electronics

MDPI

Article

Comparative Analysis of Metaheuristic Optimization Methods for Trajectory Generation of Automated Guided Vehicles

Eduardo Bayona [1,2,*], Jesús Enrique Sierra-García [1,2,*] and Matilde Santos [3]

[1] Department of Digitalization, University of Burgos, 09001 Burgos, Spain
[2] UBU-MICHELIN Joint Research Unit in Automation and Smart Industry, Arainnov Michelin Aranda, 09400 Aranda de Duero, Spain
[3] Institute of Knowledge Technology, Complutense University of Madrid, 28040 Madrid, Spain; msantos@ucm.es
* Correspondence: ebayona@ubu.es (E.B.); jesierra@ubu.es (J.E.S.-G.)

Abstract: This paper presents a comparative analysis of several metaheuristic optimization methods for generating trajectories of automated guided vehicles, which commonly operate in industrial environments. The goal is to address the challenge of efficient path planning for mobile robots, taking into account the specific capabilities and mobility limitations inherent to automated guided vehicles. To do this, three optimization techniques are compared: genetic algorithms, particle swarm optimization and pattern search. The findings of this study reveal the different efficiency of these trajectory optimization approaches. This comprehensive research shows the strengths and weaknesses of various optimization methods and offers valuable information for optimizing the trajectories of industrial vehicles using geometric occupancy maps.

Keywords: automatic guided vehicle; metaheuristic optimization; industry 4.0; trajectories

Citation: Bayona, E.; Sierra-García, J.E.; Santos, M. Comparative Analysis of Metaheuristic Optimization Methods for Trajectory Generation of Automated Guided Vehicles. *Electronics* **2024**, *13*, 728. https://doi.org/10.3390/electronics13040728

Academic Editors: Sergio Montenegro, Michael Strohmeier and Nikolay Hinov

Received: 25 December 2023
Revised: 4 February 2024
Accepted: 6 February 2024
Published: 11 February 2024

1. Introduction

Automatic guided vehicles (AGV) are industrial vehicles that have favored the implementation of industry 4.0. With the new and more recent technologies, they are also becoming a pillar for industry 5.0, where the interaction of robots with humans is key. These vehicles take care of many logistical tasks, tedious for the human operator. In many cases, they share space with workers [1]. In the realm of autonomous guided vehicles (AGVs), trajectory planning is crucial for safe and efficient navigation. Vagale et al. in [2] emphasize the importance of path design methodologies and guidance laws, with a focus on simplicity and efficiency. New models for path planning optimization, addressing the complexity of manufacturing networks and the stochastic nature of AGVs as introduced in [3]. The authors in [4,5] underscore the significance of safe trajectory planning, with the authors in [4] specifically focusing on the safety aspect. Tamizi et al. in [5] discuss the implications of trajectory planning on the design and use of automatic machines. The selection of an appropriate trajectory planning algorithm is crucial for achieving the safest trajectory for AGVs. Even more, the generation of trajectories has to consider not only obstacle avoidance but also the inherent constraints and limitations of this kind of industrial vehicle, such as maneuverability, complex and non-linear dynamics, etc. [6]. Therefore, a research topic of industrial interest is the development of optimal trajectories that are safe, efficient and feasible.

The optimization of trajectories has been addressed using various methods, including metaheuristic techniques. Metaheuristic optimization techniques are iterative procedures that guide a search through the solution space to find the best possible solution, particularly useful for solving combinatorial optimization problems, nonlinear programming and multi-objective optimization [7]. Metaheuristic search techniques, such as simulated

annealing, genetic algorithm, and particle swarm optimization, have been applied in engineering problems, demonstrating their versatility and effectiveness [8,9]. Furthermore, metaheuristic optimization techniques have been used for optimization problems with continuous variables, showcasing their applicability across different domains [10]. The selection of an appropriate metaheuristic technique for a specific problem is contingent upon the problem's nature, the structure of the search space, and the problem's constraints is explored by the authors in [11]. Metaheuristic techniques encompass a diverse range of methods, each with its own characteristics and suitability for specific types of problems as explored in [12]. Currently, metaheuristic optimization techniques are the focus of active research, and their application continues to expand as new approaches are developed and adapted to emerging challenges in various domains [13].

The trajectory planning for mobile robots or vehicles is a critical aspect that requires careful analysis and evaluation of different strategies. This work's primary contribution is the comparison and evaluation of three meta-heuristic optimization methods for AGV trajectory planning: genetic algorithms (GA), particle swarm optimization (PSO) and pattern search (PS). A strategy for general optimization, independent of the optimization method, is proposed to obtain a solution that prioritizes a safe trajectory. The methodology to perform the comparison is also described in detail. Three scenarios of increasing complexity are created to test the different techniques. These optimization strategies have been widely applied in trajectory planning for robotic systems. The genetic algorithm, inspired by natural selection, has demonstrated effectiveness in trajectory planning. Similarly, PSO, simulating the social behavior of birds, has shown effectiveness in optimizing trajectories for robotic applications. Furthermore, pattern search algorithms have been applied for trajectory planning in diverse domains, offering a different approach to trajectory optimization.

Text is structured as follows: Section 2 summarizes some related works. Section 3 presents the general optimization methodology used, including the problem modeling (occupancy map, the trajectory definition and the collisions with obstacles). It also presents the configuration and application of the optimization strategies and the metrics used for the comparison. Section 4 presents the experiments and the evaluation of the solutions for each proposed scenario, discussing the problems found in each case. Finally, in Section 5 conclusions and future works are recommended.

2. Related Works

Metaheuristic optimization methods, such as genetic algorithms, particle swarm optimization, and pattern search, have been widely applied in power engineering, engineering optimization, and many other fields due to their ability to provide good solutions to complex problems [14,15]. These methods are characterized by their flexibility and efficiency in solving a wide range of optimization problems, making them suitable for real-world applications [16]. The effectiveness of these methods has been demonstrated in various environments and areas, in a wide range of domains [17]. The relevance of these strategies is further supported by the authors in [18], which discusses decentralized motion planning and scheduling of AGVs. Additionally, the study by Altché and Fortelle in [19] considers trajectory planning and control for autonomous ground vehicles in the presence of obstacles, emphasizing the importance of considering the link to the scheduling of interacting machines. Furthermore, the authors in [20] focus on optimal trajectory optimization for robots, considering constraint conditions such as speed and acceleration. These references collectively support the significance of evaluating and comparing different trajectory-planning strategies for mobile robots and vehicles.

Some studies have explored the use of genetic algorithms for AGV path planning, each proposing different improvements. The authors in [21] introduced real number coding, heuristic population initialization, and obstacle avoidance and smoothness modules to enhance the traditional genetic algorithm. Cao et al. in [22] focused on the use of a grid model and chamfer operator to speed up convergence in AUV path planning. The authors in [23] applied an improved genetic algorithm with three-exchange crossover

heuristic operators and double-path constraints for multi-AGV path planning. Based on a constructed ground map, a hybrid path planning algorithm is proposed by Li et al. in [24] to optimize the planned path. A genetic algorithm is used for global path planning, and a local rolling optimization is used to constantly optimize the results of the genetic algorithm. These studies collectively demonstrate the potential of genetic algorithms in mobile robots in general, and specifically for AGV path planning, with each proposing unique enhancements to improve efficiency and effectiveness.

Recent literature has also explored the use of particle swarm optimization (PSO) for trajectory optimization in various contexts. For instance, its application in generating stochastic trajectories for unmanned aerial vehicles (UAVs) is shown in [25]. Authors in [26] provided a comprehensive review of swarm intelligence algorithms, highlighting the wide application and discussion of extended algorithms. Additionally, recent studies have focused on improving PSO for path planning, such as the work by Lu et al. in [27], which introduced an improved simulated annealing PSO algorithm for mobile robot path planning. Moreover, the research by Huang et al. in [28] proposed a novel PSO algorithm based on reinforcement learning for autonomous underwater vehicle (AUV) path planning, addressing the consideration of ocean currents.

Path planning is a critical aspect of autonomous systems, and the use of pattern search algorithms has gained significant attention in recent research. The authors in [29] proposed a mobile robot path planning method using globally guided reinforcement learning, demonstrating its effectiveness across various map types and obstacle densities. Additionally, Ma et al. in [30] introduced an improved Q-learning algorithm based on a continuous local search policy for mobile robot path planning, emphasizing the exhaustive search of state-action pairs in the environment. Furthermore, Ren et al. presented in [31] the Multi-Objective Path-Based D* Lite algorithm, highlighting its efficiency in navigation through unknown terrains. These recent studies underscore the growing interest in leveraging pattern search techniques for enhancing path planning in dynamic and complex environments.

A comparison of genetic algorithm (GA) with particle swarm optimization (PSO) or pattern search (PS) for trajectory planning reveals the strengths and weaknesses of each approach. PSO has been found to be superior to GA in terms of searching speed and convergence [32], and has been successfully applied in trajectory optimization of industrial robots [33]. However, a novel stochastic gradient PSO algorithm has been proposed to address the premature convergence and poor accuracy issues of standard PSO [34].

3. Optimization Methodology

This work compares three optimization algorithms for generating safe trajectories for automated guided vehicles (AGVs) in industrial environments. Three different work scenarios have been defined. To begin with, it is important to clearly define the comparison methodology, the optimization strategy and the modeling of the elements involved in the procedure. The comparison methodology is shown in Figure 1.

The mathematical modeling of the trajectories that are going to be generated by the different methods and the steps of the process must be defined. For clarity reasons, the problem modeling is explained in Section 3.1 and the optimization strategy proposed is described in Section 3.2. As will be shown, the optimization strategy is independent of the heuristic technique applied. Some meaningful metrics are defined and are recorded for each iteration of the optimization methods. These values are used to evaluate and compare the metaheuristic techniques at different points of the optimization process. These metrics include, for instance, the time required to achieve a collision-free solution and the time needed to obtain the best solution in each case. They are explained in Section 3.3. In addition to these comparison metrics, is also necessary to define the scenarios to compare the trajectories (Section 3.4). Once the whole frame is defined, the metaheuristic techniques are configured as explained in Section 3.5, and then tested following the process described in Section 3.6. For each run, a log file with the metrics to be evaluated is obtained. This

information is used to perform the comparison of techniques, and the results are discussed in Section 4.

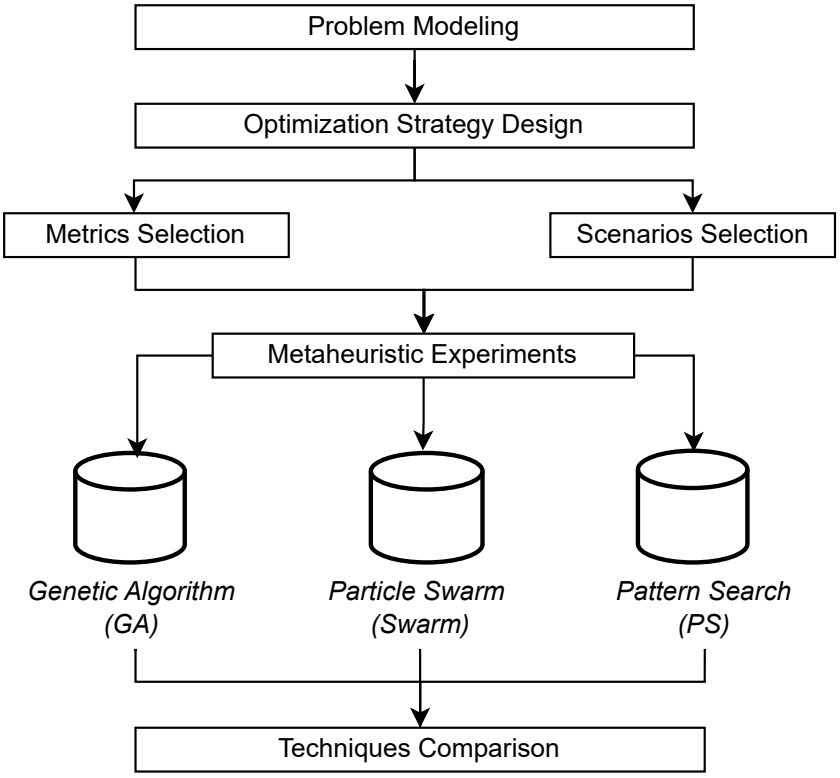


Figure 1. Optimization methodology.

3.1. Problem Modeling

Figure 2 describes the trajectory generation process considering the specific requirements of the logistic application. The logistic task is defined by a computer-aided design (CAD) occupancy map which contains the dimensions of the workspace and the obstacles, together with the initial and final points and angles. The accurate definition of the initial and final angles is necessary to ensure the docking at the stations. An automatic method, shown by the authors in [35], analyzes the occupancy map to extract the obstacles list which contains the polylines that define the boundaries of the obstacles. This methodology simplifies the calculation of the distance between the vehicle and the obstacles and makes the calculation of the fitness function of the optimization process more efficient.

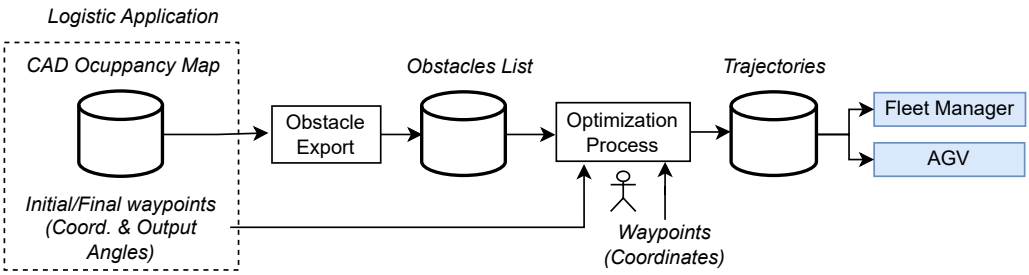


Figure 2. Trajectory generation process.

The user defines a set of waypoints to facilitate the work of the optimization tool and this way helps accelerate the optimization process. This definition of the waypoints does not need to be very accurate; indeed, the user can locate the obstacles in the occupancy map and give some tentative points in the available space. This way the optimization process only must focus on the optimization of the angles of the waypoints. Therefore the

optimization process receives the obstacles list, the start and end points with the angles, and the list of waypoints selected by the user. Then, iteratively, it proposes different trajectories to find the one that offers the best safety conditions, that is, the trajectory furthest away from any obstacle. These trajectories can be sent to the AGV or can be managed by a fleet control system.

To show the trajectories the kinematic variables, such as position and velocity, have to be obtained, which gives an understanding of the dynamics of the vehicle. Differential geometry is used to model these trajectories as it allows us to describe the behavior of an object following a specific path in the space [36]. This discipline describes not only the object's trajectory but also specifies key parameters, such as the curvature and torsion of the path. In some parametrization techniques, these variables are part of the differential equations that describe the vehicle kinematics, such as in the Frenet–Serret equations [37]. In these curves, three elementary vectors are identified: tangent, normal and binormal vectors. The Frenet–Serret formalism has been used in this work because it enhances curve smoothness and the ability to avoid obstacles with more natural trajectories. Frenet equations in (1) define a flat curve in $\mathbb{R}^2$, being T the tangent vector and N the normal vector that satisfies this equation.

$$\begin{aligned} T's &= ksNs \\ N's &= -ksTs \end{aligned} \tag{1}$$

where s is the distance from the origin at each point, k is the curvature of the trajectory, $T(s)$ is the tangent vector, $N(s)$ is the normal vector, all of them at distance s, and the symbol $'$ denotes the perpendicular vector.

Clothoid curves have various applications in planning and generating trajectories for autonomous vehicles or robots. This article uses clothoid curves expressed through Frenet formulas to determine paths of the AGV within the occupancy map. To do it, an algorithm following the work of Bertolazzi and Frego in [38] is applied to solve the G1 Hermite interpolation problem, obtaining a clothoid curve that connects two points with unit tangent vectors in a plane. The interpolation problem is expressed as a set of three nonlinear equations. Solutions for this optimization problem are defined as a set of elements: $\theta(N_w)$, where N_w is the number of intermediate points and θ is the output angle of the trajectory at those intermediate points. Therefore, the solution will have as many elements as intermediate points. The angle range is between 0 and 2π. These conditions are formally expressed in (2).

$$\theta_i \in [0, 2\pi] \; ; \; i \in \{\mathbb{N} \leq N_w\} \tag{2}$$

The calculation of the distance between obstacles and the vehicle uses the singular points of the vehicle, i.e., the vertices of the vehicle, and the segments that delineate the map obstacles. The location of the vertices of the AGV in the inertial frame S_0 changes while the AGV is following the trajectory. However, the position of the vertices in the coordinate frame S_R located at the center of the AGV, is constant. The set of vertices in the coordinate S_R is denoted by $V_R = (W/2, L/2), (-W/2, L/2), (W/2, -L/2), (-W/2, -L/2)$, where W is the width of the vehicle, and L is its length. Given the position $P = (P_x, P_y, 0)$ of the center of the vehicle, and its orientation θ, both in the inertial frame S_0; it is possible to obtain the location in S_0 of any vertex $v_R \in V_R$ by using the coordinate transformation defined in Equation (5). The vertex in S_0 is $v_0 = (v_{0_x}, v_{0_y})$. The translation matrix is T_{tr} and T_{rot} is the rotation matrix around the z-axis, as expressed in Equations (3) and (4), respectively.

$$T_{\text{tr}}(P_x, P_y) = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_{\text{rot}}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} v_{0_x} \\ v_{0_y} \\ 0 \\ 1 \end{bmatrix} = T_{\text{tr}}(P_x, P_y) \cdot T_{\text{rot}}(\theta) \begin{bmatrix} v_{R_x} \\ v_{R_y} \\ 0 \\ 1 \end{bmatrix} \tag{5}$$

In Figure 3 the robot is represented, with the coordinate systems (S_0) and (S_R), and the vertices of the vehicle. The centre of the vehicle P is shown in blue and the vertices in yellow. It is possible to see how the location of the vertices in S_0 depends on the orientation θ and the position P, while the vehicle follows the trajectory.

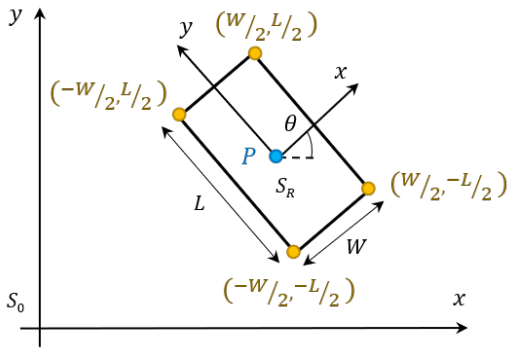


Figure 3. Vehicle singular points representation in general and relative reference frames.

To determine the distance between the obstacle j and the vertices in the set V_0, each vertex $v_0 \in V_0$ is projected onto the lines containing the segments defining the obstacles on the map. If the projection Q falls within the obstacle segment AB, the distance is calculated as the norm of the vector connecting the point with its projection, $||\vec{Qv_0}||$. Otherwise, the distance is obtained as the norm of the vector connecting the point with the nearest vertex of the obstacle, $||\vec{Av_0}||$. The calculation of this distance is further explained in [39]. Let $D(v_0, \vec{AB})$ be the distance of a vertex $v_0 \in V_0$ to the segment $\vec{AB}$, that is, the distance from the vehicle boundary of the vehicle to the closest segment of the occupancy map, $\vec{AB}$. If point Q belongs to the segment $\vec{AB}$, the distance is the norm of $\vec{Qv_0}$. In case Q is outside the $\vec{AB}$ segment, the distance is the norm of $\vec{Av_0}$, being A the nearest obstacle's vertex to v_0.

The distance $D(V_0, \vec{AB})$ from the vertex v_0 to the occupancy map is expressed as in Equation (6):

$$D(v_0, \vec{AB}) = \begin{cases} ||\vec{Qv_0}|| = \sqrt{(Q_x - v_{0_x})^2 + (Q_y - v_{0_y})^2} & \text{if } Q \in \vec{AB} \\ ||\vec{Av_0}|| = \sqrt{(A_x - v_{0_x})^2 + (A_y - v_{0_y})^2} & \text{if } Q \notin \vec{AB} \end{cases} \tag{6}$$

where (v_{0_x}, v_{0_y}) denotes the coordinates of the vehicle's vertex in S_0.

This calculation is repeated for all segments of all the obstacles, in order to detect the closest segment. This distance is called D_{ijk}, and it represents the distance from the vertex k of the trajectory point i to the obstacle j. If the point is inside an obstacle, this means collision and the distance is named $D_{c_{ijk}}$. Figure 4 shows examples of non-collision minimum distance D_{ijk} and collision minimum distance $D_{c_{ijk}}$ to an obstacle j.

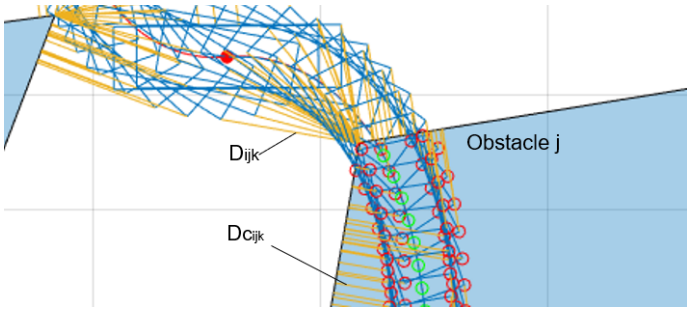


Figure 4. Graphical representation of metrics components.

3.2. Optimization Strategy

To determine the optimal trajectory of the automated guided vehicle (AGV), an iterative computational procedure is proposed (Figure 5). The metaheuristic method gives the output angles of the intermediate points θ_i. These optimized output angles θ_i, along with the coordinates of the intermediate points (x_i, y_i) and the coordinates and angles of the start and end points (x_s, y_s, θ_s) and (x_e, y_e, θ_e), are used to generate the trajectory described in Section 3.1. The trajectory is tested with the obstacle list to assess distances to any obstacle D and the collision checking D_c. This information is considered by the fitness function f_c to evaluate the solution quality. Then, the metaheuristic method receives the fitness function value to generate a new set of angles θ_i and the iterative process continues.

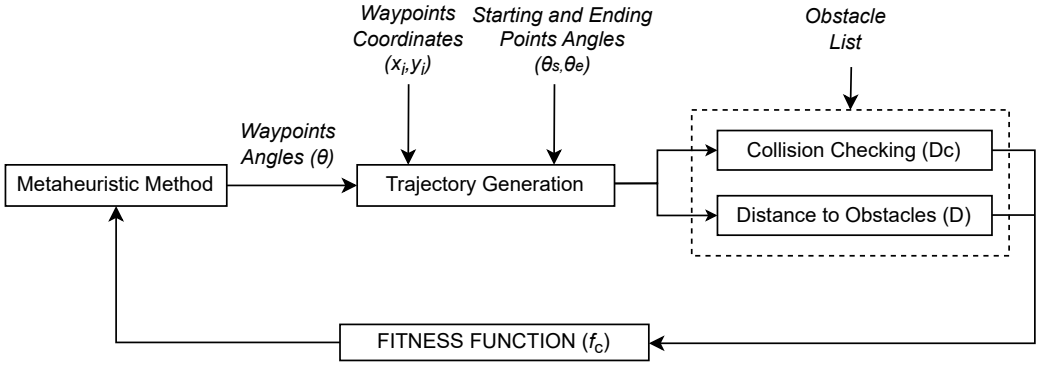


Figure 5. Optimization strategy.

The main parameters of this optimization problem are summarized below.

- Constraints and fixed parameters:
  - a. Coordinates of the start point: (x_s, y_s)
  - b. Output angle of the start point: θ_s
  - c. Coordinates of the end point: (x_e, y_e)
  - d. Input angle of the end point: θ_e
  - e. Coordinates of the way-points: $(x_i, y_i) \in \mathbb{R}^2\ i \in \{\mathbb{N} \leq N_w\}$
  - f. Physical dimensions of the AGV
- Optimization variables:
  - g. Output angles of the way-points: $\theta_i \in [0, 2\pi]\ i \in \{\mathbb{N} \leq N_w\}$

A dual fitness function has been defined to evaluate the best solutions in presence and absence of collisions. The function is piece-wise and has two components. The first component calculates the total invaded distance of the collision space for collision scenarios, while the second component works with the average distance to the obstacles for collision-free scenarios. To ensure continuity of the piece-wise function, both components are normalized to 1. The fitness function is expressed formally as follows:

$$f_c = \begin{cases} 1 + \sum_{i=1}^{n} min_{j \in N_o, k \in 1...4} D_{c_{ijk}} & \text{if } D_c \neq 0 \\ 1 - \dfrac{\sum_{i=1}^{n} min_{j \in N_o, k \in 1...4} D_{ijk}/n}{max_{j \in N_o} D_{ijk}} & \text{if } D_c = 0 \end{cases} \quad (7)$$

where n is the number of points of the trajectory, N_o is the number of obstacles in the occupancy map, D_{ijk} represents the distance from the vehicle's vertex k in the trajectory point i to the obstacle j when there is no collision, and $D_{c_{ijk}}$ is the distance when a collision occurs. Dc is the sum of distances to all obstacles in a collision. Thus, when there is no collision ($Dc = 0$), the fitness function aims to maximize the distance to obstacles. On the other hand, when collision occurs ($D_c \neq 0$), the fitness function is designed to minimize the length of the trajectory inside the obstacles.

This optimization strategy can be synthesized and formalized with the Algorithm 1.

Algorithm 1 Optimization algorithm

Require: $N_w \geq 0, n > 0, |list_{obs}| = N_o \geq 0, W > 0, L > 0$
  $f_c \leftarrow \infty$
  while $iter \leq iter_{MAX}$ **do**
    $\{\theta_i\} \leftarrow f_{opt}(f_c),\ i \in \{\mathbb{N} \leq N_w\}$
    $t_{frenet} \leftarrow f_{frenet}(\theta_i, (x_e, y_e), (x_s, y_s), \{(x_i, y_i)\}),\ i \in \{\mathbb{N} \leq N_w\}$
    $\left\{D_{ijk}\right\} \leftarrow get_{dis}(t_{frenet}, list_{obs}, W, L),\ i \in \{\mathbb{N} \leq n\}, j \in \{\mathbb{N} \leq N_o\}, k \in \{\mathbb{N} \leq 4\}$
    $\left\{D_{c_{ijk}}\right\} \leftarrow get_{col}(t_{frenet}, list_{obs}, W, L),\ i \in \{\mathbb{N} \leq n\}, j \in \{\mathbb{N} \leq N_o\}, k \in \{\mathbb{N} \leq 4\}$
    $D_c \leftarrow \sum_{i=1}^{n} \sum_{j=1}^{N_o} \sum_{k=1}^{4} D_{c_{ijk}}$
    if $D_c \neq 0$ **then**
      $f_c \leftarrow 1 + \sum_{i=1}^{n} min_{j \in N_o, k \in 1...4} D_{c_{ijk}}$
    else
      $f_c \leftarrow 1 - \dfrac{\sum_{i=1}^{n} min_{j \in N_o, k \in 1...4} D_{ijk}/n}{max_{j \in N_o} D_{ijk}}$
    end if
  end while

$list_{obs}$ is the set of the obstacles, f_{opt} denotes the function executed by the metaheuristic algorithm, $t_{frenet} \in \mathbb{R}^{3n}$ denotes the Frenet trajectory, get_{dis} denotes the function to calculate the distances to the obstacles, and get_{col} indicates the function which computes the collision distances.

3.3. Evaluation Metrics

- Non-collision time (NCT): The NCT measures how long it takes the optimization algorithm to identify a solution that meets the safety condition of no collisions.
- Best-solution time (BST): The BST time measures the duration it takes the optimization algorithm to pinpoint the trajectory that achieves the highest average distance from obstacles, representing the most favorable outcome.
- Minimum distance to obstacles (MDO): The MDO is the minimum distance that the vertices of the AGV will have to the obstacles of the occupation map. The calculation involves determining the minimum distance to obstacles for each vertex of the vehicle along the trajectory, and selecting the smallest value from this set of values. When there is a collision registered, the MDO value is zero. Formally, this value is given by Equation (8).

$$MDO = min_{i \in 1...n,\ k \in 1...4}(min_{j \in N_o} D_{ijk}) \quad (8)$$

Being n the number of trajectory points and N_o the number of obstacles in the occupancy map. D_{ijk} denotes the distance from the vehicle's vertex k in the trajectory point i to the obstacle j when there is no collision.

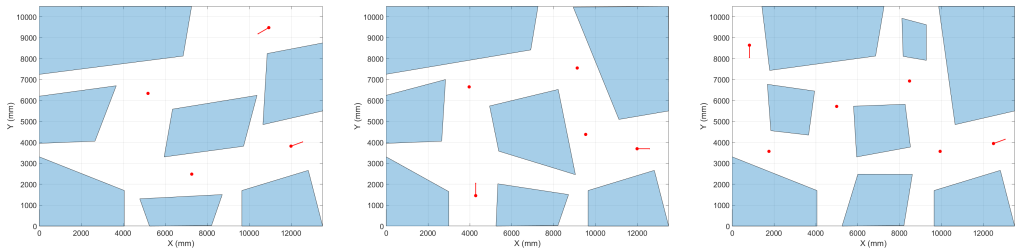
- Average distance to obstacles (ADO): The average value of the set of minimum distances to obstacles from the vertices of the vehicle along the trajectory (ADO) is formally expressed in Equation (9). This value analyses the average distance of the trajectory to the occupation map, thus maximizing this metric means maximizing the anti-collision safety conditions.

$$ADO = \frac{\sum_{i=1}^{n} min_{j \in N_o, k \in 1...4} D_{ijk}}{n} \tag{9}$$

3.4. Evaluation Scenarios

Three simulation scenarios were meticulously crafted to assess the effectiveness of the optimization algorithms. All scenarios share dimensions of 13,500 × 11,000 mm and are classified based on their complexity (low, medium and high) indicating varying levels of intricacy and challenge. The complexity is derived from both the quantity and strategic positioning of obstacles within the occupancy map. Figure 6 illustrates these scenarios, depicting the starting and ending points of each trajectory, as well as their respective output angles. The low-complexity scenario features a sparse distribution of obstacles, creating a relatively straightforward environment for trajectory planning. In contrast, the medium-complexity scenario introduces a moderate increase in obstacle density, demanding a more intricate approach to trajectory planning. Finally, a more complicated waypoint positioning within the obstacle arrangement is presented in the high-complexity scenario. These complexity levels are further characterized by the inclusion of a greater number of intermediate points, progressively elevating the optimization process's complexity for the selected methods.

In terms of design, the occupancy map of each scenario was constructed based on a conceptual framework that emphasizes the strategic placement of blocks and waypoints to find a trajectory through a network of obstacles. The idea behind this design approach is to emulate different environments, allowing a comprehensive evaluation of the optimization algorithms across a spectrum of complexity. In Figure 6, the images represent occupancy maps, where blue polygons denote obstacles and white spaces indicate free areas within the map. The inclusion of start and end points, along with small arrows indicating the exit angle of the trajectory at these points, provides a visual representation of the experimental setup. Taken together, these elements serve as important visual aids, providing a clear understanding of the scenarios and facilitating the interpretation of the performance of the optimization algorithm in different environmental conditions.



(**a**) (**b**) (**c**)

Figure 6. Graphical representation of the three scenarios considered for optimization experiments. (**a**) Low complexity; (**b**) medium complexity; (**c**) high complexity.

Throughout the experimental evaluations, a specific guided automated vehicle (AGV) is used. The AGV has some specific dimensions, 1800 mm length and 550 mm width. This experimental setup allows for a comprehensive exploration of the algorithms' adaptability and performance under varying conditions.

3.5. Configuration of Metaheuristic Techniques

In this article, three different metaheuristic optimization methodologies are used to compare their performance, advantages, and disadvantages: Genetic algorithms, particle swarm optimization and pattern search. The configuration of each of them is as follows.

3.5.1. Genetic Algorithms

Genetic algorithms are used to optimize the output angles of the intermediate points as tuning variables. Initial output angles are randomly generated subject to the constraints previously established.

The population size is 50 individuals. The crossover operator randomly selects two individuals (parents) and averages the corresponding variables of the parents to create offspring. The averaging process introduces a random weight for each variable, enhancing the diversity of the new individual.

Additionally, boundary mutation is adopted. This operator introduces random modifications to the selected variables of a solution within a range, in this case, 20% of the total variable range. Individuals are then selected to generate a new population through a combination of proportional fitness selection and elitism. The selection probability for each individual is determined by normalizing their fitness values. At each generation, two elite individuals, representing the best solutions of the current generation, are directly included in the next generation. All these parameters are summarized in Table 1.

Table 1. Genetic algorithm parameters.

Parameter	Value
Individuals per generation	50
Genetic Operator	Crossover & Mutation
Chromosomes	Number of intermediate waypoints
New population selection	Hybrid (Proportional and Elitism)

3.5.2. Particle Swarm Optimization

The particle swarm optimization algorithm begins by creating the initial particles and assigning them initial velocities. The fitness function is evaluated at each particle location, determining the best value (lowest) and the best location. Subsequently, new velocities are selected based on the current velocity, the best individual locations of the particles, and the best locations of their neighbors.

The particle locations are iteratively updated, with the new location being the old one plus the velocity, modified to keep the particles within the specified limits of velocities and neighbors. In this case, the limits are set between 0 and 360 degrees, which represent the possible output angles for the intermediate points of the trajectory. Additionally, an inertia parameter of 0.5 is used. All these parameters are summarized in Table 2.

Table 2. Particle swarm parameters.

Parameter	Value
Number of particles per swarm	50
New swarm particles Limits	[0, 360]
Inertial Parameter	0.5

3.5.3. Pattern Search

Pattern search involves the iterative exploration of a multidimensional search space to find the best solution, without requiring the calculation of gradients. Initially, a starting point is established in the search space, which in this work is randomly chosen within

the limits of the solution (output angle of the waypoint). Then, searching directions are generated and solutions are evaluated at nearby points. These search directions are scaled by the current mesh size, which is dynamically adjusted during the process. The evaluation of solutions on the mesh guides the update of the starting point and the generation of new search directions, continuing the process until it converges to the best solution.

To decide the initial mesh size and the contraction and expansion factors, various approaches can be considered. The initial mesh size can be adjusted based on the problem domain and the scale of the variables. In this work, an initial mesh size of 1 has been used, taking into account the optimization parameters used.

Regarding the contraction and expansion factors, small values of the contraction factor and large values of the expansion factor can lead to a broader exploration of the search space, while the opposite can focus the search on local regions. Based on experimentation and experience, contraction and expansion factors of 0.995 and 2, respectively, have been selected. All these parameters are summarized in Table 3.

Table 3. Pattern search parameters.

Parameter	Value
Initial Solution	Random in [0, 360]
Mesh Initial Size	1
Contraction Factor	0.995
Expansion Factor	2

3.6. Simulation Experiments

The experiments were conducted systematically to provide a comprehensive insight into the performance of the three optimization algorithms. Results plots and a detailed occupancy map were generated to visually represent the results of the optimization algorithms. Both the algorithms and the visualization figures were implemented and simulated in Matlab r2022a software.

Generations of 50 individuals were established for both the genetic algorithm and the particle swarm. For pattern search, an initial population was implemented by randomly distributing 360 values between 0 and 360. In this case, three different tests were carried out under identical conditions but with different initial populations.

For each scenario, specific conditions were established based on their complexity, which are consistent across all selected methods. The iterative process was carried out with the same fitness function, monitoring the results at each iteration. For each metaheuristic technique, a log file was generated with the following information (one row in the log was written per iteration): the timestamp, the corresponding generation, the individual number within the generation, the fitness function value, the angle value of the solution, the MDO and the ADO. This information allowed it to compare individual solutions among the metaheuristic techniques, as well as the evolution of the metrics during the optimization process.

The data were organized and saved in a CSV file, with the evolution of the metrics throughout the process so as to be able to analyze the trajectories generated. It allows not only to measure the quality of the final optimal solution but also the continuous evaluation of all intermediate solutions. This methodology gives a detailed view of the dynamics of the algorithm throughout the iterations, facilitating the analysis and understanding of the effectiveness of the method in finding optimal solutions to the path optimization problem.

The evaluation process involves the analysis of the collected metrics, including timestamp, generation, individuals, fitness function, minimum and average distances to obstacles and the time required to achieve collision-free and best solutions.

4. Results

This section compares graphically and numerically the results obtained by the application of the three different metaheuristic techniques to the AGV trajectory generation

problem. To obtain the results, the Matlab r2022a software has been used, along with the Global Optimization 4.7 and Navigation 2.2 toolboxes. The first allows you to run genetic algorithms, pattern search and particle swarm algorithms. On the other hand, the Navigation toolbox has been used to generate the Frenet routes. The representation of the trajectory of the best solution found with each technique and the evolution of the comparison metrics during the optimization process is shown.

The figures that show the trajectory of the best solution use the following code. Red lines indicate the solution trajectories, with red dots representing the input parameters (start point, end point, and intermediate points). The AGV at each point of the trajectory is shown in blue. Additionally, yellow lines indicate the minimum distance between the vehicle vertices and the obstacles in the occupancy map.

On the other hand, in the figures of the evolution of each metric, blue lines represent the results of the GA, red lines are the results of the PS, and yellow lines are the results of the PSO technique. As the simulation experiments are quite long a zoom has been applied to better visualize the first part of the optimization process where more changes occur.

4.1. Low Complexity Scenario

To define the trajectories, a low-density occupancy map has been first used, with seven obstacles in a small space. Additionally, three intermediate points are used along with the start and end points. The output angles of the start and end points are given, while the output angles of the intermediate points are considered the optimization variables. The maximum optimization time to obtain the solutions has been set to 3600 s, that is, one hour.

Figure 7 shows the optimal trajectories for the three metaheuristic methods, GA, PS and PSO. As shown in Figure 7a,c, the GA and PSO methods successfully find a collision-free solution within the given time, while PS is unable to do so as shown in Figure 7b. Due to its mode of operation, if the initial conditions are far from the solution, the system may not be able to reach it because it is a local optimization technique.

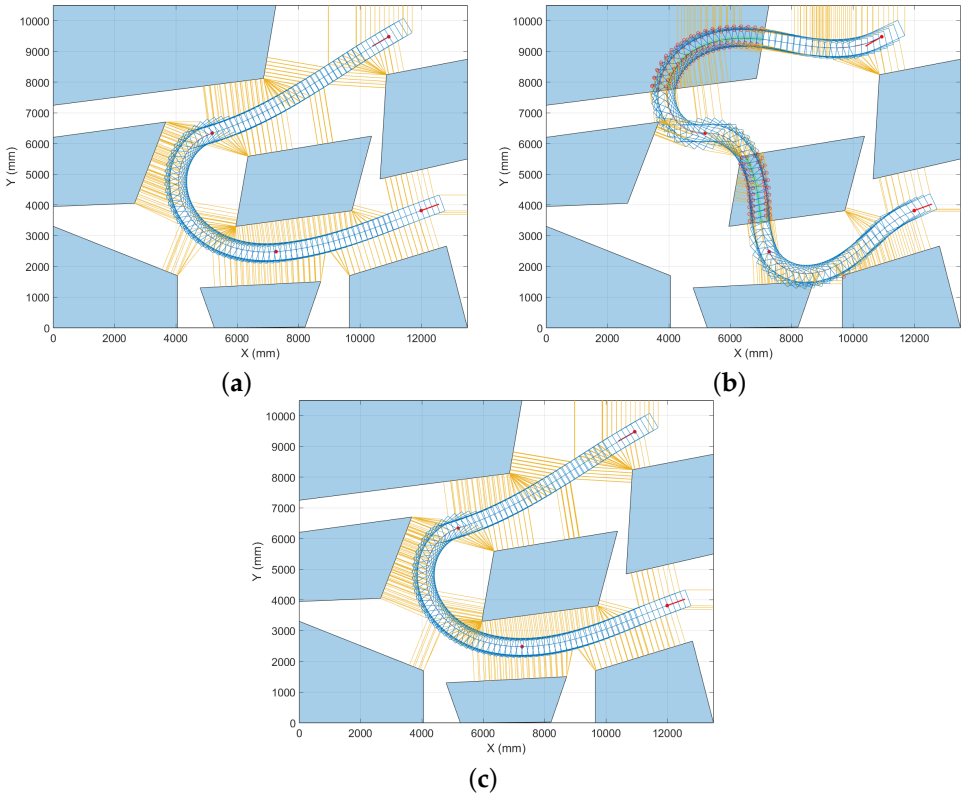


Figure 7. Low complexity scenario, optimized trajectories. (**a**) Genetic algorithm. (**b**) Pattern search. (**c**) Particle swarm.

Figures 8–10 show the results obtained. Figure 8 shows the evolution of the best fitness function at each iteration for each technique during the optimization process. The time indicates the elapsed time between the beginning of the optimization and the time when the solution was found. Although GA and PSO reach similar final values, GA advances faster in the optimization process, reaching feasible trajectories without collision more quickly. PS not only fails to find a viable solution to the problem but also takes longer to improve results as iterations progress.

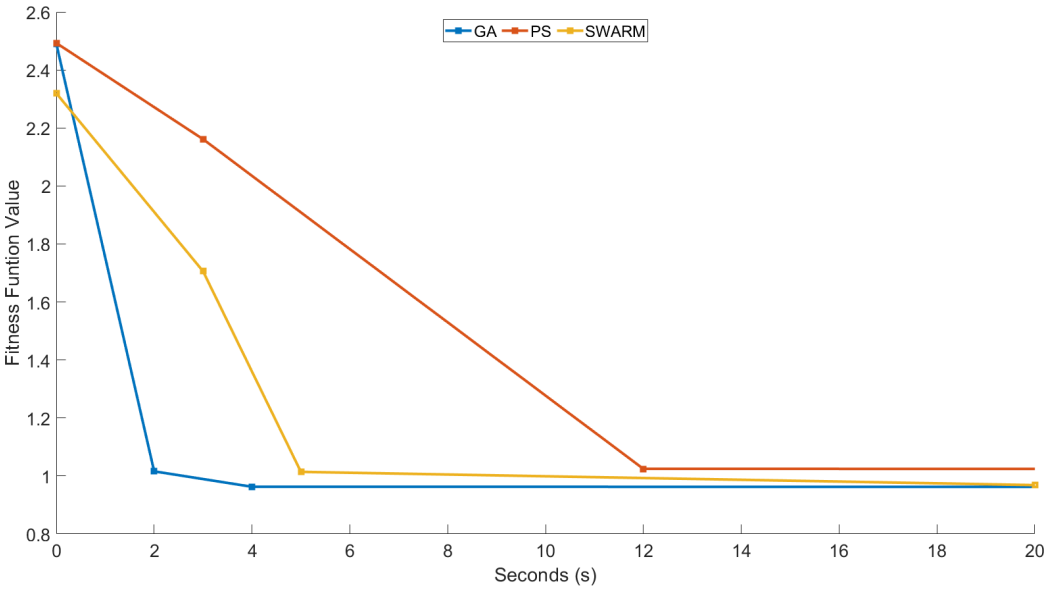


Figure 8. Optimization methods, fitness function evolution in the low complexity scenario.

Figure 9 shows the evolution of the ADO. As expected, this value increases when the fitness function reaches values smaller than 1, indicating the point at which the trajectory enters the non-collision zone and moves away from obstacles as far as possible. When PS is not in a non-collision state, the mean value is much lower than with the other techniques.

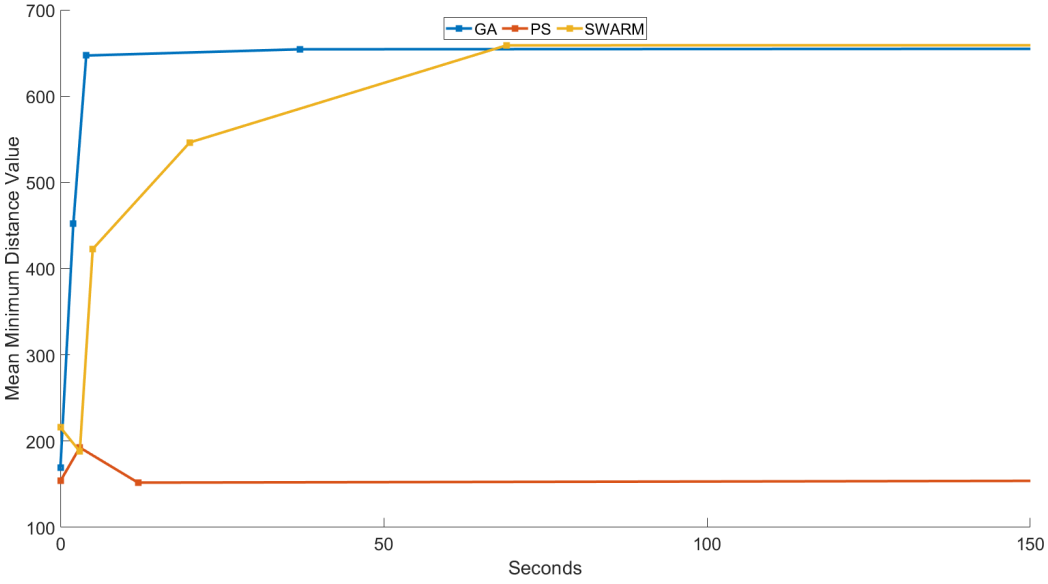


Figure 9. Optimization methods, ADO evolution in the low complexity scenario.

Figure 10 shows the evolution of the MDO. It is possible to see how the MDO for PS is always zero. As anticipated, since these iterations always involve collisions, the minimum distance to obstacles for PS will be always zero. In the case of GA and PSO, it can be observed that, although the trajectories obtained seem to be practically identical, there is

a slight difference in the minimum values of the distance to obstacles obtained by each algorithm. In this case, GA maintains a greater minimum distance in its final solution than PSO.

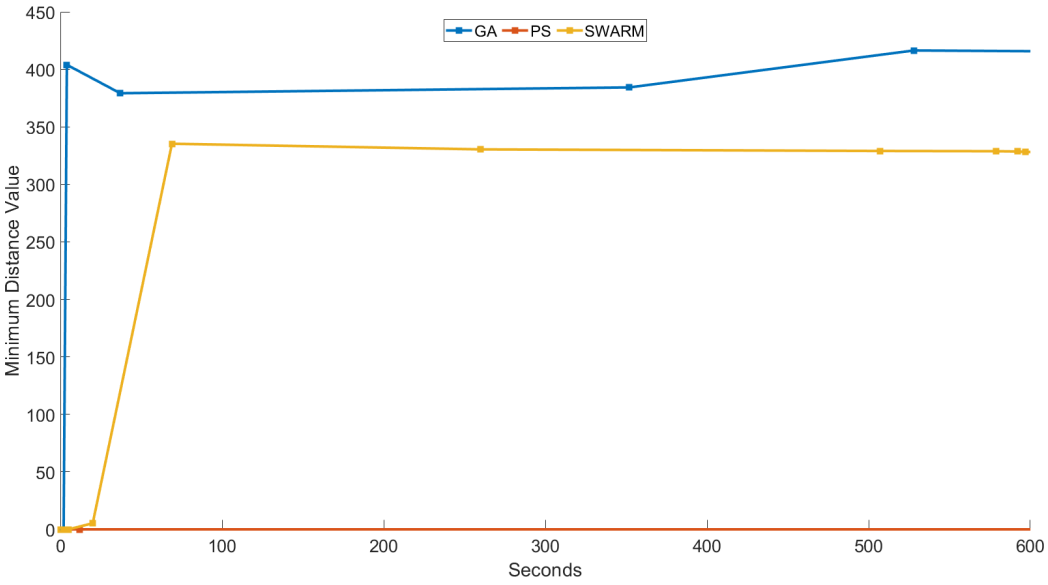


Figure 10. Optimization methods, MDO evolution in the low complexity scenario.

Tables 4 and 5 list all analytical results shown in Figures 8–10. It includes data for the first iteration where a collision-free solution was obtained, as well as the iteration with the best solution during the optimization process.

Table 4. Low complexity scenario, first non-collision iteration analytic results.

Technique	Non-Collision Time (s)	Non-Collision Iterations	ADO [mm]	MDO [mm]	Fitness Function
GA	4	3	647.07	403.99	0.9628
PSO	20	12	546.26	5.30	0.9686
PS	-	-	-	-	-

Table 5. Low complexity scenario, best solution analytic results.

Technique	Best Solution Time (s)	Best Solution Iterations	ADO [mm]	MDO [mm]	Fitness Function
GA	1929	1051	660.33	404.22	0.9624
PSO	2052	1151	659.21	327.61	0.9621
PS	6482	3521	178.93	0	3.1648

4.2. Medium Complexity Scenario

An occupancy map with intermediate obstacle density is also used, defined with seven obstacles and larger occupancy space than the previous case, with four intermediate points in addition to the starting and end points of the trajectory. The maximum optimization time to obtain the solutions is set to 7200 s.

For each optimization technique, the best trajectories are shown in Figure 11. The GA and PSO effectively find a collision-free solution within the allotted time (Figure 11a,c); however, PS is unable to do so as shown in Figure 11b. As it happened in the previous scenario, because of its randomization if the starting conditions are far from the final solution, it might not be able to obtain a feasible solution.

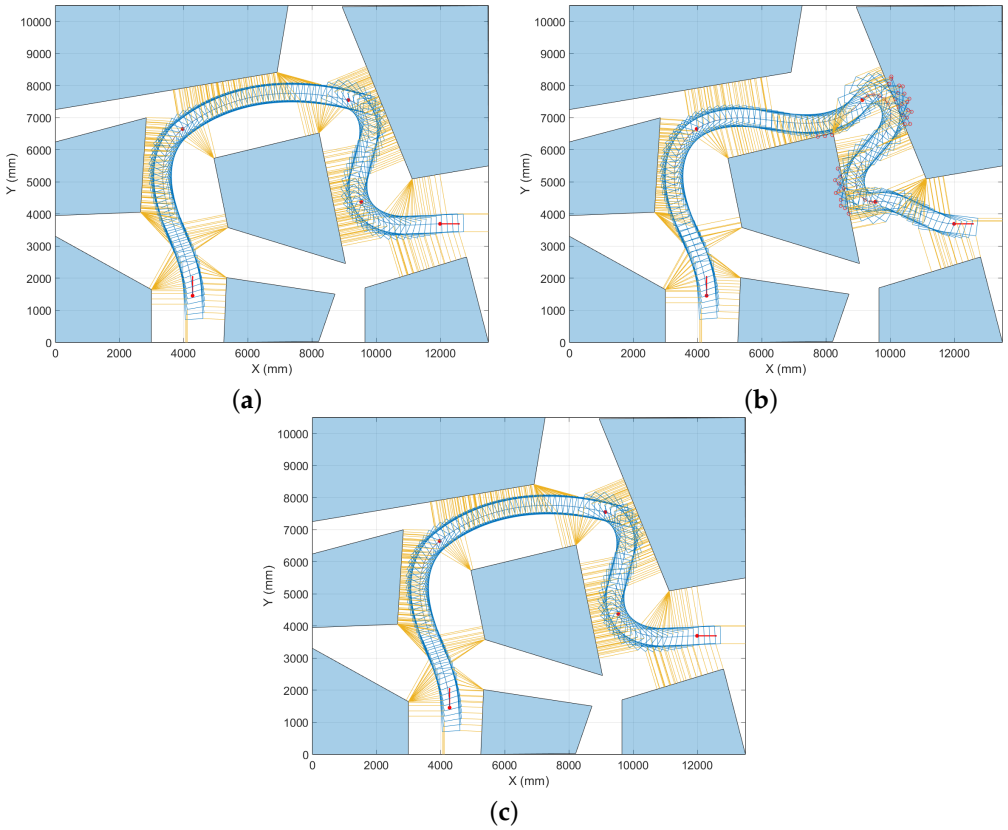


Figure 11. Medium complexity scenario, optimized trajectories. (**a**) Genetic algorithm. (**b**) Pattern search. (**c**) Particle swarm.

The evolution of the fitness function for each approach during the optimization process is shown in Figure 12. GA and PSO arrive at a similar final value and trajectory, and the optimization process is approximately equally fast in reaching feasible paths without colliding, although GA gets a better optimization value sooner. As iterations go on, PS not only is unable to come up with a workable solution, but it also takes longer to get better outputs.

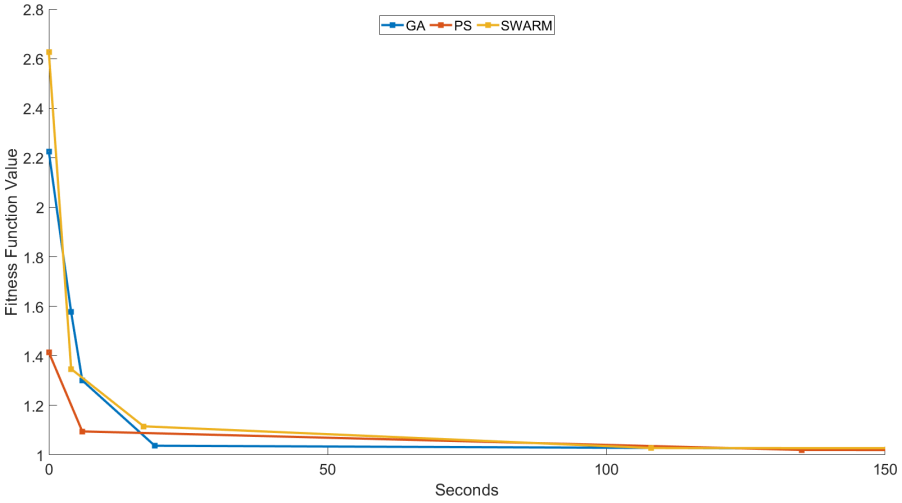


Figure 12. Optimization methods, fitness function evolution in the medium complexity scenario.

The evolution of the ADO is shown in Figure 13. As the fitness function gets values smaller than 1, this value rises, meaning the moment at which the trajectory leaves the collision zone and travels as far away from obstacles as it can. Previous to that, the values can get higher or lower given the colliding solutions the different iterations can reach.

The mean value for PS is significantly smaller than with the other methods due to its inability to reach a feasible trajectory.

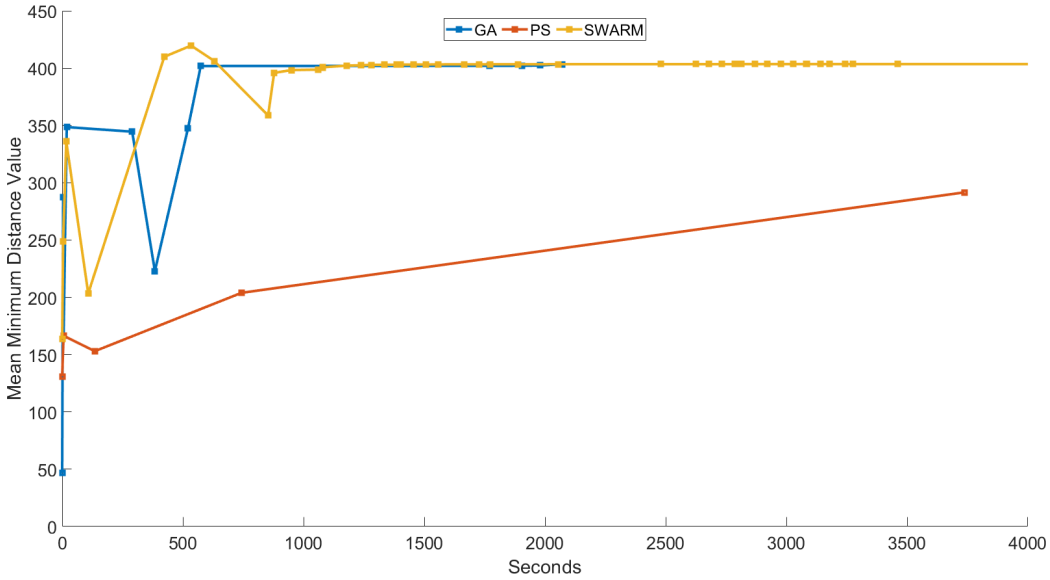


Figure 13. Optimization methods, mean minimum distance evolution in the medium complexity scenario.

Figure 14 shows the MDO. Again, it is possible to see that the MDO for PS is always zero. As expected, the minimum distance to obstacles in this situation will always be zero since these iterations always produce collisions. It is evident that in the case of GA and PSO, the minimum distances to obstacles during the optimization differ widely, taking some time for the algorithms to stabilize and converge into more similar values as established in Tables 6 and 7.

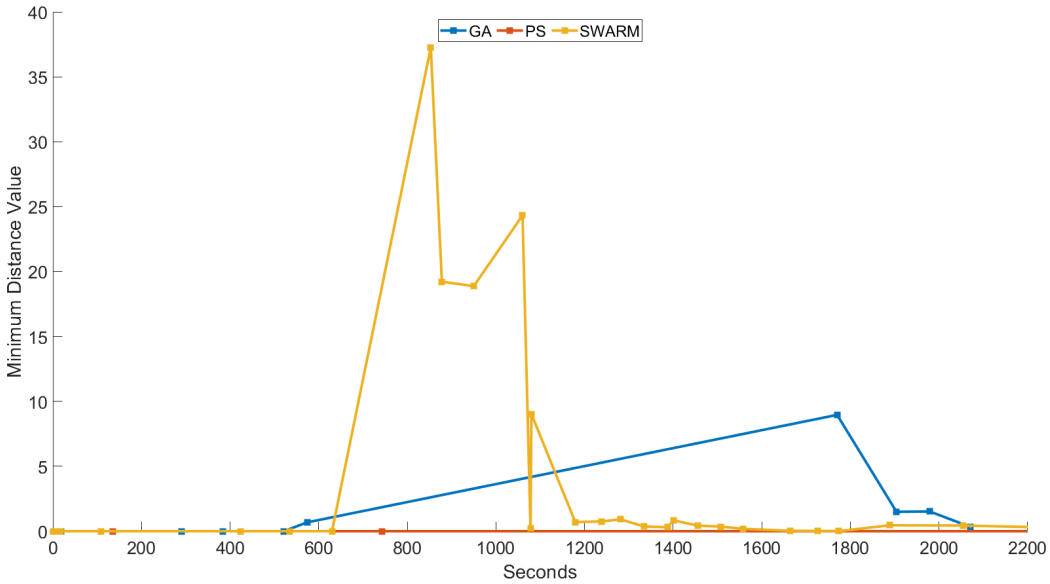


Figure 14. Optimization methods, MDO evolution in the medium complexity scenario.

Analytical data from Figures 12–14 are shown in Tables 6 and 7. It contains information from both the iteration with the best solution at the end of the optimization process and the initial iteration where a collision-free solution was found.

Table 6. Medium complexity scenario, first non-collision iteration analytic results.

Technique	Non-Collision Time (s)	Non-Collision Iterations	ADO [mm]	MDO [mm]	Fitness Function
GA	574	312	401.85	0.67	0.9769
PSO	853	462	358.70	37.26	0.9793
PS	-	-	-	-	-

Table 7. Medium complexity scenario, best solution results.

Technique	Best Solution Time (s)	Best Solution Iterations	ADO [mm]	MDO [mm]	Fitness Function
GA	2070	1152	403.42	0.33	0.9768
PSO	12,196	6984	403.51	0.004	0.9768
PS	3738	2095	291.63	0	1.1078

4.3. High Complexity Scenario

A map of occupancy is defined with a high density of obstacles. The scenario includes eight obstacles and the largest occupancy space. The trajectory includes five intermediate points in addition to the start and end points. The maximum optimization time to obtain the solutions is set to 9600 s, that is, 160 min.

Figure 15 shows the optimal trajectories for the three optimization strategies. In this case, all methods were able to reach a solution without collisions. This result shows the sensitivity of the PS technique to the initial conditions and randomness. Indeed, as shown in this case, depending on the initial conditions it may give good solutions in complex scenarios meanwhile it was not able to obtain solutions in simpler cases.

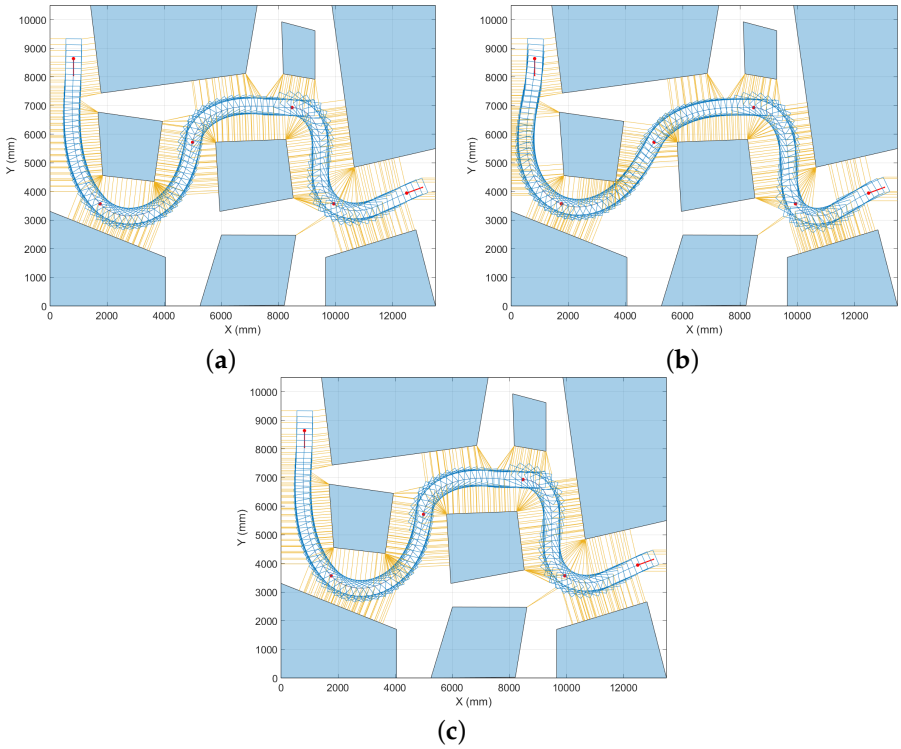


Figure 15. High complexity scenario, optimized trajectories. (**a**) Genetic algorithm. (**b**) Pattern search. (**c**) Particle swarm.

Figure 16 shows the evolution of the fitness function for each strategy along the optimization process. This scenario demonstrates that all methods have found feasible

solutions, although with small differences. This is evident in the evolution of the fitness function, which tends to be below 1, indicating success in finding collision-free solutions. It is noteworthy that the GA was the fastest in finding the optimal solution. Additionally, this case highlights that PS is faster than PSO. Therefore, it can be concluded that when the initial conditions of PS are suitable, it is faster than PSO but not faster than GA. This highlights the importance of setting up PS with initial conditions aligned as closely as possible with the final solution. PS should not be used in cases where the initial conditions do not seem to have any relation to the possible final solution.

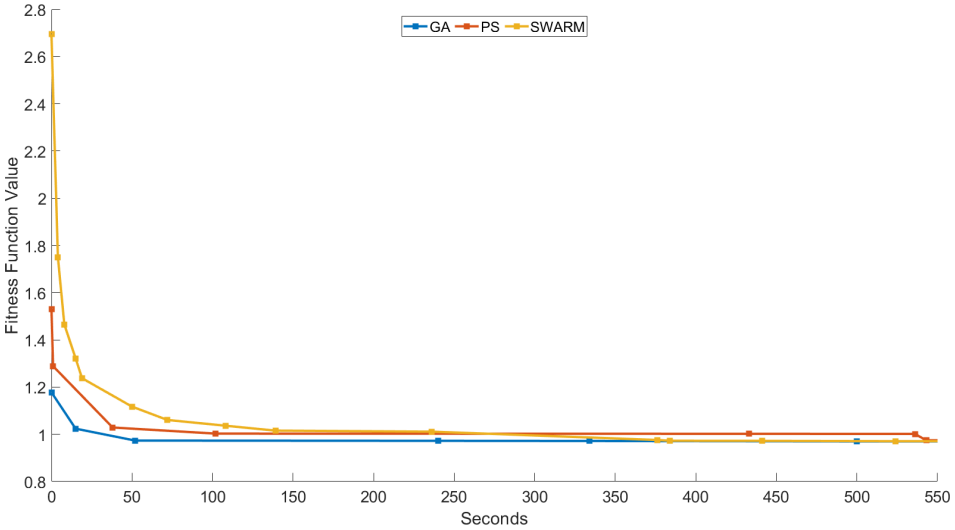


Figure 16. Optimization methods, fitness function evolution in the high complexity scenario.

As with the other results, Figure 17 shows the ADM fluctuation until the methods find a collision-free solution. Afterwards, the average value increases in line with the improvement of the fitness function. GA and PSO exhibit similar behavior, with GA initially achieving better results without collisions. However, at around 2000 s, PSO begins to obtain better solutions more quickly. Similarly, while PS gives a collision-free solution during the process, subsequent improvements to the solutions are significantly smaller, resulting in a lower-quality solution in terms of this parameter.

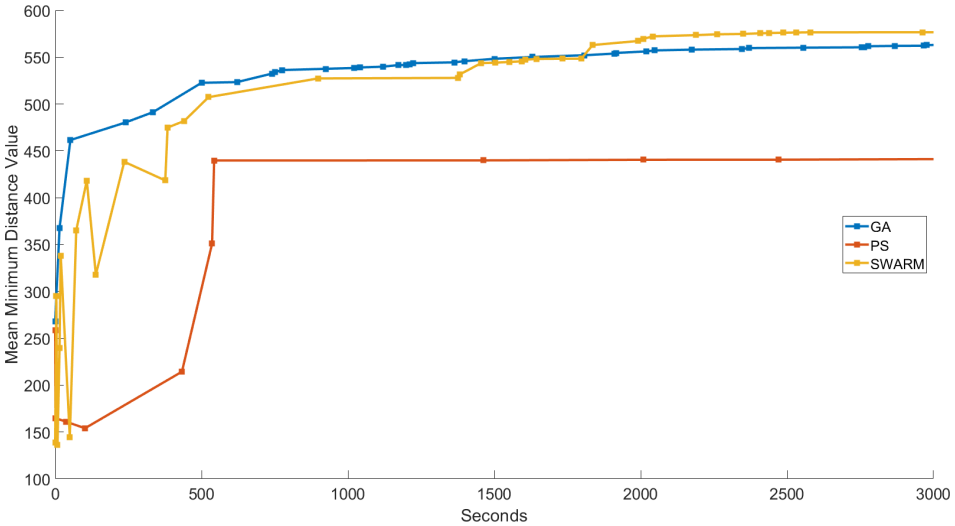


Figure 17. Optimization methods, ADO evolution in the high complexity scenario.

Figure 18 shows the MDO evolution for the most complex scenario. The plot above represents the function over simulation time, while the plot below emphasizes the area near

the start to provide a clearer visualization of that part of the response. In this case, PS does not have zero distance. Instead, it shows a saw-tooth-like evolution with an upward trend. Although the minimum distance has a positive value, it remains lower than the values achieved by GA and PSO. These last two solutions converge to a very similar minimum value, although GA initially achieves better values during the iterative process.

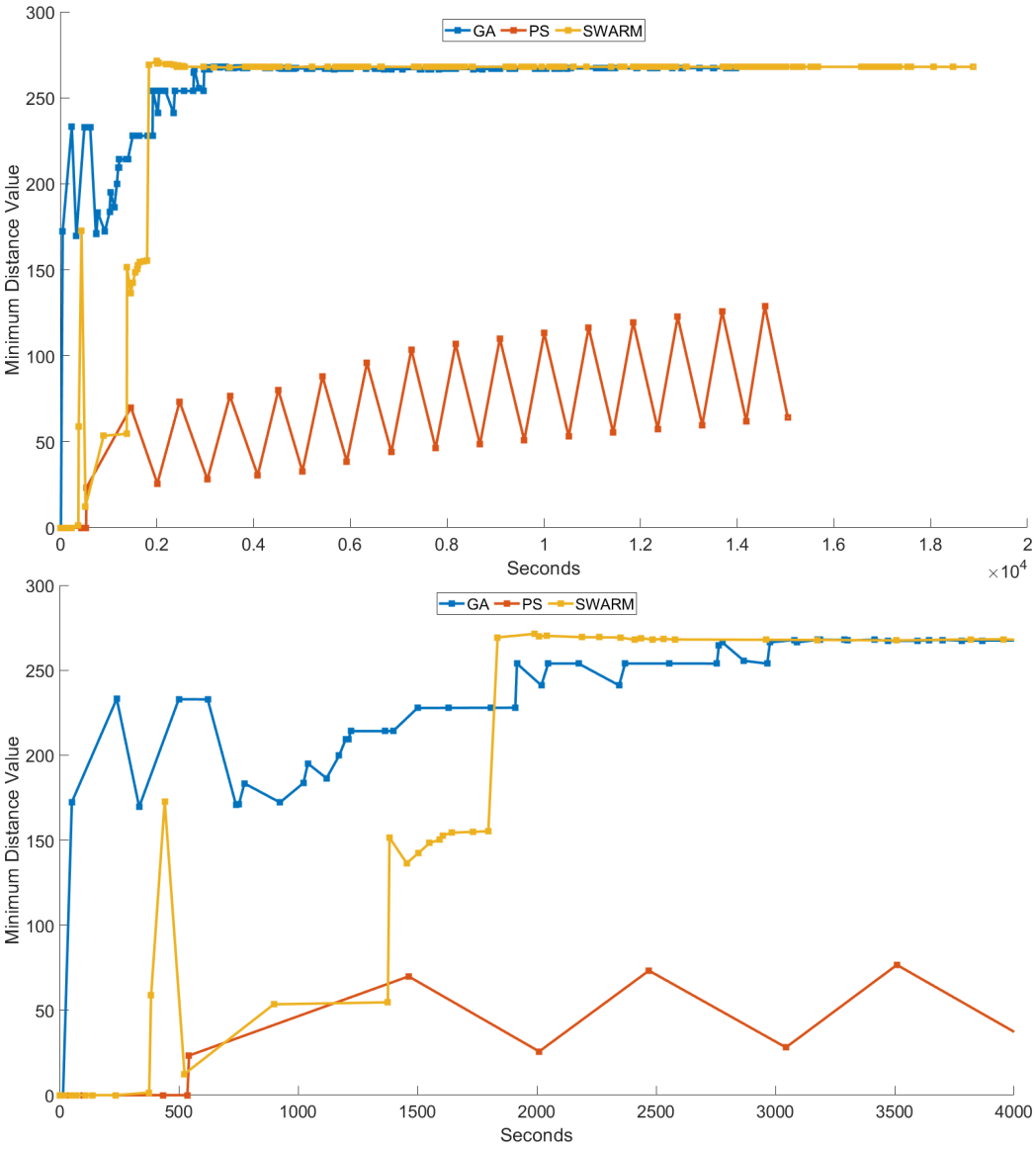


Figure 18. Optimization methods, MDO evolution in the high complexity scenario.

Tables 8 and 9 present the analytical values obtained, indicating the first collision-free solution and the best value of the whole series.

Table 8. High complexity scenario, first non-collision iteration analytic results.

Technique	Non-Collision Time (s)	Non-Collision Iterations	ADO [mm]	MDO [mm]	Fitness Function
GA	52	25	461.53	172.46	0.9734
PSO	376	172	418.44	1.50	0.9759
PS	543	661	439.73	23.27	0.9747

Table 9. High complexity scenario, best solution analytic results.

Technique	Best Solution Time (s)	Best Solution Iterations	ADO [mm]	MDO [mm]	Fitness Function
GA	13,970	6818	576.66	267.23	0.9668
PSO	18,874	9492	576.81	267.98	0.9668
PS	15,044	19,838	452.49	64.08	0.9739

Finally, Figure 19 compares the trajectories of the first solution without collision (blue lines) and the best trajectories (red lines) of the three optimization techniques. It is possible to see that in all cases there are significant differences between these trajectories.

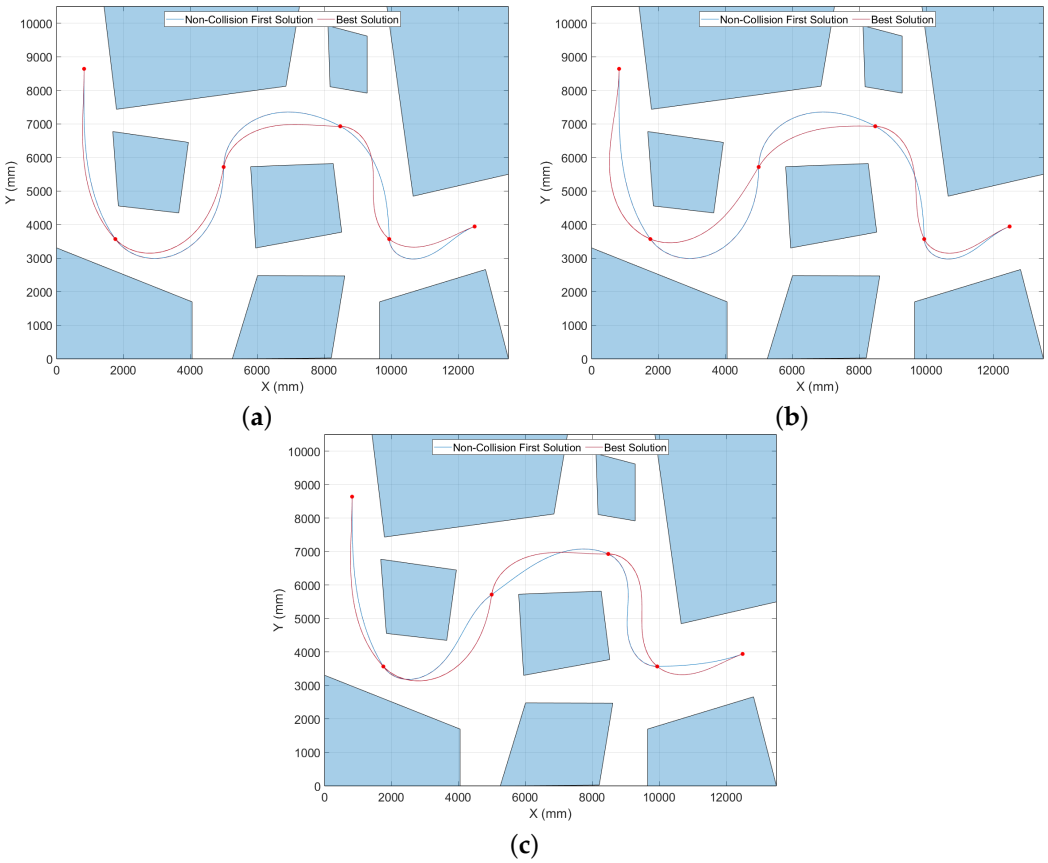


Figure 19. High complexity scenario, first non-collision solution and best solution trajectories. (**a**) Genetic algorithm. (**b**) Pattern search. (**c**) Particle swarm.

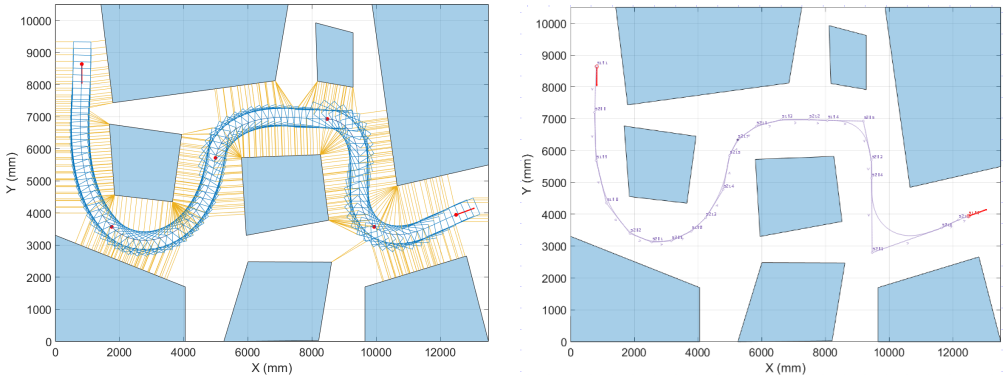
Although the first collision-free trajectories are feasible, during the iterative process the improvement of the trajectories with the best fitness function in terms of safety is notable. The largest possible distance to the obstacles is considered the safest solution. This finding shows the effectiveness of the method in improving safety, as the optimal solution manages to maintain the largest average distance from obstacles.

4.4. Experimental Results

In order to demonstrate that the optimized trajectories can be followed by an AGV, an experimental test has been carried out with an industrial AGV of the company ASTI Mobile Robotics. The experimentation was conducted in a laboratory using a tow AGV equipped with simultaneous location and mapping (SLAM) navigation. The best trajectory obtained in the high-complexity scenario has been configured in the navigation software of the AGV. This trajectory has been selected considering that is the longest and the hardest to

follow. We have observed that the AGV smoothly follows the trajectory and does not enter any of the areas delimited by the obstacles.

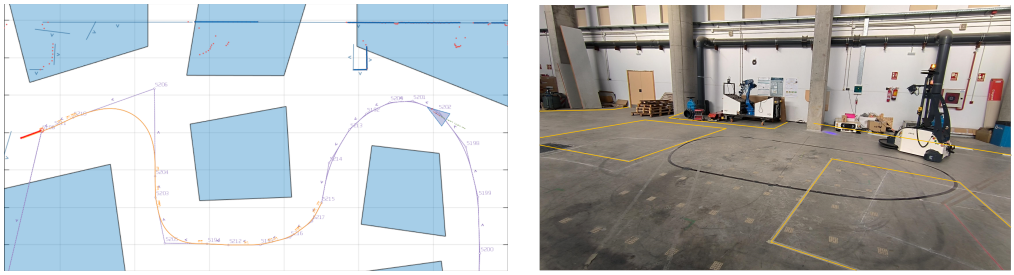
Figure 20b illustrates the implemented trajectory in the AGV's navigation software. This visualization overlays the real trajectory on top of the graphical representation of the occupancy map. Figure 20b shows the same trajectory obtained in MATLAB. It is possible to see that both trajectories are equivalent.



(**a**) (**b**)

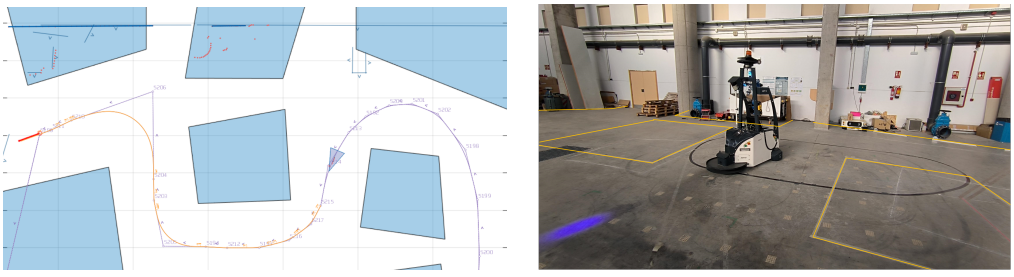
Figure 20. High complexity GA trajectory result. (**a**) Matlab representation. (**b**) Navigation software implementation.

Figures 21–23 shows snapshots taken while the AGV follows the trajectory in the laboratory. These images show the correspondence between the real position of the AGV and its representation in the navigation software at specific moments. To improve visualization consistency with laboratory test images, the trajectories in the navigation software are rotated 180 degrees to match the perspective of the corresponding laboratory images. To improve visual clarity, we have added yellow lines to all laboratory images. These lines depict the boundaries of the obstacles in the occupancy map on the laboratory floor, providing a clearer visual reference in an environment with multiple marked lines.



(**a**) (**b**)

Figure 21. High complexity GA trajectory result in laboratory experiment. First captured moment: (**a**) Position representation in navigation system. (**b**) Laboratory representation.



(**a**) (**b**)

Figure 22. High complexity GA trajectory result in laboratory experiment. Second captured moment: (**a**) Position representation in navigation system. (**b**) Laboratory representation.

(**a**) (**b**)

Figure 23. High complexity GA trajectory result in laboratory experiment. Third captured moment: (**a**) Position representation in navigation system. (**b**) Laboratory representation.

The experimental results demonstrate the feasibility of implementing these trajectories with industrial AGVs. The observations prove that the evaluated optimization techniques have practical applicability in real-world scenarios, as the AGV perfectly follows the trajectory configured. These experimental findings represent a significant step towards the practical validation of the assessed optimization methods.

4.5. Discussion of the Results

Based on the results obtained, several conclusions can be drawn about the usefulness of these three optimization methods to generate AGV trajectories. The first deduction is that all three methods can be used to obtain optimal trajectories that AGVs can follow.

The time needed to obtain the first collision-free solution tends to grow with the complexity of the scenario. It is just a tendency due to the randomness implicit in the generation of solutions, so it does not always happen that way. A similar trend is also observed for the time at which the best solution is found.

GA and PSO techniques consistently produce acceptable results in terms of collision-free trajectories, which typically deviate from obstacles as much as possible. PS depends significantly on the initial conditions of the optimization process, producing feasible results when these conditions are very close to the final result, but infeasible paths (including collisions) when these conditions are far from the solution. Therefore, a good approach could be the hybridization of these methods: it is suggested to use GA and PSO to explore solutions in the entire search space when there is no initial information about the feasible solutions. On the other hand, PS can be very useful to refine solutions obtained by GA or PSO.

When the optimization process runs for a long time, GA and PSO tend to provide similar fitness function values. However, according to the experiments carried out, the fastest algorithm to produce the first feasible collision-free solutions is GA. Another interesting result is that if the PS has good initial conditions, the time needed to execute each iteration is considerably less than the time required by GA or PSO. For example, in the high complexity scenario, the average time to execute an iteration is 0.75 s for PS, 2 s for GA, and 1.98 s for PSO. However, when the initial conditions are far from the solution, the average execution times of the iterations are similar for all optimization techniques.

It is also possible to observe that the value of the fitness function during the first iterations is higher for the PSO, although it decreases rapidly during the first iterations. In fact, the largest changes in the fitness function in the initial iterations appear when PSO is applied. For example, in the medium complexity scenario, PSO outperforms GA during the first few iterations but then its rate of decline slows down as the iterations increase. Thus, in the tested cases, PSO never reached a feasible trajectory faster than GA. Therefore, GA stands out as the recommended method to optimize the generation of automated guided vehicle trajectories.

In general, it can be concluded that this route optimization approach can be an aid for the co-design of routes in real industrial scenarios, delineating them in a way that allows trajectories that are efficient for the AGV tasks.

5. Conclusions and Future Works

In this work, three metaheuristic optimization techniques have been compared to obtain collision-free trajectories of an AGV industrial vehicle. The results indicate that genetic algorithms (GA) generally outperform particle swarm optimization (PSO) and pattern search (PS) in finding optimal or collision-free trajectories in the given scenarios. GA tends to converge faster and achieve better solutions. Optimization with PS, while it can be effective in some cases, is sensitive to initial conditions and may have difficulty finding viable solutions. In terms of speed, GA is consistently faster in reaching feasible collision-free trajectories. PSO and PS show variability in their performance, with PSO occasionally outperforming PS. The simulation experiments carried out show that it is important to consider the specific characteristics and requirements of the optimization problem in question when choosing a metaheuristic technique. The selection of the optimization method may also depend on factors such as the nature of the search space and the availability of computational resources. These findings underline the importance of selecting an appropriate optimization algorithm based on the problem characteristics, and in this context, GA emerges as a reliable option for trajectory optimization in the given scenarios.

Based on the results obtained, there is potential to explore hybrid optimization approaches that combine different methodologies. This will allow exploiting the strengths of each method and improve the speed and accuracy of obtaining feasible solutions. The proposal is to use GA and PSO initially to acquire solutions quickly and efficiently, since both methods have proven effective in converging towards optimal solutions at different stages. These obtained solutions could serve as initial conditions to start the pattern search (PS) process and this way, to reach solutions closer to a global optimum. The combination of GA and PSO for initial solutions, followed by PS for refining and enhancing those solutions, could be particularly beneficial in scenarios where the speed of obtaining results is crucial. Furthermore, this strategy could contribute to better exploration of the search space and overcome potential stagnation points that each method might encounter individually.

In summary, the application of hybrid optimization techniques, taking advantage of the complementary characteristics of GA, PSO and PS, represents an interesting direction for future research, provided there is prior analysis. This approach could be especially valuable in applications that demand efficient and accurate results, approaching the prospect of implementing real-time methods for trajectory optimization and similar problems.

Author Contributions: Conceptualization, E.B., J.E.S.-G. and M.S.; Software, E.B. and J.E.S.-G.; Formal analysis, J.E.S.-G.; Investigation, E.B.; Writing—original draft, E.B., J.E.S.-G. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

References

1. Sierra-García, J.E.; Fernández-Rodríguez, V.; Santos, M.; Quevedo, E. Development and Experimental Validation of Control Algorithm for Person-Following Autonomous Robots. *Electronics* **2023**, *12*, 2077. [CrossRef]
2. Vagale, A.; Oucheikh, R.; Bye, R.T.; Osen, O.L.; Fossen, T.I. Path planning and collision avoidance for autonomous surface vehicles I: A review. *J. Mar. Sci. Technol.* **2021**, *26*, 1292–1306. [CrossRef]
3. Vis, I.F.A. Survey of research in the design and control of automated guided vehicle systems. *Eur. J. Oper. Res.* **2006**, *170*, 677–709. [CrossRef]
4. Schouwenaars, T.; How, J.P.; Feron, E. Receding horizon path planning with implicit safety guarantees. In Proceedings of the 2004 American Control Conference, Boston, MA, USA, 30 June–2 July 2004; Volume 6, pp. 5576–5581. [CrossRef]
5. Tamizi, M.G.; Yaghoubi, M.; Najjaran, H. A review of recent trend in motion planning of industrial robots. *Int. J. Intell. Robot. Appl.* **2023**, *7*, 253– 274. [CrossRef]
6. Sierra-Garcia, J.E.; Santos, M. Combining reinforcement learning and conventional control to improve automatic guided vehicles tracking of complex trajectories. *Expert Syst.* **2022**, *41*, e13076. [CrossRef]
7. Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278. [CrossRef]

8. Neto, A.; Canuto, A.; Xavier-Júnior, J. Hybrid metaheuristics to the automatic selection of features and members of classifier ensembles. *Information* **2018**, *9*, 268. [CrossRef]
9. Adekanmbi, O.; Green, P. Conceptual comparison of population based metaheuristics for engineering problems. *Sci. World J.* **2015**, *2015*, 936106. [CrossRef]
10. Mohammadi, R.; Ghasemof, A. Performance-based design optimization using uniform deformation theory: A comparison study. *Lat. Am. J. Solids Struct.* **2015**, *12*, 18–36. [CrossRef]
11. Blum, C.; Puchinger, J.; Raidl, G.; Roli, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput.* **2011**, *11*, 4135–4151. [CrossRef]
12. Puchinger, J.; Raidl, G.R. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 41–53. [CrossRef]
13. Ramírez, A.; Barbudo, R.; Romero, J. An experimental comparison of metaheuristic frameworks for multi-objective optimization. *Expert Syst.* **2021**, *40*, e12672. [CrossRef]
14. Radosavljevic, J. *Metaheuristic Optimization in Power Engineering*; Institution of Engineering and Technology: London, UK, 2018. [CrossRef]
15. Bozorg-Haddad, O.; Solgi, M.; Loáiciga, H.A. *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*, 1st ed.; Wiley: Hoboken, NJ, USA, 2017. [CrossRef]
16. Nesmachnow, S. An overview of metaheuristics: Accurate and efficient methods for optimisation. *Int. J. Metaheuristics* **2014**, *3*, 320. [CrossRef]
17. Kareem, S.W.; Hama Ali, K.W.; Askar, S.; Xoshaba, F.S.; Hawezi, R. Metaheuristic algorithms in optimization and its application: A review. *J. Adv. Res. Electr. Eng.* **2022**, *6*, 7–12. [CrossRef]
18. Demesure, G.; Defoort, M.; Bekrar, A.; Trentesaux, D.; Djemai, M. Decentralized Motion Planning and Scheduling of AGVs in an FMS. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1744–1752. [CrossRef]
19. Altché, F.; Fortelle, A.d.L. Partitioning of the free space-time for on-road navigation of autonomous ground vehicles. In Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, Australia, 12–15 December 2017. [CrossRef]
20. Li, Q.; Ju, H.; Xiao, P.; Chen, F.; Lin, F. Optimal trajectory optimization of 7r robot for space maintenance operation. *IEEE Access* 2020, *early access*. [CrossRef]
21. Huang, F.; Guo, W.; Zhao, H. AGV Path Planning Based on Improved Genetic Algorithm. In Proceedings of the 2023 2nd International Symposium on Control Engineering and Robotics (ISCER), Hangzhou, China, 17–19 February 2023; pp. 3323–3327. [CrossRef]
22. Cao, J.; Li, Y.; Zhao, S.; Bi, X. Genetic-Algorithm-Based Global Path Planning for AUV. In Proceedings of the 2016 9th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 10–11 December 2016; pp. 79–82. [CrossRef]
23. Han, Z.; Wang, D.; Liu, F.; Zhao, Z. Multi-AGV path planning with double-path constraints by using an improved genetic algorithm. *PLoS ONE* **2017**, *12*, e0181747. [CrossRef]
24. Li, J.; Deng, G.; Luo, C.; Lin, Q.; Yan, Q.; Ming, Z. A Hybrid Path Planning Method in Unmanned Air/Ground Vehicle (UAV/UGV) Cooperative Systems. *IEEE Trans. Veh. Technol.* **2016**, *65*, 9585–9596. [CrossRef]
25. Salamat, B.; Tonello, A. Stochastic Trajectory Generation Using Particle Swarm Optimization for Quadrotor Unmanned Aerial Vehicles (UAVs). *Aerospace* **2017**, *4*, 27. [CrossRef]
26. Tang, J.; Liu, G.; Pan, Q. A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1627–1643. [CrossRef]
27. Lu, J.; Zhang, Z. An Improved Simulated Annealing Particle Swarm Optimization Algorithm for Path Planning of Mobile Robots Using Mutation Particles. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 2374712. [CrossRef]
28. Huang, H.; Jin, C. A Novel Particle Swarm Optimization Algorithm Based on Reinforcement Learning Mechanism for AUV Path Planning. *Complexity* **2021**, *2021*, 8993173. [CrossRef]
29. Wang, W.; Tao, Q.; Cao, Y.; Wang, X.; Zhang, X. Robot Time-Optimal Trajectory Planning Based on Improved Cuckoo Search Algorithm. *IEEE Access* **2020**, *8*, 86923–86933. [CrossRef]
30. Ma, T.; Lyu, J.; Yang, J.; Xi, R.; Li, Y.; An, J.; Li, C. CLSQL: Improved Q-Learning Algorithm Based on Continuous Local Search Policy for Mobile Robot Path Planning. *Sensors* **2022**, *22*, 5910. [CrossRef]
31. Ren, Z.; Rathinam, S.; Likhachev, M.; Choset, H. Multi-Objective Path-Based D* Lite. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3318–3325. [CrossRef]
32. Han, H.T.; Ji, W.F.; Zhang, Y.Q.; Sha, D.P. Comparative Study of Path Planning by Particle Swarm Optimization and Genetic Algorithm. *Appl. Mech. Mater.* **2014**, *687–691*, 1420–1424. [CrossRef]
33. Zeng, Y.; Wu, Z. Time-optimal trajectory planning based on particle swarm optimization. In Proceedings of the 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, New Zealand, 15–17 June 2015; pp. 1794–1796. [CrossRef]
34. Li, Z.; Hu, C.; Ding, C.; Liu, G.; He, B. Stochastic gradient particle swarm optimization based entry trajectory rapid planning for hypersonic glide vehicles. *Aerosp. Sci. Technol.* **2018**, *76*, 176–186. [CrossRef]

35. Bayona, E.; Sierra-García, J.E.; Santos, M. Generation of Optimum Frenet Curves by Genetic Algorithms for AGVs. In *Artificial Intelligence Applications and Innovations*; Springer Nature: Cham, Switzerland, 2023; Volume 676, pp. 454–464. [CrossRef]
36. Alencar, H.; Santos, W.; Silva Neto, G. *Differential Geometry of Plane Curves*; Number Volume 96 in Student Mathematical Library; American Mathematical Society: Providence, RI, USA, 2022. .
37. Martins, G.D.M.; Naruto, I.d.L.; Danner, P.; Frencl, V.B. A Trajectory Simulator Using Frenet–Serret Formulas Applied to Punctual Objects. In Proceedings of the 13th IEEE International Conference on Industry Applications (INDUSCON), Sao Paulo, Brazil, 12–14 November 2018; pp. 750–755. [CrossRef]
38. Bertolazzi, E.; Frego, M. G1 fitting with clothoids. *Math. Methods Appl. Sci.* **2015**, *38*, 881–897. [CrossRef]
39. Bayona, E.; Sierra-García, J.E.; Santos, M. Keeping Safe Distance from Obstacles for Autonomous Vehicles by Genetic Algorithms. In *18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023)*; Lecture Notes in Networks and Systems; Springer Nature: Cham, Switzerland, 2023; Volume 750, pp. 300–310. [CrossRef]