sensors

MDPI

Article

Validation of a Dynamic Planning Navigation Strategy Applied to Mobile Terrestrial Robots

Caroline A. D. Silva, 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 59078-970, Brazil; carolads@gmail.com (C.A.D.S.); avfmo.engcomp@gmail.com (Á.V.F.M.d.O.)

* Correspondence: mfernandes@dca.ufrn.br; Tel.: +55-84-3215-3771

Received: 3 November 2018; Accepted: 27 November 2018; Published: 7 December 2018

Abstract: This work describes the performance of a DPNA-GA (Dynamic Planning Navigation Algorithm optimized with Genetic Algorithm) algorithm applied to autonomous navigation in unknown static and dynamic terrestrial environments. The main aim was to validate the functionality and robustness of the DPNA-GA, with variations of genetic parameters including the crossover rate and population size. To this end, simulations were performed of static and dynamic environments, applying the different conditions. The simulation results showed satisfactory efficiency and robustness of the DPNA-GA technique, validating it for real applications involving mobile terrestrial robots.

Keywords: genetic algorithms; mobile robots; autonomous navigation; dynamic planning

1. Introduction

In most of the studies concerning Genetic Algorithms (GAs) encountered in the literature, global or local planning strategies are employed. The former provides optimum routes, at high computational cost associated with a priori knowledge of the environment, while the latter provides suboptimal routes, at lower computational cost and with complete, or almost complete, lack of knowledge concerning the environment [1,2]. Global or local planning can be applied to static and dynamic environments, although in the case of dynamic environments, global planning strategies require the use of external observation devices to periodically transmit the current state of the environment to the robot [3].

Several studies [1,3–8] have described navigation strategies employing GAs, with global planning in which the individuals (or chromosomes) are composed of all the possible routes between the initial and final points. In all cases, a priori knowledge is required of the environment, which is represented using a bidimensional grid. Several of the proposed techniques are specific to static environments [1,4–7,9], while the proposal presented in Refs. [3,8] is aimed at dynamic environments, although an external observation device is needed to transmit the state of the environment to the robot at a speed faster than the speed of changes in the environment. Although efficient results have been reported in these earlier studies, three issues need to be highlighted. The first is that the size of the individual is variable and is 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 displacement of the robot. This can significantly increase the spatial and temporal complexity of the GA, making it unviable for use in limited hardware systems such as microcontrollers (MCUs), digital signal processors (DSPs), and others. The second issue is that it is not always possible to obtain the a priori knowledge of the environment that is important for global planning strategies. The third point is that these global strategies are better suited to static environments, due to the necessity of using external observation equipment for dynamic environments.

Navigation strategies with GAs based on local planning have been presented for dynamic [10,11] and static [2,12] environments. In these proposals, the individuals possess a dynamic size (dynamic dimension) and store the nodes that compose the route. Despite making use of local planning strategies, these proposals have the same problem described for the global planning methods, where the complexity of the GA is a function of the complexity of the environment and the resolution of the displacement of the robot. Another strategy is shown in Ref. [13] where the chromosomes are formed with the obstacle distances (left, right, front), angle and direction. This approach uses the individuals (chromosomes) with a static dimension which enables have a computation complexity less than the works proposed in Refs. [2,10–12].

The works presented in Refs. [14,15] propose global navigation planning, similar to the works [1,4–7,9] with other population-based metaheuristics algorithms [16]. The research [14] uses the ant colony optimization, and the work [15] uses the pseudo-bacterial genetic algorithm. Already, the work presented in Ref. [17] proposes the local planning for mobile robot navigation using firefly algorithm. Other approaches using artificial intelligence techniques such as machine learning, fuzzy systems, Q-learning are presented in Refs. [18–21].

Different to the studies cited above, the work described in Ref. [22] presents a navigation strategy called the Dynamic Planning Navigation Algorithm optimized with Genetic Algorithm (DPNA-GA). This strategy employs a navigation scheme with local planning, in which the environment is a priori unknown, and the sizes of the individuals are fixed (only representing possible local objectives through which the robot could move) and are 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 each displacement. Even though navigation techniques with local planning provide suboptimal solutions, it can be seen from the results presented that in many cases it is possible to obtain routes very close to the optimum.

Nonetheless, the work presented in Ref. [22] does not provide details of the operation of the DPNA-GA as a function of the genetic parameters. Therefore, the purpose of this work is to present results obtained using the DPNA-GA for static and dynamic environments, varying the genetic parameters to validate and generalize the technique proposed in Ref. [22].

2. DPNA-GA Strategy

To facilitate understanding of the results, this section details the DPNA-GA strategy presented in Refs. [22,23].

It is assumed that the robot possesses a location sensor, which returns its spatial position, $p^R = \left(x^R, y^R\right)$, and a set of n evenly distributed distance sensors. The navigation strategy based on the DPNA-GA generates a route composed of M local displacement events to reach the final objective, $p^{of} = \left(x^{of}, y^{of}\right)$. In each m-th displacement event, there is a local objective, $p^{ol}(m) = \left(x^{ol}(m), y^{ol}(m)\right)$, to which the robot moves.

The selection of the local objective, $p^{ol}(m)$, in each m-th event, is performed by a GA that considers the current position of the robot, $p^R(m)$, the distance to the final objective,p^{of}, and the obstacles detected by the n distance sensors. All the positions, $p^R(m)$, already visited by the robot up to the m-th displacement are stored in the vector, $\mathbf{p}^R$, expressed by

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

and are also used to optimize the GA, avoiding searches in areas that have already been explored.

The algorithm ends when the current position of the robot is the same as the final objective, so that, $p^R(m) = \left(p^{of} \pm \epsilon\right)$ 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 both static and dynamic environments, because at each m-th displacement event there is a new search for obstacles and for a new local objective, $p^{ol}(m)$. The steps processed by the 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** $p^R(m) \neq \left(p^{of} \pm \epsilon\right)$ **AND** $m < M_{max}$ **do**
4: $\quad \mathbf{p}^{DP}(m) = Scanning$
5: $\quad \mathbf{p}^O(m) = ObstaclesDetection\left(\mathbf{p}^{DP}(m)\right)$
6: $\quad p^{ol}(m) = LocalObjectiveSearch\left(\mathbf{p}^O(m), \mathbf{p}^{DP}(m), \mathbf{p}^R\right)$
7: $\quad p^R(m+1) = Displacement\left(p^{ol}(m)\right)$
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 Step

In this 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 j-th sensor, in the m-th event, returns a signal, $s_j(m)$, proportional to the range, d_{max}, of the sensor, so that

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

where d_j is the distance measured by the j-th sensor coupled to the robot.

During the scan, the angular displacement, α, can be expressed by

$$\alpha = \frac{360^\circ}{n \cdot p}, \quad (3)$$

where n is the number of distance sensors and $p - 1$ represents the number of angular displacements that the robot can make on its axis, with the aim of decreasing the resolution and hence requiring a

small number of sensors. At the end of the scan process, the DPNA-GA generates a polygon, called the delimiting polygon (DP), composed of a set of K points, expressed by the vector

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

where $K = p \times n$ and (x_k, y_k) represents the in-plane coordinates of the k-th point associated with the DP. This polygon is used to delimit the search space associated with the genetic algorithm, such that the points (individuals) generated within the polygon are more suitable than points generated outside it. Meanwhile, it is not only the fact of being within or outside the polygon that defines the suitability of each individual. Also considered are the distances between the point generated and the obstacles detected in the scan, among other factors. Figure 1 illustrates the polygon generated by the DPNA-GA for the case of $n = 4$ and $\alpha = 10°$.

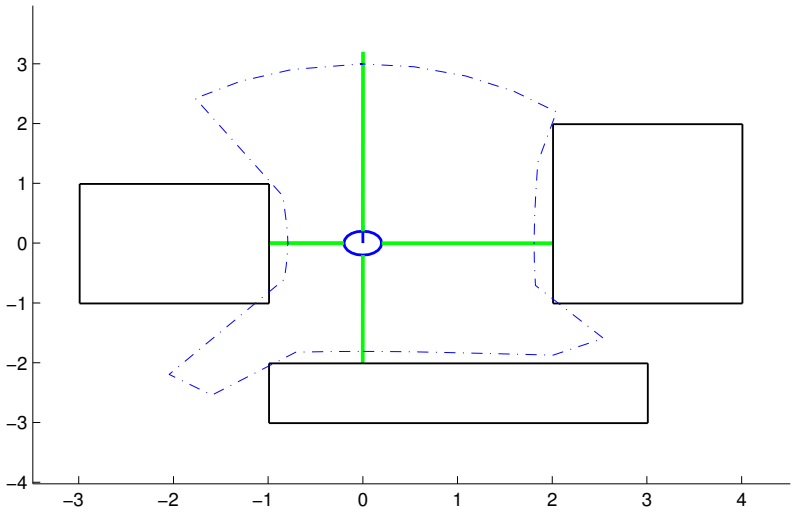

Figure 1. Example of the delimiting polygon (dashed blue line) for the case where $n = 4$ and $\alpha = 10°$ ($p = 9$).

2.2. Detection of Obstacles

The scan step is followed by initiation of the step for detection of the obstacles (line 5 of Algorithm 1). In addition to the DP, a virtual polygon (VP) is generated that describes a circumference centered on the position of the robot (p^R), with radius r_{PV}, slightly less than the range of the sensors, d_{max}, such that

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

where η is a factor limited to the range $0 < \eta \leq 0,1$. The objective of the VP is to detect only those points of the DP that are associated with obstacles, here denoted p^O. Hence, after this step, a new set of L points is generated, represented by the vector

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

where

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

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

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

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

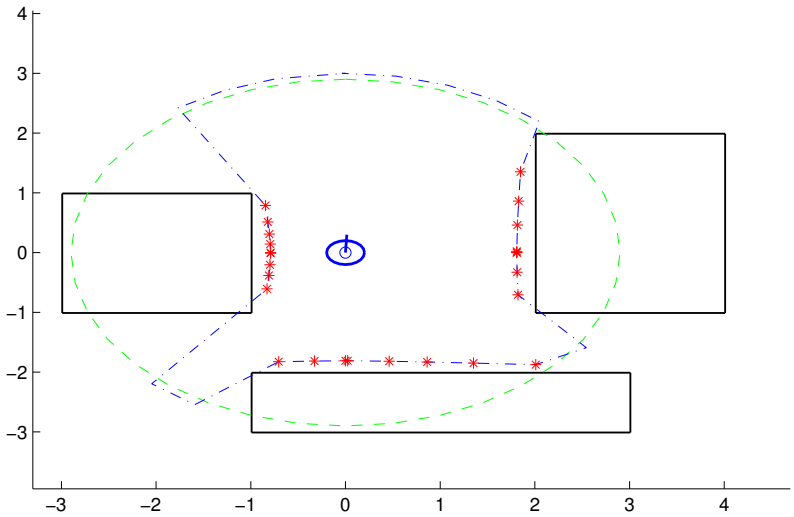

Figure 2. Example illustrating the VP (green line) for a case where $\eta = 0.01$, and the set of points, $\mathbf{p}^O$, detected (red asterisks) that avoid the obstacles.

2.3. Local Objective Search

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

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

where $p_j^{GA}(h,m)$ represents the j-th individual of the population of size J, associated with the h-th generation of the m-th displacement of the robot. In each generation, h, all the individuals are generated according to the nonlinear restriction expressed by

$$r_d \geq \sqrt{dx_j^{GA}(h,m) + dy_j^{GA}(h,m)} \tag{10}$$

where

$$dx_j^{GA}(h,m) = \left(x_j^{GA}(h,m) - x^R(m)\right)^2 \tag{11}$$

and

$$dy_j^{GA}(h,m) = \left(y_j^{GA}(h,m) - y^R(m)\right)^2 \tag{12}$$

This restriction limits the individuals of the population to a circumference with radius r_d, centered on the position of the robot at the m-th instant, $p^R(m)$. Usually, DP occupies most of the circle with radius r_d, so that a few individuals are created outside DP. Thus, using the coordinates of DP as constraints on population creation would result in a much more complex creation routine, with few practical compensations.

The evaluation function associated with the j-th individual of the h-th generation in the m-th displacement is expressed by

$$\begin{aligned} g_j(h,m) &= d_j^{of}(h,m) + \beta(m)\tfrac{1}{d_j^o(h,m)} \\ &\quad + \beta(m)C_j(h,m) + A_j(h,m) \end{aligned} \tag{13}$$

where $d_j^{of}(h,m)$ is the Euclidean distance between the j-th individual of the h-th generation and the final objective, p^{of}, such that

$$d_j^{of}(h,m) = f_{ed}\left(p_j^{GA}(h,m), p^{of}\right) \tag{14}$$

and $d_j^o(h,m)$ is the shortest Euclidean distance between the j-th individual of the h-th generation and all the L obstacles encountered, which can be expressed as

$$\begin{aligned} d_j^o(h,m) &= \min f_{ed}\left(p_j^{GA}(h,m), p_l^O(m)\right) \\ &\quad \text{for } l = 0, \ldots, L-1 \end{aligned} \tag{15}$$

The variables $\beta(m)$, $C_j(h,m)$ and $A_j(h,m)$ can be considered as penalty factors added to each j-th individual of the GA. If no obstacle is encountered in the m-th displacement event ($L = 0$), it is assumed that the optimum evaluation function is simply $d_j^{of}(h,m)$, such that

$$\beta(m) = \begin{cases} 1 \text{ for } L \neq 0 \\ \\ 0 \text{ for } L = 0 \end{cases} \tag{16}$$

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

$$C_j(h,m) = \begin{cases} 1 & \text{if} \\ & \nexists i \in \{0,\cdots,m-1\}: \\ & f_{ed}\left(p_j^{GA}(h,m), p^R(i)\right) < r_d \\ \\ Z & \text{if} \\ & \exists i \in \{0,\cdots,m-1\}: \\ & f_{ed}\left(p_j^{GA}(h,m), p^R(i)\right) < r_d \end{cases} \tag{17}$$

where Z is a relatively large number. Hence, if an individual, $p_j^{GA}(h,m)$, is located within any of the m circumferences of radius r_d, centered in the vector of the center, $\mathbf{p}^R$, it will be positively penalized, reducing its chances of selection. Finally, the penalty, $A_j(h,m)$, is referenced to the individuals, $p_j^{GA}(h,m)$, located outside the DP, where

$$A_j(h,m) = \begin{cases} 0 & \text{if } \in DP \\ \\ \infty & \text{if } \notin DP \end{cases} \tag{18}$$

In this last case, individuals that receive this penalty will have little chance of surviving to the next generation. Figure 3 illustrates the calculation of the evaluation function for a j-th individual, $p_j^{GA}(h,m)$.

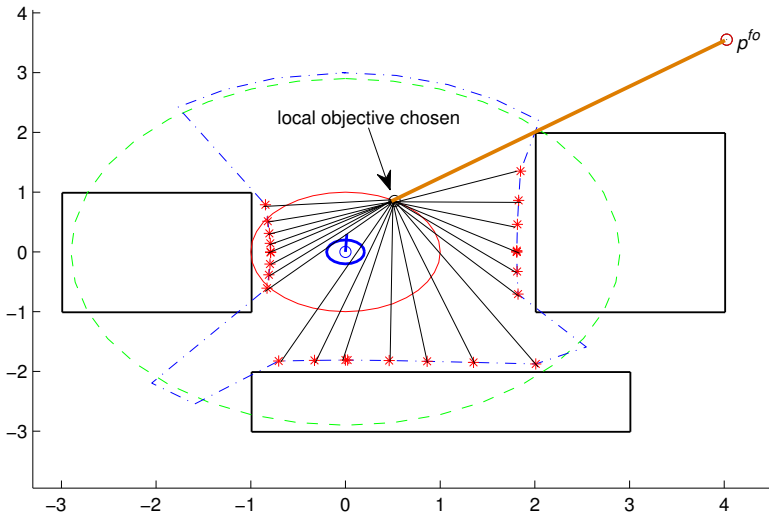

Figure 3. Example illustrating the calculation of the evaluation function for a j-th individual, $p_j^{GA}(h,m)$, in relation to the final objective, p^{of}, and the obstacles, $\mathbf{p}^O$.

The evaluation function, presented in Equation (13), follows the same principle as the potential fields technique [24], in which $d_j^{of}(n,m)$ (the Euclidean distance between the j-th individual and the final objective, p^{of}) represents a traction force to the final point, and $\frac{1}{d_j^o(n,m)}$ (the smallest Euclidean

distance between the j-th individual and all the points associated with the obstacles) represents the greatest force of repulsion between the j-th individual and all the obstacles encountered. At the end of H generations, the point with the smallest evaluation function is selected as the local objective, $p^{ol}(m)$, associated with the m-th displacement event.

2.4. Displacement

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

$$p^R(m+1) \neq \left(p^{ol}(m) \pm \epsilon\right) \tag{19}$$

where ϵ is an allowed tolerance in relation to the local objective. This tolerance is essential to the robot with restricted movements, such as nonholonomic robots [24] and errors from real measures. Figure 4 shows a sequence of $M = 6$ displacements to the final point, p^{of}.

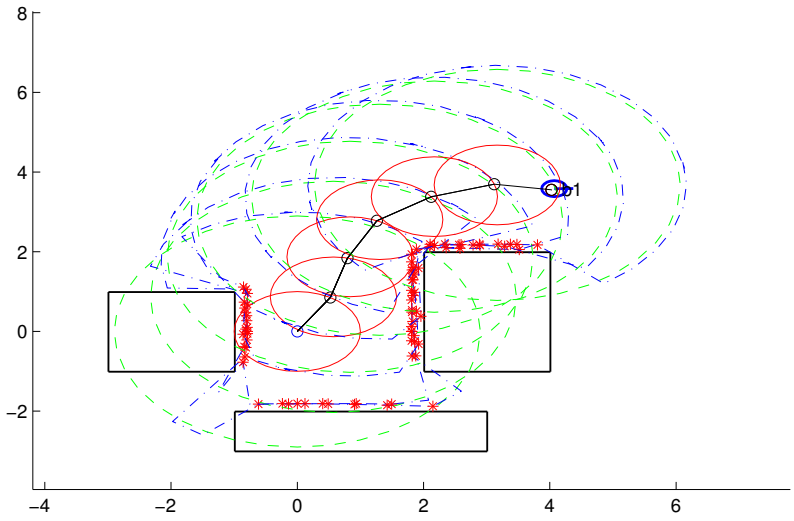

Figure 4. Example illustrating the displacements ($M = 6$) made by the robot towards the final point, p^{of}.

3. Simulation Results

To validate the functioning of the DPNA-GA (for static and dynamic environments), considering its robustness in terms of the genetic parameters, simulations were conducted using two types of environment (A_1 and A_2), varying the number of generations (H), the size of the population (J), and the crossover rate (R_c). The simulations were performed in MATLAB, using the updated version of the iRobot Create toolbox [25,26]. The toolbox simulated a circular nonholonomic robot with variable action and four distance sensors spaced at 90°. Table 1 presents the fixed parameters used in the simulations. Each simulation (Figures 5–12) was executed ten times, and the results are associated with the average of the values obtained in all executions.

Table 1. Common parameters used in the simulations.

Number of sensors (n)	4
Maximum sensor range (d_{max}) in meters	3 m
Angular displacement (α)	10°
Radius (r_d) in meters	1 m
Z (Equation (17))	1000
Codification of individuals	Real number
Selection method	Stochastic uniform
Elitism	Yes (2 individuals)
Crossover operator	Uniform random $[0, 1]$
Mutation operator	Gaussian random $N(0, 1)$

The individuals used the real number encoding method, the crossover operator used the intermediate scheme where the offspring ($p_i^{GA}(h+1,m)$ and $p_v^{GA}(h+1,m)$) are chosen using the uniform random number, that is,

$$p_i^{GA}(h+1,m) = p_l^{GA}(h,m)r(h,m) + p_k^{GA}(h,m)(1-r(h,m)) \tag{20}$$

and

$$p_v^{GA}(h+1,m) = p_l^{GA}(h,m)(1-r(h,m)) + p_k^{GA}(h,m)r(h,m) \tag{21}$$

where the $p_l^{GA}(h,m)r_j(h,m)$ and $p_k^{GA}(h,m)r_j(h,m)$ are individuals chosen from $\mathbf{p}^{GA}(h,m)$ in selection step and $r(h,m)$ is a uniform random number between 0 and 1. Using Equation (9), Equations (20) and (21) can be rewritten as

$$x_i^{GA}(h+1,m) = x_l^{GA}(h,m)r(h,m) + x_k^{GA}(h,m)(1-r(h,m)), \tag{22}$$

$$x_v^{GA}(h+1,m) = x_l^{GA}(h,m)(1-r(h,m)) + x_k^{GA}(h,m)r(h,m), \tag{23}$$

$$y_i^{GA}(h+1,m) = y_l^{GA}(h,m)r(h,m) + y_k^{GA}(h,m)(1-r(h,m)), \tag{24}$$

and,

$$y_v^{GA}(h+1,m) = y_l^{GA}(h,m)(1-r(h,m)) + y_k^{GA}(h,m)r(h,m). \tag{25}$$

As the mutation operator, it was used the Gaussian mutation operator expressed as

$$p_i^{GA}(h+1,m) = p_i^{GA}(h,m) + g(h,m) \tag{26}$$

where the $g(h,m)$ is the Gaussian random variable of median zero and variance σ, $N(0,\sigma)$, associated of the h-th generation of the m-th displacement of the robot. Using Equation (9), Equation (26) can be rewritten as

$$x_i^{GA}(h+1,m) = x_i^{GA}(h,m) + g_x(h,m), \tag{27}$$

and,

$$y_i^{GA}(h+1,m) = y_i^{GA}(h,m) + g_y(h,m), \tag{28}$$

where the $g_x(h,m)$ and $g_y(h,m)$ are the Gaussian random variable of median zero and variance σ, $N(0,\sigma)$, associated of the x and y coordinates, respectively. In all simulations, it was used variance, $\sigma = 1$.

Eight simulations were made, four for environment A_1 (simulations S_1, S_2, S_3, and S_4) and four for environment A_2 (simulations S_5, S_6, S_7, and S_8). For each simulation, Tables 2 and 3 show the data for the length of the route, c_p, travelled by the robot (in meters), the processing time associated with all the displacements along the route, t_p (in seconds), and the number of displacement events, M. The displacements of the robot in each simulation are illustrated in Figures 5–12, where the route is indicated by a continuous black line, the m displacement events are shown as circles along

the route lines, and the DPs associated with each displacement are indicated by dashed blue lines. The simulations were performed using a computer with a 64 bits CPU (Intel(R) Core(TM) i5-3210M), 2.5 GHz clock speed, and 8 GBytes of RAM.

During the development of the work, several combinations of GA parameters were tested. How the target of the proposed method (DPNA-GA) is for embedded systems with low processing (such as microcontrollers), it was measured the time processing in seconds per displacement event (s/disp), t_d, for all simulations. After that, it was observed that the combinations with high population size ($J > 30$), large generations number ($H > 30$) and low crossover rate ($R_c < 60\%$) achieved high values of td ($t_d > 3$ s/disp). However, values of $t_d > 3$ s/disp can reduce the continuity of movement associated with the robot. Thus, the eight simulations (S_1 to S_8) were chosen using the criterion of $t_d < 2.5$ s/disp (time processing in seconds per displacement event).

The size of the population at $J = 10$ is a relatively low amount for the GA standards, and increasing that amount generally implies a slight improvement in GA convergence values, but a considerable increase in processing time in environments with many obstacles. A similar situation is the one that concerns the crossover rate. Lowering the crossover rate, R_c, from 60% to close values or increasing the crossover rate, R_c, from 80% to close values, the results did not show significant differences. Lowering from 60% crossover rate to distant values has led to bad results that were already expected. Based on these results, we decided to include in our analysis only the most significant ones.

In the case of environment A_1 (results shown in Table 2 and Figures 5–8), it can be seen that the navigation strategy showed little variation in terms of the number of displacement events, m (mean of 19.5 and standard deviation of 2.5), and the length of the route, c_p (mean of 18.1 m and standard deviation of 1.67 m). However, greater variability was found for the processing time (mean of 28.92 s and standard deviation of 12.88 s).

Comparing simulations that have the same crossover rate (simulations S_1 and S_2, and simulations S_3 and S_4), it can be observed that the ones with a larger population size (simulations S_1 and S_3) have slightly higher performances in terms of route size and number of displacements. However, this little increase in performance does not compensate for the increase in processing time, which is more than twice its counterparts are. On the other hand, when comparing simulations with the same population size (simulations S_1 and S_3, and simulations S_2 and S_4), the increase in crossover rate meant a smaller route size and less displacements, with a less significant increase in processing time than in the previous comparison.

Lowering the crossover rate led to a more elitist configuration of the population, letting more individuals continue unchanged in the next generation. For a routing problem, reaching a few points in the search space can lead to a poor convergence of the algorithm. A larger population increases the variability of solutions reached in the search space, as can be seen in simulation S_3 (Figure 7), but it does not quite compensate for the lower crossover rate in this case. The simultaneous decrease of these parameters of genetic variability, as shown in simulation S_4 (Figure 8), leads to the worst convergence of the algorithm among the simulations, causing the robot to do bad and/or unnecessary displacements.

Finally, another important point to emphasize is that reduction of the size of the population only slightly increased the number of displacements (from 17 to 19) and greatly reduced the total time associated with the displacements (from 39.31 s to 15.64 s).

Table 2. Parameters used in the simulations of environment A_1.

	Population Size (J)	**Generations Number (H)**	**Crossover Rate (R_c)**	**Length of Route (c_p)**	**Processing Time (t_p)**	**Displacement Events (M)**	**Time per Displacement Event (t_d)**
S_1	30	30	80%	16.52 m	39.31 s	17	2.31 s/disp
S_2	10	30	80%	17.10 m	15.64 s	19	0.82 s/disp
S_3	30	30	60%	18.46 m	40.60 s	19	2.14 s/disp
S_4	10	30	60%	20.28 m	20.15 s	23	0.88 s/disp

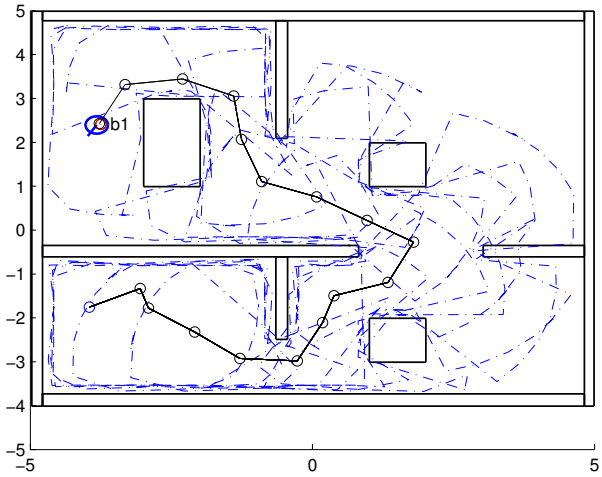

Figure 5. Displacement of the robot in simulation S_1 ($J = 30$, $H = 30$ and $R_c = 80\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

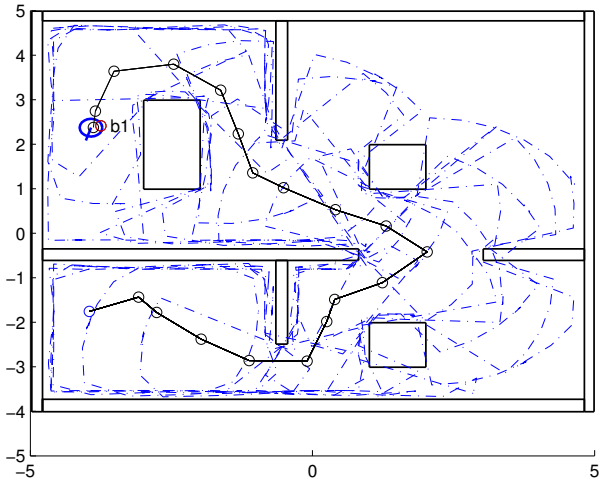

Figure 6. Displacement of the robot in simulation S_2 ($J = 10$, $H = 30$ and $R_c = 80\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

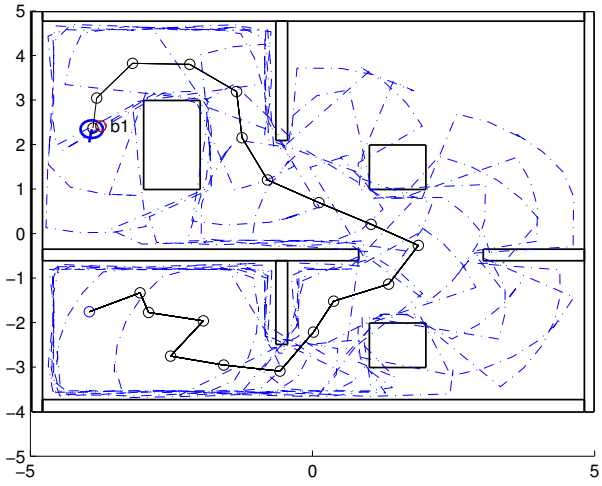

Figure 7. Displacement of the robot in simulation S_3 ($J = 30$, $H = 30$ and $R_c = 60\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

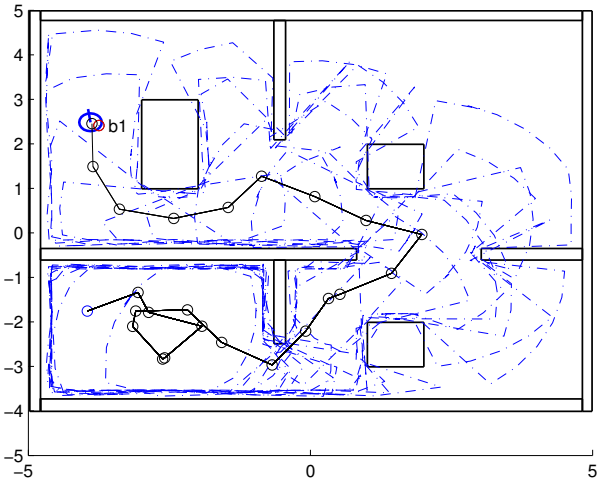

Figure 8. Displacement of the robot in simulation S_4 ($J = 10$, $H = 30$ and $R_c = 60\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

In the simulations using environment A_2 (S_5, S_6, S_7, and S_8), the robot encountered a dynamic obstacle (red rectangle) and had to avoid it. The results of these simulations are shown in Table 3 and Figures 9–12. Different to the results for environment A_1, the data for the length of the route, c_p (mean of 13.25 m and standard deviation of 0.47 m), the processing time, t_p (mean of 10.43 s and standard deviation of 0.67 s), and the number of displacement events, M (mean of 13.5 and standard deviation of 0.58) showed very low variability associated with changes in the genetic parameters. This result could be explained by the alternation between the size of the population, J, and the number of generations, H, in simulations S_5, S_6, S_7, and S_8.

Compared to the findings for environment A_1, environment A_2 showed poorer results, with a crossover rate of 80%. This could be explained by the simplicity of environment A_2, relative to environment A_1, which did not require a high rate of renewal of the individuals of the population.

The data shown in Tables 2 and 3 demonstrate that the execution time of the DPNA-GA is much shorter than for the strategies presented previously [6] for an environment similar to A_1. This difference is mainly associated with the size of each individual, the number of generations, and the size of the population, which in the case of the DPNA-GA were limited to 2, 30, and 30, respectively. The proposal described in Ref. [1], for example, employed populations of up to 2000 individuals with size of around 140 values (in the best case), for an environment similar to A_1. In other work, a population of 50 individuals was used, together with 2000 generations [11].

Table 3. Parameters used in the simulations of environment A_2.

	Population Size (J)	**Generations Number (H)**	**Crossover Rate (R_c)**	**Length of Route (c_p)**	**Processing Time (t_p)**	**Displacement Events (M)**	**Time per Displacement Event (t_d)**
S_5	30	10	80%	13.94 m	10.65 s	14	0.76 s/disp
S_6	10	30	80%	13.19 m	11.25 s	14	0.80 s/disp
S_7	30	10	60%	12.87 m	9.68 s	13	0.74 s/disp
S_8	10	30	60%	12.93 m	10.16 s	13	0.78 s/disp

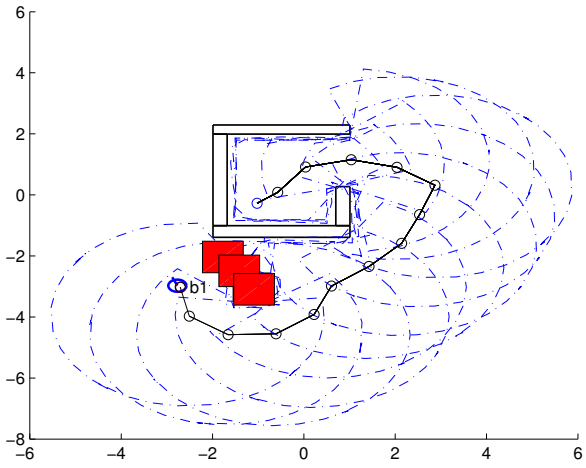

Figure 9. Displacement of the robot in simulation S_5 ($J = 30$, $H = 10$ e $R_c = 80\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

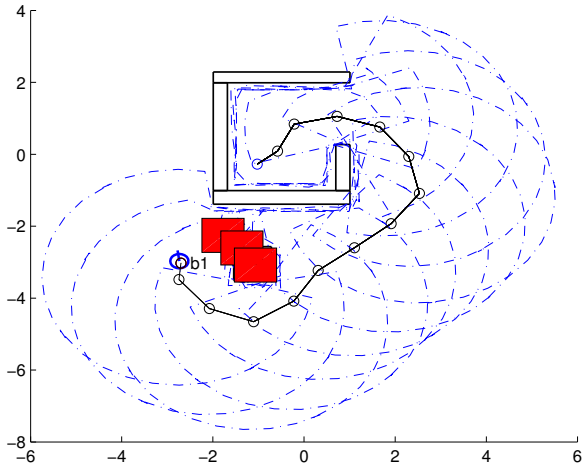

Figure 10. Displacement of the robot in simulation S_6 ($J = 10$, $H = 30$ e $R_c = 80\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

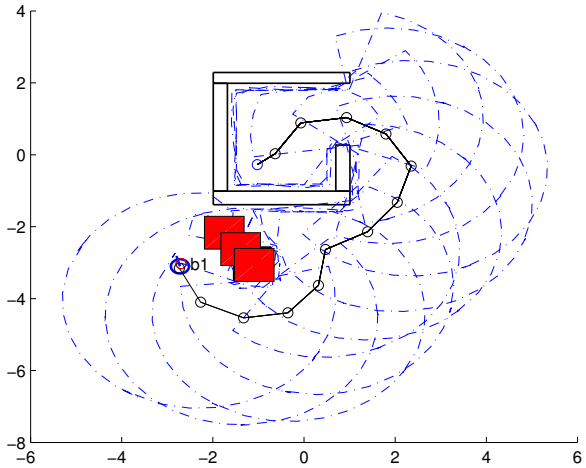

Figure 11. Displacement of the robot in simulation S_7 ($J = 30$, $H = 10$ e $R_c = 60\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

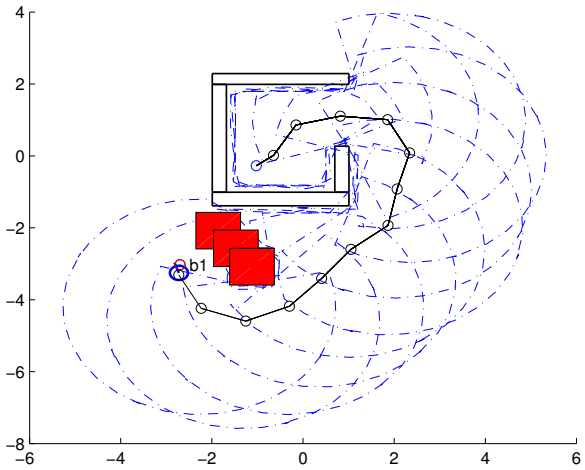

Figure 12. Displacement of the robot in simulation S_8 ($J = 10$, $H = 30$ e $R_c = 60\%$). The red circumference—labeled as b1—indicates the final point of the displacement.

3.1. Comparison with Other Approaches

To compare the results with other works in the literature, the DPNA-GA was simulated with environments used in works presented in Refs. [9,13,14,17,21]. Figures 13–16 show the displacement of the DPNA-GA in the environment proposed in Refs. [9,13,14] and [17,21], respectively. Table 4 shows the parameters and the results of the DPNA-GA in the environments shown in the Figures 13–16. Finally, Table 5 compares the result between the DPNA-GA and the literature works presented in Refs. [9,13,14,17]. Each simulation (Figures 13–16) was executed ten times, and the results are associated with the best cases. The genetic parameters for DPNA-GA were $J = 30$, $H = 10$ and $R_c = 80\%$, corresponding to the best configuration found in the simulations with static environment (see Table 2, S_2). The techniques compared were the Improved Genetic Algorithm (IGA) [9], the Matrix-Binary Codes-based Genetic Algorithm (MGA) [13], the Ant Colony Optimization (ACO) [14], the ACO with the Influence of Critical Obstacle (ACOIC) [14], the Firefly algorithm [17] and the Fuzzy System [21].

Table 4. Results of the simulations of environments shown in the Figures 13–16.

Environment		Length of Route (c_p)	Processing Time (t_p)	Displacement Events (M)	Time per Displacement Event (t_d)
Reference [9]	Figure 13	25.95	15.83 s	10	1.58 s/disp
Reference [13]	Figure 14	33.92	7.92 s	6	1.32 s/disp
Reference [14]	Figure 15	2119.94	8.39 s	5	1.68 s/disp
Reference [17,21]	Figure 16	69.93	10.51 s	7	1.50 s/disp

Table 5. Route length comparison between DPNA-GA and other navigation approaches.

Environment		Technique	Other Approaches		DPNA-GA	
			Length of the Best Route	Average Route Length	Length of the Best Route	Average Route Length
Reference [9]	Figure 13	IGA	22.83	22.83	25.05	25.95
Reference [13]	Figure 14	MGA	35.1	35.1	33.92	36.77
Reference [14]	Figure 15	ACO	2700	3513.3	2119.94	2720.74
		ACOIC		3446.7		
Reference [17]	Figure 16	Firefly	95	95	69.93	72.96
Reference [21]		Fuzzy	97.97	98.53		

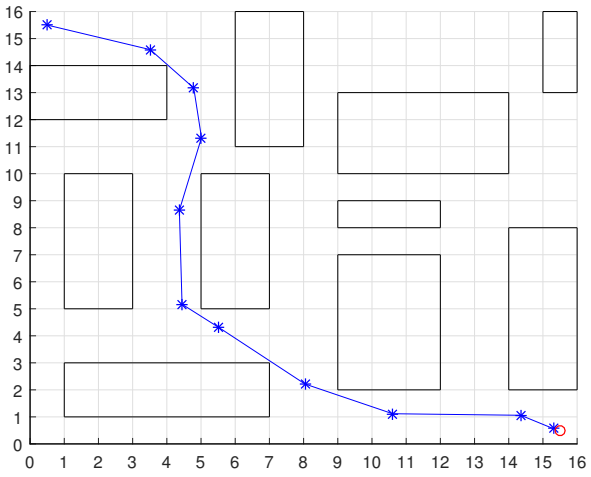

Figure 13. Displacement required for the robot using DPNA-GA in the environment proposed in Ref. [9]. The red circumference indicates the final point of the displacement.

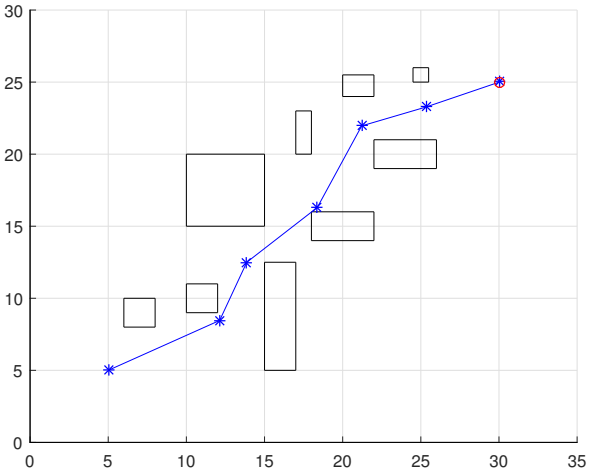

Figure 14. Displacement required for the robot using DPNA-GA in the environment proposed in Ref. [13]. The red circumference indicates the final point of the displacement.

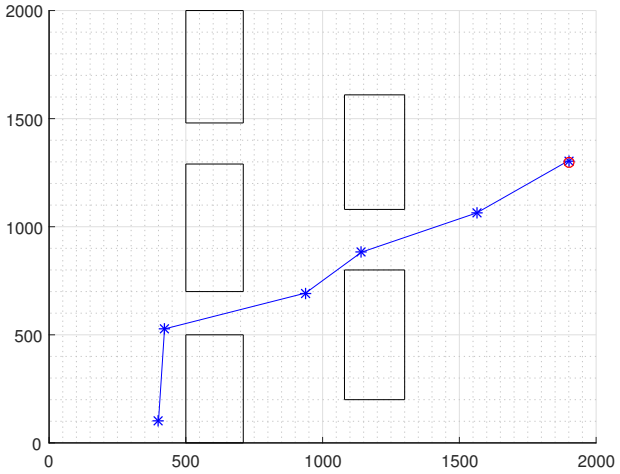

Figure 15. Displacement required for the robot using DPNA-GA in the environment proposed in Ref. [14]. The red circumference indicates the final point of the displacement.

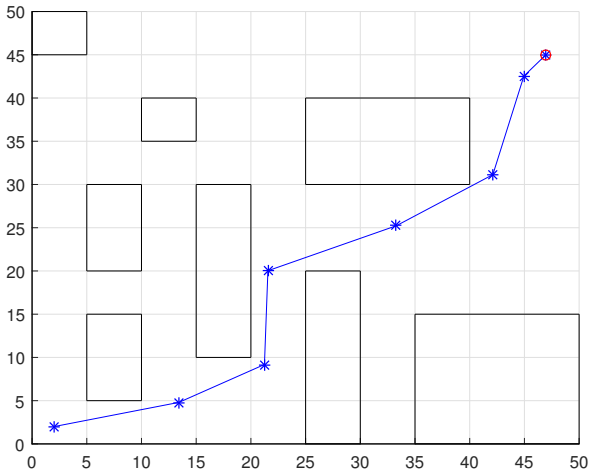

Figure 16. Displacement required for the robot using DPNA-GA in the environment proposed in Ref. [17]. The red circumference indicates the final point of the displacement.

The comparative results in Table 5 show that the DPNA-GA had a route length gain in most cases. Table 6 presents the route length saved by DPNA-GA for works [13,14,17,21]. For research presented in Ref. [9] the DPNA-GA had a slightly worse result (<10%) however, it is important to emphasize that the work shown in Ref. [9] uses a GA navigation strategies with global planning in which each individual (or chromosome) is coded as a possible routes between the initial and final points and this increase the chromosome size and GA processing. The DPNA-GA uses the fixed chromosome size regardless of route length.

Table 6. The route length, in %, saved by DPNA-GA.

Environment		Technique	Route Length Saved by DPNA-GA
Reference [13]	Figure 14	MGA	3.47%
Reference [14]	Figure 15	ACO ACOIC	27.36%
Reference [17] Reference [21]	Figure 16	Firefly Fuzzy	37.28% 40.00%

4. Conclusions

The objective of this work was to validate the robustness of a dynamic planning navigation technique for mobile terrestrial robots, based on genetic algorithms, denoted DPNA-GA. The validation was performed by varying some of the genetic parameters, in two different types of environment. Starting with strategies described in the literature as a basis, the DPNA-GA comprises a navigation scheme with local planning (applied to static and dynamic environments), in which the environment is unknown a priori and the size of the individuals is independent of the complexity of the environment. This property is fundamental from the point of view of practical implementation. The simulations showed that the DPNA-GA provided viable route solutions for different types of environment, following changes in the genetic parameters, hence demonstrating robustness at a relatively low cost, compared to other global and local planning strategies.

Author Contributions: All the authors have contributed in various degrees to ensure the quality of this work. (e.g., M.A.C.F. and Á.V.F.M.d.O. conceived the idea and experiments; M.A.C.F. and Á.V.F.M.d.O. designed and performed the experiments; M.A.C.F., C.A.D.S. and Á.V.F.M.d.O. analyzed the data; M.A.C.F. and C.A.D.S. wrote the paper).

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)—Finance Code 001.

Acknowledgments: The authors wish to acknowledge the financial support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for their financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ismail AL-Taharwa, A.S.; Al-Weshah, M. A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment. *J. Comput. Sci.* **2008**, *4*, 341–344. [CrossRef]
2. Shamsinejad, P.; Saraee, M.; Sheikholeslam, F. 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), Chengdu, China, 9–11 July 2010; Volume 8, pp. 115–120. [CrossRef]
3. Tuncer, A.; Yildirim, M. Dynamic path planning of mobile robots with improved genetic algorithm. *Comput. Electr. Eng.* **2012**, *38*, 1564–1572. [CrossRef]
4. Yongnian, Z.; Lifang, Z.; Yongping, L. An improved genetic algorithm for mobile robotic path planning. In Proceedings of the 2012 24th Chinese Control and Decision Conference (CCDC), Taiyuan, China, 23–25 May 2012; pp. 3255–3260. [CrossRef]
5. Zhang, L.; Min, H.; Wei, H.; Huang, H. Global path planning for mobile robot based on A* algorithm and genetic algorithm. In Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou, China, 11–14 December 2012; pp. 1795–1799. [CrossRef]
6. Samadi, M.; Othman, M.F. Global Path Planning for Autonomous Mobile Robot Using Genetic Algorithm. In Proceedings of the 2013 International Conference on Signal-Image Technology Internet-Based Systems (SITIS), Kyoto, Japan, 2–5 December 2013; pp. 726–730. [CrossRef]
7. Chaari, I.; Koubaa, A.; Bennaceur, H.; Trigui, S.; Al-Shalfan, K. smartPATH: A hybrid ACO-GA algorithm for robot path planning. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC), Brisbane, Australia, 10–15 June 2012; pp. 1–8. [CrossRef]
8. Wang, Y.; Zhou, H.; Wang, Y. *Mobile Robot Dynamic Path Planning Based on Improved Genetic Algorithm*; AIP Conference Proceedings; AIP Publishing: Melville, NY, USA, 2017; Volume 1864, p. 020046.
9. Zhang, X.; Zhao, Y.; Deng, N.; Guo, K. Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 91. [CrossRef]
10. Yun, S.C.; Parasuraman, S.; Ganapathy, V. Dynamic path planning algorithm in mobile robot navigation. In Proceedings of the 2011 IEEE Symposium on Industrial Electronics and Applications (ISIEA), Langkawi, Malaysia, 25–28 September 2011; pp. 364–369. [CrossRef]
11. Shi, P.; Cui, Y. Dynamic path planning for mobile robot based on genetic algorithm in unknown environment. In Proceedings of the 2010 Chinese Control and Decision Conference (CCDC), Xuzhou, China, 26–28 May 2010; pp. 4325–4329. [CrossRef]
12. Cosío, F.A.; neda, M.P.C. Autonomous robot navigation using adaptive potential fields. *Math. Comput. Model.* **2004**, *40*, 1141–1156. [CrossRef]
13. Patle, B.; Parhi, D.; Jagadeesh, A.; Kashyap, S.K. Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot. *Comput. Electr. Eng.* **2018**, *67*, 708–728. [CrossRef]
14. Han, J.; Park, H.; Seo, Y. Path planning for a mobile robot using ant colony optimization and the influence of critical obstacle. In Proceedings of the International Conference on Industrial Engineering and Operations Management Detroit, Southfield, MI, USA, 23–25 September 2016.
15. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. Pseudo-bacterial potential field based path planner for autonomous mobile robot navigation. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 81. [CrossRef]
16. Rajakumar, R.; Dhavachelvan, P.; Vengattaraman, T. A survey on nature inspired meta-heuristic algorithms with its domain specifications. In Proceedings of the International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 21–22 October 2016; pp. 1–6.
17. Patle, B.; Parhi, D.R.; Jagadeesh, A.; Kashyap, S.K. On firefly algorithm: Optimization and application in mobile robot navigation. *World J. Eng.* **2017**, *14*, 65–76. [CrossRef]
18. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]

19. Algabri, M.; Mathkour, H.; Ramdane, H.; Alsulaiman, M. Comparative study of soft computing techniques for mobile robot navigation in an unknown environment. *Comput. Hum. Behav.* **2015**, *50*, 42–56. [CrossRef]
20. Mohanty, P.K.; Sah, A.K.; Kumar, V.; Kundu, S. Application of Deep Q-Learning for Wheel Mobile Robot Navigation. In Proceedings of the 2017 3rd International Conference on Computational Intelligence and Networks (CINE), Odisha, India, 28 October 2017; pp. 88–93.
21. Garcia, M.P.; Montiel, O.; Castillo, O.; Sepúlveda, R.; Melin, P. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Appl. Soft Comput.* **2009**, *9*, 1102–1110. [CrossRef]
22. de Oliveira, A.V.F.M.; Fernandes, M.A.C. Dynamic planning navigation strategy for mobile terrestrial robots. *Robotica* **2016**, *34*, 568–583. [CrossRef]
23. de Oliveira, A.V.F.M.; Fernandes, M.A.C. Dynamic Planning Navigation Technique Optimized with Genetic Algorithm. In Proceedings of the 2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol, Sao Carlos, Brazil, 18–23 October 2014; pp. 91–96. [CrossRef]
24. Siegwart, R.; Nourbakhsh, I.; Scaramuzza, D. *Introduction to Autonomous Mobile Robots*; Intelligent Robotics and Autonomous Agents; MIT Press: Cambridge, MA, USA, 2011.
25. Esposito, J.M.; Barton, O.; Kohler, J. Matlab Toolbox for the iRobot Create. Available online: https://www.usna.edu/Users/weapron/esposito/_files/roomba.matlab/ (accessed on 30 November 2018).
26. Cameron Salzberger, K.Y.D.F.; Kress-Gazit, H. MATLAB-Based Simulator for the iRobot Create. Available online: http://sourceforge.net/projects/createsim/files/Toolbox/ (accessed on 30 November 2018).