Multi-objective Grasshopper Optimization Algorithm for Robot Path Planning in Static Environments

Zahra Elmi
Autonomous Systems Laboratory
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
E-mail: Zahra.elmi@gmail.com

Mehmet Önder EFE
Autonomous Systems Laboratory
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
E-mail: onderefe@hacettepe.edu.tr

Abstract—**Finding the most appropriate path in robot navigation has been an interesting challenge in recent years. A number of different techniques have been proposed to address this problem. Heuristic methods are one of them that have been efficiently used in many complex and multi-dimensional optimization problems. In this paper, we present a new algorithm for robot path planning in a static environment. The main aim is to use a multi objective method to minimize several metrics such as cost, distance, energy or time. Distance, path smoothness and robot path planning time is optimized in the current work. The contribution of this work is to calculate an appropriate fitness function at each iteration to achieve the best solution. The obtained result is compared with the Particle Swarm Optimization (PSO) algorithm. The proposed algorithm displays better performance characteristics in terms of time and path smoothness than PSO algorithm and the obtained path lengths are shorter than those obtained with PSO.**

Keywords—Path Planning, Mobile Multi-Robot, Grasshopper Optimization Algorithm, Obstacle Avoidance, Static Environment.

I. Introduction

Mobile robots are used in many different fields such as industry, military, medicine, space exploration, architecture and agriculture. Due to the limits in human capabilities and high precision in robot movements and ability to work in an environment that is harmful to human health or to prevent human error in repetitive and tasks, the use of robots is inevitable. The most important stage in mobile robot applications is the path planning. In recent years, this problem has attracted remarkable attention of many researchers.

In [1], a real time navigation system using electronic chart display is used to improve ant colony algorithm for path planning. To obtain a suitable navigation system, chart display, display, navigation operations, route calculation and navigation marks are all integrated. The aim in [1] is to design safe, collision-free and shortest path navigation. This has been done with improved basic ant colony to get a good selection strategy of ants and pheromone update strategy by using adaptive pseudo random selection rules. In [2], a global path planning method using ant colony algorithm in an unknown environment is introduced. The global information of environment as target attraction function is used to improve the selection probability of the optimal path from start position to target. Also, to overcome local optimum problem a new updating rule such as the rule of wolf colony is used. In [3], a heuristic dynamic method for online path planning is suggested. In this method, model learning is used in a Markov decision process. Furthermore, the authors in [3] claim that the proposed method has a significant advantage to enhance efficiency of environment model for multi robot cooperation. In [4], for surveillance missions, a path planning algorithm based on Kalman filter is proposed. The main aim is to find a set of commands for network for minimizing a cost function in three phases. First one is using Kalman filter and Bayesian network for target tracking. Then, the object function is defined according to the relative position between aircraft and target. Finally, a heuristic method to find the set of commands for network is used. In [5], two new methods of PSO algorithm with nonlinear inertia weight and simulated annealing PSO for path planning are used. The nonlinear inertia weight coefficients are used for global and local search accuracy. The local optimum problem is remedied by combining PSO with simulated annealing. In [6], max-min ant system algorithm is used for finding a desired path in an unknown environment. This system is similar to ant colony algorithm that finds the best solution set. The performance of the proposed algorithm is compared with genetic algorithms.

The aim of path planning for mobile robot is the ability of finding an optimal path from a starting point to a target point among obstacles with collision-free. To find an appropriate path, some of the parameters such as cost, distance, energy and time that play an important role in navigation of mobile robot must be optimized [5-6]. To solve the path planning problem, several methods have been proposed including classical methods and meta-heuristic algorithms. Since the path planning problem is a NP-Hard problem, the classical methods have to pay high computational costs and handle large computational complexities. Specially, it is complicated in largo scale and high degree of freedom problems [7-8]. Therefore, meta-heuristic methods have been proposed to overcome the problems arising with the classical methods. These methods are successfully used in many fields such as robotics and they are inspired from the natural phenomena, the evolutionary processes or swarm intelligence. These methods often used creatures, [9]. The fundamental goal of every creature is to survive. To achieve this goal, creatures have been adapting and evolving in different ways. These methods are classified into two main groups, namely, single-solution based and multi-solution based. In the former, a single random solution is generated and improved for a special problem. In the latter, for a given problem, multi-solution set is generated and enhanced. Besides, due to the improvement of multiple solutions during optimization, the multi solution based algorithms have higher capability of avoidance from getting trapped to local optima. Therefore, to escape from a local optimum, a modified solution around the local optimum can be used by other solutions. These solutions explore a larger subspace of the search space, so the probability of finding the global optimum is high [10]. Due to the exchange of the knowledge of search space between multi solution options, moving toward the global optimum is fast and the result is obtained quickly. Therefore, these optimization methods are used to solve difficult problems of path planning of robot. However, the obtained paths from these methods are not smooth and this is a problem to be remedied. In this paper, we consider a two dimensional (2D) map, which contains a set of obstacles. The initial position of the robot is known as well as the target position. The problem is to synthesize a path in between the two points according to given criteria.

This paper is organized as follows: The second section introduces the grasshopper optimization algorithm. The third section describes the PSO based solution of the problem in hand. The fourth section gives the simulation results and the conclusions constitute the last part of the paper.

II. Background of the Problem

A. The Grasshopper Optimization Algorithm

The grasshopper optimization algorithm (GOA) was inspired from the life cycle of grasshoppers, [9]. These creatures are seen individually in nature but they are considered as one of the largest swarms. The unique aspect of such a swarm is the swarming behavior that can be seen in both nymph and adulthood. The main characteristics of the swarm in the larval phase are slow movements and small steps for the young grasshoppers. The search space in the algorithm inspired from nature is divided into two parts, namely, exploration and exploitation. In the exploration, the agents that search are suddenly encouraged to move, but in exploitation they tend to move locally. These two functions and the target searching are fulfilled by the grasshoppers simultaneously. To simulate the swarming behavior of grasshopper, the following mathematical model is given,

$$X_{i+1} = S_i + G_i + A_i \tag{1}$$

where X_i is the position of i^{th} grasshopper, S_i is the social interaction quantity of the i^{th} grasshopper, G_i is the force of gravity on the i^{th} grasshopper and A_i is the wind advection.

$$S_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} s(d_{ij})\hat{d}_{ij} \tag{2}$$

where $\hat{d}_{ij} := \frac{x_j - x_i}{d_{ij}}$ is a unit vector between the i^{th} grasshopper and j^{th} grasshopper, d_{ij} is the distance between the i^{th} grasshopper and j^{th} grasshopper, s is a function that presents the strength of social forces designed as below.

$$s(r) = fe^{\frac{-r}{l}} - e^{-r} \tag{3}$$

where f is the attraction intensity and l is the scale of attraction length. There are social interactions between grasshoppers that can be defined as attraction and repulsion. As mentioned in [9], the distance is considered between 0 and 15 and repulsion occurs in the interval [0, 2.079]. If the distance of a grasshopper from one another is 2.079, that is, there is neither attraction nor repulsion between them. This is named as the comfort zone. However, the function s is able to partition the space into two parts between grasshoppers, that is, repulsion or attraction region and comfort zone. The value of s function is close to zero and distance is greater than 10. Therefore, we cannot use this function for strong forces between grasshoppers with large distances of them. The value of G in (1) is calculated as follows:

$$G_i = -g\hat{e}_g \tag{4}$$

where g is the constant of gravitational and $\hat{e}_g$ is a unity vector that is towards the center of earth. Besides, the A component in (1) is obtained as follows:

$$A_i = u\hat{e}_w \tag{5}$$

where u is a drift constant and $\hat{e}_w$ is a unity vector in the wind direction.

This mathematical model cannot be applied directly to solve the optimization problem because the reaching of grasshoppers to the comfort zone is very quick and the swarm system is not able to converge the target position. The modified version of grasshopper position that is used for update of grasshopper position is as follows:

$$x_{i+1}^d = c\Big(\sum_{\substack{j=1 \\ j\neq i}}^{N} c\frac{ub_d - lb_d}{2} s\big(\big|x_j^d - x_i^d\big|\big)\frac{x_j - x_i}{d_{ij}}\Big) + \hat{T}_d \tag{6}$$

where ub_d is upper boundary in the d^{th} dimension, lb_d is lower boundary in the d^{th} dimension, and $\hat{T}_d$ is the best solution found so far and c is a reduction coefficient to reduce size of comfort zone, repulsion zone and attraction zone. $s(\big|x_j^d - x_i^d\big|)$ is a function of social forces strength and $\frac{x_j - x_i}{d_{ij}}$ is a unit vector from the i^{th} grasshopper to the j^{th} grasshopper.

The first term of (6), which is within parentheses, is considered as the position of other grasshoppers and implements the interaction of grasshoppers in nature. The second term ($\hat{T}_d$) simulates the tendency of movement towards food source. Also, c parameter demonstrates the speed approaching of grasshoppers to the food source and consuming it. To introduce random behavior, it is possible to multiply both terms in (6) with random variables. As mentioned in [9], this is different from PSO that is briefly explained in the sequel and we compare our results with the results obtained from PSO algorithm.

In order to balance exploration and exploitation, the c parameter is needed to reduce proportionally the iteration count. With increasing number of iterations, the exploitation is promoted. Also, c parameter reduces comfort zone proportional to the count of iteration and obtained as follows:

$$c = c_{\max} - l\frac{c_{\max} - c_{\min}}{L} \tag{7}$$

where c_{max} is the maximum value and c_{min} is the minimum value and l shows the current iteration and L is the maximum number of iterations. This factor causes the swarm to converge the target, properly track the mobile target and reduce the comfort zone. These features lead to observe the convergence quickly without getting trapped to local optimum.

B. The Particle Swarm Optimization

The particle swarm optimization is a population-based optimization technique that was introduced by Kennedy and Eberhart in 1995, [11-12]. This method is inspired from the social behavior of birds searching for food. Due to its prominent features, such as the simplicity of the search mechanism, computational efficiency and easy implementation, the algorithm is widely used in many areas of optimization. Each particle has a little mass, and each particle in the swarm is represented as a solution in a high dimensional search space. This solution is a vector that consists of the position and its velocity. During the search process, the position of each particle in the search space is determined by its personal best denoted by P_{best} and global best denoted by G_{best}. At each iteration, each particle updates its position and velocity as follows:

$$x_i = x_{i-1} + v_i \tag{8}$$

$$v_i = \omega v_{i-1} + c_1 r_1 (P_{best} - x_{i-1}) + c_2 r_2 (G_{best} - x_{i-1}) \tag{9}$$

where ω is inertia weight, x_i is the position of the particle in i^{th} iteration and v_i is velocity of the particle in i^{th} iteration. r_1, r_2 are random numbers from the interval [0, 1] and c_1, c_2 represent the individual and group learning rates respectively. These are the important factors affecting the efficiency and performance of the PSO, which determine the global and local search ability and accuracy.

III. Simulation Results

The main aim is to obtain a proper path between an initial position and a target position in 2D environment containing static obstacles. We use multi objective function. These functions represent the effects of shortest distance, minimum energy consumption and minimum time, which are the variables of optimization. The used objective functions result in a collision free path in the configuration space by connecting the initial and target positions. In order to simulate the proposed algorithm, the environment map is designed as a grid of 1000×1000 pixels. The coordinates of the initial and target positions and the number of obstacles can be set or reset by using a MATLAB GUI. The positions of the obstacles are generated randomly in the workspace and the obstacles have rectangular shapes.

First, in the workspace, we randomly generate points that are considered as configuration in free space $V_i(x,y)$, $i=1,2,..,N$. Then we try to find an index sequence of the collision free vertices that connect the starting position $SP(x,y)$ to the target position $TP(x,y)$. In the workspace, the positions of the selected points demonstrate an index to the next vertex $V_n(x,y)$, $n \in [1, N+1]$ where the final index represents the target position in the environment $V_{n+1}(x,y)=TP(x,y)$. In addition to approximate the length of path between the initial and target positions, it is supposed that the initial and target positions are SP_0 and TP_{n+1}. The function used for the length of path is as follows.

$$l(p) = \sum_{i=0}^{n} d(SP_i, TP_{i+1}) \tag{10}$$

where $l(p)$ is the length of the path and $d(SP_i, TP_{i+1})$ is distance between SP_i and TP_{i+1}. The distance between the initial and the target positions is divided into $n+1$ equal segment. Therefore, we can calculate the value of $d(SP_i, TP_{i+1})$ as given below.

$$d(SP_i, TP_{i+1}) = \sqrt{(x'_{SP_i} - x'_{TP_{i+1}})^2 + (y'_{SP_i} - y'_{TP_{i+1}})^2} \tag{11}$$

To perform the path planning for the mobile robot, two objective functions are utilized. The first function ($F_1(p)$) is Euclidean distance that is used for finding the shortest path between start and target positions.

$$F_1(p) = \sum_{i=0}^{n-1} d_i \tag{12}$$

If there is an obstacle on the resulting path, the nearest point to the obstacle's border is selected. The other objective function is smoothness path. This function ($F_2(p)$) is expressed as the angle between the two lines that are connected the target point to the robot's two successive positions in each iteration.

$$F_2(p) = \sum_{i=0}^{n-1} \varphi_i + \delta l \tag{13}$$

where φ_i is the angle between two lines ($0 < \varphi < \pi$), δ is a positive constant and l is the number of line segments in the result path.

In problem of multi objective optimization, an obtained solution cannot be compared with other ones by relational operators because there is more than one criterion for comparison. Therefore, to obtain the optimal solution of multi objective optimization, we use the Pareto optimal solution that is able to measure and find how much a solution result is better than others. The operator of Pareto dominance is defined as follows:

$$\forall i \in \{1,2,...,n\} : f_i(\vec{x}) \le f_i(\vec{y}) \wedge \exists i \in \{1,2,...,n\} : f_i(\vec{x}) < f_i(\vec{y}) \tag{14}$$

where $\vec{x} = (x_1, x_2, .., x_n)$ and $\vec{y} = (y_1, y_2, .., y_n)$. This equation shows that all components of x are smaller than the corresponding component of y, or at least one component is smaller. A point x^* is called a non-dominated solution if no solution can be found to dominate on it. The Pareto multi objective can be mathematically defined as follows:

$$\forall i \in \{1,2,..., n\} : \{\nexists \vec{y} \in X \mid f_i(\vec{y}) \prec f_i(\vec{x})\} \qquad (15)$$

where *X* is the solution set. This solution is a Pareto optimal solution because it cannot be dominated by the solution *x*. To minimize a problem with *d*-dimensional decision vector and *h* objectives, the multi objective optimization is given by

$$\text{Minimize } F(x) = (f_1(x), f_2(x),..., f_h(x))$$

$$\text{Subject to } x \in [x_L, x_U] \qquad (16)$$

where x is a decision vector as a set of $(x_1, x_2,.., x_n) \in X \in R^d$ and *F(x)* is the objective function with the objective vector as a set of $(f_1, f_2,.., f_n) \in Y \in R^h$. x_L and x_U are constraints of lower and upper bounds of agent range, respectively. The feasible set of decision space for all the search agents that meeting the constraints forms is $\Omega = \{x \in R^d \mid x \in [x_L, x_U]\}$. As already mentioned, the aim of optimization is to find the Pareto optimal solution. It is randomly started with a generated population of search agents that should uniformly distribute among the search space. We consider *F(p)* as an objective function for path planning problem with the two objective functions defined by Eq. (12) and (13). Therefore, from Eqs. (12), (13) and (16) can be formulated as in the optimum mathematical form for MGOA.

$$\text{Minimize } F(p) = (f_1(\hat{x}_l, \hat{y}_l), f_2(\hat{x}_l, \hat{y}_l))$$

$$\text{Subject to } (\hat{x}_l, \hat{y}_l) \in (\hat{x}_L, \hat{y}_L), (\hat{x}_U, \hat{y}_U) \qquad (17)$$

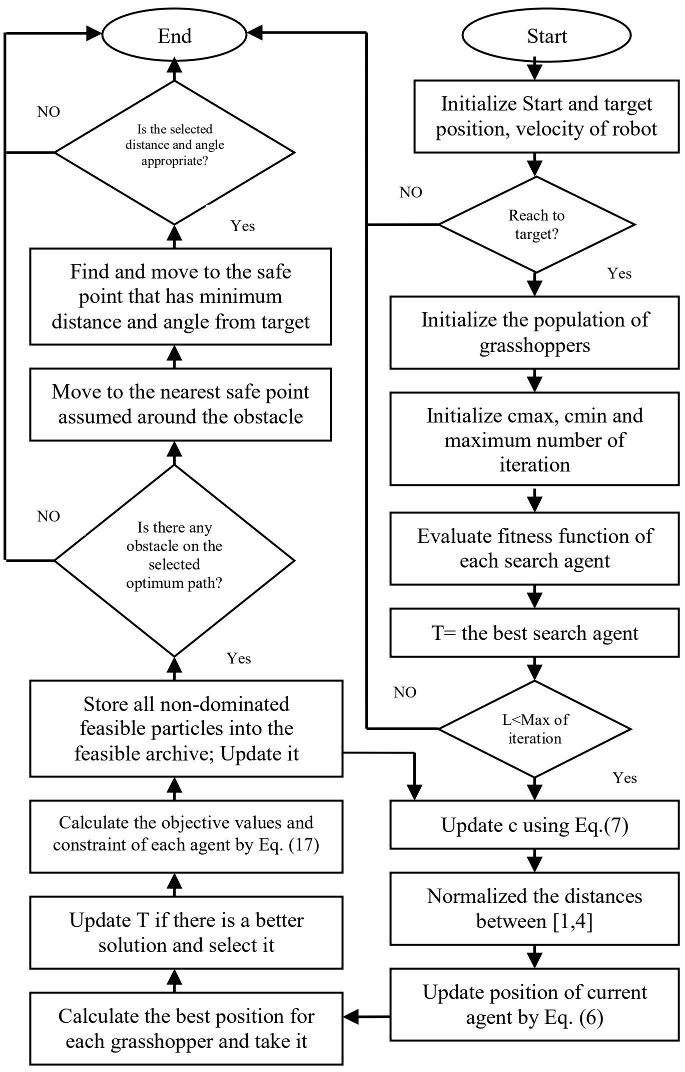
$$i = m+1,..., n$$

where $p = (\hat{x}_l, \hat{y}_l)$ is decision vectors that consist of the estimated coordinates corresponding to solutions for GOA. f_l is the objective function of length path constraint, f_2 is the objective function of smoothness path constraint and $(\hat{x}_L, \hat{y}_L), (\hat{x}_U, \hat{y}_U)$ are the lower and upper bound constraints, respectively. The main goal of a multi objective optimal model for path planning with both the shortest and smoothest path constraints is to obtain multi objective Pareto optimal solution. Therefore, according to the decision space feasible set Ω and the Pareto optimal solution *F(p*)*, the main nature of MGOA can be described as determining the dominant relationship that saves Pareto optimal solution set *X* in an archive by Eq. (16) and updates the best solution for problem.

The flowchart of proposed algorithm is shown in Figure1. The proposed algorithm is implemented on a system with Intel i7 processor (2.8 GHz) and 16 GB RAM. The Parameters used in the simulation have been described in Table 1.

TABLE I. PARAMETER SETTINGS FOR SIMULATION

Parameters	Values
Swarm Size (N)	500
Number of Iteration (Max_iter)	100
Maximum Value (cMax)	1
Minimum Value (cMin)	0.00001



Start
Initialize Start and target position, velocity of robot
Reach to target?
NO
Yes
Initialize the population of grasshoppers
Initialize cmax, cmin and maximum number of iteration
Evaluate fitness function of each search agent
T= the best search agent
L<Max of iteration
NO
Yes
Update c using Eq.(7)
Normalized the distances between [1,4]
Update position of current agent by Eq. (6)
Calculate the best position for each grasshopper and take it
Update T if there is a better solution and select it
Calculate the objective values and constraint of each agent by Eq. (17)
Store all non-dominated feasible particles into the feasible archive; Update it
Is there any obstacle on the selected optimum path?
Yes
NO
Move to the nearest safe point assumed around the obstacle
Find and move to the safe point that has minimum distance and angle from target
Is the selected distance and angle appropriate?
Yes
NO
End

Fig. 1. The flowchart of proposed algorithm for path planning

Performance and efficiency of the proposed algorithm is tested in several environments with different number of obstacles at different places. Also, we use two criterions to evaluate performance of multi objective optimization algorithms that measure the probability whether the result solution is the actual Pareto optimal solution or not. The first criterion is related to error rate which can be calculated as follows:

$$ER = \frac{\sum_{i=1}^{n} x_i}{n} \qquad (18)$$

where *n* is the number of the obtained optimum solution in Pareto. When x_i is equal to 0, that is, the obtained solution is an actual Pareto element. Otherwise x_i is equal to 1.

The second is the set coverage metric and for two sets of non-dominated solution A and B is given by:

$$SCM(A,B) = \frac{\left|\{b \in B \mid \exists a \in A : a \preceq b\}\right|}{|B|} \qquad (19)$$

where $\preceq$ illustrates the weak dominance relation.

In environment #1 the initial position and the target position are taken to be *SP*(70.6, 847.2) that is shown by circle node and *TP*(855.57, 152.80) that is represented by square node, respectively. In Figure 2, the black shaded circles represent obstacles in environment that the number of obstacles is 80 with random position and size determined by our algorithm and the dashed line represents the optimized and smooth path obtained by the proposed algorithm.

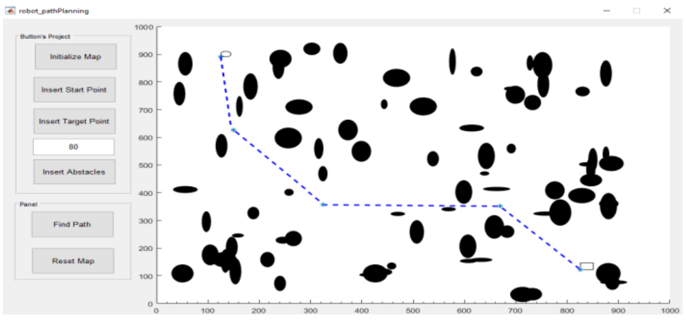


Fig. 2. The simulation of proposed algorithm for path planning with 80 obstacles.

Environment #2 consists of 50 obstacles in different places and the start position and target position are taken to be *SP*(88.73, 845.39) and *TP*(681.9948,179.92), respectively. The found path is represented by the dotted line as shown in Figure 3.

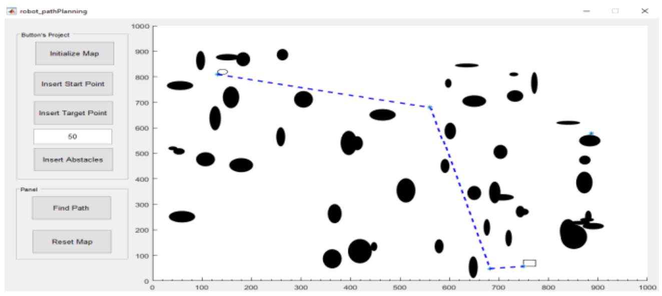


Fig. 3. The simulation of proposed algorithm for path planning with 50 obstacles.

To evaluate performance and efficiency, the proposed algorithm is compared with the PSO algorithm results. In the proposed algorithm, the next position of grasshopper is computed based on its current position, the position of target and the position of all grasshoppers. Therefore, the status of all grasshoppers is considered to define the location of search agents around target. The result is different from what we get with PSO. In the PSO algorithm, there are two vectors for each particle, namely position and velocity vectors as opposed to MGOA, where there is only one vector for each search agent. The other main difference between two algorithms is that PSO updates the position of a particle based on its current position, personal best and the global best values. While the MGOA updates the position of an agent based on its current position, global best and the position of all agents. This means that in PSO none of the other particles contribute to update particle position but in MGOA need the position all grasshoppers to obtain next position of each grasshopper. Also, the model of environment #1 with *SP*(66.71, 758.59) and *TP*(797.28, 304.70) and the model of environment #2 with *SP*(219.56, 789.33) and *TP*(801.16, 64.2) are used for the PSO algorithm and the obtained results are shown in Figures 4 and 5, respectively. In these figures, it can be seen that the PSO algorithm cannot find the shortest and optimal path in the both environments. The computational complexity of proposed algorithm is of O(MN^2) where *M* is the number of objectives function and *N* is the number of obtained solutions.

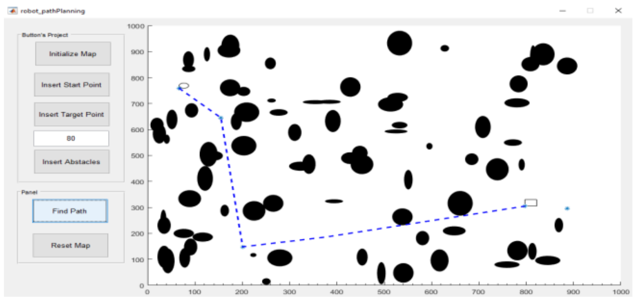


Fig. 4. The simulation of PSO algorithm for path planning with 80 obstacles.

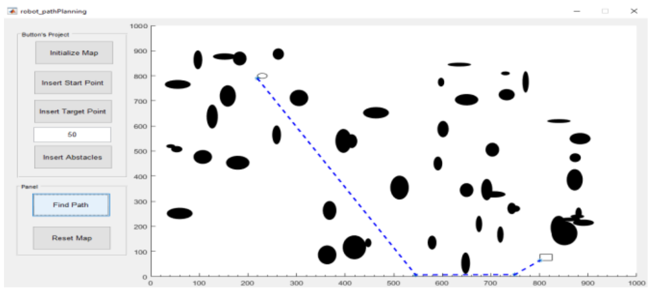


Fig. 5. The simulation of PSO algorithm with 50 obstacles.

To evaluate and demonstrate the superiority of the proposed algorithm to the PSO, it is compared obtained path length and execution time versus different swarm size in environments #1 and #2. The results for the environment #1 are shown in Figure 6 and Figure 7. Also, the obtained path length and execution time for the environment #2 are illustrated in Figure 8 and Figure 9. The results of MGOA and MPSO with two metrics are presented in Table 2. Table 2 includes average, standard deviation, median, best, and worst value for ER and SCM obtained by algorithms after 10 independent runs. From Table 2, it is clear that MGGOA outperforms MPSO because ER is a good indicator of the accuracy of algorithms to approximate Pareto optimal solutions. Therefore, the obtained results show that MGOA closely converge towards the Pareto optimal solution as well.

TABLE II. THE COMPARISON RESULTS FOR TWO ALGORITHMS MGOA AND MPSO.

Algorithm	**Ave.**	**Std.**	**Median**	**Best**	**Worst**
MGOA (ER)	0. 022	0.0008	0.025	0.0023	0.022
MPSO (ER)	0.0101	0.005	0.017	0.0143	0.028
SCM(MGOA,MPSO)	0.692	0.105	0.58	0.712	0.447
SCM(MPSO,MGOA)	0.221	0.122	0.257	0.339	0.162

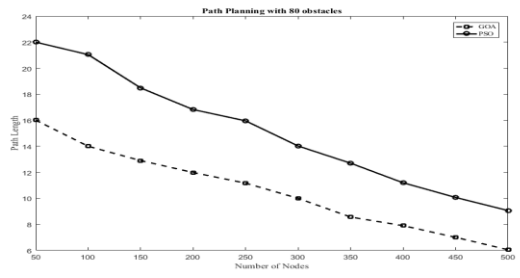


Fig. 6. The obtained path length versus number of nodes in environment #1

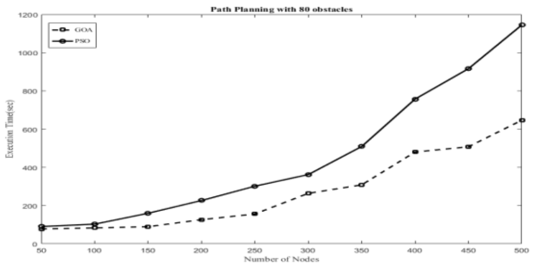


Fig. 7. The execution time of proposed algorithm versus number of nodes in environment #1.

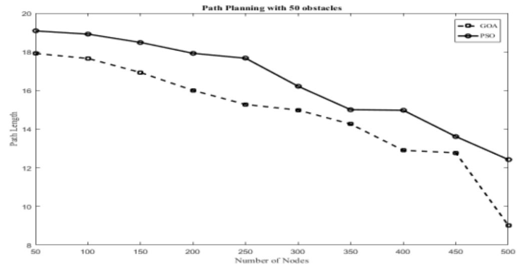


Fig. 8. The obtained path length versus number of nodes in environment #2.

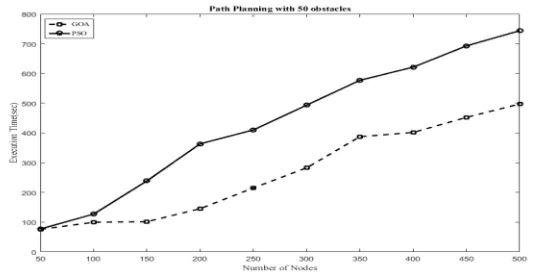


Fig. 9. The execution time of proposed algorithm versus number of nodes in environment #2.

It is clear that the proposed algorithm achieves shorter and optimal solution in shorter time than the PSO. The reason is that as mentioned above, the grasshopper optimization algorithm uses one position vector for each agent while the PSO algorithm have two vectors of position and velocity for each particle. The results obtained show that the performance of the proposed algorithm to find the shortest and smoother path is better than the PSO algorithm. Therefore, in the real world applications the proposed algorithm may be a good alternative.

IV. Conclusion

This paper presents a new algorithm, which is based on grasshopper optimization, for mobile robot path planning. The robot workspace includes obstacles at random coordinates, an initial position and a target position. During the optimization stage, multiple objectives are used to obtain an appropriate navigation route. The proposed algorithm optimizes distance, time and synthesizes a smooth path. The result of the grasshopper optimization scheme is compared with those of the PSO and it is observed that the proposed approach displays prominent features in terms of time and smoothness. Besides, the resulting path is shorter in the proposed approach than that of PSO.

References

[1] H. Meng, X. He, J. Song and Z. Liu," Path planning research based on the improved ant colony algorithm in ECDIS". 35th IEEE Chinese Control Conference(CCC), Chengdu, China, pp. 5504-5508, July 2016.

[2] H. Che, Z. Wu, R. Kang and C.Yun, " Global path planning for explosion-proof robot based on improved ant colony optimization". In Conference on Intelligent Robot Systems (ACIRS), Asia-Pacific, pp. 36-40, 2016.

[3] S. Al Dabooni and D. Wunsch, "Heuristic dynamic programming for mobile robot path planning based on Dyna approach". In International Joint Conference on Neural Networks (IJCNN), pp. 3723-3730, 2016.

[4] D. Gentilini, N. Farina, E. Franco, A. E. Tirri, D. Accardo, R. S. L. Moriello and L. Angrisani, "Multi agent path planning strategies based on Kalman Filter for surveillance missions". In IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), pp. 1-6, 2016.

[5] Z. Nie, X. Yang, S. Gao, Y. Zheng, J. Wang and Z. Wang, "Research on autonomous moving robot path planning based on improved particle swarm optimization". In IEEE Congress on Evolutionary Computation (CEC), pp. 2532-2536, 2016.

[6] V. D. C. Santos, F. S. Osório, C. F. Toledo, F. E. Otero and C. G. Johnson, "Exploratory path planning using the Max-min ant system algorithm". In IEEE Congress on Evolutionary Computation (CEC), pp. 4229-4235, 2016 .

[7] J. Yu and S. M. LaValle,"Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics". IEEE Transactions on Robotics, October 2016, vol. 32, pp.1163-1177.

[8] X. Du, X. Li, D. Liu and B. Dai, "Path planning for autonomous vehicles in complicated environments". In IEEE International Conference on Vehicular Electronics and Safety (ICVES), pp. 1-7, July 2016.

[9] S. Saremi, S. Mirjalili and A. Lewis, "Grasshopper optimisation algorithm: Theory and application". Advances in Engineering Software, March 2017, vol. 105, pp. 30-47.

[10] C. Copot, A. Hernandez, T. T. Mac, and R. De Keyse, "Collision-free path planning in indoor environment using a quadrotor". In 21st International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 351-356, August 2016.

[11] M. S. Alam, M. U. Rafique and M. U. Khan, "Mobile Robot Path Planning in Static Environments using Particle Swarm Optimization". In international Journal of Computer Science and Electronics Engineering (IJCSEE), 2015, vol. 3. pp. 2320–4028.

[12] L. Liao, X. Cai, H. Huang, and Y. Liu, "Improved dynamic double mutation particle swarm optimization for mobile robot path planning". In Control and Decision Conference (CCDC), Chinese, pp. 3235-3239, May 2016.