



Solving Single Nesting Problem Using a Genetic Algorithm

C. Șerban, C.Ș. Dumitriu* and A. Bărbulescu

Abstract

Since the Bin Packing Problem (BPP) has application to industry and supply chain management problems (to mention only the most important ones), it attracted attention from its formulation. The Single Nesting Problem treated here is a particular case of this optimization problem, which different methods, mainly combinatorial, can solve. In this article, we propose using a genetic algorithm for solving the single nesting problem formulated in a previous article by the authors. The results comparisons prove that this approach is an excellent alternative to the combinatorial ones.

1 Introduction

The Bin Packing Problem (BPP) is one of the most studied problems in Combinatorial Optimization. Exact and heuristics methods have been proposed for solving it after the '80s, even if they have been studied much earlier (from the '30s) [21].

The idea of BPP is to consider some bins with the same capacity, c , and n items, each of them with an integer weight $w_j, j = \overline{1, n} (0 < w_j < c)$, and to distribute the items in a minimum number of bins such that the total size of the items in a bin is less than c [19].

Key Words: Optimization, Combinatorics, Bin Packing Problem, Genetic Algorithms
2010 Mathematics Subject Classification: Primary 90C08, 90C59; Secondary: 68T20
Received: 15.06.2021
Accepted: 25.09.2021

* Corresponding author

Let consider the bins numbered from 1 to m , $m \in N^*$ being the minimum number of bins necessary to pack all the items,

$$x_{ij} = \begin{cases} 1, & \text{the item } j \text{ is packed into the bin } i \\ 0, & \text{the item } j \text{ is not packed into the bin } i \end{cases} \quad i = \overline{1, m}, j = \overline{1, n} \quad (1)$$

$$y_i = \begin{cases} 1, & \text{the bin } i \text{ is utilized in the solution} \\ 0, & \text{the bin } i \text{ is not utilized in the solution} \end{cases} \quad i = \overline{1, m} \quad (2)$$

Find

$$\min \sum_{i=1}^m y_i \quad (3)$$

such that

$$\sum_{j=1}^n w_j x_{ij} \leq cy_i, i = \overline{1, m} \quad (4)$$

$$\begin{aligned} \sum_{i=1}^m x_{ij} &= 1, j = \overline{1, n} \\ y_i &= \overline{0, 1}, i = \overline{1, m} \\ x_{ij} &= \overline{0, 1}, i = \overline{1, m}, j = \overline{1, n} \end{aligned} \quad (5)$$

So, the constraints ensure that each item must be packed into a single bin and the capacity of each bin must not be exceeded. The optimization aim may be described by the inequality.

A generalization of BPP is the Cutting Stock Problem [7]. A particular case of BPP and different approaches for finding approximate solutions are presented in [22]. Among the exact algorithms employed for this goal, the most used are the Enumerative algorithm [15], Branch-and-bound [18], MTP [22], DP-flow [8]. For a review of these approaches, the reader may refer to [9],[25].

Last period, artificial intelligence and hybrid methods have been employed for solving BPP problems, to improve the solutions found by other algorithms, or to decrease the computational time required for running these algorithms [1],[2],[16],[23],[24]. Furthermore, statistical methods such as time series analysis, fitting, classification, Bayesian methods or density estimation, to mention a few, were used to address many challenges in artificial intelligence. They were

applied for solving different practical applications, to further better assessment of important features, uncover hidden patterns in data and, ultimately, improve understanding [3],[4],[5],[6],[10],[11],[12],[14].

In the presented context, we propose a genetic algorithm for solving the single-nesting problem that is a one-dimensional case of BPP.

Following this introduction, the proposed approach for solving the single-nesting problem is presented in Section 2. In Section 3, analysis of the results of the proposed method is provided. Finally, conclusions are discussed in Section 4.

2 Methodology and Data

2.1 Formulation of the Single Direction Nesting (SDN) problem

The formulation of the SDN problem is: Find the best arrangement of n bars with different lengths into m standard bars, each of them with a fixed length, lb . The goal is to determine the best combination that has all the pieces placed into the set of bars such that

$$\sum_i (lp_i)_j \leq lb \quad (6)$$

where $(lp_i)_{i \in 1, \dots, n} \in \mathbb{N}$ is the length of the piece i placed into the bar j , $j = \overline{1, n}$.

The best arrangement is the one that provides smallest sum of residual.

2.2 Evolutionary Approach of the SDN Problem

A genetic algorithm (GA) is a heuristic search process for optimization, which has been widely applied to solve combinatorial problems. GAs are designed to use the concept of evolutionary computation that employs techniques inspired by evolutionary biology, such as natural selection and reproduction.

The fundamental concept of GA is to encode the decision variables of the problem as a finite length string, called chromosomes, and to calculate their fitness. The chromosomes with a high fitness level have a higher probability of survival.

The evolution generally starts from an initial population of randomly generated chromosomes and happens in generations. In each generation, multiple individuals are selected from the current population based on their fitness, and, through the crossover and mutation process, are modified to form the next generation. When a satisfactory fitness level has been reached or the stop criterion has been met, the best chromosome found across all the generations is chosen as the final problem solution [26].

Designing a GA means establishing the encoding scheme of variables, the fitness function, and the genetic operators.

In the present model, a chromosome is coded as a string of n positive integers, which are referred to as genes. Each gene $(g_i)_{i=\overline{1,n}}$ holds the number of the bar, $j = \overline{1,m}$, on which the piece i should be placed. For example, given the chromosome in Table 1, the layout of the SDN problem should be the following: pieces 1, 4, and 7 are set on bar 1, pieces 2, 5, and 6 are set on bar 2, and pieces 3 and 8 are set on bar 3.

1	2	3	1	2	2	1	3
---	---	---	---	---	---	---	---

Table 1: Chromosome of the coding scheme

The objective of the SDN problem is to minimize the sum of allowed loose per bar:

$$f_{obj} = \min \sum_{j=1}^m |lb - \sum_{i=1}^n lp_i [g_i = j]| \tag{7}$$

where the notation $[.]$ implies the Iverson bracket, defined by

$$[M] = \begin{cases} 0, & \text{if } M \text{ is false} \\ 1, & \text{if } M \text{ is true} \end{cases} \tag{8}$$

where M is a mathematical statement.

Since the chromosomes may not satisfy the inequality constraints from (6), $h_j(x) = lb - \sum_i (lp_i)_j$, penalty functions must be added to the objective function.

$$f_{obj}(x) = \begin{cases} f_{obj}(x), & \text{if } x \in \text{feasible region} \\ f_{obj}(x) + P(x), & \text{otherwise} \end{cases} \tag{9}$$

Further on, a dynamic penalty function described by [20], which changes as the GA proceeds, is used to design the penalty function.

$$P(x, \alpha, \beta) = \rho_k^\alpha \times SVC(\beta, x) \quad (10)$$

$$\rho_k = C \times k \quad (11)$$

$$SVC(\beta, x) = \sum_{j=1}^m p_j^\beta(x), \beta = 1, 2, \dots \quad (12)$$

$$p_j(x) = \begin{cases} 0, & h_j(x) \geq 0 \\ \sum_i (lp_i)_j, & \text{otherwise} \end{cases} \quad (13)$$

where α, β, C are parameters of the method, and k is the number of the current generation considered. The parameters of the method were empirically tuned, the best results being obtained with $C = 1, \alpha = 1$ and $\beta = 10$.

The fitness function evaluates the performance of the chromosomes. A GA performs best when a feasible solution maximizes the objective function. Therefore, a function inversion scaling was applied to model the fitness function of the GA. Thus, a minimization problem was converted into an equivalent maximization one.

$$fitness(x) = \frac{1}{f_{obj}(x)} \quad (14)$$

GAs are recognized as powerful methods to generate solutions very close to optimal with less time compared to other tools [17] due to the choice of well-designed genetic operators and optimal parameters.

The selection operator drives the search process towards the regions of the best individuals. Chromosomes with higher fitness values are more likely to be selected for the mating pool, becoming the parents of the new generation. Here, tournament selection is used [27].

The crossover operator is based on the concept that the exchange of information between good individuals will generate even better offspring. Individuals are selected from the mating pool with a user-definable probability, called "crossover rate". The standard one-point crossover operator is applied to the selected pair of parents by recombining their genes and producing two offspring [26].

The mutation operator randomly modifies, with a user-definable probability called "mutation rate", one or more genes of a chromosome, thus increasing the structural diversity of the population and avoiding premature convergence. In this paper, a random resetting operator is applied: a random value from the set of permissible values $([1, m])$ is assigned to a randomly chosen gene. In order to have feasible chromosomes and preserve the stochastic characteristics of the genetic algorithm, the constraints from (6) are then imposed only

for the bar affected by the mutation operator, by applying a procedure that distributes the surplus to the other bars on the chromosome.

A GA has several configuration parameters that must be considered to increase the accuracy of the results: population size, crossover rate, and mutation rate. These parameters are problem-specific, and there is no best global value for them. A large population would lead to a better exploration of the search space, but also an increase in the runtime of the algorithm [26]. Regarding the crossover and mutation rates, if they are too high then the search will pass over good solutions, and if they are too low then the search will pass over the entire regions of solution space. Most studies in the field of GAs recommend a population size between 20 and 100 individuals, a crossover rate over 60%, and a mutation rate of at most 10%. In this paper, to set the most suitable parameters, multiple tests were configured and ran, each test being repeated ten times and the results being averaged to increase their precision.

2.3 Data Series

The data we are working with in order to validate the proposed approach of the SDN problem is the first case discussed in [13] (**SDN1** - Table 2).

n	40
m	12
lb	100
Pieces: No./Length	1/30; 2/50; 3/10; 4/20; 5/60; 6/30; 7/5; 8/10; 9/20; 10/60; 11/14; 12/50; 13/10; 14/20; 15/10; 16/30; 17/25; 18/10; 19/15; 20/60; 21/30; 22/50; 23/10; 24/45; 25/35; 26/30; 27/23; 28/10; 29/78; 30/2; 31/10; 32/20; 33/10; 34/30; 35/50; 36/10; 37/15; 38/60; 39/30; 40/50

Table 2: The parameters of the problem from [13] - **SDN1**

Further on, for evaluating the efficiency of the proposed algorithm with the tuned parameter set, it was applied to a test problem of higher scale, randomly generated, illustrated by Table 3.

<i>n</i>	160
<i>m</i>	92
<i>lb</i>	2000
Total Bars Length	149301
Valid Bars Combinations	719391
Pieces: No./Length	1/767;2/278;3/1293;4/917;5/1518; 6/1340;7/442;8/981;9/809;10/1198; 11/1104;12/898;13/1124;14/823;15/1294; 16/1243;17/1611;18/797;19/1431;20/1375; 21/754;22/696;23/1117;24/1279;25/1011; 26/717;27/929;28/1318;29/1416;; 30/1116;31/1186;32/520;33/907;34/1195;35/1395; 36/1134;37/923;38/602;39/427;40/1476; 41/1184;42/1266;43/280;44/872;45/418; 46/1459;47/1081;48/1448;49/571;50/1524; 51/856;52/596;53/1003;54/1206;55/853; 56/1317;57/1592;58/281;59/435;60/1050; 61/543;62/772;63/334;64/881;65/1193; 66/1197;67/1084;68/414;69/736;70/1216; 71/1488;72/1012;73/1445;74/478;75/948; 76/733;77/870;78/872;79/1542;80/1514; 81/1202;82/480;83/1157;84/965;85/361; 86/1038;87/265;88/1586;89/1448;90/1136; 91/1089;92/813;93/638;94/967;95/565; 96/684;97/1537;98/1489;99/263;100/926; 101/1090;102/1189;103/1450;104/1273;105/327; 106/938;107/942;108/1009;109/286;110/828; 111/699;112/278;113/1124;114/526;115/274; 116/1113;117/1482;118/975;119/949;120/517; 121/1135;122/1329;123/703;124/1343;125/513; 126/329;127/804;128/773;129/1534;130/1199; 131/583;132/578;133/760;134/391;135/1498; 136/1295;137/1520;138/395;139/1013;140/370; 141/1140;142/402;143/1206;144/475;145/621; 146/1342;147/336;148/1285;149/586;150/1281; 151/626;152/476;153/426;154/515;155/1040; 156/480;157/256;158/874;159/528;160/1538

Table 3: The parameters of the test problem - **SDN2**

3 Solution of the SDN problem

3.1 GA algorithm for finding the solution of the SDN problem

The designed genetic algorithm is presented in the following.

Input: The number and the length of the nesting length bars (m and lb , respectively), the number of pieces (n), the length of each piece (lp_i).

Output: The best solution (x), which gives us the combination of pieces for which the sum of the lengths is closer to the length of the bar they are placed on.

Begin

1. **Initialize** a random population of chromosomes, which are represented as positive integers strings of length n
2. **Evaluate** each chromosome in the population according to its fitness value
3. **Select** some chromosomes and create a mating pool by applying the selection operator
4. **Create** the offspring through crossover
 - (a) **Select** some chromosomes from the mating pool (the number of selected individuals is defined by the crossover rate)
 - (b) **Apply** the crossover operator described in 2.2 to generate new offspring
 - (c) **Copy** the remaining chromosomes (that were not recombined) to the next generation
5. **Modify** some chromosomes by applying the mutation operator (the number of selected individuals is defined by the mutation rate)
6. If the maximum number of generations is reached or most of the population (97%) has the same fitness value **then** stop, **else** go to step 2

End

The GA for this study was coded in MATLAB R2015a on a Windows 8, Intel i5 1.7 Ghz, 10 GB RAM computer.

3.2 Determination of the best parameters in the proposed GA for solving the problem 2.3

The issue of setting the appropriate values of configuration parameters of an evolutionary algorithm is crucial for good performance. Hence, multiple

settings of the GA configuration parameters were tested. In the following we shall present the stages of solving the problem presented in 2.3.

Assertion 3.1. *The optimal population size is 60.*

Proof. To choose the proper population size, the crossover rate was set to 0.75, and the mutation rate to 0.01. Running the algorithm with the population size between 20 and 100, we found that the highest fitness value had been obtained for a size of 60, as it is presented in Fig. 1.

The proof is complete. \square

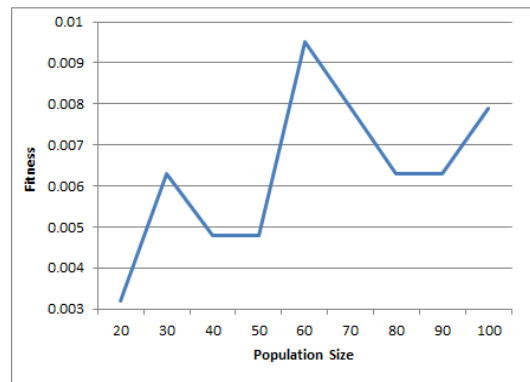


Figure 1: The effect of population size over the fitness function

Next, we perform a grid search to determine the optimal crossover and mutation rates.

Assertion 3.2. *The optimal crossover rate is 0.6.*

Proof. The tests were run using the population size previously determined (60), and varying the crossover and mutation rates from 0.6 to 0.95, and 0.01 to 0.1, respectively. The highest fitness value has been determined for a crossover rate of 0.6. The assertion is proved. \square

Assertion 3.3. *The optimal mutation rate is 0.05.*

Proof. The tests were run using the population size previously determined (60), whilst the crossover and mutation rates took values from 0.6 to 0.95, and 0.01 to 0.1, respectively. The highest fitness value has been recorded for a mutation rate of 0.05. \square

Mutation Rate	Crossover rate							
	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
0.01	0.0095	0.0032	0.0063	0.0095	0.0111	0.008	0.0111	0.0048
0.02	0.0095	0.0095	0.0095	0.0095	0.0063	0.01	0.0063	0.0063
0.03	0.0095	0.0095	0.0032	0.0127	0.0127	0.01	0.0095	0.0127
0.04	0.0063	0	0.0127	0.0111	0.0095	0.011	0.0063	0.0095
0.05	0.0127	0.0095	0.0095	0.0079	0.0048	0.011	0.0063	0.0063
0.06	0.0127	0.0063	0.0032	0.0111	0.0095	0.01	0.0079	0.0095
0.07	0.0032	0	0	0.0048	0.0048	0.002	0.0032	0.0079
0.08	0.0032	0	0	0.0016	0	0	0	0
0.09	0	0	0	0	0	0	0	0
0.1	0	0	0	0	0	0	0	0

Table 4: The effect of crossover and mutation rates over the fitness function

Pop. Size	Cross. Rate	Mut. Rate	Time (s)	Fitness (Max)	Fitness (Mean)
60	0.60	0.05	3.73	0.0159	0.0111

Table 5: The fitness and parameters settings of the proposed GA

Remark. Table 4 shows the fitness values, given the two rates. One can see that the fitness varies from 0 (which means that the fitness value was very low, due to violation of the inequality constraints from (1)) to 0.0127. The latter value is obtained two times with a crossover rate of 0.6, which leads us to choose the rates as following: the crossover rate as 0.6 and the mutation rate as 0.05. These values reassure that there is a good genetic diversity and that the search will explore much of the solutions space.

3.3 Solution of the problem SDN1

Firstly, we apply the implemented algorithm on the problem illustrated by Table 2. Table 5 shows the parameters used to implement the proposed genetic algorithm (determined in the previous section to be the best point of view of the fitness value), the execution time (column 4), and the best and average fitness (columns 5 and 6).

Tables 6 and 7 show the results from the present and the previous study. Both algorithms were run on the same machine. We remark that the total residual sum has been kept the same (63 cm, in this example), whilst the GA takes less time to perform.

3.4 Solution of the problem SDN2

We consider that the set of parameters determined in Section 3.2 (columns 1, 2 and 3 in Table 5), is the most adequate one to get the best arrangement of n bars with different lengths into m standard bars, thus, the proposed Genetic

Present study			[13]	
Bar No.	Pieces	Sum of Lengths	Pieces	Sum of Lengths
1	12, 40	100	2, 12	100
2	2, 22	100	22, 35	100
3	17, 19, 38	100	1, 33, 38	100
4	5, 15, 39	100	8, 13, 14, 15, 17, 27,30	100
5	14, 21, 35	100	4, 6, 7, 18, 23, 36, 37	100
6	4,18, 20, 33	100	3, 9, 16, 28, 31, 32	100
7	24, 27, 34	98	11, 25, 40	99
8	8, 9, 16, 28, 36, 37	95	19, 29	93
9	6, 10	90	20, 39	90
10	23, 29, 30	90	5, 34	90
11	13, 25, 26, 31	85	10, 26	90
12	1, 3, 7, 11, 32	79	21,24	75

Table 6: The results of the proposed GA applied on the first case from [13]

Process duration times	Time (s)	
	[13]	Present Study
Generation	3	-
Sorting	9	-
Selecting	1	-
Total	13	3.73

Table 7: Comparison of the nesting time

Algorithm may be used to solve other instances of the SDN problem. We applied it on the problem **SDN2**, given by Table 3.

The test problem **SDN2** was also solved by using the algorithm introduced in [13]. The results of this computational experiment are presented in Tables 8 and 9. The total residual sum has been kept the same (34699 cm), and the running time of the GA is better. We also notice that there are more bars with smaller residual in the arrangement computed with the proposed GA than in the one assessed with the algorithm in [13]. Moreover, the implemented GA solution reveals that most bars are filled more with fewer pieces, which, in some practical applications, may be more efficient.

Results	[13]	Present Study
Total residual sum	34699	34699
Number of bars with <i>residual</i> > 1000	1	0
Number of bars with <i>residual</i> ∈ [500, 1000]	40	33
Number of bars with <i>residual</i> < 500	51	59
Number of pieces / bar (average)	2.10	1.74

Table 8: Comparison of the results

Process duration times	Time (s)	
	[13]	Present Study
Generation	5	-
Sorting	30	-
Selecting	2	-
Total	37	30.17

Table 9: Comparison of the nesting time

4 Conclusion

The proposed algorithm shows to be a good competitor for the classical one, due to its flexibility and efficient implementation. It provides a computationally viable approach for those who need a fast and good solution to the SDN problem. From the computational viewpoint, the proposed GA performs faster than the algorithm from [13]. Furthermore, it is designed to provide an easy way to be applied to similar problems only by changing the input data. In future work, we shall apply this algorithm in the case when a technological loss is also allowed.

References

- [1] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, and S. Mirjalili, *An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems*, Pers Ubiquit. Comput **22** (2018), 11171132.
- [2] A. C. F. Alvim, C. C. Ribeiro, F. Glover, and D. J. Aloise, *A hybrid improvement heuristic for the one-dimensional bin packing problem*, <https://leeds-faculty.colorado.edu/glover/TSP%20-%20bin%20packing%20-%20Ceslso.pdf>
- [3] A. Bărbulescu, C. Ş. Dumitriu, *On the connection between the GEP Performances and the Time Series Properties*, Mathematics **9(16)** (2021), 1853, <https://doi.org/10.3390/math9161853>.
- [4] A. Bărbulescu, C. Ş. Dumitriu, *Mathematical aspects of the study of the cavitation in liquids*, in Series on Mathematical Modelling of Environmental and Life Sciences Problems, Proceedings of the 4th Workshop, Sept. 2005, Constanta, Romania, S. Ion, G. Marinoschi, C. Popa (eds.), Ed. Academiei Romne, Bucureti (2006), 7-15.
- [5] A. Bărbulescu, C. Şerban, M.-L. Indrean, *Improving spatial interpolation quality. IDW versus a genetic algorithm*, Water **13** (2021), 863, <https://doi.org/10.3390/w13060863>.
- [6] A. Bărbulescu, C. Şerban, S. Caramihai, *Assessing the soil pollution using a genetic algorithm*, Romanian Journal of Physics **66(3-4)** (2021), 806.

- [7] H. Ben Amor and J. Valério de Carvalho, *Cutting Stock Problems*. In: G. Desaulniers, J. Desrosiers, M. M. Solomon (eds), Column Generation. Springer, Boston, MA. (2005), 131-161.
- [8] H. Cambazard and B. O' Sullivan, *Propagating the Bin Packing Constraint Using Linear Programming*. In: D. Cohen (ed), Principles and Practice of Constraint Programming CP 2010. CP 2010. Lecture Notes in Computer Science, **6308**. Springer, Berlin, Heidelberg, (2010), 129-136.
- [9] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo, *Bin Packing Approximation Algorithms: Survey and Classification*. In: P. Pardalos, D. Z. Du, R. Graham (eds), Handbook of Combinatorial Optimization. Springer, New York, NY. (2013), 455-531.
- [10] F.-L. Dragomir, G. Alexandrescu, *Applications of artificial intelligence in the decision fundamentation*, Bulletin of "Carol I" National Defence University **4(2)** (2017), 56-61 (in Romanian).
- [11] F.-L. Dragomir, *The modeling of the decisional problems*, Bulletin of "Carol I" National Defence University, **01** (2017), 72-75.
- [12] F.-L. Dragomir, *The axiomatic character of decision*, Bulletin of "Carol I" National Defence University **4(2)** (2017), 56-61.
- [13] C. Ş. Dumitriu, A. Bărbulescu, *A method for the single direction nesting computation*, The 7th Balkan Conference on Operational Research, BACOR 05, Constanta, May 2005, Romania, https://www.academia.edu/44635040/A_Method_for_the_Single_Direction_Nesting_Computation.
- [14] C. Ş. Dumitriu, A. Bărbulescu, *Studies about the copper base alloys used in naval constructions modeling the loss mass in different media*, Sitech, Craiova (2007).
- [15] S. Eilon and N. Christofides, *The loading problem*, Manag. Sci. **17(5)** (1971), 259-278.
- [16] H. Feng, H. Ni, R. Zhao, and X. Zhu, *An Enhanced Grasshopper Optimization Algorithm to the Bin Packing Problem*, J. Control Sci. Eng., 3894987 (2020).
- [17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Pearson Education, Singapore (2002).
- [18] M. S. Hung and J. R. Brown, *An algorithm for a class of loading problems*, Naval Res. Logist. Q. **25(2)** (1978), 289-297.
- [19] M. Hyde, G. Ochoa, J. Vázquez-Rodríguez, and T. Curtois, *A HyFlex Module for the One Dimensional Bin Packing Problem*, (2011), <http://www.asap.cs.nott.ac.uk/external/chesc2011/reports/BinPackingHyFlex.pdf>.

- [20] J. Joines and C. Houck, *On the Use of Non-Stationary Penalty Functions to Solve Constrained Optimization Problems with Genetic Algorithms*, Proceedings of the 1st IEEE Conference on Evolutionary Computation, Orlando, 27-29 June 1994, 579-584.
- [21] S. Martello, *Packing problems in one or more dimensions*, (2018), http://www.or.deis.unibo.it/staff_pages/martello/Slides_Estoril_Martello.pdf.
- [22] S. Martello and P. Toth, *Knapsack problems. Algorithms and Computer Implementation*, John Wiley & Sons, Chichester, West England (1990).
- [23] E. A. Mukhacheva, G. N. Belov, V. M. Kartack, and A. S. Mukhacheva, *Linear one-dimensional cutting-packing problems: numerical experiments with the sequential value correction method (SVC) and a modified branch-and-bound method (MBB)*, *Pesqui. Oper.* **20(2)** (2000), 153 - 168.
- [24] C. Munien, S. Mahabeer, E. Dzitiro, S. Singh, S. Zungu, and A. El-Shamir Ezugwu, *Metaheuristic Approaches for One-Dimensional Bin Packing Problem: A Comparative Performance Study*, *IEEE Access* 8, 227438 (2020).
- [25] G. Scheithauer, *Introduction to Cutting and Packing Optimization. Problems, Modeling Approaches, Solution Methods*, Springer International Publishing (2018).
- [26] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer-Verlag Berlin Heidelberg (2008).
- [27] S. L. Yadav and A. Sohal, *Comparative Study of Different Selection Techniques in Genetic Algorithm*, *Int. J. Eng. Sci. Math.* **6(3)** (2017), 174-180.

Cristina ȘERBAN,
Ovidius University of Constanta,
Bdul Mamaia 124, 900527 Constanta, Romania.
Email: cgherghina@gmail.com

Cristian Ștefan DUMITRIU,
3 S.C. Utilnavorep S.A.,
55 Aurel Vlaicu Bd., 900055, Constanta, Romania.
Email: cris.dum.stef@gmail.com

A. BĂRBULESCU,
Transilvania University of Brasov,
5 Turnului Str., 900152, Brasov, Romania.
Email: alina.barbulescu@unitbv.ro