smartPATH: A Hybrid ACO-GA Algorithm for Robot Path Planning

Imen Chaari*, Anis Koubâa *, Hachemi Bennaceur ‡, Sahar Trigui*, Khaled Al-Shalfan ‡

* CES Research Unit, National School of Engineers of Sfax, University of Sfax, Tunisia.
‡ COINS Research Group, Al-Imam Mohamed bin Saud University, Saudi Arabia.
‡ CISTER Research Unit, Politechnic Institute of Porto (ISEP/IPP), Portugal.

I. INTRODUCTION

Robotics is gaining a lot of interest in our daily life and in several areas including industrial applications, manufacturing and constructions, search and rescue, environment exploration and others. Mobile robots’ applications encompass several challenging problems, among them path planning has been considered as a core problem that has attracted a lot of research works, so far. This paper investigates the path planning problem in the context of the RTRACK research project [1], whose purpose is to design a multi-robot indoor surveillance system that takes advantage of wireless sensor networks based services to support the operation of a team of collaborative robots. Indeed, it is essential for the application to provide a path planning solution to support the robot navigation to a certain destination. This represents the main motivation of the present work.

Basically, path planning aims at the construction of a collision-free path for the robot starting from its initial (or current) location to the target location while avoiding the obstacles scattered in a workspace, based on some knowledge about the environment. The constructed path must satisfy a set of optimization criteria including traveled distance, processing time and energy consumption. The traveled distance represents a typical metric of interest since it has a direct impact on time and energy. Designing an efficient path planning algorithm is an essential objective since the path quality influences the efficiency of the whole application.

The path planning problem typically considers both static and dynamic environments: a static environment is unchanging, i.e. the start and goal positions are fixed, and obstacles do not change locations over time. However, in a dynamic environment, the mobile robot is exposed to unexpected situations as the locations of obstacles and the target may change over time. According to the knowledge that the robot has about the environment, path planning can be divided into two classes: In the first class, the robot has a priori knowledge about the environment modeled as a map. As such, the path can be planned offline based on the available map. This category of path planning is known as global path planning. The second class of path planning assumes that the robot does not have a priori information of its environment. Thus, it has to sense the locations of the obstacles and construct an estimated map of the environment in real-time during the search process in order to avoid obstacles and to get a suitable path toward the goal state. This type of path planning is known as local path planning. In this paper, we address the problem of global path planning in a static environment.

Contributions of the paper: This paper has two major contributions:

- We propose smartPATH, a new hybrid ACO-GA algorithm for robot global path planning. The key added value of smartPATH consists in using several powerful mechanisms in the ACO phase in order to prune the searching space, accelerate the search time and improve the solution quality, including (1) definition of a heuristic distance information probability that helps in selecting relevant initial paths, which speeds-up the search process, (2) the proposal of a modified transition rule probability
that considers a lower-bound estimation of the remaining distance to the destination, which discards future expensive paths during the search process, (3) incorporation of ant control mechanism and a mutation operator in the ACO phase to early discard non-relevant solutions and improve the solution quality, respectively. Further, a cross-over GA operator complements the ACO phase to avoid local optimum.

- We present an extensive simulation study to evaluate the performance of smart-PATH and compare it against classical ACO, classical GA approaches and Bellman-Ford shortest path algorithm. We show that smart-PATH clearly outperforms the classical search approaches and improves on the solution quality up to 48.3% in comparison with CACO and the execution time up to 64.9% in comparison with Bellman-Ford shortest path method.

II. RELATED WORKS

The research on the path planning problem started in late 60’s. Afterwards, several research initiatives, aiming at providing different solutions in both static and dynamic environments, have emerged. Numerous approaches to design these solutions have been attempted which can be widely classified into two main categories [2]:

- Classical methods: they are variations and/or combination of a few general approaches such as Roadmap [3], Potential Field [4], Cell Decomposition [5] and Mathematical programming. These methods dominated this field during the first 20 years. However, they were deemed to have some deficiencies in global optimization, time complexity, robustness, etc.

- Heuristic methods: these methods were designed to overcome the aforementioned limits of classical methods. Several techniques to solve the path planning problem have emerged including Genetic Algorithms [6], Neural Networks [7], Tabu Search [8], Ant Colony Optimization [9], Particle Swarm Optimization (PSO) [10], in addition to several other techniques.

Among all these techniques, ACO and GA are the most widely used heuristics in solving the path planning problem [11], [12],[13], [14] and [15]. Some other research efforts have proposed hybrid solutions combining ACO and GA. The difference with our work consists in using different models of the environment. For instance, in [16] two solutions were proposed for path planning where the first is based on GA and the second is based on ACO. A comparison study was performed between both techniques on a real-world deployment of multiple robotic manipulators with specific spraying tools in an industrial environment. In this study, it has been argued that both solutions provide very comparable results for small problem sizes, but with increasing the size and the complexity of the problem, the ACO-based algorithm achieves better quality of solution at the cost of a higher execution time, as compared to the GA-based algorithm. In [17], the authors presented an intensified ACO algorithm for the global path planning problem and compared the performance of their proposal with GA. It was proven that both solutions are able to find the optimal path. Nonetheless, the ACO algorithm was shown to be more robust and effective in finding the optimal path. In [18], a comparative performance study of the two aforementioned approaches has been conducted. The algorithms have been tested in three workspaces with different complexities and it was demonstrated that the ACO method was more effective and outperformed the GA method in terms of time complexity and number of iterations.

Although, ACO and GA have shown their effectiveness in the resolution of path planning problem, these two techniques suffer from some limits. In fact, ACO has a stronger local search capability and faster convergence speed but the algorithm can easily sink into a local optimum if the size of the problem is large. On the other side, GA belongs to random optimize processes, so the local convergence problem does not appear; however, this makes its convergence speed slower. Thus, we believe that a hybrid ACO and GA approach could be a promising alternative, which represents the focus of our proposal. In the literature, some works proposed solutions based on the combination between ACO and GA. For instance, in [19] the authors presented a path planning method based on a hierarchical hybrid algorithm that aims at finding a path in a road traffic network. The network is divided into several sub-networks; ACO is applied on each sub-network, the best paths generated by ant colony optimization algorithms will be the initial population of genetic algorithms. Simulations were conducted and showed the effectiveness of the hybrid algorithm. In [20], a combination between GA and ACO algorithms to solve the robot navigation problem was presented. A special function was proposed to improve the quality of the paths generated by the ants at the beginning of the algorithm. Crossover and mutation operators are applied to improve the quality of solution generated by the ACO algorithm.

In this paper, we propose a new solution based on a combination of ACO and GA. Our solution fundamentally differs from others as it adopts a different environment map model based on WSNs. The optimal paths made by ACO at every generation are taken as an initial population of a crossover operator. The crossover is a kind of post-optimization or local search that avoids getting trapped in a local optimum. Our solution is presented in the next section.

III. THE SMARTPATH ALGORITHM

A. Environment Model

![Fig. 1: Environment Model](image-url)
Many global path-planning methods use a grid-based model to represent the workspace of a robot. In such a model, the space is partitioned into grids. An obstacle may occupy one or more grids, depending on the size of the obstacle relative to the size of the vehicle. In this paper, we have considered another environment model that reduces the complexity of the problem, such as the model used in [12], [19]. The mobile robot workspace is described by a 2-D map. The map concentration. On the other hand, GA [6] is another search method used for solving several optimization problems, and its fitness evaluation, selection, crossover and mutation are the base operations of GA. The key idea of smartPATH consists in using wireless sensor nodes as:

- Signposts to locate the mobile robot, using a certain localization algorithm such as RSS-based localization [21];
- Waypoints to guide the robot towards the desired destination.

The robot must choose the best series of nodes to follow in order to find the best path, which will be encoded as a sequence of sensor nodes IDs; for example 0-4-5-10-9-15 is a path with start node 0 and goal node 15. The robot should move between sensor nodes that are connected, if two sensor nodes are not connected this means that there is an obstacle between them. The obstacles are detected by the robot capabilities.

B. Description of the smartPATH Algorithm

ACO [9] is a metaheuristic inspired from the foraging behavior of ants; in the nature ants are able to select the shortest path from their nest to the food source. In fact, ants put pheromone in their paths, rendering ants converging towards the shortest path with the maximum pheromone concentration. On the other hand, GA [6] is another search method used for solving several optimization problems, and it is based on the laws of natural selection and genetics: fitness evaluation, selection, crossover and mutation are the base operations of GA. The key idea of smartPATH consists in combining these two approaches together. The smartPATH algorithm comprises two phases: (i.) improved ACO algorithm (IACO) that ensures a fast convergence towards optimal path through intelligence probabilistic path selection mechanisms (ii.) Genetic Algorithm phase acting as a post optimization (or local search) that improves the quality of solution found in the previous phase. The pseudo-code of smartPATH is presented in Algorithm 1. In what follows, we describe the two phases.

1) Improved ACO Phase (IACO): The ACO phase consists in generating a set of optimal paths by an improved ACO algorithm. The model of the workspace is abstracted to a graph, and the ants must find the shortest path in the graph from the start to the end positions. Each ant has a current position in the graph and it can move from the current sensor node to another sensor node, and has to make a decision about its new position. It is executed in three steps:

- **Step 1: Paths’ Finding:** The ants search for the shortest path in the environment from the start to the goal positions. In a current position, an ant has to smartly decide the next sensor node on its path towards the destination. We have devised two new functions to optimize the decision of an ant in the path planning process:
  
  - **The Heuristic distance information probability:** it is used in the first iteration of the algorithm: indeed, in the classical ACO algorithm, the ant’s decision is made based on the transition rule probability function [9], which depends on the quantity of pheromone. However, in the initial paths construction, the quantity of pheromone is not significant, and has not a great impact on the construction of the solution. This renders the choice of the ant not obvious and increases the time for finding the optimal path. In order to avoid these shortcomings, we devised the

---

**Algorithm 1. The SmartPATH Algorithm**

```
1: S = (S_i)_{i=1}^{n}: Set of sensor nodes
2: S_initial : Initial position of the robot
3: S_goal : Goal position of the robot
4: NCACO : Maximum number of iterations of IACO algorithm
5: NCGA : Maximum number of iterations of GA algorithm
6: place m ants in S_initial
7: MaxLength: Length of the robot’s path
8: S_AGO: Set of the best paths found by ACO
9: S_GA: Set of the best paths found by GA
10: repeat
11: for each ant k do
12: Add S_initial to the ants paths
13: end do
14: while MaxLength is not reached do
15: for each ant k do
16: if (the ant k is not discarded) then
17: if (NCACO = 1) then
18: the ant chooses the next sensor node S_i according to the heuristic distance information probability given in equation (1)
19: Else the ant chooses the next sensor node S_i according to the modified transition rule probability given in equation(2)
20: end if
21: Add S_i to the kth ant’s path
22: calculate the distance traveled by the kth ant DT_k
23: if (DT_k ≥ the length of the current shortest path) then
24: the ant is eliminated
25: end if
26: end if
27: end do
28: end do
29: Generate BestPath_AGO found in the current iteration
30: MBestPath_AGO = mutation operator(BestPath_AGO)
31: S_AGO=S_AGO ∪ MBestPath_AGO
32: Update pheromone
33: Until (endACO)
34: Generate BestPath_ACO from S_ACO
35: repeat
36: Select randomly two paths P_1 and P_2
37: NewPath = crossover operator(P_1, P_2)
38: S_GA=S_GA ∪ NewPath
39: Until (endGA)
40: select BestPath_GA from S_GA
41: if (BestPath_ACO length ≤ BestPath_GA length) then
42: RobotPath=BestPath_ACO
43: else RobotPath=BestPath_GA
44: end if
```

---
heuristic distance information probability function, introduced in [11], to calculate the probabilities of transition. This function is expressed as:

\[ p^k_{i,j} = \frac{\tau^k_{i,j} * D^\beta}{\sum_{j \in \text{allowed}(i)} (\tau^k_{i,j} * D^\beta)} \] (1)

Where \( p^k_{i,j} \) is the probability of transition of the \( k \)th ant from sensor node \( i \) to sensor node \( j \), \( \text{allowed}(i) \) is the set of sensor nodes in the neighborhood of sensor node \( i \) which the \( k \)th ant has not visited yet, \( d_{j,\text{goal}} \) is the Euclidian distance from the sensor node \( j \) to the destination, \( \text{Max} D_{\text{allowed}(i),\text{goal}} \) is the maximum of all \( d_{j,\text{goal}} \) and \( \lambda, \mu, \alpha \) are constants.

- \textit{A modified transition rule probability:} It is used for the rest iterations of the algorithm; this function is defined by the following formula:

\[ p^k_{i,j} = \frac{\tau^k_{i,j} * D^\beta}{\sum_{j \in \text{allowed}(i)} (\tau^k_{i,j} * D^\beta)} \] (2)

Where \( p^k_{i,j} \) is the probability of transition of the \( k \)th ant from sensor node \( i \) to sensor node \( j \), \( \tau_{i,j} \) denotes the quantity of pheromone in the edge joining node \( i \) and node \( j \), \( \text{allowed}(i) \) is the set of neighboring nodes of node \( i \) which the \( k \)th ant has not visited yet, \( \alpha \) and \( \beta \) are two parameters to weight the significance of pheromone and distance in the selection of next node and \( D = \frac{1}{d_{i,j} + \lambda \cdot d_{j,\text{goal}}} \) where \( d_{i,j} \) is the Euclidian distance between the current node \( i \) and the next node \( j \) and \( d_{j,\text{goal}} \) denotes the Euclidian distance between the next node \( j \) and the goal node. The reason behind using the parameter \( D \) instead of \( \eta_{i,j} = \frac{1}{d_{i,j}} \) used in the conventional transition rule probability of CACO is that \( D \) has a greater attraction to the goal wireless sensor node. As a consequence, it reduces the number of bad solutions that might be selected by the ants, and thus, accelerates the convergence speed to find the optimal path.

\textbf{Remark: Varying the value of } \alpha \text{ and } \beta \text{. It has to be noted that } \alpha \text{ and } \beta \text{ are two important constants of the modified transition rule probability. These parameters indicate the importance of the remaining pheromone on each edge joining two nodes, and the importance of the heuristic information, respectively. In CACO, } \alpha \text{ and } \beta \text{ don’t change during the execution of the algorithm, and this induces a negative impact on the performance of the algorithm. Thus, we propose varying the values of } \alpha \text{ and } \beta \text{ as follows. In the beginning of the algorithm, the impact of the distance on the transition probability is more significant than the impact of pheromone, so we must consider values such that } \alpha < \beta. \text{ After a period of time the influence of pheromone becomes more important as several valid paths would appear; thus we consider values such that } \alpha \gg \beta. \text{ }

- \textit{Step 2: Path Optimization:}

  - \textit{Control of the ants:} During the construction of the paths, the ants are monitored, meaning that if an ant walks more than a certain threshold distance, which is the cost of the current best solution found during the searching process, it will be discarded (as it will certainly not produce the optimal path). This helps to reduce the search space and the execution time by early elimination of bad solutions.

  - \textit{Mutation operator:} After an iteration of the IACO algorithm, a near optimal path will be generated. Then, a mutation operator is applied on this path in the quest of getting a better solution. The main idea of the mutation operator is to check all the nodes, except the start and the goal nodes, and try to change one or more nodes in the path if the length of the resulting new path is shorter than the length of the generated path.

- \textit{Step 3: Pheromone Update:} After each iteration of the smartPATH algorithm, the quantity of pheromone is updated by all the ants that have built solutions. The quantity of pheromone \( \tau_{i,j} \) associated with each edge joining two sensor nodes \( i \) and \( j \) is updated as follows:

\[ \tau_{i,j}(t + 1) = (1 - \rho) * \tau_{i,j}(t) + \sum_{k=1}^{m} \Delta \tau^k_{i,j}(t) \] (3)

where \( 0 \leq \rho \leq 1 \) is the evaporation rate, \( m \) is the number of ants and \( \Delta \tau^k_{i,j} \) is the quantity of pheromone laid on edge \((i, j)\) joining two sensor nodes \( i \) and \( j \) by an ant \( k \).

\[ \Delta \tau^k_{i,j}(t) = \left\{ \begin{array}{ll} \frac{Q}{L_k} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{array} \right. \] (4)

where \( Q \) is a constant and \( L_k \) is the length of the tour constructed by ant \( k \).

2) \textit{GA Phase.:} This phase is a kind of post optimization or local search that improves the quality of solution found by IACO. It consists in applying a modified crossover operator on the set of optimal paths generated by the ACO algorithm after \( N \) iterations. The key idea of the proposed crossover operator consists in selecting two common nodes \( N_1 \) and \( N_2 \) from two randomly selected parent paths \( P_1 \) and \( P_2 \) and comparing the three sub-parts of \( P_1 \) and \( P_2 \) existing \((i. )\) before \( N_1 \), \((ii. )\) between the two selected nodes \( N_1 \) and \( N_2 \) and \((iii. )\) after the second node \( N_2 \). The best parts are selected to form the new path. The new path has the same length as \( P_1 \) and \( P_2 \). Consider the following two paths as an example:

\[ P_1 = S_0 \rightarrow S_4 \rightarrow S_6 \rightarrow S_7 \rightarrow S_9 \rightarrow S_{10} \rightarrow S_{11} \rightarrow S_{12} \rightarrow S_{15} \]
\[ P_2 = S_0 \rightarrow S_3 \rightarrow S_6 \rightarrow S_4 \rightarrow S_5 \rightarrow S_{9} \rightarrow S_{10} \rightarrow S_{15} \rightarrow S_{15} \]

The two common nodes selected are \( S_4 \) and \( S_{10} \). We compare the different sub-parts ("S0-S4-S6-S7-S9-S10") and ("S4-S5-S10" and "S4-S6-S7-S9-S10") and ("S11-S12-S15" and "S9-S15-S15") by calculating the Euclidian distance between the different sensor nodes. The new crossover generates only one path formed by the shortest parts from the two parents.
P1 and P2: The length of the generated path is smaller than the length of P1 and P2, so the target sensor node (S15) is added to the path in order to have equality in terms of length between all paths.

<table>
<thead>
<tr>
<th>New Path:</th>
<th>S0</th>
<th>S4</th>
<th>S5</th>
<th>S10</th>
<th>S9</th>
<th>S15</th>
<th>S15</th>
<th>S15</th>
</tr>
</thead>
</table>

C. Discussion

The smartPATH algorithm has several advantages which makes it believed to be better than the CACO algorithm. In fact, it is reinforced by several mechanisms to quickly converge to good solutions, dynamically prune the search space and reduce the execution time. For instance, the dynamic setting of $\alpha$ and $\beta$ parameters during the search operation leads to diversify the exploration of the state space and thus to improve the quality of solutions. Similarly, the mutation operator represents a greedy local search technique applied on solutions produced after each IACO iteration. This also improves the solution quality. On the other hand, some other techniques contribute to reduce the search space and time. As a matter of fact, smart-PATH discards weak ants that went astray in paths longer than currently best path. Another possible extension of this technique would be to discard paths that would be predicted to be longer than the currently best path using for instance the residual distance to the destination. This will be investigated in the future. Moreover, the modified transition rule probability function that incorporates an underestimation of the remaining distance to the destination efficiently predicts expensive paths and exclude them from the solution set at an early stage. This idea is similar to the heuristic used in the evaluation function of the A* technique.

In the next section, we will demonstrate through simulations the validity of our intuition about the effectiveness of the aforementioned techniques to simultaneously improve the solution quality and reduce the execution time.

IV. Performance Evaluation

A. Simulation Model

In this section, we present an extensive simulation study of the proposed smartPath algorithm applied to four environments with different complexities illustrated in Fig.2. We implemented a simulation MATLAB model. All simulations are implemented on a PC with an Intel Core i3 CPU @ 2.27GHz and 4GB of RAM under Windows 7.

- **Environment 1:** (Fig. 2.a) this environment is the simplest one, it has the smallest number of obstacles and the smallest number of nodes (16 sensor nodes). The mobile robot has to find the shortest and collision-free path from sensor node 0 to sensor node 15.
- **Environment 2:** (Fig. 2.b) this environment is of medium complexity, and comprises 27 sensor nodes. The robot has to reach sensor node 25 from the starting sensor node 1.
- **Environment 3:** (Fig. 2.c) this environment is a slightly complicated environment, it contains 50 sensor nodes. The start position of the mobile robot is sensor node 1 and the goal position is sensor node 50.
- **Environment 4:** (Fig. 2.d) this environment is of high complexity, it contains the largest number of sensor nodes and obstacles with 150 sensor nodes. The start position of the mobile robot is sensor node 1 and the goal position is sensor node 148.

B. Simulation Results

In this section we present an extensive simulation study to evaluate the efficiency of the smartPATH algorithm. The objective of the simulation is two-folded: First, we present a comparison between smartPATH, the Improved ACO algorithm (without considering the GA part in smartPATH), smartPATH-WHF (without considering the heuristic function in the IACO phase), the classical ACO, the classical GA approach and Bellman-Ford Exact algorithm. This comparative study will help to demonstrate the added value of combining ACO and GA approaches, and the improvement carried out by modifying the ACO algorithm. Second, we examine the impact of varying a set of ACO parameters, namely the number of ants and the evaporation rate, on the quality of solution and on the execution time. Our goal is to identify the most appropriate parameters’ settings that produce the best results. To evaluate the efficiency of the smartPATH algorithm, three performance metrics were assessed: (1) the path length: it represents the length of the shortest path found by an algorithm, (2) the execution time: it is the time spent by an algorithm to find its best (or optimal) solution, (3) the number of iterations: it represents the number of repetition that one algorithm takes to converge to the shortest path. For each environment, we performed 30 different runs for each algorithm. For each run, we recorded the length of the generated path, the execution time and the index of iteration that corresponds to the best (or optimal) solution. The parameters of the smartPATH algorithm are presented in Table I.
1) Optimality and Convergence Time: From Fig. 3 and Table II, it can be noticed that using the default value of ants (n=10 ants), the smartPATH algorithm generates the same optimal path generated by Bellman-Ford shortest path method for environments 1, 2 and 3. However, for environment 4, much larger number of ants (n=20) is needed in order to produce the same optimal path found by Bellman-Ford shortest path method as 10 ants are insufficient to explore the environment and to find the best solution. In addition, Fig. 4 and Table III show that smartPATH finds the optimal solution much faster than Bellman-Ford exact method. The results prove the efficiency and the importance of using heuristic method for solving global path planning problem.

Also, we notice that smartPATH generates better solutions but in a larger time and more iterations than smartPATH-WHF except for environment 1, both algorithms provide the same solution but smartPATH-WHF generates sub-optimal paths (18 out of 30 runs), which demonstrates the efficiency of using the heuristic distance information probability function in the beginning of the algorithm.

Furthermore, we observe from Fig. 3 and Table II that smartPATH provides the optimal paths for all the four environments, in contrast to the Classical ACO algorithm and the GA algorithm that fail to find the optimal solution in medium-scale and large-scale environments. However, for small problem size (small number of sensor nodes as in environment 1), the quality of solution generated by the algorithms after 30 runs is the same, expect for CACO which finds sub-optimal paths (0-4-6-7-13-15), (0-3-8-13-15) most of the time (27 out of 30 runs), which demonstrates the efficiency of using the heuristic distance information probability function in the beginning of the algorithm.

In addition, it is noted also from Table III, Table IV and Fig. 4, that CACO finds its best solution, which is not optimal, in a shorter time as compared to smartPATH, which finds better and optimal solution in slightly larger time and greater number of iterations. This does not mean that the CACO algorithm is faster than smartPATH because the former fails to find the optimal solution after the 30 iterations of each of the 30 runs. The GA algorithm always exhibits the lowest quality and longest execution time, which is expected as it is known to have slower convergence speed as compared to ACO. For that reason, we only used GA for post optimization reasons to improve on the ACO solution quality. For environment 4, the GA algorithm fails to find a solution as it is always blocked in a dead-end position.

Moreover, Fig. 3 and Table II show that for small and medium scale environments (as in environment 1 and environment 2), smartPATH generates the same solution found by IACO. However, for large scale environments (environment 3 and environment 4) smartPATH provide better solution than IACO in a larger time and iteration. One possible reason why the IACO does not generate the same solution in large environments is that the first and the second environments improve on the ACO solution quality. For environment 4, the GA algorithm fails to find a solution as it is always blocked in a dead-end position.

Moreover, Fig. 3 and Table II show that for small and medium scale environments (as in environment 1 and environment 2), smartPATH generates the same solution found by IACO. However, for large scale environments (environment 3 and environment 4) smartPATH provide better solution than IACO in a larger time and iteration. One possible reason why the IACO does not generate the same solution in large environments is that the first and the second environments prove the efficiency and the importance of using heuristic method for solving global path planning problem.

### Table I: smartPATH parameter specification

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mc: number of ants</td>
<td>10</td>
</tr>
<tr>
<td>@: Pheromone trail coefficient</td>
<td>0.5 and 1</td>
</tr>
<tr>
<td>_j: Heuristic coefficient</td>
<td>0.5 and 1</td>
</tr>
<tr>
<td>@: evaporation trail</td>
<td>0.99</td>
</tr>
<tr>
<td>Q: Constant</td>
<td>100</td>
</tr>
<tr>
<td>_j(t): The initial pheromone value</td>
<td>0.05</td>
</tr>
<tr>
<td>NC_ACO: Number of iterations of ACO algorithm</td>
<td>30</td>
</tr>
<tr>
<td>W: calibration parameter</td>
<td>10</td>
</tr>
<tr>
<td>D: calibration parameter</td>
<td>2</td>
</tr>
<tr>
<td>X: calibration parameter</td>
<td>2</td>
</tr>
<tr>
<td>NC_GA: Number of iterations of GA algorithm</td>
<td>20</td>
</tr>
</tbody>
</table>

### Table II: Length of the Generated Paths of smartPATH, IACO, smartPATH-WHF, CACO, GA and Bellman-Ford Algorithms in Different Environments

<table>
<thead>
<tr>
<th>Environments</th>
<th>smartPATH</th>
<th>IACO</th>
<th>smartPATH-WHF</th>
<th>CACO</th>
<th>GA</th>
<th>Bellman-Ford</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment 1</td>
<td>117.003</td>
<td>133.7453</td>
<td>113.9003</td>
<td>125.2995</td>
<td>145.4717</td>
<td>220.1146</td>
</tr>
<tr>
<td>Environment 4</td>
<td>1.7671</td>
<td>1.7671</td>
<td>1.7671</td>
<td>1.7671</td>
<td>1.7671</td>
<td>1.7671</td>
</tr>
</tbody>
</table>

### Table III: Execution times of smartPATH, IACO, smartPATH-WHF, CACO, GA and Bellman-Ford Algorithms in Different Environments

<table>
<thead>
<tr>
<th>Environments</th>
<th>smartPATH</th>
<th>IACO</th>
<th>smartPATH-WHF</th>
<th>CACO</th>
<th>GA</th>
<th>Bellman-Ford</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment 1</td>
<td>0.1685</td>
<td>0.1685</td>
<td>0.11492</td>
<td>0.1248</td>
<td>1.6233</td>
<td>6.76</td>
</tr>
<tr>
<td>Environment 2</td>
<td>0.1685</td>
<td>0.1685</td>
<td>0.11492</td>
<td>0.1248</td>
<td>1.6233</td>
<td>6.76</td>
</tr>
<tr>
<td>Environment 3</td>
<td>0.1685</td>
<td>0.1685</td>
<td>0.11492</td>
<td>0.1248</td>
<td>1.6233</td>
<td>6.76</td>
</tr>
<tr>
<td>Environment 4</td>
<td>0.1685</td>
<td>0.1685</td>
<td>0.11492</td>
<td>0.1248</td>
<td>1.6233</td>
<td>6.76</td>
</tr>
</tbody>
</table>

### Fig. 3: Length of the Generated Paths of smartPATH, IACO, smartPATH-WHF, CACO, GA and Bellman-Ford Algorithms in Different Environments

### Fig. 4: Execution times of smartPATH, IACO, smartPATH-WHF, CACO, GA and Bellman-Ford Algorithms in Different Environments
are small in size and optimal solutions are easy to obtain, but in large and complex environments encompassing large variety of potential good solutions, so the IACO algorithm can fall into a local minimum very quickly, which is solved using the GA phase in smartPATH as it is shown in Table IV, for environment 1 and 2 smartPATH converges to the optimal solution in the IACO phase. However, for environment 3 and 4 smartPATH converges to the optimal path in the GA phase.

The good performance of the smartPATH algorithm in terms of quality of solution is justified through the marriage of the two complementary approaches. Indeed, the GA algorithm is a kind of post optimization or local search that improves the quality of solution found by the IACO algorithm, which can provide a non-optimal solution if the size of the problem is large.

Also, we clearly observe that IACO outperforms CACO in terms of quality of solution for medium and large scale environments, which demonstrates the improvement carried out by modifying the classical ACO algorithm.

The results clearly show the benefit from using a hybrid ACO-GA approach in the path planning problem and demonstrate that smartPATH reduces the execution time up to 64.9% in comparison with Bellman-Ford exact method (in the case of environment 3), while it improves the solution quality up to 48.3% in comparison with CACO (in the case environment 4).

2) Impact of ACO parameters: The ACO algorithm is a very flexible and configurable algorithm as there are a lot of parameters that need to be well selected such as the number of ants, the pheromone factor $\alpha$, the heuristic factor $\beta$, the evaporation factor $\rho$. The behavior of the IACO algorithm depends strongly on the values given to these parameters which affect the performance of the algorithm. In this section, our aim is to find the most appropriate values for the IACO algorithm parameters, such that the algorithm converges faster to a satisfying solution for the four tested environments. Simulations were performed for different values of the number of ants, evaporation trail rate $\rho$ in order to assess the behavior of the algorithm with different parameters’ settings. In each experiment one parameter is varied, and the others are all kept fixed to their default values.

- **Impact of variation of the number of ants:** In this paragraph, we examine the effect of varying the number of ants on the execution time and the quality of solution generated by the smartPATH algorithm. In each iteration, we fix the number of ants and we perform 30 runs of the algorithm and we record the length of the generated path and the execution time. Fig 5 shows the impact of varying the number of ants on the path length and Fig. 6 represents the impact of varying the number of ants on the execution time. These figures show that for a small value of the number of ants (inferior to 10 ants), the smartPATH algorithm generates a non-optimal solution in faster time for all the four environments. Whereas, using 10 ants, smartPATH generates the optimal path in environment 1, 2 and 3. For environment 4, a larger number of ants (20 ants) is needed to provide the optimal solution. Fig. 6 shows that larger values of ants reduce the execution times for small and medium scale environments. However, for large scale environments the increasing of the number of ants leads to larger execution times.

- **Impact of variation of the evaporation factor:** All simulations are done for fixed number of ants: 10 ants for environment 1, 2 and 3, and 20 ants for environment 4. The results of simulation are depicted in Fig. 8 and Fig. 7. The simulation results prove that the variation of the evaporation factor $\rho$ has an impact only on the execution times in the case of environment 1, 2 and 3. However, for environment 4 the variation of this parameter influences also the quality of solution. From Fig. 8 it can be noticed

<table>
<thead>
<tr>
<th>Environment</th>
<th>smartPATH</th>
<th>IACO</th>
<th>smartPATH- WHF</th>
<th>CACO</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment 1</td>
<td>IACO:3.5</td>
<td>5</td>
<td>IACO:3.5,8</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>Environment 2</td>
<td>IACO:1.5,20</td>
<td>10,20</td>
<td>IACO:5,12.2</td>
<td>3.4</td>
<td>18</td>
</tr>
<tr>
<td>Environment 3</td>
<td>GA:1,2,3</td>
<td>20</td>
<td>GA:1,2,3,4</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>Environment 4</td>
<td>GA:1,2,3</td>
<td>10</td>
<td>GA:1,2,3,4</td>
<td>3.4</td>
<td>21</td>
</tr>
</tbody>
</table>

Table IV: Iterations of smartPATH, IACO, smartPATH-WHF, CACO, GA and Bellman-Ford Algorithms in Different Environments

![Fig. 5: Impact of Variation of the Number of Ants on the Path Length in Different Environments](image)

![Fig. 6: Impact of Variation of the Number of Ants on the Execution Time in Different Environments](image)
that large $\rho$ values produce a faster convergence for all the three environments. However, small values ($\rho = 0.1$) produce good results in the case of environments 2. From Fig. 7, it is shown that larger values of $\rho$ provide better results in a minimum amount of time.

V. CONCLUSION

In this paper, we proposed smartPATH, a new hybrid ACO-GA algorithm to solve the global robot path planning in a static environment. The environment was modeled by a set of connected sensor nodes acting as waypoints to guide the robot to its destination. The smartPATH algorithm comprises two phases where the first phase applies an optimized ACO algorithm whose output is later processed by a GA via a modified crossover operator. We presented an extensive simulation study to evaluate smart PATH and compare it against classical ACO, GA algorithms and Bellman-Ford shortest path method. It has been shown that the modifications applied to the IACO algorithm contributed to improve the solution quality and reduce the search space and time. Also, the crossover operator of GA phase provided a fast post optimization that enables to fine tune solutions of the previous IACO phase. Currently, we are working towards the implementation of smartPATH on a real-world robotic application in the context of the RTRACK project to demonstrate the effectiveness and feasibility of our approach. In addition, we will consider improving the proposed heuristics by finding new conditions to reduce the search space at an early stage. This can be done through prediction techniques for future bad paths. The investigation of other intelligent techniques for path planning is also in our agenda.

REFERENCES