mathematics

MDPI

Article

Path-Planning for Mobile Robots Using a Novel Variable-Length Differential Evolution Variant

Alejandro Rodríguez-Molina [1,*], José Solís-Romero [1], Miguel Gabriel Villarreal-Cervantes [2,*], Omar Serrano-Pérez [2] and Geovanni Flores-Caballero [2]

[1] Tecnológico Nacional de México/IT de Tlalnepantla, Research and Postgraduate Division, Estado de México 54070, Mexico; jose.sr1@tlalnepantla.tecnm.mx
[2] Postgraduate Department, Instituto Politécnico Nacional, CIDETEC, Mexico City 07700, Mexico; omarserrano95@gmail.com (O.S.-P.); gfloresc1800@alumno.ipn.mx (G.F.-C.)
* Correspondence: alejandro.rm@tlalnepantla.tecnm.mx (A.R.-M.); mvillarrealc@ipn.mx (M.G.V.-C.)

Abstract: Mobile robots are currently exploited in various applications to enhance efficiency and reduce risks in hard activities for humans. The high autonomy in those systems is strongly related to the path-planning task. The path-planning problem is complex and requires in its formulation the adjustment of path elements that take the mobile robot from a start point to a target one at the lowest cost. Nevertheless, the identity or the number of the path elements to be adjusted is unknown; therefore, the human decision is necessary to determine this information reducing autonomy. Due to the above, this work conceives the path-planning as a Variable-Length-Vector optimization problem (VLV-OP) where both the number of variables (path elements) and their values must be determined. For this, a novel variant of Differential Evolution for Variable-Length-Vector optimization named VLV-DE is proposed to handle the path-planning VLV-OP for mobile robots. VLV-DE uses a population with solution vectors of different sizes adapted through a normalization procedure to allow interactions and determine the alternatives that better fit the problem. The effectiveness of this proposal is shown through the solution of the path-planning problem in complex scenarios. The results are contrasted with the well-known A* and the RRT*-Smart path-planning methods.

Keywords: Variable-Length-Vector optimization; Differential Evolution; mobile robots; path-planning

Citation: Rodríguez-Molina, A.; Solís-Romero, J.; Villarreal-Cervantes, M.G.; Serrano-Pérez, O.; Flores-Caballero, G. Path-Planning for Mobile Robots Using a Novel Variable-Length Differential Evolution Variant. *Mathematics* **2021**, *9*, 357. https://dx.doi.org/10.3390/math9040357

Academic Editor: Gustavo Santos-García
Received: 21 January 2021
Accepted: 7 February 2021
Published: 11 February 2021

1. Introduction

1.1. A Review of Path-Planning Methods

Robots have become fundamental tools to perform diverse activities, mainly those that are dangerous or tough for humans [1]. Particularly, mobile robots extend the operation of classical fixed-base robotic manipulators to almost any place [2].

The success of mobile robots depends on their autonomy degree, i.e., insofar as they require less human intervention [3]. Among the tasks required to achieve autonomy in robots, path-planning is one of the most critical [4].

Without loss of generality, the path-planing problem is to find a sequence of path elements (points, curves, line segments, etc.) that connect a start point to a target one avoiding threats (usually, collisions with obstacles or risky regions) and achieving the best performance (typically, the shortest length) [5]. Since the path's shape is strongly related to the optimization of energy consumption, the robot's safety, navigation time and the path's overall cost [6], the relevance of the path-planning task is highlighted.

In the last decades, several path-planning solution approaches, also named path-planners, have been proposed and can be classified in five categories [7]: (i) sampling-based algorithms, (ii) node-based optimal algorithms, (iii) mathematical-model-based algorithms, (iv) bio-inspired algorithms and (v) multifusion-based algorithms.

The path-planners in (i) randomly or heuristically sample the path elements to find a feasible and short path. Within this category, the Rapidly-exploring Random Trees

(RRT) [8], and the Probabilistic Road Maps (PRM) [9] stand out. In RRT, random points in the workspace are incrementally sampled in a radius to generate branched feasible path segments. Therefore, the resulting tree contains the searched path. On the other hand, the PRM samples random nodes in a radius within the workspace and creates collision-free connections among them. After this, PRM selects a feasible path between the start and target nodes. Despite the rapid exploration of the workspace with RRT and PRM, the probability of finding the shortest path is extremely low [10]. Improved RRT and PRM versions like RRT* and PRM* have been proposed to obtain the shortest possible path by taking into account each branch length and scaling the sampling radius gradually, respectively [11]. The main shortcoming of the methods in (i) is that calculating the best path depends on probability, while their main strength is the computational efficiency. Moreover, a human intervention must establish the proper sampling radius and the number of sampled points or branches, affecting the robot autonomy. If the above parameters are settling incorrectly, it may result in a very low-performance path or excessive calculation time.

Category (ii) includes techniques that require the workspace to be described in a graph. These techniques then explore the graph to find the shortest path between the start and target nodes, usually as variants of the Dijkstra algorithm. Four representative methods are the A* [12], the Fast Marching (FM) [13], the Potential Field (PF) [14] and the Visibility Graphs (VG) [15]. A* is a greedy and deterministic technique that heuristically finds sub-optimal paths in informed graphs. A well-known variant of A* is the D* that deals with dynamical workspaces [16]. FM solves a discretized version of the Eikonal wave propagation differential equation in a graph. The solution to this equation is related to the wave's arrival time at each node in the graph from a given origin node. The propagation of the wave generates a velocity field at each node used to calculate the arrival times to its neighbors. The best route is the one that takes the wave from the origin to the destination in the shortest cumulative time, and is obtained by the gradient descent. The PF method applies a potential field to the workspace. This field generates an attractive force between any node in the graph and the target, and induces a repulsive effect to the obstacles or risky regions. Based on the above two effects, a potential can be calculated for each node. Then, the best path can be found as the lowest cumulative potential alternative through the gradient descent. In VG, the vertices of each obstacle in the workspace are selected as the nodes of a graph, and the connections among them are established based on visibility, i.e., a collision-free edge between each pair of nodes. The best path is obtained from that graph. Since all the above methods discretize the workspace, their effectiveness depends on how the graph is built. The most accurate workspace representation requires an infinite graph of which the best path cannot be found in finite time. In practice, the human establishes a proper trade-off between planning precision and computational cost. If each node in the graph represents a wide region of the workspace, the obtained path may be very long, but the calculation could be fast. The opposite can happen if each node is associated with a smaller region.

The methods in (iii) require an accurate mathematical model of the workspace and the mobile robot. Based on that model, a linear or non-linear dynamic programming problem is established to find the control actions that take the robot through the best feasible path. The above problem, also known as the optimal control problem, is typically solved by classical direct or indirect methods. The work in [17] shows a clear example of this category. In that proposal, a dynamic control problem is solved in the path-planning for a spherical mobile robot. Although the methods in (iii) can achieve the lower cost path theoretically, their performance is limited by model accuracy. All models are imprecise, and the above coupled with the presence of uncertainties and disturbances [18] in the real-world can lead to poor performance in path-planning.

The bio-inspired approaches in (iv) use analogies with biological or physical phenomena to solve the path-planning problem. In this sense, Artificial Intelligence (AI) techniques such as Fuzzy Logic (FL), Neural Networks (NNs) and meta-heuristics are recurrently found in the specialized literature. In [19], the Ant Colony Optimization (ACO) algorithm

improved with FL is used to obtain cost-effective paths with an unmanned aerial vehicle in dynamic environments. Another example is the work in [20], where a NN is used to dynamically predict and avoid collisions in the path-planning for mobile robots. On the other side, the meta-heuristic approaches are referred to as algorithms from evolutionary computation and swarm intelligence that solve a formal optimization problem to find the path elements that minimize its overall cost. These methods are discussed further in this document. The methods in (iv) have shown outstanding performance, especially when the path-planning problem is hard, e.g., when there are moving obstacles or the path cost involves other criteria beyond the overall length such as energy, time, shape, etc. Nevertheless, the computational cost of these methods can be considered higher than the previous categories. Particularly, the FL and NN methods need proper training to generalize an expert human's knowledge before its use.

Finally, the techniques in (v) combine different features of the other four categories. The above aims to enhance the path-planning effectiveness by decreasing the implicit drawbacks of a single category. An example of these techniques is found in [21]. In the above research, an NN improves the RRT* algorithm by determining a suitable sampling distribution.

1.2. Optimization in Path-Planning

Optimization is required in almost every engineering application due to the current limitations and challenges of modern life, and path-planning is not an exception. It is used either to reduce costs [22], improve accuracy [23], raise the incomes [24] or enhance other performance indicators. The systematic procedure in the general optimization process for engineering is noticeable. First, the system or process is studied in-depth to determine the relationships among the involved variables (design variables), the performance needs (objectives) and the overall limitations (constraints). The above elements are established, through the mathematical language, as a formal optimization problem. Finally, an optimizer is used to find suitable solutions that can be implanted in the final application. Therefore, the path-planning problem can be formally stated as an optimization problem, where the path requirements can be expressed as the objectives, the path elements are the design variables and the constraints can be set to prevent risky behaviors [25]. By its own nature, the path-planning optimization problem is Non-Polynomial Hard (NP-Hard) one [26], and deterministic optimization techniques may not achieve the best performance. Fortunately, the use of soft computing techniques such as meta-heuristics has shown outstanding results compared to deterministic solution methods and state-of-the-art sampling and node-based approaches [25,27]. In this sense, meta-heuristics are stochastic computational techniques that can find good solutions to complex optimization problems using a reasonable amount of resources [28]. Those based on evolutionary computation and swarm intelligence stand out among meta-heuristic methods.

On the other hand, for a wide variety of optimization problems associated with real-world applications, the engineer knows the variables involved in the process or system, which must be adjusted to optimize the outcome. Nevertheless, there are more intricate problems where the identity or the number of the involved variables is uncertain [29]. Examples of the above problems are observed in controller design [30], re-configurable robotics [31], machine scheduling [32], among others.

Path-planning for mobile robots is part of the problems described in the previous paragraph since the number of path elements cannot be known a priori. A common way to address this issue requires human intervention to propose or decide the number of path elements (points in the Cartesian plane) to be adjusted. For example, an improvement of the Particle Swarm Optimization (PSO) is proposed in [33] to optimize a mobile robot's trajectory. PSO can place the points that the robot travels in each iteration, solving an optimization problem that considers the distance to the target point and the obstacles' separation. The proposal in [34] introduces a new crossing operator in the GA to optimize the location of the points that make up a trajectory for mobile robots. GA uses a cost function that considers the total distance, the robot's energy and the safety of the path.

In [35], a method is proposed to find the point local positions of a trajectory sequentially (points that the robot travels from an initial location to a final one) through the Differential Evolution (DE) algorithm. The method finds a new feasible point position on each iteration to minimize the distance to an end point. The work in [36] presents a hybridization between PSO and DE to solve the problem of trajectory planning for mobile robots. The problem aims to minimize a single target that considers the total distance and the trajectory's safety. The trajectory is made up of consecutive points located over vertical lines of action within the Cartesian space. The research in [37] proposes a hybridization between a GA and the A* planning technique to find the spline segments of a trajectory for the differential wheeled robot. An optimization problem is formulated that weights four objectives, the distance, the travel time, the smoothness and the safety of the trajectory (based on how close the mobile robot passes to the obstacles).

It can be observed that the main drawback of the above approach is the reduction of autonomy in mobile robots, i.e., mobile robots cannot decide the number of path elements by themselves and the human help is needed. Additionally, if the number of path elements is not well established, a large value can over-define the path (this can complicate the post-processing task aspects [38] such as path smoothing [39], control [40]) and energetic efficiency [41], while a lower one can reduce the chances of finding the optimal path [42]. A way to deal with the above difficulty is to treat the path-planning as a Variable-Length-Vector Optimization Problem (VLV-OP) to determine the number of the variables (points) and their values (Cartesian coordinates) enhancing the application performance. The above implies an additional complexity in the problem, and its solution requires more sophisticated optimization mechanisms, even in the meta-heuristics.

To the best of the author's knowledge, the path-planning as a VLV-OP has been investigated little. In this way, two representative works [43,44] can be found in the specialized literature. Both alternatives take advantage of the GA's binary encoding to deal with the VLV-OP in discrete workspaces modeled as finite grids. In [43], the GA with a simple one-point crossover, a swap mutation and a gene elimination mechanism is adopted to evolve the generation of mobile robots' paths. On the other hand, the work in [44] finds optimal trajectories for a robotic manipulator using the GA. The adopted one-point crossover uses information regarding nodes' location in two paths to select the split point. Simultaneously, the mutation strategy alters random points on the path based on the location of its neighbors.

Although only a couple of works address the path-planning problem as a VLV-OP, different solution approaches can be found from other application contexts. For example, GAs are also proposed to solve VLV-OPs related to the preventive maintenance scheduling problem [45], the headway optimization and the scheduling combination problem for vehicles [46] and the routing problem for informed graphs [47,48]. Those works require fewer modifications of the GAs to solve VLV-OPs, and their effectiveness is limited to discrete search spaces.In the case of path-planning problems, discrete workspaces reduce the accuracy of the found paths in the sense that optimizers cannot find the optimal solution [25]. However, variations of well-known meta-heuristic optimizers such as DE and PSO have been proposed to solve VLV-OPs in engineering with continuous search spaces. Related to DE, the work in [49] proposes a DE variant to solve the satellite reconfiguration problem using a fixed-length vector alongside an expression vector that determines the activation of a chromosome. A similar approach is found in [50], where DE was used for clustering. In this DE variant, the chromosome is divided into activation genes (threshold to decide which centroid is active) and genes that include the cluster centroids. Since the vector size remains fixed for all solutions, classical DE operators are used by their own nature. In the operational amplifier design, circuit components are represented by blocks with a variable number of genes into a fixed-length vector, where unusable genes are set to zero in [51]. To allow the use of classical DE operators, truncation and expansion procedures are adopted to standardize chromosomes' size. In PSO to solve VLV-OPs, some applications with continuous search spaces have also been developed.

In [52], the maintenance scheduling problem for a pressurized water reactor is solved by a PSO variant. The particles represent variable groups of interventions on maintenance days. The particle information is included in a fixed-length vector, where a variable equal to or lower than zero denotes a state of non-activation. Then, the traditional PSO operators can be applied without additional considerations. Relevant applications of PSO for VLV-OPs are also related to the optimization of neural networks. In [53], PSO optimizes the layers of convolutional neural networks. It uses a fixed-length vector with binary encoding and a range of variable values to represent a disabled state of a layer. On the other hand, the work published in [54] uses PSO to automatically select the number of hidden neurons, the input weights and the hidden biases in a single hidden layer feedforward. A stochastic strategy was proposed to update particle sizes and apply the original PSO operators over actual variable-length solutions to achieve it. In the case of DE and PSO to solve VLV-OPs in engineering with continuous search spaces, most of the alternatives found in literature use a fixed-vector to represent individuals or particles and strategies to distinguish the active variables (i.e., those variables that are considered in the problem evaluation). For techniques with actual variable-length-vectors, the previously mentioned strategies aim to stochastically standardize the size of the vectors to apply traditional meta-heuristic operators. However, the above ways to handle VLV-OPs with continuous search spaces may not provide relevant information for the search, i.e., they do not consider the variable values of the current solution alternatives for the standardization.

1.3. Scope and Contributions of the Proposal

In this work, the path-planning for mobile robots is conceived as a VLV-OP where the number of path points and their locations must be determined simultaneously to minimize the overall path length. The VLV-OP is established considering the features of a four mecanum wheeled omnidirectional mobile robot and can be easily scaled to other kinds of mobile systems.

The VLV-OP aims to adjust a variable number of sequential path points in a continuous workspace to minimize the total length between the start and target points. In order to avoid collisions with obstacles, a hard equality constraint is imposed on the VLV-OP. In this way, a path is assigned to a simple 2D shape to ensure the robot safety in the path tracking, and obstacles are treated as polygons to support the collision detection. The above VLV-OP is then solved by a novel variant of Differential Evolution for Variable-Length-Vector optimization (VLV-DE). This optimizer uses an initial VLV population generated by the weighted variant of the A* path-planning method [55] to speed up the finding of the first set of feasible paths. The candidate solution vectors of different sizes are then adapted through a normalization procedure to allow interactions with the ordinary DE operators, i.e., scaling and recombination. This normalization can increase or decrease the vector length by incorporating valuable solution information from the current vector variables. The above helps to generate improved paths that better fit the problem.

For instance, the proposal's effectiveness is shown through the solution of the path-planning problem in three complex scenarios with different obstacle distributions and shapes. The results are contrasted with those of the original A* method and the RRT*-Smart technique.

The rest of the present work is organized as follows. Section 2 presents the general VLV-OP and introduces the path-planning problem for mobile robots. Section 3 details the operation of the proposed VLV-DE. The experiment details and results are described in Section 4 to show the effectiveness of the proposal. Finally, Section 5 draws some conclusions.

2. The Path-Planning Problem

2.1. General Variable-Length-Vector Optimization Problem

The general Variable-Length-Vector Optimization Problem (VLV-OP) shown in (1), is to find a VLV x of design variables (variables involved in a process or system) that minimizes the value of the objective function J (an indicator of the process or system

performance). Unlike classical global optimization problems, where only the value of the design variables in x is searched, the VLV-OP additionally requires the number of design variables to be found, i.e., the size of x, denoted by $s(x)$, is unknown. Moreover, the search space can be delimited by a set of n_g inequality constraints $g_i(x)$ or n_h equality constraints $h_j(x)$ (both usually describe the process or system limitations regarding the cost, space, resources, to name just a few) that describe the feasible search space $\Omega \in \mathbb{R}^{s(x)}$. In addition, the box constraints described by x_k^{min} and x_k^{max}, can be considered to limit the possible values that each design variable x_k can take, while the constraints over $s(x)$ control the size of the vector x.

$$\begin{gathered} \mathbf{min}_{x\in\Omega}\ J(x) \\ \textit{subject to:} \\ g_i(x) \leq 0, i = 1, \ldots, n_g \\ h_j(x) = 0, j = 1, \ldots, n_h \\ x_k^{min} \leq x_k \leq x_k^{max}, k = 1, \ldots, s(x) \\ L_{min} \leq s(x) \leq L_{max} \end{gathered} \tag{1}$$

2.2. The Path-Planning for Mobile Robots as a Variable-Length-Vector Optimization Problem

The path-planing problem for mobile robots is to find the shortest way (a set of consecutive control points p_l) from a start point p_s to a target one p_t within the workspace W. The workspace W is considered as a finite subspace of $\mathbb{R}^2$. In this case, W is related to the plane's coordinates where the mobile robot is placed. Additionally, W includes a set of obstacles or threats. Hence, the obtained path must somehow ensure collision avoidance between the mobile robot and the obstacle geometries.

In this work, the four mecanum wheeled mobile robot (4MWMR) is used to illustrate the development of the path-planning problem. Nevertheless, it can be intuitively extrapolated to other mobile configurations.

The 4MWMR is an omnidirectional mobile robot whose wheels are independently controlled by DC motors. Each wheel includes a single hub circumference, which is circumferentially coupled with a series of rollers oriented at 45° concerning it. A remarkable feature of the 4MWMR is its capability to track nonlinear paths by keeping a constant orientation [56].

The above path-planning problem is put in the form of the general variable-length-vector optimization problem in (1), and its elements are detailed as follows:

2.2.1. The Variable-Length-Vector of Design Variables

The variable-length-vector x in (2) includes the information (2D Cartesian coordinates) of the control points $p_l, l = 1, \ldots, s(x)$ in the path, i.e., the ordered sequence of positional configurations that the mobile robot must track to reach p_t from p_s.

$$x = [p_1, \ldots, p_{s(x)}]^T \tag{2}$$

2.2.2. The Objective Function

The objective function J in (3) measures the total path length, i.e., the sum of the Euclidean distance (denoted by d_e) of each pair of consecutive points in the path (start point p_s, target point p_t and control points p_l, included) as observed in Figure 1.

$$J(x) = d_e(p_s, p_1) + \sum_{l=1}^{s(x)-1} (d_e(p_l, p_{l+1})) + d_e(p_d, p_t) \tag{3}$$

2.2.3. The Constraints

The studied path-planning problem does not consider any inequality constraint but only one equality constraint. The latter is established to strictly avoid the collisions of the mobile robot geometry with the obstacles in W.

In practice, objects of complex geometries are represented by simpler shapes (e.g., boxes, circles, polygons, etc.). The use of these shapes notably improves the computational algorithms' performance for collision detection [57]. For the above reason, 2D shapes were assigned to all the elements in W, as shown in Figure 1. These shapes are described next:

- **Obstacles:** Polygons $P_m, m = 1, \ldots, n_o$, defined by an arbitrary number of vertices which describe all the n_o obstacles in the workspace W.
- **Path points:** Circles $C_l, l = 1, \ldots, s(x)$ are assigned to each control point in the path. Each circle is centered in the control point and has a diameter of $2r$ that matches the length of the longest diagonal of the 4MWMR in the plane. This shape is selected to allow possible orientation changes of the 4MWMR.
- **Path edges:** In order to improve the collision detection procedure, rectangular geometries $R_l, l = 1, \ldots, s(x) + 1$ are attached to each edge in the path. An edge is the line segment between a pair of consecutive points in the path. The width of each R_l is equal to the edge length, its height is $2r$ and its orientation matches the inclination $\alpha, l = 1, \ldots, s(x) + 1$ of the edge. The R_l is centered in the middle of the two points.

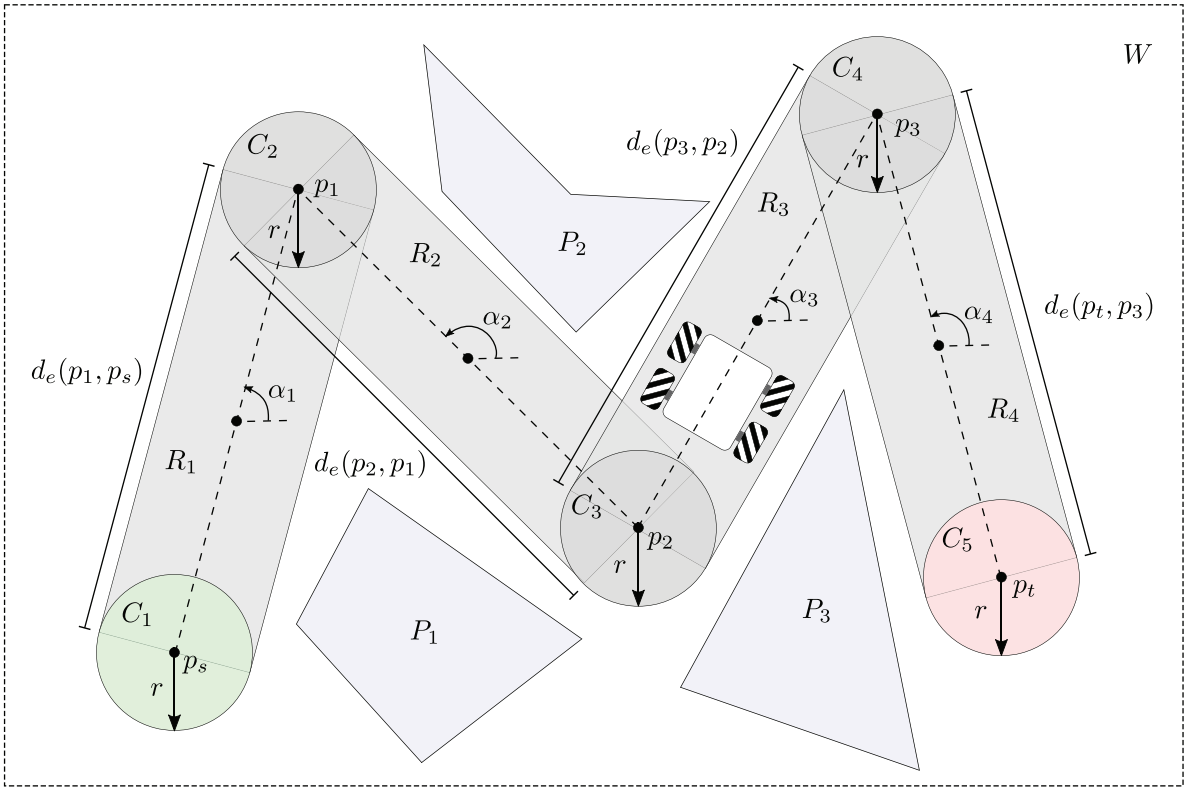


Figure 1. Example path computed for a workspace W with several obstacles.

Once the elements in W are assigned to a 2D shape, it is possible to determine if the path geometry (this includes the C_l and R_l geometries) collides with the ones of the obstacles (the polygons P_m). For this reason, the function (4) implements a polygon-polygon and a polygon-circle collision detection based on the Separating Axis Theorem (SAT) [58,59].

Based on (4), the single equality constraint for the path-planning problem is established in (5), which indicates the times the elements on the path collide with the obstacles in W.

$$c(G_1, G_2) = \begin{cases} 1, & \textbf{if } G_1 \text{ collides } G_2 \\ 0, & \textbf{otherwise} \end{cases} \tag{4}$$

$$h(x) = \sum_{m=1}^{n_o} \sum_{l=1}^{s(x)} c(P_m, C_l) + \sum_{m=1}^{n_o} \sum_{l=1}^{s(x)+1} c(P_m, R_l) \tag{5}$$

Additionally, all the control points p_l must be inside W. The above conform to the box constraints of the problem. It is important to remark that the VLV-OP for path-planning has particular features distinguishing it from other specialized literature problems. In this sense, all variables are of the same type (2D points), and their position within the VLV of design variables is relevant. The above two features are taken into account while designing the novel optimizer in the next section.

3. The Novel Variable-Length-Vector Differential Evolution

Differential Evolution (DE) is a well-known evolutionary algorithm that has been successfully used to find outstanding solutions to global optimization problems [60]. Moreover, DE has proven to be particularly effective when these optimization problems are related to engineering applications [61].

DE uses an evolutionary procedure over a population (a set of candidate solutions) to explore (i.e., to find potentially good search regions) and exploit (i.e., to obtain outstanding solutions within a search region) the search space towards the global solution [62]. The individuals in the population are initially random, and during a given number of generations, they are recombined and mutated to generate an offspring population. By the end of each generation, the original and offspring populations' individuals compete to determine those solutions that survive for the next generation. The best solution is found in the population of the last generation. Several variants of DE have been proposed over time, and they differ in the type of used evolutionary operators that improve its explorative and exploitative capacity [63].

The proposed Variable-Length-Vector Differential Evolution (VLV-DE) introduces a novel normalization procedure that allows the interaction of variable-length solutions through the original DE evolutionary operators (mutation, crossover and selection designed for searching in continuous spaces) to solve the VLV-OP associated with the path-planning task for mobile robots.

The above normalization aims to incorporate as much relevant information as possible of a given solution (candidate path) to increase or decrease its number of variables (path nodes).

Unlike the optimizers for continuous VLV-OPs found in the specialized literature, the proposed VLV-DE handles actual variable-length solutions. The above implies the representation of solutions through the use of actual variable-length-vectors.

The operation of the proposed VLV-DE is described in Algorithm 1 and is based on the DE/best/1/bin variant [63]. Nevertheless, it can be easily extended to other DE variants.

The elements of VLV-DE are detailed as follows:

Algorithm 1: Variable-Length-Vector Differential Evolution (VLV-DE)

Input: Scaling factor (F), Crossover rate (CR)
Output: Best solution (x_b).

```
Generate new population X randomly in the search space
Evaluate individuals in X
G ← 1
while G ≤ G_max do
    Get best individual x_b from X
    foreach x_i ∈ X do
        Get two random individuals x_{r1} and x_{r2} from X such that i ≠ r1 ≠ r2 ≠ b
        Select N_s randomly from {s(x_i), s(x_{r1}), s(x_{r2}), s(x_b)}
        Variate N_s
        x_k^n ←Normalize(x_k, N_s), k = i, r1, r2, b
        Generate a mutant individual v_i using (6)
        Generate an offspring individual u_i using (7).
        Evaluate u_i
    Select individuals that conform X for G + 1
    foreach x_i ∈ X do
        x_i ←LocalSearch(x_i)
    G ← G + 1
Get best individual x_b from X
```

3.1. Initialization

In the first step, as in any DE variant, NP individuals in the population X are randomly generated within the search space (the box constraints given by x_k^{min} and x_k^{max} in (1) delimit this space).

Due to the complexity of the path-planning problem, it is improbable to generate a feasible solution during the random initialization. Depending on the workspace conditions and the problem complexity (i.e., the number, location and size of obstacles), it could take several generations to find a feasible solution through the evolutionary process, and even it may never be found.

To address the above difficulty, the weighted A* search algorithm [55] is used to generate a feasible initial population in VLV-DE with different variable-length paths. The weighted variant of A* establishes a trade-off between search performance and computational cost through the weighting factor w [64]. With the larger w, the performance increases, as does the cost, while the opposite happens when w is decremented. The adopted A* uses a grid discretization of the workspace with a cell width and height equal to $2r$ (to fit the mobile robot dimension), the Manhattan heuristic model [65], the cost function $f(n) = 1 + h(n) + w(g(n) - 1)$ and a uniformly distributed weight w in $[0, 1]$ to generate different sub-optimal paths.

Once X is generated, the fitness of each individual is determined by calculating the values J, g_i and h_j in (1).

3.2. Evolutionary Cycle

During a maximum of G_{max} generations, the individuals in X interact to generate mutants. Then, these mutants are recombined with the original individuals in X to obtain offsprings. Finally, the offsprings compete with the individuals in X based on their fitness, and the fittest alternatives persist in X for the next generation.

3.3. Normalization

DE/best/1/bin requires three different individuals x_{r_1}, x_{r_2} and x_b, to generate a single mutant v_i. The individuals x_{r_1} and x_{r_2} are randomly selected from the current population X for each solution x_i. On the other hand, x_b is selected as the fittest solution in X at the

start of the current generation. Moreover, DE/best/1/bin uses the mutant v_i along with an original individual x_i to create an offspring u_i.

Computation of both individuals v_i and u_i is performed through evolutionary operators that require the base solutions (i.e., x_i, x_{r_1}, x_{r_2} and x_b) to be the same size.

Nevertheless, VLV-DE conceives a population X with individuals of different sizes to solve the VLV-OP in (1), and a normalization procedure needs to be performed to allow interactions by applying the original evolutionary operators of DE/best/1/bin (mutation and crossover).

The normalization procedure, described in Algorithm 2, temporarily adjusts the size of an arbitrary solution to a given value N_s. The value of N_s is first randomly selected from $\{s(x_i), s(x_{r_1}), s(x_{r_2}), s(x_b)\}$ to favor the similarity of the generated mutant or offspring with at least one of the involved solutions, i.e., to enhance the exploitation capability of VLV-DE regarding the vector size. Next, the exploration is improved by applying a variation operator to N_s. The Gaussian mutation [66] is adopted for this purpose.

If the size of a given solution is greater than N_s, its variables must be compressed to obtain an individual of reduced size. On the contrary, if this size is smaller than N_s, a decompression procedure is used to increase the original individual size.

Algorithm 2: Normalize(x_k, N_s)

Input: x_k (arbitrary solution), N_s (normalization size)
Output: x_k^n (normalized solution)

```
if s(x_k) ≠ N_s then
    if s(x_k) < N_s then
        x_k^n ← Decompress(x_k, N_s)
    else
        x_k^n ← Compress(x_k, N_s)
else
    x_k^n ← x_k
```

3.4. Compression

This procedure is observed in Algorithm 3 and aims to iteratively reduce the size of an arbitrary individual while losing as little path information as possible. For each iteration, the compression procedure can reduce the size of the individual x_k in two ways based on probability:

- Selecting a pair of consecutive path points and replacing them with a new single one that preserves information from both. The new variable is obtained as the pair's mean and can be altered through a variation operator (the polynomial mutation [67]) to enhance the exploitative search.
- Removing the first or the last path point.

The compression stops once the individual size reaches N_s.

3.5. Decompression

Unlike compression, the procedure described in Algorithm 4 aims to iteratively include new valuable variables in the individual to increase its size. As long as the individual size is different from N_s, the decompression procedure can increase the size of the individual x_k in two ways based on probability:

- Selecting a pair of consecutive path points to generate a single one based on their information, which is then inserted in the middle of both. This new variable is calculated as the pair's mean and can be modified by using a variation operator (the polynomial mutation) to improve the exploratory search.
- Including altered versions of the first or the last path points (obtained through the polynomial mutation), at the beginning or the end of the solution, respectively.

Algorithm 3: Compress(x_k, N_s)

Input: x_k (arbitrary solution), N_s (normalization size)
Output: x_k^c (compressed solution)

$x_k^c \leftarrow x_k$
while $s(x_k^c) \neq N_s$ **do**
  $j \leftarrow randi(0, s(x_k^c))$
  if $j \neq 0$ ***and*** $j \neq s(x_k^d)$ **then**
    Compute mean variable $\bar{x}_{k,j} \leftarrow \frac{x^c_{k,j+1} - x^c_{k,j}}{2}$
    Variate $\bar{x}_{k,j}$ using Polynomial Mutation
    $x^c_{k,j} \leftarrow \bar{x}_{k,j}$
    Remove $x_{k,j+1}$ from x_k^c
  else
    if $j = 0$ **then**
      Remove $x_{k,1}$ from x_k^c
    else
      Remove $x_{k,s(x_k^c)}$ from x_k^c

Algorithm 4: Decompress(x_k, N_s)

Input: x_k (arbitrary solution), N_s (normalization size)
Output: x_k^d (decompressed solution)

$x_k^d \leftarrow x_k$
if $s(x_k^c) = 1$ **then**
  Insert random variable in x_k^d
  Shuffle x_k^d
while $s(x_k^d) \neq N_s$ **do**
  $j \leftarrow randi(0, s(x_k^d))$
  if $j \neq 0$ ***and*** $j \neq s(x_k^d)$ **then**
    Compute mean variable $\bar{x}_{k,j} \leftarrow \frac{x^d_{k,j+1} - x^d_{k,j}}{2}$
  else
    if $j = 0$ **then**
      $\bar{x}_{k,j} \leftarrow x^d_{k,1}$
    else
      $\bar{x}_{k,j} \leftarrow x^d_{k,s(x_k^d)}$
  Variate $\bar{x}_{k,j}$ using Polynomial Mutation
  Insert $\bar{x}_{k,j}$ in x_k^d after $x^d_{k,j}$

3.6. Evolutionary Operators

At this point, all the individuals required for mutation and crossover have acquired the temporarily forms x_i^n, $x_{r_1}^n$, $x_{r_2}^n$ and x_b^n, which have the same size. Then, the evolutionary operators of the original variant DE/best/1/bin can be applied as usual to generate interactions between individuals. These operators include the mutation in (6), with the scaling factor $F \in [0, 1]$, and the crossover in (7), with the crossover rate $CR \in [0, 1]$.

$$v_i = x_b^n + F \cdot (x_{r_1}^n - x_{r_2}^n) \tag{6}$$

$$u_{i,j} = \begin{cases} v_{i,j}, & \textbf{if } rand(0,1) \leq CR \textbf{ or } j = j_{rand} \\ x_{i,j}^n, & \textbf{otherwise} \end{cases} \tag{7}$$

3.7. Selection

At the end of each generation, every individual x_i from X is compared to its corresponding offspring u_i regarding their fitness. Then, the fittest alternative persists in X for the next generation. Any individual's fitness considers two important aspects: The compliance with restrictions and the value of the objective function. Due to the above, the binary tournament selection operator in [68] is used to determine the fittest individual by pondering first the feasibility and then the convergence.

When the last generation ends, the best individual is selected as the fittest one in X. This solution can then be implanted in the motion planner for the mobile robot.

3.8. Local Search

An additional local search procedure, shown in Algorithm 5, is applied to all solutions in X to improve their fitness at the end of each generation. This procedure generates a new local solution by altering the size (using Gaussian mutation over the original size and performing the normalization) and the variables (using the polynomial mutation) of the original solution x_i. The new solution is compared to the original one to determine the local best alternative.

Algorithm 5: LocalSearch(x_i)

Input: x_i (arbitrary solution)
Output: x_i^l (local best solution)
1 $N_s \leftarrow s(x_i)$
2 Variate N_s using Gaussian mutation
3 $x_i^n \leftarrow$Normalize(x_i,N_s)
4 Variate x_i^n using Polynomial Mutation
5 Evaluate x_i^n
6 Select x_i^l as the best between x_i and x_i^n

4. Results an Discussion

4.1. Details of the Experiment

Three test cases, (A), (B) and (C), based on the scenarios (Scenarios are available to the public in the website of the Intelligent and Mobile Robotics Group in the Department of Cybernetics of the Czech Technical University in Prague (http://imr.ciirc.cvut.cz/planning/maps.xml).) in Figure 2, are selected to test the VLV-DE in the solution of the path-planning problem for mobile robots. For all test scenarios in Figure 2, the lower-left corner coincides with the origin of the Cartesian plane (i.e., the path-planning is performed on the first quadrant), the mobile robot radius is set as $r = 0.25$ (m), and the minimum and the maximum number of control points are established as $L_{min} = 2$ and $L_{max} = 100$, respectively. The obstacles are represented with black objects, and the location of p_s and p_t is indicated with a green and a red circle, respectively.

The information of these scenarios is compiled in Table 1. It includes the number of obstacles, the size of the workspace, the start and target points' location and the complexity of the VLV-OP associated with each test case.

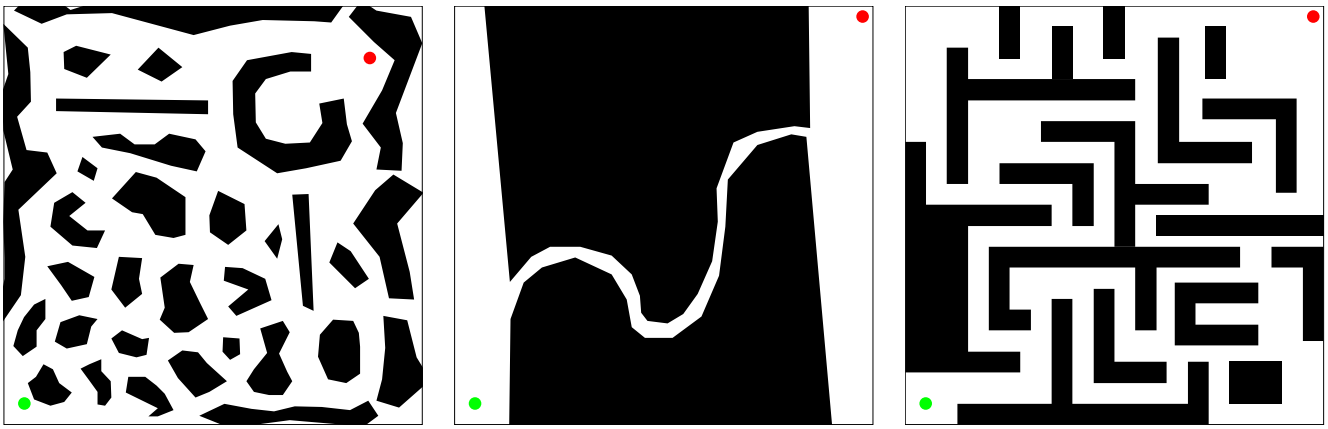
The complexity in Table 1 indicates the rate of feasible alternatives found in a set of 1E8 randomly generated solutions (each solution has a random size in $[L_{min}, L_{max}]$ and random variable values within the box bounds). As it can be noticed, no feasible solutions were found, and the complexity is high.

The VLV-DE parameters are set as follows: the maximum number of generations is $G_{max} = 5000$, the population size $NP = 50$, the crossover rate is $CR = 0.5$, the scaling factor is $F = 0.5$, the distribution index in the polynomial mutation is $\eta_m = 10{,}000$ and the standard deviation in the Gaussian mutation is $\sigma = 1.0$.

The parameters G_{max} and NP are established empirically. The rest of the parameters are tuned using the "i-race" method [69] over ten uniformly distributed samples of the

following parameter ranges: $CR \in (0,1]$, $F \in (0,1]$, $\eta_m \in$ (0,10,000] and $\sigma \in (0,1]$. The cost function provided to "i-race" considers the sum of the J values of the three test cases.

The described experiments are developed in the Java programming language and executed on a PC with a 2.90 GHz i5-10400F processor. The obtained results are discussed next.



(**A**) (**B**) (**C**)

Figure 2. Workspaces used in test cases (**A**), (**B**) and (**C**), respectively.

Table 1. Details of the workspaces used in test cases (A), (B) and (C), respectively.

Test Case	Obstacles	Size (m, m)	p_s (m, m)	p_t (m, m)	Complexity
(A)	32	(100, 100)	(5, 5)	(87.5, 87.5)	0/1E8
(B)	2	(100, 100)	(5, 5)	(97.5, 97.5)	0/1E8
(C)	17	(100, 100)	(5, 5)	(97.5, 97.5)	0/1E8

4.2. Discussion of the Results

Due to the stochastic nature of VLV-DE, a set of 30 independent runs was performed for each test case to determine the effectiveness of the proposal.

The results achieved by VLV-DE are compared to the best solutions found by the weighted A* (i.e., when $w = 1$), and the paths computed by the sampling-based planning method RRT*-Smart [70] (a rapid convergence implementation of RRT*), after a number of $N =$ 30,000 iterations.

The results of VLV-DE are shown in Table 2. This table's information is grouped into blocks that describe the mean, best and worst results regarding the values of J and $s(x)$. The best and worst groups include the value of the objective function J (which represents the overall length of the path from p_s to p_t) and the number of control points $s(x)$ required to achieve it. Analogously, the mean groups include the mean values of J and $s(x)$, and besides, shows the standard deviation (STD) and the 95% confidence interval (C.I.).

Concerning the number of control points, the STD of $s(x)$ samples shows that the paths achieved by VLV-DE do not differ significantly from the mean path (less than two points). This indicator describes a high convergence level in the size of the found paths.

The STD of J samples is related to the number of obstacles and their arrangement in each scenario (see Figure 2). In case (A), the number of possible feasible paths (related to the multiple choices to overcome obstacles in a dense but reasonably well-distributed arrangement) is visibly greater than in case (C) (related to the limited number of maze solutions). In contrast, for case (B), only a single feasible path is observed (the one that goes through the wavy corridor).

Table 2. Differential Evolution for Variable-Length-Vector (VLV-DE) results for test cases (A), (B) and (C).

	Test Case (A)	**Test Case (B)**	**Test Case (c)**
Mean $s(x)$	9	13.2666	13.9666
STD	1.2317	1.2015	1.5196
C.I.	[8.61788, 9.3821]	[12.8939, 13.6394]	[13.4952, 14.4380]
Mean J (m)	2627.0570	3453.4441	4095.0317
STD (m)	45.9620	11.4728	48.2714
C.I. (m)	[2612.7988, 2641.3152]	[3449.8850, 3457.0032]	[4080.0571, 4110.0064]
Best $s(x)$	9	15	13
Best J (m)	2548.8696	3415.7757	3989.3064
Worst $s(x)$	11	13	17
Worst J (m)	2731.2721	3471.7303	4194.8301

The above indicates that scenarios such as (A) and (C), with a greater number of feasible choices, require a deeper exploration of the search space. On the other hand, an exploitative search is best used in scenarios similar to (B), with a limited number of feasible choices. Since the original evolutionary operations of DE/best/1/bin are preserved in VLV-DE, the trade-off of the above exploratory and exploitative capabilities can be adjusted by varying CR and F.

About repeatability of results, the C.I. for both J and $s(x)$, gives a range of values in which a new solution is not statistically significant at the 5% level.

Table 3 shows the best results achieved by the A* method. This table includes the total path length J and the number of control points $s(x)$ between the start and target points. Compared to the paths achieved by VLV-DE, the ones of A* require a noticeable greater number of control points (with at least a ratio of 1:15) as observed in Tables 2 and 3. As a result of the smaller number of control points in VLV-DE, the relevant aspects in the computational and control effort required to perform post-planning tasks are reduced. Such aspects include the path smoothing [39], the development of control problems [40] and also the energy consumption [41]. As a result, VLV-DE shows outstanding behavior.

Table 3. A* results for test cases (A), (B) and (C).

Test Case	$s(x)$	J **(m)**
(A)	166	3314.1421
(B)	219	4374.1421
(C)	232	4634.1421

The aforementioned advantage is related to the capability of VLV-DE to deal with the problem of multimodality induced by the number of control points. Figure 3a,b shows an example of how this multimodality is manifested in path-planning. Figure 3a describes a path with minimum length (i.e., with the minimum value of J) and with the minimum number of control points, while Figure 3b shows an alternative path with the same length but using an increased number of points.

On the opposite side, increasing the number of control points during the search improves the exploration of alternative shorter paths. An example of the above can be observed in Figure 4a,b. Figure 4a shows a path with minimum length using two control points, while Figure 4b depicts an alternative path with fewer points but longer. In this way, VLV-DE applies the variation operators to the normalization size of N_s, intending to search for different solution lengths to fit the problem solution with balanced exploration and exploitation.

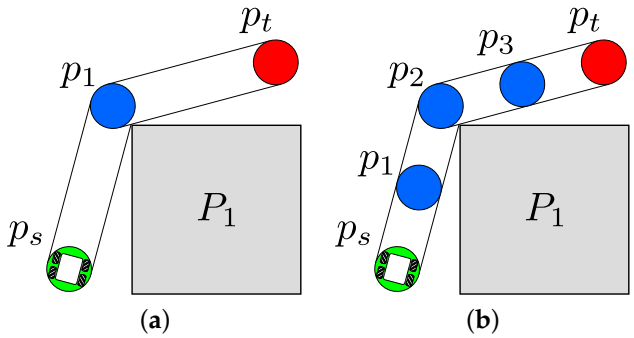


Figure 3. Multimodality in the path-planning problem: (**a**) a path with minimum length and a minimum number of control points and (**b**) the same path using more control points.

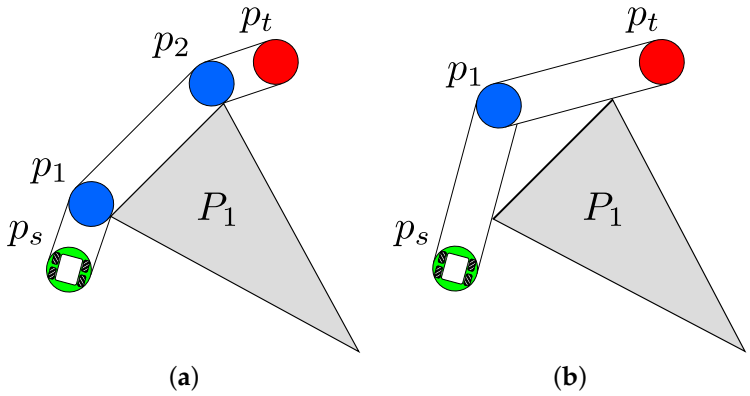


Figure 4. Exploration in the path-planning problem: (**a**) a path with minimum length using two control points and (**b**) a longer path using a single control point.

On the other hand, if the worst path lengths obtained by VLV-DE are compared to the constant lengths of A*, larger paths are found by this last with significant differences of around 583 (m), 902 (m) and 439 (m), for the test cases (A), (B) and (C), respectively. Considering the current energetic limitations of autonomous mobile robots (concerning the capacity of the onboard battery [71]), the benefits of the shortest paths found by VLV-DE are highlighted.

Concerning RRT*-Smart, Table 4 includes the results obtained for each test case in terms of the path length J and the number of control points $s(x)$. Contrasting the results with those of VLV-DE, the number of points in the paths of RRT*-Smart is about less by one for the mean value in Table 2. Therefore, the path smoothness and the computational and control effort required to perform post-planning tasks are similar.

Table 4. RRT*-Smart results for test cases (A), (B) and (C).

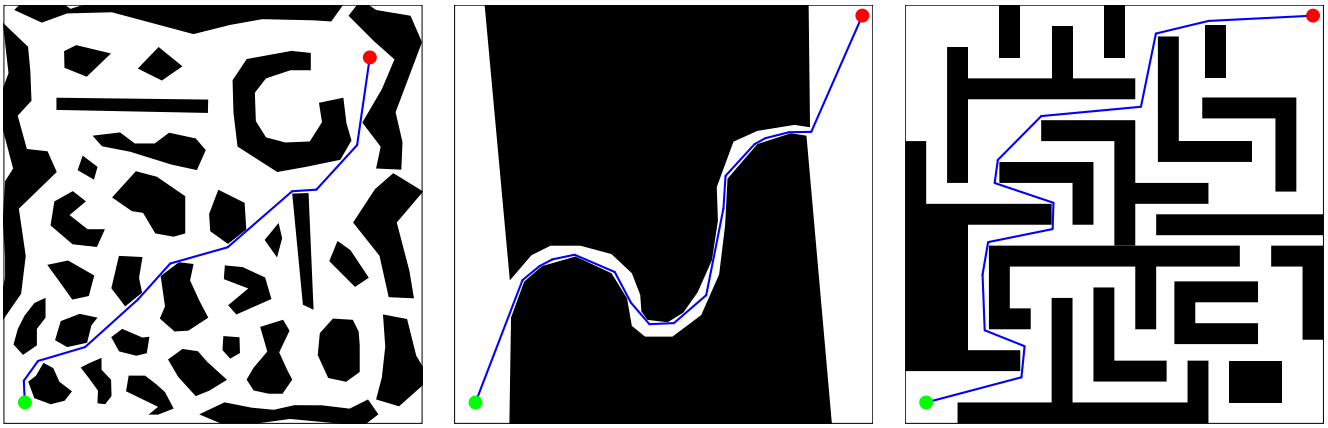
Test Case	$s(x)$	J (m)
(A)	8	3334.5370
(B)	12	3562.2011
(C)	15	4559.8692

If J is taken into account, there are appreciable differences regarding the path lengths achieved by VLV-DE and RRT*-Smart. For cases (A) and (C), where there is a more dense distribution of obstacles than in case (B), the performance of RRT*-Smart regarding J is comparable to that of A*, and VLV-DE found noticeable shorter alternatives, even in the worst run. When there is a more limited number of feasible path alternatives, as in case (B), the paths of VLV-DE and RRT*-Smart have similar lengths.

The advantages of VLV-DE over RRT*-Smart regarding J are attributed to the former's exploitative capacity. In this way, VLV-DE can refine the paths found so far with a local

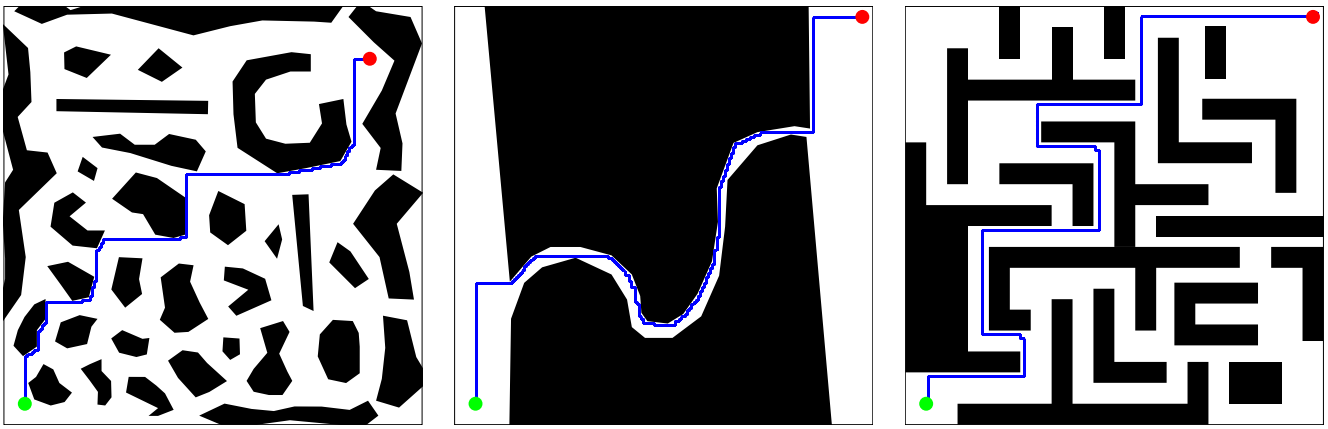
search. Moreover, VLV-DE can also combine normalized paths with the original DE operators to enhance the current point positions of the candidate alternatives in the population.

The best solutions found by VLV-DE for the three study cases can be observed in Figure 5, and correspond to the numerical values described in the best block in Table 2. In the case of A*, the best deterministic paths are displayed in Figures 6 and 7, and match the information in Tables 3 and 4, respectively.



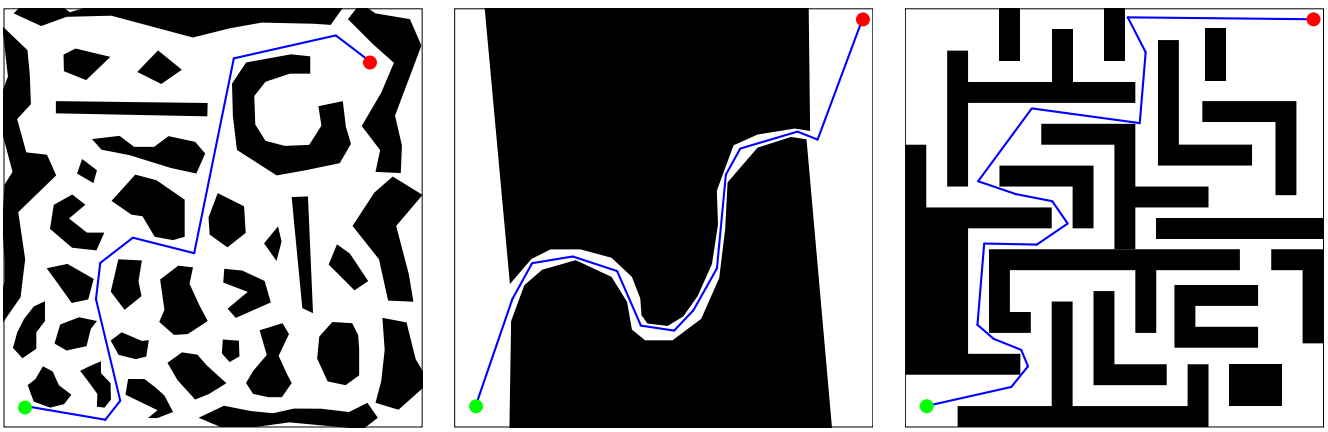
(**A**) (**B**) (**C**)

Figure 5. Best paths achieved by VLV-DE for test cases (**A**), (**B**) and (**C**), respectively.



(**A**) (**B**) (**C**)

Figure 6. Paths achieved by A* when $w = 1$ for test cases (**A**), (**B**) and (**C**), respectively.



(**A**) (**B**) (**C**)

Figure 7. Paths achieved by RRT*-Smart for test cases (**A**), (**B**) and (**C**), respectively.

DE-VLV refines and improves the initial set of candidate solutions given by A*, through the evolutionary process. Still, it can also explore and exploit regions that A* does not consider through compression and decompression procedures. The above results in an enhanced path as observed in the test case (A) of Figures 5 and 6. In those figures, substantial differences between the best path found by A* and the best one found by VLV-DE are observed.

Graphically, the paths obtained by A* are less smooth than those of VLV-DE. This is attributed to the difference in the number of control points. Moreover, the solutions of A* are somewhat distant from the path that could be estimated as optimal, contrary to what happens with the paths of VLV-DE, for which it is difficult to identify a better alternative.

Finally, it is important to comment that the computational complexity of VLV-DE is at most the same of any original variant of DE [72], i.e., $O(c(L_{max}) \cdot G_{max} \cdot NP)$ in a "big O" notation, where $c(L_{max})$ is the complexity of the objective function that depends on the maximum dimension of the search space L_{max}. The proposal's computation time is strongly related to $c(L_{max})$ which also hangs on the number of obstacles and their shapes. In scenario (A), which has a dense distribution of obstacles, the mean computation time of the proposal is 278.95 (s). Since most of the consumed time is associated with the collision detection task in the proposal, the overall execution time could be reduced using a more efficient collision detection approach such as divide and conquer [73] and exploiting the parallelizable features of DE [74].

5. Conclusions and Future Work

This work proposes a path-planning approach based on VLV optimization for mobile robots. The VLV-OP aims to find the number of control points and their values in a continuous search space to minimize an objective function related to the total path length and meet a single equality constraint for collision avoidance. A novel variant of DE named VLV-DE is proposed to solve the VLV-OP in the path-planning. This optimizer can deal with actual variable-length candidate solutions that interact with each other to enhance their fitness.

One of the main advantages of the proposed VLV-OP in the path-planning is the no necessity of the a priori knowledge of the number of control points, such that human actors do not affect the robot autonomy.

The proposal's main feature is the capability of improving the exploration and exploitation of the search space through the evolutionary process by including the normalization procedure for the homogeneity of the solution sizes and incorporating the local search procedure. Therefore, the proposed VLV-DE can find suitable solutions to the multimodality problem of the mobile robot path-planning.

Comparative results of the VLV-DE with the deterministic node-based technique A* through three complex test cases indicate that the proposal finds shorter and smoother paths with a notably smaller number of control points. The above leads to potential benefits in terms of energy consumption and lower cost post-planning processing. Compared to the sampling-based planning method RRT*-Smart, VLV-DE can find shorter paths with a similar number of control points.

The improvement of the computational time required by this proposal is suggested as the future work by using a divide and conquer approach with high-performance parallel computing. Moreover, other computational intelligence approaches such as neural networks can speed up the convergence and provide it with adaptability to handle dynamic scenarios.

Author Contributions: Conceptualization, A.R.-M.; methodology, A.R.-M., J.S.-R., and M.G.V.-C.; software, A.R.-M., O.S.-P., and G.F.-C.; validation, A.R.-M. and M.G.V.-C.; formal analysis, A.R.-M. and M.G.V.-C.; investigation, A.R.-M., J.S.-R., M.G.V.-C., O.S.-P., and G.F.-C.; resources, A.R.-M., J.S.-R., and M.G.V.-C.; data curation, O.S.-P. and G.F.-C.; writing—original draft preparation, A.R.-M. and J.S.-R.; writing—review and editing, M.G.V.-C., O.S.-P., and G.F.-C.; visualization, O.S.-P., and G.F.-C.; supervision, M.G.V.-C.; project administration, A.R.-M., J.S.-R., and M.G.V.-C.; funding acquisition, A.R.-M., J.S.-R., and M.G.V.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the economic support program of the Comisión de Operación y Fomento de Actividades Académicas (COFAA) of the Instituto Politécnico Nacional, and in part by the Dirección de Posgrado, Investigación e Innovación of the Tecnológico Nacional de México through the projects No. 9996 and No. 11013.

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

References

1. Naranjo, J.E.; Lozada, E.C.; Espín, H.I.; Beltran, C.; García, C.A.; García, M.V. Flexible architecture for transparency of a bilateral tele-operation system implemented in mobile anthropomorphic robots for the oil and gas industry. *IFAC-PapersOnLine* **2018**, *51*, 239–244.
2. Wang, Y.; Lang, H.; De Silva, C.W. A hybrid visual servo controller for robust grasping by wheeled mobile robots. *IEEE/ASME Trans. Mechatron.* **2009**, *15*, 757–769.
3. Orozco-Rosas, U.; Picos, K.; Montiel, O. Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots. *IEEE Access* **2019**, *7*, 156787–156803.
4. Primatesta, S.; Guglieri, G.; Rizzo, A. A risk-aware path planning strategy for uavs in urban environments. *J. Intell. Robot. Syst.* **2019**, *95*, 629–643.
5. Alexopoulos, C.; Griffin, P.M. Path planning for a mobile robot. *IEEE Trans. Syst. Man Cybern.* **1992**, *22*, 318–322.
6. Niu, H.; Lu, Y.; Savvaris, A.; Tsourdos, A. An energy-efficient path planning algorithm for unmanned surface vehicles. *Ocean Eng.* **2018**, *161*, 308–321.
7. Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y. Survey of robot 3D path planning algorithms. *J. Control Sci. Eng.* **2016**, *doi:10.1155/2016/7426913*.
8. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Tech. Rep. 98-11; Department of Computer Science, Iowa State University: Ames, IA, USA, 1998.
9. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580.
10. Karaman, S.; Frazzoli, E. Incremental sampling-based algorithms for optimal motion planning. *Robot. Sci. Syst. VI* **2010**, *104*, doi:10.15607/RSS.2010.VI.034.
11. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894.
12. Ayanian, N.; Kallem, V.; Kumar, V. Synthesis of feedback controllers for multiple aerial robots with geometric constraints. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3126–3131.
13. Sethian, J.A. Fast marching methods. *SIAM Rev.* **1999**, *41*, 199–235.
14. Barraquand, J.; Langlois, B.; Latombe, J.C. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst. Man Cybern.* **1992**, *22*, 224–241.
15. Janet, J.A.; Luo, R.C.; Kay, M.G. The essential visibility graph: An approach to global motion planning for autonomous mobile robots. In Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, 21–27 May 1995; Volume 2, pp. 1958–1963.
16. Stentz, A.; others. The focussed D* algorithm for real-time replanning. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, QC, Canada, 20–25 August 1995 ; Volume 2, pp. 1652–1659.
17. Britzelmeier, A.; Gerdts, M. A Nonsmooth Newton Method for Linear Model-Predictive Control in Tracking Tasks for a Mobile Robot With Obstacle Avoidance. *IEEE Control Syst. Lett.* **2020**, *4*, 886–891.
18. Liu, L.; Wang, D.; Peng, Z. Path following of marine surface vehicles with dynamical uncertainty and time-varying ocean disturbances. *Neurocomputing* **2016**, *173*, 799–808.
19. Song, Q.; Zhao, Q.; Wang, S.; Liu, Q.; Chen, X. Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization. *IEEE Access* **2020**, *8*, 62107–62115.
20. Singh, N.H.; Thongam, K. Neural network-based approaches for mobile robot navigation in static and moving obstacles environments. *Intell. Serv. Robot.* **2019**, *12*, 55–67.
21. Wang, J.; Chi, W.; Li, C.; Wang, C.; Meng, M.Q.H. Neural RRT*: Learning-based optimal path planning. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1748–1758.

22. Amer, M.; Namaane, A.; M'sirdi, N. Optimization of hybrid renewable energy systems (HRES) using PSO for cost reduction. *Energy Procedia* **2013**, *42*, 318–327.
23. Han, M.; Wang, M.H.; Fan, J.C. Trajectory optimization based on improved differential evolution algorithm. *Control Decis.* **2012**, *27*, 247–251.
24. Huang, J.; Qingyun, W. Robust optimization of hub-and-spoke airline network design based on multi-objective genetic algorithm. *J. Transp. Syst. Eng. Inf. Technol.* **2009**, *9*, 86–92.
25. Flores-Caballero, G.; Rodríguez-Molina, A.; Aldape-Pérez, M.; Villarreal-Cervantes, M.G. Optimized Path-Planning in Continuous Spaces for Unmanned Aerial Vehicles Using Meta-Heuristics. *IEEE Access* **2020**, *8*, 176774–176788.
26. Karami, A.H.; Hasanzadeh, M. An adaptive genetic algorithm for robot motion planning in 2D complex environments. *Comput. Electr. Eng.* **2015**, *43*, 317–329.
27. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120.
28. Jones, D.F.; Mirrazavi, S.K.; Tamiz, M. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *Eur. J. Oper. Res.* **2002**, *137*, 1–9.
29. Li, H.; Deb, K. Challenges for evolutionary multiobjective optimization algorithms in solving variable-length problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 2217–2224.
30. Meza, G.R.; Ferragud, X.B.; Saez, J.S.; Durá, J.M.H. *Controller Tuning with Evolutionary Multiobjective Optimization: A Holistic Multiobjective Optimization Design Procedure*; Springer International Publishing: Cham, Switzerland, 2016; Volume 85.
31. Onal, C.D.; Tolley, M.T.; Wood, R.J.; Rus, D. Origami-inspired printed robots. *IEEE/ASME Trans. Mechatron.* **2014**, *20*, 2214–2221.
32. Rabadi, G.; Moraga, R.J.; Al-Salem, A. Heuristics for the unrelated parallel machine scheduling problem with setup times. *J. Intell. Manuf.* **2006**, *17*, 85–97.
33. Dewang, H.S.; Mohanty, P.K.; Kundu, S. A robust path planning for mobile robot using smart particle swarm optimization. *Procedia Comput. Sci.* **2018**, *133*, 290–297.
34. Lamini, C.; Benhlima, S.; Elbekri, A. Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Comput. Sci.* **2018**, *127*, 180–189.
35. Martinez-Soltero, E.G.; Hernandez-Barragan, J. Robot navigation based on differential evolution. *IFAC-PapersOnLine* **2018**, *51*, 350–354.
36. Tang, B.; Zhu, Z.; Luo, J. Hybridizing particle swarm optimization and differential evolution for the mobile robot global path planning. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 86.
37. Oleiwi, B.K.; Roth, H.; Kazem, B.I. Multi objective optimization of path and trajectory planning for non-holonomic mobile robot using enhanced genetic algorithm. In Proceedings of the 8th International Conference on Neural Networks and Artificial Intelligence (ICNNAI 2014), Brest, Belarus, 3–6 June 2014; pp. 50–62.
38. Niederberger, C.; Radovic, D.; Gross, M. Generic path planning for real-time applications. In Proceedings of the Computer Graphics International, Crete, Greece, 19 June 2004; pp. 299–306.
39. Yang, K.; Jung, D.; Sukkarieh, S. Continuous curvature path-smoothing algorithm using cubic B zier spiral curves for non-holonomic robots. *Adv. Robot.* **2013**, *27*, 247–258.
40. Son, T.D.; Ahn, H.S.; Moore, K.L. Iterative learning control in optimal tracking problems with specified data points. *Automatica* **2013**, *49*, 1465–1472, doi:10.1016/j.automatica.2013.02.008.
41. Davoodi, M.; Panahi, F.; Mohades, A.; Hashemi, S.N. Clear and smooth path planning. *Appl. Soft Comput.* **2015**, *32*, 568–579.
42. Shwail, S.H.; Karim, A.; Turner, S. Probabilistic multi robot path planning in dynamic environments: A comparison between A* and DFS. *Int. J. Comput. Appl.* **2013**, *82*.
43. Tu, J.; Yang, S.X. Genetic algorithm based path planning for a mobile robot. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003), Taipei, Taiwan, 14–19 September 2003; Volume 1, pp. 1221–1226.
44. Riquelme, J.; Ridao, M.; Camacho, E.; Toro, M. Using genetic algorithms with variable-length individuals for planning two-manipulators motion. In *Artificial Neural Nets and Genetic Algorithms*; Springer: Vienna, Austria, 1998; pp. 26–30.
45. Limbourg, P.; Kochs, H.D. Preventive maintenance scheduling by variable dimension evolutionary algorithms. *Int. J. Press. Vessels Pip.* **2006**, *83*, 262–269.
46. Chuanjiao, S.; Wei, Z.; Yuanqing, W. Scheduling combination and headway optimization of bus rapid transit. *J. Transp. Syst. Eng. Inf. Technol.* **2008**, *8*, 61–67.
47. Ahn, C.W.; Ramakrishna, R.S. A genetic algorithm for shortest path routing problem and the sizing of populations. *IEEE Trans. Evol. Comput.* **2002**, *6*, 566–579.
48. Qiongbing, Z.; Lixin, D. A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. *Expert Syst. Appl.* **2016**, *60*, 183–189.
49. Chen, Y.; Mahalec, V.; Chen, Y.; Liu, X.; He, R.; Sun, K. Reconfiguration of satellite orbit for cooperative observation using variable-size multi-objective differential evolution. *Eur. J. Oper. Res.* **2015**, *242*, 10–20.
50. Das, S.; Abraham, A.; Konar, A. Automatic clustering using an improved differential evolution algorithm. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2007**, *38*, 218–237.

51. Yuan, H.; He, J. Evolutionary design of operational amplifier using variable-length differential evolution algorithm. In Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, China, 22–24 October 2010; Volume 4, pp. V4–610.
52. Pereira, C.M.; Lapa, C.M.; Mol, A.C.; Da Luz, A.F. A particle swarm optimization (PSO) approach for non-periodic preventive maintenance scheduling programming. *Prog. Nucl. Energy* **2010**, *52*, 710–714.
53. Wang, B.; Sun, Y.; Xue, B.; Zhang, M. Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (IEEE CEC 2018), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
54. Xue, B.; Ma, X.; Gu, J.; Li, Y. An improved extreme learning machine based on variable-length particle swarm optimization. In Proceedings of the 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China, 12–14 December 2013; pp. 1030–1035.
55. Wilt, C.M.; Thayer, J.T.; Ruml, W. A comparison of greedy search algorithms. In Proceedings of the Third Annual Symposium on Combinatorial Search, Atlanta, GA, USA, 8–10 July 2010; pp. 130-136.
56. Alakshendra, V.; Chiddarwar, S.S. Adaptive robust control of Mecanum-wheeled mobile robot with uncertainties. *Nonlinear Dyn.* **2017**, *87*, 2147–2169.
57. Sánchez, P.; Casado, R.; Bermúdez, A. Real-Time Collision-Free Navigation of Multiple UAVs Based on Bounding Boxes. *Electronics* **2020**, *9*, 1632.
58. Lin, M.C.; Manocha, D.; Cohen, J.; Gottschalk, S. Collision detection: Algorithms and applications. In *Algorithms for Robotic Motion and Manipulation*; A K Peters/CRC Press: New York, NY, USA, 1997; pp. 129–142.
59. van der Spuy, R., Collisions Between Polygons. In *AdvancED Game Design with Flash*; Apress: Berkeley, CA, USA, 2010; pp. 223–303, doi:10.1007/978-1-4302-2740-3_4.
60. Madavan, N.K. Multiobjective optimization using a Pareto differential evolution approach. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1145–1150.
61. Chakraborty, U.K. *Advances in Differential Evolution*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 143.
62. Price, K.V. Differential evolution. In *Handbook of Optimization*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 187–214.
63. Mezura-Montes, E.; Velázquez-Reyes, J.; Coello Coello, C.A. A comparative study of differential evolution variants for global optimization. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06), Seattle, WA, USA, 8–12 July 2006; pp. 485–492.
64. Thayer, J.T.; Ruml, W. Faster than Weighted A*: An Optimistic Approach to Bounded Suboptimal Search. In Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008), Sydney, Australia, 14–18 September 2008 ; pp. 355–362.
65. Seet, B.C.; Liu, G.; Lee, B.S.; Foh, C.H.; Wong, K.J.; Lee, K.K. A-STAR: A mobile ad hoc routing strategy for metropolis vehicular communications. In Proceedings of the International Conference on Research in Networking, Athens, Greece, 9–14 May 2004; pp. 989–999.
66. Fogel, D.B.; Atmar, J.W. Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems. *Biol. Cybern.* **1990**, *63*, 111–114.
67. Zeng, G.Q.; Chen, J.; Li, L.M.; Chen, M.R.; Wu, L.; Dai, Y.X.; Zheng, C.W. An improved multi-objective population-based extremal optimization algorithm with polynomial mutation. *Inf. Sci.* **2016**, *330*, 49–73.
68. Goldberg, D.E. *Genetic Algorithms*; Pearson Education: Tamil Nadu, India, 2006.
69. Montero, E.; Riff, M.C. Effective collaborative strategies to setup tuners. *Soft Comput.* **2020**, *24*, 5019–5041.
70. Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M.; Muhammad, M.S. RRT*-SMART: A rapid convergence implementation of RRT. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 299.
71. Broderick, J.A.; Tilbury, D.M.; Atkins, E.M. Characterizing energy usage of a commercially available ground robot: Method and results. *J. Field Robot.* **2014**, *31*, 441–454.
72. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyses. *Swarm Evol. Comput.* **2019**, *44*, 546–558.
73. Hasegawa, T.; Terasaki, H. Collision avoidance: Divide-and-conquer approach by space characterization and intermediate goals. *IEEE Trans. Syst. Man Cybern.* **1988**, *18*, 337–347.
74. Tasoulis, D.K.; Pavlidis, N.G.; Plagianakos, V.P.; Vrahatis, M.N. Parallel differential evolution. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE CEC 2004), Portland, OR, USA, 19–23 June 2004; Volume 2, pp. 2023–2029.