Applied Mathematics & Information Sciences
*An International Journal*

http://dx.doi.org/10.18576/amis/100408

# Evolutionary Optimization of Set-Covering Problem

*Eva Volna*[*] *and Martin Kotyrba*

Department of Informatics and Computers, University of Ostrava, 30 dubna 22, Ostrava, Czech Republic

Received: 21 Mar. 2016, Revised: 7 May 2016, Accepted: 8 May 2016
Published online: 1 Jul. 2016

**Abstract:** The article aims how to use evolutionary algorithms in solving a set-covering problem. We have focused on the bin packing problem. This problem is known to be NP-hard; hence many heuristic procedures for its solution have been suggested. We propose a new solution of the problem by a genetic algorithm. The included experimental study presents the use of a genetic algorithm to find an optimal layout for the placement of regular patterns of fixed sizes and simple shapes to minimize the waste. This study indicates that genetic algorithms can effectively be used to obtain highly efficient solutions.

**Keywords:** Global optimization, combinatorial optimization, set-covering problem, genetic algorithms, bin packing problem.

## 1 Global optimization

Global optimizations utilize techniques that can distinguish between the global optimum and numerous local optima within a region of interest. Global optimization problems usually take from of unconstrained optimization; that is, the problem is one of minimizing or maximizing a function in the absence of restrictions [5]. In general, an unconstrained optimization problem can be represented mathematically as follows (1), [1]:

$$min\, f(x), \quad \text{subject to} \quad x \in \Omega \tag{1}$$

where f is real-valued function and , the feasible set, is a subset of $E^n$. When attention is restricted to the case where $\Omega = E^n$, it corresponds to the completely unconstrained case. In many applications we just need to consider the case where  is particular subset of $E^n$. A point $x^* \in \Omega$ is said to be a local minimum of f over  if there is an $\varepsilon > 0$ such that $f(x) \le f(x*)$ for all $x \in \Omega$ within a distance $\varepsilon$ of $x^*$. A point $x^* \in \Omega$ is said to be a global minimum of $f$ over $\Omega$ if $f(x) \ge f(x^*)$ for all $x \in \Omega$. Even though most practical optimization problems have side restrictions that must be satisfied, the study of techniques for unconstrained optimization provides a basis for further study. Conventional global optimization methods can roughly be categorized into two classes: a) deterministic methods and b) stochastic methods. Genetic algorithms have been fairly successful at solving problems of the type that are too all-behaved, nondifferentiable, and discontinuous for conventional hill-climbing and derivative-based techniques. Examples of such problems are multimodal nondifferentiable, and discontinuous problems. Since the emergence of genetic algorithms in early 1970s, global optimization has been one of their major targets, and a lot of effort has been devoted to developing powerful algorithms for global optimization problems [12].

## 2 Evolutionary algorithms - genetic algorithms

Evolutionary algorithms (EAs) have many interesting properties and have been widely used in various optimization problems from combinatorial problems such as job shop scheduling to real valued parameter optimization [1], [21]. In computer science, evolutionary computation is a subfield of artificial intelligence (more particularly computational intelligence) that involves combinatorial optimization problems. Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution. As evolution can produce highly optimised processes and networks, it has many applications in computer science [4]. Problem solution using evolutionary algorithms is shown in Figure 1.

[*] Corresponding author e-mail: eva.volna@osu.cz

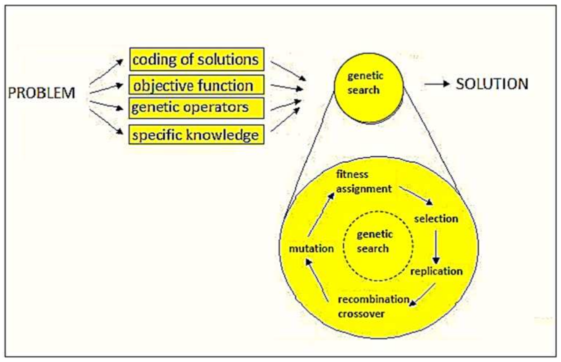


**Fig. 1:** Problem solution using evolutionary algorithms (adapted from http://jpmc.sourceforge.net )

A genetic algorithm is a type of a searching algorithm. It searches a solution space for an optimal solution to a problem. The key characteristic of the genetic algorithm is how the searching is done. The algorithm creates a population of possible solutions to the problem and lets them evolve over multiple generations to find better and better solutions. The generic form of the genetic algorithm is shown in Figure 2. The items in bold in the algorithm are defined here [2], [17].

The population consists of the collection of candidate solutions that we are considering during the course of the algorithm. Over the generations of the algorithm, new members are born into the population, while others die out of the population. A single solution in the population is referred to as an individual. The fitness of an individual is a measure of how good is the solution represented by the individual. The better solution has a higher fitness value  obviously, this is dependent on the problem to be solved. The selection process is analogous to the survival of the fittest in the natural world. Individuals are selected for breeding (or cross-over) based upon their fitness values. The crossover occurs by mingling two solutions together to produce two new individuals. During each generation, there is a small chance for each individual to mutate. Genetic optimization deals with problems of minimizing or maximizing a function with several variables usually subject to equality and/or inequality constrains. It plays a central role in operations research, management science, and engineering design. Many industrial engineering design problems are very complex and difficult to solve using conventional optimization techniques [19]. In recent years, genetic algorithms have received considerable attention regarding their potential as a novel optimization technique. Because of their simplicity, ease of operation, minimal requirements, and parallel and global perspective, genetic algorithms have been applied successfully in a wide variety of problem domains [5].

## 3 Combinatorical optimization problems

Combinatorical optimization study problems, which are characterized by a finite number of feasible solutions, abound in everyday life, particularly in engineering design. An important and widespread area of applications concerns the efficient use of scarce resources to increase productivity. Typical engineering design problem relate to set covering, bin packing, knapsack packing, quadratic assignment, minimum spanning tree determination, machine scheduling, sequencing and balancing, cellular manufacturing design, vehicle routing, network design, facility location and layout, traveling salesman assignment, and so on. Although, in principle, the optimal solution to such problems can be found by simple enumeration, in practice it is frequently impossible, especially for practical problems of realistic size, where the number of feasible solutions can be extremely high. One of the most challenging problems in combinatorial optimization is to deal effectively with the combinatorial explosion. A major trend in solving such difficult problems is to the use of genetic algorithms.

## 4 Set-Covering problem

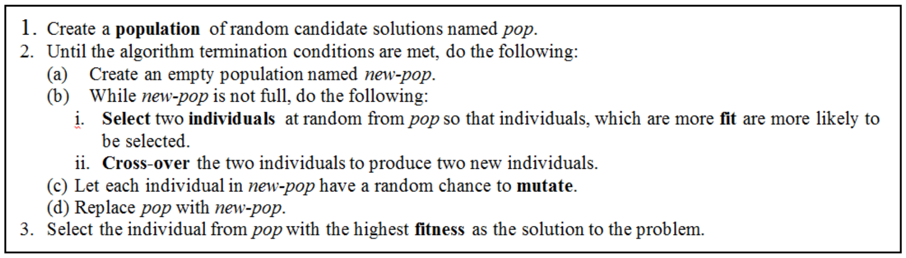
The set-covering problem is a typical problem in combinatorial optimization. The problem is to cover the rows of an $m$-row/$n$-column zero-one matrix with subset of columns at minimal cost. Consider a vector $x$ such that $x_j = 1$ if column $j$ (with a cost $c_j > 0$) is in solution and $x_j = 0$ otherwise ($j = 1,2,\dots,n$). The set-covering problem is then formulated as

$$\min \quad z(x) = \sum_{j=1}^{n} c_j x_j \tag{2}$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1,2,\dots,n \tag{3}$$

$$x_j \in \{0,1\}, j = 1,2,\dots,n \tag{4}$$

Equation (3) ensures that each row is covered by at least one column and equation (4) is the integrality constraint. If all the cost coefficients $c_j$ are equal, the problem is called a unicost set-covering problem. If equation (2) is an equality function, the problem above is referred to as a set partitioning problem. Several network problems can be modeled as a set-covering problem with a particular $A = (a_{ij})$ matrix, such as the node-covering problem of Salkin a Saha [14], the matching problem of Balinski [2], and the maximum problem of Salkin and Mathur [13]. Besides, many applications of real-world problems have been reported in [5]. The set-covering problem has been proven to be NP (nondetermisted polynominal) - complete. As with other sizable combinatorial problems, there has recently been an increasing interest in evolutionary computing solution



```
1. Create a population of random candidate solutions named pop.
2. Until the algorithm termination conditions are met, do the following:
   (a) Create an empty population named new-pop.
   (b) While new-pop is not full, do the following:
       i. Select two individuals at random from pop so that individuals, which are more fit are more likely to
          be selected.
       ii. Cross-over the two individuals to produce two new individuals.
   (c) Let each individual in new-pop have a random chance to mutate.
   (d) Replace pop with new-pop.
3. Select the individual from pop with the highest fitness as the solution to the problem.
```

**Fig. 2:** The genetic algorithm

techniques. Jacobs and Brusco developed a simulated annealing algorithm and reported considerable success on problem with up to 1000 rows and 10000 columns [7]. Sen investigated the performances of simulated annealing algorithm and a simple genetic algorithm [15]. The genetic algorithm approach to set-covering problem proposed by Beasley and Chu is summarized in Figure 3 [3]. In [22], authors investigate a largely underexplored issue: the approximation performance of EAs in terms of how close the solution obtained is to an optimal solution. Authors study an EA framework named simple EA with isolated population (SEIP) that was implemented as a single- or multi-objective EA. Their theoretical analysis suggests that EAs can achieve solutions with guaranteed performance. Specifically, they analysed SEIP using a set cover problem. In [23], authors concerned with the approximated solution of large scale set covering problems arising from crew scheduling in airline companies. They propose an adaptive heuristic-based evolutionary algorithm, whose main ingredient is a mechanism for selecting a small core subproblem which is dynamically updated during the execution. Experiments conducted on real-word benchmark instances from crew scheduling in airline companies.

## 5 Bin Packing problem

In one dimensional bin packing problem the objects have one dimension. Objects value is weight, size, cost or time. The term bin here is in fact a generic name which could stand for a container, as in the transportation context, work stations in industrial assembly lines (line balancing), a space in time in scheduling, or a surface area, as in metal working, for example. The one dimensional bin packing problem can be stated as follows [9]. We are given a set of $n$ objects each with a given weight (or size)$(w_i > 0)$. We want to place these objects into bins of a given capacity C $(C > w_i)$ so that the total number of bins needed is minimized. Other words, the problem is to find a best assignment of objects to bins such that the total weight of the objects in each bin does not exceed its capacity and the number of bins is minimized. This problem has many practical applications: Trucks are to be loaded with different items. The aim is to use as few trucks as possible to carry the loads without exceeding the capacity of each truck. Another example is where tubes or cables are to be cut from quantities of standard length C. We want to use as few tubes or cables of standard length as possible to meet the demand. The same idea is used in metal working where steel sheets of different sizes must be cut from ”master” sheets. Yet another example is in scheduling, where tasks of varying duration must be allocated using the least number of machines or processors. The most obvious is the two dimensional bin packing problem, where in addition to weight, the volume of the objects too, have to be taken into account. That is instead of one set of constraints, we have two sets of constraints [6], [20]. A variation on this two dimensional problem is the ”strip packing” problem. The problem is to pack a set of $n$ rectangles into an open ended bin of fixed width C so that the height of the bin is minimized. The rectangles must not overlap. There are also other extensions such as three dimensional and maximum value problems, for which the reader is referred to many good references on this subject such as [11].

## 6 Bin packing problem as instance of set-covering problem

Given a bin $S$ of size $C$ and a list of $n$ items with sizes $w_1,\ldots,w_n$ to pack, find an integer number of bins $B$ and a $B$ partition $S_1\cup,\ldots,\cup S_B$ of the set $\{1,\ldots,n\}$ such that for all $k$=1,..., $B$. A solution is optimal if it has minimal $B$. The $B$ value for an optimal solution is denoted OPT. Mathematically the one dimensional bin packing problem can be formulated as [5]:

$$\min \quad B = \sum_{i=1}^{n} y_i \tag{5}$$

$$\text{subject to} \sum_{j=1}^{n} w_j x_{ij} \leq C y_j, \forall i \in \{1,\ldots,n\} \tag{6}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in \{1,\ldots,n\} \tag{7}$$

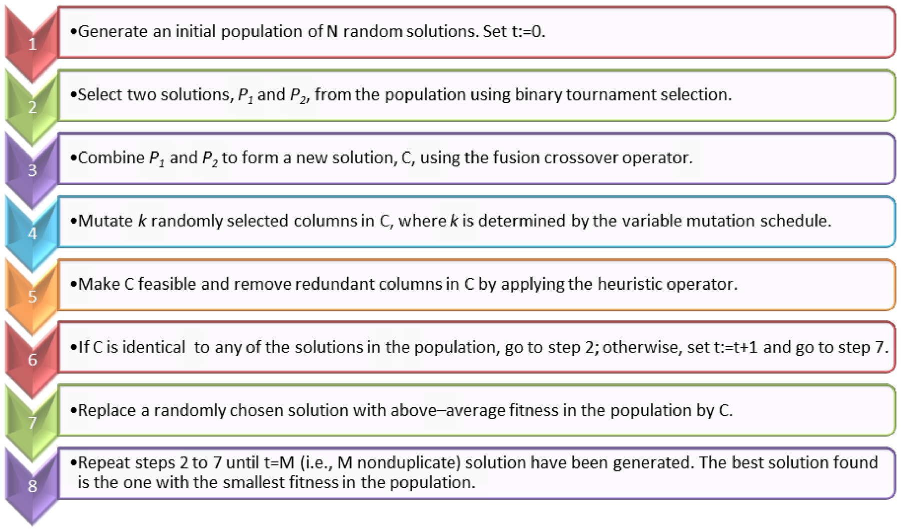

1
•Generate an initial population of N random solutions. Set t:=0.
2
•Select two solutions, $P_1$ and $P_2$, from the population using binary tournament selection.
3
•Combine $P_1$ and $P_2$ to form a new solution, C, using the fusion crossover operator.
4
•Mutate *k* randomly selected columns in C, where *k* is determined by the variable mutation schedule.
5
•Make C feasible and remove redundant columns in C by applying the heuristic operator.
6
•If C is identical to any of the solutions in the population, go to step 2; otherwise, set t:=t+1 and go to step 7.
7
•Replace a randomly chosen solution with above–average fitness in the population by C.
8
•Repeat steps 2 to 7 until t=M (i.e., M nonduplicate) solution have been generated. The best solution found is the one with the smallest fitness in the population.

**Fig. 3:** Genetic algorithm for the set-covering problem.

$$y_i \in \{0,1\}, \forall i \in \{1,\ldots,n\} \quad \text{and} \tag{8}$$

$$x_{ij} \in \{0,1\}, \forall i \in \{1,\ldots,n\}, \forall j \in \{1,\ldots,n\} \tag{9}$$

where

$$y_i = \begin{cases} 1 \textit{ if bin i is used} \\ 0 \textit{ otherwise} \end{cases} \tag{10}$$

$$x_{ij} = \begin{cases} 1 \textit{ if object j is put into bin i} \\ 0 \textit{ otherwise} \end{cases} \tag{11}$$

In this model the objective minimizes the total number of used bins (5). Constraints (6) ensure that the weights of objects placed in bin do not exceed the capacity of the bin. Note that here we assumed that all the bins have the same capacity, C, but in general this need not be the case. Finally, constraints (7) ensure that each object is placed only in one bin. This is a very straightforward greedy approximation algorithm. The algorithm processes the items in arbitrary order. For each item, it attempts to place the item in the first bin that can accommodate the item. If no bin is found, it opens a new bin and puts the item within the new bin. It is rather simple to show this algorithm achieves an approximation factor of 2, that is, the number of bins used by this algorithm is no more than twice the optimal number of bins. In other words, it is impossible for 2 bins to be at most half full because such a possibility implies that at some point, exactly one bin was at most half full and a new one was opened to accommodate an item of size at most C/2. But since the first one has at least a space of C/2, the algorithm will not open a new bin for any item whose size is at most C/2. Only after the bin fills with more than C/2 or if an item with a size larger than C/2 arrives, the algorithm may open a new bin. Thus if we have $B$ bins, at least $B$ 1 bins are more than half full. Therefore in 12,

$$\sum_{i=1}^{n} w_i > \frac{B-1}{2} C \tag{12}$$

because 13,

$$\frac{\sum_{i=1}^{n} w_i}{C} \tag{13}$$

is a lower bound of the optimum value OPT, we get that $B$ 1 $<$ 2OPT and therefore $B \leq$ 2OPT [18]. The bin packing problem can be solved using standard procedures for the solution of integer programs, such as branch and bound; but since the number of cases to be considered by enumerating all possible combinations is, the computational efforts needed to solve this problem is prohibitive for large *n's*. It is known that the bin packing problem is NP-Hard. Hence, heuristic procedures such as next fit, first fit, best fit, to name a few, are proposed for this problem. Since these methods are easily implemented, they are usually embedded in a genetic algorithm to enhance its performance. We have adopted this practice here, too.

## 7 The proposed approach based on genetic algorithms

We present a proposed methodology to generate a suitable shape layout solution using a genetic algorithm. Figure 4

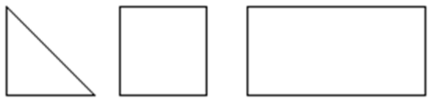


**Fig. 4:** Types of pieces: triangle, square, rectangle

shows convex shapes, which are used in the experiment study [16]. These shapes are modeled as polygonal objects. Each of the shapes has a default initial orientation, but they may be rotated. A solution is a structure with the following format: $S = [(P_1, O_1)$, $(P_2, O_2)$, $\ldots$, $(P_n, O_n)$, L] where $S$ is the solution, $P$ represents objects, $O$ is the orientation of these objects: 0 for 0 degrees, or 1 for 90 degrees, 2 for 180 degrees or 3 for 270 degrees, and $L$ is the length (cost) of the solution.

Every individual $I_k$ is represented by its chromosome that is a define set of pieces with their own orientations, see Figure 5. $I_k = (P_1, O_1)$, $(P_2, O_2)$, $\ldots$, $(P_n, O_n)$; $O_i \in \{0, p/2, p, 3p/2\}$; $P_i$ = rectangle, square, triangle; $i = 1, \ldots,$ n ($n$ is number of pieces in the $i$-th individual); $k = 1, \ldots, N$ ($N$ is number of individuals in the population).

Particular individuals are strips with the fixed width $W$ (e.g. $W = 2 \cdot l$, where l is a constant). In our experimental work, we have used 10 pieces (e.g. 2 rectangle pieces, 4 square pieces, and 4 triangle pieces). All these pieces are used just once in each chromosome. This model of chromosome is similar to that in [10]. Each piece is placed starting at the upper-left edge of the strip. If there is no space to place the piece, we move downwards until there is a space or we run to the right on the strip. When placing a new piece we have to check that a space is available. We do this by means of a simple 2-D graphics algorithm for checking that none of the vertices of our polygon is inside another, previously placed polygon. To encode an individual into a string, we use a triangle grid (Figure 6).

The proposed methodology based on genetic algorithm works as follows. The initial population is created by randomly generating N individuals. Number of individuals in the population was constant during the whole calculation. The fitness function value of each chromosome is calculated as the follows (14)

$$F_i = K - k_i \tag{14}$$

where $i = 1, \ldots, N$ ($N$ is number of individuals in the population). $K$ is a constant that represents the maximal number of triangles $k_j$ in the grid (e.g $k_j = 1, \ldots K$). $k_i$ represents the maximal used grid in the individual $i$. Figure 6 shows the worst potential individual in the population. According to the figure $k_i = 168$ and $K$ should be bigger than 176. In our experimental study, we have used K equals 200. We can state that the fitness function value of each chromosome is proportional to the reciprocal of value of its strips length:$F \approx \frac{1}{L} = \frac{l}{a \cdot l} = \frac{1}{a}$, where $a$ represents number of rectangles with width $W$ ($W = 2 \cdot l$). The best individual is automatically included to the next generation. Then, for each fitness function the probability of reproduction of its existing individual is calculated by means of standard method [8]. All of the calculated fitness function values of the two consecutive generations are sorted descending and individuals attached to the first half creates the new generation. A new individual may be created by either a crossover or a mutation.

The *crossover* runs in the following steps:

1.We pick a suitable chromosome from our population to crossover at random. It represents the chosen parent.
2.We generate a number (a strips position) that is bounded above by its strips length.
3.The first new individual includes the first substring of the parent and then we insert all the remaining pieces from the parent in its second substring - that are randomly located, but their orientations are given.
4.The second new individual includes the second substring of the parent and then we insert all the remaining pieces from this parent in its first substring - that are randomly located, but their orientations are given.

If the input condition of *mutation* is fulfilled (e.g. if a randomly number is generated that is equal to the defined constant), the *crossover* runs in the following steps:

1.One of the individuals is randomly chosen.
2.We generate a number (a strips position) that is bounded above by its strips length.
3.If the piece is at the position, its orientation is randomly changed.

The calculation is finished when an upper limit on the number of generation is reached or if the chance of achieving significant changes in the next generation is excessively low (e.g if the significant majority of individuals in the population are the same).

## 8 Experimental results

We have used a population which size throughout our experiments was constant. Every individual consists of all 10 pieces from the defined set of pieces. The initial population was created by randomly generating individuals (Figure 7). Each piece was placed in the chromosome (e.g. a strips with the fixed width $W = 2 \cdot l$, where l is a constant) at random and its orientation was generated from the set of the possible orientations. Parameters of the experimental part are the following:

1.the number of individuals: 30
2.probability of crossover: 0.5
3.probability of mutation: 0.01
4.the maximal number of triangles in the grid: $K$ equals 200

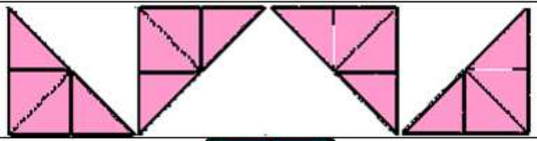
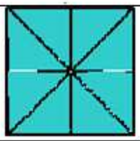
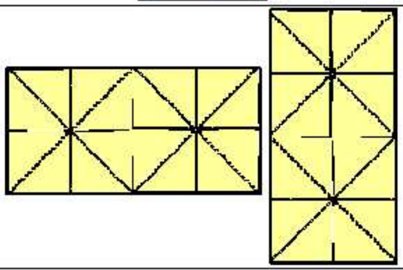
| Object | Number of occurrences | Orientation: 0°, 90°, 180°, 270° |
|---|---|---|
| triangle | 4 |  |
| square | 4 |  |
| rectangle | 2 |  |

**Fig. 5:** Pieces in each individual.

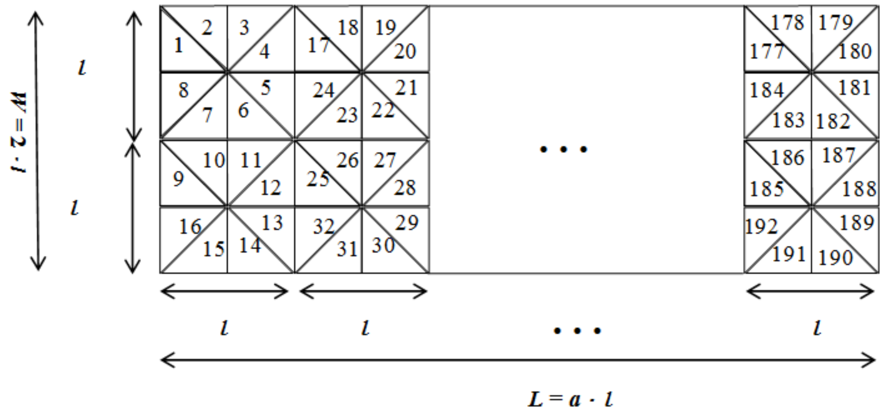


**Fig. 6:** The triangle grid.

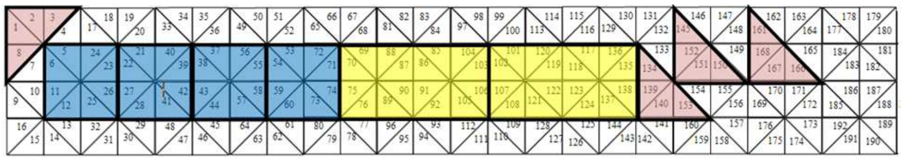


**Fig. 7:** The worst individual, which may appear in the population

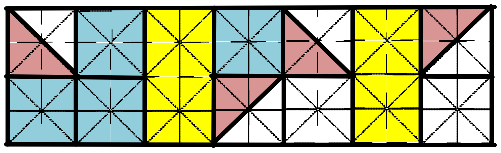


**Fig. 8:** The best individual from the initial population. Its fitness value equals 96.

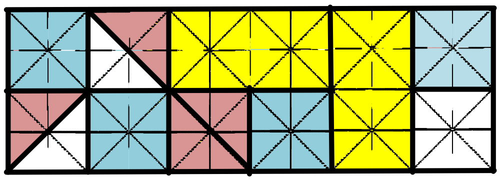


**Fig. 9:** The best individual from the 100th generation. Its fitness value equals 112.

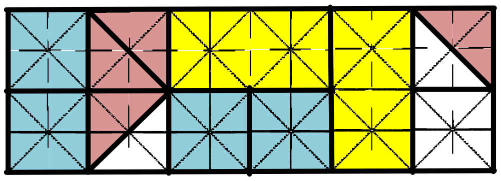


**Fig. 10:** The best individual from the 200th generation. Its fitness value equals 115.

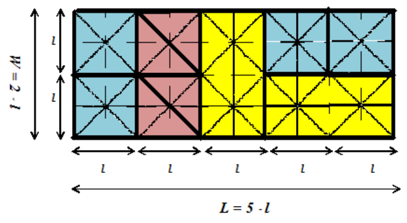


**Fig. 11:** The best individual from the final population (e.g. solution). Its fitness value equals 120.

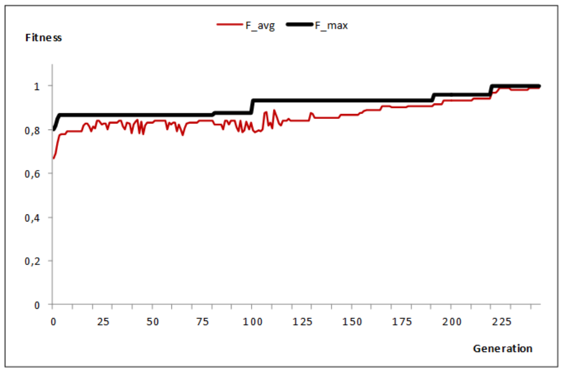


**Fig. 12:** The history of fitness function value during whole calculation.

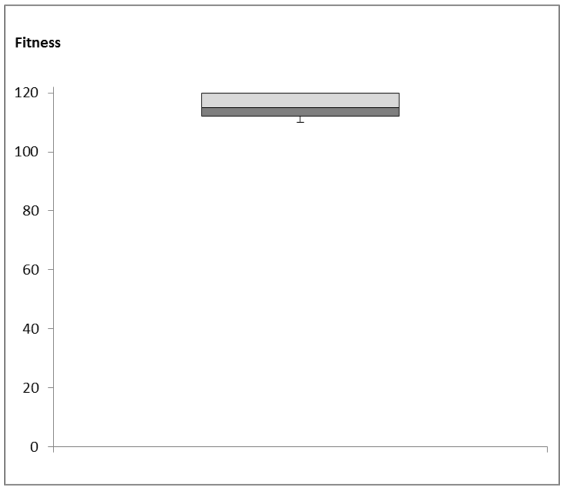


**Fig. 13:** Results representing fitness values of the best individual in the last generation.

An evolution of the population during the calculation is shown in Figures 8-11.

Our experimental results indicate that our genetic algorithm is reasonably good. The calculation was finished in the 244th generation to be characterised by the population of the same individual. Figure 9 shows the best individual (e.g. solution) of the final population: its fitness function value corresponds to the minimal value of the strips length. All used 10 pieces (e.g. 2 rectangle pieces, 4 square pieces, and 4 triangle pieces) are distributed over a strip of width $W = 2 \cdot l$ and length $L = 5 \cdot l$. The area of the entire strip is fully utilized, there is no waste. If we compare our results to those of human experts, we can observe that it is the only possible solution.

In Figure 12, the history of fitness function values is shown as: (a) the best individual in the population and (b) the average individual in the population during whole calculation. Fitness function is represented here in a relative way so that value one means the upper-most possible fitness function value and value zero means the lowest fitness function value. Other numerical simulations give similar results. The box-plots in Figures 13 and 14 represent fitness values of the best individual in the last generation as well as the generations, when the calculation was finished. Every calculation was run 50 times. The median is indicated by the horizontal line that lies inside the box dividing it into two parts.

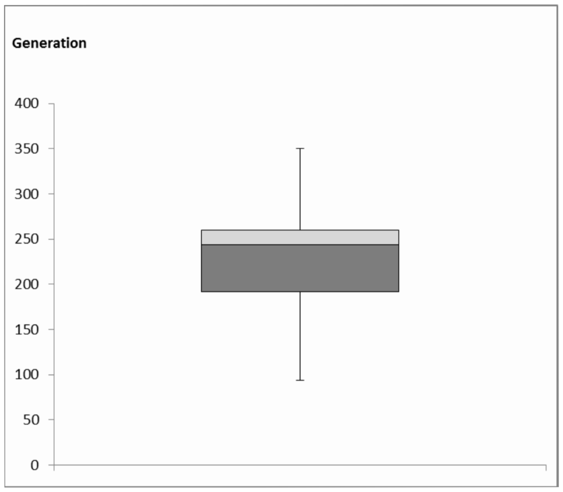


**Fig. 14:** Results representing generations, when the calculation was finished.

## 9 Conclusion

We have proposed a new algorithm which can be seen as a variant of the unicost set covering problem. The approach proposed in this paper we consider as a very promising way in order to find minimal covers of various on two dimensional bin packing problems. Results from our experimental study on various pattern designs indicate that genetic algorithms can effectively be used to obtain highly efficient solutions. A comparison of our results with other researchers is rather difficult, since the overall efficiency depends on the shape of the patterns used.In [21], three techniques are used to solve set covering problem: LINGO, genetic algorithm and ant colony optimization. Comparing the experimental study with our research, authors [21] used set of benchmark set covering problems and genetic algorithm was performed by using MATLAB Genetic Algorithm Tool, but despite of its flexibility genetic algorithm in [21] does not perform well in solving problems that have larger search space. Our future step will be a comparison of our technique with that of other researchers using the same test data. Before we do that a standard test data set has to be established. On the other hand, if we compare our experimental results to those of human experts, we can observe that we have received the only possible solution.

## Acknowledgement

The research described here has been financially supported by the University of Ostrava grant SGS/PrF/2015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

## References

[1] Back, T., Hammel U. and Schwefel H.-P. 1997. Evolutionary Computation: Comments on the History and Current State. IEEE Trans. on Evolutionary Computation, pp. 3-17.

[2] Balinski, M. L. Labelling to obtain a maximum matching. In: Combinatorial Mathematics and Its Applications (Proceedings Conference Chapel Hill, North Carolina. (1967). p. 585-602.

[3] Beasley, J. E., Chu, P. C. A genetic algorithm for the set covering problem. European Journal of Operational Research 94.2 (1996): 392-404.

[4] Bujok, P. Synchronous and Asynchronous Migration in Adaptive Differential Evolution Algorithms. NEURAL NETWORK WORLD. 2013, pp. 17-30.

[5] Gen, M. and Cheng, R. Genetic algorithms and engineering optimization, John Wiley & Sons, Inc. New York, NY, USA (2000).

[6] Hayek J. E., Moukrim A., Negre S. New resolution algorithm and pretreatments for the two-dimensional bin-packing problem, Computers & Operations Research, 35, 10, (2008) 3184-3201.

[7] Jacobs, L. W., Brusco, M.J. ”Note: A localsearch heuristic for large setcovering problems.” Naval Research Logistics (NRL) 42.7 (1995): 1129-1140.

[8] Lawrence, D. Handbook of genetic algorithms, Van Nostrand Reinhold, New York 1991.

[9] Mohamadi, N. Application of genetic algorithm for the bin packing problem with a new representation scheme. Mathematical Sciences, 4, 3, (2010) 253-266.

[10] Pargas, R. P. and Jain, R. A Parallel Stochastic Optimization Algorithm for Solving 2D Bin Packing Problem, IEEE Proc. of the 9th Int. Conf. on Artificial Intelligence for Applications, 1993, pp. 18-25.

[11] Peeters M.,Degraeve Z. Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem, European Journal of Operational Research, 170, 2, (2006) 416-439.

[12] Reiner, H., Romeijn, H. E. (eds). Handbook of global optimization. Vol. 2. Springer, (2002).

[13] Salkin, H. M.; Mathur, K. Foundations of integer programming. New York etc.: North-Holland, (1989).

[14] Salkin, H., Saha, J. Set covering: algorithms, results and codes. Bulletin of the Operations Research Society of America, 20, suppl 2, (1972).

[15] Sen, M. K., et al. Stochastic reservoir modeling using simulated annealing and genetic algorithm. SPE Formation Evaluation, 1995, 10.01: 49-56.

[16] Volna, E. Genetic algorithms for two dimensional bin packing problem, In Proceedings 12th International Conference of Numerical Analysis and Applied Mathematics, ICNAAM 2014. (in press)

[17] Volna, E., and Kotyrba, M. A comparative study to evolutionary algorithms, In Proceedings 28th European Conference on Modellingand Simulation, ECMS 2014, Brescia, Italy, 2014, pp. 340-345.

[18] Xia, B., and Tan, Z. Tighter bounds of the First Fit algorithm for the bin-packing problem. Discrete Applied Mathematics 158.15 (2010): 1668-1675. doi:10.1016/j.dam.2010.05.026.

[19] Zacek, J., Hunka, F. CEM: Class executing modelling. Procedia Computer Science. 2011, ro. 2011, s. 1597-1601.

[20] Zacek, M., Lukasova, A., Miarka, R. Modeling knowledge base and derivation without predefined structure by Graph-based Clausal Form Logic. Proceedings of the 2013 International Conference on Advanced ICT and Education. (2013). pp. 546-549.

[21] Gouwanda, D., and Ponnambalam, S. G. Evolutionary search techniques to solve set covering problems. World Academy of Science, Engineering and Technology, 2008, 39.4: 20-25.

[22] Yu, Yang; Yao, Xin; Zhou, Zhi-Hua. On the approximation ability of evolutionary optimization with application to minimum set cover. In: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, 2013. p. 3190-3194.

[23] Marchiori, E., Steenbeek, A. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: Real-World Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2000. p. 370-384.

**Eva Volna** is an associate professor at the Department of Computer Science at University of Ostrava, Czech Republic. Her interests include artificial intelligence, artificial neural networks, evolutionary algorithms, and cognitive science. She is an author of more than 50 papers in technical journals and proceedings of conferences.

**Martin Kotyrba** is an assistant professor at the Department of Computer Science at University of Ostrava, Czech Republic. His interests include artificial intelligence, formal logic, soft computing methods and fractals. He is an author of more than 40 papers in technical journals and proceedings of conferences.