Taylor & Francis
Taylor & Francis Group

Journal of Experimental & Theoretical Artificial Intelligence

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/teta20

Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment

Zahra Elmi & Mehmet Önder Efe

To cite this article: Zahra Elmi & Mehmet Önder Efe (2021) Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment, Journal of Experimental & Theoretical Artificial Intelligence, 33:3, 467-485, DOI: 10.1080/0952813X.2020.1764631

To link to this article: https://doi.org/10.1080/0952813X.2020.1764631

JOURNAL OF EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE
2021, VOL. 33, NO. 3, 467–485
https://doi.org/10.1080/0952813X.2020.1764631

Taylor & Francis
Taylor & Francis Group

ARTICLE

Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment

Zahra Elmi and Mehmet Önder Efe

Department of Computer Engineering, University of Hacettepe, Ankara, Turkey

ABSTRACT

The navigation of mobile robots using heuristic algorithms is one of the important issues in computer and control sciences. Path planning and obstacle avoidance are current topics of navigational challenges for mobile robots. The major drawbacks of conventional methods are the inability to plan motion in a dynamic and unknown environment, failure in crowded and complex environments, and inability to predict the velocity vector of obstacles and non-optimality of the synthesised path. This paper presents a novel path planning approach using a grasshopper algorithm for navigation of a mobile robot in dynamic and unknown environments. To accomplish this goal, two different approaches are presented. First, a sensory system is used to detect the obstacles and then a new method is developed to predict and avoid static and dynamic obstacles while the velocity of obstacles is unknown. The robot uses the obtained information and finds a collision-free, optimal and safe path. The controller proposed in this paper is tested in crowded and complex environments. Simulation results show that the approach is successful in all test environments. Also, the proposed controller is compared with several heuristic methods. The comparison work stipulates that the introduced controller here is promising in terms of running time, optimality, stability and failure rate.

ARTICLE HISTORY
Received 10 March 2019
Accepted 30 April 2020

KEYWORDS
Path planning; grasshopper optimisation algorithm; dynamic environment; mobile robot navigation; obstacle avoidance

Introduction

Mobile robots are widely used not only in many industrial areas including aerospace systems, nuclear applications, and mining equipment but also in household areas (Behroo & Banazadeh, 2015). One of the essential issues in robotics is path planning for mobile robots. The problem of path planning is a well-known NP-hard problem (B. Chen & Quan, 2008) where the complexity increases with the degrees of freedom of the vehicle. The main aim of path planning is to find a safe and smooth path in a dangerous environment for a mobile robot so that the robot moves from the starting point to the destination point without colliding with obstacles.

The path planning methods are investigated based on their properties namely static or dynamic, local or global and complete or heuristic (Buniyamin et al., 2011). In static path planning, the environment contains stationary obstacles whereas, in dynamic path planning, the environment contains moving obstacles. Meanwhile, path-planning methods are developed in partially known and fully unknown environments in the presence of static or dynamic obstacles (Patle et al., 2018). Global path planning methods are usually used for known environments where the surrounding information such as environment, obstacles, and the target is identified for the robot. The main advantage of this method is to find the optimal path and to avoid the local minimum. The advantage of the global path planning algorithms is that a continuous collision-free path can always be found

CONTACT Zahra Elmi zahra.elmi@gmail.com Department of Computer Engineering, University of Hacettepe, Ankara, Turkey

by analysing the connectivity of the free space. The main drawback of global path planning is the inability to handle uncertainty. Most of global path planning approaches are time-consuming, also they need prior knowledge of the environment. Consequently, they are executed offline and they are not suitable for unknown and dynamic environment. In local path planning, the robot does not require to get early information about the environment. The robot explores the surrounding environment using sensors to obtain information about location, shape, and size of the obstacles and then uses the information to find the local path. The robot can avoid the dynamic obstacle. Also, these are more efficient and less costly than global path planning methods. The major drawback of the local navigation methods is that they are basically steepest descent-based optimisation methods.

The navigation of the robot can be classified into two categories: conventional and heuristic methods. The conventional methods are considered as global path planning such as road map, Voronoi diagram, cell decomposition, and potential field. These methods are not mutually exclusive, and many methods use their hybridisations (Kamil et al., 2017). The heuristic methods are often suggested to overcome the limitations of conventional methods. Most of them are proposed based on heuristic and meta-heuristic algorithms. To solve the problem of mobile robot navigation, these methods are usually used due to their capability to search for finding the optimal solution over problem space. The significant approaches of this category are graph search algorithms (P. C. Chen & Hwang, 1998), sampling-based algorithms (Karaman & Frazzoli, 2011), probabilistic roadmap (PRM) (Santiago et al., 2017), rapidly-exploring random tree (RRT) (Adiyatov & Varol, 2013), artificial neural network (Duan & Huang, 2014), genetic algorithm (GA) (Nazarahari et al., 2019), ant colony optimisation (ACO) (Xiong et al., 2019), artificial particle swarm optimisation (PSO) (Mandava et al., 2019), simulated annealing (SA) (H. J. I. J. o. C. S. Miao & Security, 2010), bee algorithm (Li et al., 2018), bacteria forging algorithm (Liang et al., 2013), cuckoo search algorithm (Mohanty & Parhi, 2016), and fuzzy logic (FL) (Kladis et al., 2011).

The graph search algorithms are A*, D*, and D* Lite which have been developed from a well-known approach named Dijkstra's algorithm. The approaches are used to find the shortest path between the start point and destination point over the graph. In Sudhakara and Ganapathy, a new path planning algorithm has been presented based on the modified A* in the unknown and complex environment in the presence of static obstacles. To improve the optimal movement of the robot, a new parameter named the number of turnings (p(n)) has been added to the A* algorithm which the robot makes those tunings during its traverse. The results show that the proposed approach improves the drawbacks of path planning algorithms and performs better compared to the A* in terms of the elapsed time of travel.

To navigate the automated guided vehicle by its real differential, A comparison study between A* and D* Lite algorithms has been presented in Setiawan et al. (2014). The aim of this comparison is based on the computation time and the obtained path length. The obtained results demonstrate that D* has better performance than A* in terms of the shortest obtained path and the faster computation time. But there are some cases when A* is more effective than D* lite. To find the shortest path from the start point to the destination point, in Kumar Das et al. (2011) heuristic D* Lite algorithm has been used which navigates Khepera II mobile robot in the presence of the static obstacles in an unknown grid-map environment. The main aim of this paper is to plan an optimal or feasible path by avoiding collision with the located obstacles over the path and minimise the costs such as time, distance, and energy.

In Yan et al. (2013), the aim is to illustrate the efficiency of the probabilistic roadmap (PRM) and genetic algorithms in path planning for the mobile robot in simple and complex environments. Both algorithms are successfully able to find a path without collision in the given environment. The extracted path by GA is smoother than PRM. Therefore, the mobile robot moves with few turns. On the other hand, PRM consumes less processing resources and computation time compared to GA. So, PRM would be a good option for a reactive application.

A new path planning approach based on the modified GA has been proposed in Alsouly & Bennaceur (2016). It carefully designs a new GA with intelligent crossover to optimise the search process in static and dynamic environments. The proposed approach has compared A* with MGA approaches in a static scenario and the Improved GA in a dynamic scenario. The simulation results illustrate that the proposed GA is successful to find an optimal solution with fast execution time compared to the three other algorithms.

In Yan et al. (2013), the aim is to illustrate the efficiency of the probabilistic roadmap (PRM) and genetic algorithms in path planning for the mobile robot in simple and complex environments. Both algorithms are successfully able to find a path without collision in the given environment. The extracted path by GA is smoother than PRM. Therefore, the mobile robot moves with few turns. On the other hand, PRM consumes less processing resources and computation time compared to GA. So, PRM would be a good option for a reactive application.

A new path planning approach based on the modified GA has been proposed in (Alsouly & Bennaceur, 2016). It carefully designs a new GA with intelligent crossover to optimise the search process in static and dynamic environments. The proposed approach has compared A* with MGA approaches in a static scenario and the Improved GA in a dynamic scenario. The simulation results illustrate that the proposed GA is successful to find an optimal solution with fast execution time compared to the three other algorithms.

In (Mandava et al., 2019), a novel path planning method for the mobile robot is presented. In this paper, a combination of the artificial potential field with particle swarm optimisation and a 3-point smoothing method for static and dynamic obstacles is used. The results are shown that after applying the 3-point smoothing method and prediction technique, the obtained path and computational time are shorter. Authors in (Liu et al., 2017) have introduced an Improved Ant Colony Optimisation (ACO) algorithm for path planning in a grid environment. To find the globally optimal path in the search process, two methods of pheromone diffusion and geometric local optimisation are combined. First, for large search space, ACO is used and then it becomes smaller by using geometric local optimisation. As a result, the obtained path is optimised twice by using these algorithms. The results of the simulation show that the proposed method is notably effective. In (Xu et al., 2019), a new path rolling planning method is introduced based on grid modelling in a static and unknown environment. To find an optimised local navigation path, the proposed algorithm places a group of ants in the robot's current view with the target point of information. Therefore, the robot moves safely towards the target. On the other hand, the PSO algorithm has been combined with the proposed algorithm to further optimise. The results show that the robot reaches the target position by creating a free-collision path.

Another heuristic method used for path planning is bacterial foraging strategy (Liang et al., 2013). In this method, to determine an optimal and collision-free path between a starting point and a target point, the robot mimics the behaviour of bacteria. To evaluate the proposed method, two scenarios in the static environment with different number of obstacles are tested. The simulation result demonstrates that the robot mimics bacterial foraging behaviour and can be used in complex environments with both satisfactory accuracy and stability.

A novel path planning algorithm using an artificial bee colony is proposed in Li et al. (2018). In this paper, an artificial bee colony algorithm based on the physical strength of bees on the robot path planning method is proposed. The goal of this paper is to find the shortest path without collisions. The performance of the proposed method is compared with the traditional bee colony algorithm. The simulation results show that the proposed algorithm is effective and has faster convergence speed and optimal path. In Mohanty & Parhi (2016), a novel meta-heuristic algorithm is developed for path planning of mobile robots in an unknown or partially known environment with static obstacles. This meta-heuristic algorithm is based on the levy flight behaviour and brood parasitic behaviour of cuckoos. In order to reach a target point, a new objective function is formulated based on the robot, target, and obstacles that lead to seek target and to avoid the collision. The result of the proposed algorithm shows that the new algorithm is effective for finding an optimal path. A new path planning

algorithm for the mobile robot named as multi-operator based simulated annealing (SA) along with an additional four mathematical operators is suggested in (H. Miao, 2010). This algorithm is used in four dynamic environments with different complexities. The proposed algorithm requires less computation time. The obtained result of the simulation is compared with classic SA and GA approaches. The proposed method not only is effective in finding the optimal solution but also is more efficient in both on-line and off-line processing for path planning of robots in dynamic environments.

This paper is organised as follows: Section 2 presents the basic concept of the grasshopper algorithm. The formulation of the objective function for the mobile robot is discussed in section 3. Section 4 explains the simulations and computational results, and comparisons. Finally, section 5 is devoted to the concluding remarks.

Grasshopper algorithm

The Grasshopper Optimisation Algorithm (GOA) is inspired by the food-seeking behaviour of grasshoppers (Saremi et al., 2017). Grasshopper is basically an insect that is considered as a pest. These creatures are seen individually in nature, but they are considered as one of the largest swarms. The life cycle of the grasshopper is consisting of two important phases: larval and adulthood. The main characteristic of the swarm in the larval phase is slow movement or movement with small steps of the grasshoppers. On the other hand, larger and unexpected movement is one of the vital features of the swarm in the adulthood phase. In both phases of GOA, the process of source/food-seeking is divided into two parts, namely, exploration and exploitation.

In the exploration, grasshoppers tend to move rapidly, while in the exploitation phase they encouraged to move locally. These two functions and the target searching are fulfilled by the grasshoppers simultaneously. To simulate the swarming behaviour of grasshopper, the following mathematical model is given,

$$X_i = S_i + G_i + A_i \tag{1}$$

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

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

where s shows the strength of social forces, d_{ij} is the distance between the i^{th} grasshopper and j^{th} grasshopper that is calculated as $d_{ij} = \left|x_j - x_i\right|$. A unit vector between the i^{th} grasshopper and the j^{th} grasshopper is calculated as $\hat{d}_{ij} := \frac{x_j - x_i}{d_{ij}}$. The social forces defined by the function s is given as follows.

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

where f is the attraction intensity, l is the scale of attraction length, and r is the distance. The G component of the model 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 unit vector that is towards the centre of the 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. After substituting S, G and A in (1), the model becomes

$$X_i = \sum_{j=1, j\neq i}^{N} s\left(\left|x_j - x_i\right|\right) \frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e}_w \tag{6}$$

Here N shows the number of grasshoppers. In the optimisation algorithm, (6) is not used because it prevents the optimisation algorithm from exploring and exploiting the search space nearby a solution. In fact, this model of nymph grasshopper is designed for the swarm grasshopper that resides in free space. In addition, this mathematical model is not used for solving the optimisation problem, because the grasshoppers reach rapidly to comfort zone and the swarm does not converge to a specific point. The modified version of grasshopper position that is used for the update of grasshopper position is as follows:

$$X_i^d = c_1 \left(\sum_{\substack{j=1 \\ j\neq i}}^{N} c_2 \frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\right) \frac{x_j - x_i}{d_{ij}} \right) + \hat{T}_d \tag{7}$$

where ub_d is the upper boundary in the d^{th} dimension, lb_d is the lower boundary in the d^{th} dimension, and $\hat{T}_d$ shows the target value i.e. the best solution. c_1 and c_2 are coefficients to shrink the comfort zone, repulsion zone, and attraction zone. In (7) the gravity component is not applied, and it is considered that the direction of the wind is always towards the target direction. To calculate the next position of the grasshopper, the target position (global best), its current position and the position of all other grasshoppers are used. This means that GOA requires all agent searches to get involved in defining the next position of each grasshopper.

As already mentioned, the first section of (7) considers the current position of grasshopper according to all other grasshoppers. While the second section is utilised in order to reduce the agent movement around the target. This means that the second section considers the exploration and exploitation of the entire swarm around the target. Specifically, c_1 parameter is responsible to reduce the grasshopper movement around the target i.e. it balances the exploration and exploitation of the entire swarm around the target. Also, c_2 parameter reduces the comfort, attraction and repulsion zones between grasshoppers i.e. c_2 reduces space linearly to guide grasshoppers for finding the optimal solution in search space. The adaptive c_1 parameter contributes to the reduction in attraction and repulsion forces proportional to the number of iterations. As the iterations continue, c_2 parameter reduces the convergence of the search around the target. In order to balance exploration and exploitation, c_1 is proportionally reduced to increase the number of iterations. This method allows GOA to carry out effective exploitation towards the later stages of the optimisation. Similarly, according to the increment in the number of iterations, the c_2 value is reduced to minimise the comfort zone. Both parameters (c_1 and c_2) are considered as a single parameter and it is calculated as follows:

$$c = c_{max} - l\frac{c_{max} - c_{min}}{L} \tag{8}$$

where c_{max} and c_{min} are the maximum and minimum value of c. l shows the number of the current iteration and L represents the maximum number of iterations. The pseudo code for GOA is illustrated in Figure 1.

Problem formulation

The goal of this paper is to develop an effective path planning algorithm for a mobile robot in the presence of static and dynamic obstacles. Path planning for the mobile robot is to find optimal parameters to satisfy a set of specific requirements based on the objective function including obstacle detection, obstacle avoidance, face the trap-like situation, avoidance of randomly walking and generation of the optimal path. First, the optimisation problem of path planning is converted to a minimisation problem and then an objective function based on the position of the target and the

1: Objective function $f(\mathbf{x})$, $\mathbf{x}=(x_1, x_2, \ldots, x_d)$, d= no. of dimensions

2: Generate initial population of n grasshoppers $\mathbf{x}i= (i=1, 2, \ldots, n)$

3: Calculate fitness of each grasshopper

4: T = the best search agent

5: **while** stopping criteria not met **do**

6: Update c_1 using Eq. (8)

7: Update c_2 using Eq. (8)

8: **for each** grasshopper *gh* in population **do**

9: Normalize the distances between grasshoppers in [1,4]

10: Update the position of the *gh* by Eq. (7)

11: If required, update bounds of *gh*

12: **end for**

13: If there is a better solution, update T

14: **end while**

15: Output the T.

Figure 1. Pseudo code of GOA.

obstacle is defined. Finally, the grasshopper optimisation algorithm is used for solving this optimisation problem. The best global value of grasshopper is selected at each iteration and the robot moves towards these positions during the execution process. The robot frequently updates the information based on the sensory information, and according to this information, the objective function of the optimisation changes. If there are not any obstacles in the movement path of the robot, the robot will be able to find the target position directly without using GOA. Otherwise, the robot uses the obstacle avoidance mechanism to navigate in the unknown environment. The path planning problem here is solved for four different scenarios, namely, the path planning among static obstacles and dynamic obstacles, dynamic target and the combination of static and dynamic obstacles.

Obstacles

The environment consists of n obstacles, i.e. $O_1, O_2, \ldots, O_n$ and their position coordinates are represented as (xO_1, yO_1), (xO_2, yO_2), …, (xO_n, yO_n). The obstacles have circular and rectangular shapes. In this paper, the obstacles can be static and dynamic. If an obstacle is static then its velocity will be zero; otherwise, its velocity is (v_x, v_y) along x and y axes. The velocity of an obstacle is set randomly and is equal to or less than the velocity of the robot. The velocity and location vectors of obstacles (their speed and orientation) are unknown for the robot. It is assumed that the obstacles are detectable by the robot and move on the arbitrary path.

When the position and velocity of the obstacles are unknowns to the robot, the robot must be equipped with detectors and range sensors to obtain the necessary surrounding information. The robot is equipped with range sensors that provide 360° proximity information with radius R. When the robot moves to a new position in the configuration space, first it calculates its distance to the surrounding obstacles by reading its proximity sensor and then saves the results in a matrix that includes the positions of the obstacles. The velocity information can be inferred from the consecutive position measurements.

Robot sensing

As already mentioned, after an obstacle enters the range of the robot, the distance between it and the robot is determined and the direction of the moving obstacle is estimated (Kamil et al., 2017). It is suggested that an obstacle with (r_x, r_y) position is entered into the robot range at time t, and the position of the obstacle at t and $t + 1$ times are equal to (x_0, y_0) and (x_T, y_T), respectively, where T is the sampling period. If the position of the obstacle does not change, then the obstacle is static otherwise it is dynamic. Then the direction of the moving obstacle is estimated, and the robot will decide to select the next step depending on the future velocity vector of the obstacle and GOA. If the next trajectory of the obstacle intersects the robot path, the robot will be away from its original path. Also, if the distance between the robot and the obstacle increases, the obstacle is dynamic and it is moving away from the robot; otherwise, the obstacle is moving towards the robot.

The range sensor contains four circles with different radiuses R_1, R_2, R_3, R_4. The largest circle is the maximum range and the smallest circle is the minimum safe distance between the robot and the obstacle. There are two other intermediate circles between the smallest and largest circles. The used sensor range for this paper is demonstrated in Figure 2(a).

As seen from Figure 2(b), these circles are divided into four regions to estimate the orientation of moving obstacles. The first region is between 0 and 90°, the second region is between 90° and 180°, the third region is between −180° and −90°, the second region is between −90 °and 0 that is shown in Figure 2(b). The robot should save the point of instruction between the obstacle position and these circles and determines which of the regions have these intersection points. To estimate the

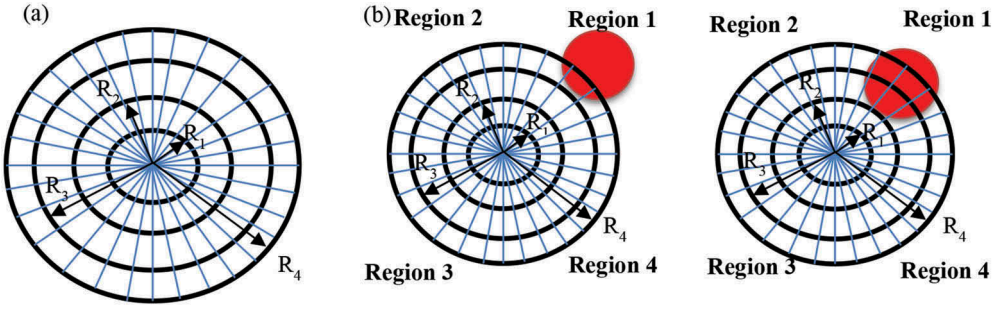

Figure 2. (a) The range sensor, (b) Detection of moving obstacle at t and $t + 1$ times.

velocity vector of each obstacle, the positions are first saved in two consecutive iterations. Then to estimate the next trajectory of the moving obstacle, the reading sensor saves 0 value for free path and 1 value for the regions inside the obstacle. If the prediction shows that the number of points of intersection does not change, then the obstacle is static; otherwise, the obstacle is dynamic. The orientation of the moving obstacle is determined by measuring the number of intersection points. If the number is increased, then the direction of the moving obstacle is towards the robot; if not, the obstacle is moving away from the robot.

Target-seeking behaviour

Here, the grasshopper is selected from the group of random grasshoppers that has the minimum distance from the target and the maximum distance from the obstacles. This is a continuously searching process for the grasshopper until it completely finds and reaches the target. The Euclidean distance between the target and grasshopper is defined by (9),

$$D_{GT} = \sqrt{(x_T - x_{G_i})^2 + (y_T - y_{G_i})^2} \tag{9}$$

where D_{GT} is the minimum Euclidean distance between the target and i^{th} grasshopper. (x_T, y_T) represents the coordinate of the target position. (x_{G_i}, y_{G_i}) is the coordinate of i^{th} grasshopper position.

Obstacle-seeking behaviour

The navigation problem is a difficult task for a mobile robot. For effective navigation in an unknown environment, the robot needs a mechanism of obstacle avoidance. When the obstacle is detected using the sensors, the grasshopper algorithm generates a random number of grasshoppers near the obstacle and the grasshopper with the best value of the objective function is selected. This grasshopper has selected in a way that its distance from the nearest obstacles is maximum. The robot occupies the position of the newly selected grasshopper and starts the procedure to search for the next grasshopper with the best value until it obtains a safe and optimal path. The best grasshopper is selected by using Euclidean distance between the grasshopper and the nearest obstacle that is represented as follows.

$$D_{GO} = \sqrt{(x_O - x_{G_i})^2 + (y_O - y_{G_i})^2} \tag{10}$$

where D_{GO} is the Euclidean distance between the neighbouring obstacle and position of i^{th} grasshopper. (x_O, y_O) shows the coordinate of the target position. Likewise, in a complex environment, the selection of the neighbouring obstacles is an important task to generate the optimal path. Therefore, the distance between the neighbouring obstacles is calculated by (11).

$$D_{RO} = \sqrt{(x_{O_n} - x_R)^2 + (y_{O_n} - y_R)^2} \tag{11}$$

Here, D_{RO} is the distance between the robot and the nearest obstacle, (x_{On}, y_{On}) coordinate shows the position of the nearest obstacle and (x_R, y_R) is the coordinate of the robot position.

The objective function of GOA for the optimisation problem of path planning is based on target-seeking and obstacle-seeking behaviours and this is formulated as in (12).

$$G_i = \lambda_1 . \frac{1}{\min O_n \epsilon O_s ||D_{GO}||} + \lambda_2 . ||D_{GT}|| \tag{12}$$

When an obstacle enters the active range of a sensor in O_s environment, its numbers are determined by the reading sensor of the robot. In (12), when the grasshoppers (G_i) get away from the obstacle,

the value of $\min O_n \epsilon O_s D_{GO}$ will be massive and when the grasshoppers become close to the target, the value of D_{GT} will decrease. Therefore, the objective function of GOA is a minimisation type of an optimisation problem that is utilised to find the optimal path for the mobile robot in the unknown environment. λ_1 and λ_2 are the controller parameters that are used for the safety of the path and the maximum and minimum path length in navigation. When λ_1 has the maximum value, the robot can move safely and without collision and when the λ_2 value is maximum, the path length is minimum. However, an appropriate selection of these parameters leads to a useful objective function for robot path planning. By integrating the aforementioned components, the path planning approach for the mobile robot is postulated. After modelling the environment, the position of the robot, the target, and the obstacles are initialised. Then, the behaviour of target seeking starts to find the target position so that the robot moves from its starting position to the target. Like most other path planning algorithms, the proposed algorithm starts by checking whether the target is reachable or not. If the target is reachable, then the robot moves towards the target straightly and the algorithm is finished. Otherwise, the proposed algorithm utilises the behaviour of obstacle seeking to predict the trajectories and positions of the obstacle(s). In this behaviour based on the range sensor, the robot senses its surrounding environment whether any obstacles exist. If the robot sensor detects an obstacle, then the next task is to determine whether it is static or dynamic. If the obstacle is static, then the grasshopper optimisation algorithm is activated, and it generates the population of grasshoppers randomly near to obstacle. The robot selects the appropriate grasshopper among the population based on (12) to reach the target which has been shown in Figure 3. But if the obstacle is dynamic, the robot first predicts the next velocity vector of moving obstacle and then determines the appropriate orientation based on GOA. If the robot finds a new position, then it will be the next position of the robot. When the robot reaches the target, the proposed algorithm calculates arrival time and path length. But if the robot could not find a new position, then there is not a promising solution for the path planning problem and the algorithm fails. The algorithmic flowchart of the proposed algorithm is illustrated in Figure 4. Steps involved in the GOA for mobile robot navigation are as follows:

1. Initialising the robot, goal and obstacle position.
2. Movement of robot towards the goal until it detects the obstacle.
3. If the obstacle exists in the path, then activate GOA.
4. Generating the population of grasshoppers randomly.
5. Selecting grasshopper with the best value of the objective function among the population to fit Equation (12).
6. Moving robot towards the current best grasshopper position.
7. Repeating steps 2 to 6 until the robot avoids the obstacle.

Results

The simulation environment

In this section, the proposed algorithm is simulated and tested in different environments. There are both static and dynamic obstacles in the environments studied. In these environments, the position (static and dynamic) and velocity of the (dynamic) obstacles are different. The starting position of the mobile robot is a hexagon and the target position is a square. The proposed algorithm is performed 50 times in each environment. The characteristics of environments are described in this section. The simulations are conducted in MATLAB 2017a environment using a 2.40-GHz Intel Core 7 Duo Processor. The simulation results of the environment with static obstacles are shown in Figure 4. The parameters used in the simulation are listed in Table 1. It is clear that the proposed algorithm provides a safe and short path along with acceptable arrival time at the first environment.

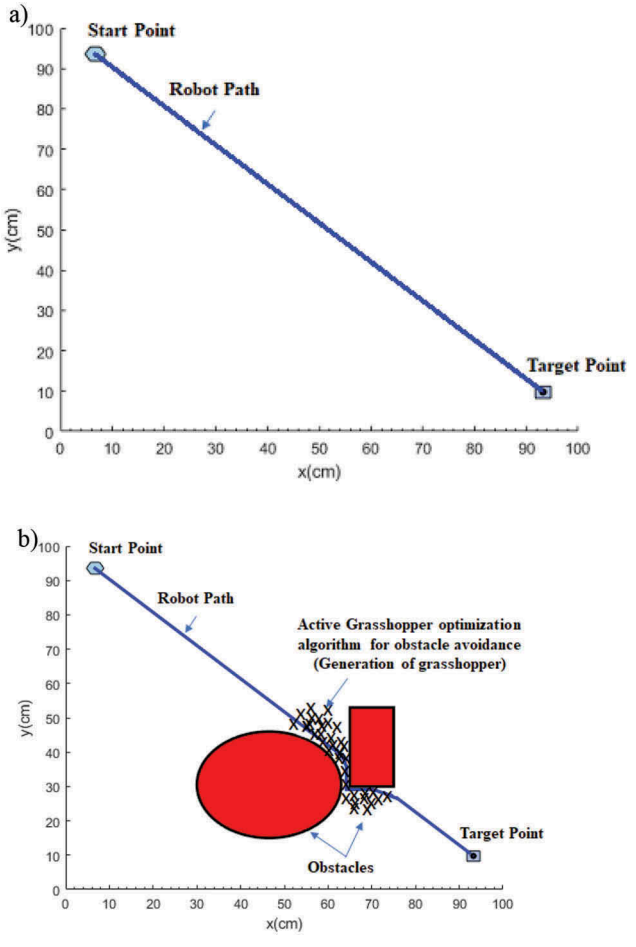

Figure 3. a) Navigation of the robot without the obstacle, b) Navigation of robot in the presence of obstacles using grasshopper optimisation algorithm.

Discussions

The simulation results are discussed in this section. In the first scenario, the robot moves from the start position to the target position. There are 30 static obstacles in different positions randomly. First, the robot navigates towards the target position according to the optimal path between starting and target positions. The environment is crowded, and the robot has to constantly check the environment after each step to make sure whether path safety is provided or not. If the sensor does not detect any obstacle on the robot path, it will continue its path towards the target and will record 0 in the sensor matrix. However, if it detects an obstacle (as seen in Figure 5b), it will try to avoid the collision with them by recording 1 in the sensor matrix and running GOA. Then the robot distinguishes whether the detected obstacle is static or dynamic. The robot utilises four circles of range sensors to calculate intersection points between obstacles and sensor circles for two-time intervals. The robot initially navigates towards the target position, but it confronts a circle obstacle on its path. The robot starts the grasshopper algorithm to avoid the collision.

First, the grasshopper algorithm generates the random number of grasshoppers near the obstacle and the grasshopper with the greatest target's objective is selected among the group of grasshoppers. The selected grasshopper has the maximum safe distance from the nearest obstacle. Then, the robot successfully passes from the near obstacles. After passing the obstacle, it returns to its

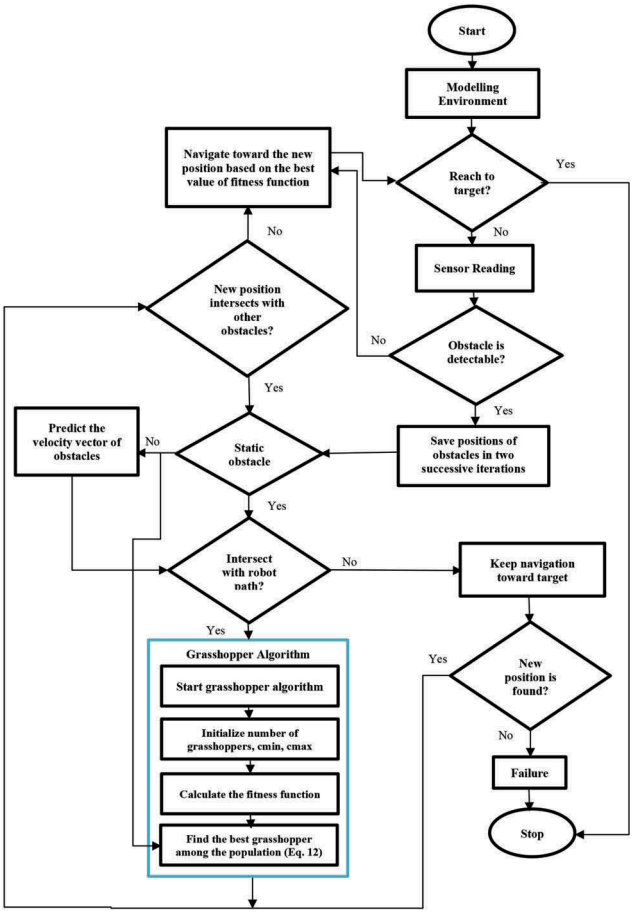

Figure 4. The flowchart of proposed navigation algorithm in unknown dynamic environment.

Table 1. Parameter settings for simulation.

Parameters	Values
Swarm Size (N)	100
Maximum Value (cMax)	1
Minimum Value (cMin)	0.00001
Fitting parameter (λ_1)	0.1–1
Fitting parameter (λ_2)	0.01–0.0001

original direction and path to reach the target on the optimal path (Figure 5c). The robot follows the optimal path until it detects another obstacle on its path. As mentioned above, the robot passes from two obstacles by using GOA and then returns to its optimal path (Figure 5d,e). In the following, the robot confronts with a circular obstacle and passes from it by applying GOA. To return to its original path, the robot is forced to cross the narrow path (Figure 5f). After passing the narrow path and returning to the original path, the robot reaches a rectangle obstacle and avoids collision with it by GOA again (Figure 5g). To reach the target, the robot continues its path until it faces a circular obstacle; and it starts the grasshopper optimisation algorithm to avoid collision (Figure 5h). Again, the robot tries to return to its optimal path but there are two obstacles on its path. After passing them, the robot reaches the target position successfully (Figure 5i). The arrival time to the target is 40.15 s. The path length based on Euclidean distance is 121.79 cm, and the minimum travelled path

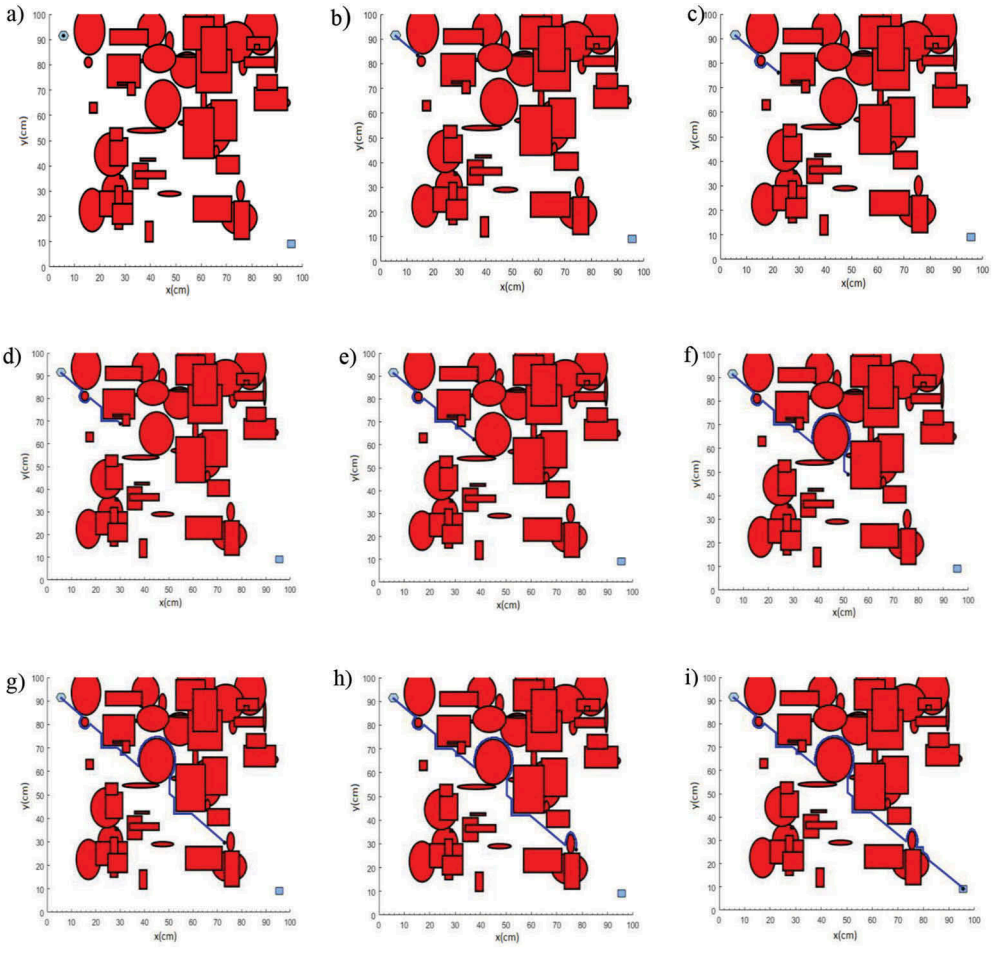

Figure 5. Simulation results of the proposed algorithm in the environment with static obstacles.

length by the robot is 140.79 cm. This difference is due to the crowded environment; therefore, the robot must cautiously pass from the near obstacles.

In the second scenario, there are 45 dynamic obstacles scattered to different positions randomly. The objects are now moving. The velocities of the obstacles are unknown, yet they are measured by the sensors. Also, we assume that the obstacles move slower than the robot so that the robot can handle the dynamic environment. The robot navigates towards the target and moves on its optimal path (Figure 6b). When the robot reaches the rectangular obstacle, it moves in (+x) and (+y) directions; at the same time, the second rectangle obstacle arrives (Figure 6c). Since the velocity of the first rectangular object is smaller, the robot waits to pass the second rectangle and moves in (-y) direction (Figure 6d). Then, the robot reaches the circular obstacle; after passing it, the robot returns to the original path again (Figure 6e). It follows the optimal path until it confronts with a rectangular obstacle (Figure 6f). Because the velocity of this rectangle is less than the velocity of the robot, it passes from obstacle by GOA and continues to its path until the robot faces another obstacle and again the robot navigates in (+x) and (+y) directions (Figure 6g). Then, the robot detects the circular obstacle and tries to avoid the collision. When the robot returns to the original path, it faces to rectangular and circular obstacles (Figure 6h). After passing the circular obstacle, the robot can achieve the target

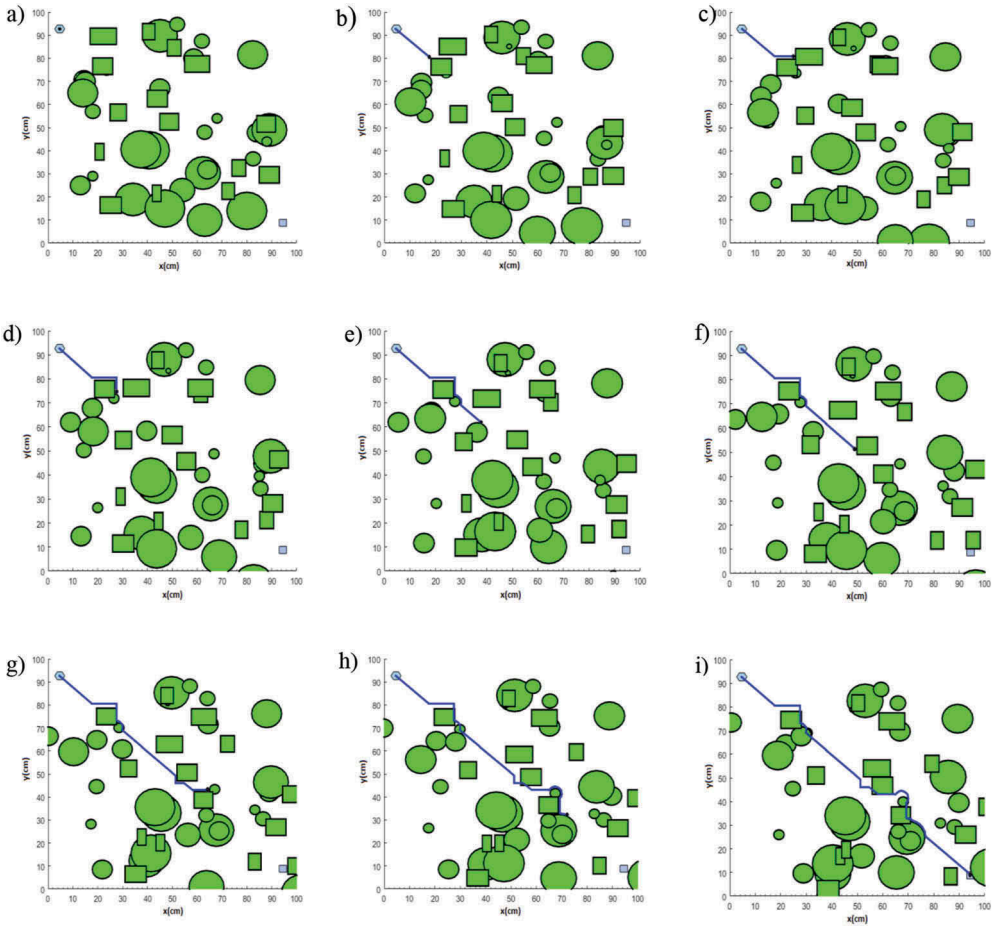

Figure 6. Simulation results of the proposed algorithm in the environment with dynamic obstacles.

because the path is free (Figure 6i). The arrival time is 38 s nearly. The path length based on Euclidean distance is 122.82 cm, and the minimum travelled path length by the robot is 135.33 cm.

The proposed algorithm is also tested for the dynamic target. The robot follows the dynamic target and reaches it successfully. All the movement steps are shown in Figure 7(a-i). The arrival time is 20.1 s in this scenario. The path length based on Euclidean distance is 95.13 cm, and the minimum path length travelled by the robot is 80.93 cm.

In the fourth scenario, the proposed algorithm is tested in the presence of a dynamic target and 30 static obstacles. The robot initially navigates towards the dynamic target. The robot moves on the original path and follows a dynamic target until it faces the circular obstacle (Figure 8b). As mentioned already, the robot passes from the obstacle and it continues its path on the optimal path (Figure 8c). Because the rectangle obstacle is on the original path, the robot utilises GOA to pass the given obstacle (Figure 8d). To return to the original path, the robot must pass from the narrow path (Figure 8e,f). The robot encounters the circular obstacle in the path (Figure 8g). After passing it, the robot arrives at two rectangular obstacles and then it starts GOA for avoiding (Figure 8h). Finally, the robot follows a dynamic target and reaches it (Figure 8i). The arrival time of this scenario is 40.8 s. Also, the path length based on Euclidean distance is 121.99 cm, and the minimum path length travelled by the robot is 126.53 cm.

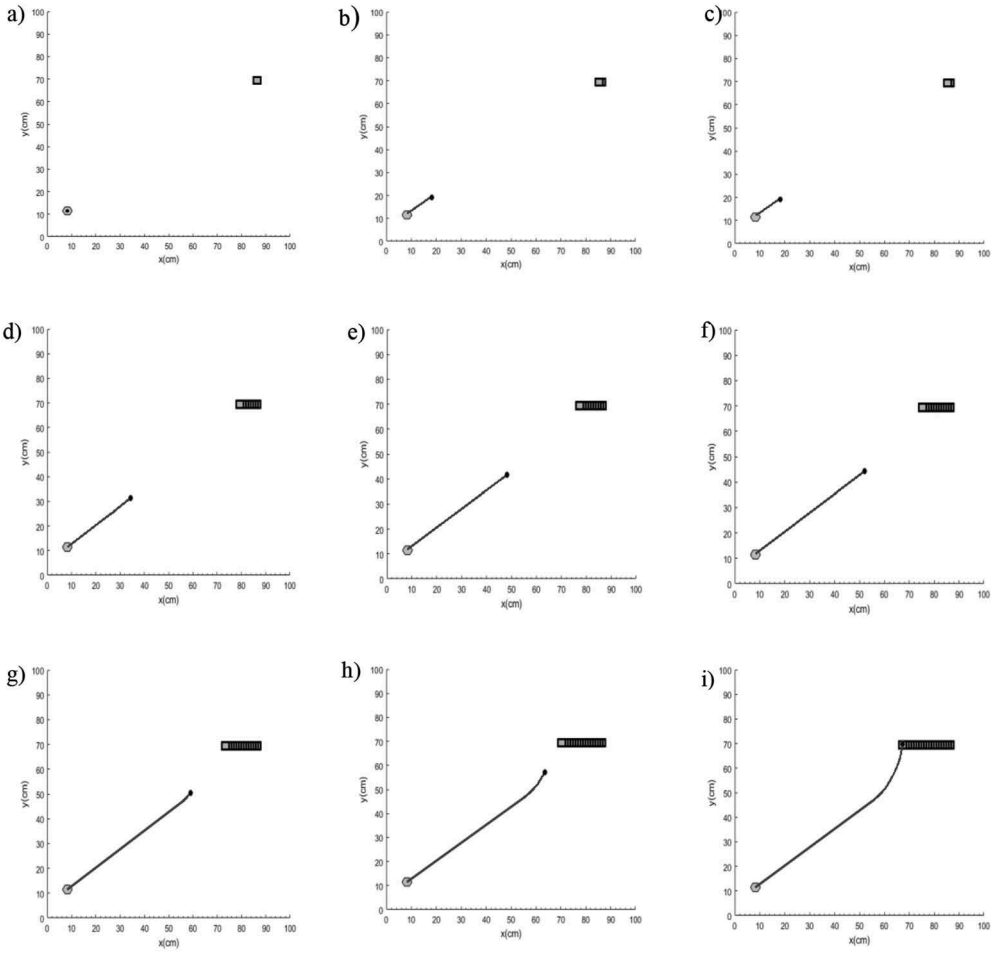

Figure 7. Simulation results of the proposed algorithm in presence of dynamic target without obstacle.

In the fifth scenario, there is a combination of static and dynamic obstacles. The number of the static obstacles is 20 which are shown with the rectangle and curved corners and the number of dynamic obstacles are 15 that are shown with rectangles having sharp corners and circles. First, the robot navigates towards the target (Figure 9b). There are static and dynamic obstacles. When the robot detects the dynamic obstacle, the velocity vector of the dynamic obstacle is recorded for two-time intervals to predict the trajectory of the obstacle. The robot subsequently decided on the best next step based on GOA (Figure 9c). The dynamic obstacle is in the +x direction and the robot decides to navigate towards the -x direction to avoid the dynamic obstacle (Figure 9d). After avoiding the dynamic obstacle, the robot continues its navigation to reach to the target until it detects two static obstacles (Figure 9e). After passing the obstacles, the robot returns to its original navigation and navigates towards the target (Figure 9f). On the original path, there are a static obstacle and a dynamic obstacle that moves circularly (Figure 9g). The robot navigates towards the optimal path after passing the obstacles (Figure 9h). Finally, the robot reaches the target (Figure 9i). The arrival time of this scenario is 37 s. Also, the path length based on Euclidean distance is 125.74 cm, and the minimum path length travelled by the robot is 140.30 cm.

The proposed algorithm is successfully implemented in all scenarios. Also, the results have shown that the proposed algorithm has important features including low running time, high optimality,

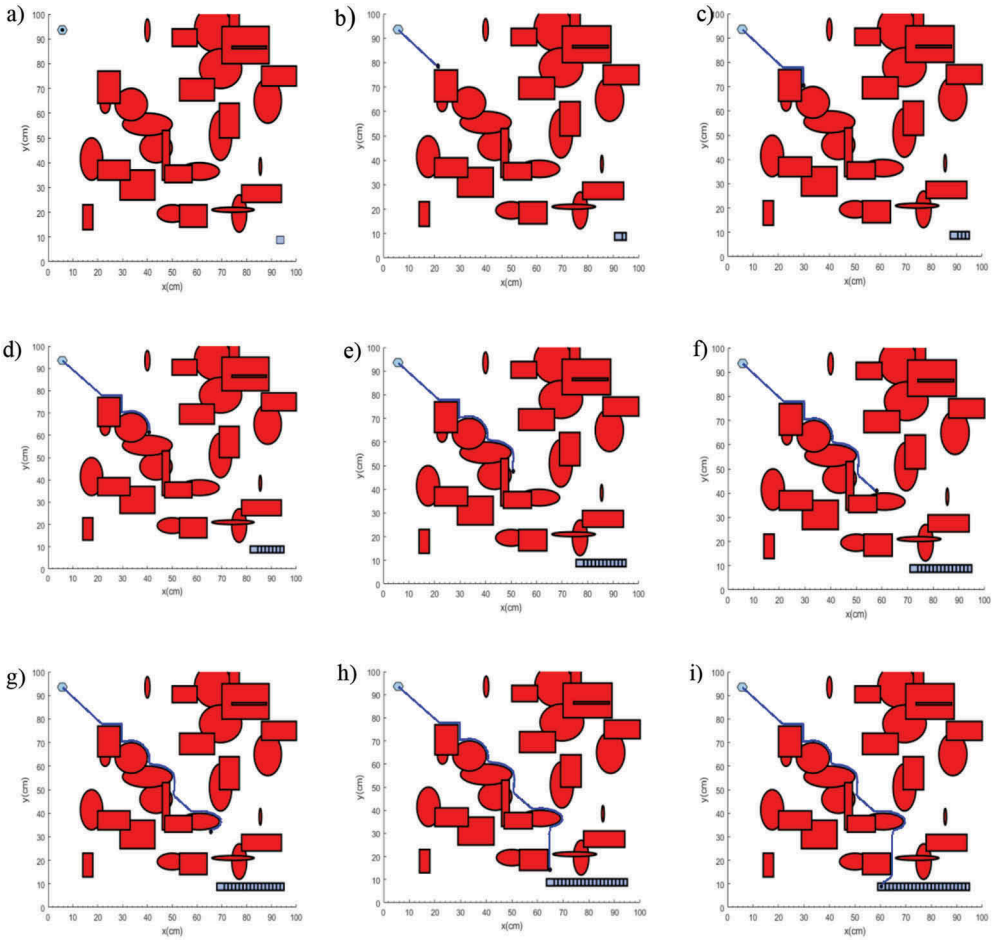

Figure 8. Simulation results of the proposed algorithm in presence of dynamic target among with obstacle.

high stability. For all test problems, the failure rate is zero. Also, the path length and arrival time of the proposed controller for all the scenarios are presented in Table 2. For example, as seen in Table 2, for static obstacle scenarios, the number of obstacles, actual path length, path length travelled by the robot and arrival time are equal to 30, 121.79, 149.79 and 42.5, respectively.

The performance of the algorithm proposed here is tested and evaluated with the several known methods such as Particle Swarm Optimisation (PSO) (Mandava et al., 2019), Genetic Algorithm (GA) (Nazarahari et al., 2019), D* (Likhachev et al., 2005), Neuro-Fuzzy (Mohanty & Parhi, 2014), and Rapidly-exploring Random Tree star (RRT*) (Lan & Di Cairano, 2015). The comparison results are illustrated in Figure 10. There are 10 static and 5 dynamic obstacles in the test environment that the movement direction of dynamic obstacles is shown by arrows. The actual distance between the start and target positions is 122.96 cm. The performance of our proposed method is compared with the performance of the methods mentioned above in terms of safety, path length, computational time and complexity. A comparison of the path lengths and computation times are presented in Table 3.

All the heuristic methods can find a safe path between start and target positions in the given environment. But some of the used methods have drawbacks such as the poor quality of the resulting path, high running times of the planner, and inability to solve complex problems effectively. It is clear from Table 3 that the path length obtained by the proposed controller is shorter than

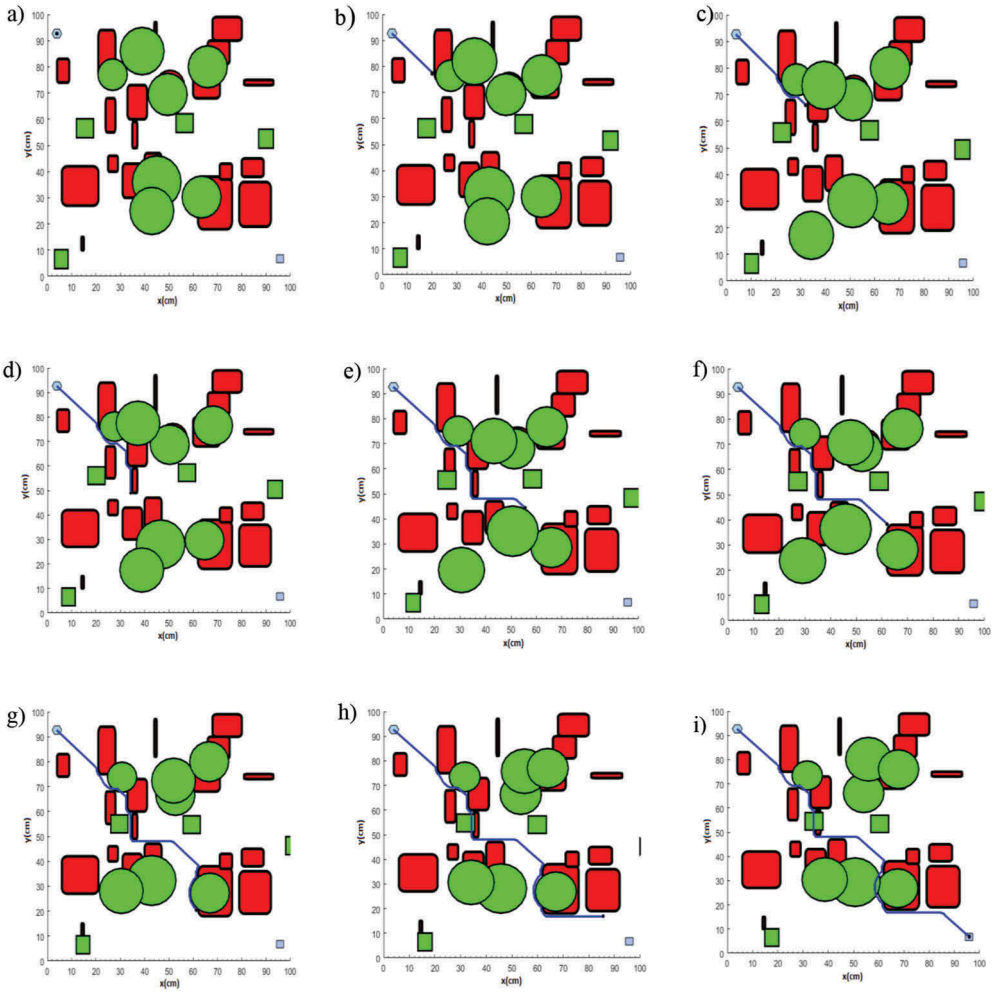

Figure 9. Simulation results of unknown dynamic environment among with static and dynamic obstacle.

Table 2. The path length and arrival time of proposed controller for all of scenarios.

Scenario	Number of obstacles	Actual path length (cm)	Path length travelled by the robot (cm)	Arrival time (Sec)
Static obstacles	30	121.79	149.79	40.15
Dynamic obstacles	45	122.82	135.33	38
Dynamic target	0	95.13	80.93	20.1
Dynamic target with static obstacle	30	121.99	126.53	33.8
Combination of static and dynamic obstacle	35	125.74	140.3	37

other heuristic controllers and computation time of the proposed approach is shorter. Therefore, the proposed method can safely find a safe and collision-free path in crowded and dynamically changing environments. The advantage of using a grasshopper algorithm for optimisation problems is handling both linear and nonlinear problems, high convergence speed, low computational cost. Because the computational cost is low, it can be used in real-time applications. In order to improve and speed up the robot for future works, we can combine the proposed algorithm with other heuristic algorithms such as Genetic algorithm, PSO, ACO, Fuzzy logic.

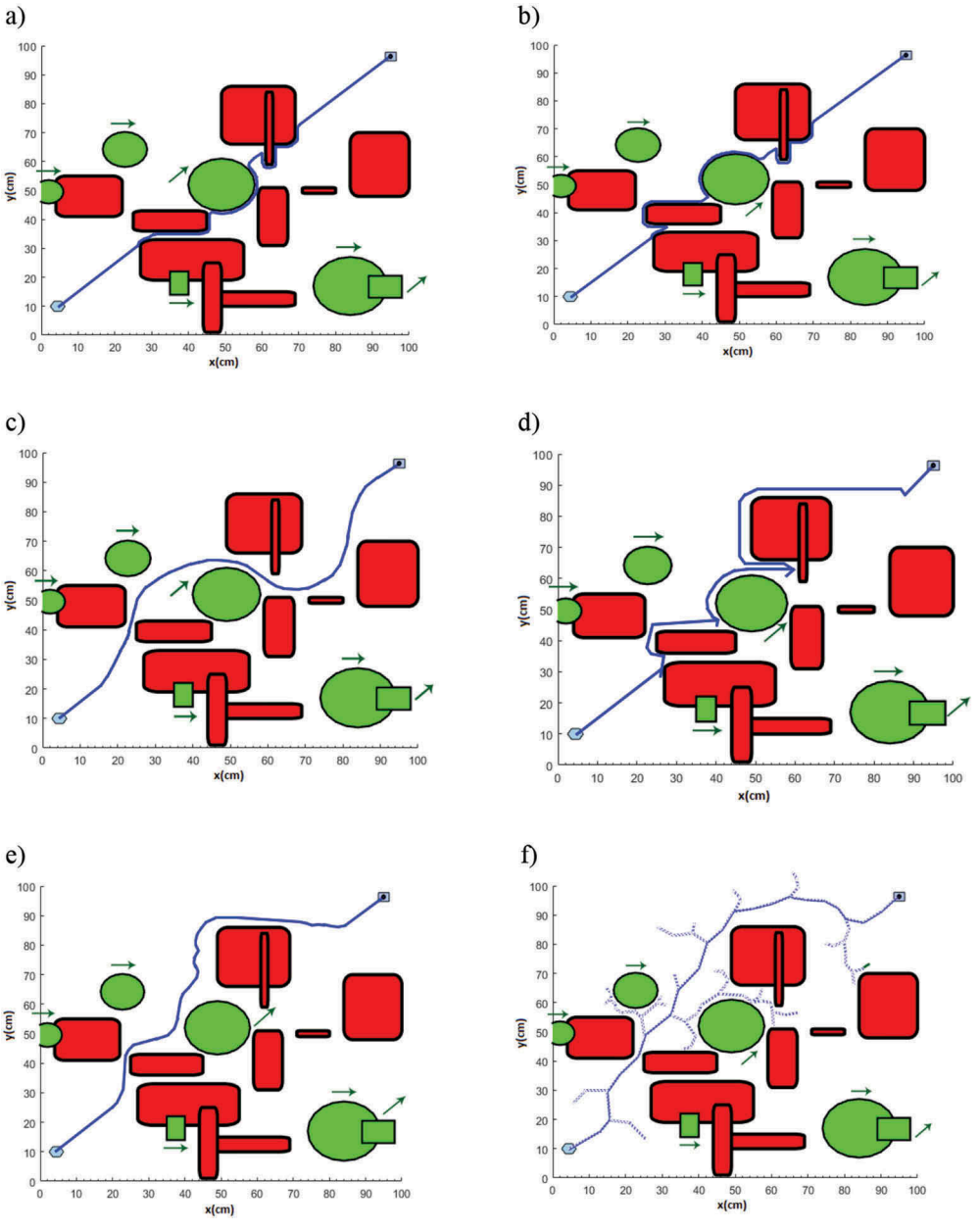

Figure 10. The performance comparison results a) GOA b) PSO c) GA d) D* e) Neuro-Fuzzy f) RRT*.

Table 3. The path length, computational time of heuristic methods for path planning.

Heuristic algorithm	Path length (cm)	Arrival time (Sec)
Grasshopper algorithm	144.34	32.02
PSO	161	68.2
GA	144.5	94.31
D*	198.08	132.8
Nero-Fuzzy	145.60	122.6
RRT*	145.96	105.5

Conclusions

In this paper, a novel path planning is presented using grasshopper algorithm in unknown and dynamic environments. For this purpose, a sensor-based online technique was used to obtain a collision-free path. To predict the obstacles, the main idea is to use a range sensor with four circles with different radii. These circles are divided into four sections that estimate the direction of dynamic obstacles. The robot saves the intersection points between the obstacle positions and these circles and then determines which of these areas have these intersection points. If the intersection points do not change in two sampling periods, then the obstacle is static; otherwise, it is dynamic. Also, the direction of the dynamic obstacle is determined by measuring the number of these intersection points. Then, the robot tries to avoid obstacles and find the optimal path by using the grasshopper algorithm. The location, shape, and velocity of obstacles are not available to the robot. Several scenarios are used to evaluate our proposed technique. The simulation results have shown that the proposed controller successfully guides the robot towards the target, effectively avoids collision and finds the shortest and optimal path in minimum time. Besides, the proposed controller is compared with PSO, GA, D*, Neuro-Fuzzy, and RRT*. The comparison results indicate the prominent features of the proposed approach. In addition, the proposed approach considers time, unexpected obstacles and velocity vector of the obstacles making it a good candidate for real-time applications.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Zahra Elmi is currently a Ph.D. student in Computer Engineering at Hacettepe University, Ankara, Turkey. She received her B.Sc. and MSc. Degrees in Computer Engineering from Islamic Azad University of Khoy and Qazvin, Iran, in 2006 and 2009, respectively. She is working on path planning using a heuristic algorithm in a dynamic environment, autonomous vehicles, vehicle dynamics and control, and advanced control methods and their real-time applications.

Mehmet Önder Efe received the Ph.D. degree in 2000 from Boğaziçi University, Istanbul, Turkey. He is currently a Professor in the Department of Computer Engineering of Hacettepe University, Ankara, Turkey. he was the head of the department (2015-2018) and the head of the computer hardware division since 2013. He has authored or coauthored more than 140 journal and conference publications focusing on the applications of computational intelligence, unmanned aerial vehicles and systems, and control theory, as well as 4 books, 11 book chapters, and 3 edited books. Dr. Efe was the Head of the IEEE CSS Turkey Chapter from January 2007 to December 2008. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the Transactions of the Institute of Measurement and Control, the International Journal of Industrial Electronics and Control and Advances in Fuzzy Systems.

References

Adiyatov, O., & Varol, H. A. (2013). Rapidly-exploring random tree based memory efficient motion planning. In *2013 IEEE international conference on mechatronics and automation* (pp. 354–359).

Alsouly, H., & Bennaceur, H. (2016). *Enhanced genetic algorithm for mobile robot path planning in static and dynamic environment*. IJCCI (ECTA).

Behroo, M., & Banazadeh, A. (2015). Near-optimal trajectory generation, using a compound B-spline interpolation and minimum distance criterion with dynamical feasibility correction. *Robotics and Autonomous Systems*, *74*, 79–87. https://doi.org/10.1016/j.robot.2015.07.003

Buniyamin, N., Ngah, W. W., Sariff, N., & Mohamad, Z. (2011). A simple local path planning algorithm for autonomous mobile robots. International journal of systems applications. *Engineering & Development*, *5*(2), 151–159. Retrieved from https://pdfs.semanticscholar.org/109e/f5b91a1411ba3da975993439c13a348a2444.pdf

Chen, B., & Quan, G. (2008). NP-hard problems of learning from examples. In *2008 fifth international conference on fuzzy systems and knowledge discovery* (Vol. 2, pp. 182–186).

Chen, P. C., & Hwang, Y. K. (1998). SANDROS: A dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, *14*(3), 390–403. https://doi.org/10.1109/70.678449

Duan, H., & Huang, L. (2014). Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning. *Neurocomputing*, *125*, 166–171. https://doi.org/10.1016/j.neucom.2012.09.039

Kamil, F., Hong, T. S., Khaksar, W., Moghrabiah, M. Y., Zulkifli, N., & Ahmad, S. A. J. E. S. W. A. (2017). New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior. *Expert Systems with Applications, 86*, 274–291. https://doi.org/10.1016/j.eswa.2017.05.059

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, *30*(7), 846–894. https://doi.org/10.1177/0278364911406761

Kladis, G. P., Economou, J. T., Knowles, K., Lauber, J., & Guerra, T.-M. (2011). Energy conservation based fuzzy tracking for unmanned aerial vehicle missions under a priori known wind information. *Engineering Applications of Artificial Intelligence*, *24*(2), 278–294. https://doi.org/10.1016/j.engappai.2010.10.013

Kumar Das, P., Patro, S., Panda, C., & Balabantaray, B. (2011). D* lite algorithm based path planning of mobile robot in static Environment. *International Journal of Computer & Communication Technology*, *2*, 32–36. Retrieved from https://www.researchgate.net/profile/Bunil_Balabantaray2/publication/260346216_D_lite_algorithm_based_path_planning_of_mobile_robot_in_static_Environment_D_lite_algorithm_based_path_planning_of_mobile_robot_in_static_Environment/links/547860020cf293e2da28e5df.pdf

Lan, X., & Di Cairano, S. (2015). Continuous curvature path planning for semi-autonomous vehicle maneuvers using RRT. In *2015 European control conference (ECC)* (pp. 2360–2365). IEEE.

Li, X., Huang, Y., Zhou, Y., & Zhu, X. (2018). Robot path planning using improved artificial bee colony algorithm. In *2018 IEEE 3rd Advanced information technology, electronic and automation control conference (IAEAC)* (pp. 603–607). IEEE.

Liang, X.-D., Li, L.-Y., Wu, J.-G., & Chen, H.-N. (2013). Mobile robot path planning based on adaptive bacterial foraging algorithm. *Journal of Central South University*, *20*(12), 3391–3400. https://doi.org/10.1007/s11771-013-1864-5

Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., & Thrun, S. (2005). *Anytime dynamic A*: An anytime, replanning algorithm*. ICAPS.

Liu, J., Yang, J., Liu, H., Tian, X., & Gao, M. (2017). An improved ant colony algorithm for robot path planning. *Soft Computing*, *21*(19), 5829–5839. https://doi.org/10.1007/s00500-016-2161-7

Mandava, R. K., Bondada, S., & Vundavilli, P. R. (2019). *An optimized path planning for the mobile robot using potential field method and PSO algorithm soft computing for problem solving*. Springer.

Miao, H. (2010). A multi-operator based simulated annealing approach for robot navigation in uncertain environments. *International Journal of Computer Science and Security*, *4*(1), 50–61.

Miao, H. J. I. J. o. C. S., & Security. (2010). A multi-operator based simulated annealing approach for robot navigation in uncertain environments. *International Journal of Computer Science and Security, 4*(1), 50–61. Retrieved from https://www.researchgate.net/profile/Anagha_Kulkarni3/publication/42766840_Cache_Coherency_in_Distriuted_File_System/links/560a432d08ae1396914baeff.pdf#page=54

Mohanty, P. K., & Parhi, D. R. (2014). A new intelligent motion planning for mobile robot navigation using multiple adaptive neuro-fuzzy inference system. *Applied Mathematics & Information Sciences*, *8*(5), 2527. https://doi.org/10.12785/amis/080551

Mohanty, P. K., & Parhi, D. R. (2016). Optimal path planning for a mobile robot using cuckoo search algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, *28*(1–2), 35–52. https://doi.org/10.1080/0952813X.2014.971442

Nazarahari, M., Khanmirza, E., & Doostie, S. (2019). Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications*, *115*, 106–120. https://doi.org/10.1016/j.eswa.2018.08.008

Patle, B., Pandey, A., Jagadeesh, A., & Parhi, D. (2018). Path planning in uncertain environment by using firefly algorithm. *Defence Technology*, *14*(6), 691–701. https://doi.org/10.1016/j.dt.2018.06.004

Santiago, R. M. C., De Ocampo, A. L., Ubando, A. T., Bandala, A. A., & Dadios, E. P. (2017). Path planning for mobile robots using genetic algorithm and probabilistic roadmap. In *2017IEEE 9th international conference on humanoid, nanotechnology, information technology, communication and control, environment and management (HNICEM)* (pp. 1–5). IEEE.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software*, *105*, 30–47. https://doi.org/10.1016/j.advengsoft.2017.01.004

Setiawan, Y. D., Pratama, P. S., Jeong, S. K., Duy, V. H., & Kim, S. B. (2014). *Experimental comparison of A* and D* Lite path planning algorithms for differential drive automated guided vehicle AETA 2013: Recent advances in electrical engineering and related sciences*. Springer.

Sudhakara, P., & Ganapathy, V. Path planning of a mobile robot using amended a-star algorithm. *International Journal of Control Theory and Applications*, 9, 489–502.

Xiong, C., Chen, D., Lu, D., Zeng, Z., & Lian, L. (2019). Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization. *Robotics and Autonomous Systems*, *115*, 90–103. https://doi.org/10.1016/j.robot.2019.02.002

Xu, S., Ho, E. S., & Shum, H. P. (2019). A hybrid metaheuristic navigation algorithm for robot path rolling planning in an unknown environment. *Mechatronic Systems and Control*, *47*(4), 46072. https://doi.org/10.2316/J.2019.201-3000

Yan, F., Liu, Y.-S., & Xiao, J.-Z. (2013). Path planning in complex 3D environments using a probabilistic roadmap method. *International Journal of Automation and Computing*, *10*(6), 525–533. https://doi.org/10.1007/s11633-013-0750-9