

# Grasshopper optimization algorithm based path planning for autonomous mobile robot

Asmaa Shareef, Salah Al-Darraji

Department of Computer Science, College of Education for Pure Sciences, University of Basrah, Basrah, Iraq

## Article Info

### Article history:

Received May 19, 2022

Revised Aug 19, 2022

Accepted Sep 6, 2022

### Keywords:

Autonomous mobile robot

Global path planning

GOA

Optimization algorithm

Swarm intelligence

## ABSTRACT

Autonomous mobile robots have become very popular and essential in our life, especially in industry. One of the crucial activities of the robot is planning the path from a start point to a target point, avoiding obstacles in the environment. Recently, path planning received more attention, and many methodologies have been proposed. Path planning studies have shown the effectiveness of swarm intelligence in complex and known or unknown environments. This paper presents a global path planning method based on grasshopper optimization algorithm (GOA) in a known static environment. This algorithm is improved using the bias factor to increase the efficiency and improve the resulting path. The resulting path from this algorithm is further enhanced using an improved version multinomial logistic regression algorithm (MLR). The algorithms were evaluated using three different large environments of varying complexities. The GOA algorithm has been compared with the ant colony optimization algorithm (ACO) using the same environments. The experiments have shown the superiority of our algorithm in terms of time convergence and cost.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Salah Al-Darraji

Department of Computer Science, College of Education for Pure Sciences, University of Basrah

Basrah, Iraq

Email: aldarraji@uobasrah.edu.iq

## 1. INTRODUCTION

The field of the autonomous mobile robot has been focused in recent years due to the various applications in which the robots can be used. Many areas should be considered when studying autonomous robots, such as perception, localization, path planning, and motion control. Some of these problems are increasingly complex, which becomes difficult to solve and optimize using conventional mathematical methods. Path planning is one of the NP-Hard problems, which requires a high computational cost to be solved using classical methods [1]-[3]. Metaheuristic methods have been proposed to overcome the complexity of classical methods. Metaheuristic algorithms are often independent of problem domains; the word "meta" means a higher level. These methods are widely used in fields like robotics and are based on natural phenomena, evolutionary processes, and swarm intelligence. Metaheuristic methods are approximate optimization algorithms that do not guarantee optimal solutions. However, they are still near optimum in the reasonable computational time used for complex and significant problems. Several classifications are used in [4]-[8] that can be summarized in Figure 1.

One of the essential metaheuristic methods is population-based algorithms that combine different solutions with high fitness values to create good or excellent solutions. Every iteration, a new solution with higher fitness values gradually replaces the one with lower fitness values, finding an optimal solution. Swarm intelligence is another essential feature of bioinspired models based on population, which describes the

behavior of animals in a collective or self-organized manner (e.g., birds, bees, ants). Path planning and optimization problems can be solved using numerical methods, bio-inspired algorithms, or hybridization. Research is being expanded on techniques inspired by nature and methods derived from birds, insect colonies, and various animal swarms to solve path planning, obstacle avoidance, and path improvement, which include: ant colony optimization (ACO) [9], genetic algorithms [10], particle swarm optimisation (PSO) [11], bee colony optimization (BCO) [12], cuckoo search (CS) [13], bat algorithm (BA) [14], firefly algorithm [15], grey wolf optimization (GWO) [16], ant lion algorithm (ALO) [17], moth-flame optimization (MFO) [18], whale optimization algorithm (WOA) [19], dragonfly algorithm (DA) [20], salp swarm algorithm (SSA) [21], grasshopper optimization algorithm (GOA) [22].

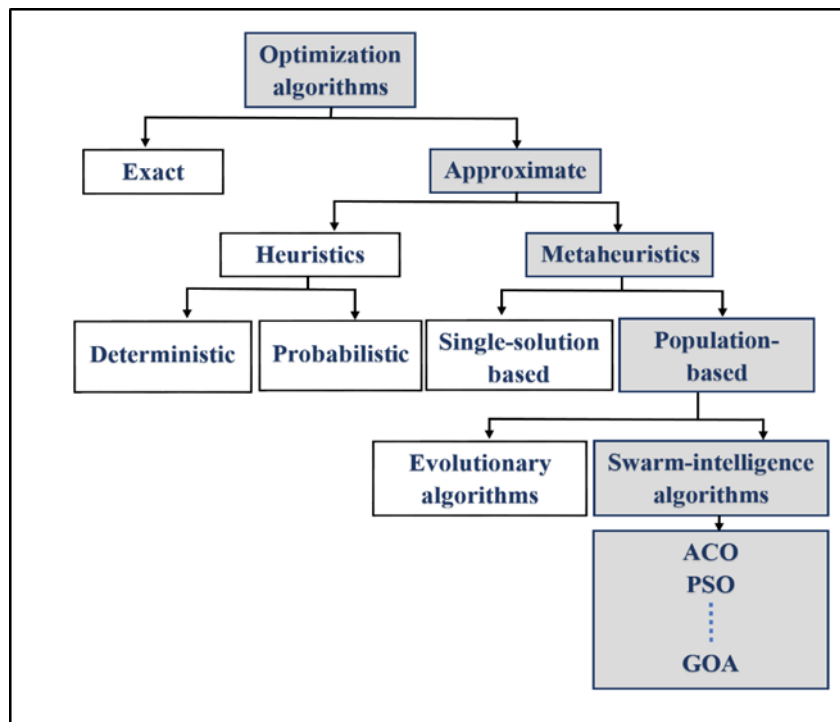


Figure 1. Optimization algorithms types

Many studies and research have focused on both fields of bio-inspired algorithms and path planning [23]. Bio-inspired algorithms incorporate mechanisms, structures, and strategies from biological systems into the design of new algorithms. Swarm intelligence and evolutionary algorithms are the most successful bio-inspired algorithms. This paper utilizes a bio-inspired algorithm GOA in autonomous robot path planning. GOA is an AI method derived from natural grasshopper swarming behavior proposed by Saremi *et al.*, [22]. It has improved complex problems in many fields, such as engineering, computer science, medicine, and economics. Elmi and Efe [24] proposed a multi-objective GOA for robot path planning in a static environment. They worked on several objectives such as distance, time, and generating smooth paths. Compared with the PSO, grasshopper optimization displays distinct advantages with time and smoothness. On the other hand, the same authors [3] presented safe and smooth pathfinding for a mobile robot that can traverse from a starting point to a destination point without colliding obstacles. Four circles (different radii on the range sensor) estimate obstacle direction. Then it finds areas where these circles and obstacles intersect. The grasshopper algorithm was used to help the robot avoid obstacles. The two papers did not discuss or provide solutions for the problems of grasshopper, local optimums, or slow convergence.

Other studies have developed and used grasshopper algorithms in various disciplines and hybridized them with different algorithms. Arora and Anand [25] used the chaos theory throughout the GOA optimizer to speed up implementation. Researchers are using two parameters, one generated by a chaotic map to balance exploration and exploitation of the entire grasshopper swarm, and the other reducing zone attraction, comfort, and repulsion to guide the grasshoppers either to explore or exploit.

On the other hand, some studies suggested hybridizing GOA with different optimization algorithms, such as meta-heuristics and machine learning. Barman and Choudhury [26] used GOA and support vector regression (SVR) to predict the forecast for short-term load during periods with significant weather changes. For securing medical data, Alphonsa and Sundaram [27] proposed a hybrid optimization approach called GOAGA, which combined GOA with genetic algorithms (GA). GOAGA proved to be more effective in preserving sensitive healthcare data than the other algorithms.

Moreover, that study was compared with the ACO, the most famous algorithm in the path planning field, an intelligent swarm algorithm. In this algorithm, the cooperative foraging behavior of ants is simulated. Among its benefits are positive feedback, high robustness, and parallel processing. In the ant colony algorithm [28]-[30], studies have been proposed to develop an algorithm such as a system based on Ant-Q, called AQS. MMAS (MAX-MIN ant system) and others were working on modifying the initial pheromone or enhancing the pathfinding by combining it with another algorithm. The contribution of this work can be summarized as follows: i) using GOA for planning robot paths, ii) get rid of falling in local minima associated with the grasshopper algorithm, iii) modifying the grasshopper algorithm by introducing the biasing towards the target with a specific probability, iv) multinomial logistic regression (MLR) with a new technique, multiphase multinomial logistic regression (MMLR), was used to reduce sharp zigzags caused by random grasshopper behavior for the initial path such that the resulting path is a near-optimal one.

The rest of the paper is organized as follows: section 2 introduces the biological behavior of grasshoppers, section 3 presents the proposed GOA path planning, section 4 discusses the experiments and results, section 5 presents the conclusion and future works.

## 2. BIOLOGICAL BEHAVIOR OF GRASSHOPPER

Grasshoppers are insects that are found in the grass and are considered to be pests because fact that they damage crop production and agriculture. Grasshoppers have three phases in their life cycle: egg, nymph (doesn't have wings), and adulthood (has wings) (Figure 2(a)). So, they create a swarm in the air and move quickly to a vast area. Grasshopper swarms have the following characteristics: i) during the nymph phase when grasshoppers do not have wings, they move slowly based on small steps, ii) as they become adults, grasshopper swarms can jump suddenly and travel long distances due to their wings, iii) during its food-seeking process, the swarm divides the process into two phases: exploration and exploitation, as shown in Figure 2(b).

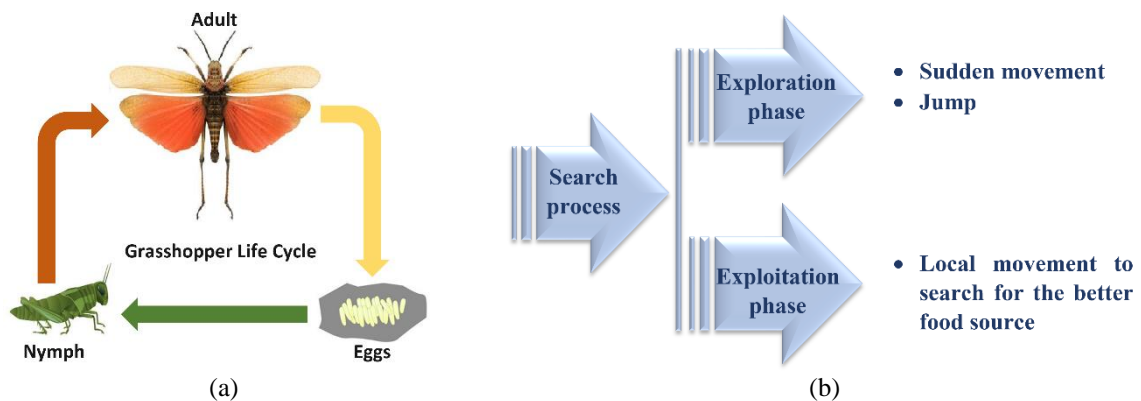


Figure 2. Grasshopper (a) life cycle and (b) exploration and exploitation

GOA is an optimization algorithm based on the grasshopper population, where each grasshopper represents a solution [22], [31], [32]. It can be mathematically formulated as in the following steps:

### 2.1. Determine the position of each solution in the swarm

The mathematical model that was used to calculate each solution's position  $X_i$

$$X_i = S_i + G_i + A_i \quad (1)$$

$S_i$  represents the social interaction between the  $i$ th solution grasshopper and the other grasshoppers,  $G_i$  represents the gravitational force acting on the  $i$ th grasshopper, and  $A_i$  represents wind advection. Adding random behavior to each solution results in (2):

$$X_i = r_1 S_i + r_2 G_i + r_3 A_i \quad (2)$$

There are three random numbers in the range,  $r_1$ ,  $r_2$ , and  $r_3$  [0, 1].

## 2.2. Social interaction force

The social interaction of a grasshopper with another grasshopper can be calculated as follows:

$$S_i = \sum_{j=1}^N s(d_{ij}) \hat{d}_{ij}, \text{ where } i \neq j \quad (3)$$

$$s(r) = f e^{-\frac{r}{l}} - e^{-r} \quad (4)$$

Where  $d_{ij} = |x_j - x_i|$  represents, the distance between the  $i$ th and  $j$ th grasshopper,  $\hat{d}_{ij} = \frac{x_j - x_i}{d_{ij}}$  describes the unit vector. Moreover,  $s$  represents the attraction and repulsion between grasshoppers, in which  $f$  is the intensity of attraction and  $l$  is the attractive length scale. The coefficients  $l$  and  $f$  are critical in grasshopper social behavior.

## 2.3. Force of gravity

Calculating gravity force,  $G_i$  is shown by (5):

$$G_i = -g \hat{e}_g \quad (5)$$

Where  $-g$  is the gravitational constant and  $\hat{e}_g$  is the unit vector toward the earth's center.

## 2.4. Wind direction

Nymphs and adult grasshoppers are strongly affected by wind direction. Therefore, they move in a wind-related pattern  $A_i$ . It can be represented as (6):

$$A_i = u \hat{e}_w \quad (6)$$

Where  $u$  is the drift constant, and  $\hat{e}_w$  is the unit vector in the wind direction. In (1) can be rewritten using  $S$ ,  $G$ , and  $A$  as (7):

$$X_i = \sum_{j=1, j \neq i}^N s(|x_j - x_i|) \frac{x_j - x_i}{d_{ij}} - g \hat{e}_g + u \hat{e}_w \quad (7)$$

Where  $N$  is the number of grasshoppers.

To preserve optimality and prevent grasshoppers from jumping into their comfort zones as well as the swarm from failing to reach the target location (global optimum), there will be some changes in (7):

$$X_i = c \left( \sum_{j=1, j \neq i}^N c \frac{UB_d - LB_d}{2} s(|x_j^d - x_i^d|) \frac{x_j^d - x_i^d}{d_{ij}} \right) + \hat{T}_d \quad (8)$$

Where  $G_i = 0$ ,  $\hat{T}_d$  is the *best solution* in the  $d$ th dimension,  $UB_d$  and  $LB_d$  are the upper and lower bounds in the  $d$ th dimension, respectively.

In (8) shows that the next grasshopper position is determined by the current position, the target position, and the position of all other grasshoppers. As iterations proceed, the inner  $c$  contributes to decreasing repulsion/attraction forces between the grasshoppers, while the outer  $c$  reduces the search coverage around the target. Iterations in exploration and exploitation must be balanced by decreasing parameter  $c$  proportionately. In this manner, exploitation increases with the number of iterations. According to (9), the coefficient  $c$  reduces the comfort zone proportionally to the number of iterations:

$$c = c_{max} - l \frac{c_{max} - c_{min}}{L} \quad (9)$$

Where  $c_{max}$  is a maximum value,  $c_{min}$  is a minimum value,  $l$  indicates the current iteration, and  $L$  is the maximum number of iterations. Algorithm 1 explains the steps of GOA.

**Algorithm 1:** Grasshopper Optimization Algorithm (GOA).

---

```

1 Initialize the swarm  $X_i(i = 1, 2, \dots, n)$ 
2 Initialization  $c_{max}$ ,  $c_{min}$ , and maximum number of iterations;
3 Calculate the fitness of each agent;
4  $T$  = the best search agent;
5 while (  $l < iterations$  ) do
6     Update  $c$  using equation 9;
7     foreach search agent do
8         Normalize the distances between grasshoppers in [1,4];
9         Update the position of the current search agent by equation
8;
10        Bring the current search agent back if it goes outside the
boundaries;
11    end
12    Update  $T$  if there is a better solution;
13     $l = l + 1$ ;
14 end

```

---

### 3. THE PROPOSED WORK

The proposed work utilizes the GOA to plan a path for an autonomous mobile robot. GOA is originally used to solve optimization problems and cannot be used directly because it produces a zigzag path, which may pass through obstacles. It may also easily fall in local optimum. This work aims at finding solutions to these two problems by enhancing the algorithm to suit the needs of the path planning problem. The following steps describe the proposed path planning method:

#### 3.1. GOA path planning

GOA is used over the entire environment to find the required path from the start point to the target point, avoiding obstacles. The grasshopper algorithm plans a path using exploration and exploitation phases. Initially, each grasshopper (solution) is assigned a location point randomly distributed around the start point. During the exploration phase, each grasshopper is evaluated using a fitness function to measure the suitability of the current position of each grasshopper. In the present work, the fitness value is the Euclidean distance from the grasshopper location to the target location. After each iteration, the location of each grasshopper is updated using (8) until one of the grasshoppers reaches the target. It is confirmed if the line segment between the old and new grasshopper locations does not intersect any environmental obstacles. Otherwise, the new location is omitted. Some parameters strongly affect the performance of this algorithm. One of these parameters is the  $C_{max}$  and  $C_{min}$ , which controls the size of the grasshopper jump at each iteration. Large  $C_{max}$  values may avoid falling to a local minimum in large environments.

Algorithm 2 shows the globally static obstacle avoidance path planning algorithm using GOA. Obstacles, start and end points, max number of iterations, individuals of a population, and boundaries are the input to the algorithm, and the path is the output. The algorithm calculates the fitness of each grasshopper in the population after each iteration. At each iteration, the algorithm calculates a new location of each grasshopper. The line segment between the new location and the previous one should not intersect with any edge of obstacles until reaching the target point. The resulting path is the successive locations of the reached grasshopper. If the max iterations have reached, the algorithm declares that no path found.

#### 3.2. Bias-based GOA

As with any probabilistic algorithm, GOA suffers from slow convergence, which means it takes a long time to reach the target. In this work, a bias parameter is added to the GOA to speed up convergence. This parameter means that, in some probability, we use the target point instead of the best solution (grasshopper) in (8). Significant probability leads to fast convergence but may fall in a local minimum. Moderated value is a trade-off between convergence and getting stuck in local minima.

#### 3.3. Regression-based path optimization

The resulting path from the GOA is always not optimal (zigzag) due to the randomization nature of the algorithm. Therefore, finding a shorter and smoother version of the original path is necessary. This work uses the MLR method to obtain a zigzag-free path as in Ansarry and Al-Darraji [33]. In this method' the original path is regarded as a set of nodes  $X = X_1, X_2, X_3, \dots, X_n$ . For all  $i = 1, 2, \dots, n$ , and starting from node  $X_1$ , all edges  $E_{ij}$  for all  $j = n, n-1, n-2, \dots, i+1$  are checked against obstacles. If no obstacle intersects edge  $E_{ij}$ , it will be added to the new path. Otherwise, it will be omitted.

However, this method is efficient with paths consisting of short segments such as those resulting from RRT and its variants. In methods like GOA that produce paths with irregular segments, MLR is not the

suitable method to get a smooth and short path. Therefore, a MMLR, an updated version of MLR, is proposed for short and semi-smooth path. Algorithm 3 presents the proposed path optimization.

---

**Algorithm 2: GOA path planning.**


---

```

Input : start_point, target_point, obstacles, max_iterations, population_size,
lower_bound, upper_bound
Output: path
1 initialize the swarm positions  $X_i$  ( $i = 1, 2, \dots, \text{population\_size}$ );
2 initialize  $C_{max}$  and  $C_{min}$  ;
3  $paths = []$  ;
4 calculate the fitness of each grasshopper ;
5  $B =$  the best grasshopper ;
6  $iter = 1$  ;
7 while  $iter \leq \text{max\_iterations}$  do
8      $C = C_{max} - iter * ((C_{max} - C_{min}) / \text{max\_iterations})$  ;
9     for  $i = 1$  to  $\text{population\_size}$  do
10         normalize the distance between grasshoppers ;
11          $new\_position = \text{calculateNewPosition}(X_i, C, B)$ 
12         if  $\text{outsideBoundaries}(new\_position, \text{lower\_bound}, \text{upper\_bound})$  then
13              $new\_position = \text{recalculate}(new\_position, \text{lower\_bound},$ 
upper_bound);
14         end
15          $line = \text{lineSegment}(X_i, new\_position)$  ;
16         if not  $\text{intersectObstacle}(line, \text{obstacles})$  then
17              $X_i = new\_position$  ;
18              $path[i].\text{append}(X_i)$  ;
19             if  $\text{fitness}(X_i) < \text{fitness}(B)$  then
20                  $B = X_i$  ;
21             end
22             if  $\text{reached}(X_i)$  then
23                  $path = paths[i]$  ;
24                 return path;
25             end
26         end
27     end
28      $iter = iter + 1$  ;
29 end
30 return no_path_found;

```

---



---

**Algorithm 3: Multiphase Multinomial Logistic Regression Algorithm.**


---

```

input : path, obstacles, segment_length, phases
output: regression_path
1 for  $phase = 0$  to  $\text{phases} - 1$  do
2     if  $phase > 0$  then
3          $path = \text{getSegmentedPath}(path, \text{segment\_length})$ ;
4     end
5      $shortpath = []$ ;
6      $startsegment = \text{round}(phase * (\text{length}(path) / 2) / \text{phases})$ ;
7     for  $p = 0$  to  $startsegment$  do
8          $shortpath.\text{append}(path[p])$ 
9     end
10     $i = startsegment$  ;
11    while  $i < \text{length}(path) - 1$  do
12        for  $k = \text{length}(path) - 1$  downto  $i$  do
13             $line = \text{lineSegment}(path[i], path[k])$ ;
14            if not  $\text{intersectObstacle}(line, \text{obstacles})$  then
15                 $shortpath.\text{append}(path[k])$  ;
16                 $i = k - 1$ ;
17                break;
18            end
19        end
20         $i = i + 1$ ;
21    end
22     $shortpath = \text{reverse}(shortpath)$  ;
23 end
24 if  $\text{phases} \% 2 == 1$  then
25      $shortpath = \text{reverse}(shortpath)$  ;
26 end
27  $regression\_path = shortpath$  ;
28 return regression_path;

```

---

This method passes through multiple phases, in which the path is curved each phase, and the resulting intermediate path is reversed to give a chance to the two endpoints of the path. To not reach the steady state, the start point at each phase is shifted by several segments as the phase increases. The first phase of this method is similar to MLR, which begins from the start point to the target point to get rid of the zigzag. The following phases re-segment the path resulting from the previous phase into a set of fixed-size segments before the MLR is applied again. Figures 3(a) and (b) depict the original and short path, respectively.

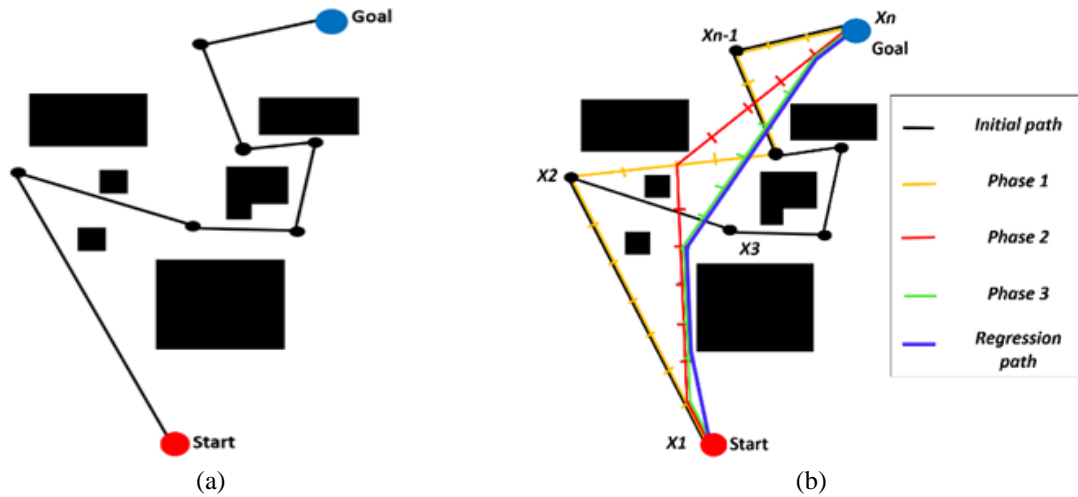


Figure 3. Path regression (a) original path and (b) short and semi-smooth path after 4 phases

**4. EXPERIMENTS AND RESULTS**

The experiments have been conducted using the robot operating system (ROS Noetic) simulation environment. This framework is operated under Linux (64-bit) operating system with (8GB) RAM and (intel Core-i7 11th Gen) processor. Three 1500x1500 cm maps with different complexities were used to evaluate the algorithm performance, as shown in Figures 4(a) to (c). The proposed algorithms were tested in different scenarios, then compared the results and evaluated their performance according to several factors (convergence time, cost, and path length). Three experiments were conducted; in each one, the algorithm was executed 100 times on the three maps. The first experiment compares several values of bias. The second compares GOA and ACO. The third experiment shows the effect of path regression.

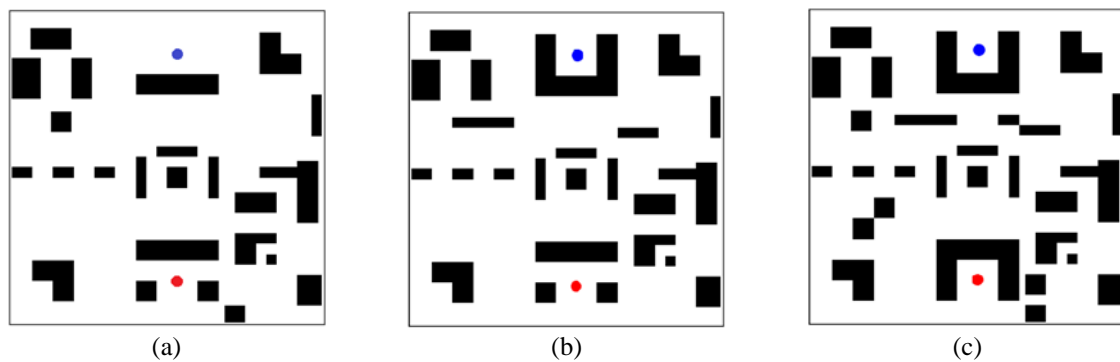


Figure 4. Experimental maps (a) map-1 (b) map-2 and (c) map-3

**4.1. Bias experiments**

These experiments focus on the bias effect on the algorithm. Bias is a probability coefficient (uniform distribution) with a certain percentage, either the grasshopper is heading toward the best grasshopper (the best solution), which is the original case (Bias=0), or towards the target point. We consider different bias values to observe their effect on the algorithm in different environments. Four bias values are

used 0, 0.25, 0.5, and 0.75. Figure 5 shows that the bias highly affects the convergence time, cost, and path length values. In map-1, the time to reach the goal is reasonable when there is no bias, whereas, in the other maps (more complex maps), it rises to 5.1 seconds. Different bias values decrease the time in each of the three maps, as shown in Figure 5(a). The experiments have also shown a high effect for the bias on the cost factor. Using a bias value highly decreased the cost, especially in map-3, as shown in Figure 5(b). Another factor is also affected by bias, which is the path length. Using bias values other than 0 decreased the path length, as shown in Figure 5(c).

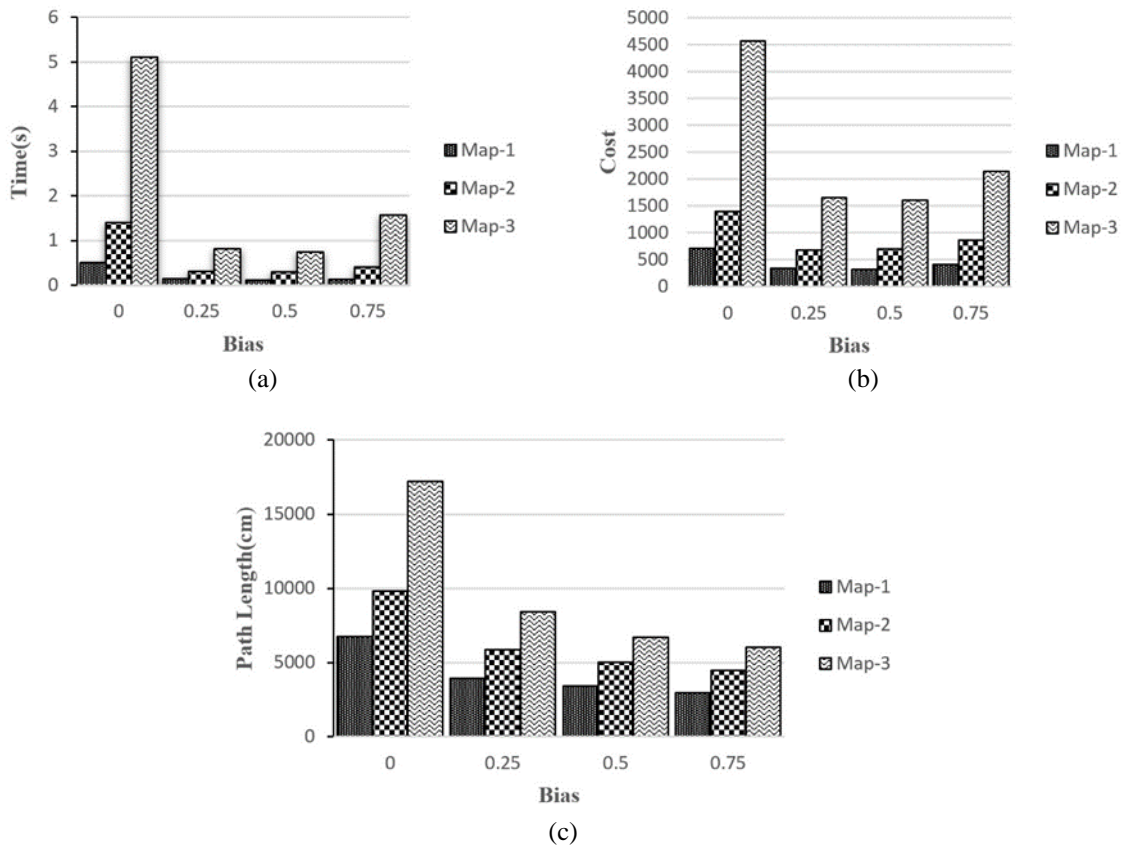


Figure 5. Bias experiments (a) time convergence (b) cost and (c) path length

**4.2. Comparative experiments**

This section presents comparative experiments between the GOA and the ACO under the same conditions and environments. As a result of these experiments, we demonstrated the superiority of GOA over ACO in terms of time and cost. Tables 1 and 2 show the average of 100 executions for both algorithms.

Table 1. Time convergence

Algorithm	map-1	map-2	map-3
ACO	5.2	8.8	7.9
GOA	0.1	0.3	0.9

Table 2. Cost

Algorithm	map-1	map-2	map-3
ACO	200945	342195	352528
GOA	350	712	2030

**4.3. Regression experiments**

In complex environments, the resulting path from the GOA is always zigzag and needs to be enhanced using some smoothing methods. This experiment shows the regression algorithm's effect on enhancing and shortening paths. Figures 6(a) to (c) show the path planning of the ACO algorithm for map-1, map-2, and map-3, respectively. Figures 6(d) to (f) show the original path of the GOA algorithm as well as the semi-optimal paths using a regression for map-1, map-2, and map-3, respectively. On the other hand, Figure 7 and Table 3 depict the difference between the length of the original path and the semi-optimal path resulting from the regression.



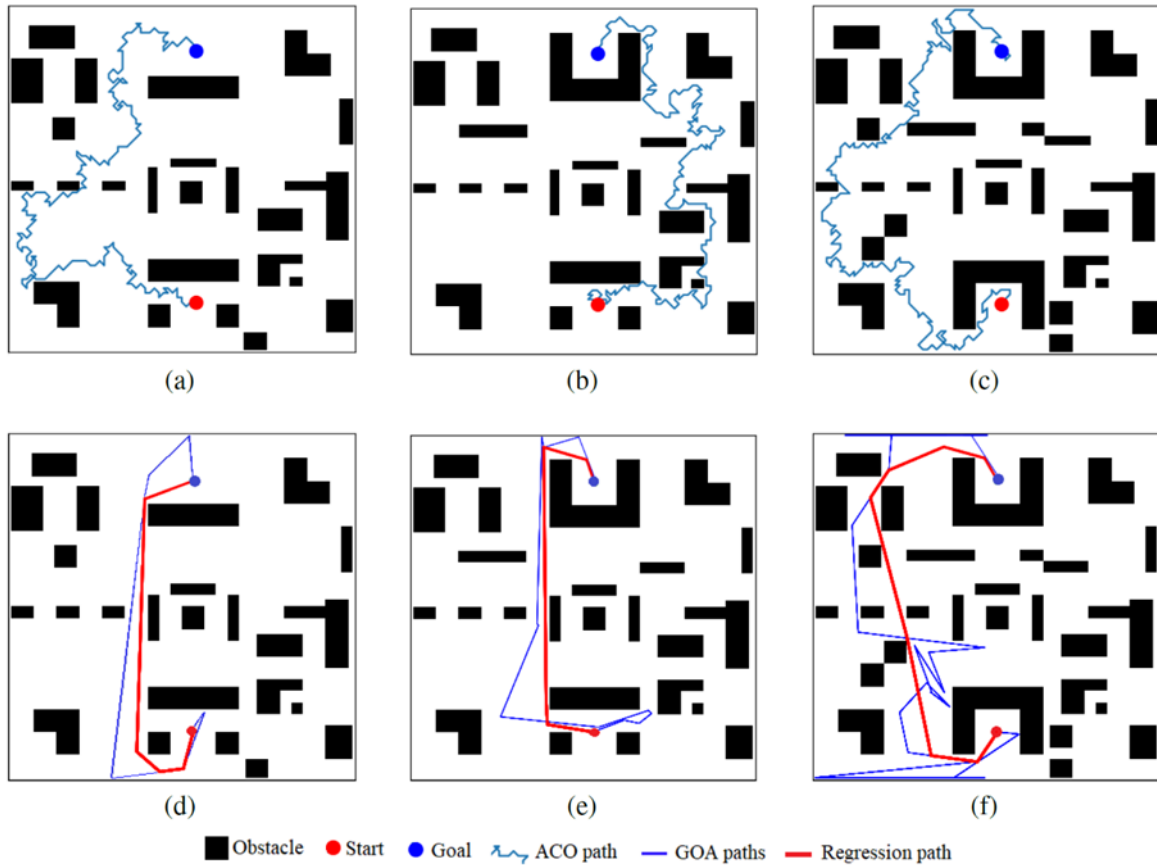


Figure 6. Experimental maps (a) map-1 ACO (b) map-2 ACO (c) map-3 ACO (d) map-1 GOA (e) map-2 GOA and (f) map-3 GOA

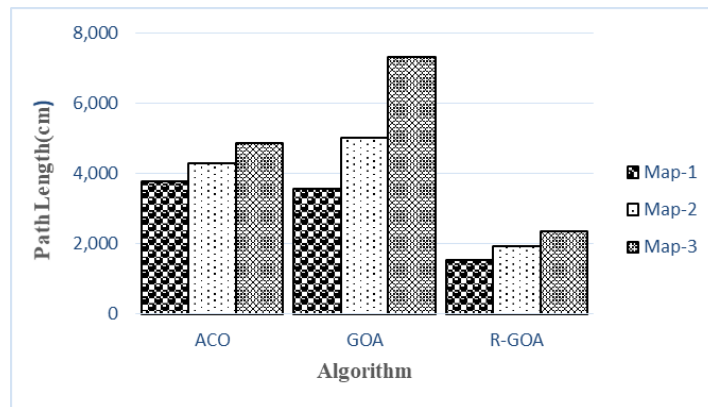


Figure 7. Path length comparison

Table 3. Path length

Algorithm	map-1	map-2	map-3
ACO	3757	4284	4865
GOA	3574	5014	7311
R-GOA	1542	1912	2339

## 5. CONCLUSION

This paper presents a global path planning method based on the grasshopper optimization algorithm. The original algorithm is enhanced to suit the path planning and obstacle avoidance problem. The bias factor

was used to improve the algorithm according to factors such as time, cost, and path length. Another enhancement on the resulting path is obtaining a new short semi-smooth path using MMLR. The algorithm was compared with ACO. The experiments have shown that our algorithm improved the time convergence and cost. Using the regression, the experiments have also demonstrated a shorter path than ACO. Future works could include using the new global GOA path planning version in dynamic environments.




## REFERENCES

- [1] S. A. -Ansarry and S. A. -Darraji, "Hybrid RRT-A\*: An Improved Path Planning Method for an Autonomous Mobile Robots," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 17, no. 1, pp. 1-9, 2021, doi: 10.37917/ijeec.17.1.13.
- [2] B. Chen and G. Quan, "NP-Hard Problems of Learning from Examples," *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, 2008, pp. 182-186, doi: 10.1109/FSKD.2008.406.
- [3] Z. Elmi and M. Ö. Efe, "Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 33, no. 3, pp. 467-485, 2021, doi: 10.1080/0952813X.2020.1764631.
- [4] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091-8126, Oct. 2021, doi: 10.1007/s11042-020-10139-6.
- [5] N. Khattar, J. Sidhu, and J. Singh, "Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 4750-4810, 2019, doi: 10.1007/s11227-019-02764-2.
- [6] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey," *International Journal Of Computer Engineering In Research Trends*, vol. 2, no. 5, pp. 296-304, May 2015.
- [7] S. Muthuraman and V. P. Venkatesan, "A Comprehensive Study on Hybrid Meta-Heuristic Approaches Used for Solving Combinatorial Optimization Problems," *2017 World Congress on Computing and Communication Technologies (WCCCT)*, 2017, pp. 185-190, doi: 10.1109/WCCCT.2016.53.
- [8] N. N. Nordin and L. S. Lee, "Heuristics and metaheuristics approaches for facility layout problems: a survey," *Pertanika Journal of Scholarly Research Reviews*, vol. 2, no. 3, pp. 62-76, 2016.
- [9] A. Colomi, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings Of Ecal91-European Conference On Artificial Life, Paris, France, Belsevier Publishing*, vol. 142, pp. 134-142, 1991.
- [10] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66-73, 1992.
- [11] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43, doi: 10.1109/MHS.1995.494215.
- [12] D. Teodorovic and M. Dell'Orco, "Bee colony optimization—a cooperative learning approach to complex transportation problems," *Advanced OR and AI methods in transportation*, vol. 51, p. 60, 2005.
- [13] X. -S. Yang and S. Deb, "Cuckoo Search via Lévy flights," *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210-214, doi: 10.1109/NaBIC.2009.5393690.
- [14] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pp. 284, pp. 65-74, 2010, doi: 10.1007/978-3-642-12538-6\_6.
- [15] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, 2010, doi: 10.1504/IJBIC.2010.032124.
- [16] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46-61, March 2014, doi: 10.1016/j.advengsoft.2013.12.007.
- [17] S. Mirjalili, "The ant lion optimizer," *Advances in engineering software*, vol. 83, pp. 80-98, 2015, doi: 10.1016/j.advengsoft.2015.01.010.
- [18] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-based systems*, vol. 89, pp. 228-249, 2015, doi: 10.1016/j.knosys.2015.07.006.
- [19] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51-67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.
- [20] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural computing and applications*, vol. 27, no. 4, pp. 1053-1073, 2016, doi: 10.1007/s00521-015-1920-1.
- [21] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in engineering software*, vol. 114, pp. 163-191, 2017, doi: 10.1016/j.advengsoft.2017.07.002.
- [22] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," *Advances in engineering software*, vol. 105, pp. 30-47, 2017, doi: 10.1016/j.advengsoft.2017.01.004.
- [23] Q. Yang and S. -J. Yoo, "Optimal UAV Path Planning: Sensing Data Acquisition Over IoT Sensor Networks Using Multi-Objective Bio-Inspired Algorithms," in *IEEE Access*, vol. 6, pp. 13671-13684, 2018, doi: 10.1109/ACCESS.2018.2812896.
- [24] Z. Elmi and M. Ö. Efe, "Multi-objective grasshopper optimization algorithm for robot path planning in static environments," *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 244-249, doi: 10.1109/ICIT.2018.8352184.
- [25] S. Arora and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Computing and Applications*, vol. 31, no. 8, pp. 4385-4405, 2019, doi: 10.1007/s00521-018-3343-2.
- [26] M. Barman and N. B. D. Choudhury, "Hybrid GOA-SVR technique for short term load forecasting during periods with substantial weather changes in North-East India," *Procedia computer science*, vol. 143, pp. 124-132, 2018, doi: 10.1016/j.procs.2018.10.360.
- [27] M. A. Alphonsa and N. MohanaSundaram, "A reformed grasshopper optimization with genetic principle for securing medical data," *Journal of Information Security and Applications*, vol. 47, pp. 410-420, 2019, doi: 10.1016/j.jisa.2019.05.007.
- [28] M. Dorigo, V. Maniezzo and A. Colomi, "Ant system: optimization by a colony of cooperating agents," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, Feb. 1996, doi: 10.1109/3477.484436.
- [29] H. Fatemidokht and M. K. Rafsanjani, "F-Ant: an effective routing protocol for ant colony optimization based on fuzzy logic in vehicular ad hoc networks," *Neural Computing and Applications*, vol. 29, no. 11, pp. 1127-1137, 2018, doi: 10.1007/s00521-016-2631-y.
- [30] T. Wang, L. Zhao, Y. Jia and J. Wang, "Robot Path Planning Based on Improved Ant Colony Algorithm," *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, 2018, pp. 70-76, doi: 10.1109/WRC-SARA.2018.8584217.




- [31] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems," *Applied Intelligence*, vol. 48, no. 4, pp. 805–820, 2018, doi: 10.1007/s10489-017-1019-8.
- [32] Y. Meraihi, A. B. Gabis, S. Mirjalili and A. Ramdane-Cherif, "Grasshopper Optimization Algorithm: Theory, Variants, and Applications," in *IEEE Access*, vol. 9, pp. 50001-50024, 2021, doi: 10.1109/ACCESS.2021.3067597.
- [33] S. A. -Ansarry and S. A. -Darraji, "MT Hybrid RRT-A\* Regression-based: An Enhanced Path Planning Method for an Autonomous Mobile Robots," *Journal of Basrah Researches ((Sciences))*, vol. 47, no. 1, 2021.

## BIOGRAPHIES OF AUTHORS



**Asmaa Shareef**    is an M.Sc student at the University of Basrah, College of Education for Pure Sciences, Computer Department. She received his BSc in computer science from the University of Basrah, Iraq, in 2006. She is currently working as a tutor and programmer at the Department of Computer Science, College of Education for Pure Sciences, University of Basrah. Her research interests include machine learning, deep learning, artificial intelligence, and robotics. She can be contacted at the emails: [asmaa.shareef@uobasrah.edu.iq](mailto:asmaa.shareef@uobasrah.edu.iq) and [asmaa.shareef@gmail.com](mailto:asmaa.shareef@gmail.com).



**Salah Al-Darraj**    is an Assistant Professor at the Department of Computer Science, College of Education for Pure Sciences, University of Basrah. He studied Computer Science at the University of Basrah (Iraq). He got a Master degree in Computer Science from the same university. He was awarded scholarship by the MoHESR (Iraq)-DAAD (Germany) cooperation program to complete his PhD in the Robotic Research Lab at the University of Kaiserslautern. During his PhD, the focus of his research is on the nonverbal communication with humanoid robot. He completed his PhD in the year 2016. His research interests includes artificial intelligence, machine learning, deep learning, computer vision, robotics, and humanoid robots. He can be contacted via [aldarraj@uobasrah.edu.iq](mailto:aldarraj@uobasrah.edu.iq).