Proc. of the 7th International Conference on Engineering and Emerging Technologies (ICEET)
27-28 October 2021, Istanbul, Turkey

Genetic Algorithm for Solving Coverage Issues in Undersea Mining

Ronald Ponguillo-Intriago
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Facultad de Ingenieria en Electricidad y Computacion
Escuela Superior Politecnica del Litoral, ESPOL
Guayaquil, Ecuador
RonaldAlberto.PonguilloIntriago@ugent.be

Daniel Ochoa
Facultad de Ingenieria en Electricidad y Computacion
Escuela Superior Politecnica del Litoral, ESPOL
Guayaquil, Ecuador
dochoa@espol.edu.ec

Angel J. Lopez
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Facultad de Ingenieria en Electricidad y Computacion
Escuela Superior Politecnica del Litoral, ESPOL
Guayaquil, Ecuador
Angel.Lopez@ugent.be

Ivana Semanjski
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Ivana.Semanjski@ugent.be

Sidharta Gautama
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Sidharta.Gautama@ugent.be

***Abstract*—To be cost-effective, robot-based undersea mining must comply several operational constraints. Among the main constraints are the time and energy required to extract the mineral from the seabed. It is also important to reduce the wear of the joints that connect the ship on the surface with the robot crawler that does the mining on the seabed, since this not only reduces operating costs, but also lengthens the useful life of these parts which increases system security. For this reason, the least amount of twisting in these pieces is preferable, so it is advisable to reduce the number of turns or changes of direction in the trajectory of the robot that extracts the mineral. In this article, we present an algorithm to optimize Coverage Path Planning using Genetic Algorithm to produce paths with longer segments, which can be used in underwater mining and reduce the effects the mentioned turning problem. The resulting paths have on average 55% less changes of directions in the trajectory than a GA with standard cost function. In addition, in tests made by placing small obstacles in a random way, 76% of useful paths were obtained and up to 59% of useful path when the obstacles were grouped into a single larger obstacle.**

Keywords—genetic algorithms, coverage path planning, deep sea mining, autonomous systems.

I. INTRODUCTION

Crawler robots have the potential of turning sub-sea mineral extraction operations into a highly lucrative business. Such robots collect minerals at the sea bottom using a number of mining tools such as specialised robotic arms or a cutter-suction dredgers. At great depths, the crawler is usually linked to a vessel by power and data wires to ensure its continuous operation. As the crawler surveys the mining area, the extracted material is pumped up to the vessel through flexible pipes. A crawler robot is one of mayor investment for a mining company. Therefore, it is imperative that the robot covers all mining sites with the least possible energy consumption and within its recommended operational parameters. This problem is equivalent to the well-known Travelling Salesman Problem (TSP). Where the mining area is represented by a regular grid and each node is labelled either as mining-place or an obstacle. Then, the shortest path across all mining-place nodes is computed.

For efficient undersea mining operations, two additional constrains must be satisfied. First, the crawler should visit each mining-place node exactly once to reduce energy consumption. Second, the crawler should avoid sharp turns to reduce the possibilities of mechanical failures or disconnection of tethered cables. Reducing the number of changes in direction of the crawler also reduces the frequency of junctions maintenance and replacement.

TSP solving approaches can be classified into: exact, heuristic, and meta-heuristic methods. Since, TSP is an NP-hard type problem with complexity $O((n-1)!/2)$. As the number of nodes n increases exact methods tend to converge to a solution at a much slower rate than other methods because the whole search space must be explored.

Metaheuristic algorithms although faster do not guarantee that the solution will be optimal. Nevertheless, in many applications a sub-optimal solution may be good enough. The most popular meta-heuristic algorithms used for TSP are: Particle Swarm Optimization (PSO)[1], Ant Colony Optimization (ACO)[2], Genetic Algorithms (GA)[3], Simulated Annealing (SA)[4] and Tabu Search (TS)[5]. A survey on methods for solving TSP can be found in [6][7].

In this work, we proposed a GA based technique to solve the TSP problem under the constrains mentioned before. The

978-1-6654-3897-1/21/$31.00 ©2021 IEEE

proposed cost function include terms that account for the length and smoothness of the crawler's path. The best candidates are used to generate new solutions until a suitable one is found. To account for changes in the crawler's trajectory, we built rules to detect sharp turns in a path. Our approach can be applied to other metaheuristic approaches to solve the TSP.

This paper is organized as follows. Section III explains the multi-objective cost function, which include the standard TSP function and our algorithm for detecting changes in robot trajectory. Then, a genetic operators are selected and tuned for optimization. To evaluate the performance of our proposed method, obstacles are added to the grid. Experimental results are presented in section IV and discussed in section V.

II. LITERATURE REVIEW

A Genetic Algorithm is an optimization method that uses meta-heuristic techniques based on the evolutionary process of species, i.e., it begins with an initial population that evolves after going through processes called selection, crossover, and mutation. The idea is to use a mechanism capable of qualifying individuals and choosing among them the best-scored ones using probabilistic methods.

Various types of crossover operators to solve TSP are available in [8] and [9]. After recombination, some of the new individuals are chosen to be mutated. The authors in [10] and [11] present specialized mutation operators to solve TSP. The crossover and mutation methods are applied with a certain probability of occurrence that is assigned when configuring the genetic algorithm.

To solve it, the coding of the individuals, based on permutations of all the nodes that are part of the search space is used. This technique exchanges the elements of the search space repeatedly to generate new individuals for obtaining a solution.

III. METHODOLOGY

A. *Genetic Operators: Mutation and Crossover*

Local Search Mutation (LSM) is a local search method that takes an individual and mutates it to generate diverse solutions in the population from this one on. This favour's the exploration of the search space and gets the algorithm out of local optimum if it were the case. In this work the three mutation methods combined in the LSM technique are flip, swap and slide that work as shown in the Fig. 1.

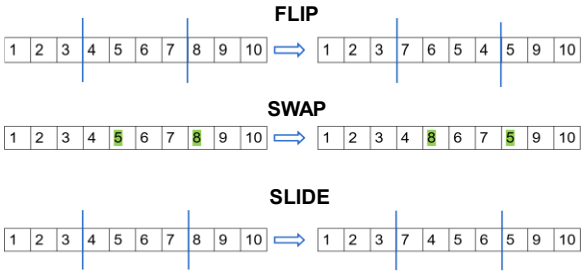


Fig. 1. Mutation operators implemented for LSM process

The Algorithm 1 presents the Order Based Crossover used in the solution proposed. Fig. 2 shows graphically the process of order-based crossover.

Algorithm 1. Order Based Crossover

1. Select a sublist of nodes from parent1.
2. Copy the sublist in the corresponding positions of the offspring.
3. From parent2, delete the points in the sublist.
4. The remaining points in parent2 are inserted into the offspring wherever necessary to complete the offspring.

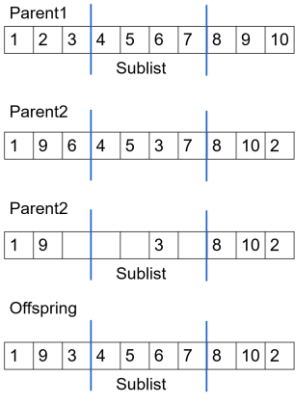


Fig. 2. Order Based Crossover graphically explained

B. *Tuning Genetic Algorithm*

Table I shows the summary of the parameters chosen for the GA. The improvement of the parameters was done based on the standard TSP mono-objective function which is the path length. For the genetic operators, we used the Order Based Crossover as stated in [12]. The mutation method used was the Local Search Mutation as explained in section III-A.

TABLE I. PARAMETERS OBTAINED AFTER THE TUNNING PROCESS FOR THE GENETIC ALGORITHM

Parameter	Description
Generations	1000
Individuals	100
Subpopulation	1
Generation Gap	4
Selection Method	Tournament
Recombination Method	Order Based Crossover
Mutation Method	Local Search Mutation
Recombination Rate	1
Mutation Rate	0.07

C. *Multi Objective Function*

The objective function used consists of two parts. The first objective presented in equation 1 is the length of the path generated by the GA and it is the same as defined in a TSP problem with Euclidean distance. The second component of the objective function in equation 2 is the number of nodes of the path where there is a change of direction in the path.

Let's define the components of the objective function somewhat formally.

Let consider a path $p = \{n_1, n_2, .., n_n\}$, where n_i is a point in a Cartesian plane, i.e., $n_i = (x_i, y_i)$.

Let define $v_k = (n_{k+1} - n_k), \forall k = 1, \dots, n-1$.

In the standard TSP the objective function corresponds to the Euclidean distance between nodes i and i+1, $\forall i = 1, \dots, n-1$ which is calculated as follow:

$$c_1 = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (1)$$

$$c_2 = \sum_{k=1}^{n-1} -sign\left(\left|\frac{\langle v_k, v_{k+1}\rangle}{\|v_k\|\|v_{k+1}\|}\right|\right) \quad (2)$$

As we are using a multi-objective function with two objectives, it is possible to use the Pareto frontier method[13][14][15] to divide the space of solutions into two areas from which the solutions could be chosen. On the other hand, there is the Weighted Sum Method [16][17], which we choose because simplifies the process by combining both costs into one, thus transforming a multi-objective cost function into a mono-objective function as is showed in the equation 3.

$$ObjF = A * c1 + B * c2 \quad (3)$$

where A + B = 1 and A and B ϵ(0,1]

Considering A+B=1, we can eliminate a variable and express equation 3 as:

$$ObjF = \alpha * c1 + (1 - \alpha) * c2 \quad (4)$$

D. *Experiment Organization*

Two types of experiments were designed to evaluate the performance of our algorithm. All experiments were evaluated on a regular grid, i.e. a Cartesian arrangement of nodes mounted on a two-dimensional plane. The distance between two consecutive nodes in the abscissa or ordinate ones is 1 and the diagonals between neighbouring nodes is $\sqrt{2}$. The α parameter is varied from 0.1 to 0.9 with steps of 0.1. Each experiment was run 100 times using the parameters showed in Table I.

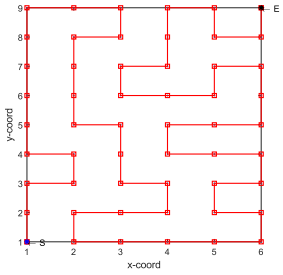
The first experiment was developed with a map without obstacles to evaluate for which values of α the genetic algorithm produces the best solutions. The results of the tests are shown in Fig. 5 and 6.

In the second experiment, noise (obstacles) is added to the map. Obstacles represent nodes on the map where the algorithm must not cross. These obstacles are placed in two ways. The first is as randomly distributed obstacles on the map simulating a salt and pepper noise effect and the second is by randomly placing one or more obstacle clusters made up of several nodes. A number of obstacle nodes of up to 5% of the map nodes was considered. This is to prove that our algorithm continues to work even in the presence of obstacles. The results of the tests are shown in Fig. 7.

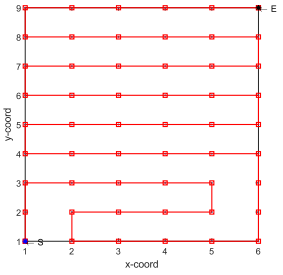
IV. RESULTS

The Fig. 3 shows typical solutions obtained when TSP is solved using genetic algorithms. The Fig. 3(a) shows an optimal solution when the TSP is solved with the objective function by default, that is, it only evaluates the length of the path. On the other hand, in the Fig. 3(b) it can be seen an optimal solution obtained with our algorithm in which is combined the standard objective function plus our objective function that detects and counts the changes of direction in the trajectory of the path. It can be seen how in both figures the generated paths have the same length (53) but in the Fig. 3(b) the sections are straighter. In the first case, the path produced has 34 nodes where there is a change in the direction of the path, while using our algorithm this number is 17 nodes.

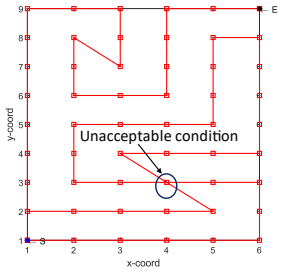
Finally, the Fig. 3(c) shows another possible output of the genetic algorithm that we qualify as an unfeasible solution since for our mining application having segment crossings in a path means being inefficient in the process because the crawler would pass through the same point more than once picking up the ore.



(a) Typical optimal solution using a standard TSP.

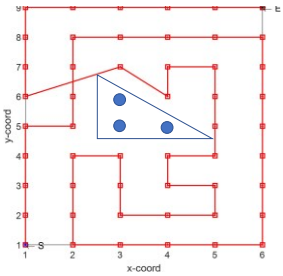


(b) Typical optimal solution using a TSP with our approach

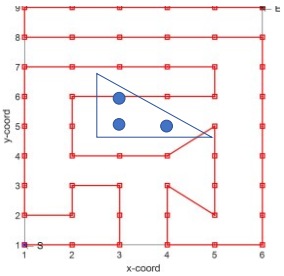


(c) Unfeasible solution for subsea mining applications

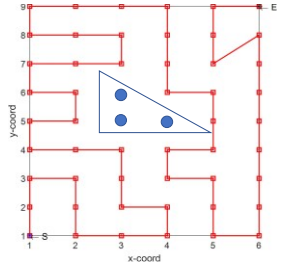
Fig. 3. Types of solution for Coverage Path Planning using Genetic Algorithm



(a) Solution with crossing segments



(b) Solution with obstacle space violation



(c) Accepted Good Solution

Fig. 4. Example of solutions obtained by our GA when obstacles are added in the form of clustered nodes

In Fig. 4 is shown an example with typical solutions produced by the GA when it solves a search space with a cluster of obstacles. The blue nodes represent the obstacle space. The Fig. 4(a) shows a solution that is discarded even though it does not invade the obstacle space because it produces a crossing of segments. In the Fig. 4(b) it can see the effect of invasion of the obstacle space. In this solution several long segments are produced, but instead there is an invasion of the obstacle space. The Fig. 4(c) shows a solution that is acceptable for our application. This solution has neither crossing of segments nor invasion of the obstacle space.

Since α represents the combination of the two objectives in the cost function, it is necessary to evaluate the characteristics of the solutions when α changes within the proposed interval. In this way, the distance obtained in the solutions, the number of nodes where there is a change in the direction of the trajectory (# Corners), the processing time and the number of generations used by the GA to converge to the solutions were evaluated. The Fig. 5 shows a summary of the results of the first experiment.

The Fig. 6 shows the percentage of each type of solution generated by the GA based on the parameter α. The red bar shows the unfeasible solutions, as defined in Fig. 3(c). The yellow bars correspond to solutions that, even having a lack of optimal length, present characteristics of very straight sections and do not have crosses in their segments. For this reason, it has been classified as sub-optimal and could be used in our applications. Finally, the green bars correspond to solutions that have optimal length and have long stretches in their trajectory, such as shown in Fig. 3(b).

The results of the second experiment are shown in Fig. 7. The blue bars correspond to the tests made on a map without obstacles. The orange bars shows the results obtained when the obstacles are randomly located and scattered within the map. Meanwhile, the purple bars show the results when the obstacles were grouped in a cluster within the map. This picture shows only those solutions that are acceptable for our application, and whose interpretation is presented in Fig. 4.

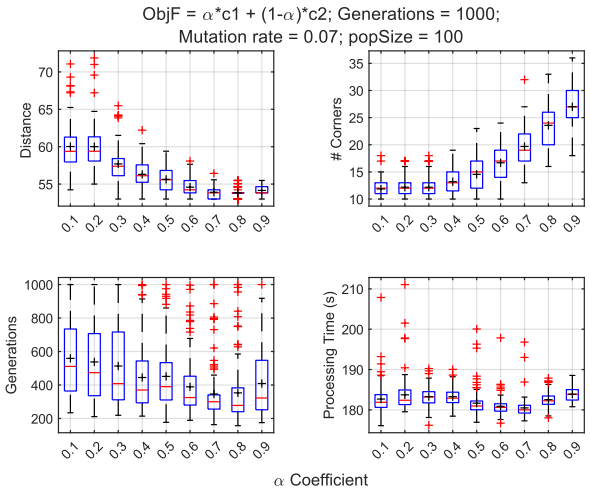


Fig. 5. GA performance in function of α coefficient

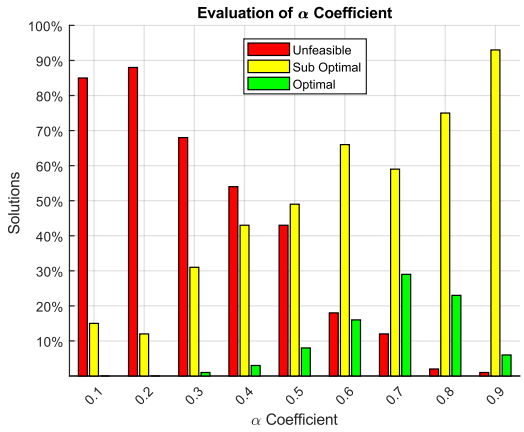


Fig. 6. α coefficient correlation with the quality of solutions

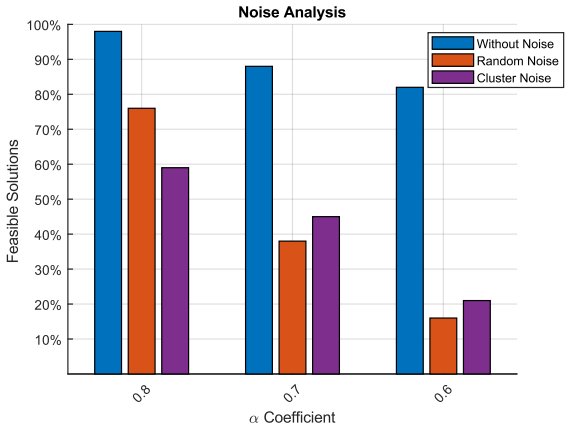


Fig. 7. Solutions performance analysis in presence of obstacles

V. DISCUSION

When solving the TSP in a regular grid it is possible to obtain several solutions that are optimal. Since TSP evaluates the length of the path, the optimal solutions will be those that, complying with the TSP rules, give the shortest length to travel by the traveler. Thus, in the Fig. 3(a) and 3(b) we see two optimal TSP solutions that differ in form. The difference between the two is that in the case of the traditional TSP, the solutions do not have diagonal sections or intersections of segments, but the optimizer does not care about changes in trajectory during the journey from the starting point to the goal, but only to meet the objective of get as little distance as possible.

On the other hand, the Fig. 3(b) shows an optimal solution in which the number of nodes where the path changes its trajectory occur has been reduced by half, which can be seen visually on longer paths, which it is desired for our underwater mining application.

In experiment 1, we tested different values of α in the objective function to determine which are the most useful values for our mining application. As you can see in the Fig. 5, the values of α that give minimum values of distance are $\alpha \geq 0.6$. As it is of interest to our application to reduce the number of direction changes in the trajectory at the same time, from the graph (#Corners) the number of direction changes in

the trajectory varies from 17 to 27 nodes on average for the interval of $0.6 \leq \alpha \leq 0.9$. Also, in the Fig. 5, for $0.6 \leq \alpha \leq 0.8$ the GA converges in fewer generations and needs less processing time. This processing time is between 180 and 185 seconds which is a reasonable time for the size of the search space with 54 nodes.

The combination of all this information leads us to choose values of α greater than or equal to 0.6 as a starting point for the following experiments. Also considering that for greater value of α the effect of eliminating changes of direction in the path disappears and with it the obtaining of straighter paths, therefore we will not consider the value of $\alpha = 0.9$. Based on the Fig. 6, we have for the values of α chosen in a scenario without obstacles, 82% feasible solutions are achieved for $\alpha = 0.6$, 88% for $\alpha = 0.7$ and 98% for $\alpha = 0.8$.

In the Fig. 7 the results of experiment 2 are shown. Here the count of the feasible solutions in the different obstacle conditions is summarized. The higher value of α for both types of obstacles, produce a greater number of feasible solutions. It is also observed that for $\alpha = 0.7$ and $\alpha = 0.8$ there is a better answer for the case of clustered obstacles. The latter is preferable since in real applications the obstacles are more like the case of cluster obstacles than to small, isolated obstacles.

VI. CONCLUSIONS

In this work, using genetic algorithms to solve a multi-objective TSP, was possible to generate good quality paths for undersea mining applications. For this type of application, it is highly desirable, reduce the number of direction changes in the robot's trajectory on the seabed, to limit the possible risks of damaging or wearing out the connection cable with the ship on the surface.

Using GA or any metaheuristics with the standard cost function for TSP, the resulting path in a regular grid can have a lot of changes in the direction of its trajectory even if the resulting path has an optimal value for the standard cost function. We were able to reduce up to 55% on average the changes on the path trajectory using our method.

Also, the developed algorithm to detect changes in the trajectory of the path, achieved 59% useful paths in the presence to up 5% of obstacles grouped in a cluster and, 76% if these obstacles are randomly scattered inside the map, showing robustness in our technique.

Although, our cost function is universal and can be applied with any metaheuristic, apply it with GA, limits its evaluation to a few hundred nodes in the search space.

In future works, this technique can be tested and extended with other optimizers that allow working with a greater number of nodes.

ACKNOWLEDGMENT

This research is supported by MarTERA, an ERA-NET Cofund scheme of Horizon 2020 European Commission. The overall goal of the ERA-NET Cofund MarTERA is to strengthen the European Research Area (ERA) in maritime and marine technologies, as well as Blue Growth. The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 728053-MarTERA. Also this work was supported in part funded by National Secretariat of Higher Education, Science, Technology and Innovation of Ecuador (SENESCYT).

REFERENCES

[1] Shi XH, Liang YC, Lee HP, Lu C, Wang Q, "Particle swarm optimization-based algorithms for TSP and generalized TSP," Information processing letters, vol. 103, no. 5, pp. 169-176, 2007.

[2] Hlaing, Zar Chi Su Su, and May Aye Khine, "Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm," International Journal of Information and Education Technology, vol. 1, no. 5, pp. 404, 2011.

[3] Yu, Y., Y. Chen, and T. Li, "Improved genetic algorithm for solving TSP," Control and decision, vol. 29, no. 8, pp. 1483-1488.

[4] Zhan, Shi-hua, Juan Lin, Ze-jun Zhang, and Yi-wen Zhong, "List-based simulated annealing algorithm for traveling salesman problem," Computational intelligence and neuroscience, vol. 2016.

[5] Y.-F. Lim, P.-Y. Hong, R. Ramli and R. Khalid, "An improved tabu search for solving symmetric traveling salesman problems," 2011 IEEE Colloquium on Humanities, Science and Engineering, pp. 851-854, 2011.

[6] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: a survey," Operations Research, vol. 16, no. 3, pp. 538-558, 1968.

[7] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," European Journal of Operational Research, vol. 59, no. 2, pp. 231-247, 1992.

[8] O. Abdoun and J. Abouchabaka, "A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem," arXiv preprint arXiv:1203.3097, 2012.

[9] W. M. Hameed and A. B. Kanbar, "A comparative study of crossover operators for genetic algorithms to solve travelling salesman problem," International Journal of Research-Granthaalayah, vol. 5, no. 2, pp. 284-291, 2017.

[10] O. Abdoun, J. Abouchabaka and C. Tajani, "Analyzing the performance of mutation operators to solve the travelling salesman problem," arXiv preprint arXiv:1203.3099, 2012.

[11] K. Deep and H. Mebrahtu, "Combined mutation operators of genetic algorithm for the travelling salesman problem," International Journal of Combinatorial Optimization Problems and Informatics, vol. 2, no. 3, pp. 1-23, 2011.

[12] M. F. Tasgetiren and A. E. Smith, "A genetic algorithm for the orienteering problem," Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), vol. 2, , pp. 910-915, IEEE, 2000

[13] D. Ščap, M. Hoić and A. Jokić, "Determination of the Pareto frontier for multiobjective optimization problem," Transactions of FAMENA, vol. 37, no. 2, pp. 15-28, 2013.

[14] B. Wilson, D. Cappelleri, T. W. Simpson and M. Frecker, "Efficient Pareto frontier exploration using surrogate approximations," Optimization and Engineering, vol. 2, no. 1, pp. 31-50, 2001.

[15] H. A. Abbass, R. Sarker and C. Newton, "PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems," Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546), vol. 2, pp. 971-978, IEEE, 2001.

[16] C. A. C. Coello, G. B. Lamont and D. A. Van Veldhuizen, "Evolutionary algorithms for solving multi-objective problems," vol. 5, Springer, 2007.

[17] Deb, Kalyanmoy, "Multi-objective optimisation using evolutionary algorithms: an introduction," Multi-objective evolutionary optimisation for product design and manufacturing, Springer, pp. 3-34, 2011.