SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ DERGİSİ

Sakarya University Journal of Science
SAUJS

e-ISSN 2147-835X | Period Bimonthly | Founded: 1997 | Publisher Sakarya University |
http://www.saujs.sakarya.edu.tr/en/

Title: A Comparative Study of Optimization Algorithms for Global Path Planning of Mobile Robots

Authors: Mustafa Yusuf YILDIRIM, Rüştü AKAY

Recieved: 2020-09-25 00:00:00

Accepted: 2021-02-22 00:00:00

Article Type: Research Article

Volume: 25
Issue: 2
Month: April
Year: 2021
Pages: 417-428

How to cite
Mustafa Yusuf YILDIRIM, Rüştü AKAY; (2021), A Comparative Study of Optimization Algorithms for Global Path Planning of Mobile Robots. Sakarya University Journal of Science, 25(2), 417-428, DOI: https://doi.org/10.16984/saufenbilder.800067
Access link
http://www.saujs.sakarya.edu.tr/en/pub/issue/60672/800067

Sakarya University Journal of Science 25(2), 417-428, 2021

A Comparative Study of Optimization Algorithms for Global Path Planning of Mobile Robots

Mustafa Yusuf YILDIRIM*[1], Rüştü AKAY[1]

Abstract

It is an essential issue for mobile robots to reach the target points with optimum cost which can be minimum duration or minimum fuel, depending on the problem. In this paper, it was aimed to develop a software for the optimal path planning of mobile robots in user-defined two-dimensional environments with static obstacles and to analyze the performance of some optimization algorithms for this problem using this software. The developed software is designed to create obstacles of different shapes and sizes in the work area and to find the shortest path for the robot using the selected optimization algorithm. Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) and Genetic Algorithm (GA) were implemented in the software. These algorithms have been tested for optimum path planning in four models with different problem sizes and different difficulty levels. When the results are evaluated, it is observed that the ABC algorithm gives better results than other algorithms in terms of the shortest distance. With this study, the use of optimization algorithms in real-time path planning of land mobile robots or unmanned aerial vehicles can be simulated.

Keywords: Mobile robot, path planning, cubic spline interpolation, optimization algorithms, simulation

1. INTRODUCTION

Path planning is performed for a mobile robot to determine the path it needs to track in order to reach its destination in an environment. With a successful planning system, mobile robots can access the desired point without any intervention. The primary purpose of optimization-based path planning for mobile robots is to find away from the start point to the target point without colliding any obstacles. Quality of the path affects all planning and the planned path should be feasible in terms of time and distance [1, 3-5].

In order for mobile robots to be used more efficiently, the distance between the start and target points must be covered in the shortest time and with the least cost without colliding the obstacles. For this purpose, many path planning algorithms have been developed for these robots [2]. Mobile robots reach the desired point most effectively with these algorithms [5].

*Corresponding author: myyildirim@erciyes.edu.tr
[1]Erciyes University, Faculty of Engineering, Mechatronics Engineering Department, 38039, Kayseri.
E-Mail: myyildirim@erciyes.edu.tr; akay@erciyes.edu.tr
ORCID: https://orcid.org/0000-0003-0302-8466; https://orcid.org/0000-0002-3585-3332

Some studies on the path planning for mobile robots in the literature can be explained as follows: Wang et al. firstly found the shortest path with the A Star algorithm and when they detected obstacles on this path, they reached the nearest safe point and optimized the path again with the PSO algorithm. [1]. Buniyamin et al. used an improved version of the Ant Colony Optimization algorithm for path planning [4]. Alajlan et al. used GA to solve the path planning problems in large-scale grid maps. [5]. Ajeil et al. used a hybridized PSO - Modified Frequency Bat algorithm for mobile robots in both static and dynamic conditions, minimizing the distance and fulfilling the path smoothness criteria. They also identified possible points that will be produced by a new Local Search algorithm integrated into this algorithm and converted into feasible solutions. [6]. Wang et al. developed a Multi-Objective PSO algorithm for path planning on rough terrain. They found that the proposed algorithm provides an advantage in obtaining Pareto optimal solutions. [7]. Zhang et al. developed a hybrid algorithm including Deep Learning, Ray Tracing, Waiting Rule, and Rapidly-Exploring Random Tree algorithms in known indoor environments. They compared the proposed algorithm with traditional and some intelligent algorithms in static and dynamic environments and proved their applicability. [8]. Dewang et al. developed the Adaptive PSO algorithm for the global path planning of mobile robots in environments with static obstacles. With this algorithm, the robot has reached the target point in a shorter time than the traditional PSO algorithm. [9]. Low et al. improved the performance of the Q-learning algorithm using the Flower Pollination algorithm because of the slow convergence rate of the Q-learning algorithm in the global path planning of mobile robots. With the proposed algorithm, the convergence of the Q-learning algorithm has been accelerated. [10]. Patle et al. developed The Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robots in both static and dynamic conditions. They found that the proposed control mechanism is optimal in terms of path and time compared to other navigation controls. [11]. Das et al. developed a version of the PSO algorithm used with evolutionary operators for path planning of multi-robot systems in known and complex environments. They have observed that the proposed algorithm performs better compared to other algorithms. [12]. Saeed et al. developed the Boundary Node Method for global path planning of mobile robots in static environments, and they found that the proposed method produced better results compared with other methods. [13]. Nazarahari et al. developed Artificial Potential Field and Enhanced GA algorithms for path planning of multi-robot systems. They used the first to identify all suitable paths, and the second to find the optimum path. They observed that the developed algorithm system performs better compared to other algorithms. [14]. Saraswathi et al. developed a hybrid version of the Cuckoo-Search and Bat Algorithm algorithms for the global path planning of mobile robots. They used the proposed algorithm in the conditions that environmental factors are unknown and found that the proposed algorithm takes less time to reach the target point than other algorithms. [15]. Rosas et al. developed a membrane evolutionary artificial potential field for global path planning problem in both static and dynamic environments. They observed that the proposed approach performed better compared to other methods. [16]. Bayat et al. developed the Electrostatic potential field approach for the global path planning of mobile robots. They tested the proposed approach in static and dynamic environments and proved its applicability. [17]. Qu et al. developed a grey wolf optimizer algorithm with reinforcement learning for path planning. The proposed algorithm has been proven to be successful in complex environments. [18]. Patle et al. used the Firefly Algorithm for mobile robot navigation in environments where environmental conditions change. They found that the proposed study produced better results compared to other intelligent navigation approaches. [19]. Song et al. used the Compact Cuckoo Search algorithm for the 3D path planning, and proposed a new Parallel Communication Strategy. The proposed method produced better results compared to other algorithms. [20]. Elhoseny et al. developed a Bézier curve based approach that uses Modified Genetic Algorithm for global path planning in

dynamic environments. The proposed approach is efficient method regarding the energy consumption of the robot in harsh environments. [21]. Patle et al. increased the performance of the Fuzzy Logic algorithm alone using the Probability and Fuzzy Logic algorithm for robot navigation. They found that this study produced better results compared to other navigation approaches. [22]. Goel et al. used the Glow-worm Swarm Optimization algorithm for 3D path planning. The proposed method has provided high convergence speed and accuracy compared to other heuristic algorithms. [23]. Li et al. developed a navigation system with a sensor network using a novel Artificial Potential Field algorithm. This method is used in environments with dynamic conditions. The tests and simulations proved the accuracy of this system. [24].

Unlike the studies in the literature, in this paper, a MATLAB-based generalizable simulation interface where different optimization algorithms are applied for global path planning has been designed and the performances of these optimization algorithms are evaluated in terms of shortest path and algorithm running time. This software will contribute to the literature by guiding in simulating a desired environment in which a robot moves, determining the best performanced optimization algorithm and the optimum path for real-time applications. In addition, this interface is designed not only for the optimization algorithms used in this paper, but also to be easily integrated for other methods used for path planning in the literature.

The rest of paper is organized as follows: In the second section, the optimization algorithms used in this paper are briefly explained. In the third chapter, material and method are presented, and the fourth section shows the results of the simulation. The fifth part is dedicated the conclusion.

2. OPTIMIZATION ALGORITHMS

2.1. Particle Swarm Optimization

Particle swarm optimization algorithm is a swarm-based optimization algorithm developed by Eberhart and Kennedy in 1995 [25]. This algorithm was developed inspired by the social behaviours of birds and fishes. The algorithm is often used for numerical problems. The basic steps of the algorithm are shown in Algorithm 1.

Algorithm 1 The basic steps of the PSO algorithm [25]

1: Initialize the control parameters of algorithm
2: Generate initial positions and velocities of particles
3: **While (stopping criteria not met)**
4: Evaluate fitness values
5: Determine the current best position for all particles
6: Update the global best position
7: Update the velocity and position of all particles
8: **End while**
9: Return the best solution

In this algorithm, the initial positions of the particles are generated randomly. Generally, the initial velocities of the particles are zero. The fitness values of the randomly generated positions are calculated using the objective function. The best position of each particle so far ($x_{best,i}$) and the best position of the population (g_{best}) are determined by their fitness values.

Using these positions, the velocities of the particles are updated. Equation 1 shows the velocity update equation.

$$v_i(t+1) = v_i(t) + r_1.c_1.\left(x_{best,i} - x_i(t)\right) + r_2.c_2.\left(g_{best} - x_i(t)\right);\ \ i = 1, \dots, P \quad (1)$$

where $v_i(t+1)$ is the current velocity of the particle i, $v_i(t)$ is the old velocity of the particle i, $x_i(t)$ is the old position of the particle i, r_1 and r_2 are random numbers generated between (0, 1), c_1 and c_2 are learning coefficients, P is the number of particles (size of the population). Positions of the particles are updated using this updated velocity. Equation 2 shows the position update equation.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where $x_i(t+1)$ the current position of the particle i. The best solution is stored in memory after velocity and position update, and the iterative

process continues until the stop criterion is met [26].

2.2. Artificial Bee Colony

Artificial bee colony algorithm is a swarm-based optimization algorithm developed by Karaboga in 2005 [25]. This algorithm has been developed by modeling bees' foraging behaviors. The basic steps of the algorithm are shown in Algorithm 2. In this algorithm, initial food sources are generated randomly using Equation 3.

Algorithm 2 The basic steps of the ABC algorithm [25]

1: Initialize the control parameters of algorithm
2: Generate initial population
3: **While (stopping criteria not met)**
4: Apply employee bee phase to generate new food sources
5: Calculation of probabilities according to the information from the employee bees.
6: Apply onlooker bee phase to generate new food sources
7: Abandonment of consumed resources and generating scout bees (scout bees phase)
8: **End while**
9: Return the best solution

$$x_{i,j} = x_j^{min} + rand(0,1)\,.\,\left(x_j^{max} - x_j^{min}\right);\quad i = 1,\dots,SN;\ j = 1,\dots,D \tag{3}$$

where is food source, SN is food source number, D is number of parameters to be optimized, x_j^{min} is the lower limit of the parameter j and x_j^{max} is the upper limit of the parameter j. Employed bees are dispatched to food sources. It determines the new food source adjacent to the source it works and evaluates its quality. The equations used at this stage are shown in Equations 4, 5 and 6.

$$v_{i,j} = x_{i,j} + \varphi_{i,j}.\left(x_{i,j} - x_{k,j}\right);\ k = 1,\dots,SN \tag{4}$$

$$v_{i,j} = \begin{cases} x_j^{min} & ;\ v_{i,j} < x_j^{min} \\ v_{i,j} & ;\ x_j^{max} < v_{i,j} < x_j^{max} \\ x_j^{max} & ;\ v_{i,j} > x_j^{max} \end{cases} \tag{5}$$

$$fit_i = \begin{cases} 1/(1+f_i) & ;\ f_i \geq 0 \\ 1 + abs(f_i) & ;\ f_i < 0 \end{cases};\ f_i = f\left(v_{i,j}\right) \tag{6}$$

where $v_{i,j}$ is a new food source, $\varphi_{i,j}$ is a random number generated between [-1, 1], $x_{k,j}$ is the food source randomly selected for the parameter j and fit_i is fitness value of the source $v_{i,j}$. The greedy selection process is applied according to this fitness value. If the new solution is better, replaces the old solution with this new solution. Otherwise, a counter is generated for the old solution, and this counter increases by one. By using fitness values, probabilities to be used by the onlookers in the selection are calculated as in Equation 7.

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{7}$$

where p_i is the probability of selecting the source i. By using these values, a random number is generated in the range of [0, 1] according to the roulette wheel, and if the value p_i is greater than this random number, the scouts produce a new solution using Equation 4. If this new solution is better, replaces with the old solution with this new solution and the counter generated for the old solution is reset. Otherwise, the counter increases by one. At the end of the iteration, the counters are checked. The resources which their counters are above a certain threshold (limit) are abandoned, and a new food source is searched. The best solution is stored in memory, and the iterative process continues until the stop criterion is met [25].

2.3. Genetic Algorithm

The genetic algorithm is an evolutionary algorithm based on natural selection and genetic laws developed by Holland in 1975 [25]. The basic steps of the algorithm are shown in Algorithm 3. In this algorithm, the initial population is generated randomly and then, the fitness values of these solutions are calculated. The probability of survival of solutions is calculated based on their fitness values. By generating a random number, it is determined which solutions will survive. This process is called selection stage.

Parents are randomly selected from surviving solutions for crossover. A random number is generated, and if this number is smaller than the

crossover rate, the relevant parents are crossed to find new solutions. The crossover rate is usually chosen from 0.6 to 1. After crossover, each solution is subjected to mutation. Another random number is generated and if this number is smaller than the mutation rate, some bits of the relevant solution will change. The mutation rate is usually chosen from 0.001 to 0.1. The best solution is stored in memory, and the iterative process continues until the stop criterion is met [27].

Algorithm 3 The basic steps of the GA [25]

1: Initialize the control parameters of algorithm
2: Generate initial population
3: **While (stopping criteria not met)**
4: Calculation of fitness values
5: Selection Stage
6: Crossover Stage
7: Mutation Stage
8: **End while**
9: Return the best solution

3. MATERIAL AND METHODS

3.1. Simulation Software

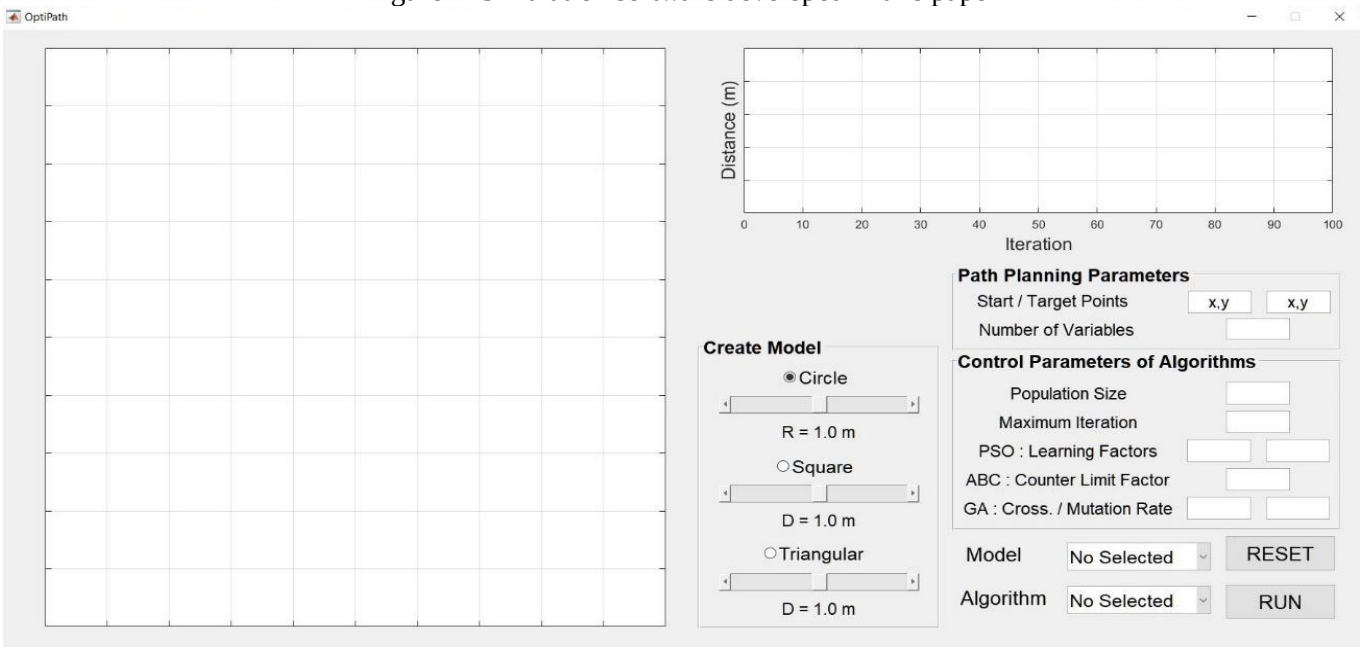
The The simulation software was designed through the MATLAB GUI. The software shown in Figure 1 includes environment screen, convergence screen, model creation interface, model and algorithm selections, path planning parameters, and control parameters of algorithms.

On the environment screen, start and target points of the mobile robot, obstacles and planned optimum paths are shown. The environment screen has an area of 100 m2. The convergence screen shows the convergence graph of the algorithm used at the end of each global path planning. This graph represents the shortest distance in meters. In the model creation interface, obstacles are created in the form of circles, squares and equilateral triangles. Obstacles create using the mouse left click. In this paper, the radius of the circle, an edge length of the square and triangle are defined as sizes, and the size of each shape can be selected with a step length of 0.1 meters between [0.2, 2] meters. In path planning parameters, the start and target points of the robot and the number of parameters can be determined. In control parameters of algorithms, population size and the maximum number of iterations can be determined. Also, learning coefficients for PSO, the limit for ABC, crossover and mutation rates for GA can be used as inputs. In the selection of the model, there are four different models. When one of these models is selected, it is displayed on the environment screen. In the algorithm selection, one of GA [27], PSO [28] and ABC [29] algorithms is selected. The RUN button is used to run the path planning optimization, and the RESET button is used to initialize the system.

3.2. Problem Formulation

The optimum path of the mobile robot is planned with cubic spline interpolation in this paper. Some points are required for this purpose. The number of these points is expressed as the number of parameters (d). The solution is determined as the coordinates of these points. In each iteration, these points are transferred to an array with start and target points. This array is interpolated using the cubic spline function and q query points. The interpolated path contains n points and these points are expressed as P_i.

Figure 1 Simulation software developed in this paper



The objective function used in the paper is composed of two parts. The first calculates the length of the path, and the second calculates the feasible distance between the robot and obstacles. The objective function to be optimized is shown in Equation 8.

$$\min\ F(P_i, O_j) = L(P_i).\left[1 + \beta.V(P_i, O_j)\right];\quad i = 1, \dots, q;\ j = 1, \dots, o \tag{8}$$

where O_j is the location of obstacles, $L(P_i)$ is path length function, β is obstacle violation factor, $V(P_i, O_j)$ is violation function, o is the number of obstacles. Violation function is required to calculate the feasible distance between the robot and obstacles. The path length function is shown in Equation 9, and the violation function in Equation 10.

$$L(P_i) = \sum_{i=1}^{n-1} \sqrt{(P_{ix} - P_{(i+1)x})^2 + (P_{iy} - P_{(i+1)y})^2} \tag{9}$$

$$V(P_i, O_j) = \frac{1}{n}\sum_{i=1}^{o}\left[\sum_{j=1}^{n} \max\left(1 - \frac{\sqrt{(P_{ix} - O_{jx})^2 + (P_{iy} - O_{jy})^2}}{a_j}\right)\right] \tag{10}$$

where a_j is the radius if the obstacle j is circle, half the length of the diagonal if the obstacle j is square, and the height if the obstacle j is triangle [30].

4. ANALYSIS OF RESULTS

The simulation software was developed using the MATLAB 2019a programming language and was run in a computer of the Windows 10 operating system with an INTEL CORE i7 processor and 16 GB of RAM. In the software, the start point is symbolized by a square mark and the target point by a star mark. The shortest distance without obstacle is 14.142 meters. Limit values of the problem are determined as [0.5 9.5] for x and y. The obstacle violation factor β is 100, the query point for the cubic spline interpolation q is 100. In the ABC algorithm, the counter limit value changes according to population size and the number of parameters. In the GA algorithm, the selection process is carried out by the roulette wheel method, and the crossing process is performed by the one-point crossing method. The mutation process is carried out by the value coding method. Random numbers r_1 and r_2 in PSO algorithm, random number $\varphi_{i,j}$ in ABC algorithm, random numbers used for selection/crossover/mutation operations in the GA algorithm are generated differently in each iteration.

PSO, ABC and GA optimization algorithms have been applied in four different models in the software. The number of parameters was

determined as 2, 3, 4, 5, 6, and 8. Algorithms for each model and number of parameters were run with 30 runs. Control parameters of algorithms are shown in Table 1, the models in the software in Figure 2, parameters of obstacles in Table 2, the sample convergence graphs for each model and number of parameters in Figures 3, 4, 5 and 6, average shortest distances, average CPU times and success rates in problem solving (path planning with obstacle avoidance) in Table 3.

Table 1
Control parameters of algorithms [25, 27, 31]

Control Parameters	PSO	ABC	GA
Number of Iteration	100	100	100
Population Size (npop)	50	50	50
c_1 / c_2	2 / 2	-	-
Counter limit Value	-	(0.5).(npop).(d)	-
Crossover / Mutation Rates	-	-	0.98 / 0.1

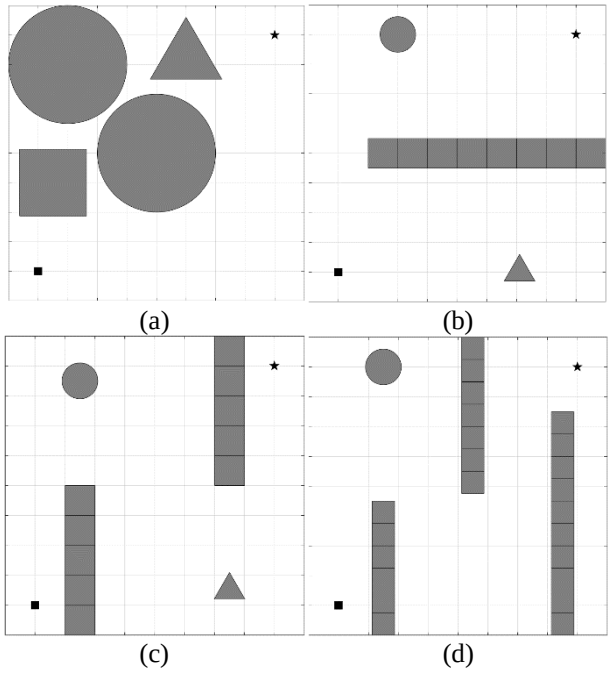


Figure 2 The models in the software:
(a) Model 1, (b) Model 2, (c) Model 3, (d) Model 4

Table 2
Parameters of obstacles: number (size (m))

Obstacle	Model 1	Model 2	Model 3	Model 4
Circle	2 (2)	1 (0.6)	1 (0.6)	1 (0.6)
Square	1 (2.25)	8 (1)	10 (1)	23(0.75)
Triangle	1 (2.4)	1 (1)	1 (1)	-

Considering the results, the fastest running algorithm was determined as GA according to the CPU times in Table 3. When the shortest distance values in the same table are examined, the ABC algorithm has shown the best performance in all models. Considering the convergence graphics of all models, although the convergences of the algorithms are close to each other in all models, it can be said that the ABC algorithm converges slightly faster than the other algorithms examined in this study for the best number of parameters calculated. When the success rates of problem solving are evaluated, the ABC algorithm has also performed best. However, it can be said that the problem solving ability of algorithms is weakened as the environment models become more difficult. Difficulty in problems causes algorithms to plan paths by ignoring obstacles.

When the success rate in Table 3 is examined, it is seen that the algorithms cannot solve the problem in some of 30 runs in difficult models. Besides, it is seen that the number of parameters in difficult models is effective in the success rate of problem solutions. In easy models such as Model 1, 2 and 3, the optimum number of parameters is 2, while in Model 4, which is a difficult model, the optimum number of parameters is 3. Apart from this, it is observed that the path with the shortest distance cannot be planned in cases where the number of parameters is higher. The sample paths found by the algorithms using the optimum number of parameters are shown in Figure 7.

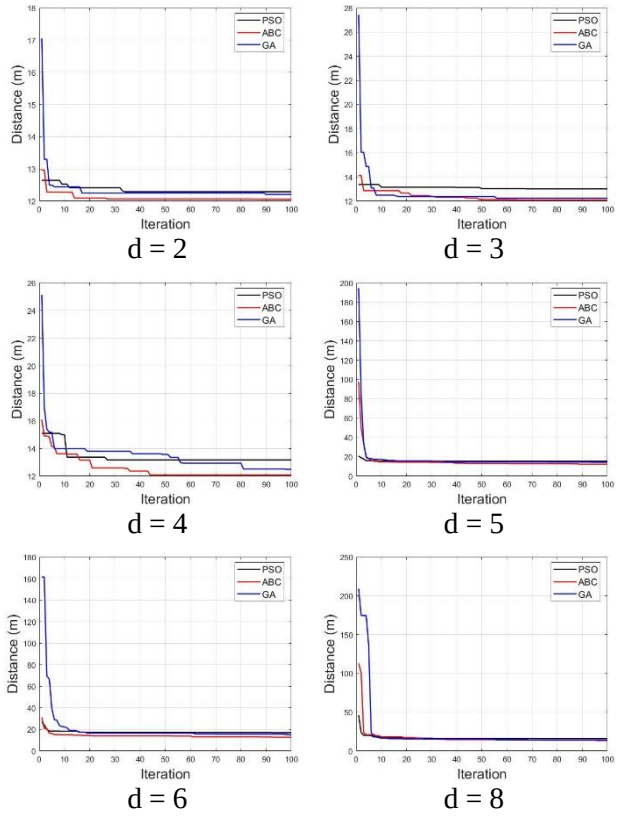


Figure 3 The sample convergence graphs for Model 1

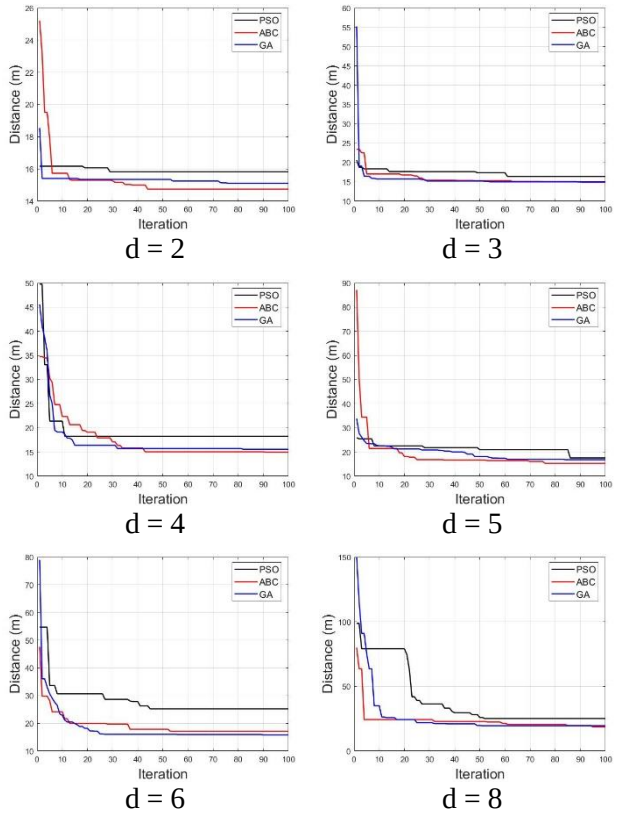


Figure 5 The sample convergence graphs for Model 3

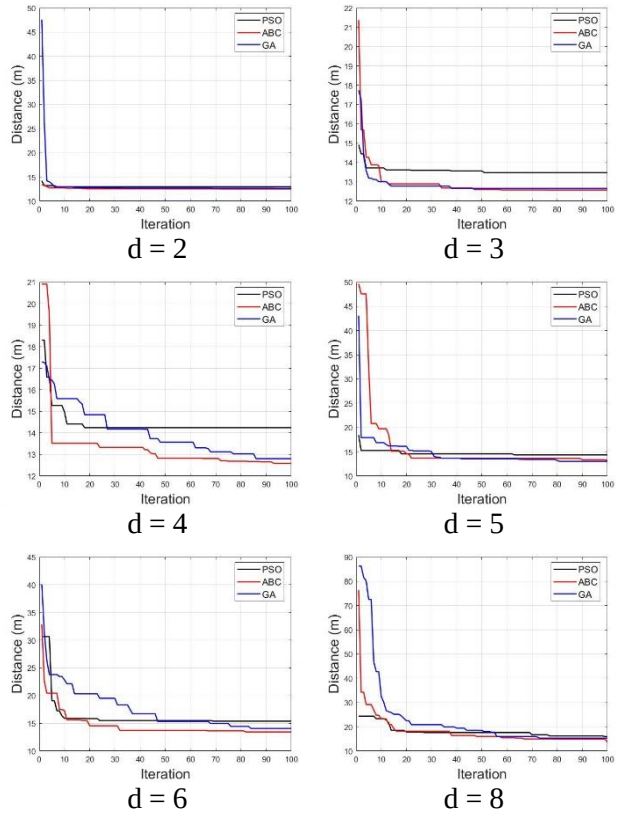


Figure 4 The sample convergence graphs for Model 2

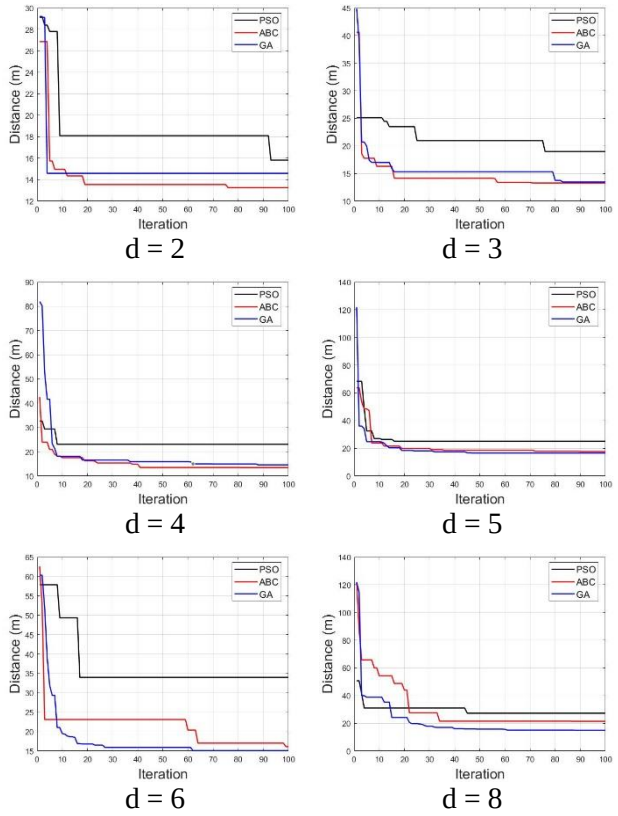


Figure 6 The sample convergence graphs for Model

Table 3
Average shortest distances, average CPU times and success rates in problem solving for each model and number of parameters (These results are average of 30 runs.)

d	Alg.	Model 1			Model 2			Model 3			Model 4		
		Shortest Distance (m)	**CPU Times (s)**	**Success Rate**	**Shortest Distance (m)**	**CPU Times (s)**	**Success Rate**	**Shortest Distance (m)**	**CPU Times (s)**	**Success Rate**	**Shortest Distance (m)**	**CPU Times (s)**	**Success Rate**
2	**PSO**	12.2516	13.0671	30/30	12.7961	13.0696	30/30	16.1726	13.9323	30/30	23.5283	16.3769	10/30
	ABC	**12.0513**	13.1331	**30/30**	**12.4981**	13.2297	**30/30**	**14.7748**	13.8110	**30/30**	14.8430	16.4044	30/30
	GA	12.1320	10.2890	30/30	12.5950	9.9604	30/30	15.1270	10.7339	30/30	19.4570	12.3910	23/30
3	**PSO**	12.8807	13.7011	30/30	13.3248	13.2030	30/30	18.6413	14.0377	25/30	20.4702	16.0140	13/30
	ABC	12.0766	13.4714	30/30	12.5489	13.2239	30/30	14.8989	14.0964	30/30	**13.4868**	16.2611	**30/30**
	GA	12.3257	10.4291	30/30	12.7055	9.8898	30/30	17.2975	10.7320	28/30	16.4833	12.9633	25/30
4	**PSO**	14.0715	13.7768	30/30	14.2845	13.2504	30/30	19.0725	15.0044	26/30	21.9463	16.5527	13/30
	ABC	12.1255	12.7223	30/30	12.6628	12.6753	30/30	15.0692	13.7729	30/30	14.3990	16.4541	30/30
	GA	12.6411	9.4862	30/30	13.0401	9.9745	30/30	17.4845	10.9125	27/30	17.9373	12.8952	20/30
5	**PSO**	15.1391	13.4782	30/30	15.4226	13.2664	29/30	20.7300	14.0262	25/30	23.5203	15.6928	11/30
	ABC	12.3318	13.4728	30/30	12.9595	13.1346	30/30	15.6299	14.1110	30/30	14.8681	16.2983	30/30
	GA	12.2479	10.4494	30/30	13.6584	10.0040	29/30	17.3574	10.8266	26/30	17.6135	12.8570	22/30
6	**PSO**	16.7272	13.8275	30/30	16.6338	13.3001	29/30	25.3959	14.0232	25/30	23.7252	16.4367	11/30
	ABC	12.7744	13.4735	30/30	13.5077	12.9580	30/30	16.3155	14.0974	30/30	16.2651	16.4173	28/30
	GA	13.5840	10.3748	30/30	14.2737	10.1366	29/30	18.8168	10.8185	25/30	19.4753	12.9667	13/30
8	**PSO**	18.1445	11.9220	30/30	18.8914	13.2407	26/30	29.2721	14.1039	27/30	28.1395	16.2067	6/30
	ABC	14.3306	11.5610	30/30	15.7884	13.0625	30/30	18.6938	13.7123	30/30	20.7455	16.5368	25/30
	GA	14.9606	9.1455	30/30	17.1932	10.0974	24/30	22.3556	10.2123	23/30	19.8974	12.9854	15/30

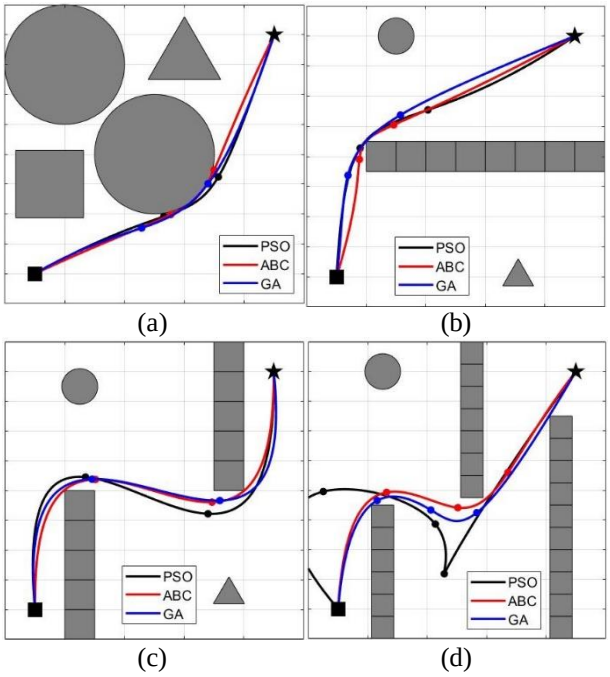


Figure 7 The sample paths found by the algorithms using the optimum number of parameters: (a) Model 1, (b) Model 2, (c) Model 3, (d) Model 4

5. CONCLUSION

In this paper, a software was developed for path planning of mobile robots in static environments, and the performances of different optimization algorithms were evaluated. Four model of different difficulty levels were used for the evaluation. As a result, while the fastest running algorithm is GA, the best performing algorithm is ABC. However, increasing the problem size (number of parameters in path planning) and making the models more difficult caused deterioration in problem solving abilities of these algorithms.

In future works, global paths can be planned for multi-robots in static environments, different approaches can be proposed to improve the performance of the algorithms, or global paths can be planned for single and multi-robots in the environments including static and dynamic obstacles.

The Declaration of Conflict of Interest/ Common Interest

No conflict of interest or common interest has been declared by the authors.

Authors' Contribution

Mustafa Yusuf YILDIRIM contributed to the conceptualization of this study, methodology, software, data improvement, visualization and writing a draft document. Rüştü AKAY contributed in supervision, methodology, reviewing and editing the draft document.

The Declaration of Ethics Committee Approval

The authors declare that this document does not require an ethics committee approval or any special permission.

The Declaration of Research and Publication Ethics

The authors of the paper declare that they comply with the scientific, ethical and quotation rules of SAUJS in all processes of the article and that they do not make any falsification on the data collected. In addition, they declare that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science.

REFERENCES

[1] Y. Wang, F. Cai, and Y. Wang, “Dynamic Path Planning for Mobile Robot Based on Particle Swarm Optimization,” AIP Conference Proceedings 1864, 020024, pp. 1–4, 2017.

[2] M. E. Dere, “Optimum path planning for mobile robots,” MS Thesis, Konya Technical University, 2019.

[3] E. Bogar, “A Hybrid Optimization Method for Single and Multi-Objective Robot Path Planning Problem,” MS Thesis, Pamukkale University, 2016.

[4] N. Buniyamin, N. Sariff, W. A. J. Wan Ngah and Z. Mohamad, "Robot Global Path Planning Overview and A Variation of Ant Colony System Algorithm," International Journal of Mathematics and Computers in Simulation, vol. 1, no. 5, pp. 9–16, 2011.

[5] M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur and A. Ammar, "Global Path Planning for Mobile Robots in Large-Scale Grid Environments using Genetic Algorithms," International Conference on Individual and Collective Behaviors in Robotics (ICBR), pp. 1–8, 2013.

[6] F. H. Ajeil, I. K. Ibraheem, M. A. Sahib and A. J. Humaidi, "Multi-Objective Path Planning of an Autonomous Mobile Robot using Hybrid PSO-MFB Optimization Algorithm," Applied Soft Computing, vol. 89, 106076, pp. 1–13, 2020.

[7] B. Wang, S. Li, J. Guo and Q. Chen, "Car-Like Mobile Robot Path Planning in Rough Terrain using Multi-Objective Particle Swarm Optimization Algorithm," Neurocomputing, vol. 282, pp. 42–51, 2018.

[8] L. Zhang, Y. Zhang and Y. Li, "Path Planning for Indoor Mobile Robot Based on Deep Learning," Optics, vol. 219, 165096, pp. 1–17, 2020.

[9] H. S. Dewang, P. K. Mohanty and S. Kundu, "A Robust Path Planning for Mobile Robot using Smart Particle Swarm Optimization," Procedia Computer Science, vol. 133, pp. 290–297, 2018.

[10] E. S. Low, P. Ong and K. C. Cheah, "Solving The Optimal Path Planning of a Mobile Robot using Improved Q-Learning," Robotics and Autonomous Systems, vol. 115, pp. 143–161, 2019.

[11] B. K. Patle, D. R. K. Parhi, A. Jagadeesh and S. K. Kashyap, "Matrix-Binary Codes Based Genetic Algorithm for Path Planning of Mobile Robot," Computers & Electrical Engineering, vol. 67, pp. 708–728, 2018.

[12] P. K. Das and P. K. Jena, "Multi-Robot Path Planning using Improved Particle Swarm Optimization Algorithm through Novel Evolutionary Operators," Applied Soft Computing, vol. 92, 106312, pp. 1–24, 2020.

[13] R. A. Saeed, D. R. Recupero and P. Remagnino, "A Boundary Node Method for Path Planning of Mobile Robots," Robotics and Autonomous Systems, vol. 123, 103320, pp. 1–21, 2020.

[14] M. Nazarahari, E. Khanmirza and S. Doostie, "Multi-Objective Multi-Robot Path Planning in Continuous Environment using an Enhanced Genetic Algorithm," Expert Systems with Applications, vol. 115, pp. 106–120, 2019.

[15] M. Saraswathi, G. B. Murali and B. B. V. L. Deepak, "Optimal Path Planning of Mobile Robot using Hybrid Cuckoo Search-Bat Algorithm," Procedia Computer Science, vol. 133, pp. 510–517, 2018.

[16] U. O. Rosas, O. Montiel and R. Sepúlveda, "Mobile Robot Path Planning using Membrane Evolutionary Artificial Potential Field," Applied Soft Computing, vol. 77, pp. 236–251, 2019.

[17] F. Bayat, S. S. Najafinia and M. Aliyari, "Mobile Robots Path Planning: Electrostatic Potential Field Approach," Expert Systems with Applications, vol. 100, pp. 68–78, 2018.

[18] C. Qu, W. Gai, M. Zhong and J. Zhang, "A Novel Reinforcement Learning Based Grey Wolf Optimizer Algorithm for Unmanned Aerial Vehicles (Uavs) Path Planning," Applied Soft Computing, vol. 89, 106099, pp. 1–12, 2020.

[19] B. K. Patle, A. Pandey, A. Jagadeesh and D. R. Parhi, "Path Planning in Uncertain Environment by using Firefly Algorithm," Defence Technology, vol. 14, no. 6, pp. 691–701, 2018.

[20] P. C. Song, J. S. Pan and S. C. Chu, "A Parallel Compact Cuckoo Search Algorithm for Three-Dimensional Path Planning," Applied Soft Computing, vol. 94, 106443, pp. 1–16, 2020.

[21] M. Elhoseny, A. Tharwat and A. E. Hassanien, "Bezier Curve Based Path Planning in A Dynamic Field using Modified Genetic Algorithm," Journal of Computational Science, vol. 25, pp. 339–350, 2018.

[22] B. K. Patle, D. R. K. Parhi, A. Jagadeesh and S. K. Kashyap, "Application of Probability to Enhance the Performance of Fuzzy Based Mobile Robot Navigation," Applied Soft Computing, vol. 75, pp. 265–283, 2019.

[23] U. Goel, S. Varshney, A. Jain, S. Maheshwari and A. Shukla, "Three Dimensional Path Planning for UAVs in Dynamic Environment using Glow-Worm Swarm Optimization," Procedia Computer Science, vol. 133, pp. 230–239, 2018.

[24] H. Li and A. V. Savkin, "An Algorithm for Safe Navigation of Mobile Robots by a Sensor Network in Dynamic Cluttered Industrial Environments," Robotics and Computer-Integrated Manufacturing, vol. 54, pp. 65–82, 2018.

[25] D. Karaboga, "Artificial Intelligence Optimization Algorithms," Nobel Publishing, 2017.

[26] A. Ayari and S. Bouamama, "A New Multiple Robot Path Planning Algorithm: Dynamic Distributed Particle Swarm Optimization," Robotics and Biomimetics, vol. 4, no. 8, pp. 1–15, 2017.

[27] A. Altay, O. Ozkan and G. Kayakutlu, "Prediction of Aircraft Failure Times using Artificial Neural Networks and Genetic Algorithms," Journal of Aircraft, vol. 51, no. 1, pp. 47–53, 2014.

[28] M. K. Heris, Particle Swarm Optimization in MATLAB (URL: https://yarpiz.com/50/ypea102-particle-swarm-optimization), 2015.

[29] M. K. Heris, Artificial Bee Colony in MATLAB (URL: https://yarpiz.com/297/ypea114-artificial-bee-colony), 2015.

[30] E. Chołodowicz and D. Figurowski, "Mobile Robot Path Planning with Obstacle Avoidance using Particle Swarm Optimization," Pomiary Automatyka Robotyka, vol. 21, no. 3, pp. 59–68, 2017.

[31] Y. He, W. J. Ma and J. P. Zhang, "The Parameters Selection of PSO Algorithm Influencing on Performance of Fault Diagnosis," MATEC Web of Conferences 63-02019, pp. 1–5, 2016.