Journal of Circuits, Systems, and Computers
Vol. 20, No. 3 (2011) 457–478

DOI: 10.1142/S0218126611007396

World Scientific
www.worldscientific.com

ROUTE PLANNING AND OPTIMIZATION OF ROUTE USING SIMULATED ANT AGENT SYSTEM*

KASHIF ZAFAR†, RAUF BAIG‡, NABEEL BUKHARI§
and ZAHID HALIM¶

National University of Computer and Emerging Sciences,
Islamabad, Pakistan
†*kashif.zafar@nu.edu.pk*
‡*rauf.baig@nu.edu.pk*
§*i060315@nu.edu.pk*
¶*zahid.halim@nu.edu.pk*

Received 30 April 2010
Accepted 30 November 2010

This research presents an optimization technique for route planning using simulated ant agents for dynamic online route planning and optimization of the route. It addresses the issues involved during route planning in dynamic and unknown environments cluttered with obstacles and objects. A simulated ant agent system (SAAS) is proposed using modified ant colony optimization algorithm for dealing with online route planning. It is compared with evolutionary technique on randomly generated environments, obstacle ratio, grid sizes, and complex environments. The evolutionary technique performs well in simple and less cluttered environments while its performance degrades with large and complex environments. The SAAS generates and optimizes routes in complex and large environments with constraints. The traditional route optimization techniques focus on good solutions only and do not exploit the solution space completely. The SAAS is shown to be an efficient technique for providing safe, short, and feasible routes under dynamic constraints and its efficiency has been tested in a mine field simulation with different environment configurations and is capable of tracking the moving goal and performs equally well as compared to moving target search algorithm.

Keywords: Agent; ant colony optimization; optimization; route planning; swarm intelligence.

1. Introduction

A swarm is defined as a set of nature-inspired mobile agents that collectively carry out a distributed problem solving. These mobile agents can achieve higher objectives when working in collaborative manner. They are able to communicate with each other directly or indirectly by acting on their local environment. Swarm intelligence

*This paper was recommended by Regional Editor Majid Ahmadi.

is the new area in artificial intelligence, inspired from swarm behavior. Swarm intelligence[1] is the property of a system whereby the collective behaviors of unsophisticated agents interacting locally with their environment cause coherent functional global patterns to emerge.

Ant colony optimization[1] and particle swarm optimization[1] 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,[2] 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 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.[3–6] This paper presents simulated ant agents with route planning capabilities. 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 complete design objectives. The ant agents use the modified ant algorithm for finding the optimum route between the start and the goal.

2. Problem Formulation

The route planning and optimization of routes have been considered as NP complete problem.[7] 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[7] and meta-heuristic approach is one of them.

An offline planner generates the complete plan before the task is performed.[3] The traditional offline planners often assume that the environment is completely known and they try to find the plan based on the shortest distance criteria. They cannot handle dynamic environments and are limited to their initial plan. When the environment remains constant throughout the planning process, the route planning is called static offline route planning. A*,[8] RTA*[10–13,29] 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[9] or unknown and requires a 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.[3] The 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 is often prone to slow response time. They require a sophisticated algorithmic approach to find the shortest route.

There are different approaches for online route planning. Some of the online planners generate a 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 the 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 the moving target by ACO can be considered as a dynamic online planning problem. There are a 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. Ant Colony Optimization

Ant colony optimization[1] is a swarm based algorithm that has been successfully used for optimization problems. Ants are tiny insects that find shortest routes from the nest to the food sources by using collective swarm behavior. The ant colony algorithm was presented by Marco Dorigo *et al.*[1] in 1996 and was found to be a robust optimization algorithm for a number of problems. Each ant constructs a complete solution and evaluates the solution for fitness measure. The communication between ants is achieved by the concept called as stigmergy,[1] i.e., each ant deposits a pheromone during traversing and the ants that follow them use the pheromone trace as the fitness of the path and selects the path with maximum pheromone deposit. There is a method of pheromone update across each path. The pheromone decay can be implemented by using different techniques.

The ants deposit pheromone at a constant rate and the path with the maximum accumulation of pheromone will be the optimum path. The shortest path has the maximum pheromone deposit as compared to the larger path. The algorithm is initialized with a specific population size and iterations are pre-defined. The ants

traverse different paths and update the pheromone level. The selection of the next node for each ant "k" is selected by using the state transition rule,[1] as shown in Eq. (1). The allowed k is the set of available states from which the kth ant can choose.

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\Sigma_{k\varepsilon \text{allowed } k}[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}} \,. \tag{1}$$

It acts like a probabilistic function for the node selection. It uses distance and the pheromone level in deciding the next node for ant. The alpha and beta parameter are used to balance the contribution of pheromone level and the heuristic function. The Manhattan distance and Euclidean distance have been used and compared in experimentations. Here τ_{ij} used in Eq. (1) is the amount of pheromone level on the edge (ij), $\eta_{ij} = 1/d_{ij}$ is a heuristic value, and α and β are constants that determine the relative influence of the pheromone value and that of the heuristic value on the decision of the ant. For some constant $mbest \leq m$, the $mbest$ ants having found the best solutions are allowed to update the pheromone matrix. Every ant that is allowed to update adds pheromone to every edge (ij) which is on its tour. The amount of pheromone added to such an edge (ij) is Q/L where L is the length of the tour that was found and Q is a constant as shown in Eq. (2).

$$\tau_{ij} = \tau_{ij} + \frac{Q}{L} \,. \tag{2}$$

But before that is done some of the old pheromone is evaporated by multiplying it with a factor $\rho < 1$ as shown in Eq. (3). The previous pheromone level should not have a too strong influence on the future decision. A bad choice node may have been selected as a component of a good solution, and a high pheromone value may have been deposited on the link. The solution may have been even better, if the node had been replaced by another node. But now, due to high pheromone value it will continue to have a strong probability of selection for a considerable length of time (until pheromone values on nodes in competition become much higher). To shorten that time of wrong influence, the pheromone values are decayed. This implies that a high level of pheromones will be associated only with links that continuously get selected in good solutions.

$$\tau_{ij} = \rho \cdot \tau_{ij} \,. \tag{3}$$

The algorithm stops when some stopping criterion is met, e.g., a certain number of generations have been done or the best found solution has not changed for several generations. Ants are able to find their way to and from food, and this technique is modeled in a new approach called the ant algorithm.

4. Literature Survey

Swarm intelligence has been used for optimization problems like the traveling salesman problem,[14] job-shop scheduling,[15] clustering,[16] constraint satisfaction

problems,[17] and planning,[18] etc. Mei *et al.*[6] used the combination of artificial potential field (APF)[19] with ACO for path planning of a robot in a dynamic environment. This paper presents the concept of local and global path planning. The pheromone was used to prevent the 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 Vien *et al.*[5] This paper presents path planning based on the 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. The Ant-Q algorithm performs better than the genetic algorithm with a higher convergence rate. It is inspired from Q-learning and ant behaviors. It requires a good cost function for the moving path along with an effective Ant-Q value method. The cost function helps in choosing collision-free and shortest routes while the updating method helps to find optimum solution with faster convergence. It uses the pseudo-random proportional action choice rule as state transition rule along with three categories for updating, i.e., local updating, mixture updating and global updating for optimum solutions. They compared the results with the 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 require testing in different environment representations.

Tambouratzis *et al.*[24] presents a progressive optimization of organized colonies of ants (POOCA) algorithm for robot navigation. It focuses on route planning in an unknown environment cluttered with obstacles. With restricted number of ants in the colony, the path convergence is achieved faster. The POOCA uses a combination of co-operation inherent in ACO along with the spread of activation around the winner node during training of the self organizing maps (SOM). The paths are smooth as compared to the Ant-Q algorithm and they cater to dynamic environments as well. The obstacles appear, disappear or move and similarly goal changes its location as well. The environment has been discretized in time and space, i.e., ant motion is performed in terms of time steps and synchronously for all the ants while

the environment is represented as rectangular grid. The location of the end point remains unknown until discovered by the first ant to be reached at the end point. It employed a combination of the offline and online pheromone updating. A concept of pheromone leakage has been given to the neighboring nodes and incorporates the algorithm. It reduces with each subsequent trip of the ant and is proportional to the number of path cells. In this technique, the ant completes a sufficient number of round trips and it depends on the size and complexity of the environment.

Qiang *et al.*[4] presents a global path planning approach based on the ACO algorithm. They present the concept of the neighboring area and smell area as modification in the ACO. The path has been divided into three segments, i.e., path between the nest and initial position of each ant, path between the initial position of each ant and the position that ants enter the smell area and finally the path between the position that ants enter the smell area and the food. Similarly, the concept of the neighboring area has been introduced for directing the ants towards the goal while avoiding obstacles. It also helps in finding the shortest route for the ants. This technique is useful for the optimization of global path but lacks the capability for dealing with moving obstacles and dynamic environments.

5. Route Planning

Route planning can be categorized into online and offline based on the methodology for environment exploration. In online route planning, the agent explores the environment as it moves around while in offline route planning, the agent has already got the environment information beforehand. Different techniques have been explored for route planning such as neural networks,[25] genetic algorithm,[26] fuzzy logic,[27] artificial potential field,[19,20] frontier-based exploration,[21–23] etc. It is a difficult job to find a safe, strategic route in a terrain with obstacles and optimization constraints. Some papers use graph based methods[8] and some use potential fields[19] for space representation. This paper focuses on grid-based environment representation. The details for route planning strategies are discussed below.

5.1. *Off-line route planning*

Off-line route planning[3] uses a prioritized directional approach to find the shortest path between two given points. The start and the goal (destination) points are provided beforehand on the map. The agents then use an exhaustive search to find the feasible paths. This technique works well with static environments. Static environments are those environments which, once discovered completely, do not change with respect to the obstacles. But if this approach is applied on the environment which is dynamic, and the path which initially calculated changes, the agent has to re-compute everything from scratch with respect to the path exploration algorithm. For example, if the agent was to move from point A to point B, if

somewhere in between, the path dynamics are changed from that point onwards, the agent will have to re-compute everything from that mid-point to the goal point, say B. When the environment is completely known beforehand, the route planning is called offline.

5.2. *On-line route planning*

In on-line route planning,[3] agents can reconfigure the planning phase and re-evaluate the route from that particular point to the goal point. Different techniques have been used for online planning specially the evolutionary algorithm. This approach was based on the assumption that the agents would be dealing with dynamic environment instead of static, and it would neither be feasible nor practical to re-compute the whole path just because there has been a little change in the environment. This approach successfully mutates an infeasible path to find an alternative route. Where such a mutation is not possible, it simply uses the method of cross over to link up a point on our current path, to a secondary (alternative) path, without re-computing the whole scenario.

Initially, it computes a few random paths from point A to point B, and then applies the processes of crossover and mutation on different paths, to generate a more feasible, practical, and optimal path. The process of crossover simply tries to join two (or more) different paths, in order to find a better solution. The mutation part, simply introduces slight changes in the route in order to find a better turn.

5.3. *Ant agent based route planning*

The simulated ant agents are used for online route planning. At the start of simulation, the ant agents traverse the grid area and try to find the goal node. Once the goal node has been discovered, the pheromone level of that particular path is updated. The ant agent selects the next node probabilistically by using a roulette wheel method and the ant agent traverses a path and updates the pheromone level for each of the solution found. Each ant agent constructs a complete solution and an evaluation function evaluates each generated path continuously. The algorithm runs till the desired number of iterations completed.

Ant agents use the phenomenon of stigmergy for indirect communication between different ant agents. After the selection of the number of ants and number of iterations, the user loads the map with a local and global view. Four different map sizes have been used for the simulation, i.e., $20 \times 20, 40 \times 40, 60 \times 60$, and 80×80. Each map has a different number of mines, obstacles, starting state and goal state parameters. After the selection and loading of the map, the pheromone deposit is initialized randomly to a small value between 0.01 to 0.09. The heuristic function consists of the pheromone level, distance from the start state to the goal state, clearness and smoothness of the route.

6. Simulated Ant Agent System

Ants are tiny insects capable of finding the 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[1] is the technique inspired by these tiny ants for optimization problems like the 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 the 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 the 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. The stigmergy is a technique of indirect communication used by ant agents. Ant agents update the solution path by updating the 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 control the convergence process. The alpha parameter is related with the 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 a modified ACO algorithm. The goal state remains fixed and environment is static and unknown for ant agents. After loading of the map with the 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 the execution. The ant agent starts exploring the environment randomly and the selection of the next node takes place by using the probability of the selection formula given in Eq. (1). The formula consists of the 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.[8] The Manhattan distance is the number of vertical and horizontal moves to reach the goal

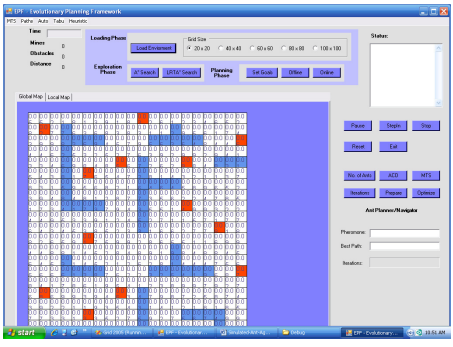
state. Before taking the step, each ant agent calculates the probability for each adjacent node and selects the next node by using the roulette wheel. The ant agent continuously explores the static environment for goal state. As soon as it finds the goal state, the pheromone at the resultant path is updated by using the updating equation. Finding the solution in a dynamic environment is difficult when multiple constraints are enforced. The SAAS has been applied for the dynamic environment in the same way as the moving target search (MTS).[11,29,30] The simulated ant agent is capable of locating the moving target by using the modified ACO. Again the 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 the moving target. During the execution of the task, goal changes and the ant agent reconfigures the planning phase for the new goal state. SAAS has been tested for different map sizes and configurations.

7. Experimentation

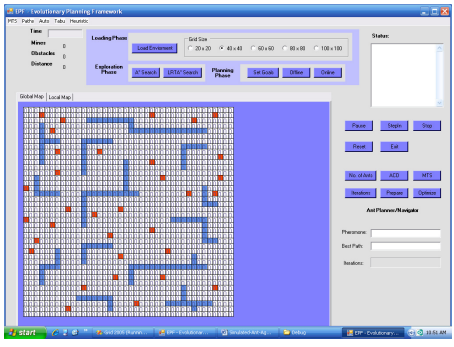
The experimentation has been divided into two phases, i.e., static and dynamic environment. For static environment, the goal state remains static. We use four different size environments, i.e., $20 \times 20, 40 \times 40, 60 \times 60$, and 80×80 grid maps as shown in Fig. 1.

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 the 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 the number of iterations, number of ants, and randomly changing goals for incorporating dynamism in experimentation.

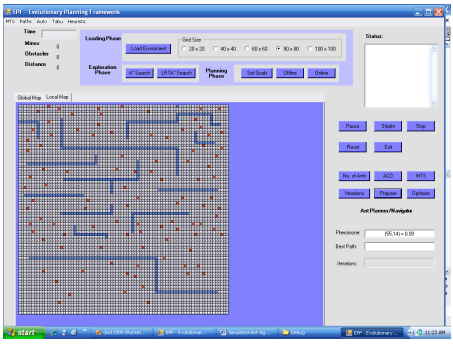
The first experiment deals with static environment and we use 50 iterations, 1 ant and 30 experimental runs. The results are shown in Tables 1−4. Four different maps have been used and each with different obstacle ratio and grid size. A high obstacle ratio and randomly generated maps have been used to simulate complex environments. The main parameters for the simulation are the 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. The SAAS has been tested by using the 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



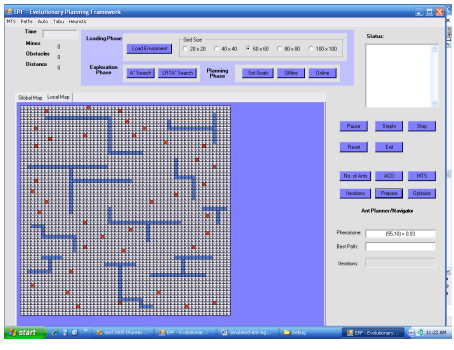
(1) 20 x 20



(2) 40 x 40



(3) 60 x 60



(4) 80 x 80

Fig. 1. 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									
Alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
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									
Alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
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									
Alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
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

the v-edges of the path. After the generation of the path, the optimization module measures the distance of the generated route and compares the distance after repairing the path. If the distance remains the same as of the generated path, then the system regenerates the new optimized path. 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

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

	Beta									
Alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
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

provides two options for changing the goal. 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 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 the different map considerations used in the experiments.

8. Results

The experiments have been conducted separately for static goal and for the dynamic goal environment. The starting point and the goal point remains constant throughout each run. Each run consists of 50 iterations with a 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. The probability of selection for the 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 20×20, 40×40, 60×60, and 80×80 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 alpha value corresponds to the pheromone level and beta value corresponds to the heuristic, i.e., distance measure from the start point to the goal point. The results show the influence of beta on the cells traversed by guiding the ant agents towards the goal. The more the value of beta, the better the results. Each experiment has been conducted 30 times and the average value for each 30 runs is reported in the above figures. The starting point and the goal points have been selected as two extreme positions on the map. The maps are generated randomly with a different number of mines, obstacle ratio and complexity. The color coding has been used to distinguish mine, obstacle and empty cell that can be traversed. The feasible routes are of the shortest distance while avoiding mines and obstacles.

The graphs in Figs. 2–5 show the cells traversed for each value of beta while keeping the value of alpha constant. It is observed in each of the graphs below that the route becomes shorter with an increasing value of beta. The results have been compared with ideal solutions using the Manhattan distance as well as the Euclidean distance for heuristic measure and the performance of the simulated ant agent system is comparable with ideal solutions generated by the A* algorithm. The SAAS has performed equally well as compared to the A* algorithm that is known to be the best algorithm to find the shortest distance between two given points. The experiments show the scalability and robustness of the simulated ant agent system by using larger grid maps and its performance remains consistent throughout the simulation run.

The results for simple to complex environments have shown the scalability of the simulated ant agents. The convergence for each run depends on the values of alpha and beta. The table list has been used to restrict the ant agents to new routes and avoids loops. The SAAS without using a table list explores the same nodes again and again and gets trapped in cycles. The table list has been maintained to enhance the performance of SAAS by removing cycles and avoiding loops. The number of mines, obstacles and complexity of the grid maps play an important role in the convergence of the algorithm. The SAAS has been tested for static as well as dynamic goals and shows promising results.

The SAAS has been applied to the dynamic environment for presenting a moving target search. After the selection for the number of times to change the goal state, the SAAS runs for tracing the first goal. After tracing the first goal with the shortest route the goal changes randomly to a new random configuration. The next goal appears within 10% of the area of the previous goal. The ant agents continue to evaluate the probabilities of the new node selection and again run the roulette wheel. SAAS is capable of acquiring the moving goal with the shortest route. ACO is primarily an optimization technique and SAAS uses ACO for moving target search in the dynamic environment. The results showed the robustness and scalability of the SAAS in a dynamic environment.

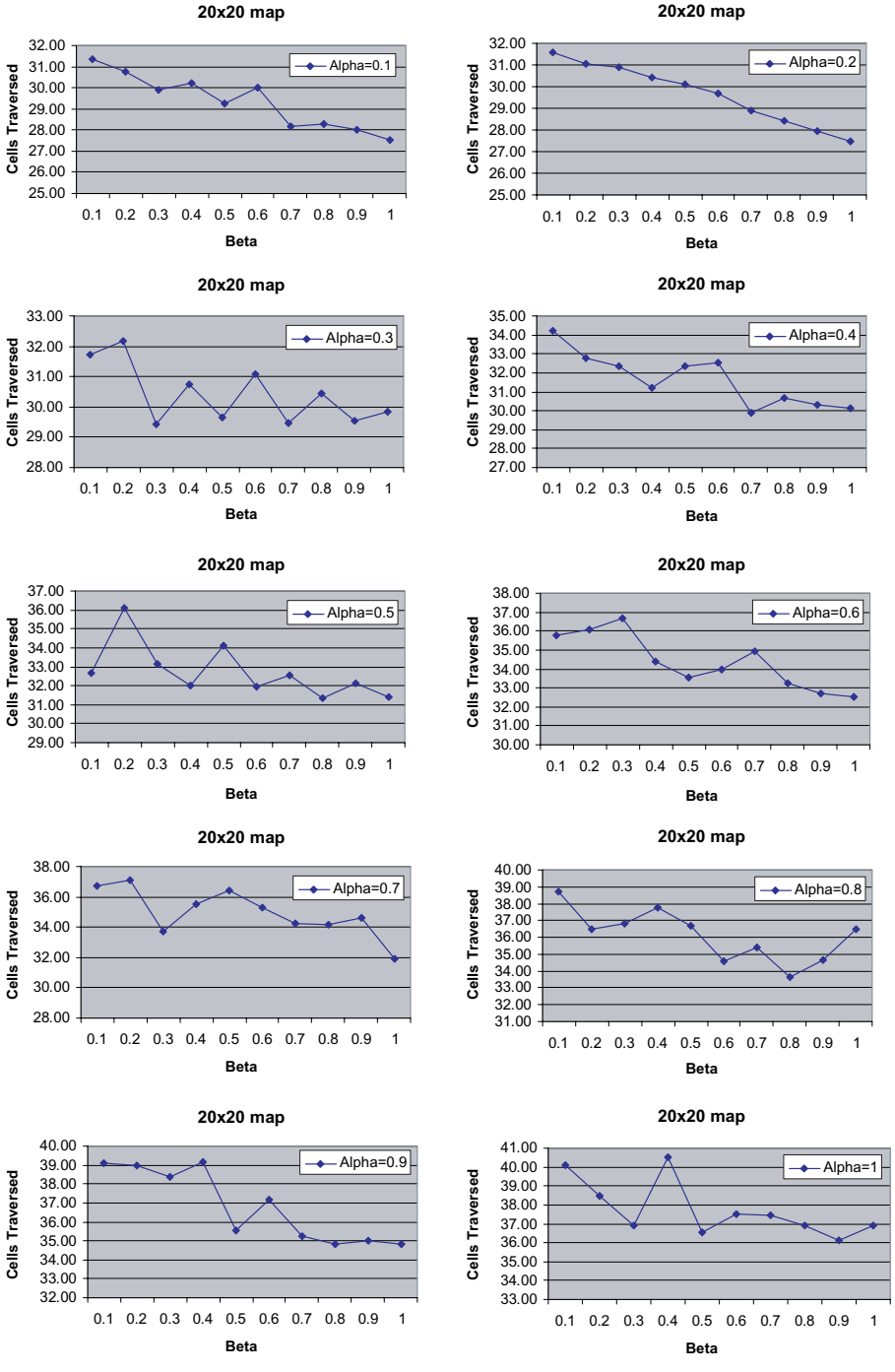


Fig. 2. 20×20 map results with different values for alpha and beta.

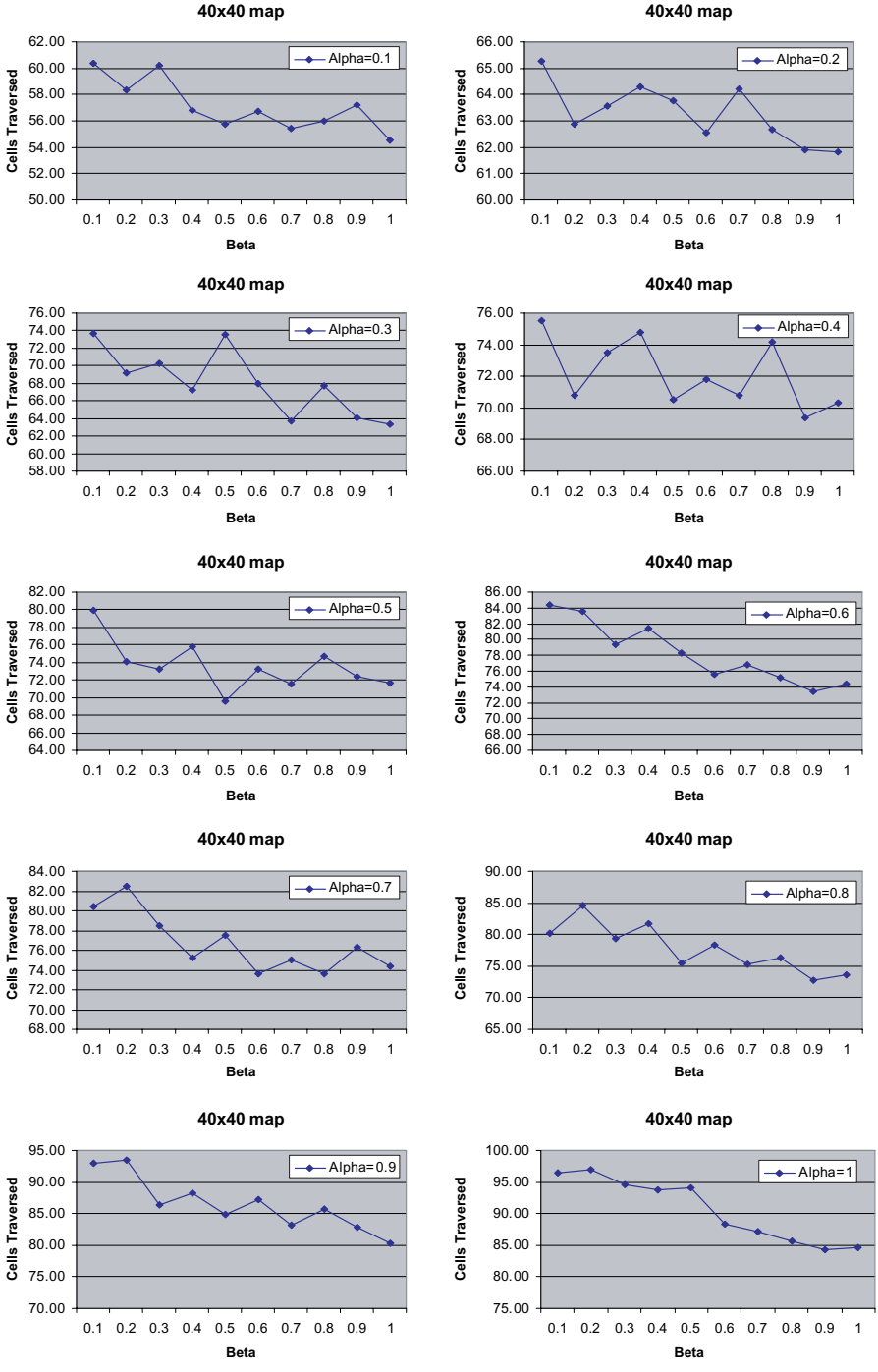


Fig. 3. 40×40 map results with different values for alpha and beta.

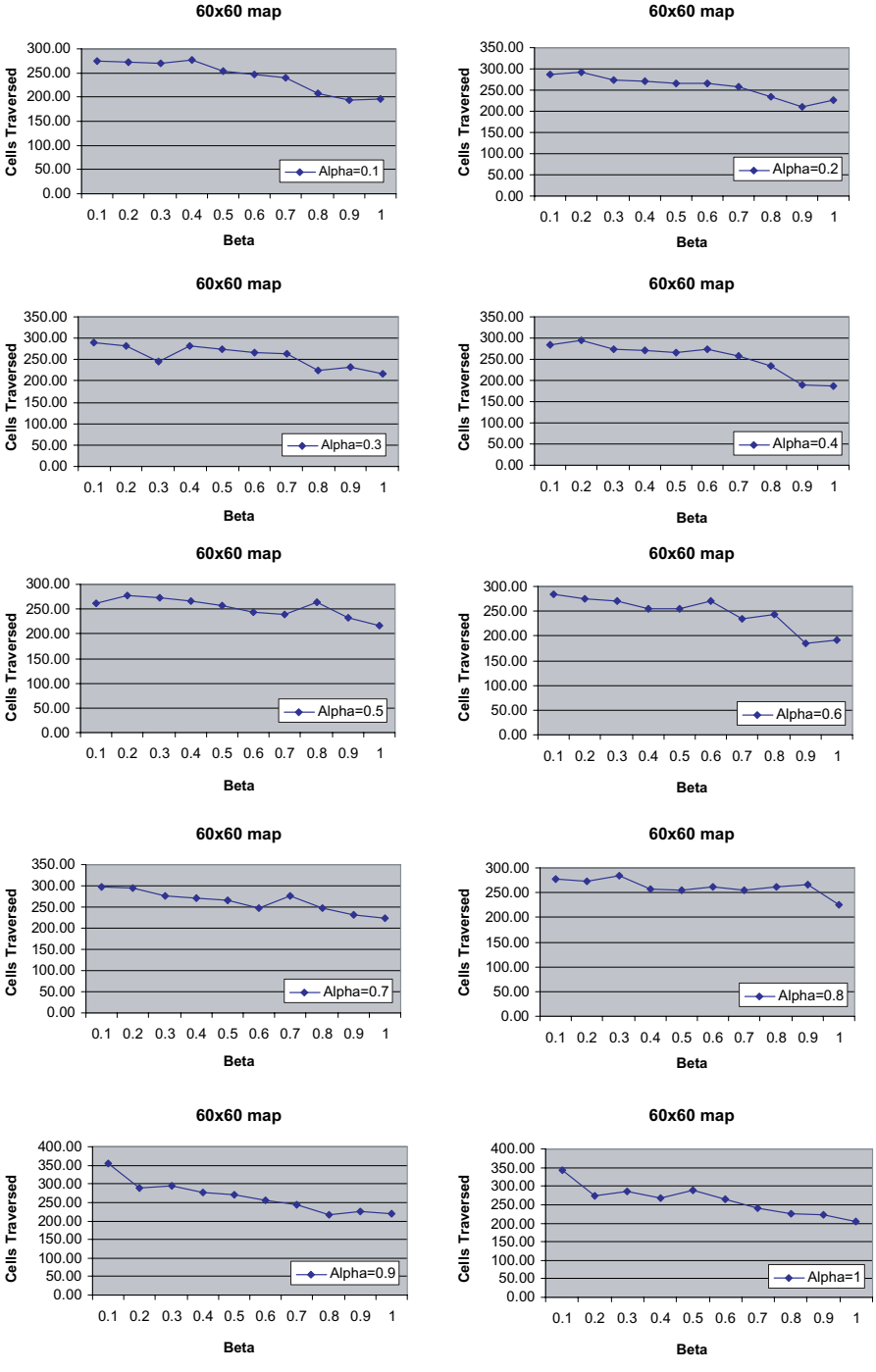


Fig. 4. 60×60 map results with different values for alpha and beta.

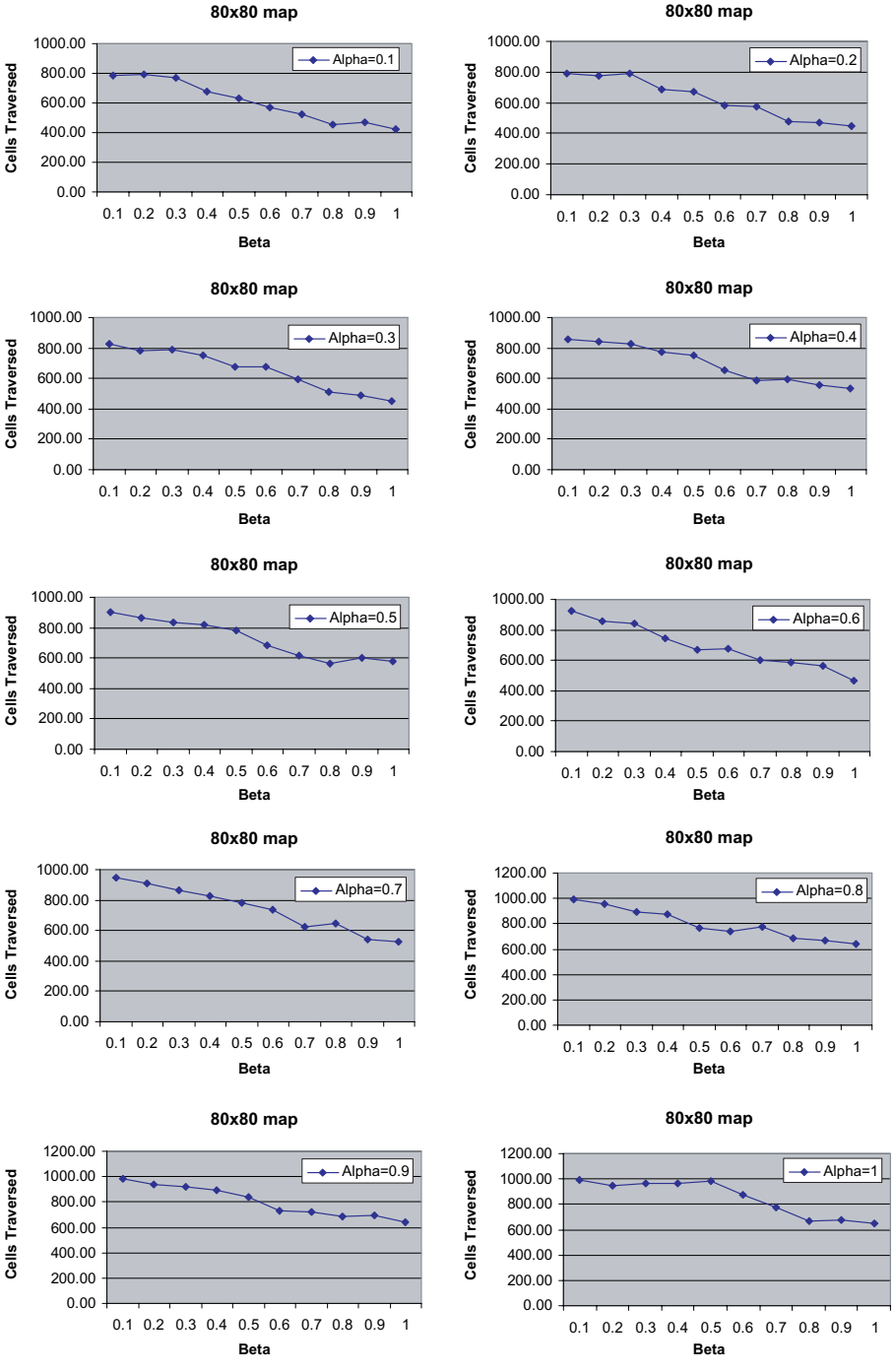


Fig. 5. 80×80 map results with different values for alpha and beta.

Table 5. Comparison of simulated ant agent system for moving target search with A* algorithm.

Map = 20 × 20	Moving target search using simulated ant agent system (alpha = 0.1 and 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-A*	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-A*	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-A*	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-A*	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-A*	6	10	9	7	17	6	8	9	15	11
No. of iterations	40	40	40	40	40	40	40	40	40	40

The experiments are conducted with 200 iterations, 5 dynamic targets generated randomly on a 20×20 grid size as shown in Table 5. The intervals are fixed to 40 for each run. The distance of the new goal along with the distance from the previous goal was taken into account for comparison. The path traversed by SAAS has been compared with both the heuristics, i.e., Manhattan distance as well as the Euclidean distance. Each experiment is conducted with different obstacle ratios and map sizes.

The tracking of the moving target has been implemented using a simulated ant agent system with different parameters and for different grid sizes. Each experiment uses 5 different goals generated at random locations with 200 iterations as one run. Table 5 shows the results for each goal and compares the route generated by SAAS as cells traversed with the Manhattan distance as well as with the Euclidean distance. The experimentation uses a different obstacle ratio, grid sizes and heuristic measure. The simulation has been tested with different options to generate new goals randomly. In option 1, we select the number of times to change the goal. In option 2, we provide the percentage of goal change during the complete run of iterations while the third option provides the exact gap between iterations to change the goal.

Finally, the results obtained by SAAS in a static environment are compared with the evolutionary technique using the genetic algorithm as shown in Table 6.

The paths are represented as chromosomes and a population size of 20 has been used. The mutation and one point crossover have been used along with the fitness function. The paths obtained have been compared with SAAS again with $20 \times 20, 40 \times 40, 60 \times 60$, and 80×80 maps and SAAS outperforms the evolutionary algorithm. The convergence rate of genetic algorithm is found to be slow as compared to the convergence rate of SAAS. The results shown in Table 6 clearly show the difference in cells traversed by both the techniques. The smallest values for each map have been used with the best values of alpha and beta for comparison. The number of cells traversed for SAAS and GA has been taken as the average of 30 runs.

9. Optimization of Route Planning

A separate module for route optimization has been implemented to repair the generated routes. The number of v-edges are counted in each generated route and fixed by taking the route with the same number of cells. It does not reduce the number of cells traversed. It only increases the smoothness of the route. By replacing the v-edge

Table 6. SAAS and GA with average of 30 runs.

Map size	Alpha	Beta	SAAS	GA
20×20	0.2	1	27.52	28.65
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	753.56

with the adjacent cell, the route becomes smooth and straight. The route optimizer compares all the combinations of the v-edges along the route and removes the v-edges by replacing them with diagonal cells along with keeping same cells and distance.

The SAAS not only finds the shortest distance route, it also optimizes the generated route as shown in Figs. 6 and 7. The route for each goal in the moving target search has also been optimized by using the route optimizer. The routes generated by

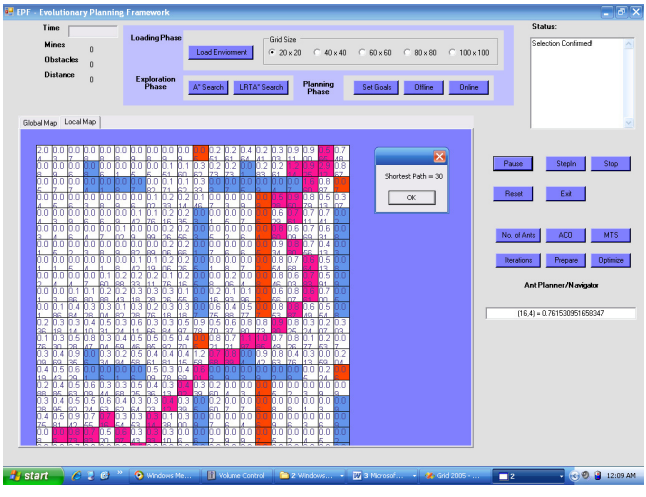


Fig. 6. Shortest path without optimization.

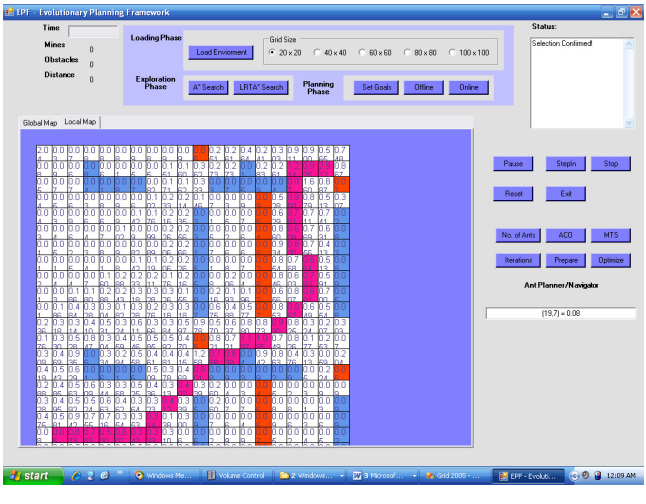


Fig. 7. Shortest path with optimization.

the moving target search are available in the menu list with the number of v-edges in each route. By applying the route optimizer, we repair each route and it provides clear and smooth routes from the start state to each of the goals in the moving target search. The diagonal movement generates v-edges that need to be repaired by using the route optimizer. The smoothness of the routes depends on the number of v-edges and clearness of the route depends on the obstacle avoidance and avoiding mines. The clearness of the route has been incorporated in the fitness function but the smoothness of the route requires a separate module. The main objective of this module is to repair the path without altering the original route plans. For this purpose, each generated route has been compared with the number of cells traversed for each of the repaired path and the resultant selected route will have the same number of cells and distance measure.

10. Conclusion

The simulated ant agent system has been used successfully for route planning and optimization of routes in static and dynamic environments. The SAAS has been tested with simple to complex environments and was found to be a robust, efficient and scalable online line route planning system. The route optimizer repairs the generated routes while keeping the number of cells and distance same. The results for the static environment for each map size are also compared with the evolutionary technique and SAAS performed better then the genetic algorithm. For dynamic environments, the SAAS has been compared with the moving target search algorithm and proved to be an efficient method for dealing with the dynamic environment. The target goals for the SAAS have been changed and SAAS effectively tracked the new goals. The distance of each new goal is compared with the A* algorithm and found to be comparable. This paper implements SAAS for mine detection and optimized online route planning for static as well as for the dynamic environment. The consistent experimental results with different size maps have shown the scalability and robustness of the system for handling the dynamic environment. The SAAS can be further tested for other constraint satisfaction and multi-objective optimization problems in different application areas.

References

1. A. P. Engelbrecht, *Computational Intelligence*: *An Introduction* (John Wiley & Sons, 2002).
2. M. Dorigo and K. Socha, An introduction to ant colony optimization, *Technical Report Series*, IRIDIA, Bruxelles (2007).
3. J. Xiao, Z. Michalewicz, L. Zhang and K. Trojanowski, Adaptive evolutionary planner/navigator for mobile robots, *IEEE Trans. Evol. Comput.* **1** (1997).
4. Z. Qiang and Z. Xing, *Global Path Planning Approach Based on Ant Colony Optimization Algorithm* (Springer-Verlag, 2006).

5. N. A. Vien and N. H. Viet, *Obstacle Avoidance Path Planning for Mobile Robot Based on Ant-Q-reinforcement Learning Algorithm* (Springer-Verlag, 2007).
6. H. Mei, Y. Tian and L. Zu, A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment, *Int. J. Infor. Technol.* **12** (2006).
7. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms* (MIT Press, Cambridge, 2001).
8. S. J. Russell and P. Norvig, *Artificial Intelligence — A Modern Approach* (Prentice Hall, Englewood Cliffs, New Jersey, 1995).
9. A. Stentz, Optimal and efficient path planning for partially-known environments, *Proc. IEEE Int. Conf. Robotics and Automation*, May (1994).
10. R. E. Korf, Real-time heuristic search: New results, *Proc. AAAI* (1988), pp. 139–144.
11. R. E. Korf, Real-time heuristic search, *Artif. Intell.* **42** (1990) 189–211.
12. R. E. Korf, Real-time heuristic search: First results, *Proc. AAAI* (1987), pp. 133–138.
13. R. E. Korf, Real-time heuristic search, *Artif. Intell.* **42** (1990) 189–211.
14. T. Stutzle and M. Dorigo, ACO algorithms for the traveling salesman problem, *Proc. Conf. EUROGEN* (1999).
15. J. Heinonen and F. Pettersson, Hybrid ant colony optimization and visibility studies applied to job-shop scheduling problem, *Appl. Math. Comput.* **187** (2007) 989–998.
16. J. Handl, J. Knowles and M. Dorigo, On the performance of ant based clustering in design and application of hybrid intelligent systems, Frontiers in Artificial Intelligence and Applications, Vol. 104 (IOS Press, Amsterdam, 2003), pp. 204–213.
17. L. Schoofs and B. Naudts, Ant colonies are good at solving constraint satisfaction problems, *Proc. Congress on Evolutionary Computation* **2** (2000) 1190–1195.
18. J. E. Bell and P. R. McMullen, Ant colony optimization techniques for the vehicle routing problem, *Adv. Eng. Infor.* **18** (2004) 41–48.
19. Z. Li and X. Chen, *A New Motion Planning Approach Based on Artificial Potential Field in Unknown Environment*, PDCAT 2004 (Springer-Verlag, 2004), pp. 376–382.
20. A. Howard, M. J. Mataric and G. S. Skhatme, Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, *6th Int. Symp. Distributed Autonomous Robotics Systems* (2002), pp. 299–308.
21. B. Yamauchi, A frontier based approach for autonomous exploration, *Proc. IEEE Int. Symp. Computational Intelligence on Robotics and Automation* (1997).
22. B. Yamauchi, Frontier based exploration using multi robots, *Proc. 2nd Int. Conf. Robotics and Automation* (1998).
23. K. Zafar, S. B. Qazi and A. R. Baig, Mine detection and route planning for military warfare using multi agent system, *Proc. Conf. Engineering Semantic Agent Systems*, IEEE-ESAS (2006).
24. T. Tambouratzis, *Progressive Optimization of Organized Colonies of Ants for Robot Navigation: An Inspiration from Nature* (Springer-Verlag, 2007).
25. N. Bin, C. Xiong and Z. Liming, *Recurrent Neural Network for Robot Path Planning*, PDCAT 2004 (Springer-Verlag, 2004), pp. 188–191.
26. S. Liu and Y. Tian, Multi mobile robot path planning based on genetic algorithm, *Proc. 5th World Congress on Intelligent Control and Automation* (2004), pp. 4706–4709.
27. H. R. Beom and H. S. Cho, A sensor-based navigation for mobile robot using fuzzy logic and reinforcement learning, *IEEE Trans. SMC* **25** (1995) 464–477.
28. M. Dorigo, V. Maniezzo and A. Colomi, Ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man, and Cybernetics* **26** (1996).
29. T. Ishida and R. E. Korf, Moving target search: A real-time search for changing goals, *Proc. IEEE PAMI* **17** (1995) 609–619.
30. T. Ishida and R. E. Korf, Moving target search, *Proc. IJCAI* (1991), pp. 204–210.