Robotica (2016) volume 34, pp. 568–583. © Cambridge University Press 2014
doi:10.1017/S0263574714001702

Dynamic planning navigation strategy for mobile terrestrial robots

Átila V. F. M. de Oliveira and Marcelo A. C. Fernandes*

Department of Computer Engineering and Automation, Center of Technology, Federal University of Rio Grande do Norte – UFRN, Natal, Brazil

(Accepted June 4, 2014. First published online: July 4, 2014)

SUMMARY

This paper proposes a new dynamic planning navigation strategy for use with mobile terrestrial robots. The strategy was applied to situations in which the environment and obstacles were unknown. After each displacement event, the robot replanned its route using a control algorithm that minimized the distance to the target and maximized the distance between the obstacles. Using a spatial localization sensor and a set of distance sensors, the proposed navigation strategy was able to dynamically plan optimum routes that were free of collisions. Simulations performed using different types of environment demonstrated that the technique offers a high degree of flexibility and robustness, and validated its potential use in real applications involving mobile terrestrial robots.

KEYWORDS: Genetic algorithms; Mobile robots; Autonomous navigation; Dynamic planning.

1. Introduction

The evolution of robotics research in the last 15 years has had an important influence on human activities in many sectors of society. In the case of mobile robotics, the main objective is to develop autonomous intelligent robots that are able to effectively plan their movements in static or dynamic environments that are unknown.[1]

Methods involving artificial intelligence (AI) are among the techniques that can be used for this purpose, and there have been several reports of new navigation strategies showing a high level of precision in terms of avoidance of obstacles and achievement of optimum routes. In the area of AI, many navigation techniques employ fuzzy logic, artificial neural networks (ANNs), genetic algorithms (GAs), or combinations of these procedures.[2,3] Techniques based on GAs have been widely used due to their robustness, and in most cases have been able to provide optimum routes in both static and dynamic environments.[4–13]

In most of the studies that have been reported in literature, navigation techniques involving the use of GAs employ global or local planning strategies. In the case of the former, excellent routes can be obtained at high computational cost and with a requirement for *a priori* knowledge of the environment. In the case of the latter, suboptimal routes can be obtained at lower computational cost and with zero or negligible knowledge of the environment.[4,7] Both global and local planning can be applied to static and dynamic environments. However, for dynamic environments, global planning strategies need to use external observation devices in order to transmit the current state of the environment to the robot at a suitable rate.[8]

In refs. [4, 8, 10–13], navigation strategies using GA have been presented that employ global planning in which the individuals (or chromosomes) consist of all possible routes between the initial and final points. In all cases, *a priori* knowledge is required of the environment, which is represented by means of a bi-dimensional grid. The techniques proposed in some of these studies[4,10–13] are specific to static environments. The method described in ref. [8] is applicable to dynamic environments, although an external observation device is needed to provide the robot with information concerning the state of the environment at a rate that exceeds the rate at which the environment changes. Despite the fact that all these techniques were shown to be effective, three issues need to be highlighted. The

* Corresponding author. E-mail: mfernandes@dca.ufrn.br

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

first is that the size of the individual is variable as a function of the length of the route (the greater the complexity of the environment, the greater the length) and the resolution of the grid associated with the movement of the robot. This can significantly increase the spatial and temporal complexity of GA, making it unsuitable for use in limited hardware systems such as microcontrollers (MCUs), digital signal processors (DSPs), and others. Second, it is not always possible to obtain *a priori* knowledge of the environment that is important for global planning strategies. The third point is that these global strategies are best suited for static environments because of the problem of the need for external observation equipment in the case of dynamic environments.

Navigation strategies using GA based on local planning have been proposed for both dynamic[5,6] and static[7,9] environments. In these techniques, the individuals have sizes that are dynamic, and store the knots that compose the route. Despite using a local planning strategy, these procedures face the same problem as the global planning methods, where the complexity of GA is a function of the complexity of the environment and the resolution of the movement of the robot.

This work proposes a new dynamic planning navigation strategy based on GA, called the Dynamic Planning Navigation Algorithm optimized with GA (DPNA-GA). The objective of this strategy was to be able to identify an optimum collision-free route to a spatially defined objective. The DPNA-GA employs a set of distance sensors and a spatial localization sensor in order to reach the objective without any *a priori* knowledge of the environment.

Different from the methods reported previously,[4–13] the DPNA-GA strategy uses a navigation scheme involving local planning, in which the environment is unknown and the sizes of the individuals are fixed (only representing possible local objectives within which the robot can move) and independent of the complexity of the environment or the size of the route. Another important point is that the DPNA-GA can be applied to static or dynamic environments, given that the planning is reformulated at every movement. Despite the fact that navigation techniques with local planning provide suboptimal solutions, it can be seen from the results that in many cases it is possible to obtain routes that are very close to the optimum.

The main contributions of this work are (1) proposal of a new navigation strategy with local planning, called DPNA-GA, (2) description of a robot that uses DPNA-GA and therefore requires no *a priori* knowledge of its environment and obstacles' location, (3) consideration of both static and dynamic environments, and (4) individuals associated with the chromosomes that have a fixed size, independent of the complexity of the environment.

The work is organized as follows. Section 2 presents a detailed description of the algorithm on which the DPNA-GA technique is based; Section 3 describes the simulation results obtained for application of DPNA-GA in different types of environment; Section 4 compares DPNA-GA with other techniques; and finally, Section 5 presents the conclusions of the work.

2. Description of the Navigation Strategy

It was assumed that the robot possesses a localization sensor that returns its spatial position, $p^R = (x^R, y^R)$, and a set of n evenly distributed distance sensors. The navigation strategy based on DPNA-GA generates a route comprising M local displacements in order to reach the final objective, $p^{\text{fo}} = (x^{\text{fo}}, y^{\text{fo}})$. In each mth displacement event, there is a local objective, $p^{\text{lo}}(m) = (x^{\text{lo}}(m), y^{\text{lo}}(m))$, to which the robot moves.

The selection of the local objective $p^{\text{lo}}(m)$ in each mth event is performed by a GA that uses robot's current position, $p^R(m)$, taking into consideration the distance to the final objective, p^{fo}, and the obstacles detected by the n distance sensors. All the positions $p^R(m)$ already visited by the robot up to the mth displacement are stored in vector centers, $\mathbf{p}^R$, expressed by

$$\mathbf{p}^R = \begin{bmatrix} p^R(0) \\ p^R(1) \\ \vdots \\ p^R(m-1) \\ p^R(m) \end{bmatrix} = \begin{bmatrix} \left(x^R(0), y^R(0)\right) \\ \left(x^R(1), y^R(1)\right) \\ \vdots \\ \left(x^R(m-1), y^R(m-1)\right) \\ \left(x^R(m), y^R(m)\right), \end{bmatrix} \tag{1}$$

and are also used in the GA optimization function, avoiding searches in areas that have already been investigated.

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

The algorithm terminates when the current position of the robot is the same as the final objective, with $p^R(m) = (p^{\text{fo}} \pm \epsilon)$, where ϵ is a tolerance factor, or when the number of displacement events exceeds a maximum value, $M_{\max}$.

The DPNA-GA can be used in static and dynamic environments because at every mth movement event, there is a new search for obstacles and for a new local objective, $p^{\text{lo}}(m)$. The steps processed by DPNA-GA are presented in Algorithm 1, and are described in detail in the following sections.

Algorithm 1 DPNA-GA

1: $m = 0$
2: $\mathbf{p}^R = \left[p^R(0)\right]$
3: **while** $p^R(m) \neq \left(p^{\text{fo}} \pm \epsilon\right)$ **AND** $m < M_{\max}$ **do**
4: $\quad \mathbf{p}^{\text{DP}}(m) = \textit{Scanning}$
5: $\quad \mathbf{p}^O(m) = \textit{ObstacleDetection}\left(\mathbf{p}^{\text{DP}}(m)\right)$
6: $\quad p^{\text{lo}}(m) = \textit{SearchLocalObjective}\left(\mathbf{p}^O(m)\,, \mathbf{p}^{\text{DP}}(m), \mathbf{p}^R\right)$
7: $\quad p^R(m+1) = \textit{Displacement}\left(p^{\text{lo}}(m)\right)$
8: $\quad \mathbf{p}^R = \left[\mathbf{p}^R, p^R(m+1)\right]^T$
9: $\quad m = m + 1$
10: **end while**

2.1. Scanning

In the scanning step (line 4 of Algorithm 1), the DPNA-GA forces the robot to perform a 360° scan of the environment around its axis. From this scan, each jth sensor, during the mth event, returns an analog signal, $s_j(m)$, proportional to the distance limit, $d_{\max}$, in which

$$s_j(m) = \begin{cases} d_j(m) & \text{for } d_j(m) \leq d_{\max} \\ d_{\max} & \text{for } d_j(m) > d_{\max}, \end{cases} \tag{2}$$

where d_j is the distance measured by the jth sensor attached to the robot.

The value of angular displacement, α, can be expressed as

$$\alpha = \frac{360^\circ}{n \cdot p}, \tag{3}$$

where n is the number of distance sensors and $p - 1$ represents the number of angular displacements the robot can make in its axis, considering the reduced resolution due to a small number of sensors. At the end of the scanning process, the DPNA-GA generates a polygon, known here as a delimiting polygon (DP), comprising a set of K points, expressed by the vector

$$\mathbf{p}^{\text{DP}}(m) = \begin{bmatrix} p_0^{\text{DP}}(m) \\ \vdots \\ p_k^{\text{DP}}(m) \\ \vdots \\ p_{K-1}^{\text{DP}}(m) \end{bmatrix} = \begin{bmatrix} \left(x_0^{\text{DP}}(m), y_0^{\text{DP}}(m)\right) \\ \vdots \\ \left(x_k^{\text{DP}}(m), y_k^{\text{DP}}(m)\right) \\ \vdots \\ \left(x_{K-1}^{\text{DP}}(m), y_{K-1}^{\text{DP}}(m)\right) \end{bmatrix}, \tag{4}$$

where $K = p \times n$ and (x_k, y_k) represent the plane coordinates of the kth point associated with DP. This polygon is used to delimit the search space of GA. In other words, the individual points generated within the polygon have greater value than points generated outside it. Meanwhile, the ability of each individual point is not only determined by whether or not it is within the polygon, but also by the distance between the point generated and the obstacles detected by the scan, as well as other factors. Figure 1 illustrates the polygon generated by DPNA-GA for a case where $n = 4$ and $\alpha = 10^\circ$ ($p = 9$).

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

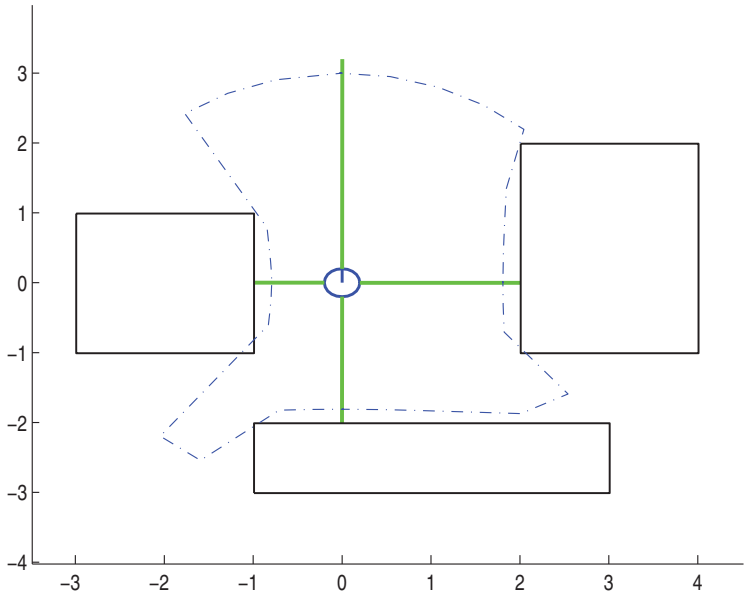


Fig. 1. Example of the delimiting polygon (blue line) for a case where $n = 4$ and $\alpha = 10^\circ$ ($p = 9$).

2.2. *Detection of obstacles*

Once the scanning stage is complete, the detection of obstacles phase begins (line 5 of Algorithm 1), in which a virtual polygon (VP) is generated in addition to the delimiting polygon. The VP considers the circumference centered on the robot (at position p^R), with radius r_{PV} that is slightly shorter than the maximum range ($d_{\max}$) of the sensors. Then,

$$r_{\text{PV}} = d_{\max}(1 - \eta), \tag{5}$$

where the factor η is limited to $0 \leq \eta < 0.1$. The role of VP is to detect only those points of DP that are associated with the obstacles, denoted as p^O. Hence, after this step, a new set of L points is generated, represented by the vector

$$\mathbf{p}^O(m) = \begin{bmatrix} p_0^O(m) \\ \vdots \\ p_l^O(m) \\ \vdots \\ p_{L-1}^O(m) \end{bmatrix} = \begin{bmatrix} \left(x_0^O(m), y_0^O(m)\right) \\ \vdots \\ \left(x_l^O(m), y_l^O(m)\right) \\ \vdots \\ \left(x_{L-1}^O(m), y_{L-1}^O(m)\right) \end{bmatrix}, \tag{6}$$

where

$$p_l^O(m) = p_k^{\text{DP}}(m) \text{ se } f_{\text{ed}}\left(p^R(m), p_k^{\text{DP}}(m)\right) \leq r_{\text{PV}}, \tag{7}$$

and $L \leq K$. The function $f_{\text{ed}}(\cdot, \cdot)$, which calculates the Euclidean distance between any two points, can be described by

$$f_{\text{ed}}(p_i, p_b) = \sqrt{(x_i - x_b)^2 + (y_i - y_b)^2}. \tag{8}$$

Figure 2 illustrates an example of VP (green circle) and the set of points $\mathbf{p}^O$ (red asterisks).

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

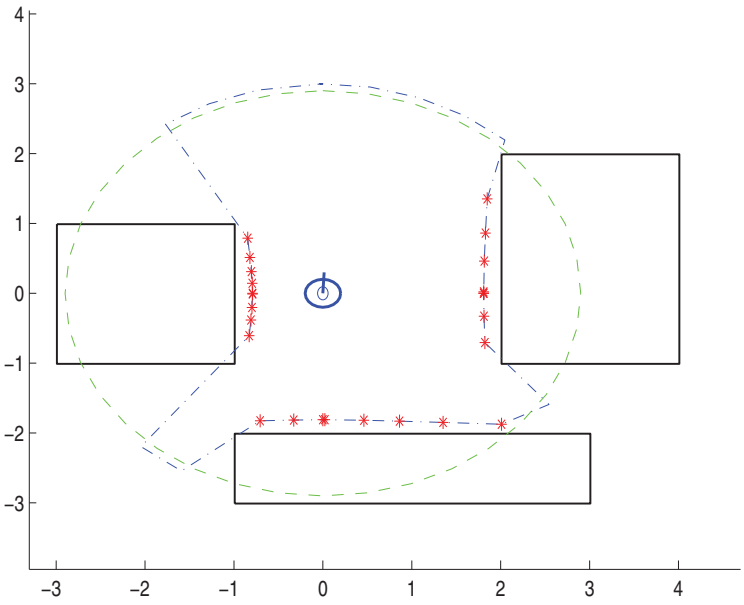


Fig. 2. Example for the case where $\eta = 0.01$, illustrating VP (green line) and the set of detected points ($\mathbf{p}^O$, red asterisks) that surround the obstacles.

2.3. *Search for the local objective*

In this step, (line 6 of Algorithm 1), the proposed navigation strategy uses a GA to find a possible local objective, p^{lo}, to which the robot can move. For each mth displacement event, GA, with a new population, is executed for H generations. The individuals are characterized by the vector

$$\mathbf{p}^{\text{GA}}(h,m) = \begin{bmatrix} p_0^{\text{GA}}(h,m) \\ \vdots \\ p_j^{\text{GA}}(h,m) \\ \vdots \\ p_{J-1}^{\text{GA}}(h,m) \end{bmatrix} = \begin{bmatrix} \left(x_0^{\text{GA}}(h,m),\, y_0^{\text{GA}}(h,m)\right) \\ \vdots \\ \left(x_j^{\text{GA}}(h,m),\, y_j^{\text{GA}}(h,m)\right) \\ \vdots \\ \left(x_{J-1}^{\text{GA}}(h,m),\, y_{J-1}^{\text{GA}}(h,m)\right) \end{bmatrix}, \tag{9}$$

where $p_j^{\text{GA}}(h)$ represents the jth individual of the population with size J, associated with the hth generation of the mth displacement of robot. In each generation, h, all the individual points are generated according to the nonlinear restriction given by

$$r_d \geq \sqrt{\left(x_j^{\text{GA}}(h,m) - x^R(m)\right)^2 + \left(y_j^{\text{GA}}(h,m) - y^R(m)\right)^2}. \tag{10}$$

This restriction limits the individual points of the population to a circumference with radius r_d, centered on the position of the robot in the mth instant, $p^R(m)$.

The evaluation function associated with the jth individual point of the hth generation of the mth displacement is expressed by

$$\begin{aligned} g_j(h,m) = d_j^{\text{fo}}(h,m) + \beta(m)\frac{1}{d_j^o(h,m)} \\ +\beta(m)A_j(h,m)C_j(h,m), \end{aligned} \tag{11}$$

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

where $d_j^{\text{fo}}(h, m)$ is the Euclidean distance between the jth point of the hth generation and the final objective, p^{fo}, characterized as

$$d_j^{\text{fo}}(h, m) = f_{\text{ed}}\left(p_j^{\text{GA}}(h, m),\, p^{\text{fo}}\right), \tag{12}$$

and $d_j^o(h, m)$ is the shortest Euclidean distance between the jth point of the hth generation and all the L obstacles found, expressed as

$$\begin{aligned} d_j^o(h, m) = \min\, & f_{\text{ed}}\left(p_j^{\text{GA}}(h, m),\, p_l^O(m)\right) \\ & \text{for } l = 0, \ldots, L-1. \end{aligned} \tag{13}$$

The variables $\beta(m)$, $C_j(h, m)$, and $A_j(h, m)$ can be characterized as penalty factors added to each jth individual of GA. When no obstacle is found in the mth displacement event ($L = 0$), it is assumed that the optimum evaluation function is simply $d_j^{\text{fo}}(h, m)$, so that

$$\beta(m) = \begin{cases} 1 & \text{for } L \neq 0 \\ 0 & \text{for } L = 0. \end{cases} \tag{14}$$

Starting from the principle that the circumferences with radius r_d, centered in the center vector $\mathbf{p}^R$, are areas that have already been visited, the penalty factor, $C_j(h, m)$, can be characterized as

$$C_j(h, m) = \begin{cases} 1 & \text{if} \\ & \nexists i \in \{0, \ldots, m-1\} : \\ & f_{\text{ed}}\left(p_j^{\text{GA}}(h, m),\, p^R(i)\right) < r_d, \\ Z & \text{if} \\ & \exists i \in \{0, \ldots, m-1\} : \\ & f_{\text{ed}}\left(p_j^{\text{GA}}(h, m),\, p^R(i)\right) < r_d \end{cases} \tag{15}$$

where Z is a relatively large number. Hence, an individual point $p_j^{\text{GA}}(h, m)$, located within one of the m circumferences with radius r_d, centered in the center vector, $\mathbf{p}^R$, would be positively penalized, reducing its chances of being selected. Finally, the penalty, $A_j(h, m)$, refers to individual points, $p_j^{\text{GA}}(h, m)$, located outside the DP, where

$$A_j(h, m) = \begin{cases} 1 & \text{if } \in PD \\ \infty & \text{if } \notin PD. \end{cases} \tag{16}$$

Individual points that are subject to this penalty will have minimal possibility of surviving until the next generation. Figure 3 illustrates the calculation of the evaluation function for the jth point $p_j^{\text{GA}}(h, m)$.

The evaluation function presented in Eq. (11) follows the same principle as the potential fields technique,[1] in which $d_j^{\text{fo}}(n, m)$ (the Euclidean distance between the jth point and the final objective, p^{fo}) represents a force of attraction to the final point. The term $\frac{1}{d_j^o(n,m)}$ (the shortest Euclidean distance between the jth individual point and all the points associated with obstacles) represents the greatest repulsive force between the jth individual point and all the obstacles found. After H generations, the smallest evaluation point is chosen as the local objective, $p^{\text{lo}}(m)$, associated with the mth displacement event.

2.4. Displacement

The displacement step (line 7 of Algorithm 1) is characterized by movement of robot to the local objective found in the previous step. After this movement, a new center point $p^R(m+1)$ is generated, described by

$$p^R(m+1) \neq \left(p^{\text{lo}}(m) \pm \epsilon\right), \tag{17}$$

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

Table I. Parameters employed in simulations.

Number of distance sensors (n)	4
Sensor range ($d_{\max}$)	3 m
Angular displacement (α)	10°
Radius (r_d)	1 m
Z (Eq. (15))	1000
Number of generations (H)	30
Population size (J)	30
Individual (chromosome) encoding	Real number
Selection method	Stochastic uniform
Elitism	Yes
Elite count	2
Crossover operator	Uniform
Crossover rate	80%
Mutation operator	Gaussian (0, 1)

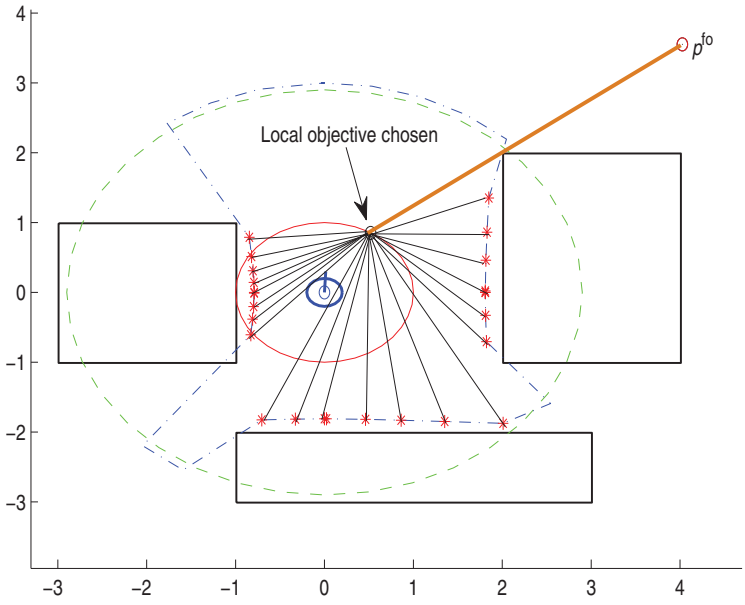


Fig. 3. Example illustrating calculation of the evaluation function for the jth individual point, $p_j^{\mathrm{GA}}(h, m)$, in relation to the final objective, p^{fo}, and obstacles, $\mathbf{p}^O$.

where ϵ is the admissible tolerance related to local objective. This tolerance can be most useful in the case of robots that have restricted movement, such as non-holonomic robots.[1] Figure 4 presents a sequence of $M = 6$ displacements up to the final point, p^{fo}.

3. Simulation Results

Simulations with different environments were used to validate the use of the DPNA-GA and compare its performance with other procedures. The simulations were implemented in Matlab employing an updated version of the iRobot Create toolbox,[14] simulating a circular non-holonomic robot with differential action, possessing four distance sensors with 90° spacing. Table I presents the parameters utilized in the simulations. The results are presented in Figs. 5–9, which show the routes followed by the robot (continuous black lines), the m displacement events (circles on route lines), and the DP associated with each displacement (dashed blue lines).

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

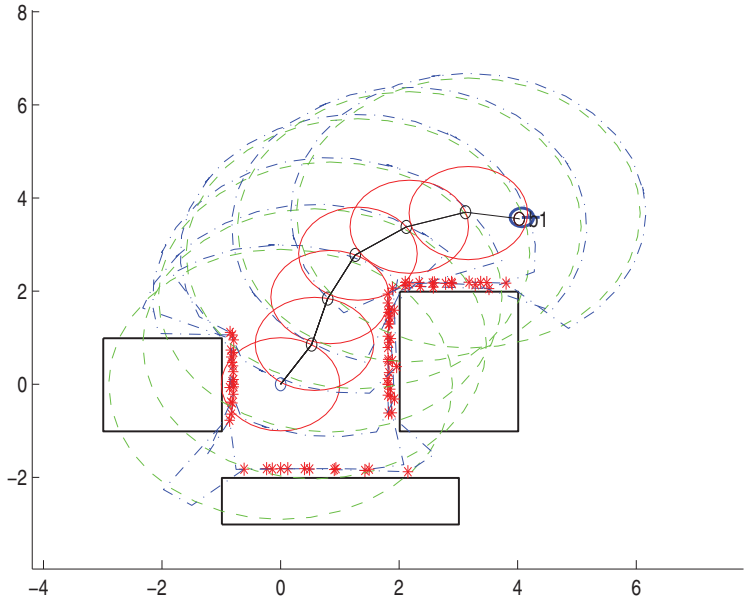


Fig. 4. Example illustrating the displacements ($M = 6$) made by robot up to the final point, p^{fo}.

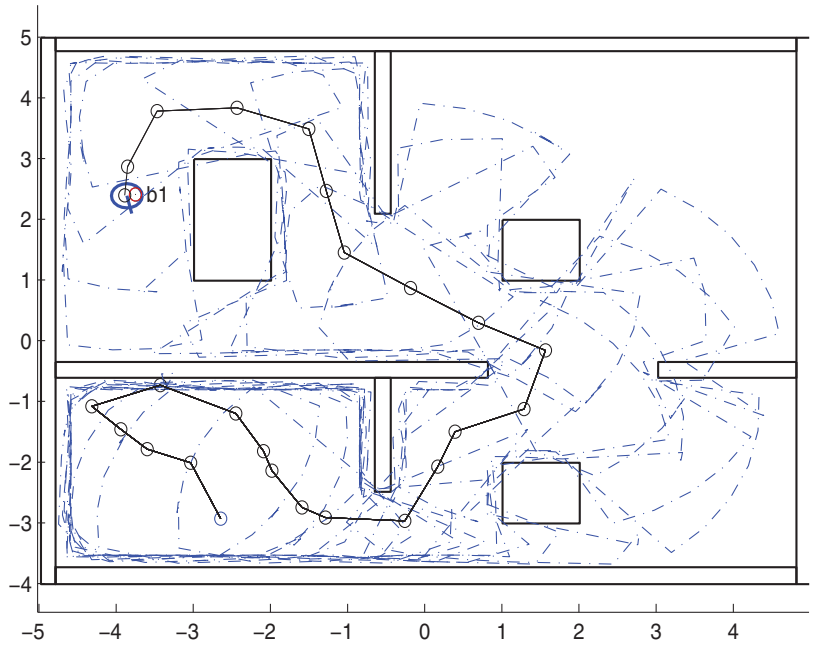


Fig. 5. Displacement of robot in a moderately complex environment. In a route that was close to the optimum, the robot required $M = 24$ displacement events.

Figure 5 shows robot's displacement in a moderately complex environment, where the robot required $M = 24$ displacement events to arrive at its destination. It can be seen that in the first instance ($0 \leq m \leq 10$), the DPNA-GA gave a sub-optimal route in which the robot was strongly attracted to the target point. However, for $m > 11$, the DPNA-GA provided an optimum route to the destination.

For the scenario shown in Fig. 6, robot was required to enter a type of U-shaped garage. In this case, the DPNA-GA required approximately $M = 18$ displacement events and provided the optimum path. In the scenario shown in Fig. 7, the robot was faced with a situation opposite to that of the previous

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

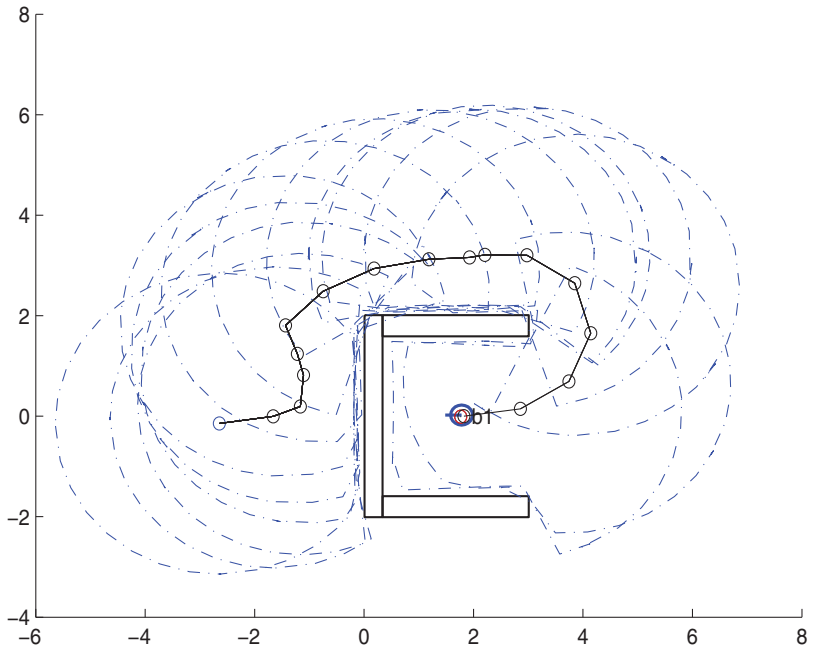


Fig. 6. Displacements required for the robot to enter a U-shaped environment using an optimum route ($M = 18$).

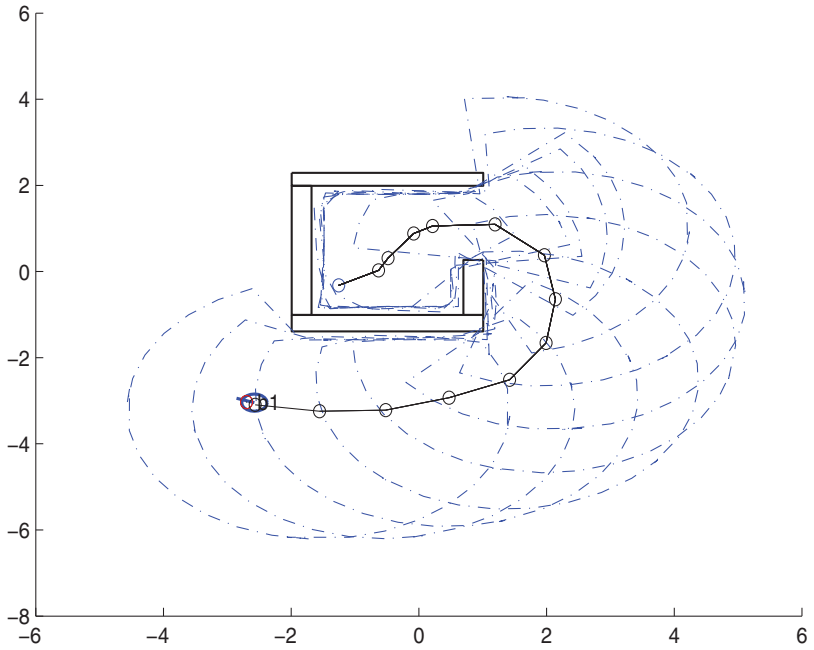


Fig. 7. Displacements required for the robot to leave a U-shaped environment using an optimum route ($M = 14$).

example, in which it was required to leave garage. Here the robot required $M = 14$ displacement events, and once again followed the optimum route.

The scenarios illustrated in Figs. 8 and 9 show the performance of navigation algorithm in the case of open environments with randomly scattered obstacles. Here the robot followed routes that were very close to optimum, requiring $M = 17$ and $M = 18$ displacements respectively.

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

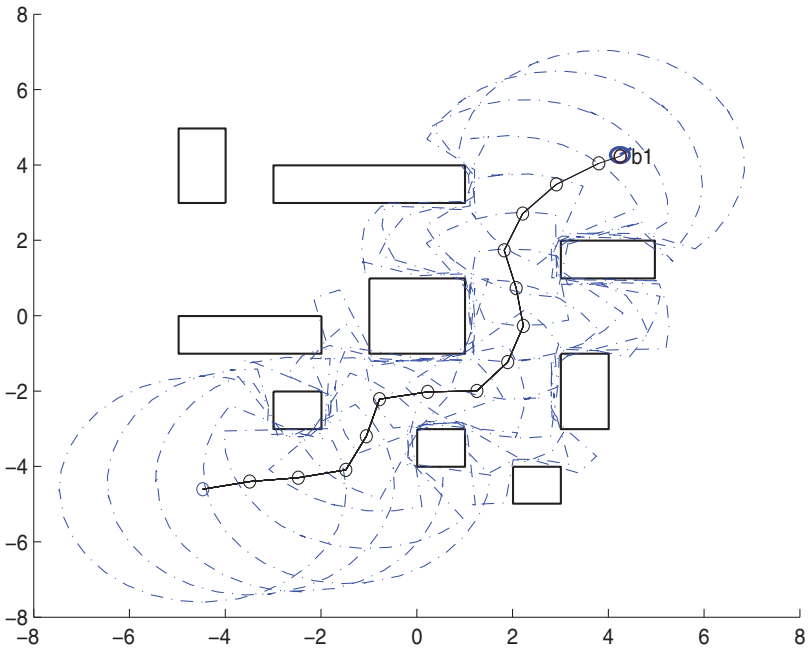


Fig. 8. Displacement of robot in an open environment with randomly scattered obstacles. In a route that was close to optimum, the robot required $M = 17$ displacement events.

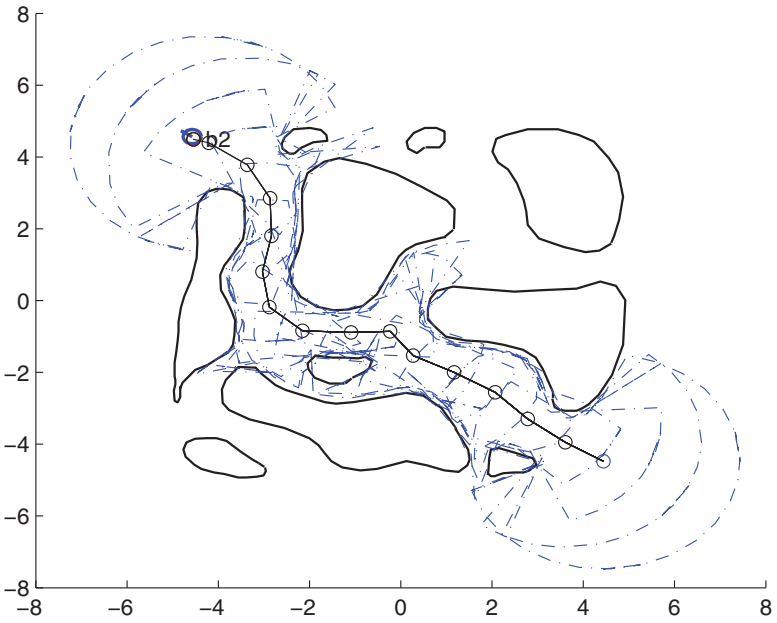


Fig. 9. Displacement of robot in an open environment with randomly scattered round obstacles. In a route that was close to optimum, the robot required $M = 18$ displacement events.

The scenarios illustrated in Figs. 10–12 show the navigation performance of mobile robot under duct environment. Similar to other cases, the robot followed routes that were very close to optimum, requiring $M = 17$ for L-shaped structure (Fig. 10), $M = 9$ for U-shaped structure (Fig. 11), and $M = 15$ for S-shaped structure (Fig. 12).

In order to observe the behavior of DPNA-GA in dynamic environments, simulations were performed using the scenarios presented in Figs. 6 and 7 with a dynamic obstacle that appears

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

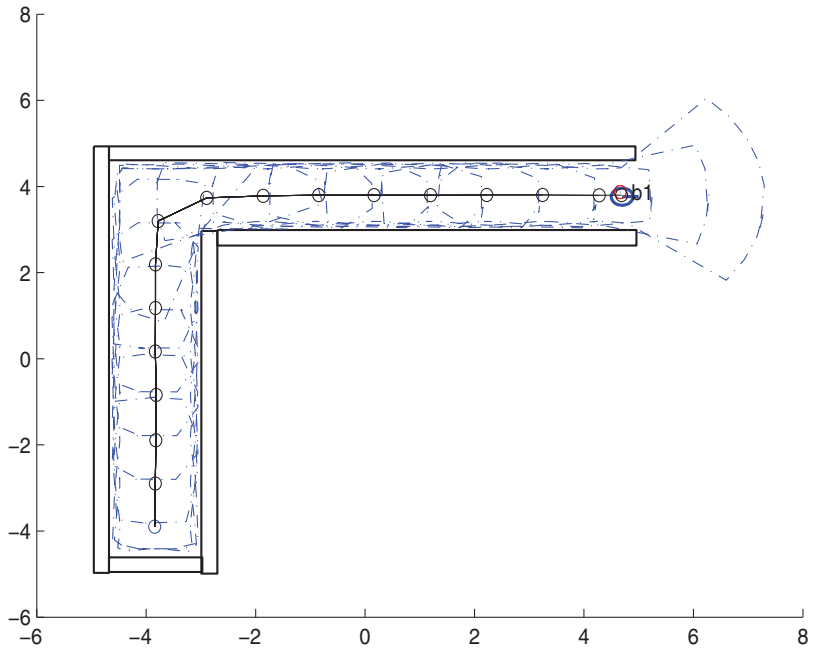


Fig. 10. Displacement of mobile robot under duct environment (L-shaped structure). In a route that was close to optimum, the robot required $M = 17$ displacement events.

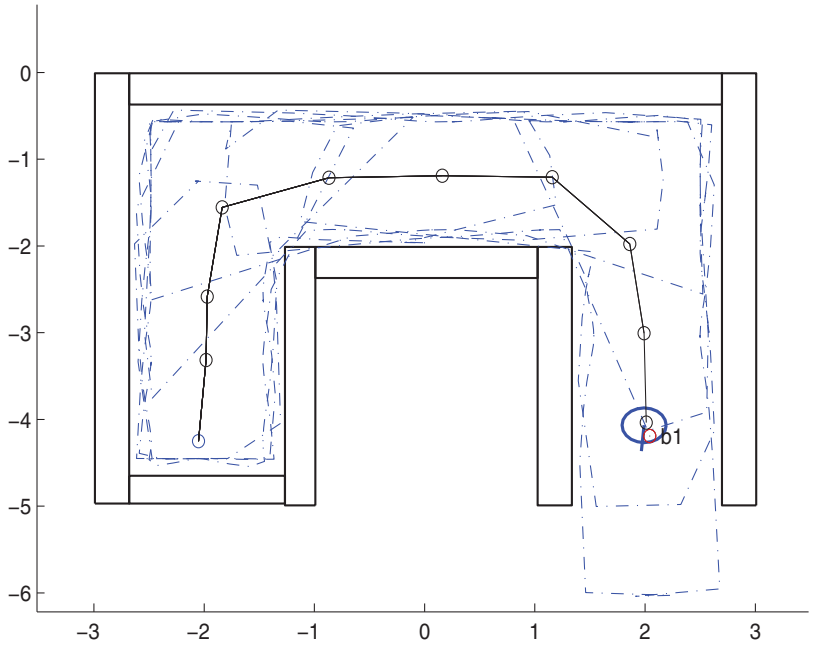


Fig. 11. Displacement of mobile robot under duct environment (U-shaped structure). In a route that was close to optimum, the robot required $M = 9$ displacement events.

suddenly in robot's front. The results, shown in Figs. 13 and 14, illustrate the route followed by robot in the presence of a new obstacle, demonstrating the effectiveness of the method for this type of environment.

Table II shows the average execution times of DPNA-GA for all simulated scenarios. Simulations were performed using a computer with a 64-bit CPU, a 2.4-GHz clock, and 4-GByte RAM. The

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

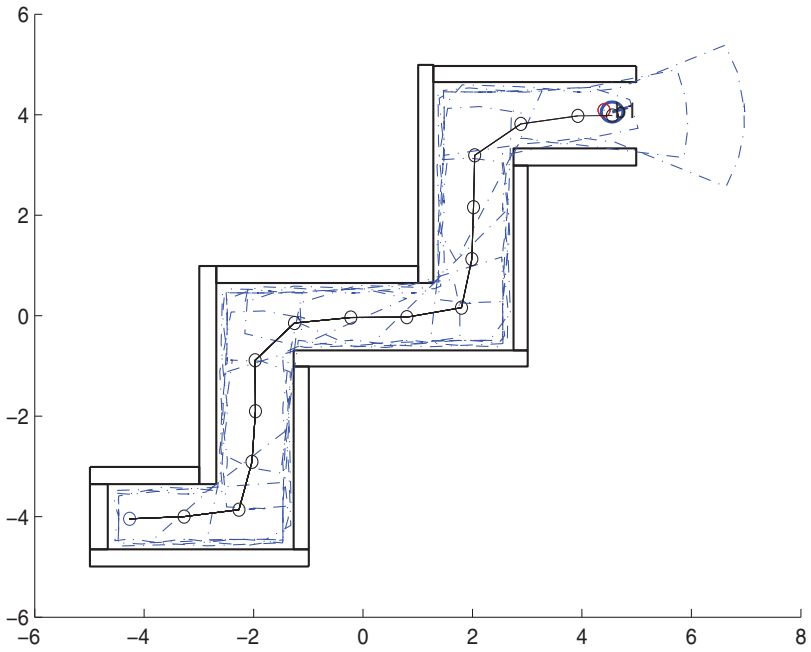


Fig. 12. Displacement of mobile robot under duct environment (S-shaped structure). In a route that was close to optimum, the robot required $M = 15$ displacement events.

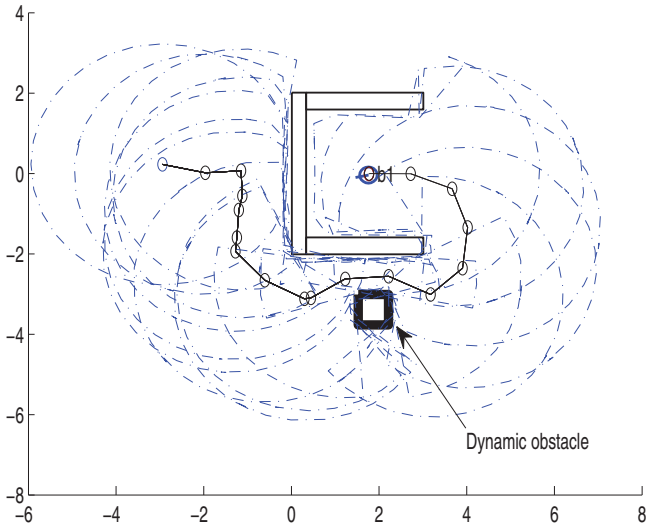


Fig. 13. Displacements required for the robot to enter a U-shaped environment using an optimum route ($M = 18$) with a dynamic obstacle that appears suddenly in robot's front.

data confirmed that the execution time of DPNA-GA was much shorter, compared with the strategies described earlier[12] (average value of 300 s), for scenarios similar to the one shown in Fig. 5. This difference was mainly related to the size of each individual, the number of generations, and the size of the population, which in DPNA-GA were limited to 2, 30, and 30, respectively. The method described in ref. [4], for example, used populations of up to 2000 individuals with sizes of around 140 values (in the best case) for a scenario similar to the one shown in Fig. 5. The technique described in ref. [6] used a population of 50 individuals associated with 2000 generations.

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

Table II. The average execution time of DPNA-GA for all the simulated scenarios.

Scenarios	Time (s)
Figure 5	51.0888
Figure 6	35.5072
Figure 7	28.5831
Figure 8	32.3145
Figure 9	34.0170
Figure 10	35.0304
Figure 11	20.3310
Figure 12	35.5635
Figure 13	35.8292
Figure 14	31.3140

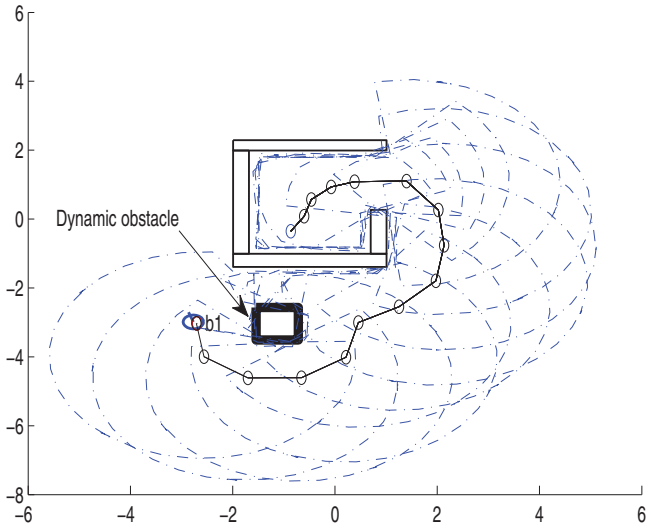


Fig. 14. Displacements required for the robot to leave a U-shaped environment using an optimum route ($M = 16$) with a dynamic obstacle that appears suddenly in robot's front.

4. Comparisons with Other Methods

In this section, comparisons are made between DPNA-GA and the algorithms proposed in refs. [8] and [10]. Four environments were compared: environments A (Fig. 15) and B (Fig. 16) were proposed in ref. [8], while environments C (Fig. 17) and D (Fig. 18) were proposed in ref. [10]. The algorithm described in ref. [8] presents a new genetic mutation operator applied to dynamic trajectory planning. In this, *a priori* knowledge is required of the location of obstacles in a grid-based map. For environment A, the algorithm described in ref. [8] required 100 iterations to find the best route, of which nine returned optimum routes, 78 returned routes close to the optimum, and 13 returned non-viable routes. The time required for each iteration was 1.68 s, so that 168 s were therefore required for the algorithm of ref. [8] to identify an optimum solution. In contrast, the DPNA-GA required 67 s to identify a route very close to the optimum (with a difference of 1.2 m). The difference between the distances can be explained by the fact that the DPNA-GA does not approach the edges of obstacles, hence avoiding robot's lateral collisions. Environment B simulates the appearance of an obstacle in the center of the map of environment A to demonstrate the dynamic planning ability of the algorithm. The algorithm in ref. [8] required 69 s to identify an optimum solution, with a route distance of 24.71 m, while the DPNA-GA required around 75 s to find a near-optimum solution with a distance of 27.31 m. It should be noted here that the DPNA-GA avoids paths that are relatively narrow. The improved GA for route planning described in ref. [10] is a global planning algorithm where knowledge of

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

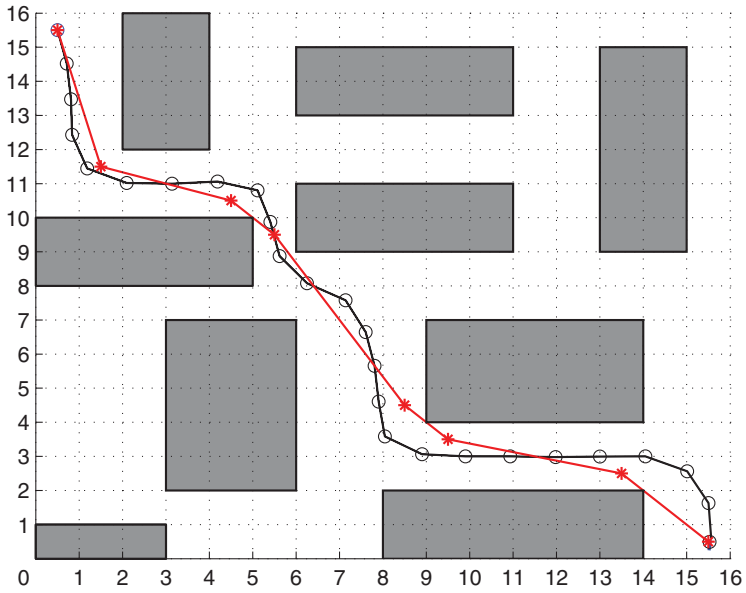


Fig. 15. Displacement required for the robot in environment A. Comparison between the method presented in ref. [8] (red line with asterisks) and the DPNA-GA (black line with circles).

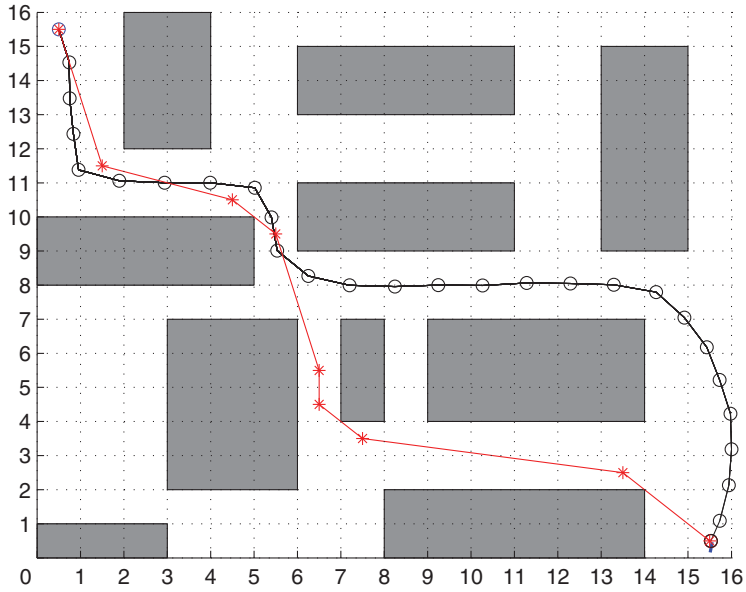


Fig. 16. Displacement required for the robot in environment B. Comparison between the method presented in ref .[8] (red line with asterisks) and the DPNA-GA (black line with circles).

the map is required. In contrast, the DPNA-GA is a local planning algorithm that requires no prior knowledge of the location of obstacles. In environment C, the DPNA-GA showed advantages in terms of distance and time, while in environment D the algorithm proposed in ref .[10] provided superior performance, which can be explained by the fact that the DPNA-GA plans robot's trajectory in real time. Comparative data for different algorithms are summarized in Table III.

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

Table III. Comparison between the methods presented in refs. [8, 10] with DPNA-GA.

	Proposed in refs. [8, 10]		DPNA-GA	
Environments	Route distance (m)	Time (s)	Route distance (m)	Time (s)
A	24.6800	168.0000	25.8905	67.2969
B	24.7100	69.0000	27.3064	75.2578
C	15.6569	49.0000	14.7790	36.5487
D	15.0711	49.0000	21.1726	56.7202

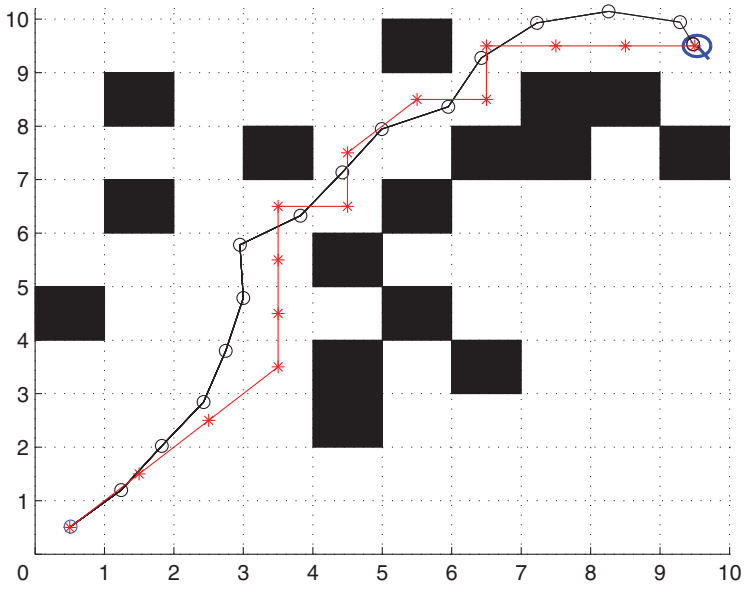


Fig. 17. Displacement required for the robot in environment C. Comparison between the method presented in ref. [10] (red line with asterisks) and the DPNA-GA (black line with circles).

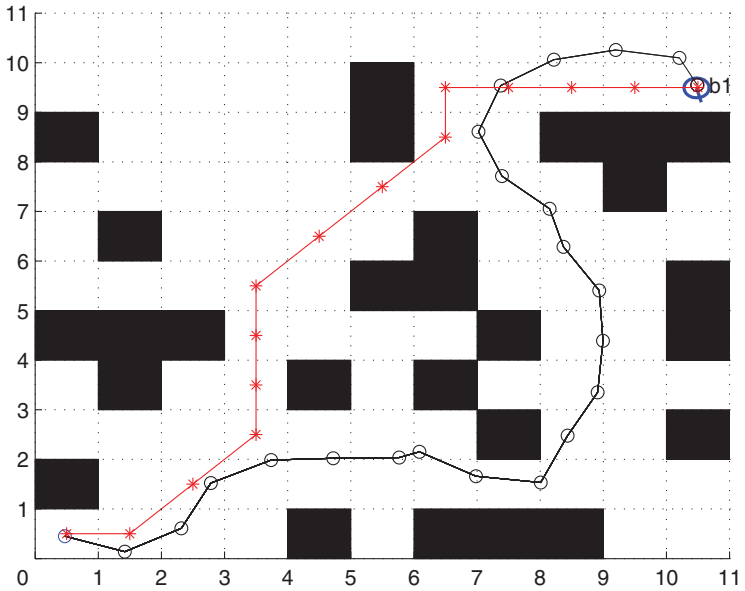


Fig. 18. Displacement required for the robot in environment D. Comparison between the method presented in ref .[10] (red line with asterisks) and the DPNA-GA (black line with circles).

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press

5. Conclusions

A new strategy based on GA was developed to implement a dynamic planning navigation scheme for use with mobile terrestrial robots that have no *a priori* knowledge of environment. The performance and robustness of the technique, known as DPNA-GA, was evaluated using simulations representing different environments.

Using the strategies described in literature as a starting point, the DPNA-GA offers a navigation scheme with local planning (applied to static and dynamic environments), without requiring *a priori* knowledge of the environment, and the size of the individuals is independent of the complexity of the environment. This feature is fundamental from the point of view of practical implementation.

The simulations demonstrate that the DPNA-GA can provide viable solutions for routes in various types of environment at a cost that is low compared with other global and local planning strategies. The results indicated that DPNA-GA should be suitable for use in practical tests with terrestrial robots.

In the future work, it is intended to analyze the performance of DPNA-GA using different values of the genetic parameters (such as the size of the population, number of generations, type of crossing and codification, and others) as well as to undertake practical testing of DPNA-GA using real robots.

References

1. R. Siegwart, I. Nourbakhsh and D. Scaramuzza, "Introduction to Autonomous Mobile Robots," **In**: *Intelligent Robotics and Autonomous Agents* (R. C. Arkin, ed.) (MIT Press, Cambridge MA, 2011) pp. 369–423.
2. R. Martinez-Soto, O. Castillo, L. Aguilar and I. Baruch, "Bio-inspired Optimization of Fuzzy Logic Controllers for Autonomous Mobile Robots," **In**: *2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, Berkeley, CA, USA (Aug. 6–8, 2012) pp. 1–6, IEEE. doi:10.1109/NAFIPS.2012.6291053.
3. C.-F. Juang and Y.-C. Chang, "Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments," *IEEE Trans. Fuzzy Syst.* **19**(2), 379–392 (2011). doi:10.1109/TFUZZ.2011.2104364.
4. A. S. Ismail AL-Taharwa and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *J. Comput. Sci.* **4**(4), 341–344 (2008). doi:10.3844/jcssp.2008.341.344.
5. S. C. Yun, S. Parasuraman and V. Ganapathy, "Dynamic Path Planning Algorithm in Mobile Robot Navigation," *Proceedings of the 2011 IEEE Symposium on Industrial Electronics and Applications (ISIEA)*, Langkawi, Malaysia (Sep. 25–28, 2011) pp. 364–369. doi:10.1109/ISIEA.2011.6108732.
6. P. Shi and Y. Cui, "Dynamic Path Planning for Mobile Robot Based on Genetic Algorithm in Unknown Environment," *Proceedings of the Control and Decision Conference (CCDC), 2010 Chinese* (May 26–28, 2010) pp. 4325–4329. doi:10.1109/CCDC.2010.5498349.
7. P. Shamsinejad, M. Saraee and F. Sheikholeslam, "A New Path Planner for Autonomous Mobile Robots Based on Genetic Algorithm," **In**: *Proceedings of the 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, vol. 8 (2010) pp. 115–120. doi:10.1109/ICCSIT.2010.5563666.
8. A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Comput. Electr. Eng.* **38**(6), 1564–1572 (2012). doi:10.1016/j.compeleceng.2012.06.016.
9. F. A. Cosío and M. P. C. Neda, "Autonomous robot navigation using adaptive potential fields," *Math. Comput. Modelling* **40**(9–10), 1141–1156 (2004). doi:10.1016/j.mcm.2004.05.001.
10. Z. Yongnian, Z. Lifang and L. Yongping, "An Improved Genetic Algorithm for Mobile Robotic Path Planning," *Proceedings of the 24th Control and Decision Conference (CCDC), 2012 Chinese* (2012) pp. 3255–3260. doi:10.1109/CCDC.2012.6244515.
11. L. Zhang, H. Min, H. Wei and H. Huang, "Global Path Planning for Mobile Robot Based on a*; Algorithm and Genetic Algorithm," *Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (2012) pp. 1795–1799. doi:10.1109/ROBIO.2012.6491228.
12. M. Samadi and M. F. Othman, "Global Path Planning for Autonomous Mobile Robot using Genetic Algorithm," *Proceedings of the 2013 International Conference on Signal-Image Technology Internet-Based Systems (SITIS)* (2013) pp. 726–730. doi:10.1109/SITIS.2013.118.
13. I. Chaari, A. Koubaa, H. Bennaceur, S. Trigui and K. Al-Shalfan, "Smartpath: A Hybrid Aco-ga Algorithm for Robot Path Planning," *Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC)* (2012) pp. 1–8. doi:10.1109/CEC.2012.6256142.
14. J. M. Esposito, O. Barton and J. Kohler, "Matlab toolbox for the irobot create," available at: www.usna.edu/Users/ weapsys/ esposito/ roomba.matlab/ (2011) (accessed June 5, 2013), online.

https://doi.org/10.1017/S0263574714001702 Published online by Cambridge University Press