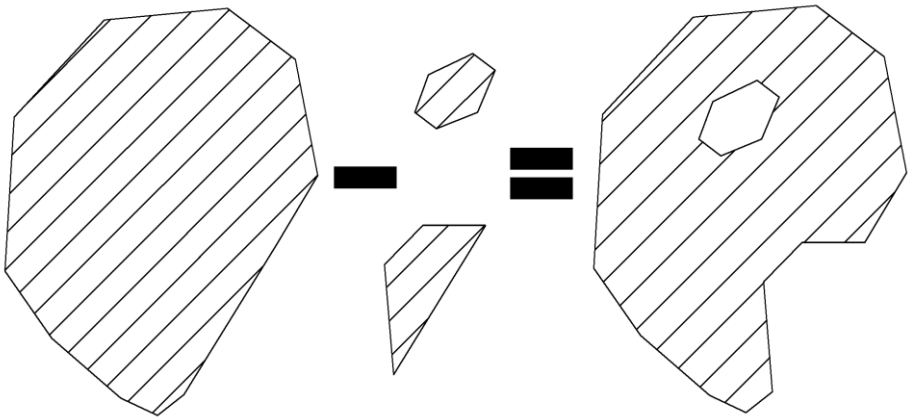

Figure 9. Representation of complex polygons.

In a simple case, if excluded area is presented as convex polygon, space outside it can be presented as a union of areas created by each line of the polygon. Therefore, excluded convex polygon can be represented mathematically as a system:

$$\begin{bmatrix} a_1\mathrm{x} + b_1\mathrm{y} + c_1 \geq 0 \\ \dots \\ a_n\mathrm{x} + b_n\mathrm{y} + c_n \geq 0 \end{bmatrix} \tag{9}$$

Polygon is presented as a disjunction of inequalities. In order to solve this issue technique described in (3) is applied. Using the transformation introduced in Section 3.2 excluded polygon can be represented as:

$$\begin{cases} a_1\mathrm{x} + b_1\mathrm{y} + c_1 + Mz_1 \geq 0 \\ \dots \\ a_n\mathrm{x} + b_n\mathrm{y} + c_n + Mz_n \geq 0 \\ \sum_{i=1}^{n} z_i \leq n - 1 \end{cases} \tag{10}$$

According to the second part of definition of boundary constraints, all the buildings should be outside restricted (excluded) areas. It can be reformulated as: all the vertices of the buildings should satisfy the system (10):

$$\begin{cases} a_1\mathrm{x}_{lb} + b_1\mathrm{y}_{lb} + c_1 + Mz_1 \geq 0 \\ \dots \\ a_1\mathrm{x}_{rb} + b_1\mathrm{y}_{rb} + c_1 + Mz_1 \geq 0 \\ \dots \\ a_n\mathrm{x}_{lb} + b_n\mathrm{y}_{lb} + c_n + Mz_n \geq 0 \\ \dots \\ a_n\mathrm{x}_{rb} + b_n\mathrm{y}_{rb} + c_n + Mz_n \geq 0 \\ \sum_{i=1}^{n} z_i \leq n - 1 \end{cases} \tag{11}$$

Intuitively, regarding system (11), all vertices of the building should be placed aside from one of the lines forming excluded polygon. Nevertheless, (11) is not the

case and it will be demonstrated in Section 3.6 that some additions to the system are required.

3.6 Addition to boundary constraints

Figure 10 illustrates a simple situation. There is an excluded area on the left-hand side of the figure and a building on the right-hand side. Shaded area represents the allowed area formed by the polygon.

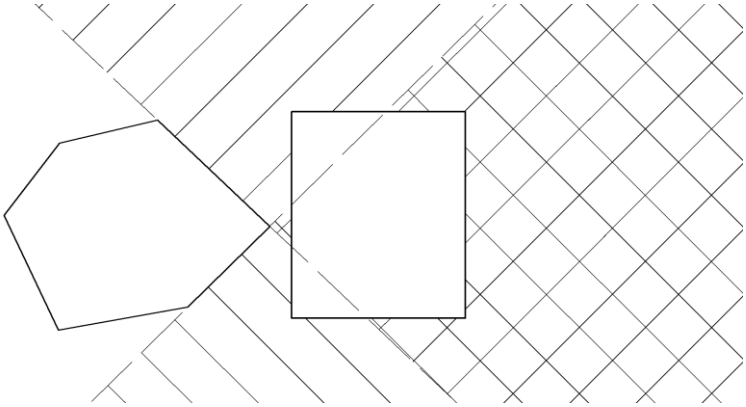

Figure 10. Possible disposition of building and excluded area.

It can be seen from the figure, that the second part of boundary constraint regarding excluded areas is satisfied because the building lies outside the restricted area. Nevertheless, it can be seen that constraint (11) is not satisfied, because not all vertices of the building are placed on one side from the line. Therefore, the system (11) should be improved.

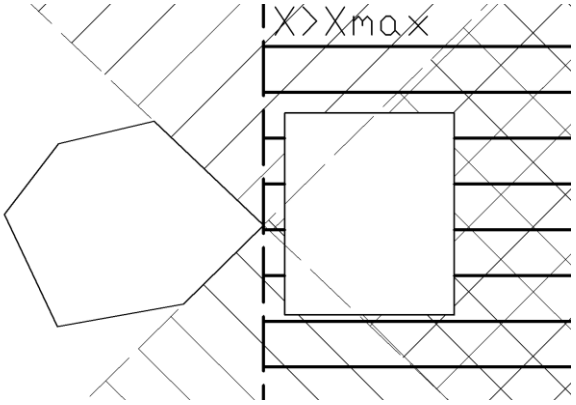

Figure 11. Introducing new constraints.

In order to combat this issue, extra constraints should be added to the current model. The way of solving the problem is presented graphically in Figure 11.

It can be seen that new constraint presented as $x \geq x_{max}$ was introduced. x_{max} in the inequality is maximal coordinate x of the excluded area. Similarly constraints on maximal and minimal coordinates of the excluded polygon should be imposed:

$$\begin{bmatrix} x \geq x_{max} \\ x \leq x_{min} \\ y \geq y_{max} \\ y \leq y_{min} \end{bmatrix} \tag{12}$$

Values $x_{max}, x_{min}, y_{max}, y_{min}$ are maximal and minimal coordinates of vertices of the excluded polygon. The system should be satisfied by coordinates of all the vertices of the buildings. The issue presented in this section can be solved by adding system (12) to (11). Now it can be guaranteed that the second part of boundary constraint is satisfied if at least one inequality has a solution. The way of combating disjunction in (12) is the same as presented in (3). Therefore, boundary constraints can be presented as a system:

$$\begin{cases} a_1 x_{lb} + b_1 y_{lb} + c_1 + M z_1 \geq 0 \\ \dots \\ a_1 x_{rb} + b_1 y_{rb} + c_1 + M z_1 \geq 0 \\ \dots \\ a_n x_{lb} + b_n y_{lb} + c_n + M z_n \geq 0 \\ \dots \\ a_n x_{rb} + b_n y_{rb} + c_n + M z_n \geq 0 \\ x_{lb} + M z_{n+1} \geq x_{max} \\ \dots \\ x_{rb} + M z_{n+1} \geq x_{max} \\ -x_{lb} + M z_{n+2} \geq -x_{min} \\ \dots \\ -x_{rb} + M z_{n+2} \geq x_{min} \\ y_{lb} + M z_{n+3} \geq y_{max} \\ \dots \\ y_{rb} + M z_{n+3} \geq y_{max} \\ -y_{lb} + M z_{n+4} \geq -y_{min} \\ \dots \\ -y_{rb} + M z_{n+4} \geq -y_{min} \\ \sum_{i=1}^{n+4} z_i \leq n + 4 - 1 \end{cases} \tag{13}$$

3.7 Objective function

Objective function is a cost function defining the total benefit of the construction on the site and it should present all possible combinations of all the benefits of the buildings. Building in the function can be presented as a binary variable multiplied by the

benefit of the building, and objective function can be presented as a sum of products of binary variables and benefits.

In order to define all possible combinations of the buildings and consequently formulate the function the maximal number of buildings of each type should be calculated. It can be found simply dividing the area of the main polygon by the area of the building of particular type as it is presented in the formula:

$$n_i^{max} = \left[\frac{A_{pol}}{A_i}\right] \tag{14}$$

In presented formula A_i defines the area of the building of type i, A_{pol} is the area of the polygon. It is not the case that n_i^{max} buildings of type i should be presented in the solution. Nevertheless, it is guaranteed that $n_i^{max} + 1$ buildings of type i cannot be a solution to the problem.

Area of the building can be found as a product of its width and length because all the buildings are rectangular. In order to calculate area of the polygon it should be presented as a sequence of points entered in particular order. Area of the polygon entered in a counter clockwise direction can be calculated using the formula:

$$A_{pol} = \frac{\sum_{i=1}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) + x_n y_1 - x_1 y_n}{2} \tag{15}$$

$x_i y_i$ are vertices of the polygon and n is a number of vertices in the polygon.

The cost function defining combination of the buildings can be presented as a sum of benefits of all the buildings of all the types:

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i^{max}} n_{ij} B_i \tag{16}$$

In the formula $n_{ij} = \{0,1\}$ is a binary variable defining building j of type i, $B_i > 0$ is a benefit of the building type i and m is a number of building types.

Now when the objective function is formulated it can be seen that binary variables defining buildings in the function are not connected to the continuous variables in the constraints. The way of providing this binding is discussed in the next subsection.

3.8 Modified constraints

Binary variables n_{ij} of the objective function and continuous variables $x_i y_i$ from the constraints are not bound. Some modifications were made to combat this issue. Constraints were modified in order to set dependence between objective function and constraints. Intuitively, if $n_{ij} = 1$ then building is considered to be in the solution and therefore, it should satisfy all the constraints. Otherwise, if $n_{ij} = 0$ then constraints for this building can be relaxed as the building defined by this binary variable does not contribute into the final solution.

As a result, non-overlapping constraints can be presented as:

$$\begin{cases} -x_{ij} + x_{kl} - M_1 z_1 - M_2 n_{ij} - M_2 n_{kl} \le 2M_2 - \frac{w_{ij}+w_{kl}}{2} \\ x_{ij} - x_{kl} - M_1 z_2 - M_2 n_{ij} - M_2 n_{kl} \le 2M_2 - \frac{w_{ij}+w_{kl}}{2} \\ -y_{ij} + y_{kl} - M_1 z_3 - M_2 n_{ij} - M_2 n_{kl} \le 2M_2 - \frac{h_{ij}+h_{kl}}{2} \\ y_{ij} - y_{kl} - M_1 z_4 - M_2 n_{ij} - M_2 n_{kl} \le 2M_2 - \frac{h_{ij}+h_{kl}}{2} \\ z_1 + z_2 + z_3 + z_4 \le 3 \end{cases} \quad (17)$$

It can be seen, that new arbitrary number $M_2 \sim M_1$ was introduced. Intuitively, if building is not considered to be in the solution, the coordinates of the building's centre can be arbitrary. Otherwise, buildings should be placed without overlapping and strict constraints should be satisfied.

Respectively, boundary constraints were reformulated in the same manner.

$$\begin{cases} a_1 x + b_1 y + c_1 - M_2(1 - n_{ij}) \le 0 \\ \dots \\ a_l x + b_l y + c_l - M_2(1 - n_{ij}) \le 0 \end{cases} \quad (18)$$

Intuitively, if building is not in the solution, coordinates of the building centre can be out of the region's bounds.

The same technique is applied to the constraints defining excluded regions:

$$\begin{cases} a_1 x + b_1 y + c_1 + M_1 z_1 + M_2(1 - n_{ij}) \ge 0 \\ \dots \\ a_l x + b_l y + c_l + M_1 z_l + M_2\left(1 - n_{ij}\right) \ge 0 \\ x + M_1 z_{l+1} + M_2(1 - n_{ij}) \ge x_{max} \\ -x + M_1 z_{l+2} + M_2(1 - n_{ij}) \ge -x_{min} \\ y + M_1 z_{l+3} + M_2(1 - n_{ij}) \ge y_{max} \\ -y + M_1 z_{l+4} + M_2(1 - n_{ij}) \ge -y_{min} \\ \sum_{i=1}^{l+4} z_i \le l + 4 - 1 \end{cases} \quad (19)$$

3.9 Problem representation

All types of constraints and objective function were formulated. Therefore, the problem can be stated formally:

$$Maximize: \sum_{i=1}^{m} \sum_{j=1}^{n_i^{max}} n_{ij} B_i \quad (20)$$

with the constraints (17), (18), (19).

4 Optimization

This section describes enhancements introduced to the model to decrease a number of candidates to be a solution and therefore, increase performance of the system. They are breaking symmetry and binary search.

4.1 Breaking symmetry

A simple scenario can be introduced to describe the approach of symmetry breaking. For example, there is only one building type and the maximal number of building of this type equals three. Possible combinations containing one building are: 001, 010 and 100. These solutions have the same total benefit. Hence, only one of them should be considered.

The idea of symmetry breaking is to impose constraints on binary variables presented in the objective function in order to reduce a number of considered solutions. These constraints can be presented as following:

$$\begin{cases} n_{11} \le n_{12} \le \cdots \le n_{1n_1^{max}} \\ \cdots \\ n_{m1} \le n_{m2} \le \cdots \le n_{mn_m^{max}} \end{cases} \tag{21}$$

It can be seen from (21) that only one candidate (001) will satisfy constraint and all the others will be eliminated.

4.2 Binary search

Binary search is another improvement to the model. The main idea of the approach is to apply standard binary search method to the value of the objective function. The highest total benefit of the construction can be calculated and therefore, lower and upper bounds of objective function can be found. Initially lower bound equals 0 and upper bound is a highest total benefit. Current exploring bound can be obtained from the formula:

$$Bound = \frac{Upper+Lower}{2} \tag{22}$$

Now solution with total benefit greater than the current bound should be found. It requires setting a constraint on the objective function:

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i^{max}} n_{ij} B_i \ge Bound \tag{23}$$

If solution for the constrained function was found then current bound becomes lower bound (otherwise current bound becomes upper bound), the next current bound is calculated using the formula (22) and problem with a new constraint (23) on the objective function should be solved.

The process of binary search stops when difference between lower and upper bound is less or equal to the lowest benefit amongst all the building types. The last solution is considered to be the best.

5 System design

This section presents design decisions made during the process of system development and describes main components of the system.

5.1 High-level system architecture

High-level architecture of the system is illustrated in Figure 12. The main components of the system are Graphical User Interface, database and solver.

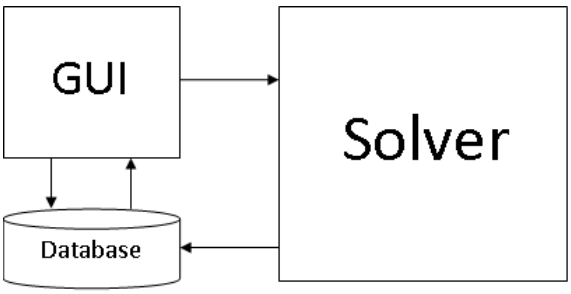

Figure 12. High-level system architecture.

5.2 Database

Database aims to save input data (polygons and building types) and all intermediate solutions for the problem.

Main and excluded polygons can be stored as a sequence of the points. Building type contains name, width, length, benefit and colour. Name of the building type is restricted to be unique. Solution contains a set of coordinates of building centres associated with its type.

5.3 Graphical User Interface

Graphical User Interface provides a set of windows for data inputting, manipulating optimization process and result outputting. Solutions are displayed on the map as they are found. Also GUI specifies a set of error messages indicating invalid input.

5.4 Solver

Solver performs computation of solution maximizing total benefit. It is a standalone application, and communication between GUI and solver should be performed through the remote calls from the developed application to the solver.

6 System implementation

Implementation of the system for area planning is discussed in this section. Also there is a class diagram of the main components and short overview of presented classes.

6.1 Language

The system is implemented in Java language. It was used because chosen solver provides Java API. Also applications written in Java are supported by the majority of existing platforms and it provides all the features needed for the system (such as graphical libraries).

6.2 Database

Database is implemented as static Java ArrayLists and HashMaps, and data access objects are presented by static methods of the classes. Database was developed using dynamic types in order to simplify the implementation of the whole system and focus on the improvement to the model. Any type of the database (for example, MySQL) can be used in the system and it was considered as a trivial task. Class diagram of the classes functioning as a database is shown in Figure 13.

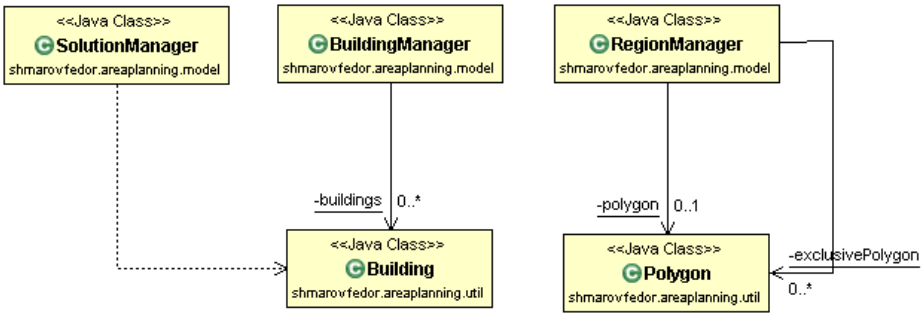

Figure 13. UML diagram of the database.

BuildingManager stores information about the building types in the HashMap and implements methods of accessing this information. The name of the building type is a key in the HashMap which guarantees the uniqueness of the building type names. The value in the map is an instance of the Building class defining the building type.

RegionManager stores main and excluded polygons in the ArrayLists and contains methods for creating polygons from a list of points and accessing these objects. Main region and excluded areas are stored as instances of the Polygon class. Polygon class stores information about the region such as coordinates of its vertices and parameters of each line forming the polygon. Also it contains methods for calculating the area of the polygon and checking whether the polygon is convex.

SolutionManager stores information about the solutions in ArrayList. Each solution at the same time is an ArrayList of instances of the Building class. Each instance of the Building class in the solution presents a building with its coordinates, width, length and benefit. Also SolutionManager stores the value of objective function and upper and lower bounds.

6.3 Graphical User Interface

Graphical User Interface was implemented using Java Swing library. UML diagram of GUI is presented in Figure 14.

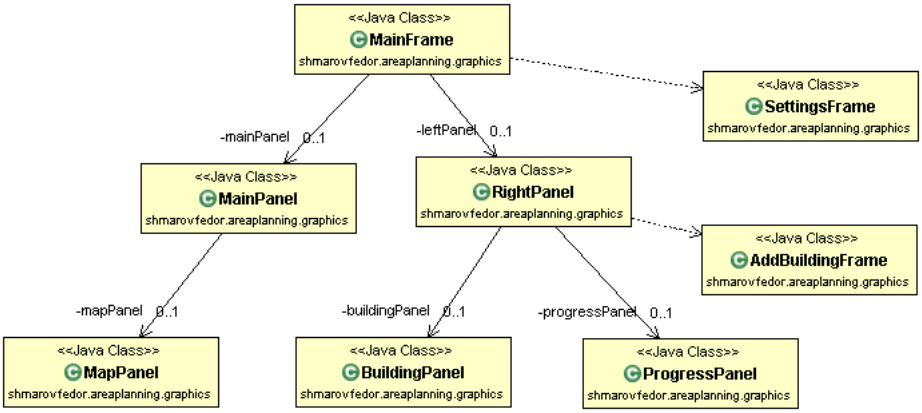

Figure 14. UML diagram of GUI

SettingFrame provides a window for inputting settings such as time limit and lower bound.

AddBuildngFrame provides a window for entering name, width, length, benefit and colour of the building type.

MainFrame contains all the main parts of the GUI such as map panel, progress panel and building panel. MapPanel provides an input of polygons and displays solutions on the map. Also MainPanel contains buttons (e.g. Start, Terminate) for manipulations the optimization process. RightPanel contains ProgressPanel and BuildingPanel. ProgressPanel displays information about the progress of the execution and number of solutions found. BuildingPanel shows information about building types.

The result of the implementation of Graphical User Interface is illustrated in Figure 15.

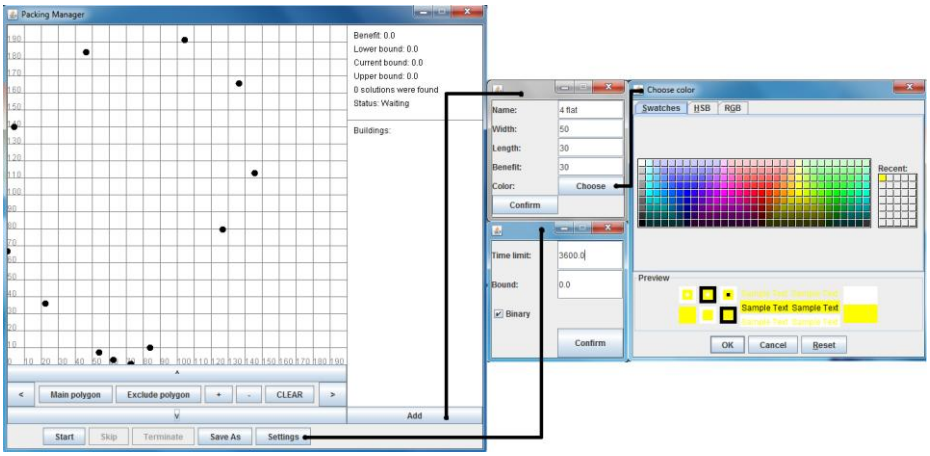

Figure 15. Graphical User Interface of the system

6.4 User to application interaction

Map. User should input the region as a set of points. Points should be entered in a counter clockwise direction. The example of input is presented in Figure 16.

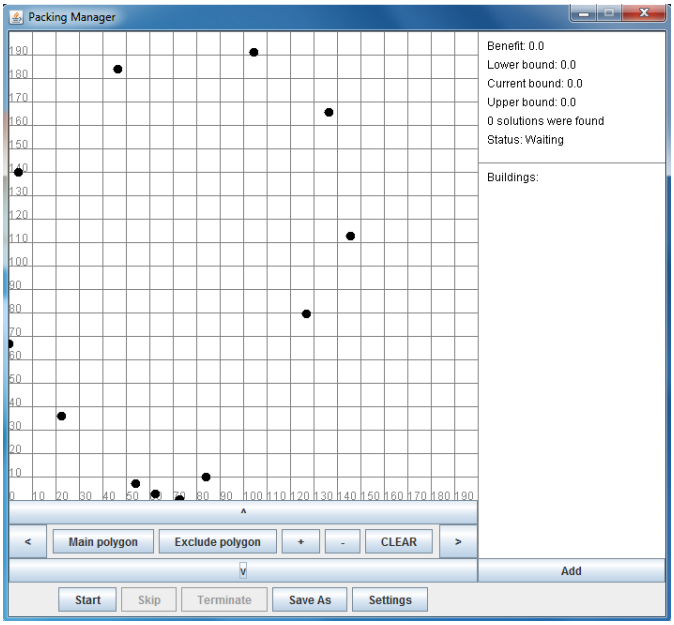

Figure 16. Example of region input

User can specify the region pressing the “Main polygon” button and exclude areas using “Exclude polygon” button. All the polygons entered by the user should be convex. If polygon input by the user is not convex, than error message appears (Figure 17) and all the points are removed from the map.

Figure 17. Error message

Also there are buttons performing zooming and moving the map. Finally, entered points can be removed using “Clear” button.

Building types input. In order to add a building type to the application user should click “Add” button in the main frame, and a window for entering information about building types will be displayed (Figure 18).

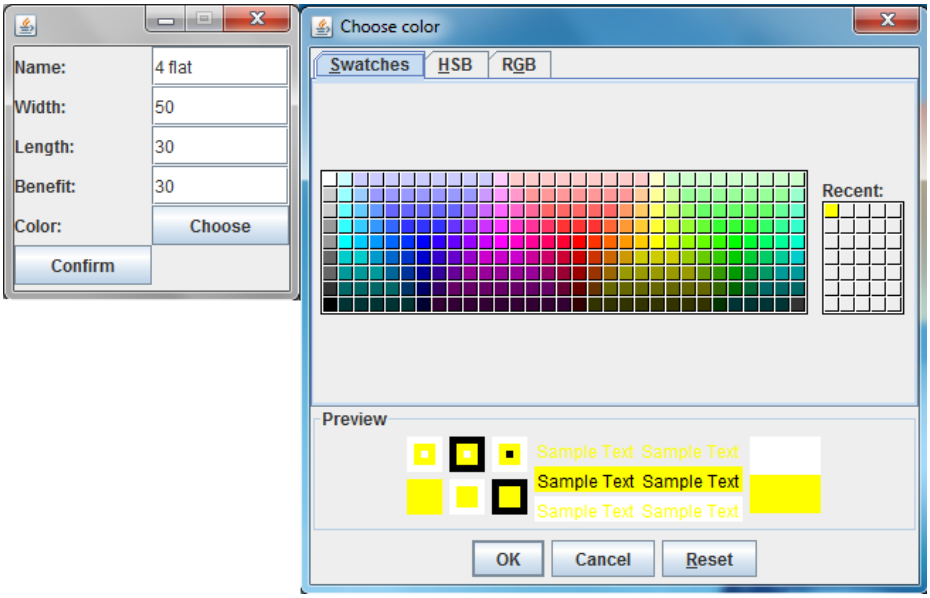

Figure 18. Window for entering building types

User has to enter the name of the building type, its width, length and benefit. Also there is a standard window for choosing colour which will be used to display all the buildings of this type on the map. Name of the building must be unique. If user tries

to create two building types with the same name an error message will be shown (Figure 19).

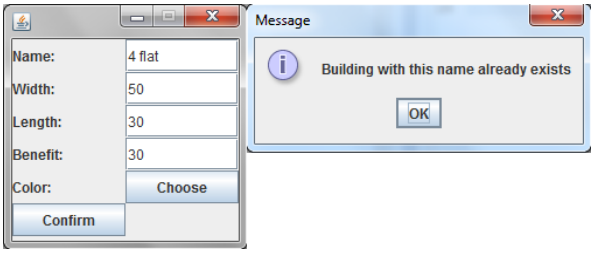

Figure 19. Entered name which already exists

Width, length and benefit should be positive. In the case if one of these parameters is negative error message is demonstrated (Figure 20) and entered data is not saved.

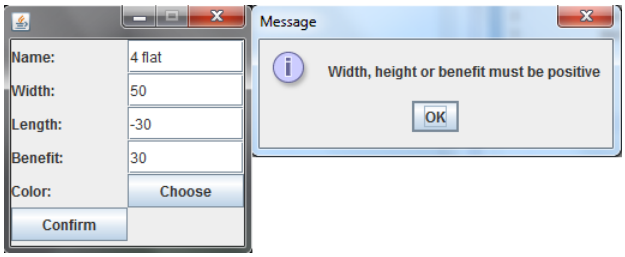

Figure 20. Example of entering negative parameter

Settings. User can specify several settings. Window with the settings is invoked pressing “Settings” button in the main frame (Figure 21).

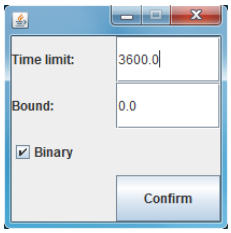

Figure 21. Window for entering settings.

It is shown in the figure that there are three settings available. Checkbox “Binary” allows using binary search. If the checkbox is empty then calculation will be performed for the benefit entered in the “Bound” text field. Also there is a time limit for the optimization process. If no solutions found within this time bounds optimization

process is terminated. If binary search enabled then time limit is set for each iteration in the search. Time limit must be positive and if user enters not positive number error message will be presented (Figure 22).

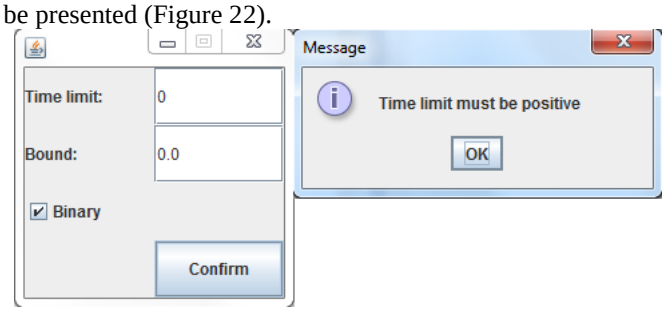

Figure 22. Example of entering non positive time limit.

Optimization execution. When all the data is input user can run the optimization process pressing "Start" button. Once the optimization started "Start" button becomes disabled and buttons "Skip" and "Terminate" are enabled. "Skip" button is only used in the binary search mode. Pressing this button user can enforce the termination of current iteration and current bound of the benefit will become an upper bound and new iteration will be performed. If "Terminate" button is pressed whole optimization process is terminated. Once termination is performed, "Terminate" and "Skip" buttons become inactive and "Start" button is enabled and user is allowed to start the process again.

Screenshot of the application with entered data is illustrated in Figure 23. Region and excluded areas are displayed on the map; building types are shown in the panel in the right-hand side of the main window. It presents colour, name, width, length and benefit of each building type. Panel in the top-right corner of the main frame displays information about the optimization process (the best benefit obtained so far, lower and upper bounds for the binary search mode, benefit computed currently, number of solutions found and status of optimization process). Also there are several statuses (see Table 2) of the optimization process.

Table 2. Possible statuses of optimization process

Status	Description
Waiting	Application is waiting for an input
Initialization	Model is being built
In progress	Optimization is running
Terminated	Optimization process is over or it is terminated by the user
Recalculation	Current bound is being recalculated

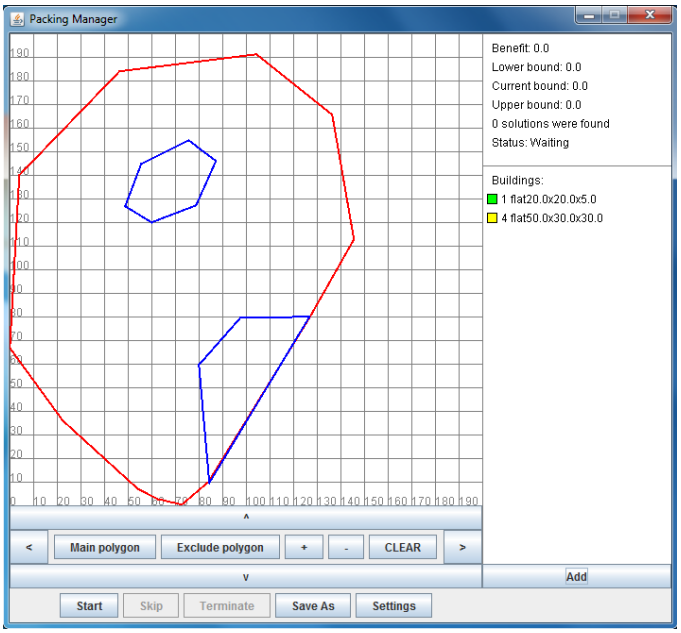

Figure 23. Example of the execution with defined region, excluded areas and building types.

6.5 Application to solver interaction

Gurobi Optimizer was chosen as solver for the system. It provides a wide range of APIs for many programming languages (e.g. Java, C++). Gurobi Optimizer is a separate application installed on the computer, and interaction between GUI and solver in the system is performed through the API contained in the *gurobi.jar* archive. Application to solver interaction in the system is implemented in the class OptimizationManager which plays a role of a wrapper (Figure 24). It contains a set of static methods (which allows allow using them without creating an instance of the class) for creating and optimizing the model and terminating the optimization. Also it stores information about the progress of the optimization.

6.6 Callback

Gurobi Optimizer [9] provides an opportunity to get current solution while the rest of the computation is being performed. In order to perform such functionality a class extending GRBCallback class should be created. In the implemented system there is a class Callback (Figure 24) with the overridden callback() method. All the code related to the result retrieval is placed there.

Moreover, Callback helps to manipulate the optimization process terminating the computation after the first solution for the current bound was obtained.

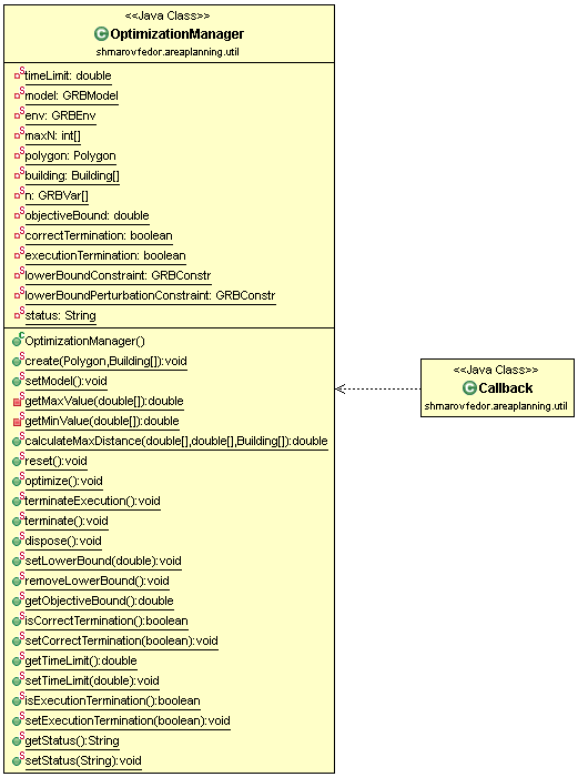

Figure 24. UML diagram of OptimizationManager and Callback classes.

6.7 Multithreading

There is a need to use multithreading in order to obtain intermediate solutions while the computation is still being performed. Therefore, GUI should be running in the main thread and optimization process in background thread. Multithreading was achieved by using means of Swing library. In the implemented system there is a class BackgroundWorker extending SwingWorker class [18]. A sequence of actions which should be done in the background are defined in doInBackground() method. This method was overridden and contains operations performing computation of the benefit. GUI and database are running in the main thread. Hence, it allows displaying intermediate results without being blocked by the optimization process.

7 Testing

This section discusses testing performed after the system was developed. Two types of testing are described in this chapter: unit testing and integration testing.

7.1 Unit testing

Unit testing was performed for the Building and Polygon classes and BuildingManager, RegionManager and SolutionManager in order to check their behaviour, and correct the implementation if obtained results were not the same as expected ones. JUnit [10] test framework was used for this purpose.

7.2 Unit testing results

Buidling class implements methods for calculating the area and creating an instance of the class. Input parameters for the constructor of the class are width, length, benefit, name and colour. If name is null or empty constructor throws an exception. Also exception is thrown if values of width, length or benefit are not positive. Results of JUnit test are presented in Table 3.

Table 3. Results of JUnit test of Building class

Test case	Expected result	Obtained result	Evaluation
name is empty	IllegalArgumentException	IllegalArgumentException	PASS
width less or equal to 0	IllegalArgumentException	IllegalArgumentException	PASS
length less or equal to 0	IllegalArgumentException	IllegalArgumentException	PASS
benefit less or equal to 0	IllegalArgumentException	IllegalArgumentException	PASS

Polygon class implements methods for creating an instance of the class, calculating area of the polygon and checking whether the polygon is convex. Constructor takes a list of point as an input parameter. If the list is empty or entered polygon is not convex then exception is thrown. Results of JUnit test are shown in Table 4.

Table 4. Results of JUnit test of Polygon class

Test case	Expected result	Obtained result	Evaluation
list of points is null	IllegalArgumentException	IllegalArgumentException	PASS
list of points is empty	IllegalArgumentException	IllegalArgumentException	PASS
polygon is not convex	IllegalArgumentException	IllegalArgumentException	PASS

Database classes are mainly wrappers of standard methods of HashMap and ArrayList. Hence, only specific functionality was tested.

RegionManager class contains methods for setting main polygon and adding excluded polygons. It works with the objects of Polygon class. If the polygon is not convex then exception thrown by the Polygon object is caught and method of RegionManager returns false. Results of JUnit test are presented in Table 5.

Table 5. Results of JUnit test of RegionManager class

Test case	Expected result	Obtained result	Evaluation
Main region is not convex	false	false	PASS
Excluded polygon is not convex	false	false	PASS

BuildingManager class implements methods for adding building type to the database. Building types are stored in the HashMap with the name of the building as a key. Result of JUnit test are demonstrated in Table 6.

Table 6. Results of JUnit test of BuildingManager class

Test case	Expected result	Obtained result	Evaluation
Adding building type which already exists	null	null	PASS

SolutionManager class contains methods for adding solution to the database, setting information about lower and upper bounds and storing the value of objective function. It implements only wrappers for standard methods of ArrayList class and it was not tested using JUnit test.

7.3 Integration testing

The purpose of integration testing is to verify functional and non-functional requirements of the system. There are several objectives which should be achieved during the testing:

- Check whether intermediate results are being displayed on the map.
- Check reliability of the application
- Check whether multithreading is working
- Check whether decreasing of the upper bound is working
- Check whether it is possible to save current solution displayed on the map as an image

In order to achieve objectives of the integration testing a test case was set. There is a region and excluded area shown in Figure 25 and two building types presented in Table 7.

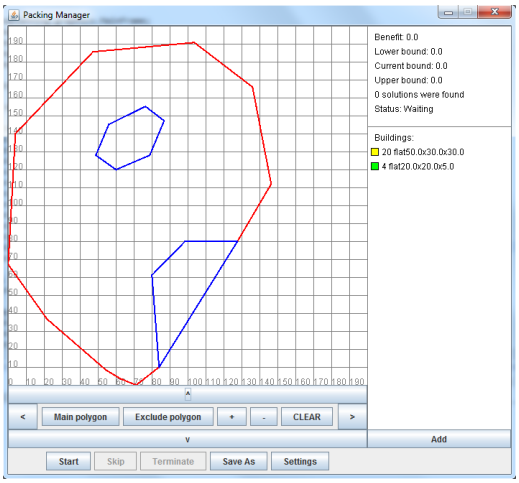

Figure 25. Region used in the test.

Table 7. Results of JUnit test of RegionManager class

Name	Width	Length	Benefit
type1	20	20	5
type2	50	30	30

Computer used for the test: CPU: Intel Core i7-2600 @ 2800 GHz with 8 GB of RAM. Additional settings to the application: binary search with the time limit of 3600 seconds per iteration.

7.4 Integration testing results

Figure 26 introduces the first iteration of the binary search. It can be seen that initial upper bound equals 630 and current bound is 315. In first minutes solution was not found and it was decided to reduce the upper bound.

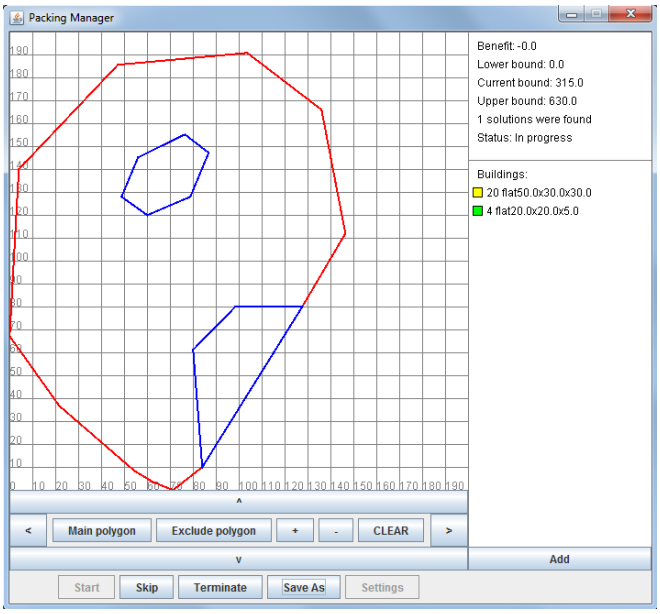

Figure 26. First iterration.

Intermediate solution which obtained using “Skip” button is illustrated in Figure 27. Solution for the benefit greater or equal to 157.5 was found and the value of objective function equals 160.

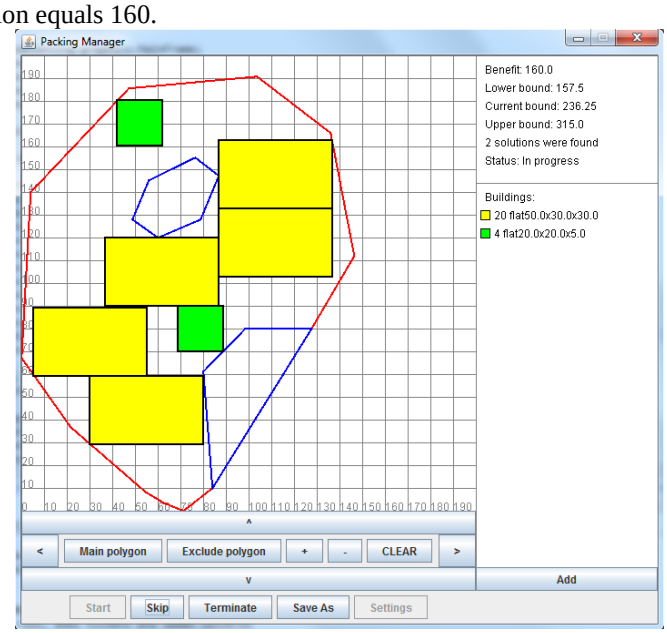

Figure 27. Intermediate solution.

Next intermediate solution was saved as an image using “Save as” button. The result is presented in Figure 28.

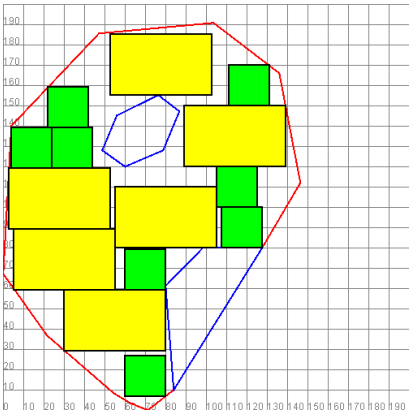

Figure 28. Intermediate result saved as an image.

The final solution obtained using binary search is illustrated in Figure 29. It can be seen that 5 intermediate solutions were found and 7 iterations were performed. The best total benefit equals 225.

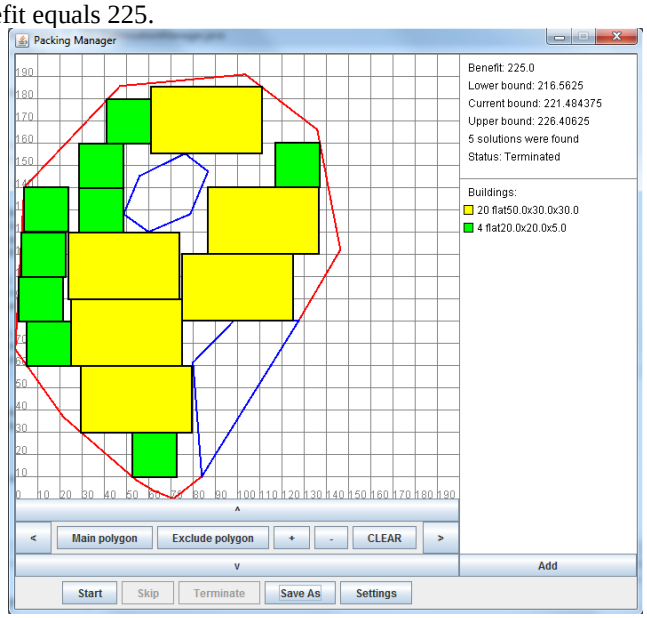

Figure 29. The best solution.

8 Evaluation

This section discusses the evaluation of the aim and objectives and functional and non-functional requirements.

8.1 Aim evaluation

The main aim of the project was to develop a software performing maximization of the total benefit having predefined region and building types as input parameters and outputting result of optimization. It was achieved as the desired system allowing user to set the problem using GUI and get a graphical representation of result was developed.

8.2 Objectives evaluation

The first objective was to evaluate existing software for area planning. This objective was achieved and several applications (e.g. SITEOPS and Autodesk Infraworks) were considered. It was concluded that none of existing system for area planning meets all the requirements introduced to the system.

The second objective was to provide a reduction of existing problem to the investigate problem. It was reached and the problem was obtained by reducing two dimensional bin packing problem. Consequently, the problem was formulated as MIP problem.

The third objective was to evaluate methods of solving the reduced problem. This objective was accomplished as several approaches were considered and it was decided to use an *"off-the-shelf"* solver. Gurobi Optimizer was chosen as the best option amongst all evaluated solvers.

The fourth objective was to implement the system in Java language. It was reached as the application is written in Java and it communicates with the solver through the API calls.

The last objective was to test the implemented system. This objective was achieved by performing a series of JUnit and integration tests. Results obtained during the testing state about the correctness of the system. It was shown that intermediate results are displayed on the map as they are found and it is possible to decrease upper bound without terminating the process. Moreover, application does not block during the optimization and it is possible to save a solution as an image.

8.3 Requirements evaluation

Functional and non-functional requirements were evaluated and result of evaluation is presented in Table 8. Conclusion column presents the evaluation of requirement and contains "yes" if it was met, "partially" if it was met only to some extent and "no" if the requirement was not met.

Table 8. Requirements evaluation

Requirement	Description	Conclusion
F1	System should provide a graphical user interface implementing windows for inputting data and outputting the result	yes
F1.1	Main window should contain a map allowing input of the region as a sequence of points in counter clockwise direction and user should be able to specify the main region and excluded areas.	yes
F1.2	The map created for an input should be suitable for displaying the result of optimization.	yes
F1.3	Main window should contain a panel reflecting the progress of optimization, current value of objective function, number of solutions found and lower and upper bound for the binary search mode.	yes
F1.4	Main window should contain a panel displaying building types involved into the optimization process.	yes
F1.5	Main window must have buttons for manipulating the optimization process allowing user to start and terminate optimization process, save the result of optimization as an image. Also user should be allowed to reduce the upper bound of objective function without terminating the process.	yes
F1.6	There should be a window for setting building types containing fields for inputting name, width, length, benefit and color of the building type	yes
F1.7	There should be a window for settings allowing user to set a time limit for iteration in binary search, lower bound for the total benefit and checkbox enabling and disabling binary search mode.	yes
F2	GUI should reply with an error message if any of input data is invalid.	yes
F2.1	If entered polygon is not convex	yes
F2.2	If user attempted to create two building types with the same name	yes
F2.3	If width, length or benefit of the building less or equal to 0	yes
F3	Application should not block during the optimization process, and intermediate solutions should be displayed on the map as they are found.	yes
N1	System should not crash during the execution	yes

9 Conclusions and Future Work

9.1 Summary

To sum up, the main aim of the project was achieved and software computing maximal benefit of the construction on the site was developed. Also existing software systems for area planning were evaluated and it was discovered that none of them fully meet all the requirements imposed on the system. Moreover, it was identified that investigated problem is NP-complete [12] and it was formulated as MIP problem [2, 9]. Methods of solving the formulated problem were evaluated and it was decided to use an *"off-the-shelf"* solver. Finally, the system was evaluated and it was concluded that all the requirements are met.

9.2 Future Work

Researched problem opened new areas for further exploration and improvements. The future work can be performed in two directions. One way of future enhancements is optimization of the existing model. Apart from the breaking symmetry and binary search other improvements to the model can be introduced.

Another way is to extend existing model by introducing new functionality and removing simplifications which were imposed on the system. For example, rotations and arbitrary shape of the buildings can be allowed. Moreover, the representation of the problem can be moved from two dimensional to three dimensional. Without a doubt, all these enhancements will make the problem more complicated as more parameters will be introduced for buildings and sophisticated landscape analysis should be undertaken, but it will make the problem representation more realistic and therefore systems developed for solving the problem will be more applicable for the real situations. Moreover, it is possible to increase performance of the system applying cloud computing to the algorithm used for solving MIP problem.

References

1. Autodesk InfraWorks - http://www.autodesk.com/ *Accessed* 28/08/2013.
2. J. E. Beasley, editor. *Advances in Linear and Integer Programming*. Oxford Science, 1996.
3. E. G. Birgin, J. M. Martınez, W. F. Mascarenhas, D. P. Ronconi. *Method of Sentinels for Packing Items within Arbitrary Convex Regions*. June 23, 2005.
4. E. G. Birgin, J. M. Martınez, F. H. Nishihara, D. P. Ronconi. *Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization*. February 3, 2005.
5. COBYLA2 - http://www.codeproject.com/Articles/508513/ *Accessed* 28/08/2013.
6. CPLEX - http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ *Accessed* 28/08/2013.
7. M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Co. pp. x+338, 1979.
8. I. Griva, S.G. Nash, A.Sofer . *Linear and Nonlinear Optimization. SECOND EDITION.* George Mason University, Fairfax, Virginia, 2009.

9. Gurobi Optimizer - http://www.gurobi.com/ *Accessed* 28/08/2013.
10. JUnit - http://junit.org/ *Accessed* 28/08/2013.
11. R.M. Karp. *Reducibility among combinatorial problems.* In R. E. Miller and J. W. Thatcher (editors). *Complexity of Computer Computations*. New York: Plenum. pp. 85–103. 1972.
12. lp_solve - http://lpsolve.sourceforge.net/5.5/ *Accessed* 28/08/2013.
13. A. Pasha. *Geometric Bin Packing Algorithm For Arbitrary Shapes*. University of Florida, 2003.
14. W.W. Royce. *Managing the development of large software systems*. Proceedings of IEEE WESCON (August): 1–9, 1970.
15. SITEOPS - http://www.siteops.com/ *Accessed* 28/08/2013.
16. SmartDraw - http://www.smartdraw.com/ *Accessed* 28/08/2013.
17. I. Sommerville. *Software Engineering. Ninth Edition.* 2011.
18. SwingWorker - http://docs.oracle.com/javase/6/docs/api/javax/swing/SwingWorker.html *Accessed* 28/08/2013.
19. N. Taylor. *Urban Planning Theory since 1945*. London, Sage, 2007.