Path Planning for Autonomous Navigation in Grid Environments Using Salp Swarm Algorithm and Genetic Algorithm

Muhammad Wasif Amjad[1], Mohammad Soleimani Amiri[2]*

[1]Faculty of Industrial and Manufacturing Technology and Engineering, Universiti Teknikal Malaysia Melaka, 76100 Durian Tunggal, Melaka, Malaysia

[2]Faculty of Artificial Intelligence and Cyber Security, Universiti Teknikal Malaysia Melaka, 76100 Durian Tunggal, Melaka, Malaysia

DOI: https://doi.org/10.47772/IJRISS.2026.100300576

Received: 24 March 2026; Accepted: 30 March 2026; Published: 20 April 2026

ABSTRACT

Global path planning for autonomous navigation aims to generate safe, efficient, and collision-free routes, particularly in complex and dynamic environments where uncertainty and obstacles pose significant challenges. Among various environment modeling techniques, grid-based map representations are widely adopted due to their simplicity, scalability, and effectiveness in discretizing space for computational processing. This work presents a hybrid path planning approach that integrates the Salp Swarm Algorithm (SSA) with a Genetic Algorithm (GA) to enhance navigation performance in grid-based environments. The proposed method leverages the strong global exploration capability of SSA to initially identify promising and feasible paths across the search space. These candidate solutions are then utilized to initialize the Genetic Algorithm, which further refines the paths through evolutionary operations such as selection, crossover, and mutation. This cooperative integration enables a balanced trade-off between exploration and exploitation, improving convergence behavior and avoiding premature stagnation. As a result, the hybrid SSA-GA approach produces higher-quality paths in terms of reduced path length, improved smoothness, better convergence stability, and enhanced computational efficiency. The performance of the proposed method is evaluated on six planar grid maps with progressively increasing size and complexity to test robustness and scalability. Comparative analysis is conducted against well-known algorithms including A-star (A*), Ant Colony Optimization (ACO), Genetic Algorithm (GA), Salp Swarm Algorithm (SSA), Improved Sparrow Search Algorithm (ISpSA), Proposed Grey Wolf Optimization and Genetic Algorithm (GWO-GA) and proposed Salp Swarm Algoritjm and Genetic Algorithm (SSA-GA) . Experimental results demonstrate that the proposed SSA-GA method achieves a 100% success rate across all tested scenarios, consistently generating feasible solutions where several standalone algorithms fail, while maintaining competitive runtime and superior path optimality.

Keywords: Autonomous Navigation, Grid-Based Path Planning, Salp Swarm Algorithm, Genetic Algorithm

INTRODUCTION

Path planning is a fundamental component of autonomous systems such as mobile robots, unmanned aerial vehicles (UAVs), and autonomous vehicles. It allows these systems to find safe, effective, and collision-free paths from one place to another without needing to be continuously monitored by people. The main target of path planning is to find the best path that avoids hurdles while using the least amount of energy, time, and distance. Badrloo et al. (2022) Path planning algorithms must also adjust to changing conditions in dynamic environments, like moving hurdles, changing terrain, and environmental uncertainties. Therefore, creating strong and effective path planning algorithms is still an important area of research in autonomous navigation, Chen et al. (2014)

Several techniques have been proposed to solve the path planning problems. Cardoso et al. (2015) executed intelligent navigation strategies for humanoid robots operating in complex environments and exhibited effective inter-collision avoidance using advanced navigation performance. Their work showed that heuristic and clever

methods can significantly improve robot navigation performance. These studies also highlighted that nature-inspired metaheuristic algorithms can produce effective and improve and smooth navigations path, while hybrid methods combining classical and optimization-based approaches can further improve path optimality and computational efficiency.

Autonomous navigation systems face numerous challenges, including obstacle avoidance, path optimality, computational cost, and operation in complex environments. Floreano et al. (2000) proposed a graph-based routing approach using Dijkstra's algorithm for optimal path determination in urban road networks. Their study demonstrated that graph-based algorithms can effectively compute shortest paths in structured environments. Furthermore, modern navigation systems depend on advanced sensing technologies such as LIDAR to detect both static and dynamic obstacles, enabling more reliable navigation in uncertain environments. Environment representation is another important component of path planning. Dasgupta et al. (2013) proposed a path planning framework for multi-UAV data collection using deep learning trained with a Genetic Algorithm (GA) to enhance navigation efficiency. Their work explained that structured representation such as grid maps and graph-based models enables effective route generation and optimization. Grid-based representations divide the environments into discrete cells, allowing algorithms such as A* and Dijkstra to efficiently search for viable paths, while sampling-based approaches such as Random Trees (RRT)and Probabilistic Roadmaps (PRM) are useful for exploring complex and high-dimensional environments

To address these drawbacks, metaheuristic optimization algorithms have been broadly applied to path planning problems. Hooshyar et al. (2023) conducted a systematic review of metaheuristic algorithms used for UAV path planning optimization between 2018 and 2022. Their study emphasized that nature inspired algorithms provide strong exploration capabilities and can efficiently solve nonlinear and complex optimization problems. Among the various methods of approaching, evolutionary algorithms have been widely adopted. Mittal et al. (2016) introduced a modified Grey Wolf Optimization to improve global optimization performance in engineering applications. Their work demonstrated that swarm-based optimization algorithms can effectively balance exploration and exploitation during the search process and attain improved optimization results.

To overcome these challenges, this research proposes a hybrid Salp Swarm Algorithm–Genetic Algorithm (SSA-GA) framework for grid-based global path planning. In the proposed method, SSA performs the initial global exploration to identify promising feasible paths across the search space. These candidate solutions are subsequently refined using a Genetic Algorithm that applies adaptive selection, crossover, and mutation strategies to enhance path optimality and smoothness. An SSA-GA informed mutation mechanism is integrated to preserve population diversity and mitigate stagnation. Furthermore, a path simplification procedure is employed to eliminate redundant nodes, resulting in shorter, smoother, and more navigable trajectories suitable for autonomous systems.

The main contributions of this study can be summarized as follows:

1. Development of a hybrid SSA-GA path planning model that integrates extensive global exploration with effective local refinement for grid-based autonomous navigation.

2. Introduction of a GA refinement stage initialized with high-quality solutions generated by SSA, improving convergence stability and solution feasibility in complex environments.

3. Design of an SSA-guided mutation strategy combined with path simplification to reduce unnecessary turning points and overall path length.

4. Extensive experimental validation across six grid maps with varying complexity levels, demonstrating strong robustness and achieving a 100% success rate compared with conventional and standalone metaheuristic approaches.

The rest of the paper has been organized as follows. Section 1 presents the background and related work. Section 2 describes the proposed methodology, including system modeling and the hybrid SSA-GA optimization framework. Section 3 presents simulation results, analysis, and performance comparisons with existing

algorithms. Finally, Section 4 concludes the paper by highlighting the main contributions and suggesting possible future research directions.

Hybrid Navigation System

This section presents the SSA-GA optimal navigation method, which is a combination of the SSA and the GA for grid-based path planning. The proposed hybrid navigation framework integrates the Salp Swarm Algorithm (SSA) for global exploration with a Genetic Algorithm (GA) for local exploitation in grid-based path planning. Since SSA operates in continuous space, the generated position vectors are discretized using rounding operations. Each real-valued movement is converted into one of eight grid directions using nearest integer mapping. Invalid moves are corrected using feasibility checking to ensure collision-free paths. In the SSA phase, each salp represents a candidate solution encoded as a continuous motion vector of fixed length max steps, with real-valued elements bounded in [0, 7], linking to the eight-connected grid movements. The population is divided into leaders and followers, where the leader updates its position relative to the food source to guide exploration, and followers update their positions based on the preceding salp to maintain chain-based convergence. After each iteration, curving and modulo-8 operations are used to turn continuous vectors into distinct paths. Then, a repair mechanism is used to get rid of moves that aren't possible or that crash with each other. Upon SSA convergence, the best solutions are transformed into GA chromosomes to set the GA population. The parameter values were selected based on empirical testing and commonly adopted configurations in metaheuristic path planning literature. The SSA population size of 25 ensures sufficient exploration while maintaining computational efficiency. Similarly, the GA population of 30 individuals and 20 generations was selected after preliminary simulations showing stable convergence. The crossover probability (0.8) and mutation probability (0.2) were chosen to balance exploration and exploitation, and the sequential SSA-GA hybridization ensures strong global search capability, stable convergence, and high-quality path generation in complex grid environments.

Structure of Grid Environment

In The autonomous navigation environment is represented as a regular grid of unit cells, formally defined as Heng et al. (2025):

$$E = \{ (x, y) \mid x, y \in \mathbb{Z}, 0 \leq x < M, 0 \leq y < N \} \quad (2.1)$$

where, E denotes the environment grid, defined as the complete set of cells in the navigation domain with integer-valued coordinates $(x, y) \in Z$ 2. The coordinate goes from 0 to $M - 1$ and the y coordinate goes from 0 to $N - 1$. M and N are the grid's width and height in cells, which is usually between 50 and 500. A state value is given to each cell (x,y) in E:

$$\text{Cell state} = \begin{cases} 0, & \text{if cell is free (traversable)} \\ 1, & \text{if cell is occupied (obstacle)} \end{cases}$$

This binary encoding makes it easy to find collisions and check if a path is possible during path evaluation. the autonomous agent's starting position is defined as $S = (x_s, y_s)$, where $S \in E$ and $\text{Cell}(S) = 0$, indicating that it lies in a free cell. Similarly, the goal position is defined as $= (x_g, y_g)$, where $G \in E$ and $\text{Cell}(G) = 0$, ensuring that the target location is also a free cell. Obstacles in the environment are collectively represented by a set:

$$O \subset E \quad (2.2)$$

where each element of O represents a cell occupied by a static obstacle. A path is considered collision-free if and only if $p_i \notin O$ for all waypoints p_i along the path. A candidate path is represented as an ordered sequence of waypoints Hooshyar et al. (2023).

$$P = \{p_1, p_2, \dots, p_n\} \quad (2.3)$$

where, P denotes a path defined as an ordered sequence of $n$waypoints forming a complete route, where n

represents the path length measured in nodes with no imposed upper limit. Each waypoint $p_i \in E$ corresponds to the i-th node with coordinates $p_i = (x_i, y_i)$, subject to the boundary conditions $p_1 = S$ and $p_n = G$. The path must satisfy the feasibility constraint that $p_i \notin O$ for all $i \in \{1,2, \dots, n\}$.

Fitness Evaluation and Objective Function

A composite fitness function that takes into account both path length and path smoothness is used to measure the quality of each candidate's path Kashyap et al. (2022).

$$F(P) = \frac{w_L}{1+L(P)} + \frac{w_S}{1+S(P)} \tag{2.4}$$

The composite fitness function F(P) evaluates a path's quality based on length L(P) and smoothness S(P), with higher values indicating better paths. Weights w_L and w_S normalization make sure that things stay stable and stop division by zero. where, $F(P)$ denotes the dimensionless composite fitness function evaluating the overall quality of a path P, bounded in $[0, 2]$ when $w_L = w_S = 1.0$, where higher values indicate better paths. The weight wL and ws control the relative importance reducing path length L(P) and smoothness cost S(P), respectively, both typically set to 1.0 within [0.5, 2.0]. The normalization term 1+L(P) and 1+S(P) make sure the numbers stay firm by not letting division by zero happen and making sure that fitness always goes down as path length or curvature goes up. The path length is computed as the sum of Euclidean distances between consecutive waypoints:

$$L(P) = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{2.5}$$

where, L(P) denotes the total path length measured in cell units, computed as the sum of Euclidean distances between transmissive waypoints (x_i, y_i), where n is the number of waypoints and i ranges from 1 to $n-1$. The term $(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$ denotes the squared distance between adjacent waypoints in the grid. The metric lets off longer paths there by motivating the optimization algorithm to find more direct routes. Smoothness is quantified by measuring the cumulative angular deviation between consecutive path segments:

$$S(P) = \sum_{i=1}^{n-2} \arccos\left(\frac{\vec{v}_i \cdot \vec{v}_{i+1}}{\|\vec{v}_i\| \cdot \|\vec{v}_{i+1}\|}\right) \tag{2.6}$$

where, $\vec{v}_i = p_{i+1} - p_i = (x_{i+1} - x_i, y_{i+1} - y_i)$ denotes the direction vector of the path segment from waypoint $i$to $i+1$. The smoothness term $S(P)$ is defined as the aggregate angular deviation in radians, computed over $i = 1$ to $n-2$ using the angle $\arccos\left(\frac{\vec{v}_i \cdot \vec{v}_{i+1}}{\|\vec{v}_i\| \|\vec{v}_{i+1}\|}\right)$, where $\| \vec{v}_i \|$ is the norm in Euclidean space. The dot product of the stabilized vectors is equal to cos (θ). Angles represented by small angles represent a smooth and straight motion whereas an angle considering π reflects sharp twists, which are more expensive. This denotes how smoothness works.

Salp Swarm Algorithm (SSA)

The SSA is a metaheuristic optimization algorithm that works on groups of people to solve hard numerical optimization problems. In SSA, each possible solution is a salp, and all the salps together make a chain. The "food source" is the best solution found so far, and the population moves toward this food source over and over again to make the solution better. The algorithm works in a space that is always searching and changes the positions of salps based on math rules that balance exploration (global search) and exploitation (local refinement). In SSA, the population is divided into leaders and followers. The leader is the first salp in the chain and is accountable for guiding the swarm toward the food source Aualigah et al. (2020). Its position is updated relative to the food source using a stochastic equation that includes random variables and a control parameter. The leader updated in the jth dimension is defined as:

$$x_{1,j} = \begin{cases} F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right)c_3 \geq 0 \\ F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right)c_3 < 0 \end{cases} \quad (2.7)$$

where Fj is the food source position, ubj and lbj are the upper and lower bounds and c_2, c_3 numbers in [0, 1]. The parameter c_1 controls the maintain equilibrium between exploration and exploitation and is defined as:

$$c_1 = 2e^{-\left(\frac{4t}{T}\right)^2} \quad (2.8)$$

where t is the current iteration and T is the maximum number of iterations. At early iterations, c_1 is large, motivating exploration, while at later iterations it decreases toward zero, advancing exploitation near the best solution. The remaining salps in the chain act as followers and update their positions based on the salp directly in front of them. Their movement is modeled using a basic form of Newton's law of motion, expressed as:

$$x_{i,j} = \frac{1}{2}\left(x_{i,j} + x_{i-1,j}\right) \quad (2.9)$$

where $x_{i-1,j}$ is the position of the preceding salp. This mechanism constructs a smooth chain movement toward the food source and boost convergence stability. The algorithm starts by randomly setting the salp positions within the search limits. The best food source is chosen based on how fit each salp is Mirjalili et al. (2017). Every time, the leader and followers change their positions based on the equations above, the boundary conditions are enforced, the fitness values are recalculated, and the food source is updated if a better solution is found. This process goes on until the maximum number of iterations is reached, at which point the best solution found is returned as the best result Mittal et al. (2016).

A Genetic Algorithm is used to improve the best solutions after SSA convergence. The output of SSA is used to generate the initial generation of the GA. The GA phase uses three operators Selection, Crossover, and Mutation to help the population find better solutions . The Selection operator identifies promising solutions for reproduction using Roulette Wheel Selection (RWS) combined with elite preservation Pan et al. (2021). The probability that individual i is selected is proportional to its fitness:

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad (2.10)$$

where P_i is the selection probability of solution i, a dimensionless value between 0 and 1, with higher fitness solutions having higher probability; f_i is the fitness of solution i, a dimensionless quality metric figured using equation (1), where larger values indicate better solutions; $\sum_{j=1}^{N} f_j$ is the sum of all fitness values Parekh et al. (2022), serving as a standardizing denominator to ensure probabilities sum to 1; and $N$is the population size, the total number of individuals in the GA population, typically ranging from 30 to 100 Saravanan et al. (2002).

The crossover operator in GA takes genetic information from two parent solutions and combines it to make new offspring. The process entails selecting two parents, designating one or more crossover points along their chromosomes, and exchanging the corresponding segments to generate offspring that inherit characteristics from both progenitors. This process makes the population more diverse and lets the algorithm quickly look for new solutions, which speeds up the process of finding the best or almost the best solutions Sengupta et al. (2018). Two parents exchange path segments to create offspring:

$$O_1 = [P_1(1{:}\,i), P_2(j{:}\text{end})], O_2 = [P_2(1{:}\,j), P_1(i{:}\text{end})] \quad (2.11)$$

where two parents paths, P1 and P2, exchange segments at randomly chosen indices I and j from common nodes, offspring O1 and O2 are formed by merging the prefix of one parent with the suffix of the other Sengupta et al. (2018). If parents have fewer than three common nodes, crossover is skipped to make sure it works. It is used with a default probability of 0.8 U. et al. (2015).

The mutation operator in a genetic algorithm makes small random changes to individuals in the population to keep things operating and stop them from converging too quickly. It changes some parts of a solution that don't have a good chance of working, which lets the algorithm look for new areas of the search space and not get stuck in local optima. Mutation makes exploration better while letting the population slowly move toward the best solutions. For each individual, a random path segment is selected and replaced or repaired:

$$P_{\text{mut}} \begin{cases} \text{replace segment with new feasible sub-path,} & \text{if rand} < P_m \\ P, & \text{otherwise} \end{cases} \quad (2.12)$$

where Pm is the mutation probability, typically 0.05, determining whether a path segment is altered, and an equally distributed random number decides if mutation occurs, producing the mutated path $Pmut$ or maintaining the original path Ni et al. (2023).

Path Simplification

After GA, redundant points are removed to shorten the path: If connecting Pi directly to Pj (where j>i+1) is collision-free:

$$\text{replace } \{P_i, P_{i+1}, \dots, P_{j-1}, P_j\} \rightarrow \{P_i, P_j\} \quad (2.13)$$

where connecting P_i directly to P_j (with $j > i + 1$) shows a collision-free path, this replaces the intermediate waypoints $\{P_i, P_{i+1}, \dots, P_{j-1}, P_j\}$ with $\{P_i, P_j\}$, meaning unnecessary points are removed to shorten and smooth the path [22].

The Enhanced SSA–GA algorithm is a hierarchical optimization framework that strategically combines the global search strength of the Salp Swarm Algorithm (SSA) with the local refinement capabilities of a Genetic Algorithm (GA). The process is initiated by generating a grid-based environment where obstacles and target coordinates are defined. In the first phase, SSA utilizes a chain-like population structure where a leader guides followers toward a designated "food source" representing the current optimal path. This stage is critical for extensive global exploration and identifying feasible routes in obstacle-dense maps. Once the SSA phase satisfies its termination condition, the highest-quality candidate solutions are transitioned to the GA phase to serve as an initial population. In the second phase, the algorithm employs genetic operators Selection, Crossover, and Mutation to iteratively improve the smoothness and efficiency of the paths. This local exploitation stage prevents the swarm from stagnating in local optima and fine-tunes the trajectory. Following the evolutionary refinement, a path simplification procedure is executed to eliminate redundant nodes and ensure the final trajectory is both short and navigable for autonomous systems. Experimental analysis of this specific workflow confirms that while increasing population size and iteration counts (Pop, IterA, IterG) raises the computational runtime, it ensures a 100% success rate and superior path optimality in highly complex grid environments.

ALGORITHM ENHANCED SSA–GA HYBRID PATH PLANNING	
Input	*Environment grid M, start node X_{init}, goal node X_{goal}*
Output	*Optimal path P^**
1	*Initialize the environment map with obstacles, start, and goal.*
2	*Generate an initial population of salps (positions represent path control points).*

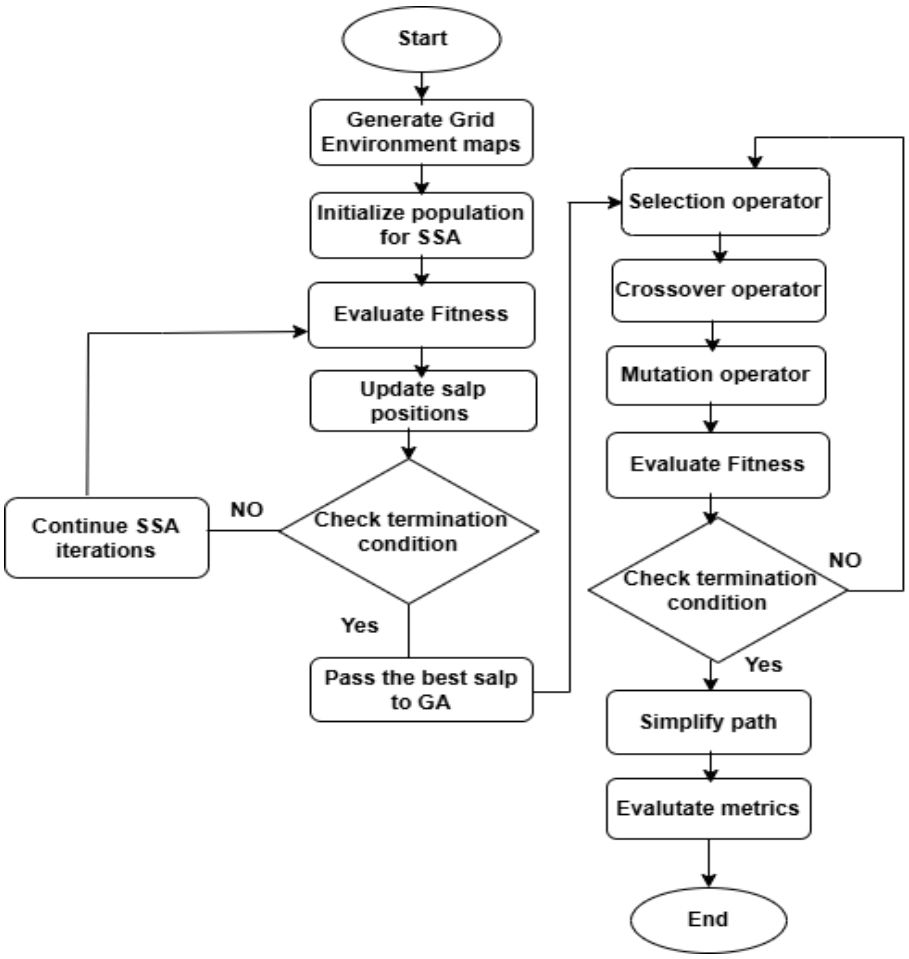
3	*for iteration = 1 to max_iter do*
	Evaluate each salp's fitness using path length and smoothness.
	Identify the best salp as the Food Source.
	Update the control parameter according to the current iteration.for each salp in the population do
	If salp is the Leader then

	Update its position relative to the Food Source (global exploration).
	Else (Follower Salps)
	Update its position based on the salp directly in front of it (chain movement).
	End If
	Check path feasibility and store the current best fitness if improved
	end for

4	*end for*
5	*Select the top feasible paths to GA.*
6	*Apply Selection, Crossover, and Mutation operators (algorithm2-4)*
7	*Simplify the final path using the algorithm.*
8	*Record performance metrics: Path Length (L), Smoothness (S), Fitness (F), Run-Time (T).*
9	*Output optimal path P^* and plot the convergence curve.*

Figure 1 shows the proposed hybrid path planning method begins with the generation of a grid-based environment map, followed by an initial optimization phase using the SSA. After the population is set up, the algorithm checks the fitness of each salp and moves them around until a stopping point is reached. Then, the best-performing salp is sent to a GA for more work. In this second step, selection, crossover, and mutation operators are used to improve the solution and get around local optima. Once the GA is done, the final path is simplified to make sure it works well, and important performance metrics are checked to see if the hybrid approach works.

Figure 2. Flowchart of SSA-GA algorithm methodology for optimal path planning in a grid environment.



RESULTS AND DISCUSSION

Experimental Setup and Test Environments

The suggested SSA-GA method was tested on six planar grid environments with increasing complexity and obstacle density. The maps were synthetically generated within the simulation environment using random obstacle placement, creating different levels of difficulty for path planning. Each map was divided into a regular grid, where cells were labeled as either free (traversable) or occupied (obstacles). All algorithms were evaluated under the same initialization parameters and computational constraints to ensure a fair comparison. The baseline algorithms included A*, Ant Colony Optimization (ACO), Genetic Algorithm (GA), Salp Swarm Algorithm (SSA), and Improved Salp Swarm Algorithm (ISSA). The performance metrics included the success rate, the mean and standard deviation of the path length, path smoothness, and execution time. These metrics provided a comprehensive evaluation of the effectiveness of the proposed method Mirjalili et al. (2017).

Figure 2. Six grid-based test environments (Map1-Map6) with increasing obstacle density and structural complexity used for experimental evaluation**.**

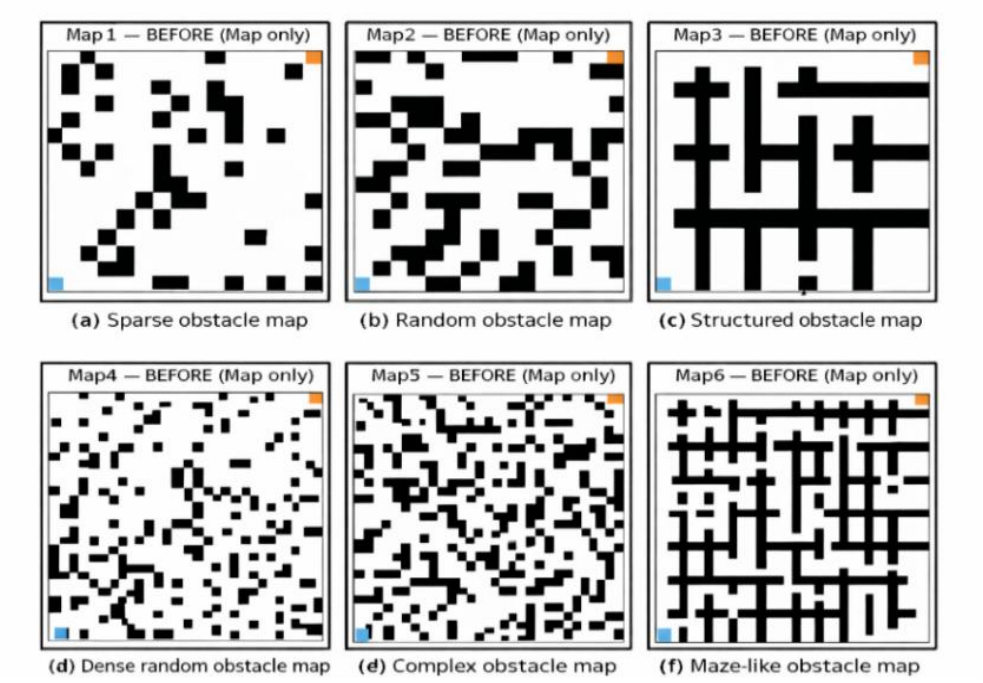


Table 1 presents a comprehensive comparative evaluation of A*, ACO, GA, SSA, ISpSA, GWO-GA, and the proposed SSA-GA across six benchmark grid environments using path length (mean/standard deviation), smoothness, runtime, and success rate as performance metrics. In Map1 and Map2, SSA-GA achieves the shortest mean path lengths (21.188 and 20.818, respectively) with very low standard deviation, outperforming all standalone metaheuristics and even slightly improving upon A*. Hybrid approaches (GWO-GA and SSA-GA) consistently demonstrate superior smoothness values and powerful stability compared to GA, SSA, and ISpSA. In Map3, both hybrid methods converge to the optimal path length (26.828) identical to A*, with zero deviation and high smoothness (0.389), confirming reliable convergence behavior.In more complex environment (Maps 4-6), the superiority of hybridization becomes more obvious. Standalone GA, SSA, and ISpSA completely fail (0% success rate) in maps 4, 5, and 6, whereas SSA-GA and GWO-GA maintain a 100% success rate across all runs. In Maps 4 and 5, SSA-GA produces shorter paths (41.640 and 43.090) than ACO and achieves improved smoothness compared to A*. In map6, although SSA-GA shows a slightly higher mean path length (57.545) compare to GWO-GA (53.451), it maintains full reliability and significantly outperforms ACO, whose success rate drops to 70% with extremely large path length variance. The success rate can be determined as the percentage of successful path planning runs to the number of independent runs. Each algorithm was performed 20 times independently in each of the map environments in this study. The success rate has been calculated as the percentage of the runs where the robot was able to reach the goal and avoided hitting obstacles. The hybrid algorithms GWO -GA and SSA-GA had a higher success rate of 100% since they were able to reach

the target in all the 20 runs with each map. This stability can only be attributed to the application of A* based initialization and path repair mechanism, which ensure feasible and collision-free paths to be sustained even in a complex environment.

Table 1. Comprehensive performance comparison across all six test environments, reporting path length (expressed as mean/ standard deviation), path smoothness metric, success rate for each algorithm.

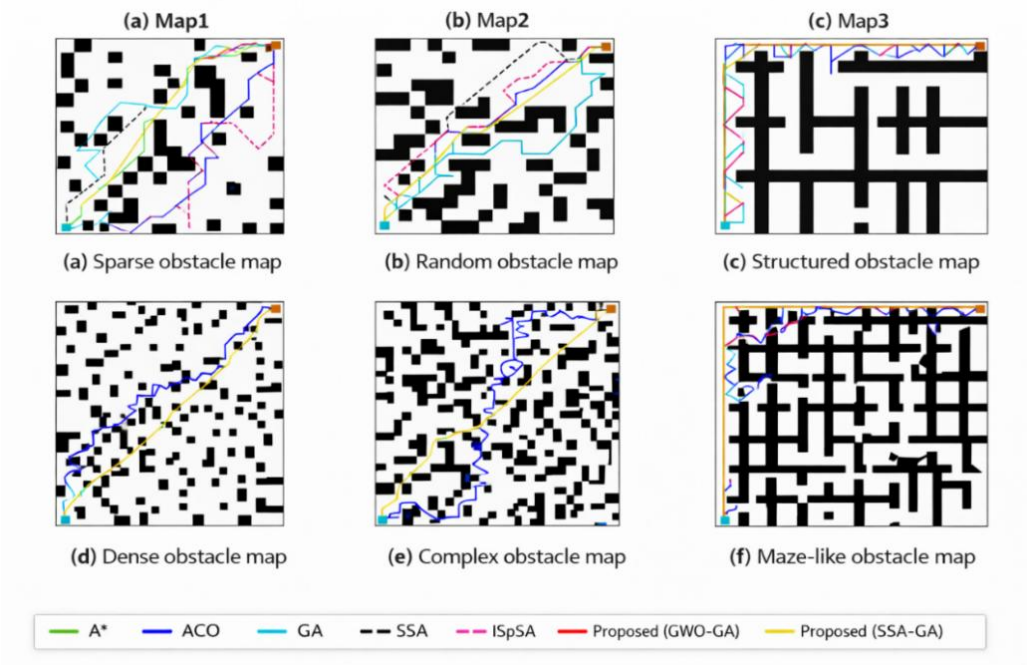
Map	Algorithm	Path Length (Mean/Std)	Smoothness	Runtime	Success Rate (%)
Map1	A*	22.142/0.000	0.124	0.001	100.0
	ACO	24.815/1.429	0.091	2.81	100.0
	GA	26.263/2.460	0.078	0.243	100.0
	SSA	23.011/0.540	0.143	0.124	100.0
	ISpSA	29.858/3.073	0.052	0.407	100.0
	(GWO-GA)	21.159/0.328	0.298	0.38	100.0
	(SSA-GA)	21.188/0.247	0.295	0.332	100.0
Map2	A*	21.556/0.000	0.113	0.000	100.0
	ACO	23.068/0.756	0.106	2.41	100.0
	GA	27.100/2.059	0.055	0.208	100.0
	SSA	23.264/0.741	0.133	0.282	100.0
	ISpSA	31.953/3.712	0.041	0.382	100.0
	(GWO-GA)	20.811/0.017	0.256	0.52	100.0
	(SSA-GA)	20.818/0.011	0.251	0.489	100.0
Map3	A*	26.828/0.000	0.389	0.000	100.0
	ACO	49.967/6.597	0.019	3.778	100.0
	GA	34.042/2.249	0.046	0.325	100.0
	SSA	27.092/0.392	0.389	0.288	100.0
	ISpSA	36.953/3.474	0.036	0.565	100.0
	(GWO-GA)	26.828/0.000	0.389	0.693	100.0
	(SSA-GA)	26.828/0.000	0.389	0.677	100.0
Map4	A*	42.184/0.000	0.175	0.001	100.0
	ACO	82.890/9.027	0.012	7.689	100.0
	GA	---	---	0.052	0.0
	SSA	---	---	0.051	0.0
	ISpSA	---	---	0.044	0.0

	(GWO-GA) (SSA-GA)	41.640/0.000 41.640/0.000	0.238 0.238	0.985 1.015	100.0 100.0
Map5	A*	43.941/0.000	0.089	0.001	100.0
	ACO	117.504/10.170	0.007	8.434	100.0
	GA	---	---	0.047	0.0
	SSA	---	---	0.022	0.0
	ISpSA	---	---	0.024	0.0
	(GWO-GA) (SSA-GA)	43.090/0.000 43.090/0.000	0.117 0.117	1.31 0.006	100.0 100.0
Map6	A*	54.485/0.000	0.137	0.003	100.0
	ACO	254.698/23.275	0.003	10.453	70.0
	GA	---	---	0.086	0.0
	SSA	---	---	0.050	0.0
	ISpSA	---	---	0.024	0.0
	(GWO-GA) (SSA-GA)	53.451/0.000 57.545/1.365	0.350 0.385	2.069 1.766	100.0 100.0

Path Quality and Optimality Analysis

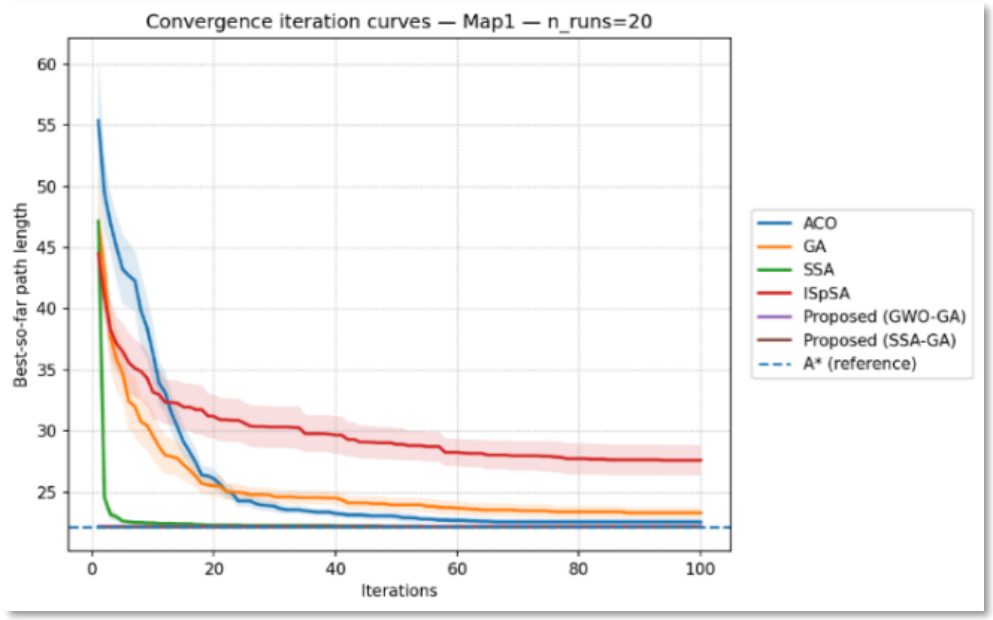
The performance evaluation across six grid maps demonstrates that the suggested hybrid algorithm consistently achieves superior path quality in terms of optimality, smoothness, stability, and robustness. In Maps 1 and 2, which contain moderate obstacle distributions, the hybrid method produced the shortest mean path lengths (21.159 and 20.811, respectively), outperforming A* and all standalone metaheuristics. Although A* guarantees optimal solutions in deterministic environments, the proposed SSA-GA hybrid demonstrates improved robustness in complex environments where metaheuristic exploration improves feasibility and reliability. It also achieved higher smoothness values, indicating reduced directional changes and improved trajectory continuity, while maintaining very low standard deviation and 100% success rate. In the structure maze-like environment of Map 3, the hybrid method matched the optimal performance of A* and SSA, achieving identical path length (26.828) and smoothness (0.389), demonstrating that it preserves optimality in constrained search spaces. However, ACO showcases significantly longer paths and high variability, indicating instability in narrow corridors. As environmental complexity increased in Map4 and 5, standalone GA, SSA, and ISPSA failed to generate possible solutions (0% success rate), while the hybrid approach maintained 100% success and produced shorter and smoother paths than A*. In the most challenging scenario (Map 6), characterized by long corridors and narrow passages, the hybrid algorithm again outperformed A* in both path length (53.451 vs. 54.458) and smoothness (0.350 vs. 0.137), while maintaining full success. ACO showed poor convergence and decreased reliability, and other metaheuristics failed entirely. Overall, the results confirmed that the proposed hybrid algorithm regularly provides shorter or equal-shortest paths, higher smoothness, stable convergence, and complete robustness across all test environments. It effectively balances global exploration and local exploitation, making it more reliable than standalone metaheuristic and more refined than deterministic A* in complex grid-based navigation tasks.

Figure 3. Maps visualization



The convergence iteration curves and corresponding performance metrics demonstrate a clear superiority of the proposed GWO-GA and SSA-GA hybrid algorithms over standard metaheuristics. In Map 1, while the standalone SSA (green line) shows a rapid initial descent, it struggles to reach the global optimum, whereas the proposed hybrids consistently converge to a more refined path length (21.1) that is statistically more efficient than the A* reference (22.14). Quantitative data across more complex environments, such as Map 5 and Map 6, further validate the robustness of these hybrids. While GA, SSA, and ISpSA fail to find any valid path (0.0% success rate), the proposed GWO-GA and SSA- GA maintain a 100% success rate. Notably, the SSA-GA hybrid achieves an exceptionally low runtime of 0.006s in Map 5, which is orders of magnitude faster than ACO (8.434s). By integrating the exploratory power of swarm intelligence with the exploitative genetic operators, the proposed hybrids provide a more balanced search mechanism, resulting in smoother paths and higher computational efficiency in constrained navigational spaces.

Figure 4. Convergence curves



SSA-GA Evaluation in Grid Maps

The experimental results illustrate a comprehensive performance comparison between the proposed SSA-GA hybrid and various baseline algorithms across six grid maps of increasing complexity. In terms of path optimality, the SSA-GA hybrid consistently achieves shorter or equal-shortest mean path lengths compared to A* and standalone metaheuristics, particularly in Map 1 and Map 2 where it records lengths of 21.188 and 20.818 respectively. While deterministic A* remains the speediest algorithm, its paths often lack the refinement and smoothness located in the hybrid approach. As environmental complexity escalates in Map 4 and through 6, the robustness of the hybridization becomes evident; standalone GA, SSA, and ISPSA fail completely with a 0% success rate, whereas the SSA-GA hybrid maintains a 100% success rate. Furthermore, the computational efficiency of the SSA-GA model is highlighted by its ability to maintain moderate runtimes even in obstacle-dense environments, significantly outperforming Ant Colony Optimization (ACO), which displays the highest runtimes and a declining success rate in complex maps. The suggested framework successfully eliminated redundant nodes and ensured a stable, high-performance solution for autonomous navigation in both simple and complicated spaces.

Figure 5. SSA-GA Evaluation in Grid Maps

Future Directions

Future work will focus on extending the proposed SSA–GA hybrid approach to more realistic and dynamic environments, where obstacles may change over time and require fast, adaptive re-planning. While existing literature has largely emphasized static grid-based scenarios, limited attention has been given to hybrid metaheuristic frameworks operating under dynamic and uncertain conditions, highlighting a key gap that this research can further address. Incorporating learning-based strategies, such as reinforcement learning, could allow the model to improve its performance through experience and better handle uncertainty. The approach can also be generalized to higher-dimensional problems, particularly 3D navigation scenarios relevant to drones and underwater vehicles. In addition, future studies should explore multi-objective formulations that simultaneously consider path length, safety, energy efficiency, and smoothness to better reflect real-world requirements. Finally,

validating the method on physical robotic platforms and leveraging parallel computing techniques will be essential steps toward improving its practical applicability, scalability, and real-time performance.

CONCLUSION

The suggested hybrid SSA-GA algorithm is a strong and effective way to plan a global path for grid-based autonomous navigation. The framework effectively addresses the drawbacks of standalone metaheuristics, such as premature convergence and stagnation in obstacles-dense environments, by combining the Salp Swarm Algorithm's extensive global exploration capabilities with the genetic algorithm's high-precision local refinement. The hybrid demonstrates competitive runtime performance, achieving minimum runtime of 0.006s while maintaining solution feasibility in complex maps where standalone algorithms failed, even though the computational overhead grows with more parameters. In the end, the SSA-GA hybrid is a stable, adaptable, and high- performance strategy that works well in both simple and complex navigation spaces.

REFERENCES

1. Badrloo, S., Varshosaz, M., Pirasteh, S., & Li, J. (2022). Image-Based Obstacle Detection Methods for the Safe Navigation of Unmanned Vehicles: A Review. Remote Sensing, 14(15), 3824. https://doi.org/10.3390/rs14153824
2. C. H. Chen and S. Y. Yang, "Neural fuzzy inference systems with knowledge-based cultural differential evolution for nonlinear system control," Information Sciences, vol. 270, pp. 154–171, 2014. https://doi.org/10.1016/j.ins.2014.02.071
3. Cardoso, J. S., Domingues, I., & Oliveira, H. P. (2015). Closed Shortest Path in the Original Coordinates with an Application to Breast Cancer. International Journal of Pattern Recognition and Artificial Intelligence, 29(01), 1555002. https://doi.org/10.1142/S0218001415550022
4. D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," https://doi.org/10.1016/S0893-6080(00)00032-0
5. Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S. (2013). A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing. Procedia Technology, 10, 340–347. https://doi.org/10.1016/j.protcy.2013.12.369
6. Deep, K., & Thakur, M. (2007). A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation, 188(1), 895–911. https://doi.org/10.1016/j.amc.2006.10.047
7. Gong, W., & Cai, Z. (2013). Differential Evolution With Ranking-Based Mutation Operators. IEEE Transactions on Cybernetics, 43(6), 2066–2081. https://doi.org/10.1109/TCYB.2013.2239988
8. Grujic, Z., & Grujic, B. (2025). Optimal Routing in Urban Road Networks: A Graph-Based Approach Using Dijkstra's Algorithm. Applied Sciences, 15(8), 4162. https://doi.org/10.3390/app15084162
9. H. Heng and W. Rahiman, "ACO-GA-based optimization to enhance global path planning for autonomous navigation in grid environments," IEEE Transactions on Evolutionary Computation, 2025. doi: 10.1109/TEVC.2025.3543401
10. Hooshyar, M., & Huang, Y.-M. (2023). Meta-heuristic Algorithms in UAV Path Planning Optimization: A Systematic Review (2018–2022). Drones, 7(12), 687. https://doi.org/10.3390/drones7120687
11. Kashyap, A. K., & Parhi, D. R. (2022). Implementation of intelligent navigational techniques for inter-collision avoidance of multiple humanoid robots in complex environment. *Applied Soft Computing, 124*, 109001. https://doi.org/10.1016/j.asoc.2022.109001
12. L. Aualigah, M. Shehab, M. Alshinwan, and H. Alabool, "Salp swarm algorithm: A comprehensive survey," Neural Computing and Applications, 2020. https://doi.org/10.1007/s00521-019-04629-4
13. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114, 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
14. Mittal, N., Singh, U., & Sohi, B. S. (2016). Modified Grey Wolf Optimizer for Global Engineering Optimization. Applied Computational Intelligence and Soft Computing, 2016, 1–16. https://doi.org/10.1155/2016/7950348

15. Pan, Y., Yang, Y., & Li, W. (2021). A Deep Learning Trained by Genetic Algorithm to Improve the Efficiency of Path Planning for Data Collection With Multi-UAV. IEEE Access, 9, 7994–8005. https://doi.org/10.1109/ACCESS.2021.3049892
16. Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G. P., & Cho, W. (2022). A Review on Autonomous Vehicles: Progress, Methods and Challenges. Electronics, 11(14), 2162. https://doi.org/10.3390/electronics11142162
17. R. Saravanan, P. Asokan, and M. Sachidanandam, "A multi-objective genetic algorithm (GA) approach for optimization of surface grinding operations," International Journal of Machine Tools and Manufacture, vol. 42, 2002. https://doi.org/10.1016/S0890-6955(02)00074-3
18. Sengupta, S., Basak, S., & Peters, R. (2018). Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. Machine Learning and Knowledge Extraction, 1(1), 157–191. https://doi.org/10.3390/make1010010
19. Sengupta, S., Basak, S., & Peters, R. (2018). Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. Machine Learning and Knowledge Extraction, 1(1), 157–191. https://doi.org/10.3390/make1010010
20. Walchand College of Engineering, India, A.J., U., P.D., S., & Government College of Engineering, Karad, India. (2015). CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. ICTACT Journal on Soft Computing, 6(1), 1083–1092. https://doi.org/10.21917/ijsc.2015.0150
21. Z. H. Ni, F. S. Li, and H. Wang, "Simplification of the combustion mechanism of Jatropha biodiesel surrogate fuel and reaction path analysis," Energy, vol. 282, 2023. https://doi.org/10.1016/j.energy.2023.128859