Soft Comput
DOI 10.1007/s00500-015-1825-z

METHODOLOGIES AND APPLICATION

Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach

Alejandro Hidalgo-Paniagua[1] · **Miguel A. Vega-Rodríguez**[1] · **Joaquín Ferruz**[2] · **Nieves Pavón**[3]

Abstract Currently, autonomous robotics is one of the most interesting and researched areas of technology. At the beginning, robots only worked in the industrial sector but, gradually, they started to be introduced into other sectors such as medicine or social environments becoming part of society. In mobile robots, the path planning (PP) problem is one of the most researched topics. Taking into account that the PP problem is an NP-hard problem, multi-objective evolutionary algorithms (MOEAs) are good candidates to solve this problem. In this work, a new multi-objective approach based on the flashing behavior of fireflies in nature, the multi-objective firefly algorithm (MO-FA), is proposed to solve the PP problem. This proposed algorithm is a swarm intelligence algorithm. The proposed MO-FA handles three different objectives to obtain accurate and efficient solutions. These objectives are the following: the path safety, the path length, and the path smoothness (related to the energy consumption). Furthermore, and to test the proposed MOEA, we have used eight realistic scenarios for the path's calculation. On the other hand, we also compare our proposal with other approaches of the state of the art, showing the advantages of MO-FA. In particular, to evaluate the obtained results we applied specific quality metrics. Moreover, to demonstrate the statistical evidence of the obtained results, we also performed a statistical analysis. Finally, the study shows that the proposed MO-FA is a good alternative to solve the PP problem.

Keywords Path planning · MO-FA · Swarm intelligence · Robotics · Realistic maps · Energy consumption

Communicated by V. Loia.

✉ Alejandro Hidalgo-Paniagua
ahidalgop@unex.es

Miguel A. Vega-Rodríguez
mavega@unex.es

Joaquín Ferruz
ferruz@cartuja.us.es

Nieves Pavón
npavon@dti.uhu.es

[1] Department of Technologies of Computers and Communications, University of Extremadura, Polytechnic School, Cáceres, Spain

[2] Department of Systems Engineering and Automation, University of Sevilla, Higher Technical School of Engineering, Sevilla, Spain

[3] Department of Information Technology, University of Huelva, Higher Technical School of Engineering, Huelva, Spain

1 Introduction

Nowadays, robotics is one of the most important areas of technology due to its rapid development and implementation in many aspects of the real world, and not only in the industrial sector. At first glance, robots can be classified in two well-differentiated groups: mobile and non-mobile robots. The first ones are equipped with a locomotion system that allows them to navigate through a specific environment. In robot navigation, the main problem is to calculate a feasible path from a starting to a target point of the environment. This problem is known as path planning (PP) problem (Shih et al. 2013). Currently, PP is one of the most researched topics in the field of robotics.

In this sense, several approaches tackling the PP problem have been proposed. The main disadvantage of these approaches is that most of them are only focused on a single objective, normally minimizing the path length (LaValle 2006). However, from our point of view, in the case of PP, it is essential to handle several objectives simultaneously with

Published online: 20 August 2015

the aim of obtaining accurate solutions. Specifically, three fundamental objectives can be taken into account: the path safety, the path length, and the path smoothness. The first one, the path safety, is related to the objects of the environment presented in the path. The paths containing fewer objects will be the best (safest). The path length objective is focused on obtaining paths as short as possible. This objective is mainly related to the robot operation time, since a shorter path will be traveled in less time. Finally, it is important to highlight that robots do not have an infinite energy source (usually batteries are used). The last objective, the path smoothness, has as a goal to minimize the number of bends of the path. The path smoothness objective is related to the energy consumption, due to the influence of the number and magnitude of turns on the consumed energy (Ahmed and Deb 2013; Jun and Qingbao 2010). Thus, a path in which the robot has to turn minimally will involve less energy consumption.

Path planning (PP) is an NP-hard optimization problem (Davoodi et al. 2013); for this reason it can be tackled by using MOEAs.

In this manuscript, we present an efficient multi-objective version of the firefly algorithm (FA) (Yang 2010), a swarm intelligence algorithm based on the flashing behavior of fireflies in nature. Some important contributions of our work are that, unlike many previous works, we tackle the PP problem in a multi-objective manner, applying an MOEA not previously used, and using realistic maps for the calculation of the paths. For the PP problem, we handle the three essential objectives explained above, that is, the path safety, the path length, and the path smoothness (directly related to the energy consumption). This is also important because many previous works use less objectives or are not aware of the energy consumption. Regarding the evolutionary operators, we developed our own specific operators to solve PP. This is another contribution of our work.

As NSGA-II (Non-dominated Sorting Genetic Algorithm II) (Deb et al. 2002) is the MOEA mainly used by other authors who tackled the same problem (with the same objectives) in a multi-objective manner, we used this algorithm as a reference, comparing the results of multi-objective firefly algorithm (MO-FA) with several implementations of NSGA-II proposed by other authors. This comparison is based on a complete statistical study, with 8 scenarios and 31 independent repetitions per experiment.

The rest of the paper is organized as follows. Section 2 discusses the related work. In Sect. 3, the environment modeling, the path encoding, and the objectives considered to solve the PP problem are clearly explained. A brief explanation of both, the MOEAs and the evolutionary operators used by them, is presented in Sect. 4. In this section, we also explain two new mechanisms to improve the initial population of MOEAs. In Sect. 5, the methodology used to test MOEAs, that is, the datasets (scenarios), the parameters configuration, and the metrics used are presented. Finally, the results, comparisons with the state of the art, and conclusions are shown and analyzed in Sects. 6 and 7, respectively.

2 Related work

In recent years, several studies have been proposed in which MOEAs are applied to tackle the PP problem. Despite this, some of these works really handle a single-objective optimization (EAs). For example, in Geetha et al. (2011), Guo et al. (2009), Hao and Qin (2011), Krishnan et al. (2009), Masehian and Sedighizadeh (2010), Masehian and Sedighizadeh (2010), and Masehian and Sedighizadeh (2010), the authors use a single-objective optimization, although in the paper titles indicate that multiple objectives are used. More specifically, all these articles are based on an aggregation scheme (weighted sum) to calculate the fitness function. On the other hand, almost all the real MOEAs applied to the PP problem used the popular NSGA-II algorithm. In Wei and Liu (2010), an NSGA-II was presented, which manages only two objectives. In particular, the algorithm optimizes the path length and the path curvature. In Ahmed and Deb (2011), Chang and Liu (2009), and Davoodi et al. (2013), different variants of NSGA-II were proposed, which also use two objectives: the path length and the path safety (referred to the obstacles), but ignoring the energy consumption. Other works in which the authors used these same objectives, also without taking into account the energy consumption, can be found in Gong et al. (2011), Geng et al. (2013), Wang et al. (2009), and Zhang et al. (2013). On the other hand, Sedaghat also applied NSGA-II and proposed a variant of these two previous objectives, in particular she used the path length in combination with the path difficulty to solve the problem in Sedaghat (2011), also ignoring the energy consumption. These studies took into account only two objectives, and the MOEA we present in this study uses three objectives to solve the problem (our third objective is the path smoothness, related to the energy consumption). For this reason, we could not compare with their obtained results. Other variants of MOEAs which optimize two different objectives, in this case the path length and the path smoothness, can be found in Mo et al. (2013) and Wang and Zhu (2013).

On the other side, some authors applied MOEAs optimizing three different objectives, for example, in Ahmed and Deb (2013) and Jun and Qingbao (2010) an NSGA-II algorithm was used that took into account the path length, the path safety, and the path smoothness. These authors optimized the same objectives as we handle in our work. For this reason, we have used these proposals to do a fair comparison with the state of the art, showing the goodness of our approach.

Finally, in Kim and Kim (2009) and Kim et al. (2009), MOEAs were applied to solve the PP problem in a robot

soccer system. This robotics environment is very different to our robotics environment. In fact, they used very different objectives. More specifically, they used the following three objectives: path time, heading direction, and posture angle error. Furthermore, none of them takes into account the energy consumption. Again, all of these authors compared their obtained results with the corresponding NSGA-II algorithm.

In this paper, we tackle the PP problem by using a swarm intelligence algorithm known as MO-FA. MO-FA is based on the flashing behavior of fireflies in nature. Unlike other authors, in this work we use realistic maps of the environment. On the other hand, due to the importance and popularity of the NSGA-II algorithm in the related literature, we also compare our results with the ones obtained by the different NSGA-II algorithms proposed by Ahmed and Deb (2013) and Jun and Qingbao (2010). To the best of our knowledge, no other works have tackled the PP problem by using a MO-FA.

3 Path planning

The PP problem consists in finding a path that allows a robot to move from a starting point to a target point in a certain environment. In the following subsections, we explain the environment modeling, the path encoding, and the handled objectives to solve the PP problem. Note that, in this work, we consider the robot and the environment geometry in a 2D space.

3.1 Environment modeling

When working with mobile robots, it is essential to build a computational representation of the environment to carry out motion planning operations. As we said at the beginning of Sect. 3, in this work, the environment is modeled in a 2D space using a grid structure. Unlike other authors who tackled the PP problem too, we used realistic maps of the environment. The process to get the final computational representation of the environment involves two steps. The first one consists of partitioning the real map of the environment. As the final representation will be based on a 2D grid, the map is partitioned into several rows and columns. Each region of the map determined by a specific row and column is known as a partition. For this reason, the resolution of the final representation will depend on the number of rows and columns used to partition the source map. Figure 1 shows an example of the map partitioning.

Taking as input the partitioned map, the second step aims to obtain a grid structure representing the occupancy of each partition. The occupancy of a partition refers to the amount of objects contained in it, and is expressed as a percentage.

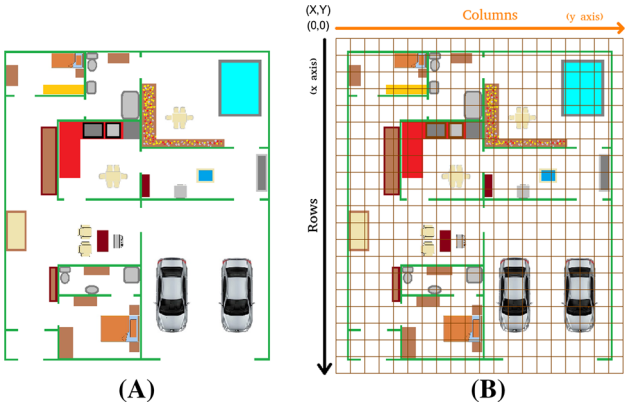

Fig. 1 Map partitioning: **a** source map, and **b** partitioned map

Taking this into account, if in a partition any object does not appear, the value in the final grid will be 0 %. Conversely, if a partition is fully occupied by objects, the corresponding value of the cell in the final grid will be equal to 100 %. In the same way, a partition neither fully occupied nor completely empty will have a value between 0 and 100 %. To detect the objects existing in the environment, we used two different colors in the source map. The background of the map is always white. This color represents the empty space. The walls of the environment have always been associated with the green color. Any other object of the environment will have any other color (neither white nor green). The reason for representing the walls with a different color is that we consider them as dangerous objects. For example, if the robot collides with a little object of the environment, probably the object will be moved from its initial position without damaging the robot. However, if the robot collides with a wall, the most likely result is that the robot will be damaged. For this reason, the partitions in which walls appear will correspond to a special value in the final structure. This value is equal to 1000 %. This penalty has as a goal to avoid, firstly, the dangerous objects during the path calculation. Finally, the resulting grid structure is shown as a gray-level image in which cells with values equal to or greater than 100 % correspond to the black color, the completely empty cells with the white color, and those cells not fully occupied with a gray-level color (see Fig. 2).

3.2 Path encoding

In this work, we handle paths as sorted lists of grid coordinates. A grid coordinate is a pair (row, column). For a specific path, the first element of a list corresponds with the starting point of the path and, respectively, the last element of the list corresponds with the target point of the path. The number of intermediate coordinates in the list is variable and, in theory, without limit. This representation of the paths implicitly

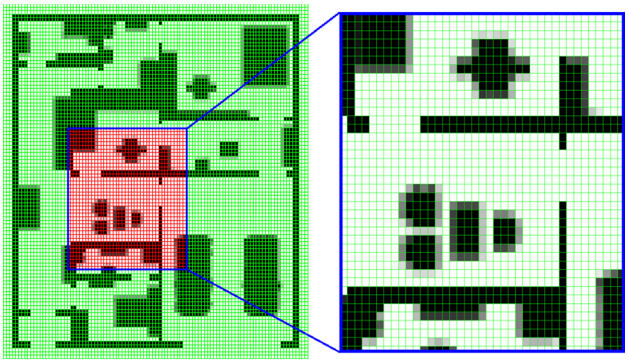

Fig. 2 Example of a final grid structure generated

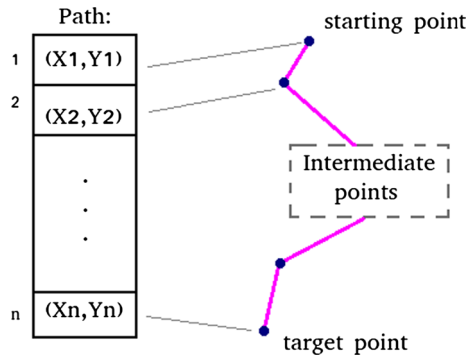

Fig. 3 Example of path encoding

encodes the segments forming the paths; thus, two consecutive coordinates in the list correspond to a segment of the path. Figure 3 shows an example of path encoding.

3.3 PP objectives

The MOEAs in this work, MO-FA and NSGA-II, manage three different objectives to get accurate and efficient solutions. These objectives are the following:

- *Path Length* (related to the robot operation time).
- *Path Safety* (related to avoiding obstacles).
- *Path Smoothness* (related to the energy consumption).

Note that all the objectives presented in this study are to be minimized by the MOEAs. The following subsections explain clearly the calculation of these objectives.

3.3.1 Path length objective

The path length objective aims to get paths as short as possible. Mathematically, it is the sum of the lengths of each segment of a path. The segment length is based on the Euclidean distance between the points (coordinates) forming the segment. The Euclidean distance between two coordinates of the path, $C_1 = (r_1, c_1)$ and $C_2 = (r_2, c_2)$, is calculated using Eq. 1:

$$d(C_1, C_2) = \sqrt{(r_1 - r_2)^2 + (c_1 - c_2)^2}. \quad (1)$$

Remembering that the path is encoded as a sorted list of n grid coordinates, the total length of a specific path can be calculated using Eq. 2:

$$PL = \sum_{i=1}^{n-1} d(L[i], L[i+1]), \quad (2)$$

where L refers to the sorted list of grid coordinates (points) forming the path, and $L[i]$ and $L[i+1]$ refer to the path segment defined by the $i-th$ and $(i+1)-th$ coordinates of the list. Note that this objective is directly related to the robot operation time; thus, a shorter path implies less time to walk the path.

3.3.2 Path safety objective

When the robot moves through space, it is important to do it without colliding with the objects of the environment. This is the aim of the path safety objective. This objective is calculated by adding the occupancy (see Sect. 3.1) of the stepped cells of the grid by the robot along the path. In this work, we consider that the robot has the same size as a cell of the grid representing the environment. Equation 3 shows how to calculate the value of this objective:

$$PS = \sum_{i=1}^{n-1} SCells_{i,i+1}, \quad (3)$$

where $SCells_{i,i+1}$ is the sum of the occupancies of the stepped cells by the robot in the segment defined by the coordinates $i-th$ and $(i+1)-th$ of the sorted list representing the path. The size of the path, in terms of number of coordinates, is determined by n.

3.3.3 Path smoothness objective

The path smoothness objective aims to measure how much snaky is a specific path. This objective is directly related to the robot energy consumption. To calculate the path smoothness for a specific path, we add the angles between each two consecutive segments of the path. Note that the best angle is the closest to 180° (this number defines a straight path). Equation 4 shows how the path smoothness is calculated.

$$P\alpha = \sum_{i=1}^{n_\alpha} (180^\circ - \alpha_i), \quad (4)$$

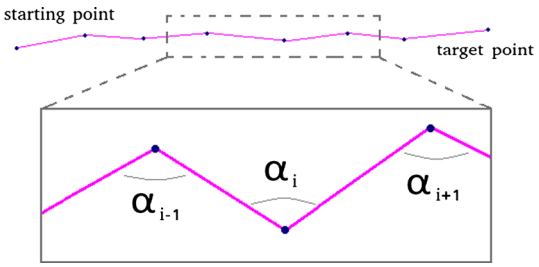

Fig. 4 Path smoothness

where n_α is the number of angles of the path, and α_i refers to the value of the ith angle of the path (measured in degrees in the range from 0° to 180°). Note that the equation has the operation $(180° - \alpha_i)$. The reason for this is that the best angle value is 180° (a straight segment) and the objective is to be minimized by the MOEAs. Figure 4 shows a graphical explanation of the path smoothness.

4 Multi-objective evolutionary algorithms

In the following subsections, the MOEAs used to solve the PP problem (MO-FA and the two versions of NSGA-II) are clearly explained.

4.1 MO-FA

In this work, the algorithm proposed to solve the PP problem is MO-FA. This algorithm is a multi-objective version of the original firefly algorithm (FA). The FA is a swarm intelligence algorithm based on the flashing behavior of fireflies in nature. It was proposed by Yang (2010).

The algorithm reproduces the attraction among the fireflies depending on the amount of light that they emit. The brightest fireflies attract the least bright ones. Every firefly represents a solution to the problem. The brightness of a specific firefly is associated with the quality of its solution (a better solution implies a brighter firefly). Algorithm 1 shows the MO-FA pseudocode.

As we can see in Algorithm 1, our multi-objective version of FA executes two basic steps. Once the initialization step is complete (lines from 1 to 8), the first one is to bring closer the less bright fireflies to the brightest ones (lines from 11 to 27). In this step, all fireflies are compared with the other ones taking into account the epsilon dominance, $\preccurlyeq_\varepsilon$ (line 17). The epsilon dominance allows a firefly to learn also from those others that are a bit worse in terms of flashing. Regarding this, this type of dominance handles an extra value, epsilon, which added to the values of the objective functions determines the fireflies from which to learn. If a firefly epsilon dominates another one, the less bright one will be brought closer to the brightest one (lines 16, 17, and 18). The obtained firefly

Algorithm 1 MO-FA Pseudocode.

```
1: NF ← Number of fireflies.
2: NDSArchive ← ∅.
3: #Initialization.
4: Fireflies ← Generate NF random fireflies.
5: generations ← Number of iterations of the algorithm.
6: Fmoved ← Number of fireflies moved in each generation
   of the algorithm.
7: StagCont ← Percentage of fireflies for stagnation control.
8: LimitF ← Maximum number of times that a firefly tries
   unsuccessfully to evolve.
9:
10: for i = 1 to generations do
11:   #Evolution of the swarm.
12:   Fmoved ← 0
13:   for j = 1 to NF do
14:     Firefly_A ← Fireflies[j]
15:     for k = 1 to NF do
16:       Firefly_B ← Fireflies[k]   #Firefly_B ≠ Firefly_A
17:       if Firefly_A ≼_ε Firefly_B then
18:         Firefly_R ← bringCloser(Firefly_B, Firefly_A)
19:         if Firefly_R ≺ Firefly_B then
20:           Fireflies[k] ← Firefly_R
21:           Fmoved ← Fmoved + 1
22:         else
23:           incrementLimit(Fireflies[k], 1)
24:         end if
25:       end if
26:     end for
27:   end for
28:
29:   #Stagnation control.
30:   if Fmoved < (NF * StagCont)/100 then
31:     for m = 1 to NF do
32:       if Limit(Fireflies[m]) > LimitF then
33:         Fireflies[m] ← new random firefly.
34:       end if
35:     end for
36:   end if
37:
38:   #Save population to the NDSArchive.
39:   exportFireflies(Fireflies, NF, NDSArchive)
40:
41: end for
```

will replace the original firefly only if it is better in terms of dominance (lines 19 and 20). In this step, the number of fireflies moved to the better ones are counted to detect if the population is stagnated (line 21). Furthermore, those stagnated fireflies are identified by increasing their limit value (line 23). The limit value indicates the number of times that a firefly tries unsuccessfully to evolve.

The second step consists in determining if the firefly population is stagnated. Taking into account the number of fireflies that moved to the brightest ones and their limit values, we consider that the population is stagnated when the number of fireflies brought closer to the brightest ones is less than a percentage, *StagCont*. If the population is stagnated, all the fireflies whose limit values are bigger than the *LimitF* value will be replaced by new random fireflies (lines from

29 to 36). Finally, at the end of each generation of the algorithm, the valid individuals of the population are saved to the *NDSArchive* to avoid losing good solutions already achieved (line 39).

Moreover, in this work, we also propose new evolutionary operators. These problem-aware operators are applied along the algorithm procedure with a priority criterion. While the value of the path safety (see Sect. 3.3.2) is greater than a certain threshold (in this case 500 %), the algorithm will apply the path safety operator. When the path safety is less than the threshold value, the algorithm will apply the corresponding operator depending on the maximum value of the path length (see Sect. 3.3.1), the path safety (see Sect. 3.3.2) and the path smoothness (see Sect. 3.3.3). That is, we evaluate which is the objective with the worst value and try to improve it with the corresponding operator. In the following subsections, the proposed evolutionary operators, the mechanisms for improving paths, and the generation of the initial population are clearly explained.

4.1.1 Path length operator

The path length operator aims to reduce the length of a path when it is too long. As we said previously (see Sect. 3.3), the length of a path is directly related to the robot operation time; thus, a shorter path will be walked by the robot in less time. To reduce the path length, this operator deletes a path coordinate (point). Deleting a path coordinate implies that a new segment from the previous to the next coordinate appears (see Fig. 5). In this sense and taking into account MO-FA, to simulate the attraction between two fireflies in terms of the path length, we perform the following procedure. Given two different fireflies (paths), A and B, we determine if A epsilon dominates B. If this is true, then the firefly B (dominated firefly) must be approached by the firefly A (non-dominated firefly). The approximation consists of deleting a firefly B coordinate executing two steps: the first step is to determine the average length of the segments of the firefly A. Once the average length has been determined, two sets of path segments will be created. One of these sets will contain the segments with length greater than the average length calculated previously and the other will contain the segments with length less than the average length calculated before. The second step is to select a random segment and finally deleting one of the coordinates forming the selected segment.

4.1.2 Path safety operator

The path safety operator aims to minimize the number of objects existing in the calculated path, reducing the number of possible collisions. To accomplish this goal, the operator takes as input a segment of the path and then tries to reduce the possible collisions presented in it. In the case of the MO-FA algorithm, and taking into account the path safety, to simulate the attraction between two fireflies we perform two basic steps. Given two fireflies as input, A and B, the first one consists of determining if A epsilon dominates B. If A epsilon dominates B, the firefly B (dominated firefly) must be approximated to the firefly A (non-dominated firefly). The approximation consists of reducing the collision probability of the segment of the firefly B whose path safety value is greater than the average path safety of the segments of the firefly A.

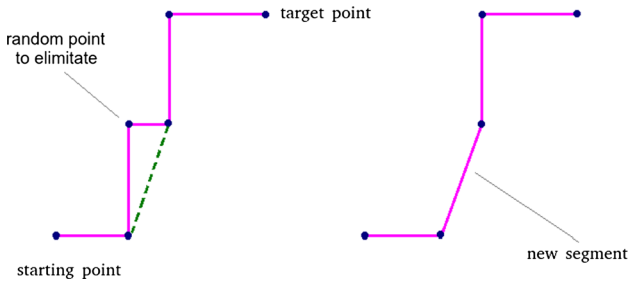

Fig. 5 Evolutionary operator for the path length

To avoid objects rapidly, as first step, the operator tries to determine the shape of the objects. Thus, we classify the objects, in terms of shape, into three categories: vertically shaped objects, horizontally shaped objects, and irregularly shaped objects. The first ones refer to the objects with a completely vertical shape, the second ones with fully horizontal objects, and the last ones with objects which have neither a clear vertical nor horizontal shape. When a segment crosses over a vertically shaped object, the operator determines the collision point and then replaces it with the aim of avoiding the object. In this type of objects, the new candidate point will be searched in the vertical direction of the collision point. Note that, applying this operator, we obtain two new candidate points, one corresponding to the top point and the other corresponding to the bottom point. Finally, to choose the best alternative, we weigh the candidate solutions. To weigh the candidate solutions, the operator takes into account the object shape, the value of the path safety (see Eq. 3) of the new possible segments, and the Euclidean distances from the point to be replaced to the new candidate points. Thus, if the object is irregularly shaped, the vertical and horizontal weights will have different values. Regarding the possible candidate segments, they will have a greater weight value when they do not cross any object. Furthermore, the candidate point with lowest Euclidean distance from the collision point will have a greater weight value. Finally, the candidate point for replacing the collision point will be the one with the greater global weight value. Figure 6 shows a graphical explanation of the path safety operator applied to vertically shaped objects.

In Fig. 6, pm is the point to be modified (the collision point), $cp1$ (candidate point 1) and $cp2$ (candidate point 2)

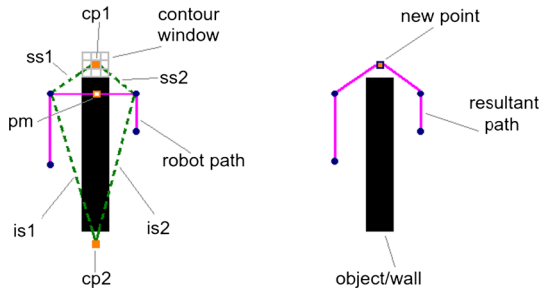

Fig. 6 Path safety operator. Vertically shaped object

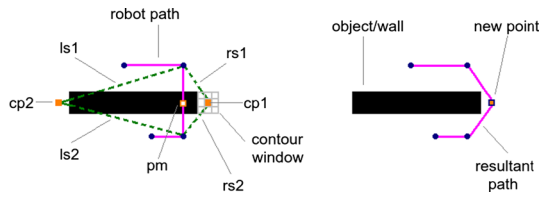

Fig. 7 Path safety operator. Horizontally shaped object

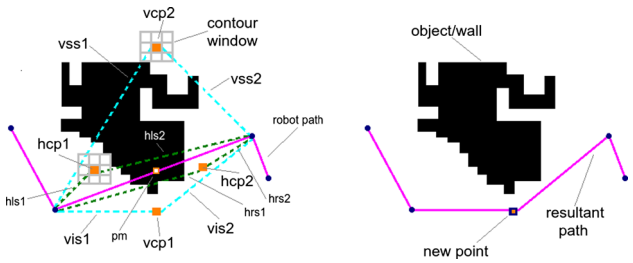

Fig. 8 Path safety operator. Irregularly shaped object

refer to the candidate points determined to avoid the object, and $ss1$ (superior segment 1), $ss2$ (superior segment 2), $is1$ (inferior segment 1), and $is2$ (inferior segment 2) correspond to the possible candidate segments that could appear when the collision point will be replaced with one of the candidate points.

In the case of horizontally shaped objects, the procedure is basically the same, but with the difference that the candidate points for replacing the collision point will be searched in the horizontal direction of the collision point. Figure 7 shows an example of the *path safety operator* applied to a horizontally shaped object.

In Fig. 7, pm refers to the point to be modified (the collision point), $cp1$ (candidate point 1) and $cp2$ (candidate point 2) refer to the candidate points determined to avoid the object, and $ls1$ (left segment 1), $ls2$ (left segment 2), $rs1$ (right segment 1), and $rs2$ (right segment 2) are the candidate segments that could appear by replacing the collision point with one of the candidate points.

Finally, when the object is irregularly shaped, the operator behaves as a mixture of the cases in which the object is horizontally shaped or vertically shaped. Figure 8 shows a graphical explanation of this operator applied to an irregularly shaped object.

Regarding Fig. 8, at first glance the object does not have a well-defined shape, so we need to handle four candidate points, two corresponding to the vertical candidates and the other ones corresponding to the horizontal candidates. In this figure, pm is again the point to be modified (the collision point). The parameters $hcp1$ (horizontal candidate point 1), $hcp2$ (horizontal candidate point 2), $hls1$ (horizontal left segment 1), $hls2$ (horizontal left segment 2), $hrs1$ (horizontal right segment 1), and $hrs2$ (horizontal right segment 2) are handled when the possible solutions are searched horizontally. Conversely, parameters $vcp1$ (vertical candidate point 1), $vcp2$ (vertical candidate point 2), $vss1$ (vertical superior segment 1), $vss2$ (vertical superior segment 2), $vis1$ (vertical inferior segment 1), and $vis2$ (vertical inferior segment 2) are handled when the possible solutions are searched vertically.

Note that, in all the cases, a contour window is used to find the new candidate points far from objects. The contour window identifies as candidate points those which are far enough from the dangerous object, reducing thus the possibility that the candidate segments collide with the object to avoid.

4.1.3 Path smoothness operator

The path smoothness operator tries to obtain a path as straight as possible. To reduce the turns magnitude of the path, the operator takes as input two consecutive segments of the path and then tries to improve the angle between them. The angle is better when it is close to 180° and, conversely, is worse when it is close to 0°. In the case of MO-FA, and to simulate an approximation in terms of smoothness, the operator takes as input two different fireflies, A and B. The first step is to determine if the firefly A epsilon dominates the firefly B. If this is true, the second step consists of improving the angle of the firefly B (dominated firefly) whose value is less than the average angle value of the firefly A (non-dominated firefly). To improve the selected angle, the operator changes the point of the path that matches with the vertex of the angle by the middle point of the next or previous segment of the path. The selection between the next and the previous segment is random. Accordingly, the segments forming the angle will be modified too. This operation produces a better angle in the path, reducing thus the turn magnitude. Remember that the smoothness is an important parameter in the PP problem because it is directly related to energy consumption. See Fig. 9 for a graphical example of this PP operator.

Note that the operator will modify both segments at the input and the point corresponding with the angle vertex.

4.1.4 Generation of the initial population

In EAs, the first step always consists of generating the initial population. This step produces a certain number of possible solutions to make them evolve along the algorithm procedure.

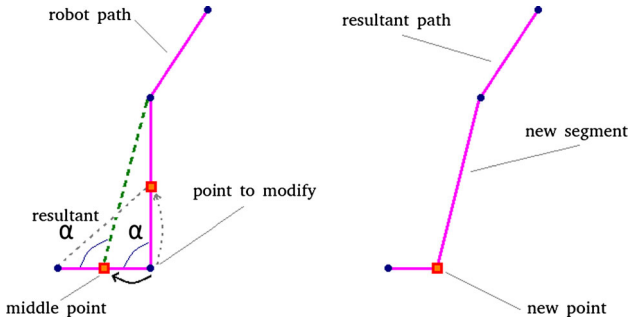

Fig. 9 Path smoothness operator

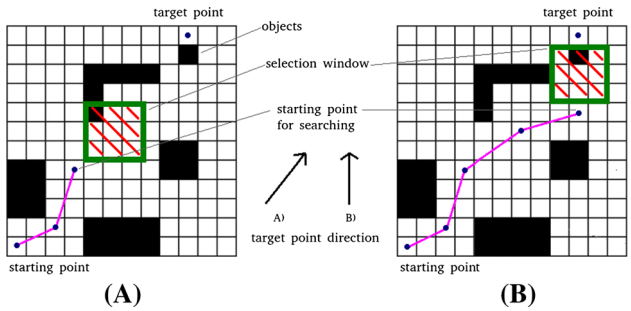

Fig. 10 Population initialization. Selection window example

In this work, we developed a selection window to generate new population individuals. The selection window selects the new valid candidate points to add to the path in the surrounding area of the last point added to the path. Among all these valid candidate points, one of them will be randomly selected. A point is valid if it can be linked safely (without crossing objects) with the last point added to the path. If valid candidate points do not exist, a random point is selected. By using the selection window, we try to generate paths with less segments crossing objects. On the other hand, this window is used to search the new candidate points in the target point direction, trying thus to obtain paths as short as possible. To select the new points in the target point direction, we place the selection window in different places of the map, applying this to the area in which the new candidates will be searched. For example, if the starting point of the search is in the upper-left corner of the map and the target point is in the lower-right corner of the map, the upper-left corner of the selection window will match with the starting point of the search. Figure 10 shows an example of the selection window behavior.

As we said previously (see Sect. 3.2), the individuals obtained in the initialization step are of variable size. In this work, the generation of an individual ends when the last point added to the path can be safely linked with the target point of the path.

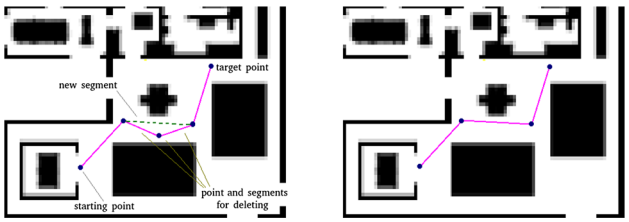

Fig. 11 Example of the mechanism for improving length

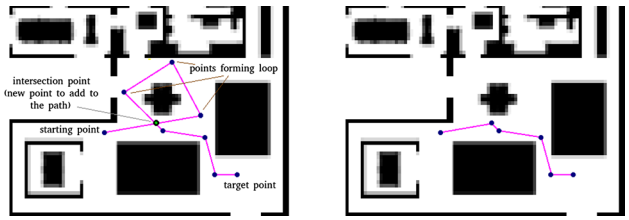

Fig. 12 Deleting loops example

4.1.5 Mechanism for improving length

After the population initialization step, usually, the items correspond to non-optimized individuals. In this sense, they are usually composed of extra segments and loops. This mechanism aims to delete the extra segments presented in the individuals of the initial population. To delete these segments, this mechanism, for each coordinate, tries to find the farthest point of the path with which it can be linked safely; that is to say, a collision-free link. The search starts from the target point and finishes when the starting point will be reached. Figure 11 shows a graphical explanation of this mechanism. Note that this operation deletes points and, consequently, segments from the original path at the input. Furthermore, this improving step is only used after the population initialization. At the end of the process, the algorithm starts running with a more or less good population at the input.

4.1.6 Mechanism for deleting loops

This mechanism aims to delete the loops presented in the initial population. To delete these useless segments of a specific path, the mechanism tries to determine if two different segments cross themselves. This indicates that there is a loop in the path. To suppress the loop, the mechanism deletes all the coordinates and segments forming the loop and then adding to the path a new point, corresponding to the intersection point of the loop. The process is complete when there are no segments crossing themselves. At the end of the process, the algorithm starts running with a free-loop population. Note that this mechanism is only used after the population initialization step. Figure 12 shows a graphical explanation of loops erasing.

Algorithm 2 NSGA-II Pseudocode.

```
1: populationSize ← Number of individuals.
2: NDSArchive ← ∅.
3: #Initialization.
4: population ← initializatePopulation(populationSize)
5: generations ← Number of iterations of the algorithm.
6: for i = 1 to generations do
7:
8:   #Crossover.
9:   offspring ← crossPopulation(population)
10:  #Mutation.
11:  offspring ← mutatePopulation(offspring)
12:  population = population + offspring
13:  population ← fastNonDominatedSort(population,
     (populationSize ∗ 2))
14:  population ← crowdingDistanceAssignment
     (population, (populationSize ∗ 2))
15:  #Replacement.
16:  population ← selectBetterIndividuals(population,
     populationSize)
17:  #Save population to the NDSArchive.
18:  exportPopulation(population, populationSize,
     NDSArchive)
19:
20: end for
```

4.2 NSGA-II from Jun and Qingbao (2010): optimizing the 3 objectives

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a well-known multi-objective metaheuristics proposed by Deb et al. (see Deb et al. 2002). This algorithm has been used in Jun and Qingbao (2010) for solving the path planning problem with the same three objectives as we handle in our work: the path length, the path safety, and the path smoothness. For this reason, in Sect. 6, we include comparisons with this algorithm, and in this Section we present a brief explanation of the algorithm. The NSGA-II algorithm tries to improve an initial solution set (population) by applying classical genetic operators: selection, crossover, mutation, and replacement. Algorithm 2 shows the NSGA-II pseudocode.

The NSGA-II algorithm (Algorithm 2) begins with the population initialization (line 4). The initialization consists of generating a random set of individuals (paths). Once the initialization is completed, the algorithm tries to evolve the population by executing four basic steps. The first step has as a goal to cross over the initial population to obtain the offspring population (line 9). The crossover uses the binary tournament to select the parents of each generated offspring. The second step consists of a mutation process. The mutation step is performed for each new obtained individual (line 11). Note that, for both, the crossover step and the mutation step, two different probabilities are used with the aim of determining when the operation should be performed. Before executing the last step, a new population formed by the initial individuals and the obtained offspring is generated, sorted, and evaluated by using the crowding distance (lines 12 – 14). Finally, in the fourth step, the best individuals are considered to continue evolving in the process (line 16). For a more detailed explanation of the algorithm, see Jun and Qingbao (2010).

Note that, in this case, the evolutionary operators for mutations, the population initialization and the mechanisms for improving individuals are the same as in the MO-FA algorithm (see Sects. 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.1.5, and 4.1.6).

4.3 NSGA-II from Ahmed and Deb (2013): Optimizing two objectives and using the smoothness as a decision maker

This algorithm has been used in Ahmed and Deb (2013) for solving the PP problem. Although these authors used the same three objectives as we manage in this work, the main difference between this algorithm and the one presented in Sect. 4.2 is that they used the *path smoothness* as a decision maker instead of an objective to be optimized (that is, there are only two objectives to optimize: the path length and the path safety). To introduce the path smoothness as a criterion to prefer smooth paths, they modify the NSGA-II's selection scheme (see Sect. 4.2). This means that, in the binary tournaments or in choosing the final front members, when two different paths have the same rank, they check the smoothness instead of using the crowding distance. The path with better smoothness (the one with the lower value) will be selected. On the other hand, the crowding distance is only taken into account when two different paths (solutions) have the same non-domination rank and smoothness values. In this case, the one with the higher crowding distance will be selected. In Sect. 6, we also include comparisons with this algorithm of the state of the art. For a more detailed explanation of the algorithm see Ahmed and Deb (2013).

Note that, to do a fair comparison among the different algorithms, the evolutionary operators for mutations, the population initialization, and the mechanisms for improving individuals are the same as in the MO-FA algorithm (see Sects. 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.1.5, 4.1.6).

5 Methodology

In this section and subsections, we present the proposed scenarios to make the study (Sect. 5.1), explain clearly the MOEAs configuration (Sect. 5.2), the metrics used to evaluate the obtained results (Sect. 5.3), and the software and hardware specifications for the tests.

5.1 Scenarios

To evaluate the MOEAs presented in Sect. 4, an input data set is required. Each data set is known as a scenario. A scenario is

formed by a map, the starting point, and the target point of the path we want to calculate. In this work, we use eight pairs of points, distributed between two different realistic maps of the environment, that is to say, eight different scenarios. Figure 13 shows the proposed maps.

Taking into account the map coordinate system (see Fig. 1), the starting and the target points of the paths have been established in the main directions of each map (paths direction). These directions are the main diagonal (**MD**), the secondary diagonal (**SD**), the horizontal axis (**HA**), and the vertical axis (**VA**). Tables 1 and 2 show the proposed scenarios.

Note that the datasets used in this study can be downloaded from http://arco.unex.es/mavega/pathplanning.html.

5.2 MOEAs configuration

Different methodologies can be used for tuning the parameters of evolutionary algorithms. Although some of these methodologies are automatic (such as irace López-Ibáñez et al. (2011)), many researchers in path planning use manual methodologies (Ahmed and Deb 2013; Jun and Qingbao 2010), as in this work. In particular, in our case, to adjust the parameter configuration of the MOEAs (see Sect. 4), numerous experiments were performed. First, we perform several runs to observe which are the most influential parameters. Thus, the most influential parameter will be the first to be set up, and so on. In each experiment, we performed 31 independent runs, with this number of executions being sufficient to ensure the statistical significance of the obtained results. To select the tested values for each parameter, we examined its range of possible values, establishing a minimum of four possible uniformly distributed values. If in any case two distinct values present similar results, we establish a new intermediate value and repeat the process. The parameter value that finally achieves the best results [using the hypervolume (Zitzler and Thiele 1999) as quality metric, because the hypervolume is regarded as a rather fair measure since it has favorable theoretical properties (Zitzler et al. 2003), giving it an outstanding importance among quality indicators (Beume and Fonseca 2009)] will be established. This process is repeated for all the parameters and all the algorithms. Tables 3 and 4 show the tested values for each parameter of each algorithm [MO-FA and NSGA-II from Ahmed and Deb (2013) and Jun and Qingbao (2010), respectively] and the selected values taking into account the configuration process.

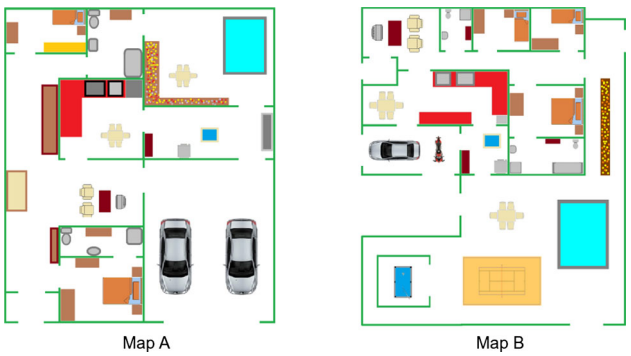

Fig. 13 Realistic maps proposed for the study

Table 1 Scenarios (Map A)

Config.Scenario	A	B	C	D
Starting point	(93,6)	(93,93)	(31,8)	(83,32)
Target point	(23,75)	(6,11)	(35,89)	(11,26)
Path direction	SD	MD	HA	VA
Map	A	A	A	A

Table 2 Scenarios (Map B)

Config.Scenario	E	F	G	H
Starting point	(94,6)	(94,90)	(28,17)	(93,48)
Target point	(9,90)	(6,6)	(46,91)	(6,49)
Path direction	SD	MD	HA	VA
Map	B	B	B	B

5.3 Quality metrics and statistical analysis

To select the best MOEA applied to the PP, different quality metrics were used. The quality metrics are the hypervolume (HV) (Zitzler and Thiele 1999) and the set coverage (SC) measurement (Zitzler et al. 2000).

Regarding the quality metrics, the HV measures the volume (in the objective function space) covered by solutions of a non-dominated set of solutions. In a multi-objective optimization problem with d objective functions to minimize, we calculate the size of the region of the objective space (HV) of a non-dominated set of solutions $A = \{a_1, a_2, \ldots, a_n\}$ bounded by a reference point $r = (r_1, r_2, \ldots, r_d)$. The corresponding hypercube for each element a_i of the set A is calculated as follows: $h(a_i) = [a_{i1}, r_1]\ x\ [a_{i2}, r_2]\ x\ \ldots\ x\ [a_{id}, r_d]$. Finally, the HV of A is calculated by the union of these $|A|$ hypercubes, with repeatedly covered hypercubes being counted once. Equation 5 shows how to calculate the HV:

$$HV(A, r) = L\left(\bigcup_{i=1}^{|A|} h(a_i) \mid a_i \in A\right). \quad (5)$$

In Eq. 5, L refers to the Lebesgue measure Bartle (2011).

On the other hand, the set coverage is an indicator that measures the fraction of non-dominated solutions in a Pareto

Table 3 MO-FA configuration parameters

Parameter	Tested values	Selected value
MO-FA Configuration parameters		
Population size	{50, 100, 150, 200, 250, and 300}	200
Generations	{100, 120, 150, 200, 300, and 600}	150
Epsilon (Dominance)	{0.025, 0.050, 0.075, 0.1, and 0.125}	0.05
StagCont %	{5, 10, 15, 20, 40, 60, and 80}	20
LimitF	{1, 5, 10, 15, 20, 25, 30, and 35}	5

Table 4 NSGA-II from Ahmed and Deb (2013) and Jun and Qingbao (2010) configuration parameters

Parameter	Tested values	Selected value
NSGA-II from Ahmed and Deb (2013) and Jun and Qingbao (2010) Configuration parameters		
Population size	{50, 100, 150, 200, 250, and 300}	200
Generations	{100, 120, 150, 200, 300, and 600}	150
Mutation probability %	{5, 10, 15, 20, and 25}	25
Crossover probability %	{70, 80, 90, and 100}	90

front $B = \{b_1, b_2, \ldots, b_m\}$, which are covered by the non-dominated solutions in $A = \{a_1, a_2, \ldots, a_n\}$ (see Eq. 6).

$$\mathrm{SC}(A, B) = \frac{|\{b \in B; \exists a \in A : a \preccurlyeq b\}|}{|B|}. \tag{6}$$

If $\mathrm{SC}(A, B) = 1$, all the solutions in B are covered by solutions in A. Conversely, if $\mathrm{SC}(A, B) = 0$ it means that none of the solutions in B is covered by the solutions in A. Note that, as the dominance operator ($\preccurlyeq$) is not symmetric, it is necessary to calculate both $\mathrm{SC}(A, B)$ and $\mathrm{SC}(B, A)$, since $\mathrm{SC}(B, A)$ is not necessarily equal to $1 - \mathrm{SC}(A, B)$. SC has been selected because it is a binary metric (in contrast to HV, which is a unary metric), and in its calculation, it is not necessary to know the optimal Pareto front. These two complementary metrics (HV and SC) were presented by Zitzler and Thiele (1998) and Zitzler and Thiele (1999). The HV combines the three criteria (convergence, distribution, and extent of the obtained non-dominated front) (Zitzler et al. 2000). On the other hand, the set coverage can be used to show that the outcomes of an algorithm dominate the outcomes of another algorithm (Zitzler et al. 2000).

Once the quality metrics have been defined, the statistical analysis is explained. In every experiment, we performed 31 independent runs. The statistical analysis aims to demonstrate formally that the differences between the results obtained by applying the previous quality tests are significant. Figure 14 shows the statistical analysis scheme.

As we can see in the statistical analysis scheme (see Fig. 14), the first step is to determine if the input data set follows a Gaussian distribution. To carry out this check, we use the Kolmogorov–Smirnov test. For non-Gaussian distribution, we perform a non-parametric analysis, in this case

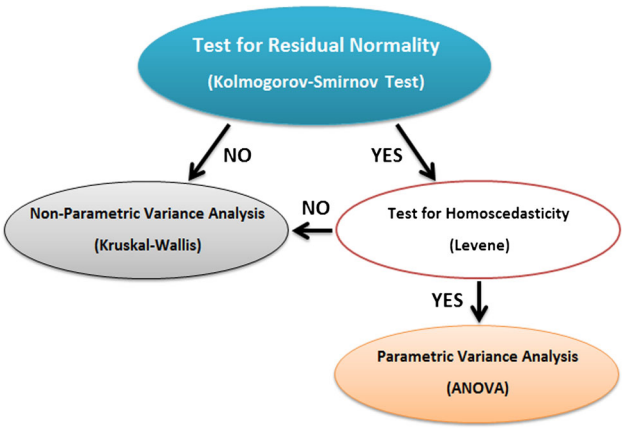

Fig. 14 Statistical analysis scheme

the Kruskal–Wallis test. However, if the results of the test assert that the input data follows a Gaussian distribution, we will need to check the homogeneity of the variances. This is the second step. To check the homogeneity of the variances, the Levene test (test for homoscedasticity) is used. If the test result is negative, we perform the Kruskal–Wallis test. Finally, if the test result is positive, the ANOVA analysis (parametric variance analysis) will be performed. A more detailed explanation of these tests can be found in Sheskin (2011).

Note that, in this paper, the confidence level in the statistical test is always 95 % (a significance level of 5 % or p-value less than 0.05). This means that the differences are unlikely to have occurred by chance with a probability of 95 %.

5.4 Software and hardware specifications

In this study, all the MOEAs have been developed using the *C/C++* programming language. The MOEAs presented in

Table 5 Hardware and software specifications

Parameter	Value
CPU	AMD Opteron 6176 / 2.3 GHz
CPU released	Q4 2010
Memory	DDR3 1333 MHz, 16 GB
Cache	12 MB L3
Storage	500 GB
O.S.	Ubuntu 12.10
Compiler	GCC 4.4.5

this study have been developed from scratch because the reference authors do not provide their source codes. It is important to remark that for the experiments, all the MOEAs execute the same number of fitness function evaluations. This number is equal to 30,000. On the other hand, Table 5 shows the hardware and software specifications in which all the experiments have been executed.

6 Results

In this section, we present and analyze the results obtained by the different MOEAs over eight different scenarios (see Sect. 5.1). Note that, in this study, the experiments were made by partitioning the scenarios into a grid with a size of 100 rows by 100 columns. Table 6 shows the median hypervolume, the interquartile range of the hypervolume values, and the results obtained by applying the statistical tests. Furthermore, to show the results graphically, Fig. 15 shows the hypervolume box plots for the different proposals.

In Table 6, the first column refers to the scenario used for the tests. The second and the third columns indicate the results obtained by the corresponding algorithm to calculate the paths specified by the different scenarios. Finally, the fourth column, SSD (Statistically Significant Differences), is the result of the statistical analysis (see Sect. 5.3). In this case, the statistical analysis indicates that the results obtained by both algorithms (MO-FA vs. NSGA-II from Jun and Qingbao (2010), or MO-FA vs. NSGA-II from Ahmed and Deb (2013)) are significantly different in all the scenarios. Note that cells corresponding to the second and the third columns have the format X_Y, where the X value refers to the median hypervolume and Y to the interquartile range of hypervolume.

As we can see, for all the scenarios, the results obtained by *MO-FA* are always better than when NSGA-II from Jun and Qingbao (2010) is used. These hypervolume values indicate that the MO-FA algorithm covers a greater space of the solution area. In the same way, almost all the results obtained by MO-FA are better than when NSGA-II from Ahmed and Deb (2013) is used.

Table 6 Median and interquartile range of the hypervolume

	NSGA-II from Jun and Qingbao (2010)	MO-FA	SSD	NSGA-II from Ahmed and Deb (2013)	MO-FA	SSD
Scenario A	$0.599_{0.082}$	$0.713_{0.021}$	✓	$0.564_{0.095}$	$0.713_{0.021}$	✓
Scenario B	$0.771_{0.042}$	$0.793_{0.003}$	✓	$0.773_{0.045}$	$0.793_{0.003}$	✓
Scenario C	$0.754_{0.014}$	$0.778_{0.008}$	✓	$0.754_{0.014}$	$0.778_{0.008}$	✓
Scenario D	$0.779_{0.020}$	$0.807_{0.005}$	✓	$0.778_{0.022}$	$0.807_{0.005}$	✓
Scenario E	$0.862_{0.011}$	$0.869_{0.001}$	✓	$0.877_{0.012}$	$0.869_{0.001}$	✓
Scenario F	$0.702_{0.061}$	$0.803_{0.012}$	✓	$0.718_{0.069}$	$0.803_{0.012}$	✓
Scenario G	$0.627_{0.135}$	$0.821_{0.005}$	✓	$0.660_{0.176}$	$0.821_{0.005}$	✓
Scenario H	$0.544_{0.132}$	$0.716_{0.009}$	✓	$0.554_{0.151}$	$0.716_{0.009}$	✓

NSGA-II from Jun and Qingbao (2010) & NSGA-II from Ahmed and Deb (2013) & MO-FA

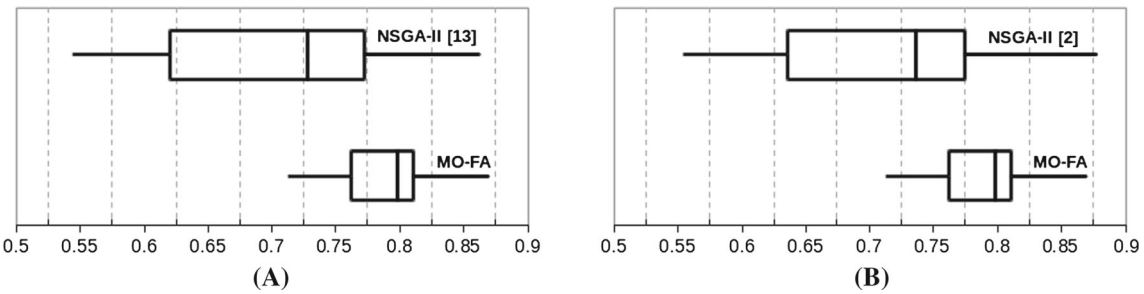

Fig. 15 Hypervolume boxplots: **a** NSGA-II from Ahmed and Deb (2013) & MO-FA, and **b** NSGA-II from Jun and Qingbao (2010) & MO-FA

Table 7 Set coverage results (MO-FA, NSGA-II from Jun and Qingbao (2010)) & (MO-FA,NSGA-II from Ahmed and Deb (2013))

SC.Scenario	A	B	C	D	E	F	G	H
SC [NSGA-II from Jun and Qingbao (2010), MO-FA]	0.064	0.068	0.177	0.047	0.098	0.069	0.014	0.017
SC [MO-FA, NSGA-II from Jun and Qingbao (2010)]	0.737	0.642	0.659	0.893	0.695	0.697	0.964	0.850
SC [NSGA-II from Ahmed and Deb (2013), MO-FA]	0.064	0.068	0.177	0.047	0.098	0.069	0.014	0.017
SC [MO-FA, NSGA-II from Ahmed and Deb (2013)]	0.737	0.642	0.659	0.893	0.695	0.697	0.964	0.850

Table 8 Scenarios reference points (path safety, path length, path smoothness)

	Ideal	Nadir
Scenario A	(0, 98.290, 0)	(8541, 288.363, 1352)
Scenario B	(0, 119.553, 0)	(11677, 333.122, 1227)
Scenario C	(0, 81.098, 0)	(12290, 285.299, 1271)
Scenario D	(0, 72.249, 0)	(12841, 273.247, 1347)
Scenario E	(0, 119.503, 0)	(10306, 376.517, 1533)
Scenario F	(0, 121.655, 0)	(7630, 325.508, 2102)
Scenario G	(0, 76.157, 0)	(10049, 339.670, 2222)
Scenario H	(0, 87.005, 0)	(13772, 344.052, 1829)

Regarding the associated interquartile range values, we can assert that the results obtained by applying the MO-FA algorithm are always less sparse than when we apply the NSGA-II algorithm in any of its two versions (Jun and Qingbao 2010 or Ahmed and Deb 2013). For this reason we can assert that the MO-FA algorithm is more reliable than NSGA-II in terms of results.

To make a more reliable assertion about the best alternative to solve the PP problem, we also used the set coverage metric. Table 7 shows the SC results.

Set coverage (SC) results (Table 7) confirm that MO-FA is always better than NSGA-II (any of its two versions, Jun and Qingbao 2010 or Ahmed and Deb 2013). For this reason, we can state that the MO-FA algorithm is a better alternative versus the NSGA-II algorithm to solve the PP problem.

Note that, in this study, two reference points were used for each scenario to compute the HV and test the MOEAs. On the one hand, an ideal reference point (the best values of

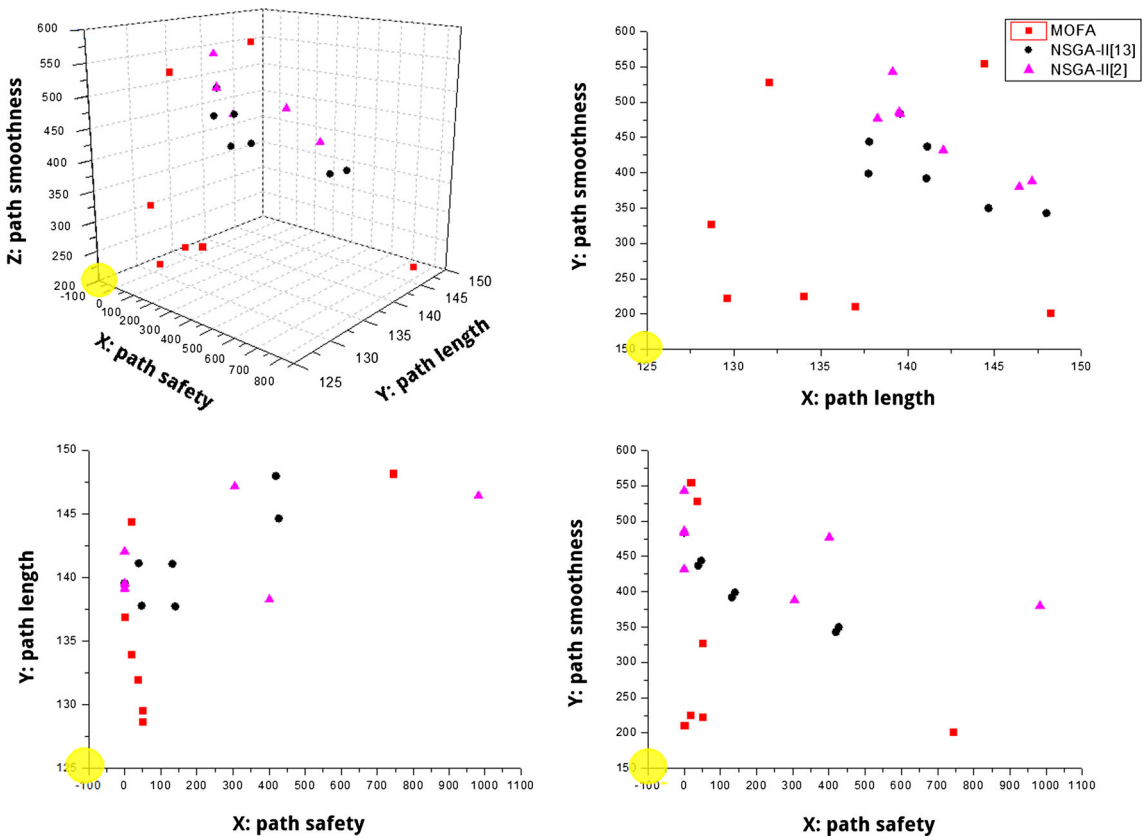

Fig. 16 Approximate Pareto fronts for scenario A

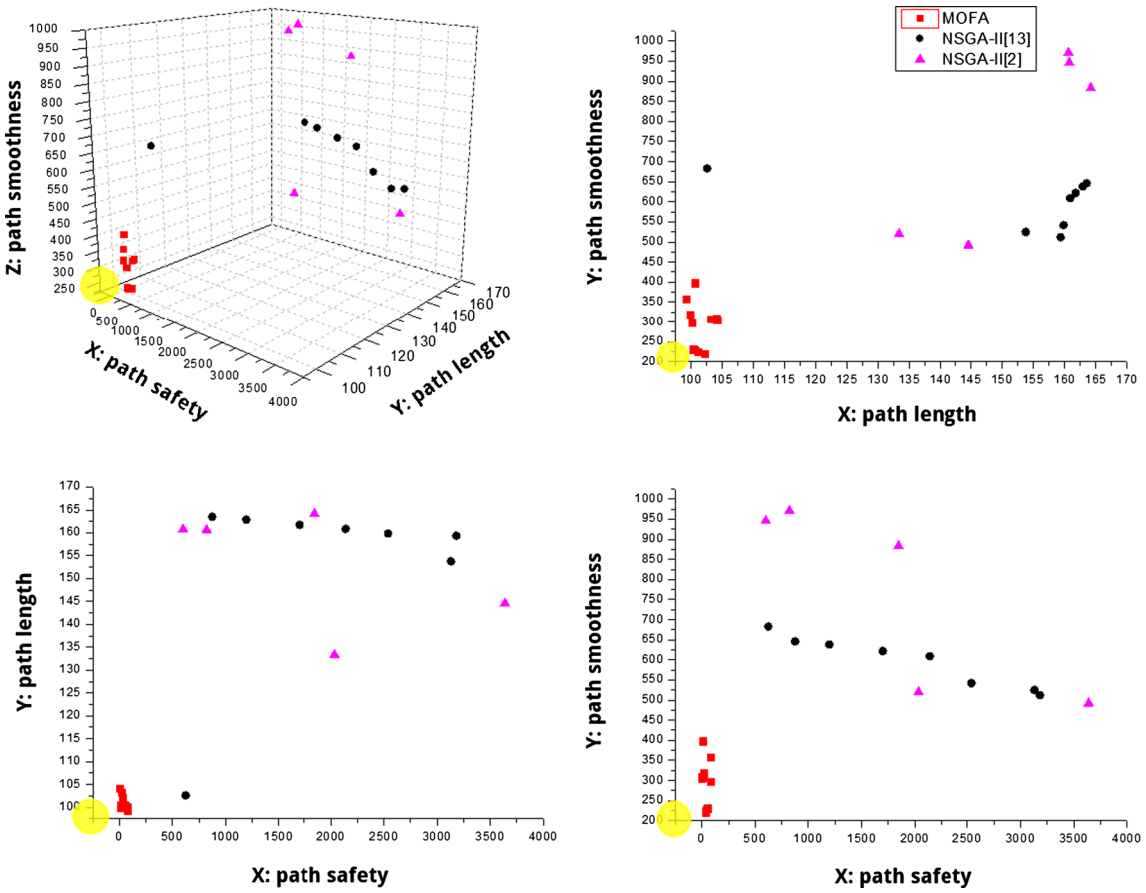

Fig. 17 Approximate Pareto fronts for scenario G

the problem objectives) is calculated as follows. For the path safety, the ideal value is obtained when a specific path does not cross any of the objects appearing in the environment, that is to say, a value equal to 0. In the case of the path length, the ideal value is obtained when the starting and the target points of the path can be linked by a single segment. For this reason, the best value for this objective is the Euclidean distance from the starting point to the target point of the path. Taking into account that the best hypothetical path is formed by a single segment, the ideal value for the path smoothness is 0. This is due to the fact that in a path formed by a single segment, no turns exist. On the other hand, a nadir reference point is formed by the worst values of the considered objectives. These values are calculated by adding an offset equal to 30 % to the worst values of the objectives, which were obtained as result of the configuration tests (see Sect. 5.2). Table 8 shows the reference points considered for each scenario.

To view the results graphically, Figs. 16 and 17 show the approximate *Pareto front* corresponding to the results obtained by applying the MOEAs over two different scenarios, in this case scenarios *A* and *G*. Please observe that we have selected a scenario for each map (see Fig. 13 and Tables 1 and 2). The Pareto fronts represented are those that

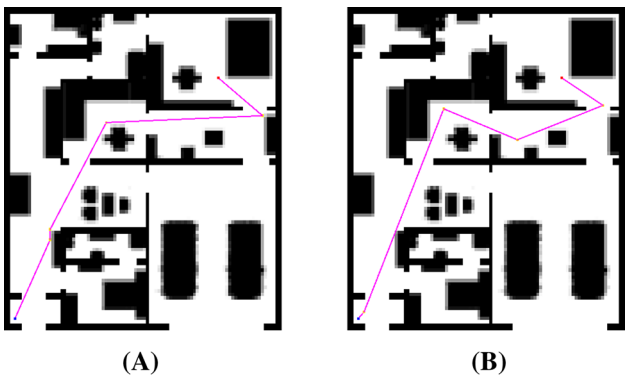

Fig. 18 Graphical solution for the scenario A: **a** MO-FA, and **b** NSGA-II from Jun and Qingbao (2010)

obtain the median hypervolume for that algorithm in each scenario.

In all graphics (Figs. 16 and 17), points representing non-dominated solutions of the MO-FA algorithm (red squares) clearly tend to approach the ideal point (represented as a yellow translucent circle), while the points belonging to NSGA-II (any of its two versions, from Jun and Qingbao (2010) or Ahmed and Deb (2013)) remain farther from the

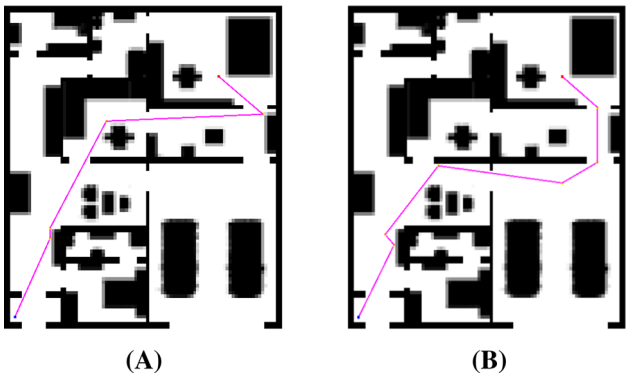

Fig. 19 Graphical solution for the scenario A: **a** MO-FA, and **b** NSGA-II from Ahmed and Deb (2013)

Fig. 20 Graphical solution for the scenario G: **a** MO-FA, and **b** NSGA-II from Jun and Qingbao (2010)

ideal solution. This again demonstrates that MO-FA is a better alternative versus NSGA-II to solve the PP problem.

Finally, Figs. 18, 19, 20, and 21 show the best paths corresponding to the approximate Pareto fronts shown in Figs. 16 and 17, respectively. These paths represent the solution closer to the ideal reference point in each case (Euclidean distance is used for this). Again, these images show the advantages of using MO-FA versus NSGA-II when they are applied to the PP problem.

7 Conclusions and future work

In this work, the use of a multi-objective approach based on the flashing behavior of fireflies in nature (MO-FA) has been proposed to solve the path planning problem. In this study, our approach handles three different objectives: the path safety, the path length, and the path smoothness (see Sect. 3.3). Furthermore, to obtain accurate and effective paths (solutions), we proposed new evolutionary operators too (see Sect. 4.1). Moreover, with the aim of comparing with the state of the art, we also used the well-known NSGA-II algorithm (specifically, the approaches proposed by Jun and Qingbao (2010) and Ahmed and Deb (2013)), which has been the most commonly used algorithm by other authors in the literature. Regarding the obtained results, they have been evaluated by using both, multi-objective metrics HV and SC, to ensure the quality of the obtained results, and a statistical analysis to demonstrate the statistical relevance of the results. The comparison showed differences in the approximate Pareto fronts and the solutions provided by each MOEA; that is, MO-FA and NSGA-II (from Jun and Qingbao (2010) and Ahmed and Deb (2013)) (see Figs. 16, 17, 18, 19, 20, and 21 in Sect. 6). To test the MOEAs, and unlike other authors, we used eight different realistic scenarios. Taking into account the results presented in Sect. 6, we can assert that MO-FA is clearly a better alternative versus NSGA-II when they are applied to the PP problem.

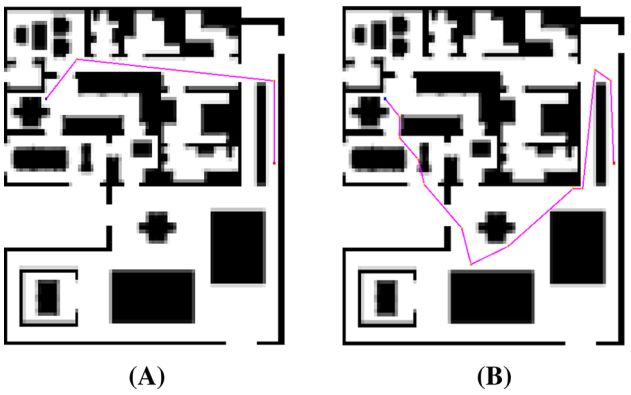

Fig. 21 Graphical solution for the scenario G: **a** MO-FA, and **b** NSGA-II from Ahmed and Deb (2013)

As future work, and taking into account the good results obtained by MO-FA, it could be interesting to compare with other multi-objective algorithms based on swarm intelligence. On the other hand, another possible future work consists of calculating paths in dynamic environments, since this work only considers static environments.

Acknowledgments This work was partially funded by the Projects of Excellence from the Junta de Andalucía (Spain) ROMOCOG I and ROMOCOG II (P09-TEP-4479 and P10-TEP-6412). The work was also partially funded by the Spanish Ministry of Economy and Competitiveness and the ERDF (European Regional Development Fund), under the contract TIN2012-30685 (BIO project).

References

Ahmed F, Deb K (2011) Multi-objective path planning using spline representation. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics (IEEE-ROBIO 2011), pp. 1047–1052. doi:10.1109/ROBIO.2011.6181426

Ahmed F, Deb K (2013) Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. Soft Comput 17(7):1283–1299. doi:10.1007/s00500-012-0964-8

Bartle R (2011) The Elements of Integration and Lebesgue Measure. Wiley, Wiley Classics Library

Beume N, Fonseca C, López-Ibáñez M, Paquete L, Vahrenhold J (2009) On the complexity of computing the hypervolume indicator. Evol Comput IEEE Trans 13(5):1075–1082. doi:10.1109/TEVC.2009.2015575

Chang, H.C., Liu, J.S.: High-quality path planning for autonomous mobile robots with n3-splines and parallel genetic algorithms. In: Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on, pp. 1671–1677 (2009). doi:10.1109/ROBIO.2009.4913252

Davoodi M, Panahi F, Mohades A, Hashemi SN (2013) Multi-objective path planning in discrete space. Appl Soft Comput 13(1):709–720. doi:10.1016/j.asoc.2012.07.023

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. Evol Comput, IEEE Trans 6(2):182–197. doi:10.1109/4235.996017

Geetha, S., Chitra, G., Jayalakshmi, V.: Multi objective mobile robot path planning based on hybrid algorithm. In: Electronics Computer Technology (ICECT), 2011 3rd International Conference on, vol. 6, pp. 251–255 (2011). doi:10.1109/ICECTECH.2011.5942092

Geng, N., Gong, D., Zhang, Y.: Robot path planning in an environment with many terrains based on interval multi-objective PSO. In: Evolutionary Computation (CEC), 2013 IEEE Congress on, pp. 813–820 (2013). doi:10.1109/CEC.2013.6557652

Gong DW, Zhang JH, Zhang Y (2011) Multi-objective particle swarm optimization for robot path planning in environment with danger sources. J Comput 6(8):1554–1561. doi:10.4304/jcp.6.8.1554-1561

Guo, F., Wang, H., Tian, Y.: Multi-objective path planning for unrestricted mobile. In: Automation and Logistics, 2009. ICAL '09. IEEE International Conference on, pp. 1046–1051 (2009). doi:10.1109/ICAL.2009.5262574

Hao, W., Qin, S.: Multi-objective Path Planning for Space Exploration Robot Based on Chaos Immune Particle Swarm Optimization Algorithm. In: H. Deng, D. Miao, J. Lei, F. Wang (eds.) Artificial Intelligence and Computational Intelligence, Lecture Notes in Computer Science, vol. 7003, pp. 42–52. Springer, Berlin Heidelberg (2011). doi:10.1007/978-3-642-23887-1_6

Jun, H., Qingbao, Z.: Multi-objective mobile robot path planning based on improved genetic algorithm. In: Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on, vol. 2, pp. 752–756 (2010). doi:10.1109/ICICTA.2010.300

Kim, Y.H., Kim, J.H.: Multiobjective quantum-inspired evolutionary algorithm for fuzzy path planning of mobile robot. In: Evolutionary Computation, 2009. CEC '09. IEEE Congress on, pp. 1185–1192 (2009). doi:10.1109/CEC.2009.4983080

Kim JH, Kim YH, Choi SH, Park IW (2009) Evolutionary multi-objective optimization in robot soccer system for education. Comput Intell Mag IEEE 4(1):31–41. doi:10.1109/MCI.2008.930985

Krishnan, P., Paw, J., Kiong, T.S.: Cognitive map approach for mobility path optimization using multiple objectives genetic algorithm. In: Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on, pp. 267–272 (2009). doi:10.1109/ICARA.2000.4803970

LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)

Masehian, E., Sedighizadeh, D.: A multi-objective pso-based algorithm for robot path planning. In: Industrial Technology (ICIT), 2010 IEEE International Conference on, pp. 465–470 (2010). doi:10.1109/ICIT.2010.5472755

Masehian E, Sedighizadeh D (2010) Multi-objective PSO- and NPSO-based algorithms for robot path planning. Adv Electr Comput Eng 10(4):69–76. doi:10.4316/AECE.2010.04011

Masehian E, Sedighizadeh D (2010) Multi-objective robot motion planning using a particle swarm optimization model. J Zhejiang Univ Sci C 11(8):607–619. doi:10.1631/jzus.C0910525

Mo, H., Xu, Z., Tang, Q.: Constrained multi-objective biogeography optimization algorithm for robot path planning. In: Y. Tan, Y. Shi, H. Mo (eds.) Advances in Swarm Intelligence, Lecture Notes in Computer Science, vol. 7928, pp. 323–329. Springer, Berlin Heidelberg (2013). doi:10.1007/978-3-642-38703-6_38

Sedaghat, N.: Mobile robot path planning by new structured multi-objective genetic algorithm. In: Soft Computing and Pattern Recognition (SoCPaR), 2011 International Conference of, pp. 79–83 (2011). doi:10.1109/SoCPaR.2011.6089099

Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures, Fifth Edition. A Chapman & Hall book. Chapman & Hall/CRC, Boca Raton (2011)

Shih BY, Chang H, Chen CY (2013) Path planning for autonomous robots - a comprehensive analysis by a greedy algorithm. J Vib Control 19(1):130–142. doi:10.1177/1077546311429841

Wang, D., Kwok, N., Liu, D., Ha, Q.: Ranked pareto particle swarm optimization for mobile robot motion planning. In: D. Liu, L. Wang, K. Tan (eds.) Design and Control of Intelligent Robotic Systems, Studies in Computational Intelligence, vol. 177, pp. 97–118. Springer, Berlin Heidelberg (2009). doi:10.1007/978-3-540-89933-4_5

Wang, F., Zhu, Z.: Global path planning of wheeled robots using a multi-objective memetic algorithm. In: Yin, H., Tang, K., Gao, Y., Klawonn, F., Lee, M., Weise, T., Li, B., Yao X. (eds.) Intelligent Data Engineering and Automated Learning IDEAL 2013, Lecture Notes in Computer Science, vol. 8206, pp. 437–444. Springer, Berlin Heidelberg (2013). doi:10.1007/978-3-642-41278-3_53

Wei, J.H., Liu, J.S.: Generating minimax-curvature and shorter n3-spline path using multi-objective variable-length genetic algorithm. In: Networking, Sensing and Control (ICNSC), 2010 International Conference on, pp. 319–324 (2010). doi:10.1109/ICNSC.2010.5461496

Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2:78–84. doi:10.1504/IJBIC.2010.032124

Zhang Y, Gong DW, Zhang JH (2013) Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing 103:172–185. doi:10.1016/j.neucom.2012.09.019

Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Eiben, A., Back, T., Schoenauer, M., Schwefel H.P. (eds.) Parallel Problem Solving from Nature PPSN V, Lecture Notes in Computer Science, vol. 1498, pp. 292–301. Springer, Berlin Heidelberg (1998). doi:10.1007/BFb0056872

Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195. doi:10.1162/106365600568202

Zitzler E, Thiele L, Laumanns M, Fonseca C, da Fonseca V (2003) Performance assessment of multiobjective optimizers: an analysis and review. Evol Comput IEEE Trans 7(2):117–132. doi:10.1109/TEVC.2003.810758

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. Evol Comput IEEE Trans 3(4):257–271. doi:10.1109/4235.797969