Multimed Tools Appl
DOI 10.1007/s11042-010-0585-0

# Optimization of route planning and exploration using multi agent system

**Kashif Zafar · Abdul Rauf Baig**



**Abstract** This research presents an optimization technique for route planning and exploration in unknown environments. It employs the hybrid architecture that implements detection, avoidance and planning using autonomous agents with coordination capabilities. When these agents work for a common objective, they require a robust information interchange module for coordination. They cannot achieve the goal when working independently. The coordination module enhances their performance and efficiency. The multi agent systems can be employed for searching items in unknown environments. The searching of unexploded ordinance such as the land mines is an important application where multi agent systems can be best employed. The hybrid architecture incorporates learning real time A* algorithm for route planning and compares it with A* searching algorithm. Learning real time A* shows better results for multi agent environment and proved to be efficient and robust algorithm. A simulated ant agent system is presented for route planning and optimization and proved to be efficient and robust for large and complex environments.

**Keywords** Ant colony optimization · Route planning · Autonomous agents · Multi agent system · Swarm intelligence

## 1 Introduction

The route planning is an important problem in artificial intelligence research and has been addressed over the years and still considered to be a challenging area and requires efficient and robust technique to solve [13, 14, 18, 19]. Planning is a process in which a planner constructs plans that achieve its goals, and then executes them. When the state space is known and accessible, a planner can use the percepts provided by the environment to build

K. Zafar (✉) · A. R. Baig
National University of Computer & Emerging Sciences, Islamabad, Pakistan
e-mail: kashif.zafar@nu.edu.pk

A. R. Baig
e-mail: rauf.baig@nu.edu.pk

Published online: 19 August 2010

a complete and correct model of the current state space of the solution. When a goal is provided to the planner, it can call a suitable planning algorithm to generate a plan of action. This research implements A* and Learning Real Time A* algorithm using hybrid frontier based architecture and presents a simulated ant agent system for route planning and optimization.

Multi agent exploration of an unknown environment is an instance of a class of problems where autonomous agents are employed to explore the environment. An important concern in exploration is how to cover an unknown area efficiently. It can be performed by using a random search; the agent explores around randomly using a random potential field. The statistics indicate that it can cover the entire area after a large period of time. Another reactive method can employ a repulsive field and the agent is repulsed by areas that have been recently visited. It can be performed by utilization of a repulsive field generated by every visited cell in a coarse occupancy grid or for every previous heading. This behavior drives the agent towards the new area when combine with random potential field. Another method based on behavioral approach is to exploit evidential information in the occupancy grid. Many cells on the grid become unknown as the agent explores the environment. The agent moves to the nearest of unknown cells as a goal and attempts to relocate towards it. As it moves to the goal along with avoiding obstacles, new sensor readings will update the occupancy grid. It reduces the amount and location of unknown areas by creating a new goal. The hybrid frontier based architecture Fig. 1 is presented that implements a hybrid of mapping, perception, detection, avoidance and route planning.

Ant colony optimization [4] and particle swarm optimization [4] are two major techniques in the family of swarm intelligence. In ACO, ants are able to find their way to and from food, and the method by which they do it is modeled in a new approach called the ant colony optimization algorithm. It is a population based approach for optimization problems. A number of artificial "ants" construct solutions to the problem at hand by the repeated selection of parts from a predefined set of solution components. The only communication between ants is called stigmergy [3] i.e. indirect communication using pheromone. These ants select components probabilistically biased by heuristic information (a problem specific heuristic measure of a component's utility) and pheromone information, and directly associated with the solution components. To simulate the real-world process by which ants find the shortest path to a food source, electronic ants deposit pheromone on components in proportion to the quality of the solutions that contain them. The physical existence of simulated ant agents is conceived in the form of a simulation that runs on computing devices. To say that they are autonomous computational entities implies that to some extent they have control over their behavior and can act without the intervention of other systems. These ant agents are goal driven and can also execute tasks in order to

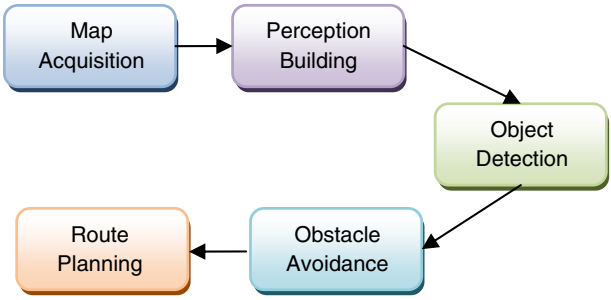



**Fig. 1** Hybrid frontier based architecture

complete design objectives. The ant agents use the modified ant algorithm for finding optimum route between start and the goal.

The following are the major contributions:

- A hybrid architecture is presented for exploration and planning
- A simulated ant agent system is presented for route generation
- Route optimization and repairing technique is introduced
- Exploration and planning in an unknown environment is presented

The Section 2 describes the problem formulation, Section 3 presents the related work, Section 4 gives the survey of coordination strategies, Section 5 presents the hybrid frontier based architecture, Section 6 presents a simulated ant agent system for route planning and optimization, While Section 7 describes the experimental details along with results and finally Section 8 presents the conclusion.

## 2 Problem formulation

The route planning and optimization of routes have been considered as NP complete problem [2]. Non-deterministic polynomial time complete is a complexity class of problems with two properties, i.e. any given solution to the problem can be verified quickly and if the problem can be solved in polynomial time then every problem in NP class can be solved. Such problems are usually tackled by approximation algorithms [2] and meta-heuristic approach is one of them.

An offline planner generates the complete plan before the task is performed [19]. The traditional offline planners often assume that environment is completely known and they try to find the plan based on shortest distance criteria. They cannot handle dynamic environments and are limited to their initial plan. When environment remains constant throughout the planning process, the route planning is called static offline route planning. A* [15], RTA* [7–10] are the algorithms that belong to this family. The planner is provided with the complete picture of the environment along with the starting and destination points. The destination point remains constant throughout the planning process. Static environments are those environments which, once discovered completely, do not change with respect to the environment components. The static environment can be known, partially known [17] or unknown and requires different approach for the each category.

On the other hand, an online planner generates a partial plan during the execution of the task. Online planners are mostly used for dynamic route planning. When the state space changes during the planning phase, the route planning is called dynamic online route planning [19]. Online planning approach is based on the assumption that the agents would be dealing with dynamic environment, and it would neither be feasible nor practical to re-plan the complete route. The online planners are mostly heuristic in nature as the problem instance reveals incrementally and often prone to slow response time. They require sophisticated algorithmic approach to find the shortest route.

There are different approaches for online route planning. Some of the online planners generate route offline initially and then modify it during task execution. Some do planning gradually with task execution. While there are some planners that always hold a backup plan as prediction and when environment changes or previous plan fails, they start executing the backup plan. The unknown and dynamic environments are most difficult to deal. The tracking of moving target by ACO can be considered as dynamic online planning

problem. There are number of constraints to be considered during planning phase. The obstacle avoidance and finding the shortest route for each changing goal are the complex tasks to deal with.

## 3 Related work

Brian Yamauchi [20, 21] presented a Frontier-based exploration technique for the exploration of unknown environments. While many robots can navigate using maps, few are equipped with the capability to map their own maps. A given territory is required to be mapped in advance undergoing a mapping process, providing either the exact locations of obstacles as in the case of metric maps or a graph representing the connectivity between open regions as in the case of topological maps. As a result, most mobile robots become unable to navigate efficiently when placed in unknown environments. Exploration has the potential to enable robots to rule out this limitation. *Exploration* is an act of navigating though an unknown environment while constructing a map that can be used for subsequent navigation. A plausible exploration is one that results in generating a complete or nearly complete map in reasonable amount of time. The central question in exploration is: Given what you know about world, where should you move to gain as much new information as possible? Initially you know nothing except what you can see from where you are standing. The intention is to build a map that describes as much of the world as possible as quickly as possible with maximum information. The exploration process transforms unknown region to a known region by gradually moving towards the unknown regions.

Another method of deciding how to explore space is to have the agent build a reduced Generalized Voronoi Graph (GVG) [1] as it moves through the world. As the agent moves, it attempts to maintain a path that places it equidistant from all objects it senses. Essentially, the agent tries to move ahead but stay in the "middle" or at a tangent to surrounding objects. This path is a GVG edge, the same as would be generated by decomposing the space into a Voronoi graph. Generating and following the path is straight forward to do with a behavior. When the agent comes to a dead, there are multiple GVG edges that the agent could follow. But in this case, the agent can perceive that both of these edges end at objects, so there is no reason to follow them. The agent can then backtrack along the path it had been on, either to the start or to another branch. If the agent encounters branches in the GVG edges, it can choose one at random to follow.

H. Mei et al. [13] used the combination of artificial potential field (APF) [12] with ACO for path planning of a robot in dynamic environment. This paper presents the concept of local and global path planning. The pheromone was used to prevent artificial potential field from getting local minimum and handles the real-time demand. The main drawback of APF is the convergence to local minimum. It integrates local and global planners to cater the dynamic environments. ACO is used for global route planning based on static environment information and then APF is used to program the local route. Instead of using the same level of pheromone field initially, they use different pheromone fields based on the distance of that current point from the obstacles. This reduces the convergence time for the ACO. They also use the information regarding obstacle-avoidance and smoothness of the route. This technique uses the obstacle-avoidance in objective function along with smoothness but lacks the information of dynamic obstacles. It opens the feasibility of collision between robot and moving obstacles. For real-time dynamic environments, this method needs to be improved. It performs better then the evolutionary method.

Another approach in this direction is the use of reinforcement learning along with ACO algorithm for obstacle avoidance and route planning for mobile robots presented by N. A. Vien et al. [18]. This paper presents path planning based on Ant-Q algorithm for static and dynamic obstacle avoidance. It is based on the metaphor of ant colonies. They use three methods for delayed reinforcement updating. Ant-Q algorithm performs better than the genetic algorithm with higher convergence rate. It is inspired from Q-learning and ant behaviors. It requires a good cost function for the moving path along with effective Ant-Q value method. The cost function helps in choosing collision free and shortest routes while updating method helps to find optimum solution with faster convergence. It uses pseudo-random proportional action choice rule as state transition rule along with using three categories for updating, i.e. local updating, mixture updating and global updating for optimum solutions. They compared the results with ant colony system and genetic algorithm. They use limited variable environments in their experiments and require more diverse and complex environments. The paths generated are not so smooth and requires testing in different environment representations.

## 4 Coordination strategies

Frontier-based exploration directs exploring agents on frontiers i.e. known space adjacent to unknown space. Frontier-based exploration can be extended to multiple agents. The obvious advantage of using multi agents is the inherited concurrency, which can greatly reduce the time needed for the exploration mission. Coordination among multi agents is necessary to achieve efficiency in the agent exploration or coverage. The performance of the exploration directly depends on the coordination. By increasing the number of agents without incorporating proper coordination mechanism will not result in improved exploration time. This research explores and incorporates the coordination mechanism for improved performance using hybrid frontier based architecture. The brief description of different coordination techniques are given below. The agents can communicate with each other and shared the maps so that better coordination can be achieved.

Presented in [21], this strategy preaches the idea of multi agents that share perceptual information such as newly explored spaces, maintaining their individual map. Based on their individual maps they independently deliberate about their successive destinations. Employing multi agents enable an individual agent to benefit of information from other comrade agents involved in the joint exploration process. Each agent has its own map that models its knowledge or belief of the environment. Whenever an agent arrives at a new frontier, it sweeps its sensors and constructs a sub map within its sensory footprint that is a map representing its current surroundings. The agents map is updated with this sub map and is broadcast to comrade agent in less time. Presented in [11], Formation based coordination agents methodically sweep the unknown space by advancing in close lines of formation. The formation is broken and restored as necessary whenever new obstacles are encountered. This strategy requires a very tight coordination among all agents.

Presented in [5] this coordination scheme is based on potential fields, proposes a deploying strategy in which a cluster of agents is evenly distributed over the environment by considering repulsive forces: agents repel each other and obstacles repel agents. The agents keep moving until repulsive forces cancel each other. Presented in [16], this technique subsequently applies a well-known unsupervised clustering algorithm (*K*-Means) in order to fairly divide the remaining unknown space into as many disjoint regions as available agents.

## 5 Hybrid frontier based architecture

The hybrid architecture [22, 23] used demonstrates the exploration of an unknown environment by a group of autonomous agents using Frontier-based exploration technique.

The two main actors of the simulation as shown in Fig. 2 are the planner agent and the coordinator agent. The anatomy and capability of the two agents (1) Planning agent and (2) Coordinating agent is given below:

### 5.1 Planning agent anatomy

The Anatomy of an agent is composed of its belief and its behaviors/capabilities.

#### *5.1.1 Planning agent belief*

The belief in an agent is represented as a two dimensional occupancy grid. Each cell of the occupancy grid symbolizes a concept of the environment i.e. free cell, obstacle cell, mine cell and unknown cell. The agent applies its deliberation capability on this belief in order to infer a goal, which it will later pursue.

#### *5.1.2 Planning agent capabilities*

The agent has the following set of capabilities, which enable it to explore a given unknown environment:

1. Sense
2. Mapping
3. Localization
4. Information exchange with peer agents

The agent's sensory mechanism is represented as footprint of eight adjacent cells to its current position. Whenever an agent performs a sensory sweep it gains new information about its environment.

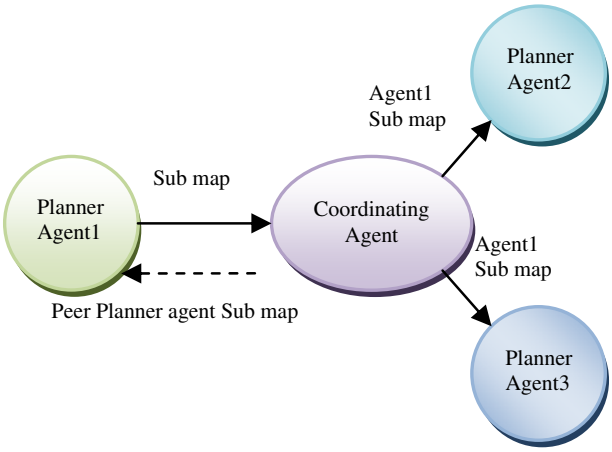



**Fig. 2** Basic coordination architecture

The agent has mapping capability implying the agent updates its current belief i.e. the occupancy grid by fusing information gained about new cells as a resultant of the sensory sweep. In this manner the agent's belief gets subsequently updated.

The agent is capable of localizing itself with respect to its current belief i.e. current mapped territory. Localization implies the re-location of the agent to a new location in order to explore unknown environment. The Localization function is an implementation of a frontier based coverage technique. The Localization process involves the following sub steps to deduce a goal location:

1. Retrieval of frontier cells
2. Calculation of cost to each frontier cell from current location
3. Declaration of least cost frontier cell as goal cell

1. Frontier cells are identified by scanning all visited free cells, a visited free cell is declared a frontier cell if it is adjacent to at least one unknown cell in the horizontal or vertical direction.

2. The cost of each frontier cell is calculated using an A*(A-star) based path planner. The path planner is configured with Manhattan distance as well as the Euclidean distance as the heuristic function. The Euclidean distance is the straight line distance also called the crow-fly distance as shown in Eq. 1.

$$\mathrm{h(n)} = \mathrm{sqrt}\left(\mathrm{abs(n.x - goal.x)}^2 + \mathrm{abs(n.y - goal.y)}^2\right) \tag{1}$$

The Manhattan distance is total number of cells moved horizontally and vertically to reach the target cell from the current cell, ignoring diagonal movement, and ignoring any obstacles as shown in Eq. 2.

$$\mathrm{h(n)} = (\mathrm{abs(n.x - goal.x)} + \mathrm{abs(n.y - goal.y)}) \tag{2}$$

The Frontier cell with the least cost is declared as the next goal state for the agent. The experiments showed that Manhattan distance performs better in grid environment as compared to Euclidean distance.

### 5.2 Coordination via coordinating agent

The planning agents coordinate with each other via the coordinating agent as shown in Fig. 2. Each planning agent after a sensory operation submits its sub map i.e. map generated as a result of a sensory sweep to the coordinating agent. The coordinating agent broadcasts this map to other planning agents, which on receiving this local map update their individual maps.

Each planning agent is required to subscribe with the coordinating agent in order to submit maps and receive maps from peer planning agents, in addition to that exploration is also regulated through the coordinating agent, all agents subscribe to a exploration event in order to simultaneously initiate exploration activity.

#### *5.2.1 Exploration activity sequence with coordination*

Following is the exploration activity sequence with Coordination:

1. Perform Sensory Sweep
2. Build Sub Map
3. Update Local Map with Sub Map

4. Submit Sub map to Coordinating Agent
5. Receive Peer Sub maps from Coordinating Agent
6. Perform Frontier Analysis
7. Select Frontier and Travel
8. Repeat until no frontiers to travel

### 5.3 Broadcast manager

In addition to the planning agents and coordinating agent, a broadcast manager is required. A given agent communicates with the coordinating agent using a broadcast manager. The broadcast manager is an abstraction of an agents networking and communication capabilities. The broadcast manager provide services to publish the presence of an agent, send agent sub map to coordinating agent and receive sub maps of useful information from peer agents.

Using these services of the broadcast manager an agent is able to make other agents aware of its presence via subscribing with the coordinating agent, to submit its sub map to the coordinating agent and receive sub maps of peer agents.

## 6 Simulated ant agent system

Ants are tiny insects capable of finding shortest routes from their nest to the food source. These ants heuristically find the shortest route and save time to get the food. Ant colony optimization [4] is the technique inspired by these tiny ants for optimization problems like traveling salesman problem, job scheduling, network routing, and classification. Ant colony optimization has been applied to solve combinatorial complex problems.

This research presents a simulated ant agent system (SAAS) for solving route planning problem both in static as well as in dynamic environments. SAAS has been applied for online planning in dynamic environments with different environmental configurations and constraints. Ant agents have been used in SAAS and they are capable of traversing the environment based on ACO algorithm technique and collectively find the shortest route from the initial state to the goal state by using indirect communication.

Ant agents are capable of generating probabilistic solutions using a roulette wheel method. They have no prior knowledge of the environment and it traverses the environment for goal state. Once the goal state is discovered, each ant agent constructs a single solution. The solution is evaluated and its information is conveyed to other ant agents by using stigmergy [3]. The stigmergy is a technique of indirect communication used by ant agents. Ant agents update the solution path by updating pheromone level and it acts as a communication for other ant agents. In the initial state space search, ant agents construct bad solutions but as these ant agents reach the goal state, they update the pheromone levels for the solution path. The next ant agent is guided by the indirect communication of this route information and probability of selecting these routes by the ant agent increases. The parameters like alpha and beta controls the convergence process. The alpha parameter is related with pheromone level and beta is related with the heuristic value i.e. Manhattan distance or Euclidean distance. The different ranges for these parameters have been tested and used in the simulation. The pheromone level and heuristic function are two components that guide the ant agents towards the goal state.

The SAAS has been applied for static route planning using modified ACO algorithm. The goal state remains fixed and environment is static and unknown for ant agents. After loading of map with specific environment, the map is prepared for SAAS. The pheromone

is initialized randomly to a small value from 0.01 to 0.09. The start state and the goal state are selected and SAAS starts execution. The ant agent starts exploring the environment randomly and the selection of the next node takes place by using the probability of selection formula given in (1). The formula consists of pheromone level as well as the heuristic function for calculating probabilities for each prospective node. The heuristic function used for static environments is the Manhattan distance [15]. The Manhattan distance is the number of vertical and horizontal moves to reach the goal state. Before taking the step, each ant agent calculates the probability for each adjacent node and selects the next node by using roulette wheel. The ant agent continuously explores the static environment for goal state. As soon as it finds the goal state, the pheromone at the resultant path is updated by using the updating equation.

Finding the solution in dynamic environment is difficult when multiple constraints are enforced. The SAAS has been applied for dynamic environment in the same way as moving target search (MTS) [6, 7, 10]. The simulated ant agent is capable of locating the moving target by using modified ACO. Again ant agent has been used for the handling of moving targets. There are two options for incorporating the moving target. In the first option, we select the number of times we want to change the target during each run. While in the second option the ratio of change of moving target generates randomly. After loading and preparation of the map, an option is selected for incorporating moving target. During the execution of the task, goal changes and ant agent reconfigures the planning phase for the new goal state. SAAS has been tested for different map sizes and configurations.

## 7 Experimentation and results

This research is focused on the issues when agents are used for multiple inter related tasks such as mapping, detection, avoidance and route planning. In the start of the research, the focus was on exploration and then after implementing the mechanism for exploration, the focused was changed to coordination. During this research, two exploration strategies have been experimented and the results from frontier based exploration were more reasonable and accurate as compare to GVG method.

Frontier based method look as the perimeter of the sensed region on an occupancy grid and then rank areas to explore. While Voronoi methods represent the explored world as a Voronoi diagram and use geometric properties to ensure that open areas are opportunistically explored as the agent moves.

The resultant hybrid architecture was tested by using a grid simulation as shown in Fig. 3 with planner agents and coordinating agent. In Fig. 4 multiple autonomous agents are shown during exploration and obstacle avoidance. After the successful exploration of mines, the next objective is to device a safe route for the movement of soldiers. The provision for the start point and the destination point is also given in the simulation. The resultant path will be a safe route after avoiding land mines and other obstacles as shown with green color in Fig. 5. The simulation has been implemented in Microsoft.Net platform using C# language. The experiments have been conducted on Intel Core 2 Duo processor with 2 GB Ram with Windows 7 as the operating system.

### 7.1 Comparison of the heuristic for A* and learning real time A* algorithm

The heuristic can be used to control the behavior of A* algorithm.

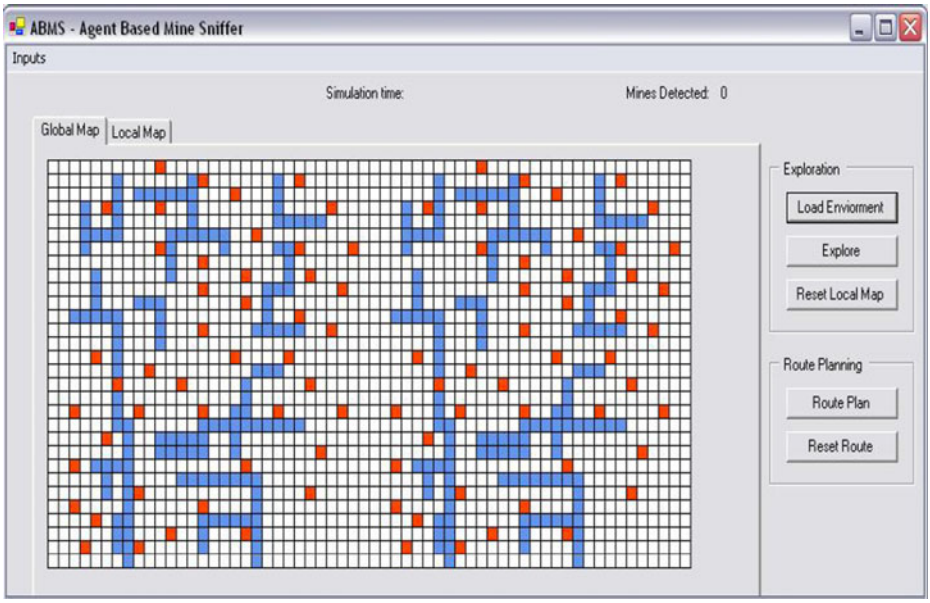



**Fig. 3** Agent based mine sniffer screen shots

At one extreme, if h(n) is 0, then only g(n) plays a role, and A* turns into Dijkstra's algorithm, which is guaranteed to find a shortest path.

- If h(n) is always lower than (or equal to) the cost of moving from n to the goal, then A* is guaranteed to find a shortest path. The lower h(n) is, the more node A* expands, making it slower.

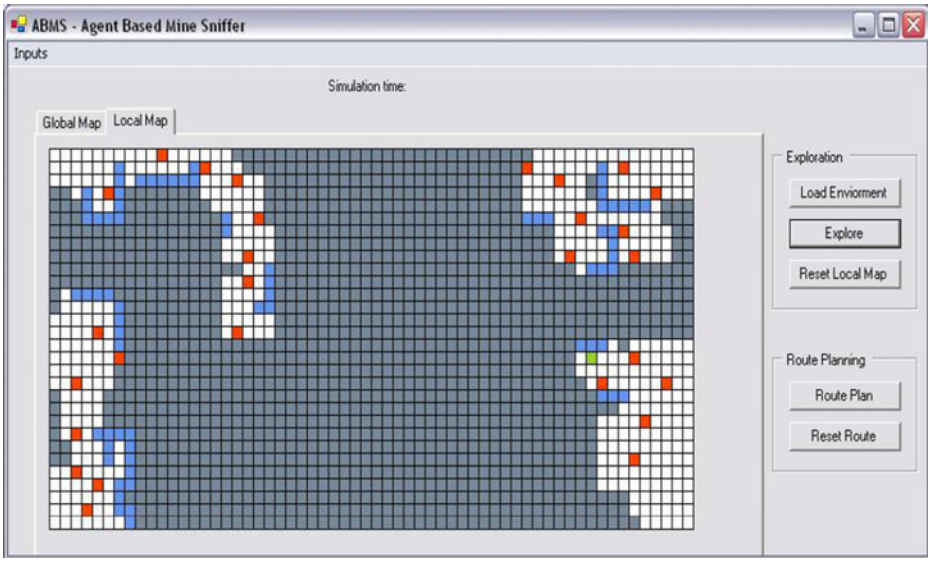



**Fig. 4** Exploration with coordinating multi agents

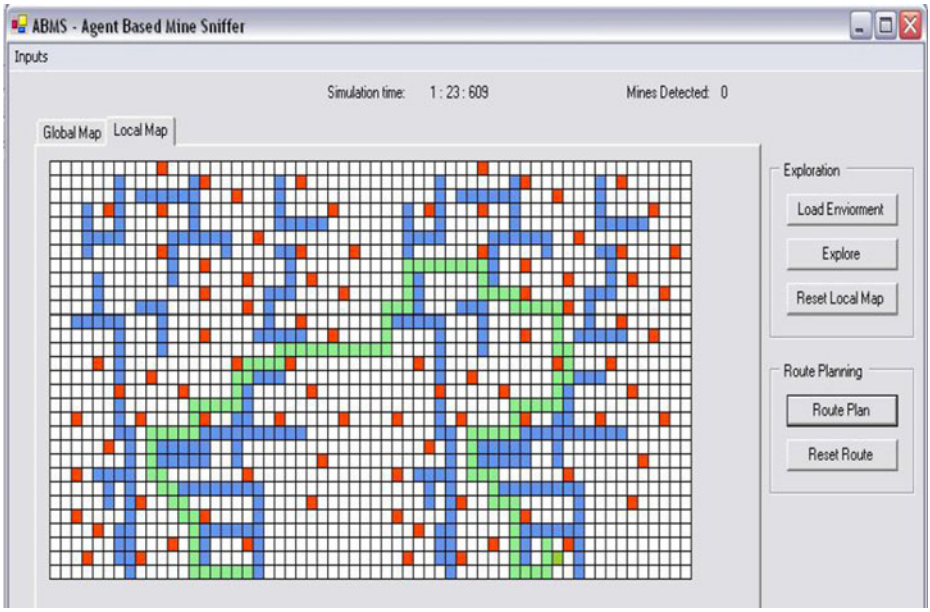



**Fig. 5** Post exploration route planning

- If h(n) is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.
- If h(n) is sometimes greater than the cost of moving from n to the goal, then A* is not guaranteed to find a shortest path, but it can run faster.
- At the other extreme, if h(n) is very high relative to g(n), then only h(n) plays a role, and A* turns into Breath First Search.

A* search is a widely known form of best-first search and it evaluated nodes by combining *g(n),* the cost to reach the node, and *h(n),* the cost to get from the node to the goal:

$$f(n) = g(n) + h(n)$$

Since *g (n)* gives the path cost from the start node to node *n*, and *h (n)* is the estimated cost of the cheapest path from *n* to the goal:

*f (n)* = estimated cost of the cheapest solution through n.

If we try to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of *g (n)+ h (n).* It turns out that this strategy is more than just reasonable, provided that the heuristic function h(n) satisfies certain conditions, A* search is both complete and optimal. The optimality of A* search is straightforward to analyze if it is used with tree-search. A* search is optimal if *h (n)* is an admissible heuristic i.e. provided that *h (n)* never overestimates the cost to reach the goal. The admissible heuristics are by nature optimistic, because they think the cost of solving the problem is less than it actually is. Since *g (n)* is the exact cost to reach *n*, we have as immediate consequence that *f (n)* never overestimates the true cost of a solution through *n.*

Technically, the **A*** algorithm should be called simply **A** if the heuristic is an underestimate of the actual cost. However, we will continue to call it **A*** because the implementation is the same, so we have an interesting situation in that we can decide what we want to get out of A*. At exactly the right point, we will get shortest paths really quickly. If we are too low, then we'll continue to get shortest paths, but it'll slow down. If we're too high, then we give up shortest paths, but A* will run faster. This property of A* can be very useful. For example, you may find that in some situations, you would rather have a "good" path than a "perfect" path. To shift the balance between g(n) and h(n), you can modify either one. A* computes f(n)=g(n)+h(n). To add two values, those two values need to be at the same scale. If g(n) is measured in hours and h(n) is measured in meters, then A* is going to consider g or h too much or too little, and you either won't get as good paths or you A* will run slower than it could. The heuristics for grid maps can be Manhattan distance, Diagonal distance, or a Euclidean distance. This research tested Manhattan as well as Euclidean distance as a heuristic.

In the experiments, three different maps were used with different number of mines and obstacles. The results showed that the performance of LRTA* is better as compared to A* as far as time is concerned. LRTA* uses less time to detect the mine and for searching operation as shown in the figures below. In Fig. 6 map1 was used with 45 mines and 82 obstacles. The detection time with LRTA* was better and efficient.

In Fig. 7 map2 was used with 50 mines and 110 obstacles and again LRTA* showed better results in terms of time and efficiency. Similarly, map3 was used with 40 mines and 134 obstacles and similar results were obtained as shown in Fig. 8. Learning Real Time A* algorithm is more robust, efficient and scalable as compared to Frontier-Based exploration using A*.

In Fig. 9, we see that A* explores the whole map, while traveling a constant distance, while the LRTA* explores the same map while traveling variable distances. This is because the former uses no intelligent information about the presence of mines, and scans the whole map in a linear fashion. On the other hand, the latter technique, while using a probabilistic approach in mine detection, traverses the area to and fro, and at times moves over the same place more than once. This is the reason why LRTA* travels more distance even in a constant map. But this does not mean that LRTA* does not outperforms A* Exploration. The graphs for A* shows only two points because it traverses the map in linear fashion and gives constant distance.

If we look at the same problem with a different angle, we come to find out the advantage of using LRTA* over A* Exploration Technique. The Fig. 10 shows us that keeping the area and the total number of mines constant, if we run both the algorithms on the same map, we see that LRTA* performs better than A*, because it is better able to predict where mine clusters would

**Fig. 6** Comparison of frontier-based A* with LRTA* for map 1

| Comparison of Frontier-Based A* with LRTA* | | | |
|---|---|---|---|
| **Frontier-Based using A* Search** | | Learning Real Time A* Search | |
| **Map1** | | | |
| **Time** | Distance | Time | Distance |
| **1:31** | 437 | 1:20 | 606 |
| **1:34** | 439 | 1:22 | 668 |
| **1:35** | 438 | 1:18 | 610 |
| **1:33** | 439 | 1:22 | 645 |
| **1:33** | 438 | 1:20 | 588 |

**Fig. 7** Comparison of frontier-based A* with LRTA* for map 2

| Comparison of Frontier-Based A* with LRTA* | | | |
|---|---|---|---|
| **Frontier-Based using A* Search** | | Learning Real Time A* Search | |
| **Map2** | | | |
| **Time** | Distance | Time | Distance |
| **1:36** | 485 | 1:10 | 615 |
| **1:36** | 504 | 1:20 | 645 |
| **1:37** | 504 | 1:22 | 618 |
| **1:42** | 487 | 1:21 | 625 |
| **1:36** | 486 | 1:23 | 686 |

be and would finish up finding the mines quicker than the naïve A* exploration algorithm. The distance in A* algorithm again remains constant for all the runs and only two points are taken on the graph, but consumes more time as compared to LRTA* algorithm shown in Fig. 10.

The experimentation has been divided into two phases, i.e. static and dynamic environment. For static environment, goal state remains static. We use four different size environments, i.e. 20×20, 40×40, 60×60, and 80×80 grid maps as shown in Fig. 11. Each grid map has different percentage of obstacles and mines. The starting and the ending points have been fixed for experiments. The number of ants is also fixed for each experiment. After loading of the map, we need to set the parameters for experiment. There is an option for enabling dynamic environment by randomly changing the goal during online planning phase. We provide the number of such changing states for each run of the simulation and goal changes randomly during planning. After loading the map, we need to enter number of iterations, number of ants, and randomly changing goals for incorporating dynamism in experimentation. The first experiment deals with static environment and we use 50 iterations, 1 ant and 30 experimental runs. The results are shown in Tables 1, 2, 3 and 4. Four different maps have been used and each with different obstacle ratio and grid size. A high obstacle ratio and randomly generated maps have been used to simulate complex environments. The main parameters for the simulation are alpha and beta value that primarily controls the convergence of the optimum solution. The alpha value represents the contribution of the pheromone level and beta value represents the contribution of heuristic function.

**Fig. 8** Comparison of frontier-based A* with LRTA* for map 3

| Comparison of Frontier-Based A* with LRTA* | | | |
|---|---|---|---|
| **Frontier-Based using A* Search** | | Learning Real Time A* Search | |
| **Map3** | | | |
| **Time** | Distance | Time | Distance |
| **1:35** | 491 | 1:09 | 598 |
| **1:33** | 484 | 1:19 | 593 |
| **1:35** | 491 | 1:20 | 605 |
| **1:35** | 492 | 1:23 | 599 |
| **1:34** | 510 | 1:21 | 585 |

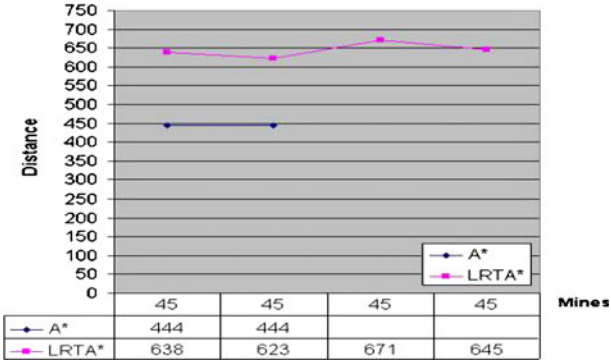



**Fig. 9** A* and LRTA*, distance vs. mines

The Table 1 represents a 20×20 map with static environment configurations. We keep the value of alpha constant and vary the value of beta for 50 iterations and 30 runs for the experiment. The convergence of the optimum solution has been noted. As alpha value represents the contribution of the pheromone and beta value represents the contribution of the heuristic function. The best values are obtained by increasing the contribution of the heuristic function. Similarly, for other maps of 40×40, 60×60 and 80×80, the best values are obtained by increasing the value of beta. The contribution of the heuristic function clearly helps the algorithm to converge to an optimum solution and gives the minimum cell count. Four values have been noted for each run i.e. minimum value, median value, average value, and maximum value. The minimum and average values are the better scale for the comparison and proved that the contribution of the heuristic function plays an important role in finding the optimum solution. The values of alpha and beta are varied from 0.1 to 1.0 for better evaluation and comparison.

The SAAS has been tested by using Euclidean distance and results have been compared with the results from Manhattan distance. We use different values for alpha and beta for each of the maps. The values for alpha and beta have been varied from 0.1 to 1.0 for each experiment and results have been recorded for comparison. There is a module for optimization of the path by repairing the v-edges of the path. After the generation of the path, the optimization module measure the distance of the generated route and compare the distance after repairing the path. If the distance remains same as of the generated path, then system regenerates the new optimized path.

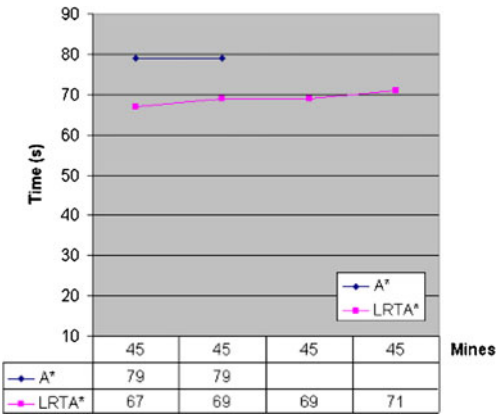
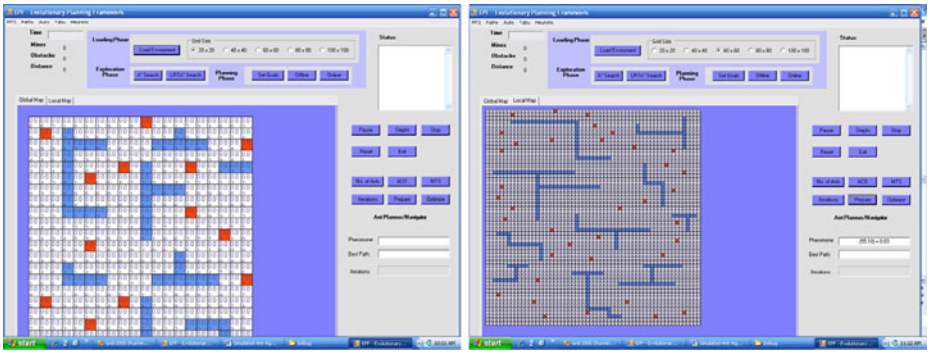
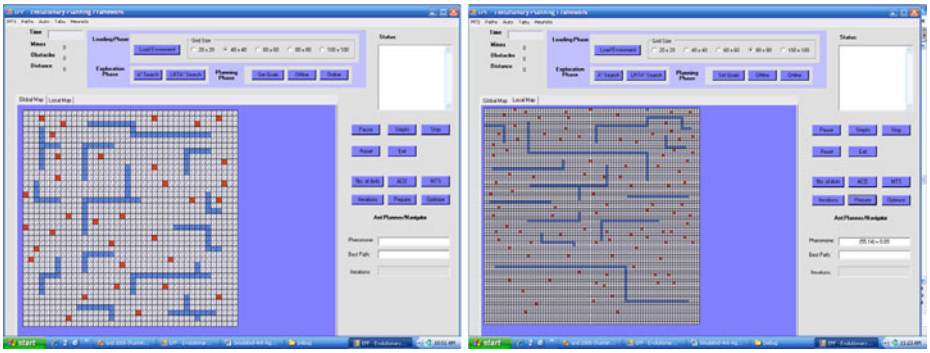



**Fig. 10** A* and LRTA*, time vs. mines



(1) 20x20 (3) 60x60



(2) 40x40 (4) 80x80

**Fig. 11** Grid maps of different sizes, obstacle ratio, and complexity

**Table 1** Map 20×20 results with different values for alpha and beta using 50 iterations and 30 runs for each experiment

| | Beta | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
| Alpha | | | | | | | | | | |
| 0.1 | 31.33 | 30.77 | 29.90 | 30.23 | 29.23 | 30.03 | 28.17 | 28.30 | 28.00 | 27.53 |
| 0.2 | 31.60 | 31.07 | 30.87 | 30.43 | 30.13 | 29.70 | 28.90 | 28.40 | 27.97 | 27.50 |
| 0.3 | 31.73 | 32.17 | 29.43 | 30.73 | 29.67 | 31.07 | 29.47 | 30.43 | 29.53 | 29.83 |
| 0.4 | 34.23 | 32.77 | 32.33 | 31.23 | 32.33 | 32.53 | 29.87 | 30.67 | 30.30 | 30.13 |
| 0.5 | 32.67 | 36.10 | 33.13 | 32.03 | 34.13 | 31.97 | 32.53 | 31.37 | 32.10 | 31.40 |
| 0.6 | 35.77 | 36.10 | 36.70 | 34.40 | 33.57 | 34.00 | 34.93 | 33.23 | 32.73 | 32.53 |
| 0.7 | 36.73 | 37.07 | 33.70 | 35.50 | 36.43 | 35.33 | 34.27 | 34.20 | 34.60 | 31.90 |
| 0.8 | 38.73 | 36.50 | 36.83 | 37.80 | 36.70 | 34.57 | 35.43 | 33.67 | 34.67 | 36.47 |
| 0.9 | 39.10 | 38.97 | 38.40 | 39.17 | 35.53 | 37.17 | 35.23 | 34.80 | 35.00 | 34.80 |
| 1 | 40.10 | 38.47 | 36.93 | 40.50 | 36.53 | 37.50 | 37.43 | 36.90 | 36.10 | 36.90 |
| Min | 31.33 | 30.77 | 29.43 | 30.23 | 29.23 | 29.70 | 28.17 | 28.30 | 27.97 | 27.50 |
| Median | 35.00 | 36.10 | 33.42 | 33.22 | 33.85 | 33.27 | 33.40 | 32.30 | 32.42 | 31.65 |
| Average | 35.20 | 35.00 | 33.82 | 34.20 | 33.43 | 33.39 | 32.62 | 32.20 | 32.10 | 31.90 |
| Max | 40.10 | 38.97 | 38.40 | 40.50 | 36.70 | 37.50 | 37.43 | 36.90 | 36.10 | 36.90 |

**Table 2** Map 40×40 results with different values for alpha and beta using 50 iterations and 30 runs for each experiment

| | Beta | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
| Alpha | | | | | | | | | | |
| 0.1 | 60.37 | 58.38 | 60.20 | 56.80 | 55.73 | 56.70 | 55.43 | 56.03 | 57.23 | 54.57 |
| 0.2 | 65.27 | 62.87 | 63.57 | 64.29 | 63.78 | 62.57 | 64.23 | 62.67 | 61.90 | 61.83 |
| 0.3 | 73.67 | 69.13 | 70.27 | 67.20 | 73.53 | 68.00 | 63.67 | 67.70 | 64.13 | 63.37 |
| 0.4 | 75.50 | 70.80 | 73.47 | 74.80 | 70.50 | 71.83 | 70.84 | 74.17 | 69.37 | 70.33 |
| 0.5 | 79.93 | 74.10 | 73.27 | 75.83 | 69.54 | 73.22 | 71.60 | 74.72 | 72.39 | 71.66 |
| 0.6 | 84.37 | 83.60 | 79.36 | 81.42 | 78.34 | 75.63 | 76.82 | 75.23 | 73.40 | 74.33 |
| 0.7 | 80.43 | 82.45 | 78.49 | 75.24 | 77.49 | 73.67 | 75.03 | 73.67 | 76.28 | 74.35 |
| 0.8 | 80.13 | 84.56 | 79.30 | 81.68 | 75.39 | 78.27 | 75.35 | 76.39 | 72.78 | 73.56 |
| 0.9 | 92.97 | 93.44 | 86.40 | 88.32 | 84.90 | 87.31 | 83.22 | 85.78 | 82.92 | 80.31 |
| 1 | 96.37 | 96.88 | 94.54 | 93.76 | 94.03 | 88.32 | 87.21 | 85.72 | 84.29 | 84.56 |
| Min | 60.37 | 58.37 | 60.20 | 56.80 | 55.73 | 56.70 | 55.43 | 56.03 | 57.23 | 54.57 |
| Median | 80.03 | 78.28 | 75.98 | 75.54 | 74.46 | 73.45 | 73.32 | 74.45 | 72.59 | 72.61 |
| Average | 78.90 | 77.62 | 75.89 | 75.93 | 74.32 | 73.55 | 72.34 | 73.21 | 71.47 | 70.89 |
| Max | 96.37 | 96.88 | 94.54 | 93.76 | 94.03 | 88.32 | 87.21 | 85.78 | 84.29 | 84.56 |

**Table 3** Map 60×60 results with different values for alpha and beta using 50 iterations and 30 runs for each experiment

| | Beta | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
| Alpha | | | | | | | | | | |
| 0.1 | 275.50 | 272.17 | 270.63 | 276.93 | 254.17 | 247.50 | 240.40 | 207.43 | 194.67 | 196.93 |
| 0.2 | 286.45 | 292.49 | 274.56 | 271.50 | 264.89 | 266.35 | 258.64 | 235.45 | 210.58 | 226.46 |
| 0.3 | 289.56 | 282.60 | 246.31 | 281.43 | 274.54 | 267.42 | 264.88 | 225.54 | 233.50 | 215.63 |
| 0.4 | 284.53 | 294.50 | 274.68 | 271.40 | 265.66 | 273.53 | 258.72 | 235.50 | 188.62 | 187.36 |
| 0.5 | 262.50 | 276.57 | 271.89 | 265.43 | 257.89 | 244.65 | 239.65 | 264.77 | 231.44 | 216.45 |
| 0.6 | 283.67 | 275.54 | 271.69 | 254.60 | 255.33 | 271.30 | 235.50 | 243.61 | 185.62 | 192.50 |
| 0.7 | 296.67 | 293.65 | 277.49 | 271.33 | 266.40 | 246.76 | 275.42 | 247.68 | 231.54 | 223.43 |
| 0.8 | 276.50 | 272.54 | 284.22 | 256.43 | 254.51 | 261.60 | 253.87 | 262.26 | 266.49 | 224.48 |
| 0.9 | 354.65 | 287.56 | 293.65 | 277.54 | 271.64 | 254.50 | 243.42 | 216.68 | 225.53 | 218.49 |
| 1 | 342.43 | 273.52 | 284.99 | 267.72 | 289.56 | 264.66 | 241.55 | 224.58 | 223.47 | 205.65 |
| Min | 262.50 | 272.17 | 246.31 | 254.60 | 254.17 | 244.65 | 235.50 | 207.43 | 185.62 | 187.36 |
| Median | 285.49 | 279.59 | 274.62 | 271.37 | 265.28 | 263.13 | 248.65 | 235.48 | 224.50 | 216.04 |
| Average | 295.25 | 282.11 | 275.01 | 269.43 | 265.46 | 259.83 | 251.21 | 236.35 | 219.15 | 210.74 |
| Max | 354.65 | 294.50 | 293.65 | 281.43 | 289.56 | 273.53 | 275.42 | 264.77 | 266.49 | 226.45 |

**Table 4** Map 80×80 results with different values for alpha and beta using 50 iterations and 30 runs for each experiment

| | Beta | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
| Alpha | | | | | | | | | | |
| 0.1 | 784.78 | 793.44 | 770.54 | 673.80 | 634.58 | 567.45 | 523.46 | 454.34 | 468.67 | 424.56 |
| 0.2 | 791.56 | 775.54 | 792.73 | 684.50 | 671.42 | 584.90 | 573.55 | 476.32 | 471.24 | 445.52 |
| 0.3 | 824.43 | 785.45 | 789.91 | 754.34 | 678.23 | 673.55 | 592.45 | 510.56 | 492.25 | 451.55 |
| 0.4 | 854.66 | 845.50 | 823.75 | 776.44 | 754.67 | 653.79 | 588.34 | 597.49 | 556.38 | 531.34 |
| 0.5 | 899.45 | 863.60 | 834.62 | 816.69 | 785.57 | 680.56 | 618.72 | 566.90 | 598.33 | 578.22 |
| 0.6 | 923.66 | 860.64 | 845.32 | 743.38 | 668.23 | 677.33 | 599.45 | 589.92 | 566.34 | 468.35 |
| 0.7 | 945.50 | 913.45 | 865.53 | 823.39 | 780.50 | 734.44 | 623.33 | 646.66 | 543.22 | 528.60 |
| 0.8 | 993.56 | 956.62 | 895.65 | 878.71 | 762.56 | 743.59 | 778.83 | 683.55 | 668.45 | 642.22 |
| 0.9 | 985.54 | 938.55 | 917.45 | 889.90 | 836.88 | 734.67 | 723.54 | 688.45 | 690.23 | 641.14 |
| 1 | 989.54 | 945.32 | 964.34 | 966.76 | 983.78 | 873.88 | 776.33 | 665.54 | 680.42 | 651.54 |
| Min | 784.78 | 775.54 | 770.54 | 673.80 | 634.58 | 567.71 | 523.45 | 454.34 | 468.67 | 424.56 |
| Median | 911.56 | 862.12 | 839.97 | 796.57 | 758.62 | 678.95 | 609.09 | 593.71 | 561.36 | 529.97 |
| Average | 899.27 | 867.81 | 849.98 | 800.79 | 755.64 | 692.44 | 639.80 | 587.97 | 573.55 | 536.31 |
| Max | 993.56 | 956.62 | 964.34 | 966.76 | 983.78 | 873.88 | 778.83 | 688.45 | 690.23 | 651.54 |

The simulated ant agent system is able to track the moving target similar to the online planning and takes care of the dynamic environment. During each run, the goal changes randomly and it behaves like a moving target search. Whenever the goal changes the SAAS is capable of tracking it and again finds the shortest route to the new goal. The experimental setup provides two options for changing the goal. The option 1 takes a number to change the goal during the run randomly. While option 2 changes the goal according to a certain percentage of change. The experiments have been performed by changing the goal 3 times and by using the option 1 first. The number of generated paths automatically generated in a drop down menu list. If the user selects to change the goal 4 times, it will have 4 paths and we can draw and optimize any of the 4 available paths. The figures below show different map considerations used in the experiments.

The experiments have been conducted separately for static goal and for dynamic goal environment. The starting point and the goal point remains constant throughout each run. Each run consists of 50 iterations with single ant. The maps have been generated randomly with different obstacle ratios and placement of mines. The environments are from simple to complex. The SAAS has been used to find the optimum route between the starting point and the goal point.

**Table 5** Comparison of SAAS with LRTA* for an average of 30 runs

| Map size | Alpha | Beta | SAAS | LRTA* |
|---|---|---|---|---|
| 20×20 | 0.2 | 1 | 27.52 | 26.25 |
| 40×40 | 0.1 | 1 | 54.57 | 66.78 |
| 60×60 | 0.6 | 0.9 | 185.62 | 212.32 |
| 80×80 | 0.1 | 1 | 424.56 | 523.35 |

**Table 6** Comparison of SAAS with GA and PSO based optimization techniques

| Map size | Alpha | Beta | SAAS | LRTA* | GA | PSO |
|---|---|---|---|---|---|---|
| 20×20 | 0.2 | 1 | 27.52 | 26.25 | 28.57 | 27.35 |
| 40×40 | 0.1 | 1 | 54.57 | 66.78 | 62.68 | 58.84 |
| 60×60 | 0.6 | 0.9 | 185.62 | 212.32 | 341.29 | 193.50 |
| 80×80 | 0.1 | 1 | 424.56 | 523.35 | 672.45 | 492.62 |

**Table 7** Comparison of simulated ant agent system for moving target search with LRTA* algorithm

| Map = 20×20 | Moving target search using simulated ant agent system (alpha = 0.1 & beta = 0.9) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Iterations = 200 | | | | | | | | | | |
| Case | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Initial State | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) |
| Goal 1 | (3,11) | (0,18) | (1,15) | (1,14) | (0,18) | (3,11) | (2,15) | (6,18) | (7,16) | (6,19) |
| Manhattan Distance | 14 | 18 | 16 | 15 | 18 | 14 | 17 | 24 | 23 | 25 |
| Cells traversed-SAAS | 11 | 18 | 15 | 14 | 18 | 11 | 15 | 18 | 16 | 19 |
| Cells traversed-LRTA* | 11 | 18 | 15 | 14 | 18 | 11 | 15 | 18 | 16 | 19 |
| No. of iterations | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| Goal 2 random | (2,10) | (3,16) | (2,17) | (4,13) | (1,17) | (2,10) | (4,13) | (5,16) | (5,18) | (3,16) |
| Distance from Start | 12 | 19 | 19 | 17 | 18 | 12 | 17 | 21 | 23 | 19 |
| Distance from Goal 1 | 2 | 5 | 3 | 4 | 2 | 2 | 4 | 3 | 4 | 6 |
| Cells traversed-SAAS | 10 | 16 | 17 | 13 | 17 | 10 | 13 | 17 | 18 | 16 |
| Cells traversed-LRTA* | 10 | 16 | 17 | 13 | 17 | 10 | 13 | 17 | 18 | 16 |
| No. of iterations | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| Goal 3 random | (1,7) | (1,14) | (4,15) | (2,10) | (2,18) | (1,7) | (3,11) | (8,13) | (8,16) | (1,14) |
| Distance from Start | 8 | 15 | 19 | 12 | 20 | 8 | 14 | 21 | 24 | 15 |
| Distance from Goal 2 | 4 | 4 | 4 | 5 | 2 | 4 | 3 | 6 | 5 | 4 |
| Cells traversed-SAAS | 7 | 14 | 20 | 10 | 18 | 7 | 11 | 13 | 16 | 14 |
| Cells traversed-LRTA* | 7 | 14 | 16 | 10 | 16 | 7 | 11 | 13 | 16 | 14 |
| No. of iterations | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| Goal 4 random | (3,4) | (3,11) | (1,12) | (1,9) | (4,17) | (3,4) | (2,9) | (5,11) | (7,18) | (3,12) |
| Distance from Start | 7 | 14 | 13 | 10 | 21 | 7 | 11 | 16 | 25 | 15 |
| Distance from Goal 3 | 5 | 5 | 6 | 2 | 3 | 5 | 3 | 5 | 3 | 4 |
| Cells traversed-SAAS | 4 | 11 | 12 | 9 | 18 | 4 | 9 | 11 | 19 | 12 |
| Cells traversed-LRTA* | 4 | 11 | 12 | 9 | 17 | 4 | 9 | 11 | 19 | 12 |
| No. of iterations | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| Goal 5 random | (6,1) | (6,8) | (2,9) | (2,7) | (5,15) | (6,1) | (1,8) | (3,9) | (6,15) | (6,11) |
| Distance from Start | 7 | 14 | 11 | 9 | 20 | 7 | 9 | 12 | 21 | 17 |
| Distance from Goal 4 | 6 | 6 | 4 | 3 | 3 | 6 | 2 | 4 | 4 | 4 |
| Cells traversed-SAAS | 6 | 10 | 9 | 7 | 17 | 6 | 8 | 9 | 15 | 11 |
| Cells traversed-LRTA* | 6 | 10 | 9 | 7 | 17 | 6 | 8 | 9 | 15 | 11 |
| No. of iterations | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |

The probability of selection for next node depends on the values of alpha and beta. The alpha value represents the contribution of pheromone level and the value of beta represents the contribution of heuristic function. The higher the value of beta, the more is the influence of the distance in the selection of the next node. The figure below shows the effect of changing the values for alpha and beta on the convergence of SAAS. The paths converge to the optimum values with increasing beta value. The experiments have been conducted for $20\times20$, $40\times40$, $60\times60$, and $80\times80$ map sizes and the obtained results are consistent in all the maps. The performance of SAAS remains consistent with the increase in map size. The performance of SAAS is better as compares to LRTA* for complex environments as shown in Table 5. The SAAS has also been compared with genetic algorithm (GA) and PSO based optimization techniques and showed better results as shown in Table 6. For $20\times20$ map, LRTA* performed better but as the environment size and complexity increases, SAAS out performs LRTA* and showed better results. The SAAS and LRTA* are further tested for moving target search and showed comparable results as shown in Table 7.

## 8 Conclusion

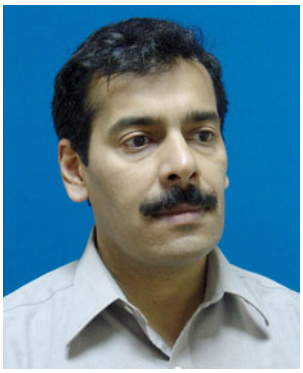
This paper presents an extension of hybrid architecture for route planning and optimization using multi agent system. The architecture is tested by using frontier based exploration, A* based exploration and LRTA* algorithm. As a result, the agent is able to construct its own map and explore with respect to it, consequently explores the entire area. After the successful implementation of the coordination mechanism using multi agents, research covers post exploration route planning, which will enable the agent to generate safe route plans for the given target destinations. This research will conclude by providing a statistical comparison of results i.e. exploration time for different environment configurations i.e. varying maps and number of agents. The frontier based exploration technique used is not the optimum technique for exploration and leaves room for further improvement. So Learning Real Time A* technique was also used and compared with the frontier based and found to be more efficient and robust for multi agent based mine detection problems. The simulated ant agent system is presented as a route optimization technique based on swarm intelligence. The routes generated by simulated ant agent system are compared with LRTA* and found to be efficient and optimized.

## References


1. Ahn S, Doh NL, Lee K, Chung WK (2003) Incremental and robust construction of Generalized Voronoi Graph (GVG) for mobile guide robot. Proceedings of Intelligent Robots and Systems, IROS
2. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms. MIT, Cambridge, ISBN-81-203-2141-3
3. Dorigo M, Socha K (2007) An introduction to ant colony optimization. IRIDIA, Bruxelles, technical report series
4. Engelbrecht AP (2002) Computational intelligence; an introduction. John Wiley & Sons, ISBN 0-470-84870-7
5. Howard A, Mataric MJ, Skhatme GS (2002) Mobile sensor network deployment using potential fields: a distributed, scalable solution to the Area coverage problem. Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems, pp 299–308
6. Ishida T, Korf RE (1991) Moving target search. Proceedings of the IJCAI, pp 204–210
7. Ishida T, Korf R (1995) Moving target search: a real-time search for changing goals. Proceedings of IEEE PAMI 17(6):609–619
8. Korf RE (1987) Real-time heuristic search: first results. Proceedings of the AAAI, pp 133–138

9. Korf RE (1988) Real-time heuristic search: new results. Proceedings of the AAAI, pp 139–144
10. Korf R (1990) Real-time heuristic search. Artif Intell 42(2–3):189–211
11. Latimer D et al (2002) Towards sensor based coverage with robot teams. Proceedings of the International Conference on Robotics and Automation, IEEE, pp 961–967
12. Li Z, Chen X (2004) A new motion planning approach based on artificial potential field in unknown Environment. PDCAT 2004, LNCS, Springer-Verlag, pp 376–382
13. Mei H, Tian Y, Zu L (2006) A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. Int J Inf Technol 12(3):78–88
14. Qiang Z, Xing Z (2006) Global path planning approach based on ant colony optimization algorithm. LNCS, Springer-Verlag
15. Russell SJ, Norvig P (1995) Artificial intelligence-A modern approach. Prentice Hall, Englewood Cliffs, ISBN 0-13-103805-2
16. Solanas A, Angel M (2004) Coordinated multi-robot exploration through unsupervised clustering of unknown space. Proceedings of the International Conference in Intelligent Robots and Systems, IEEE-IROS, pp 717–721
17. Stentz A (1994) Optimal and efficient path planning for partially-known environments. Proceedings of the International Conference on Robotics and Automation, IEEE, May
18. Vien NA, Viet NH (2007) Obstacle avoidance path planning for mobile robot based on Ant-Q-reinforcement learning algorithm. ISNN, LNCS, Springer-Verlag
19. Xiao J, Michalewicz Z, Zhang L, Trojanowski K (1997) Adaptive evolutionary planner/navigator for mobile robots. IEEE Trans Evol Comput 1(1):18–28
20. Yamauchi B (1997) A frontier based approach for autonomous exploration. Proceedings of the International Symposium on Computational Intelligence on Robotics and Automation, IEEE
21. Yamauchi B (1998) Frontier based exploration using multi robots. Proceedings of the 2nd International Conference on Autonomous Agents, ACM
22. Zafar K, Baig AR (2006) Mine detection and route planning in military warfare using multi agent system. Proceedings of the International Conference on Computer Software and Applications, IEEE-ESAS, pp 327–332
23. Zafar K, Baig AR (2009) Multi agent based mine detection and route planning using LRTA* algorithm. Proceedings of the International Conference on Computer Systems and Applications, IEEE-CSA

**Kashif Zafar** received the MS degree from the City University of New York, USA in 2002 and presently pursuing his PhD in computer science from FAST-NU, Islamabad. His research interests include swarm intelligence, machine learning, and artificial intelligence. He has publications in international conferences and he is actively involved in projects under machine intelligence group.



**Abdul Rauf Baig** received his B.Sc. degree in electrical engineering from NED University Karachi in 1986, M.S. degree in computer science from Superlec, France in 1996, and PhD degree in computer science from University of Rennes-1, France in 2000. He is the head of machine intelligence group at FAST-NU, Islamabad and working as a Professor of computer science. He has publications in reputable conferences and journals. His research interests include swarm intelligence, machine learning, data mining, and computational intelligence.