International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.7, February 2024

Optimizing Robot Path Planning in 2D Static Environments using GA, PSO and ACO Search Algorithms

Santosh Shrestha
Computer Science Department
Missouri State University
Springfield, MO 65807

Pranith Varma Appani
Computer Science Department
Missouri State University
Springfield, MO 65807

Praveen Reddy Kota
Computer Science Department
Missouri State University
Springfield, MO 65807

Alaa Sheta
Computer Science Department
Southern Connecticut State University
New Haven, CT, USA 06515

ABSTRACT

Meta-heuristic search algorithms have shown great success in robot motion planning by designing collision-free paths in static and dynamic environments. Meta-heuristic search algorithms with suitable representation (i.e., encoding) can find the optimal path from a start to endpoint effectiveness via waypoints. The waypoints are random points generated in the search environment. In this research, we investigate the use of several meta-heuristic algorithms, such as Genetic Algorithms (GA), Particle Swarm Optimizations (PSO), and Ant Colony Optimizations (ACO), to solve path planning problems in a two-dimensional static environment. The proposed path planning-based search enjoys a free robot passage in all possible space directions to operate in complex search spaces. Performance benchmarking is also carried out through simulation in various scenarios to determine and analyze the performance of each algorithm.

Keywords

Path Planning, Metaheuristic Search Algorithms, Genetic Algorithms, Particle Swarm Optimizations and Ant Colony Optimizations.

1. INTRODUCTION

In the last century, robotics become an essential element in many fields, such as manufacturing automation, where an autonomous robot system can be utilized for parts and products manufacture [1]. Robotics were also used in the car industry. In 2021, the Republic of Korea used 2,867 robots for every 10,000 employees, occupying the top position in the world. Other countries such as Germany, the United States, and Japan followed with 1500, 1457, and 1422 for every 10,000 employees, respectively [2]. Currently, one million robots work in the car industry worldwide [3]. Autonomous navigating of robot systems has become progressively critical in many applications [1,4]. Planning robot motion is one of the significant tasks in intelligent control of autonomous systems [5,6]. Autonomous vehicles dramatically reduce the contribution of human driver error and negligence as the cause of vehicle collisions [7].

Path planning involves finding an optimal collision-free path from the desired source to the destination in any given environment [8]. Solving path planning problems requires three primary considerations: 1) Finding a path or route that connects from source to destination, 2) the obtained path or route needs to be obstacle-free, and 3) the obtained path or route needs to be the shortest one [9]. It is one of the complex problems in the field of robotics and computer science, and the topic has been studied over decades, enabling wide ranges of technologies, from autonomous driving to Unmanned Aerial Vehicles (UAVs) [9, 10]. It is considered a complex problem since, in computational complexity theory, it is considered an NP (Nondeterministic Polynomial time) complete problem [11]. This means that as the environment complexity grows - so does the computational complexity of the solution drastically. Finding the most effective solution is challenging, and many attempts have been made to develop the best method to search for the space of the optimal path. The application of such effective solutions is vital in today's world.

Even though there exist many traditional approaches to solving path planning problems, they all suffer from optimization issues in one way or another [12]. As the complexity of the environment grows, it has been known that these methods do not perform well or suffer some efficiency problems [13]. These metaheuristic-based solutions perform consistently in real-world applications while requiring less memory and processing better results than the traditional approaches [14, 15]. Meta-heuristics algorithms also do not suffer from local minimum problems in path planning like other algorithms do [15]. One of the primary reasons why these algorithms

perform comparatively better is because they are not greedy and can balance between local and global optimal solutions.
This research explores the practicality of meta-heuristic-based methods to handle robot motion planning and navigation problems in static environments with several obstacles. This paper aims to explore and understand how effective meta-heuristic algorithms are in solving path-planning problems by searching for the optimal path through simulation. The nature and behavior of these algorithms are studied, and their results are shown through various experiments. Furthermore, a performance comparison is made to determine which algorithm performs the best in our given path-planning problem scenario so that we can highlight their key strengths and weaknesses. The experiments' results will clarify how these algorithms work and how they can effectively apply to path-planning problems. Benchmarking the algorithm's performance will also provide a general estimation of how well they perform compared to each other, and the information shall be helpful when deciding the type of algorithm to use to solve path-planning problems.
This paper is organized as follows. Section 2 provides a literature review of the research in robot automation and path planning using traditional and meta-heuristic search methods. Section 3, we provide a statement for the problem formulation-based path planning. A detailed discussion of various meta-heuristics search methods is provided in Section 4. Section 6 provides the setup for the planned experimentation to find the optimal path. The experimental results are provided in Section 7. Finally, we provide the work conclusions.

2. BACKGROUND AND RELATED WORK

Mobile robots show intelligent behaviors where they can execute complex tasks without dedicated supervision (i.e., autonomously) in known or unknown environments and do not rely on human assistance. Nowadays, autonomous robots are widely used in numerous applications such as autonomous driving [11], industrial robots [16], mine detection [17, 18], battlefields [19], and protein folding [20]. Among the rapidly growing field of research in robotics, robot path planning is most common in the literature [7]. The path planning problem solution offered includes three categories:

—Traditional algorithms (for example, Visibility Graph, Artificial Potential Field),

—Heuristic Algorithms (for example, A* Algorithm [21], Dijkstra's Algorithm), and

—Meta-heuristic Algorithms, also known as Evolutionary Algorithms (Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithm) [14, 22].

2.1 Traditional algorithms

Traditional methods, such as Artificial Potential Field, utilize the concept of virtual force to direct the motion of a robot through the path, creating a repulsion effect from obstacles and a gravitational attraction effect toward the goal originally proposed by Khatib [9, 14]. This approach performs better in real-time, although it suffers from significant drawbacks such as getting stuck at local minimum points and inability to access the goal due to physical environment constraints [14]. Another traditional approach is the Visibility graph, which, as the name suggests, is about building a graph by connecting the robot, destination point, and polygonal obstacles [9]. The drawback of the Visibility graph is the need for more flexibility because as the source and designation point changes, the visibility graph has to be reconstructed [9].

2.2 Heuristic Algorithms

Heuristic Algorithms are another type of path planning used widely to solve path planning problems, specifically the ones with discrete topology [23]. One popular among these algorithms is the A* Algorithm, which evaluates values of subsequent nodes or child nodes and grows by selecting the node with the lowest costs of each extension until the goal is reached [21]. The advantage of this algorithm is that it has relatively less complexity with fewer extensions and also has an excellent real-time performance [21]. However, it ignores the actual size of the robot, causing unexpected issues with the motion process in the physical environment [21]. Dijkstra's Algorithm is another popular heuristic search algorithm that traverses nodes from start to end and then compares the forward path to get the shortest possible path [24]. The main advantage of this algorithm is its robustness and ability to find the shortest path in most cases [24]. However, it still struggles with efficiency issues and is not well suited for complex environments [9].

2.3 Meta-heuristic search algorithms

Meta-heuristic search algorithms are another type of search algorithm that has succeeded in solving path planning problems [10, 23, 25]. The term "meta-heuristic" reveals that these algorithms are higher-level schemes that lead other heuristics to discover the solution space competently. These can be applied to many optimization problems since they are not problem-dependent. They offer flexible and effective instruments for locating appropriate solutions satisfactorily, although they do not necessarily guarantee the optimal solution. Natural processes around us inspire these types of algorithms. Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) are a few of the popular ones that have been used widely for several problem-solving [8, 10, 13, 25]. Although the various evolutionary algorithms require varying tuning parameters and functions to be implemented, they all follow similar evolutionary characteristics.
This paper primarily focuses on finding a global path in a static environment and follows an offline path-planning process. As a part of the algorithm, the population is initialized, the individual's cost fitness function is calculated iteratively, and the best solutions are obtained as the algorithm continues to evolve.

3. PROBLEM STATEMENT

Path planning involves choosing the best path between a source and a destination in an environment while minimizing time and avoiding obstacles. Path planning problem varies depending upon the type of plan, time, and environment [7].

—Based on plan type, path planning can be either global or local [13]. Global path planning requires prior knowledge of the environment before initiating. However, local path planning prior knowledge of the environment is unknown, and the information is collected as the robot progresses through its sensors [12, 13].

—Path planning can be online or offline based on time [12]. In Online path planning, the path from start to finish is planned and adjusted as the robot moves through sensors, but in offline path planning, the path is pre-planned, and the same path is followed.

—The Environment of the path planning problem can also be two different types: static and dynamic [7]. In a dynamic environment, obstacles are moving or dynamically vary and are not stationary, but in a static environment, obstacles are known and are stationary [13]. The environment where the path planning should be

executed varies, and that is directly correlated to the complexity of the problem [12].

In [26], the authors presented a detailed study of various methods/approaches utilized for mobile robot navigation in static and dynamic environments.

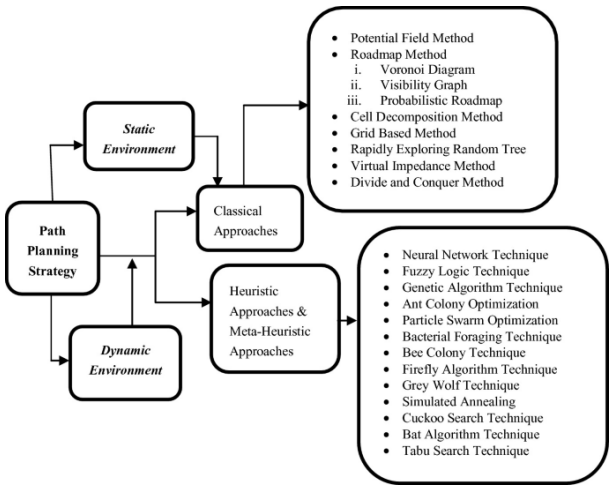

Fig. 1: Path Planning Approaches for Mobile Robot Navigation in Various Environments [26]

As illustrated in Figure 2, mobile robots move in a closed workspace, given an enclosed Euclidean 2D workspace. To assist the robot in avoiding obstacles, a map is constructed with a set of path points and a finite number of static obstacles.
Pathway points are intermediate points between the start and the goal that help direct the robot along the path. The path points can be generated randomly or pre-defined as needed. However, the criteria for defining these points should be that the path point must be within the environment boundary and should not overlap any obstacles.

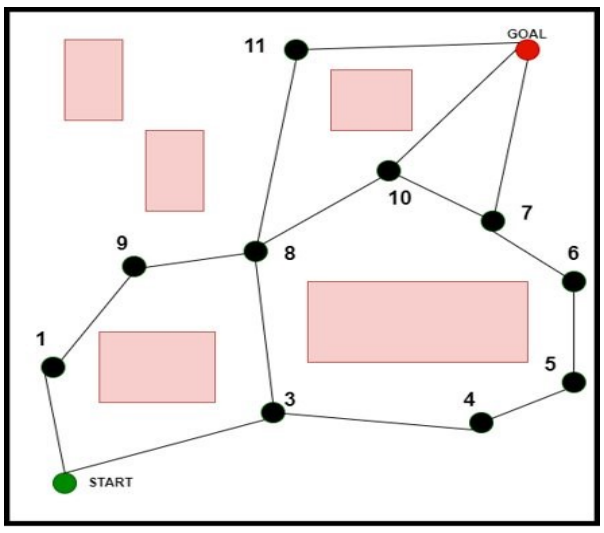

Fig. 2: Path points connecting to source to destination

4. METAHEURISTICS SEARCH ALGORITHMS

Metaheuristic search algorithms are higher-level algorithms that guide or direct significant search processes to reach near-optimal solution space [27–29]. The primary objective of these algorithms is to efficiently explore the available search space until the near-optimal solutions are reached [28]. The main characteristics of metaheuristic algorithms are that they are problem-independent and are usually non-deterministic [15] unlike traditional algorithms, which are problem-specific with limited search space. This very nature of the algorithms allows them to be applied in various problems, including NP-hard problems [15] such as path planning problems.
In the following subsections, we will discuss the evolutionary process for popular metaheuristics algorithms, including Genetic Algorithms(GA), Particle Swarm Optimizations (PSO), and Ant Colony Optimizations (ACO).

4.1 Genetic Algorithm (GA)

Genetic algorithms are popular meta-heuristic algorithms inspired by the genetic evolution process and natural selection of Darwin's evolution theory initially proposed by J.H. Holland in 1975 [30]. GA initiates with a set of candidate solutions, and through the iterative repeated processes like selection, crossover, and mutation, the best individual survives and is considered the best solution [31]. An initial population with a sequence of paths that may have potential solutions is created. The population contains candidate solutions of chromosomes, which are binary strings. Through the iterative process of selection evaluated by the fitness function, mutation, and crossover, a near-optimal solution is reached [31]. The main advantage of GAs is their ability to explore search space and integrate with other algorithms. But, the downside is the intensive computational operations it requires [31]. The GA algorithm is presented in Algorithm 1.

Algorithm 1 Genetic Algorithm [32]

Result: output the best individual found with containing path points
Initialize the starting and target point
  Generate randomly initial population using path points in each chromosome
  while *NOT(convergence condition)* **do**
    Evaluate the fitness for each chromosome in the current population
    Select the parents using Roulette selection
    Eliminate the lowest fitness chromosome
    Duplicate the highest fitness chromosome
    Apply randomly crossover process between current parents using the given probability while keeping the start and end points without change in the population
    Apply the mutation process with the given probability
    Generate the new population
end
return *the best individual found*

4.1.1 Encoding. Chromosomes are a set of candidate solutions for the problem [16, 33]. For the path planning problem, the chromosome representation is a sequence of nodes (points) with the starting point (node) of each segment at the beginning of the first segment, the intermediate nodes (via points), and the endpoint (destination) at the end of the last segment. The 36 binary bits are chosen

for the chromosome representation as the length accommodates sufficiently the number of path points for our experiment. A random population of chromosomes is generated initially, with a fixed start and goal point but with random intermediate points between each. Chromosomes look something like this:

1	0	1	1	...	1	1	1	1	0	1	0	1	0

For example, a chromosome with the enclosed values 0000010001100110011001111001111111111 represents $0-4-6-6-6-7-9-15-15$ nodes sequences given that we have a set of 16 waypoints denoting each node uniquely with source node 0 and end node 15.

4.1.2 Evaluations. The randomly generated population of chromosomes goes through an evaluation process using the fitness function described in Section 5 and Equation 7 [32]. This process evaluates the total distance covered by the nodes present in each chromosome.

4.1.3 Selection. Based on the fitness values of each chromosome in the population, a specific set of chromosomes with better fitness values needs to be selected to produce offspring and proceed to the next generation. This selection process is based on roulette wheel selection. In this process, the selection of chromosomes is given by the probability directly proportional to their fitness values. The probability p_x is given by Equation 1.

$$p_x = \frac{f_x}{\sum_n^{i=1} f_i} \tag{1}$$

where f_x is the fitness value of chromosome x [16].

4.1.4 Operators. Crossover produces two new paths in the next generation by combining two parent paths [32, 33]. In this case, we have used two crossover points. Each parent path is randomly divided into three parts, and the parts are recombined. Creating new children involves exchanging the middle parts of the first and second paths between crossover bit positions.

4.1.5 Mutation. A chromosome is altered by replacing the value of 1 and 0 and vice-versa in specific chromosome locations [16]. It is an essential part of the process as it will help maintain diversity and randomness in the population and prevent the solution from getting stuck in the local minimum. A flow chart that shows how GAs work is presented in Figure 3.

4.2 Particle Swarm Optimization (PSO)

The Particle Swarm Optimization algorithm is a popular metaheuristic algorithm that emulates the social behavior of birds in nature and exploits the idea of sharing information among individuals in a group to direct the entire population to follow in the right direction [34]. PSO was first developed by James Kennedy and Russell Eberhart as a social simulation inspired by the behavior of swarms in nature, such as a bird flock or a fish school [12]. The primary motivation for the algorithm was to simulate human social behavior. However, the algorithm's simplicity and fast convergence have found their way through implementation in several applications, and it has shown great success in solving path planning problems [12]. It uses information sharing among the group to decide the movement of the entire group to the best possible state or direction. The PSO algorithm is presented in Algorithm 2.
The PSO process begins by initializing a set of random solutions. Individual particles update their position using their personal and

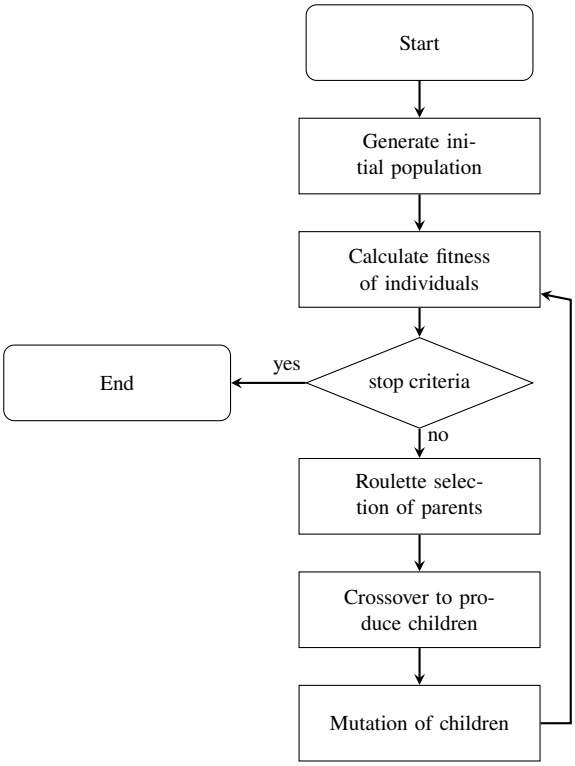

Fig. 3: GA Flowchart [32]

global best solutions to follow optimal particle position. Compared to other methods, it is simpler to implement and contains fewer parameters [12]. The objective function of a PSO system represents the candidate solutions of particles with two properties: position and velocity [34]. Iteratively establishing the fitness value for each particle based on position is the objective function. Positions with higher fitness values are considered better. During the search, each particle is updated at a dynamically adjusted velocity based on its own personal best fitness value and the global fitness values of the entire population [35].
Following are the Equations 2 and 3 that update each particle's position and velocity [35].

$$V_i = \omega V_{i-1} + c_1 r_1 (P_{best} - X_{i-1}) + c_2 r_2 (G_{best} - X_{i-1}) \tag{2}$$
$$X_i = X_{i-1} + V_i \tag{3}$$

In the Equation, X_i is the particle's position, and v_i is the particle's velocity. c_1 and c_2 are the heuristic tuning parameters that control the social and personal weights or influences, and r_1 and r_2 are random numbers between 0 and 1 assigned to each particle. ω is inertia weight coefficient and usually set to 1. P_{best} represents the personal best fitness value of a particle and G_{best} represents the global best of the entire population [34].
For PSO to handle the path planning problem, certain modifications in encoding need to be made where each particle has a priority array, which is used to find the shortest path from source to destination. During the initialization phase, these priorities are randomly chosen, and with iterations, these priorities are adjusted based on their fitness values. Following the node based on their priorities from start to finish, fitness values are calculated using Equation 1 for a single particle. p_i contains the priority array of each node in the solution

or particle. Visiting each node or path point based on the priority from start to end points is required for each solution. Figure 4 presents a flow chart showing how PSO works.

Algorithm 2 Particle Swarm Optimization [35]

Result: global best with path points based on highest priority
Initialize *Swarm Size N, Maximum Iterations it_max*
for *each particle i(i = 1 to N)* **do**
  Initialize position with random priority for each node x_i
  Initialize velocity v_i
  Initialize P_{best}
  Initialize G_{best}
end
while *(it < it_max)* **do**
  for *each particle i(i = 1 to N)* **do**
    Calculate new velocity using V_i eqn.2
    Obtain new new position using P_i eqn.3
    Compute Fitness based on the new position using eqn.7
    if P_{best} >*Fitness* **then**
      Update P_{best}
      if G_{best} >*Fitness* **then**
        Update G_{best}
      end
    end
  end
end
return *theshortestpathpoints*

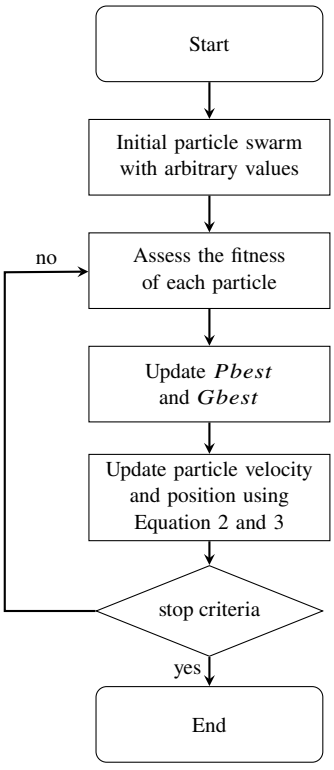

Fig. 4: PSO Flowchart [34]

4.3 Ant Colony Optimization

Another type of meta-heuristic algorithm inspired by real-world biological ants is ACO. It was proposed by M. Dorigo in 1992 in his Ph.D. dissertation [11]. The inspiration for Ant colony optimization comes from the natural behaviors of Ants, which deposit a particular type of hormone called a pheromone to find the path to their food sources. The ACO algorithm is presented in Algorithm 3.
Ant Colony Optimization (ACO) is a metaheuristic algorithm that emulates ants' behavior to find optimal solutions [36]. Each ant leaves a hormone called pheromone along the path when searching for food. The shortest path will have a higher pheromone concentration, and other ants follow the same path [11, 36]. To prevent getting stuck in local minima, pheromone evaporates over time, allowing ants to choose another path [11]. The pheromone levels on the shortest path remain high because the pheromone deposit speed is faster than its evaporation speed.
The major drawback is that as the environment complexity increases, so does the computational cost exponentially [9]. Another common meta-heuristic approach is Simulated annealing, which emulates the cooling down process of solid materials. The initial temperature and cooling state are set, allowing the temperature to decrease progressively [14]. The random spike characteristics allow exploration in search space and avoid falling into a local minimum. The disadvantage of the approach is it is slower in converging, and issues created by its randomness [37].

Algorithm 3 Ant Colony Optimization [36]

Result: shortest path from source to destination
Initialize the number of ants (N), number of iterations (it), and pheromones level
while *NOT(reached maximum iterations)* **do**
  for *each ant* **do**
    Select next node based on the probability p_k^{ij} until the end node from the start node
    if *end node reached* **then**
      Calculate fitness
      Deposit pheromones along the path taken
    end
  end
end
return *best path points with the highest pheromone level*

Consider a network where ants can travel between different nodes. Using pheromone deposits, the probability that an ant k located in node i will choose to go to another node in the network is given in [11] (See Equation 4).

$$p_k^{ij} = \begin{cases} \dfrac{(\tau_{ij}^k)^\alpha (\eta_{ij}^k)^\beta}{\sum_{l \in N_i^k} (\tau_{il}^k)^\alpha (\eta_{il}^k)^\beta} & \text{if } j \in N_i^k \\ 0 & \text{if } j \notin N_i^k \end{cases} \tag{4}$$

The evaporation rate (also called an evaporation percentage), which diminishes the value of deposited pheromone along the path (from node i to j) as time progresses, is given by Equation 5.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k}^{m} \Delta\tau_{ij}^k \tag{5}$$

The new pheromone levels are updated once pheromone evaporation occurs and the ants) that just crossed the path have deposited additional pheromones [11] (See Equation 6).

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant expression } k \text{ uses } ij \text{ in its tour} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The decision as to which edge in the graph is chosen depends upon the pheromone level deposited previously. The definition of the variables used in this system of equations is presented in Table 1. The probability of p_k^{ij} of moving from node i to j relies on product of two values: η_{ij}^k which is computed by heuristic value and τ_{ij}^k of the path. Figure 5 presents a flow chart showing how ACO works.

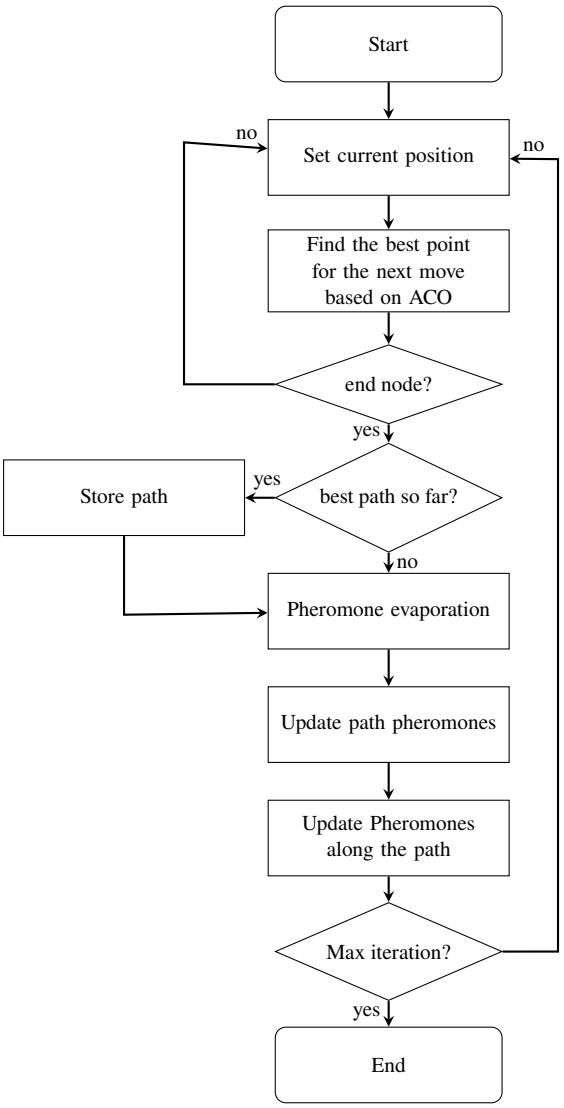

Fig. 5: ACO Flowchart [11]

5. FITNESS FUNCTION

A fitness function is a critical component of metaheuristics algorithms. The function is used to evaluate and direct existing search space to better solution space [31]. The function varies by problem, but for path planning problems, we can use Euclidean distance function [16, 23, 33].
Given a solution containing a path of a set of nodes in a graph, as shown in Figure 2, we can calculate fitness value by summing the distances from the start node to the end node. The fitness function is given by Equation 7.

$$Fitness(F) = \sum_{n=1}^{m+1} d(P_i, P_{i+1}) \quad (7)$$

Distance between two nodes is calculated by Equation 8

$$d(P_i, P_{i+1}) = \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i,)^2} \quad (8)$$

P_i and P_{i+1} can be any two nodes in the graph and m is the number of nodes in the candidate solution.

6. EXPERIMENTAL SETUP

A fixed bounded size of 100 × 100 2-D environment with varying path points and obstacles was set up for simulation. The environment's start and end points were determined and remained constant before running the simulation. The environment contained an enclosed 2-dimensional boundary with numerous path points and obstacles. The path points for the environment are randomly generated within the enclosed boundary and are obstacle-free. Obstacles are enclosed rectangular boundaries within the environment. The path points overlapping with the obstacles are ignored and removed from the path point lists. The algorithms find the shortest path from the source to the destination, combining some path points. The source and destination remain the same throughout the experiments while the level of complexity increases.
Figure 6 shows the environment for scenario 1 with obstacle-free path points and obstacles along with the source marked as a '+' point and the goal marked as an 'x' point.

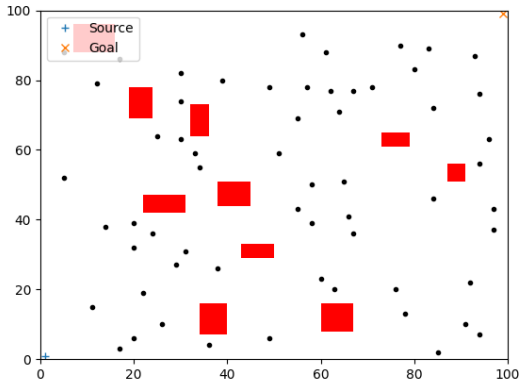

Fig. 6: Scenario 1: 2-dimensional Environment

Table 2 describes the specifications for each of the three scenarios. Scenario 1 contains an environment with ten obstacles and 60 path points. Similarly, scenarios 2 and 3 contain 15 and 20 obstacles and 120 and 180 path points, respectively. The increase in environmental complexity is obtained by increasing the number of path points and obstacles. This was set up to investigate and analyze the behavior

Parameter	Symbol	Description
Pheromone Amount	τ_{ij}	Amount of pheromone deposited when moving from node i to j
Tuning Parameter	α	Controls the significance of τ_{ij}
Tuning Parameter	β	Controls the influence of η
Desirability Factor	η	Represents the desirability factor from node i to j; prior knowledge of fitness function, $1/d_{ij}$, where d_{ij} is the distance from node i to j computed from Equation 1
Evaporation Rate	ρ	Rate of pheromone evaporation
Pheromone Amount	$\Delta\tau_{ij}^{k}$	Amount of pheromone deposited by the k^{th} ant in the path planning problem

Table 1. : Description of Parameters in Ant Colony Optimization

and performance of each meta-heuristic as the environment changes. This would also help us understand how the algorithm copes with increased complexity and what impact the change in path points and obstacles will have on their overall performance.

Scenario	Obstacles count	Path points Count
1	10	60
2	15	120
3	20	180

Table 2. : Environment set-ups

7. EXPERIMENTAL RESULTS

The simulations with three different scenarios, as stated in Table 2, each running GA, PSO, and ACO algorithms, were carried out in a machine equipped with Windows 10 Home, powered by an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, with 16GB of installed memory (RAM), and operates on a 64-bit system with an x64-based processor.

The experiments were performed at least five times to find consistent performance, and the results were observed. As the performance of these algorithms depends on the tuning parameters, optimal parameter values were determined through repeated experiments and filtering out the best possible values for each. It is essential to properly benchmark all these algorithms to the level playing field as it provides a fair ground for evaluation.

In Tables 3, 4, and 5, the tuning parameters for each of the algorithms used in the experiments are shown. These tuning parameters were obtained through trial and error and found to be the best-performing ones during the experiment.

Parameters	Scenario 1	Scenario 2	Scenario 3
Number of Iterations	80	80	80
Number of individuals	70	70	100
Crossover split size	0.5	0.5	0.5
Mutation Rate	0.3	0.3	0.3

Table 3. : GA Tuning Parameters

As seen in Figure 7, 8, and 9, all algorithms were able to discover common shortest paths after running them in simulation for scenarios 1, 2, and 3 respectively. Although some took longer than others, eventually, all algorithms could converge to common shortest paths in the tested environment.

Parameters	Scenario 1	Scenario 2	Scenario 3
Number of Iterations	80	80	80
Swarm size	30	40	40
ω	1	1	1
c_1	0.1	0.1	0.1
c_2	0.2	0.2	0.2

Table 4. : PSO Tuning Parameters

Parameters	Scenario 1	Scenario 2	Scenario 3
Number of Iterations	80	80	80
Number of ants	20	20	30
Initial pheromone value	0.8	0.8	0.8
Evaporation rate	0.1	0.1	0.1

Table 5. : ACO Tuning Parameters

Figure 10, 11, and 12 show the convergence graph of GA, PSO, and ACO when best paths in scenarios 1, 2, and 3 are discovered. It can be seen that PSO converges fastest, and GA seems to converge the slowest. ACO is converging between the two. This trend seems to follow in the remaining scenarios, 2 and 3, with a higher level of environment complexity - PSO always converges the fastest among the three algorithms.

For each scenario, the convergence time for PSO is faster than the rest of the algorithms. GA has the slowest convergence time. An increase in the complexity of the environment, as in scenarios 2 and 3, with additional path points and obstacles, impacts the overall performance of all the algorithms, but the overall pattern or ratio stays the same. Table 6 compares the exact millisecond numbers' performance.

Algorithms	Average Convergence Time (in ms)		
	Scenario 1	Scenario 2	Scenario 3
GA	1011.23	2154.85	3982.37
PSO	**122.45**	**322.78**	**954.91**
ACO	706.90	1111.43	2154.51

Table 6. : Comparison of the average convergence time for each algorithm

Table 7 shows each algorithm's average shortest path lengths. An increase in complexity with additional path points and obstacles does not seem to help discover better optimal shorter paths or improve the solution in any way given the same enclosed boundary and source/destination. Conversely, the solution or shortest path got worse with increased path points.

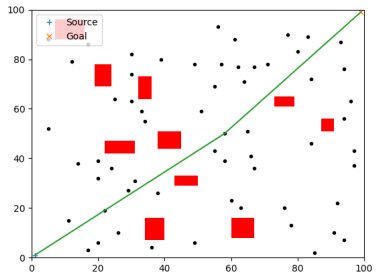

Fig. 7: Scenario 1: Best found shortest Path

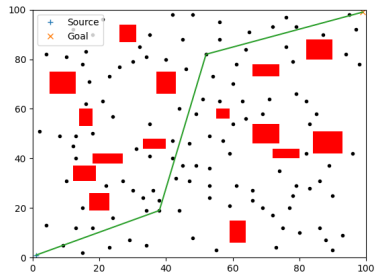

Fig. 8: Scenario 2: Best found shortest Path

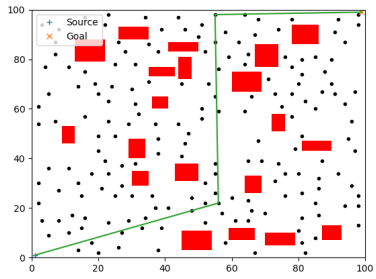

Fig. 9: Scenario 3: Best found shortest Path

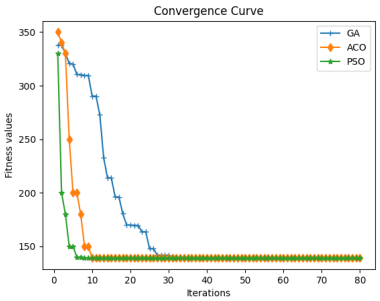

Fig. 10: Scenario 1: Convergence curve

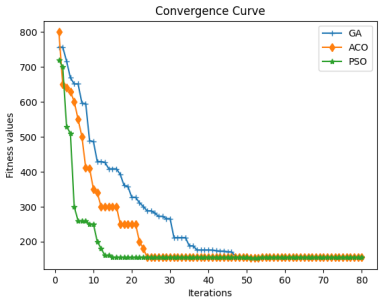

Fig. 11: Scenario 2: Convergence curve

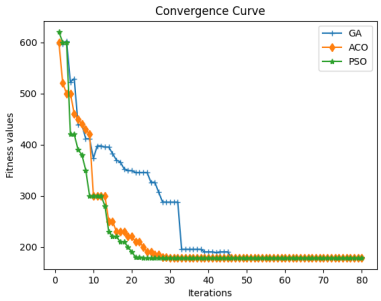

Fig. 12: Scenario 3: Convergence curve

Algorithms	Average Path Length		
	Scenario 1	Scenario 2	Scenario 3
GA	149.27	155.17	178.89
PSO	138.65	155.56	180.25
ACO	137.75	156.56	178.14

Table 7. : Comparison of average path-lengths

8. CONCLUSIONS AND FUTURE WORK

This research introduced our idea of utilizing several met-heuristics search algorithms to find the optimal path in a static environment. GA, PSO, and ACO search algorithms were explored in three environments with various complexities concerning the number of obstacles and waypoints. Our experiments show that these algorithms perform relatively similarly, but the convergence time may vary depending on the heuristic tuning parameters. We found that PSO provided superior results compared to GA and ACO. PSO found the shorted path solution in minimum time compared to other algorithms. Further investigation can be implemented by experimenting with these algorithms in a dynamic environment where obstacles are not fixed.

9. REFERENCES

[1] M. N. A. Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?" *Annual Reviews in Control*, vol. 50, pp. 233–252, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578820300675

[2] IFR, "International federation of robotics," http://www.ifr.org/, 2020.

[3] M. Bartoš, V. Bulej, M. Bohušík, J. Stanček, V. Ivanov, and P. Macek, "An overview of robot applications in automotive industry," *Transportation Research Procedia*, vol. 55, pp. 837–844, 2021, 14th International scientific conference on sustainable, modern and safe transport. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146521004543

[4] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101343, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098623000204

[5] T.-K. Dao, T.-S. Pan, and J.-S. Pan, "A multi-objective optimal mobile robot path planning based on whale optimization algorithm," in *2016 IEEE 13th international conference on signal processing (ICSP)*. IEEE, 2016, pp. 337–342.

[6] Z. Yan, J. Zhang, Z. Yang, and J. Tang, "Two-dimensional optimal path planning for autonomous underwater vehicle using a whale optimization algorithm," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 9, p. e6140, 2021.

[7] M. Abed, O. Farouq, and Q. Al-Doori, "A review on path planning algorithms for mobile robots," *Engineering and Technology Journal*, vol. 39, pp. 804–820, 05 2021.

[8] J. Hopkins, F. Joy, A. Sheta, H. Turabieh, and D. Kar, "Path planning for indoor uav using A* and late acceptance hill climbing algorithms utilizing probabilistic roadmap," *International Journal of Engineering & Technology*, vol. 9, no. 4, null. [Online]. Available: https://par.nsf.gov/biblio/10222262

[9] Z. Fu, J. Yu, G. Xie, Y. Chen, and Y. Mao, "A heuristic evolutionary algorithm of UAV path planning," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–11, 09 2018.

[10] A. Sheta, M. Braik, D. R. Maddi, A. Mahdy, S. Aljahdali, and H. Turabieh, "Optimization of PID controller to stabilize quadcopter movements using meta-heuristic search algorithms," *Applied Sciences*, vol. 11, no. 14, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/14/6492

[11] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," vol. 3, pp. V3–436–V3–440, 2010.

[12] G. Bilbeisi, N. Al-Madi, and F. Awad, "Pso-ag: A multi-robot path planning and obstacle avoidance algorithm," in *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2015, pp. 1–6.

[13] I. Altaharwa, A. Sheta, and M. Alweshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, vol. 4, 01 2008.

[14] "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?" *Annual Reviews in Control*, vol. 50, pp. 233–252, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578820300675

[15] Y. Gigras, N. Jora, and A. Dhull, "Comparison between different meta-heuristic algorithms for path planning in robotics," *International Journal of Computer Applications*, vol. 142, pp. 6–10, 2016.

[16] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018, proceedings Of The First International Conference On Intelligent Computing In Data Sciences, ICDS 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091830125X

[17] M. Zubair and M. A. Choudhry, "Land mine detecting robot capable of path planning," in *2010 Second World Congress on Software Engineering*, vol. 1, 2010, pp. 34–37.

[18] R. Sawant, C. Singh, A. Shaikh, A. Aggarwal, P. Shahane, and H. R, "Mine detection using a swarm of robots," in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, 2022, pp. 1–9.

[19] B. A. Swett, E. N. Hahn, and A. J. Llorens, *Designing Robots for the Battlefield: State of the Art*. Cham: Springer International Publishing, 2021, pp. 131–146.

[20] G. Song, S. Thomas, K. Dill, J. M. Scholtz, and N. Amato, "A path planning-based study of protein folding with a case study of hairpin formation in protein g and l," *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 240–51, 02 2003.

[21] A. K. Guruji, H. Agarwal, and D. Parsediya, "Time-efficient a* algorithm for robot path planning," *Procedia Technology*, vol. 23, pp. 144–149, 2016, 3rd International Conference on Innovations in Automation and Mechatronics Engineering 2016, ICIAME 2016 05-06 February, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017316300111

[22] D. R. Maddi, A. Sheta, A. Mahdy, and H. Turabieh, "Multiple waypoint mobile robot path planning using neighborhood search genetic algorithms," ser. AIRC '19. New York, NY, USA: Association for Computing Machinery, 2020, p. 14–22. [Online]. Available: https://doi.org/10.1145/3388218.3388225

[23] M. M. Costa and M. F. Silva, "A survey on path planning algorithms for mobile robots," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2019, pp. 1–7.

[24] F. Syed Abdullah, S. Iyal, M. Makhtar, and A. A. Jamal, "Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries," 12 2015.

[25] A. Sheta, A. Ali, A. Baareh, and S. Aljahdali, "Meta-heuristic search algorithms for solving the economic load dispatch problem," in *2022 3rd International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, 2022, pp. 87–92.

[26] A. Sanyal, M. Nayab Zafar, J. C. Mohanta, and M. Faiyaz Ahmed, "Path planning approaches for mobile robot navigation in various environments: A review," in *Advances in Interdisciplinary Engineering*, N. Kumar, S. Tibor, R. Sindhwani, J. Lee, and P. Srivastava, Eds. Singapore: Springer Singapore, 2021, pp. 555–572.

[27] K.-L. Du and M. N. S. Swamy, *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*, 1st ed. Birkhäuser Basel, 2016.

[28] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Chapter 10 - metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, ser. Intelligent Data-Centric Systems, A. K. Sangaiah, M. Sheng, and Z. Zhang, Eds. Academic Press, 2018, pp. 185–231. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128133149000104

[29] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges," *Artificial Intelligence Review*, vol. 56, pp. 13 187–13 257, 2023. [Online]. Available: https://doi.org/10.1007/s10462-023-10470-y

[30] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975, second edition, 1992.

[31] J. Tu and S. Yang, "Genetic algorithm based path planning for a mobile robot," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1, 2003, pp. 1221–1226 vol.1.

[32] G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic algorithms," 10 2004, pp. 185– 189.

[33] W. Parvez and S. Dhar, "Path planning optimization using genetic algorithm," *International Journal of Computational Engineering Research*, vol. 3, no. 4, pp. 23–28, 2013.

[34] Y. K. Ever, "Using simplified swarm optimization on path planning for intelligent mobile robot," *Procedia Computer Science*, vol. 120, pp. 83–90, 2017, 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917324237

[35] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 4, 2004, pp. 2473–2478 vol.4.

[36] S. Chakraborty, "Ant colony system: A new concept to robot path planning," *International Journal of Hybrid Information Technology*, vol. 6, pp. 11–30, 11 2013.

[37] H. Miao and Y.-C. Tian, "Dynamic robot path planning using an enhanced simulated annealing approach," *Applied Mathematics and Computation*, vol. 222, pp. 420–437, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300313007728