CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

smartPATH: An Efficient Hybrid ACO-GA Algorithm for Solving the Global Path Planning Problem of Mobile Robots

Imen Chaari
Anis Koubâa*
Hachemi Bennaceur
Adel Ammar
Khaled Al-Shalfan.

*CISTER Research Center

CISTER-TR-140413

2014/07/04

smartPATH: An Efficient Hybrid ACO-GA Algorithm for Solving the Global Path Planning Problem of Mobile Robots

Imen Chaari, Anis Koubâa*, Hachemi Bennaceur, Adel Ammar, Khaled Al-Shalfan.

*CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: aska@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

Path planning is a fundamental optimization prob- lem that is crucial for the navigation of a mobile robot. Among the vast array of optimization approaches, we focus in this paper on Ant Colony Optimization (ACO) and Genetic Algorithms (GA) for solving the global path planning problem in a static environment, considering their effectiveness in solving such a problem. Our objective is to design an efficient hybrid algorithm that takes profit of the advantages of both ACO and GA approaches for the sake of maximizing the chance to find the optimal path even under real-time constraints. In this paper, we present smartPATH, a new hybrid ACO-GA algorithm that relies on the combination of an improved ACO algorithm (IACO) for efficient and fast path selection, and a modified crossover operator to reduce the risk of falling into a local minimum. We demonstrate through extensive simulations that smartPATH outperforms classical ACO (CACO), GA algorithms. It also outperforms the Dijkstra exact method in solving the path planning problem for large graph environments. It improves the solution quality up to 57% in comparison with CACO and reduces the execution time up to 83% as compared to Dijkstra for large and dense graphs. In addition, the experimental results on a real robot shows that smartPATH finds the optimal path with a probability up to 80% with a small gap not exceeding 1m in 98%.

International Journal of Advanced Robotic Systems

OPEN ACCESS ARTICLE

SmartPATH: An Efficient Hybrid ACO-GA Algorithm for Solving the Global Path Planning Problem of Mobile Robots

Regular Paper

Imen Châari[1,3,*], Anis Koubâa[2,3,4], Sahar Trigui[1,3], Hachemi Bennaceur[5], Adel Ammar[5] and Khaled Al-Shalfan[5]

1 PRINCE Research Unit, University of Manouba (ENSI), Tunisia
2 Prince Sultan University, Saudi Arabia
3 Cooperative Robots and Sensor Networks (COINS) Research Group, Saudi Arabia
4 CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal
5 Research Unit of Sciences and Technology, Al-Imam Mohamed bin Saud University, Saudi Arabia
* Corresponding author E-mail: imen_chaari07@yahoo.fr

Received 22 Sep 2013; Accepted 26 Mar 2014

DOI: 10.5772/58543

Abstract Path planning is a fundamental optimization problem that is crucial for the navigation of a mobile robot. Among the vast array of optimization approaches, we focus in this paper on Ant Colony Optimization (ACO) and Genetic Algorithms (GA) for solving the global path planning problem in a static environment, considering their effectiveness in solving such a problem. Our objective is to design an efficient hybrid algorithm that takes profit of the advantages of both ACO and GA approaches for the sake of maximizing the chance to find the optimal path even under real-time constraints. In this paper, we present smartPATH, a new hybrid ACO-GA algorithm that relies on the combination of an improved ACO algorithm (IACO) for efficient and fast path selection, and a modified crossover operator to reduce the risk of falling into a local minimum. We demonstrate through extensive simulations that smartPATH outperforms classical ACO (CACO), GA algorithms. It also outperforms the Dijkstra exact method in solving the path planning problem for large graph environments. It improves the solution quality up to 57% in comparison with CACO and reduces the execution time up to 83% as compared to Dijkstra for large and dense graphs. In addition, the experimental results on a real robot shows that smartPATH finds the optimal path with a probability up to 80% with a small gap not exceeding $1m$ in 98%.

Keywords Mobile Robots, Path Planning, Ant Colony Optimization, Genetic Algorithms

1. Introduction

1.1 Motivation

The field of mobile robots has been gaining a lot of attention from both research and industrial communities thanks to their great potential in enabling a vast array of applications in different areas including industrial applications, manufacturing and constructions [1], search

and rescue [2], environment exploration [3], elderly people care and others. Navigation is an essential task for any mobile robots application as it defines how the robot perceives its environment (mapping), how the robot locates itself in its environment (localization) and how the robot finds its path in the map (path planning). In this paper, we concentrate on the path planning problem, which has been considered as a fundamental optimization problem in the field of mobile robots.

This paper investigates the path planning problem in the context of the iroboapp research project [4], whose purpose is to design efficient algorithms for robotic applications, with a particular focus on path planning and multi-robot task allocation (MRTA) problems. Indeed, the project aims at studying the adequacy and effectiveness of existing algorithms for solving the path planning problem, compare between their performance, and design hybrid approaches that improve on existing algorithms performance. Roughly, the project aims at responding to this general question: "among all existing optimization approaches defined in the literature, what is the best approach to solve the path planning problem?". This paper is one step towards responding to this question as it puts the light on two important optimization methods used for different problems, namely the Ant Colony Optimization (ACO) and the Genetic Algorithms (GA). This represents the main motivation of this paper. Other optimization approaches are also being investigated, such as Tabu Search and Particle Swarm Optimization.

1.2 The path planning problem

Basically, path planning [5] 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 planned 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.

Path planning makes part of the more generic navigation function of a mobile robot, that ensures how a robot moves effectively in its surrounding environment. The path planning particularly addresses the following question: "What is the best way to go there?". To answer this question, the robot must have a priori knowledge of the environment where he is moving. Basically, the robot must incorporate fundamental navigation functions to be able to plan its path: (*i.*) *Localization*: it helps the robot to know where he is, i.e, specify its location. Several ways are used for localization including GPS in outdoor environments, laser rangefinder, cameras, ultrasound sensors, received signal strength etc. (*ii.*) *Mapping*: the robot needs to have a map of its environment to be able to recognize where he has been moving around so far. The map allows the robot to have a meaning of locations and directions. The map can be set manually into the robot memory (i.e. graph representation, matrix representation) or can be gradually constructed as the robot explore new environments. (*iii.*) *Addressing*: to plan a path, the target position must be well-defined to the robot, which requires an appropriate addressing scheme that the robot can understand. The addressing scheme allows to specify where the robot will go starting from its initial location. Localization and mapping functions when coupled together form a more complex function, that is the Simultaneous Localization and Mapping (SLAM), where a robot builds the map gradually while it localizes itself at the same time.

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 [5], [6]: 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.

1.3 Contributions of the paper:

This paper has three major contributions:

- First, we propose smartPATH, a new hybrid ACO-GA algorithm for robot global path planning. The smartPATH algorithm incorporates the best features of both ACO and GA and combines them to provide a more effective search algorithm. It 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) the definition of a heuristic distance information probability that helps select more appropriate paths at the initial phase of the algorithm, which leads to faster convergence to the optimal solution, (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) the incorporation of an ant control mechanism and a GA mutation operator in the ACO phase to early discard non-relevant solutions and improve the solution quality, respectively. Furthermore, a cross-over GA operator complements the ACO phase to avoid falling into a local optimum.
- Second, we present an extensive simulation study to evaluate the performance of smartPATH and compare it against the classical ACO, the classical GA and the Dijkstra [50] algorithms. We show that smartPATH

clearly outperforms the classical search approaches and improves on the solution quality up to 57% in comparison with the classical ACO. It also outperforms the Dijkstra exact method for solving the path planning problem in dense and large-scale graph environments.
- Third, we implemented smartPATH on a real world robot and we conducted extensive experimental evaluation to demonstrate the feasability of the algorithm, and its effectiveness when operating in real-world.

2. Related Works

2.1 Background

The research on the path planning problem started in late 60's. Nevertheless, most of the effort is more recent and has been conducted during the 80's [7]. 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 [5], [8]:

- **Exact methods:** They are complete methods, they aim to find an optimal solution if one exists or prove that there is no feasible solution such as A* [10], Dijkstra [9], Bellman-Ford, D*, etc. These methods are usually time consuming to reach the optimal solution for hard instances. The classical Dijkstra, for example, may be time inefficient with respect to high connectivity degree graph environments [11] since it has a quadratic time complexity $\mathcal{O}(n^2)$ and processes the whole graph to find the optimal path.
- **Heuristic methods:** These methods were designed to overcome the aforementioned limit of the exact methods. They aim to find a good quality solution in a short time. Several techniques to solve the path planning problem have emerged including Genetic Algorithms [15], Neural Networks [16], Tabu Search [17],[18], Ant Colony Optimization [19], Particle Swarm Optimization (PSO) [20], in addition to several other techniques.

Among all these techniques, ACO and GA are widely used in solving the path planning problem [21], [22],[23], [24], [25] and [26]. In what follows, we present a literature review on different approaches that used ACO, GA, and their combinations for solving the path planning problem.

2.2 ACO Approaches

In paper [27], the authors proposed a hybrid algorithm that combines global and local path planning in dynamic environment that contains U and V-shaped obstacles. In a first step, the algorithm generates an optimal path (off-line) towards the goal based on a modified ACO algorithm considering only the static obstacles. In the second step, the rolling windows technique is applied to avoid dynamic obstacles that appear while the robot navigates, following the path found in the first step, toward the goal position. The authors evaluated the performance of the modified ACO in terms of path length and convergence speed and they compared them against the conventional ACO. They proved that the convergence speed of MACO is faster and the path length generated by MACO is shorter than CACO. Moreover, the different techniques of the rolling window enable the robot to avoid safely dynamic obstacles and also to re-plan its trajectory and choose the shortest one toward the goal position.

In [28], Zhang et al. proposed an improved ACO algorithm. The key differences with the traditional ACO algorithm are (1) the definition of an objective attraction function in the transition rule probability which is based on the attraction of the goal position. (2) The use of a modified pheromone update rule. The interesting point of this paper is the use of the RoboCupRescue simulation system to evaluate the efficiency of the algorithm. The authors compared the new algorithm with the conventional one in terms of path length and number of cycles and they proved that their algorithm provides more accurate results.

In [29], the authors presented a modified ACO algorithm to solve the global path planning problem in an indoor environment for a UAV. The workspace of the UAV is modeled as 3-dimenstional grid map. To overcome the shortcomings of the conventional ACO, the authors proposed a modified pheromone update rule: they proposed to update the quantity of pheromone only on the best path generated after an iteration of the algorithm. They also limited the quantity of pheromone to solve the problem of premature convergence of ACO. To evaluate the robot path, they added a climbing parameter in addition to the path cost, they claimed that this parameter will help the robot to choose the best direction and bypass obstacles. It was demonstrated throw simulation that the algorithm could produce good quality of solution in 10*10 grid map. In [30], Ganganath et al. proposed an off-line path planner for a mobile robot based on Ant Colony Optimization algorithm with Gaussian functions (ACO-2-Gauss). The workspace of the mobile robot is a 3D hilly landscape. Unlike ordinary ACO algorithms, the proposed path planner provides the ants an extra ïňĆexibility in making routing decisions: an ant is allowed to select any point on the circumference of a circle with a radius of R as its next position. Each ant will select a set of circles to move toward the goal position. The pheromone is distributed on the center of the different circles as a 2D Gaussian function. In their simulation study, the authors compared the proposed algorithm against a preceding version: ACO-Gauss, it was demonstrated that the quality of solution of ACO-2-Gauss was improved as compared to ACO-Gauss path planner.

2.3 GA Approaches

GA has also been extensively used for path planning in robotic applications. For instance, in [31] the authors present a path planning algorithm based on pure genetic algorithm. The authors used the grid model to model their environment. In their simulation work, they tested their proposal in several environments that differ by their complexities. The environments are classified in two categories: with and without obstacles. The authors evaluated the path cost while varying the population size and the number of iterations; they proved that their algorithm was effective as it is able to find the optimal

solution for all the different environments. In [26], the authors investigated the capabilities of genetic algorithms (GA) for solving the global path planning problem in large-scale grid maps. They proposed a GA approach for efficiently finding an (or near) optimal path in the grid map. They considered three differents crossover operators, namely one-point crossover, two-point crossover and an improved crossover. A statistical evaluation of the proposed GA approach in terms of solution quality was conducated, and compared against the well-known A* algorithm as a reference. Simulation results showed that GA is able to find the optimal paths in large environments equally to A* in almost all the simulated cases. The execution time was not evaluated in this work. In [32], the authors proposed a new path planning algorithm for a mobile robot based on the genetic algorithms approach. The authors considered both static and dynamic obstacles in an unknown environment. The interesting point of this paper is that the novel algorithm was tested on a real-world application using Pioneer III mobile robot. The authors were positive toward the effectiveness of GA for solving path planning, however we believe that the experimental results was not extensive and still need further investigation as the paper didn't provide a comparison between simulation and experimental studies. In [33], the authors proposed a genetic algorithm with only the crossover operator to improve execution time and computational cost. They used adaptive population size and fixed length chromosomes, each path is represented by two chromosomes, one for x-coordinate and one for y-coordinate. A comparative study between different metaheuristic approaches classified as trajectory-based and population-based approaches for solving the global path planning problem of mobile robots was conducted in [34] . Three methods were evaluated namely tabu search, simulated annealing and GA. It was demonstrated through simulations that simulated annealing outperforms the other planners in terms of execution time, while tabu search was proved to provide the best solution in terms of path length.

2.4 Hybrid ACO/GA Approaches

In the literature, some other research efforts have proposed two different solutions for path planning based on ACO and GA and compared between their performance. For instance, in [35] 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 [36], 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 [37], 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 [38] 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. In their simulation work, the authors proved that the novel algorithm can quickly and accurately find the optimal path in less number of iterations as compared to [39], [40]and [41]. Moreover, a combination between GA and ACO algorithms to solve the robot navigation problem was presented [42]. 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 and avoid falling into a local optimum. It was proved that the improved algorithm generates a better solution in terms of length, direction and execution time or number of iterations as compared to the pure ACO algorithm . Geetha et al. in [43] proposed a hybrid algorithm (ACOG) based on ACO and GA techniques. Darwinian Reproduction, Structure-Preserving Crossover and Structure-Preserving Mutation are the three genetic operation adopted to improve the efficiency of the ACO algorithm and to avoid falling into a local optimum. The authors claimed that their method is a Multi-objective algorithm as it takes into consideration three different parameters: length, smoothness and security of the path. In the simulation work, they compared the algorithm to literature [44], and they argued that the algorithm is able to generate near optimal path.

In [45] the authors presented an ACO-GA based algorithm. The novel algorithm introduces a modified transition rule probability function in the ACO algorithm that eliminate the parameter âĂIJdistance between two positionsâĂİ and also uses an additional parameters γ to control the behavior of the ants. The values of (α, β, γ) in the new transition rule probability are not constant as in the conventional one and they evolve using the GA approach in order to obtain the best values to improve

the accuracy of the algorithm to get the shortest path. The authors tested the new algorithm in a three real Environments, with different complexity, using the mobile robot (Khepera II). They compared their results with respect to the shortest path length obtained with Dijkstra algorithm and they succeed to obtain the shortest paths.

In this paper, we propose a new solution based on a combination of ACO and GA. 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.

3. Ant Colony Optimization and Genetic Algorithms: Background

3.1 Ant Colony Optimization

3.1.1. General Description

Biologists have demonstrated that several ant species are able to select cooperatively the shortest route among a set of alternative routes from their nest to a food source. In fact, each ant leaves information on the path that it has traversed by depositing a chemical substance called pheromone. The ants have tendency to follow a path rich on pheromone rather than a poorer one. Within a fixed period, shorter paths between the nest and the source of food are traversed more often than longer one. So, the quantity of pheromone is accentuated on shorter paths, which in turn augment the number of ants. This is called the reinforcement process.

Inspired from this behavior of real ants, Marco Dorigo proposed the ACO metaheuristic [19] in the early 1990s to find approximate solutions to difficult optimization problems. ACO was initially applied to the travelling salesman problem (TSP), and then it has been used to solve several research problems such as the path planning problem for a mobile robot.

3.1.2. ACO Pseudo-code

In general, all ACO algorithms follow a specific algorithmic scheme presented in *Pseudo-Code 1*.

At first, an initialization takes place during which the initial pheromone value and many other parameters are set. After that a main loop is repeated until a termination condition is reached. At each iteration, the ants are charged to build feasible solutions to the optimization problem then, the generated solutions are possibly improved by applying a local search procedure, and finally the quantity of pheromone is updated: the pheromone value can either increase, as ants deposit pheromone on the components used during search, or decrease, due to pheromone evaporation.

Pseudo-Code 1. Ant Colony Optimization

```
Set parameters, initialize pheromone trails
while termination condition not met do
   ConstructAntSolutions
   ApplyLocalSearch(optional)
   UpdatePheromones
endWhile
```

3.1.3. Classical ACO for Robot Path Planning

In this section we describe the classical ACO algorithm applied to the path planning problem. We assume that the working space is modeled as a graph. At each iteration of the ACO algorithm, an artificial ant is charged to build a path for the mobile robot from the start position to the goal position by walking on the edges of the graph and visiting each vertex only once. The next vertex is selected stochastically according to the Transition Rule probability expressed by the following formula:

$$p_{i,j}^{k} = \frac{\tau_{i,j}^{\alpha} * \eta_{i,j}^{\beta}}{\sum_{j \in allowed(i)} (\tau_{i,j}^{\alpha} * \eta_{i,j}^{\beta})} \quad (1)$$

Where $p_{i,j}^{k}$ is the probability of transition of the k^{th} ant from vertex i to vertex j, $\tau_{i,j}$ denotes the quantity of pheromone in the edge joining vertex i and vertex j,$\eta_{i,j} = \frac{1}{d_{i,j}}$ where $d_{i,j}$ is the Euclidian distance between the current node i and the next node j , $allowed(i)$ is the set of neighboring nodes of node i which the k^{th} ant has not visited yet, α and β are two parameters to weight the significance of pheromone and distance in the selection of next vertex. At the beginning of the algorithm, an initial pheromone value is affected to the edges of the graph. After each iteration of the 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 vertex i and j is updated as follows:

$$\tau_{i,j}(t+1) = (1-\rho) * \tau_{i,j}(t) + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}(t) \quad (2)$$

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

$$\Delta\tau_{i,j}^{k}(t) = \left\{ \begin{array}{ll} \frac{Q}{L_k} & \text{if ant k used edge(i,j) in its tour} \\ 0 & \text{otherwise} \end{array} \right\} \quad (3)$$

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

3.2 Genetic Algorithms

3.2.1. General Description

Genetic algorithms (GA) [15] is a heuristic method invented in 1975 by John Holland. It is based on the idea of natural selection and genetics to solve optimization and search problems. The notions of chromosome, gene and population constitute the base of GA. In fact, it maintains a population of candidate solutions called chromosomes, the chromosomes evolve from one population to another population using four mechanisms: *(1).* Fitness evaluation, *(2).* Selection, *(3).* Crossover and *(4).* Mutation.

3.2.2. GA Pseudo-code

In general, all GA follow a specific algorithmic scheme presented in *Pseudo-Code 2*.

At the beginning of the algorithm, a set of initial population or chromosomes is generated. In Genetic Algorithm, only the fittest solutions should survive, so that the algorithm converges towards the optimal solution. Each chromosome undergoes an evaluation to assess its fitness; this is achieved by means of fitness function. Then, genetic transformations like crossover and mutation are applied to the fittest solutions yielding new optimal solutions. This process is repeated until a termination condition is verified.

Pseudo-Code 2. Genetic Algorithms

```
1: Generate randomly the initial population
2: while termination condition not met do
3:    Evaluate the different chromosomes of the current
   population using the fitness function
4:    Apply Genetic Operators
5:       Selection
6:       Crossover
7:       Mutation
8: endWhile
```

3.2.3. Classical GA for Robot Path Planning

In this section we describe the classical GA algorithm applied to the path planning problem for a mobile robot. We assume that the environment is modeled as a graph. The first step of GA is the generation of an initial population of chromosomes. Each chromosome represents a path for the mobile robot. Each gene of a chromosome represents a vertex of the graph.

To generate the initial population, vertex that form a path are selected randomly, while adjacent vertex must be connected in order to generate a feasible path. After the generation of the initial population, each path is evaluated and ranked using a fitness function expressed by the following formula:

$$F = \frac{1}{\sum_{i=1}^{n} d_{i,i+1}} \tag{4}$$

Where $d_{i,i+1}$ is the Euclidean distance between vertex i and vertex $i+1$ in the path and n is the number of vertex forming a path. The fittest paths are selected to undergo the genetic operators; crossover and mutation. The objective of these operators is to create new population or solutions from the current solutions that have shown to be good temporary solutions. These steps are repeated until reaching the termination condition.

4. The smartPATH Algorithm

4.1 Environment Model

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 [22], [38]. The mobile robot workspace is described by a 2-D map. The map represents the start point, the goal point, and differentiates the areas that contain obstacles and free spaces. In the free space, a collection of connected waypoints are arranged in random locations, each waypoint represents a location in the environment, and it is characterized by an identifier, a (X,Y) coordinates and a set of adjacent waypoints as depicted in Fig.1 The key idea of our model consists in using the waypoints as:

- Signposts to locate the mobile robot, using a certain localization algorithm such as RSS-based localization [46];
- Landmarks to guide the robot towards the desired destination.

The robot must choose the best series of waypoints to follow in order to find the best path, which will be encoded as a sequence of waypoints IDs; for example 0-4-5-10-9-15 is a path with start position 0 and goal position 15. The robot should move between adjacent waypoints that are connected. if two positions are not connected this means that there is an obstacle between them. We note that the set of waypoints can be obtained through Voronoi decomposition or using the Probabilistic Roadmap Method (PRM) [47] where the waypoints are randomly generated in the environment such that two waypoints are connected if they have direct visibility.

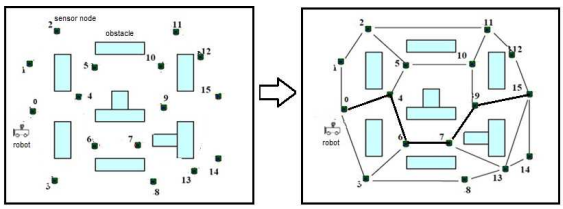


Figure 1. Environment Model

4.2 Description of the smartPATH Algorithm

The key idea of smartPATH consists in combining the ACO and GA 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.

4.2.1. 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 waypoint to another waypoint, and has to make a decision about its new position. It is executed in three steps:

Algorithm 1. The SmartPATH Algorithm

$S = \{S_i\}_{0<i<1}$: Set of waypoints
$S_{initial}$: Initial position of the robot
S_{goal} : Goal position of the robot
NC_{ACO} : Maximum number of iterations of IACO algorithm
NC_{GA} : Maximum number of iterations of GA algorithm
place m ants in $S_{initial}$
MaxLength: Length of the robot's path
S_{ACO}: Set of the best paths found by ACO
S_{GA}: Set of the best paths found by GA
repeat
  for each ant k **do**
    Add $S_{initial}$ to the ants paths
  end do
  while MaxLength is not reached **do**
    for each ant k **do**
      if (the ant k is not discarded) **then**
        if (NC_{ACO}=1) **then**
        getNextWaypoint() according to equation (1)
        Else getNextWaypoint() according to equation (2)
        end if
        Add nextWaypoint to the k^{th} ant's path
        DT_k=calculateTraveledDistance(ant k)
        if ($DT_k \geq$ currentShortesetPathCost) **then**
        remove ant k
        end if
      end if
    end do
  end do
  Generate $BestPath_i$ found in the current iteration
  $MBestPath_i$=mutation operator($BestPath_i$)
  S_{ACO}=$S_{ACO} \cup MBestPath_i$
  Update pheromone
Until{$endACO$}
Generate $BestPath_{ACO}$ from S_{ACO}
repeat
  Select randomly two paths P_1 and P_2
  NewPath=crossover operator(P_1, P_2)
  S_{GA}=$S_{GA} \cup$ NewPath
Until{$endGA$}
select $BestPath_{GA}$ from S_{GA}
if ($BestPath_{ACO}$ length $\prec$ $BestPath_{GA}$ length) **then**
RobotPath=$BestPath_{ACO}$
else RobotPath=$BestPath_{GA}$

- **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 waypoint 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 [19], 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 heuristic distance information probability function, introduced in [21], to calculate the probabilities of transition. This function is expressed as:

$$p_{i,j}^{k} = \frac{((MaxD_{allowed(i),goal} - d_{j,goal}) * \omega + \mu)^{\lambda}}{\sum_{j \in allowed(i)} ((MaxD_{allowed(i),goal} - d_{j,goal}) * \omega + \mu)^{\lambda}} \quad (5)$$

Where $p_{i,j}^{k}$ is the probability of transition of the k^{th} ant from waypoint i to waypoint j, $allowed(i)$ is the set of waypoints in the neighborhood of waypoint i which the k^{th} ant has not visited yet, $d_{j,goal}$ is the Euclidian distance from the waypoint j to the destination, $MaxD_{allowed(i),goal}$ is the maximum of all $d_{j,goal}$ and λ, μ and ω are constants.

- *A modified transition rule probability*: it is used for the rest iterations of the algorithm; this function is defined by the following formula:

$$p_{i,j}^{k} = \frac{\tau_{i,j}^{\alpha} * D^{\beta}}{\sum_{j \in allowed(i)} (\tau_{i,j}^{\alpha} * D^{\beta})} \quad (6)$$

Where $p_{i,j}^{k}$ is the probability of transition of the k^{th} ant from waypoint i to waypoint j, $\tau_{i,j}$ denotes the quantity of pheromone in the edge joining waypoint i and waypoint j, $allowed(i)$ is the set of neighboring waypoint of waypoint i which the k^{th} ant has not visited yet, α and β are two parameters to weight the significance of pheromone and distance in the selection of next waypoint and $D = \frac{1}{d_{i,j} + d_{j,goal}}$ where $d_{i,j}$ is the Euclidian distance between the current waypoint i and the next waypoint j and $d_{j,goal}$ denotes the Euclidian distance between the next waypoint j and the goal waypoint.

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 position. 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.

Remark: Varying the value of α and β. It has to be noted that α and β are two important constants of the modified transition rule probability. These parameters indicate the importance of the remaining pheromone on each edge joining two waypoints, and the importance of the heuristic information, respectively. In CACO, α and β 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 α and β 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 $\alpha \prec \beta$. After a period of time the influence of pheromone becomes more important as several valid paths would appear; thus we consider values such that $\alpha \succ \beta$. In our work, we choose to change the values α and β when the algorithm reaches the maximum number of iterations/2.

- **Step 2: Path Optimization:**
  - *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.
  - *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 waypoints, except the start and the goal waypoints, and try to change one or more waypoints in the path if the length of the resulting new path is shorter than the length of the generated path.
- **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 positions i and j is updated according to equations 2 and 3.

4.2.2. 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 waypoints $N1$ and $N2$ from two randomly selected parent paths $P1$ and $P2$ and comparing the three sub-parts of $P1$ and $P2$ existing *(i.)* before $N1$, *(ii.)* between the two selected waypoints $N1$ and $N2$ and *(iii.)* after the second waypoint $N2$. The best parts are selected to form the new path. The new path has the same length as $P1$ and $P2$. Consider the following two paths as an example:

$P1$**:**	S0	**S4**	S6	S7	S9	**S10**	S11	S12	S15
$P2$**:**	S0	S3	S6	**S4**	S5	**S10**	S9	S15	S15

The two common waypoints selected are S4 and S10. We compare the different sub-parts ("S0" and "S0-S3-S6"), ("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 waypoints. 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 waypoint (S15) is added to the path in order to have equality in terms of length between all paths.

New Path:	S0	**S4**	S5	**S10**	S9	S15	S15	S15	S15

4.3 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 α and β 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.

5. Simulation Analysis

5.1 Simulation Model

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, smartPATH-WHF (without considering the heuristic distance information probability function in the beginning of the IACO algorithm), the classical ACO, the classical GA approach and Dijkstra exact algorithm.

This comparative study will help demonstrate the added value of combining ACO and GA approaches, and the gain resulting from the improved 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 (i.e. cost) 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 and *(3) the number of explored waypoints:*

it represents the number of waypoints visited by the algorithm during the search process. This metric is typically used to assess the time complexity of algorithms and is a good implementation-independent indicator of the algorithm performance.

We considered five environments with different complexities illustrated in Figure 2:

- **Environment 1:** (Figure 2.a) this environment is the simplest one, it has the smallest number of obstacles and the smallest number of waypoints (16 waypoints). The mobile robot has to find the shortest and collision-free path from waypoint 0 to waypoint 15.
- **Environment 2:** (Figure 2.b) this environment is of medium complexity, and comprises 27 waypoints. The robot has to reach waypoint 25 from the starting waypoint 1.
- **Environment 3:** (Figure 2.c) this environment is a slightly complicated enviroment, it contains 50 waypoints. The start position of the mobile robot is waypoint 1 and the goal position is waypoint 50.
- **Environment 4:** this environment is a randomly generated graph, it is of high complexity, it contains a large number of waypoints (300 waypoints). Each waypoint in the graph is connected to more than 150 other waypoints (graph with high node connectivity). The start point is 1 and the goal point is 300.
- **Environment 5:** this environment is a randomly generated graph, it is of high complexity as it contains the largest number of waypoints and obstacles (500 waypoints). Each waypoint in the graph is connected to more than 350 other waypoints (graph with high node connectivity). The start point is 1 and the goal point is 500.

For each environment, we performed 30 different runs for each algorithm to ensure correct statistical analysis of the results (for stochastic simulations, the minimum sample size must be equal to 30 for calculating the confidence interval of the sample output). For each run, we recorded the length of the generated path and the execution time. The execution time of an algorithm in a given environment is the average of the 30 execution times recorded. The simulation parameters of the smartPATH algorithm are presented in Table 1. We mention that the selection of the default paramaters values are guided by the simulation results presented in Section *V.B.2*. We implemented a simulation MATLAB model. All simulations are implemented on a PC with an Intel Core i7 CPU @ 2.4GHz and 8GB of RAM under Windows 8.

Parameters	**Value**
m: number of ants	10
α: Pheromone trail coefficient	1 and 5
β: Heuristic coefficient	1 and 5
ρ: evaporation trail	0.99
Q: Constant	100
$\tau(0)$: The initial pheromone value	0.05
NC_{ACO}: Number of iterations of ACO algorithm	30
ω: calibration parameter	10
μ: calibration parameter	2
λ: calibration parameter	2
NC_{GA}: Number of iterations of GA algorithm	20

Table 1. The smartPATH parameters

5.2 Simulation Results

5.2.1. Optimality and Convergence Time

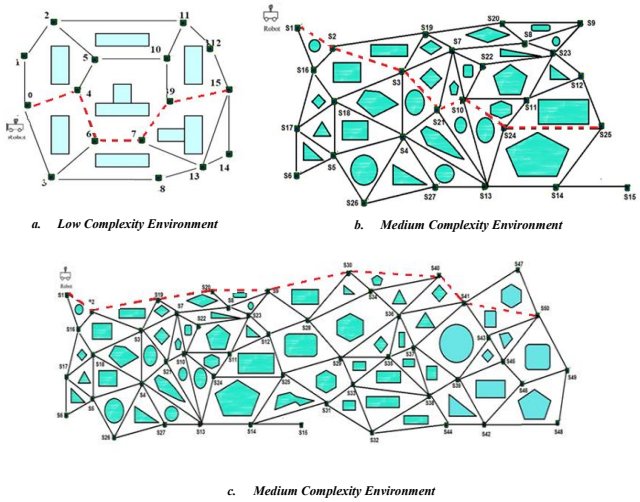
In what follows, we present the main observations pertaining to the solution quality, the convergence time and the number of explored nodes of the smartPATH algorithm and its variants.

Observation 1. For most scenarios, the smartPATH finds the optimal paths as compared to other heuristics and reduces the execution times as compared to Dijkstra exact method in large scale and highly connected graph environments. In fact, we observe in Figure 4 and Figure 3 that smartPATH finds the optimal paths for all environments. We notice from Figure 3 that Dijkstra exact method provides the optimal solution faster than smartPATH for small and medium scale environments (environments 1, 2 and 3). The main reason is that these graphs have small number of nodes with very small connectivity degree (2 to 4 neighbors). With low connectivity, Dijkstra explores a very small number of nodes as can be observed in Table 3. However, for large-scale environments with high nodes connectivity (environment 4 and environment 5), smartPATH clearly outperforms Dijkstra in terms of number of explored waypoints, in addition to reduced execution time. Indeed, smartPATH explores much fewer waypoints (2333 waypoints for environment 4 and 4343 for environment 5) in dense graphs to reach the optimal solution as compared to Dijkstra (32739 waypoints for environment 4 and 108723 for environment 5). The number of expanded nodes demonstrates the efficiency of the smartPATH algorithm in terms of time complexity independently from implementation details. This demonstrates the importance of using heuristic methods for difficult problems, such as in our case large-scale and highly connected graph [11].

As it is observed in Table 4 and Table 3, the gain in execution time is increased with the difficulty of the environment. Indeed, the results demonstrate that the gap, in terms of explored nodes and execution time, between environment 4 and environment 5 is important: smartPATH is 6.15 times faster than Dijkstra in environment 5, but it is 1.26 faster in environment 4. On the other hand, Dijkstra explores 25 times nodes more than smartPATH in environment 5, but it explores 14 times more nodes in environment 4. This confirms that the gap will be too high for larger problems.

Observation 2. The heuritic distance information probability function executed in the beginning of the smartPATH algorithm helps improve the solution quality. We observe in Table 2 and Table 4 that smartPATH generates better solutions than those provided by smartPATH-WHF. For environment 1, 2 and 3 both algorithms find the optimal path but smartPATH-WHF generates lower number of optimal paths in the different runs than the smartPATH algorithm, which demonstrates the efficiency of using the heuristic distance information.

Observation 3. The smartPATH algorithm siginificantly outperforms the classical ACO and GA algorithms in terms



a. Low Complexity Environment *b. Medium Complexity Environment*

c. Medium Complexity Environment

Figure 2. Simulation Environments. The Dashed Line Shows the Path Generated by the smartPATH Algorithm

of solution quality while maintaing reduced execution times. Indeed, looking at Figure 4 and Table 2, we observe that smartPATH provides the optimal paths for all the four environments, in contrast to the Classical ACO and the GA algorithms that both fail to find the optimal solution in medium-scale and large-scale environments. However, for the small-size problem (i.e. environment 1), the quality of solution generated by the three algorithms after 30 runs is the same and optimal, but CACO fails to find the optimal solution in 13 runs, whereas smartPATH and GA find the optimal path all the time. The advantage of smartPATH as compared to GA is that smartPATH finds the optimal path much faster than GA does in all environments, including environment 1, as it is illustrated in Table 4 and Fig 3.

We stress that even though the CACO algorithm exhibits smaller execution times than smartPATH, as depicted in Table 4, does not mean that it outperforms smartPATH because CACO fails to find the optimal solution after the 30 iterations of each of the 30 runs in large environments (i.e. environments 2, 3, and 4). Indeed, as shown in Table 4, 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.

Observation 4. The GA algorithm provides slow convergence speed and is not appropriate for solving the path planning problem in large environments. However, it is very effective when used for post optimization purposes to improve the solution quality. The GA algorithm always exhibits the lowest solution qualities and longest execution times, which is expected as it is commonly known to have slower convergence speed as compared to ACO [37]. For that reason, we only used GA for post optimization purposes to improve on the ACO solution quality. For environment 4, the GA algorithm fails to find a solution to the problem. As the environment is large, the algorithm fails to generate feasible intial population and then fails to find a solution. Indeed, a path is constitued from a set of connected waypoints. We assumed that each waypoint is visited only one time. In case of a large environment, the paths of the initial population generated are not feasible because

Environments	smartPATH	IACO	smartPATH-WHF	CACO	GA	Dijkstra
Environment 1	17.97	17.97	17.97	17.97	17.97	17.97
Environment 2	19.52	19.52	19.52	25.783	44.8521	19.52
Environment 3	35.2231	35.2231	35.2231	37.68	68.4546	35.2231
Environment 4	114.2484	114.2484	142.6610	263.2437	-	114.2484
Environment 5	349.8094	349.8094	349.8094	1255.4591	-	349.8094

Table 2. Length of the Generated Paths of smartPATH, IACO , smartPATH-WHF, CACO , GA and Dijkstra Algorithms in Different Environments (m)

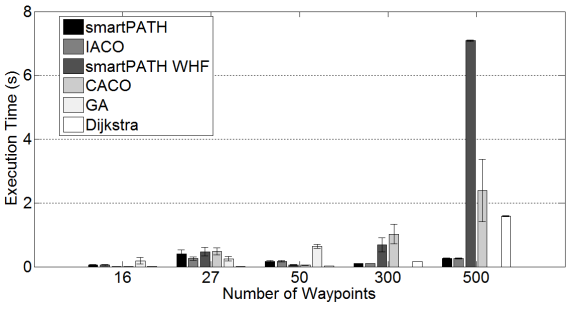


Figure 3. Execution times of smartPATH, IACO, smartPATH-WHF CACO, GA and Dijkstra Algorithms in Different Environments

they are blocked in a deadlock position (a waypoint that hasn't any adjacent waypoints not visited).

Discussion: 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 for dense graphs in comparison with Dijkstra exact method (in the case of environment 4 and environment 5), while it improves the solution quality in comparison with CACO and GA. It is clear that the hyrbidization between ACO and GA brings a lot of benefits as it considers the best features of both approaches, which contributes to improve the solution quality and to reduce the search time for large scale graph environments.

5.2.2. 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 α, the heuristic factor β, the evaporation factor ρ. 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 ρ 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 Figure 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 (less than 10 ants), the smartPATH algorithm generates non optimal solutions in faster times for all the four environments. Using 11 ants in environment 1, smartPATH generates the optimal paths. Wheras, for environment 2, 3 and 4 , a larger number of ants (20 ants) is needed to provide the optimal solution.
- **Impact of variation of the evaporation factor:** All simulations are done for fixed number of ants: 11 ants for environment 1, 20 ants for environment 2, environment 3 and environment 4. The results of simulation are depicted in Figure 7. The simulation results prove that the variation of the evaporation factor ρ has an impact only on the execution times for all the four environments. From Figure 7 it can be noticed that large ρ produces better results in a minimum amount of time for all the four environments.

Environments	smartPATH	Dijkstra
Environment 1	2135	48
Environment 2	6560	140
Environment 3	4228	257
Environment 4	2333	32739
Environment 5	4343	108723

Table 3. Number of explored nodes of smartPATH and Dijkstra Algorithms in Different Environments (s)

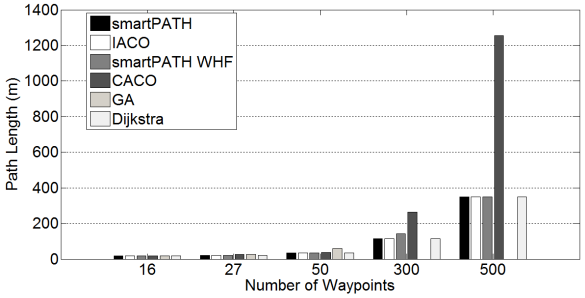


Figure 4. Length of the Generated Paths of smartPATH, IACO, smartPATH-WHF, CACO, GA and Dijkstra Algorithms in Different Environments

Environments	smartPATH	IACO	smartPATH-WHF	CACO	GA	Dijkstra
Environment 1	0.048	0.048	0.00961	0.0083	0.1875	0.0076
Environment 2	0.3978	0.25764	0.471	0.4841	0.25	0.00854
Environment 3	0.1722	0.1722	0.0551	0.04738	0.6406	0.0238
Environment 4	0.1344	0.0992	0.6832	1.0223	-	0.16912
Environment 5	0.2574	0.2574	7.0855	2.38975	-	1.5816

Table 4. Execution times of smartPATH, IACO , smartPATH-WHF, CACO , GA and Dijkstra Algorithms in Different Environments (s)

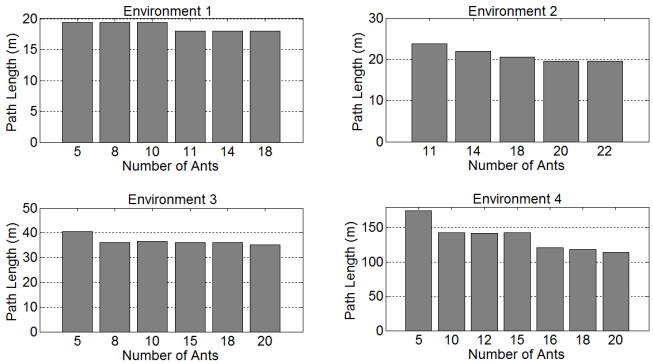


Figure 5. Impact of Variation of the Number of Ants on the Path Length in Different Environments

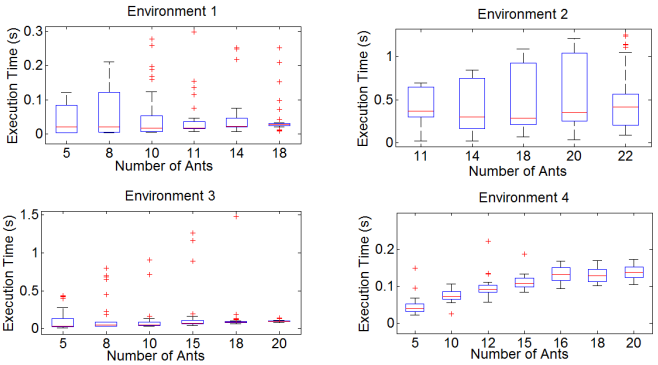


Figure 6. Impact of Variation of the Number of Ants on the Execution Time in Different Environments

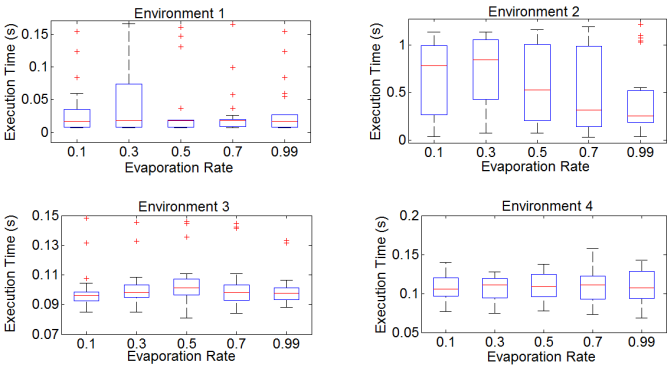


Figure 7. Impact of Variation of the Evaporation Rate ρ on the Execution Time in Different Environments

6. Experimental Study

6.1 Experiments Set-Up

In this section, we present an experimental study to demonstrate the feasibility of the smartPATH algorithm

through a real-world implementation on robots, and also to evaluate its performace in realistic scenarios.

The implementation was developed in C++ using Microsoft Visual Studio and tested on the Wifibot Lab V3 robot [48], operating under a lightweight version of Windows 7. The Wifibot Lab V3 robot is equipped with a computer board encompassing an Intel Atom D 510 Duo Core processor, with 1 GB of RAM and a compact flash of 4 GB as hard drive. The robot also has 4 infrared sensors (2 in front and 2 in back), with a detection range between 20 cm and 190 cm. The navigation of the robot is ensured using (*i.*) its wheel odometry that provides information about the traveled distance and (*ii.*) the VectorNav VN-100 device that provides the orientation (i.e. yaw) of the robot in the 2D plan.

A snapshot of the experimental environment is shown in Figure 8.

We conducted experiments in an indoor environment of $14 * 7m^2$. The obstacles in this environment are static. In addition, the different waypoints are arranged such that the robot navigates without hitting any obstacle. In other words, connected waypoints define obstacle free paths. The apriori known map is uploaded to the robot as a square matrix that represents the input to the path planning algorithms. We considered a home environment to test the performance of the algorithms in realistic conditions. This environment contains 30 waypoints as depicted in Figure 11.

6.2 Scenarios and Metrics

For ensuring a reliable statistical analysis, we considered 30 different scenarios, where each scenario is specified by the coordinates of the start waypoint and the goal waypoint. Each scenario, with specified start/goal waypoints, is repeated 30 times (i.e. 30 runs for each scenario) and average values of the metrics are then calculated with 95% of confidence interval. In total, 900 runs are performed in the performance evaluation study.

We consider three metrics to evaluate the performance of the smartPATH algorithm: (1) the probability to find the optimal path without time bound, and is defined as the ratio of the number of runs where the optimal path is found to the total number of runs, (2) the probability to find the optimal path with bounded time fixed to $100ms$, meaning that the algorithm is stopped at $100ms$ of execution time; this helps assess the effectiveness of the algorithm under real-time constraints and (3) the average gap between the optimal solution and the best solution found by the algorithm, which is evaluated as:

$$GAP = Cost(solutionfound) - Cost(optimalsolution) \quad (7)$$

6.2.1. Results Analysis

We observe in Figure 12 that the probability of finding an optimal solution of the smartPATH algorithm is about 80% when the execution time is not limited. On the other hand, with bounded execution time (100 ms), the probability of finding the optimal path is reduced down to 60%. The results demonstrate the effectiveness of the smartPATH in finding the optimal solution even when real-time constraints are imposed. In fact, smartPATH mostly likely finds the optimal path in less than 100 ms, while still ensuring a small GAP as depicted in Figure 10. Indeed, it is observed in Figure 10 that the GAP does not exceed $1m$ in 98% of cases with no time bound, and in 90% of cases with $100ms$ time bound. These results would be even better with robots having more powerful computing resources.

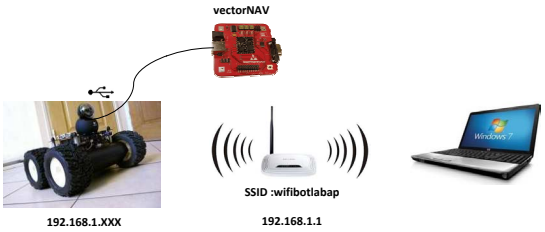


Figure 8. Experiment Set-Up

Figure 9. Real Environment

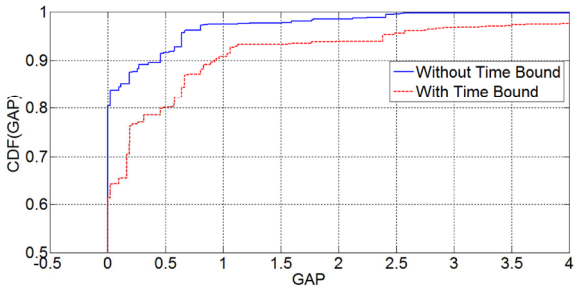


Figure 10. Cumulative Distribution Function of the GAP

Figure 13 presents the scatterplot of the Relative Time versus the Relative GAP and also confirms the good performance of smartPATH. The relative time is defined as the ratio of the time to find the best solution to the maximum time among the 900 experiments. The relative GAP is defined as the ratio of the gap between the optimal solution and the best solution found to the maximum gap. In fact, we observe that the scattered points are concentrated around the point (0,0). This means that most of the non optimal solutions have a small GAP and are found in a short relative time. We note that the maximum time (among the 900 experiments) for finding the best solution was equal to 3.125 sec.

It becomes clear that the hybridization of the ACO approach with the GA approach is quite effective in optimizing the search process of optimal paths for mobile

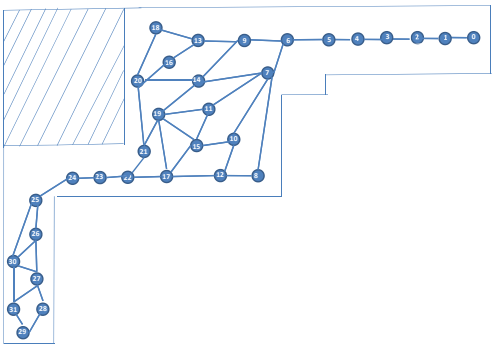


Figure 11. Experimental Environment

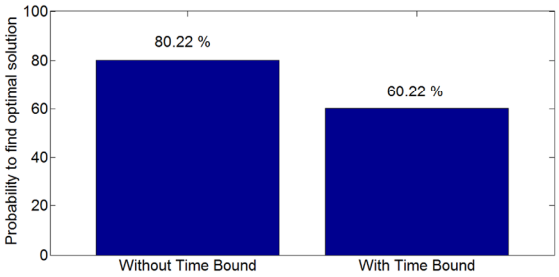


Figure 12. Global Probability to Find the Optimal Path

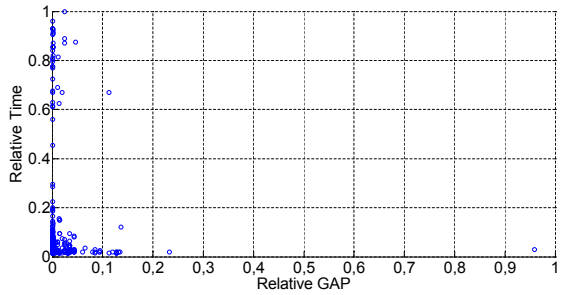


Figure 13. Scatter plot of the Relative Time versus the Relative GAP

robots. However, there is a need to investigate other optimization approaches such as Tabu Search or Particle Swarm Optimization to devise other hybrid algorithms, which is currently under consideration.

7. Conclusion

In this paper, we illustrated how hybridization of optimization algorithms is effective in improving their performance in the context of shortest path search problem, which represents a fundamental problem in robotics. The proposed smartPATH algorithm relies on a new and efficient hybrid ACO-GA approach to solve the problem of the global robot path planning in a static environment. In a nutshell, the smartPATH algorithm looks for optimal solution using an improved version of Ant Colony Optimization approach, then, tries to enhance the solution through a mutation and a modified crossover operator using Genetic Algorithms. The GA phase has the advantage of making diversification by exploring different search spaces thus reducing the risk to fall into a local minimum. smartPATH was extensively evaluated through simulation, and a real implementation on Wifibot Lab robot, and its performance was compared against the classical ACO, the classical GA algorithms and the Dijkstra shortest path method. It has been shown that the modifications applied into 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. The experimental results showed that smartPATH can achieve up to 80% of success in finding the optimal solution.

Future directions. It is clear that the results achieved in this paper demonstrate the real potential of ACO and GA algorithms in solving the path planning problem. However, one open question is: "what about the performance of other approaches in comparison to ACO and GA?". Indeed, a vast array of works in the literature has been performed to devise intelligent heuristics for the robot path planning problem, and different techniques have been used including, but are not limited to, bio-inspired techniques (e.g. swarm particle optimization (PSO), bees algorithms), and evolutionary algorithms (e.g. genetic algorithms), and other local search techniques (e.g. tabu search, simulated annealing). In the literature, there is no comprehensive comparison between existing methods that solve the path planning problem. As such, we are currently investigating the other approaches for solving the path planning problem. For instance, we noticed that the Tabu Search approach was not extensively used to solve this kind of problems in the literature although this technique has been shown to be efficient in solving some complex combinatorial problems such as the multi-Knapsack problem. We are currently working towards understanding the advantages and limitations of this method, and investigate its potential to effectively solve the path planning problem. Also, several other techniques should be assessed including Particle Swarm Optimization, Neural Networks, and a comparison between all these approaches will bring an added value.

8. Acknowledgment

This article is a revised and expanded version of a paper entitled "smartPATH: A hybrid ACO-GA algorithm for robot path planning" presented in the IEEE Congress on Evolutionary Computation (IEEE CEC 2012) that was organized in Brisbane (Australia) in June 2012.

This work is supported by the iroboapp project âĂIJDesign and Analysis of Intelligent Algorithms for Robotic Problems and ApplicationsâĂİ[49] under the grant of the National Plan for Sciences, Technology and Innovation (NPSTI), managed by the Science and Technology Unit of Al-Imam Mohamed bin Saud University and by King AbdulAziz Center for Science and Technology (KACST).

This work is partially supported by Prince Sultan University.

9. References

[1] Wu T, Duan Z.H, Wang J (2010) The design of industry mobile robot based on LL WIN function blocks language and embedded system. In: The 2nd International Conference on Computer Engineering and Technology. 2010 pp. 622–625.

[2] Stopforth R, Holtzhausen S, Bright G, Tlale N.S, Kumile C.M (2008) Robots for Search and Rescue Purposes in Urban and Underwater Environment- a survey and comparison. In: 15th International Conference on Mechatronics and Machine Vision in Practice. 2008 pp. 476–480.

[3] Nagaoka K, Kubota T, Otsuki M, Tanaka S (2009) Robotic Screw Explorer for Lunar Subsurface Investigation: Dynamics Modelling and Experimental Validation. In: International conference on Advanced Robotics. 2009 pp. 1–6.

[4] Iroboapp Research Project. 2014. Available from: http://www.iroboapp.org/ Accessed on 12 Mar 2014.

[5] Raja P, Pugazhenthi S (2012) Optimal path planning of mobile robots: A review. International Journal of Physical Sciences 7:1314–1320.

[6] Buniyamin N, Wan Ngah W, Sariff N, Mohamad Z (2011) A Simple Local Path Planning Algorithm for Autonomous Mobile Robots. International Journal Of Systems Applications, Engineering and Developement, 5:151–159.

[7] Latombe Jean Claude (1991) Robot motion planning, The Springer International Series in Engineering and Computer Science. 651.

[8] Ellips M, Sedighizadeh D (2007) Classic and heuristic approaches in robot motion planning - a chronological review. In: the proceedings of world academy of science, engineering and technology. 23:101–106.

[9] Dijkstra E.W (1959) A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1:269-271.

[10] Hart P, Nilsson N.J, Bertram R (1968) A formal basis for the heuristic determination of minimum cost paths. In: the IEEE Transactions on Systems Science and Cybernetics, 4:pp. 100-107.

[11] Peyer S, Rautenbach D, Vygen J (2009) A generalization of dijkstra's shortest path algorithm with applications to vlsi routing. Journal of Discrete Algorithms. 7:377–390.

[12] Bhattacharya P, Gavrilova M.L (2008) Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path. In: The IEEE Robotics and Automation Magazine. pp. 58–66.

[13] Warren C.W (1989) Global path planning using artificial potential fields. In: The IEEE International Conference on Robotics and Automation: 2012 USA, pp. 316–321.

[14] Lingelbach F (2004) Path planning for mobile manipulation using probabilistic cell decomposition. In: The IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 Stockholm, pp. 2807–2812.

[15] Tang K.S, Man K.F, Kwong S, He Q (1996) *Genetic algorithms and their applications*. In IEEE Signal Processing Magazine, pp. 22–37.

[16] Cao Y, Zhou X, Li S, Zhang F, Wu X, Li A, Sun L (2010) Design of path planning based Cellular Neural Network. In: the 8th World Congress on Intelligent Control and Automation (WCICA), pp.6539-6544.

[17] Chaari I, Koubaa A, Bennaceur H, Ammar A, Trigui S, Tounsi M, Shakshuki E and Youssef H (2014) On the Adequacy of Tabu Search for Global Robot Path Planning Problem in Grid Environments. In: The 5th International Conference on Ambient Systems, Networks and Technologies ANT2014: 2014 Belgium.

[18] Masehian E, Amin-Naseri M.R (2006) A Tabu Search-based Approach for Online Motion Planning. In: IEEE International Conference on Industrial Technology, pp. 2756–2761.

[19] Marco Dorigo, Thomas Stutzle (2004) Ant colony optimization. Cambridge, Massachusetts London, England,The MIT Press.

[20] Eberhart Y, Shi Y (2001) Particle swarm optimization: developments, applications and resources. In: the IEEE Proceedings of the 2001 Congress on Evolutionary Computation, pp. 81–86.

[21] Fan X, Luo X, Yi S, Yang S, Zhang H (2003) Optimal path planning for mobile robot based on intensified ant colony optimization algorithm. In: the IEEE international conference on Robotics, Intelligent Systems and Signal Processing, pp. 131–136.

[22] Nagib G., Gharieb W (2004) Path planning for a mobile robot using genetic algorithms. In : the IEEE International Conference on Electrical, Electronic and Computer Engineering. pp. 185–189.

[23] Zhao J, Zhu L, Liu G, Han Z (2009) A modified genetic algorithm for global path planning of searching robot in mine disasters. In: the IEEE International Conference on Mechatronics and Automation , pp. 4936–4940.

[24] Porta Garcia M.A, Montiel O, Castillo O, Sepulveda R, Melin P (2009) Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. Journal of Applied soft computing, 1102-1110.

[25] Lee J.W, Lee J.J (2010) Novel ant colony optimization algorithm with path crossover and heterogeneous ants for path planning. In: the IEEE International Conference on Industrial Technology, pp. 559–564.

[26] Alajlan M, Koubaa A, Chaari I, Bennaceur H, Ammar A (2013) Global Path Planning for Mobile Robots in Large-Scale Grid Environments using Genetic Algorithms. In: the International Conference on Individual and Collective Behaviors in Robotics ICBR Sousse, Tunisia.

[27] Wang D.S, Yu H.F (2011) Path planning of mobile robot in dynamic environments. In: the IEEE 2nd international conference on intelligent control and information processing, pp. 691–696.

[28] Zhang X, Wu M, Peng J, Jiang F (2009) A Rescue Robot Path Planning Based on Ant Colony Optimization Algorithm. In: the IEEE International Conference on Information Technology and Computer Science, pp. 180–183.

[29] He Y, Zeng Q, Liu J, Xu G, Deng X (2013) Path Planning for Indoor UAV Based on Ant Colony

Optimization. In: the Chinese Control and Decision Conference (CCDC), pp. 2919–2923.
[30] Ganganath N, Cheng C (2013) A 2–Dimensional ACO-based Path Planner for Off-line Robot Path Planning. 2013 In: the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. pp 302-307.
[31] AL-Taharwa I, Sheta A, Al-Weshah M (2008) A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment. Journal of Computer Science. 341–344.
[32] Zhang Y, Zhang L, Zhang X (2008) Mobile Robot Path Planning base on the Hybrid Genetic Algorithm in Unknown Environment. In: the IEEE Eighth International Conference on Intelligent Systems Design and Applications, pp. 661-665.
[33] Shiltagh N.A, Jalal L.D (2013) Path planning of intelligent mobile robot using modified genetic algorithm. International Journal of Soft Computing and Engineering (IJSCE) 3:31–36.
[34] Hussein A, Mostafa H, Badrel-din M, Sultan O, Khamis A (2012) Metaheuristic optimization approach to mobile robot path planning. 2012 In: the International Conference on Engineering and Technology (ICET), pp. 1 –6.
[35] Tewolde G.S, Weihua S (2008) Robot Path Integration in Manufacturing Processes: Genetic Algorithm versus Ant Colony Optimization. In: the IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans. pp. 278–287.
[36] Buniyamin N, Sariff N, Wan-Ngah W.A.J, Mohamad Z (2011) Robot global path planning overview and a variation of ant colony system algorithm. International Journal of Mathematics and Computers In Simulation, 5:9-16.
[37] Sariff N.B, Buniyamin N (2009) Comparative study of genetic algorithm and ant colony optimization algorithm performances for robot path planning in global static environments of different complexities. In: the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA). pp.132–137.
[38] Ma Y.J, Hou W.J (2010) Path planning method based on hierarchical hybrid algorithm. In: the International Conference on Computer, Mechatronics, Control and Electronic Engineering, pp. 74–77.
[39] Li Q, Zhang W, Yin Y.X, Wang Z.L (2006) An improved genetic algorithm for optimal path planning. Journal of Information and Control. 35:444-447.
[40] Xu R, Li Y, Liu H.L Liu P (2008) Hybrid genetic ant colony algorithm for traveling salesman problem. Journal of Computer Applications, 28:2084-2112.
[41] Wang F.Y, Pan F.Q, Zhang L.X, Zou X (2005) Optimal path algorithm of road network with traffic restriction. Journal of Traffic and Transportation Engineering. 5:92-95.
[42] Gao M, Xu J, Tian J (2008) Mobile robot path planning based on improved augment ant colony algorithm. In: the 2th International conference on Genetic and Evolutionary Computing, pp. 273–276.
[43] Geetha S, Chitra G.M, Jayalakshmi V (2011) Multi Objective Mobile Robot Path Planning Based on Hybrid Algorithm. In: the IEEE 3rd International Conference on Electronics Computer Technology (ICECT), pp. 251–255.
[44] Zhou W, Yi Z, Ruimin Y (2008) Mobile Robot Path Planning Based on Genetic Algorithm. Microcomputer Information. 24:187-189.
[45] Garro B, Sossa H, Vzquezi R.A (2008) Evolving ant colony system for optimizing path planning in mobile robots. In: the IEEE Fourth Congress of Electronics, Robotics and Automotive Mechanics, pp. 444–449.
[46] Zickler S, Veloso M (2010) RSS-based relative localization and tethering for moving robots in unknown environments. In the 2010 IEEE International Conference on Robotics and Automation (ICRA). pp.5466–5471.
[47] Corke P (2012) Robotics, Vision and Control. Springer Tracts in advanced robotics, Second Edition, 596.
[48] WIFIBOT Lab. Available from: http://www.wifibot.com/ Accessed on 29 Apr 2013.
[49] iroboapp: Design and analysis of intelligent algorithms for robotic problems and applications. Available from: http://www.iroboapp.org Accessed on 12 Mar 2014.
[50] Dijkstra Algorithm. Available from: http://www.mathworks.com/matlabcentral/fileexchange/14661-dijkstra-very-simple Accessed on 29 Apr 2013.