An ACO Path Planner Using a FIS for Path Selection Adjusted with a Simple Tuning Algorithm

Miguel Porta-Garcia, Oscar Montiel, Roberto Sepulveda

Abstract:

This paper presents a path planner application for mobile robots based on Ant Colony Optimization (ACO). The selection of the optimal path relies in the criterion of a Fuzzy Inference System (FIS), which is adjusted using a Simple Tuning Algorithm (STA). The path planner can be executed in Mode I and Mode II. The first mode only works in the virtual environment of the interface, while Mode II embraces the wireless communication with a real robot; once the ACO algorithm finds the best route, the coordinates are sent to a mobile robot via Bluetooth communication; if the robot senses a new obstacle, the computer is notified and does a rerouting routine in order to avoid the obstacle and reach the goal. In other words, the application supports dynamic search spaces.

Keywords: *ant colony optimization, ACO, autonomous mobile robot navigation, fuzzy logic, path planning.*

1. Introduction

The autonomous mobile robot navigation can be divided in two subsequent problems: path planning and control of movement [1]. Specifically, the tasks involved in navigation are the perception of the environment, the path planning, generation of paths, and finally the route following. This work is concretely focused on the path planning and generation of optimal paths, starting from previous information of the search space. The route following using the real robot is reserved as a second phase of the method, which is explained later.

Robotics is an essential part in automatization of manufacturing processes. Concerning about mobile robots, autonomous navigation entails a great challenge. Mobile robots can be very useful in different situations where humans could be in danger or when they are not able to reach certain target because of terrain conditions. Then, the path planning is an interesting and challenge subject for research, and it has many different approaches. This paper proposes a method based on an Ant Colony Optimization Meta-Heuristic (ACO-MH) for path planning finding the best route according to certain cost function.

The ACO-MH is inspired in the foraging behavior of real ants for finding the optimal path from the nest to where the food is. As the communication method, some ant species use stigmergy, a term introduced by French biologist Pierre-Paul Grassé in 1959. With stigmergy, each ant communicates with one another by modifying their local environment. The ants achieve this task by laying down pheromones along their trails [2]. ACO-MH solves mainly combinatorial optimization problems defined over discrete search spaces. The ant-based algorithms, developed as a result of studies of ant colonies, are referred as instances of ACO-MH [3]. In this work, the proposed method is based on an ACO-MH instance, called Simple Ant Colony Optimization (SACO). Some innovative ideas to improve the basic ACO algorithm used for the suggested method are presented and applied to achieve the path planning.

There are several approaches for path planning, like [4], where a method based on an exhaustive search algorithm on graphs for indoor environments. In [5], a visual tool to teach graph-based applications uses the robot motion planning as example for academic purposes, and the shortest path computation among static obstacles is made using Dijkstra's algorithm. There are some other hybrid approaches like [6], where a smooth path planning method for unknown environments with nonholonomic robots is proposed, and a fuzzy controller is used to wall following. However, few approaches like the one presented in this paper have the ability of allowing rapid changes in the search space. Taking advantage of the ACO algorithm characteristics, which permits working with dynamic optimization problems, the proposed framework has a rerouting capability for avoiding obstacles if the robot sense an object over the given path, performing the necessary procedures to recalculate the avoidance route after the computer has been notified.

Referring to emergent natural optimization methods, such as ACO, there are some similar works like [7], where the robot has to visit multiple targets, like the traveling salesman problem but with the presence of obstacles. The robot in this case is modeled as a point robot; that is, the robot occupies an exact cell in the discrete representation of the workspace. Using several robots as ants, this robot team architecture has to be in constant communication with each other at all times to share pheromone information. There are some other approaches for similar tasks, like [8], where a new meta-heuristic method of ACO is proposed to solve the vehicle routing problem, using a multiple ant colony technique where each colony works separately. On the other hand, there are many path-planning approaches by other soft computing techniques, such like Genetic Algorithms [9-15]. Then, as it can be observed in actual reports [16, 17], the focus on ant algorithms is growing becoming an interesting alternative of solution for path planning.

2. SACO algorithm

The SACO is an algorithmic implementation that adapts the behavior of real ants to solution of minimum cost path problems on graphs. A number of artificial ants build solutions for a certain optimization problem and exchange information about the quality of these solutions making allusion to the communication system of the real ants [18].

Let be the graph $G = (V,E)$, where V is the set of nodes and E is the matrix of links between nodes. G has $n_G = |V|$ nodes. Let be L^k the number of hops in the path built by ant k from the origin node to the destiny node. Therefore, it needs to be found:

$$Q = \{q_\omega ... q_f \,|\, q_i \in C\} \tag{1}$$

Where Q is the set of nodes representing a path at least continuous with no obstacles; $q_\omega ... q_f$ are former nodes of the path and C is the set of possible configurations of the free space. If $x^k(t)$ denotes a Q solution in time t, $f(x^k(t))$ expresses the quality of the solution. In general terms, the steps of SACO are as follows:

- Each link (i,j) is associated with a pheromone concentration denoted as τ_{ij}.
- A number $k = 1,\ldots,n_k$ are placed in the origin node (the nest).
- On each iteration or epoch all ants build a path to the destiny node (the food source). For the next node selection it is used the probabilistic formula:

$$p_{ij}^k(t) \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}(t)} & if \quad j \in N_i^k \\ 0 & if \quad j \notin N_i^k \end{cases} \tag{2}$$

In Eq. 2, is the set of feasible nodes connected to node i with respect to ant k; τ_{ij} is the total pheromone concentration of link ij, where α is a positive constant used as gain for the pheromone concentration influence.

- Remove cycles and compute each route weight $f(x^k(t))$. A cycle could be generated when there are no feasible candidates nodes, that is, for any node i and ant k, $N_i^k = \emptyset$; then predecessor of that node i is included as a former node of the path.
- Compute pheromone evaporation using the Eq. 3.

$$\tau_{ij}(t) \leftarrow (1-\rho)\,\tau_{ij}(t) \tag{3}$$

In Eq. 3, ρ is the evaporation rate value of the pheromone trail. The evaporation is added to the algorithm in order to force the exploration of the ants, and avoid premature convergence to sub-optimal solutions. For ρ = 1, the search is completely random. While an ant takes more time for crossing a path, there is more time for the pheromone trail to evaporate. On a short path, which is crossed quickly, the density of the pheromone is higher. Evaporation avoids convergence to local optimums. Without evaporation, the paths generated by the first ants would be excessively attractive for the subsequent ones. In this way, exploration of the search space is not too restricted.

- Update pheromone concentration by using Eq. 4.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^k(t) \tag{4}$$

- The algorithm can be ended in three different ways:
  - When a maximum number of epochs has been reached.
  - When it has been found an acceptable solution, with $f(x^k(t)) < \varepsilon$.
  - When all ants follow the same path.

3. Improvements over SACO for mobile robotic application

In general, a novel framework to achieve path planning is proposed. Several aspects were included to improve the SACO algorithm. In the algorithm it was designed a fuzzy cost function based in the expert heuristic knowledge; the original transition probabilistic formula (2) was modified to accelerate the decision process mainly in obstacle free search spaces. In addition, it was added a memory capability to avoid the algorithm stagnation. An important characteristic is the added ability to work with dynamic search spaces departing from a giving map or by discovering.

3.1 The search area design

This approach makes some improvements over the SACO algorithm and adaptations for the mobile robot routing problem of this research work. The map where the mobile robot navigates is a search space discretized into a matrix representing a graph of 50x50 nodes, where "0" means a feasible node (plain terrain) and "1" are obstacles. It is remarkable to say that each artificial ant of the algorithm is a scale representation of the real mobile robot, which means the proposed method considers robot's dimensions (for this case, the Boe-Bot); for example, there are going to be situations during the optimization process, where some paths are rejected if the robot doesn't fit in the space between two obstacles. Under this premise, several computations are saved since some nodes are rejected before the algorithm spends time using them to build paths. The 50x50 map represents a 4m^2 area, in a 1:4 scale (cm).

For this method, it is assumed all nodes are inter-connected. In a map with no obstacles, there are 2500 feasible nodes; therefore the matrix of links E would be extremely large. For this reasons E is not used, and the pheromone amount value is assigned at each node, which reduces considerably the complexity of the algorithm and then the processing time. This is equivalent to assign the same pheromone concentration to the eight links around every node. If an analogy with reality is made, this can be seen as ants leaving food traces in each node they are visiting, instead of a pheromone trail on the links.

3.2 The node selection process

Once the ants are placed in the origin node, each ant starts navigating, and the decision process for choosing the next node consist in a 3x3 window of the whole graph. The ant can choose one of the eight nodes around it, and the transition probability (2) is now:

$$p_{ij}^{k}(t)\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)}{\sum_{j\in N_i^k}\tau_{ij}^{\alpha}(t)*\xi^{\beta}} & if \quad j\in N_i^k \\ 0 & if \quad j\notin N_i^k\end{cases} \tag{5}$$

Where ξ represents the Euclidean distance between the candidate node and the destiny node, $\beta \in [0,\infty)$ amplifies the influence of ξ.

The memory capability γ has been added to ants. The value γ represents how many nodes can remember the ant. After γ iterations of the building routes process, this memory is "erased", and the count starts again. The algorithm evaluates the free space covering two nodes of distance from each candidate node around de robot, which is enough to overlay the Boe-Bot dimensions.

3.3 The fuzzy cost function and the Simple Tuning Algorithm (STA)

The cost of the path $f(x^k(t))$ to determine the optimal one is evaluated by a Fuzzy Inference System (FIS), which contemplates not only the length of the path but the difficulty for the navigation. The FIS considers two inputs: *effort* and *distance*. The first one represents the energy spent by the robot to make turns across the path; for example, the effort become increased if the robot has to make a left turn after a long straight line, because it has to decelerate more. *Distance* is the accumulated Euclidean distance at the moment between the visited nodes. The output is a weight assigned to the cost of the path; the more weight is given, the less desirable becomes the path. The output of the FIS is added to the total Euclidean distance of the path, giving the final weight of each one generated by ants. If there are different routes with the same length, the FIS should make a difference of cost giving preference to the straighter paths, like shown in Figure 1. The FIS variables can be seen in Table 1 and the associative memory of the FIS is shown in Table 2.

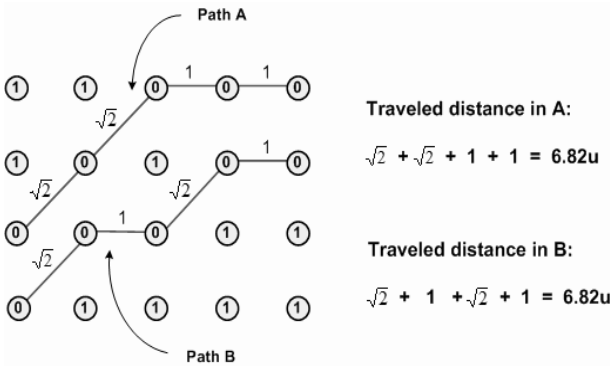

Fig. 1. Path A has same length than path B; however, A implies less effort for robot navigation.

The STA, which is fully addressed in [19], is applied on fuzzy controllers as an attempt to facilitate the tuning process of the FIS, since sometimes becomes overwhelming to find the optimal parameters necessaries for a well performance of the controller. By applying the STA, time and effort is reduced by using a single parameter, the tuning factor k. It is based on the properties of the control surface, allowing the modification of the controller's behavior by means of manipulating the ranges of the membership functions of the input variables.

Table 1. FIS variables.

Input Variables		Output Variables
Effort	Distance	Weight
NE: Normal Effort **NEE:** Normal Extra Effort **BE:** Big Effort **BEE:** Big Extra Effort **VBE:** Very Big Effort	**VSD:** Very Small Distance **SD:** Small Distance **D:** Distance **BD:** Big Distance **VBD:** Very Big Distance	**MW:** Minimum Weight **SW:** Small Weight **W:** Weight **BW:** Big Weight **VBW:** Very Big Weight

Table 2. Fuzzy associative memory of the FIS.

		Distance				
		VSD	**SD**	**D**	**BD**	**VBD**
Effort	**NE**	MW	MW	SW	SW	MW
	NEE	MW	SW	W	SW	MW
	BE	SW	W	W	W	SW
	BEE	BW	BW	W	BW	VBW
	VBE	VBW	VBW	BW	VBW	VBW

In this work, the FIS it isn't used as a controller, but as a decision support system to differ the straighter paths from the winding ones. The output surface without applying STA is shown in Figure 2 and with STA in Figure 3.

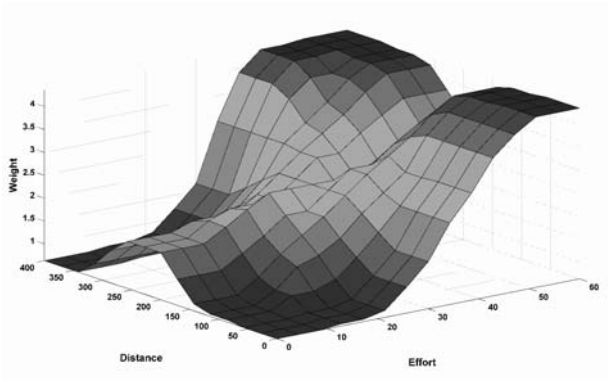

Fig. 2. Output surface before the application of STA.

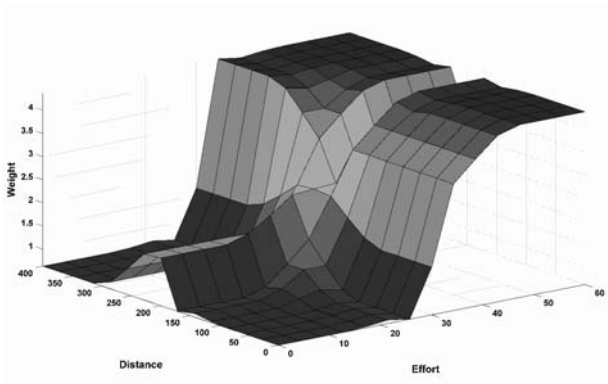

Fig. 3. Output surface of the modified FIS by STA with a tuning factor k = 0.75.

3.4 Dynamic obstacles generation

The algorithm has the capability of sensing changes in the environment, if a new obstacle is placed over the robot's route at time t, it starts a rerouting process in order to avoid the blocking object and get to the destiny node. It has to be considered that after some epochs, the pheromone concentration τ_{ij} is already increased over the visited nodes; then, when a new obstruction appears, it causes evaporation of the pheromone trail around it. This premise prevents stagnation around the obstacle, and τ_{ij} of the surrounding area is given by the minimum pheromone value over the search map at t.

4. The complete framework

A graphical interface, as well as the translation of the obtained solution (optimal path) from the virtual to the real world was implemented to test the proposed path planning method. Figure 4 shows the main screen of the graphic interface made with MATLAB, called ACO Test Center (ACOTC). ACOTC has two operation modes: Mode I and Mode II. Mode I only uses the virtual environment, while Mode II is design to work with the real robot online with the software. Figure 5 and 6 shows the flow diagram of each operation mode.

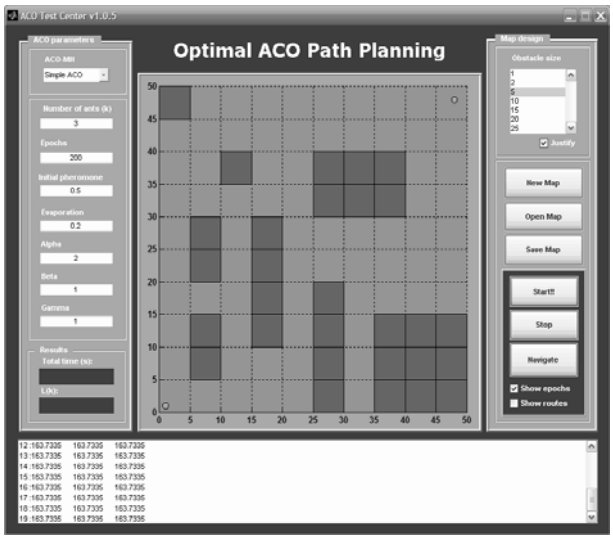

Fig. 4. Main screen of the software interface.

The ACOTC allows the user to design maps for indoor conditions over plain terrain, which later can be built on the real 4m^2 laboratory area (for Mode II), showed in Figure 7. All obstacles are assumed to be cubes of size 1, 2, 5, 10, 15, 20 and 25 map units. In Figure 7, it can be seen that the obstacles are only marked by square cardboards on the floor (according to obstacle dimensions and position of the designed map in ACOTC) covering the area of the map that the Boe-bot should not cross over its path. This is because there is no necessity of sensing obstacles for navigation, since the robot already knows the route that it must be followed. Sensing objects is only for new obstacles appearing on the path, which is discussed later in this chapter.

In Mode I, once the map is created and all ACO algorithm parameters are set up, the optimization process initiates pressing the *Start* button. Under the map display in the main screen the ACOTC displays the

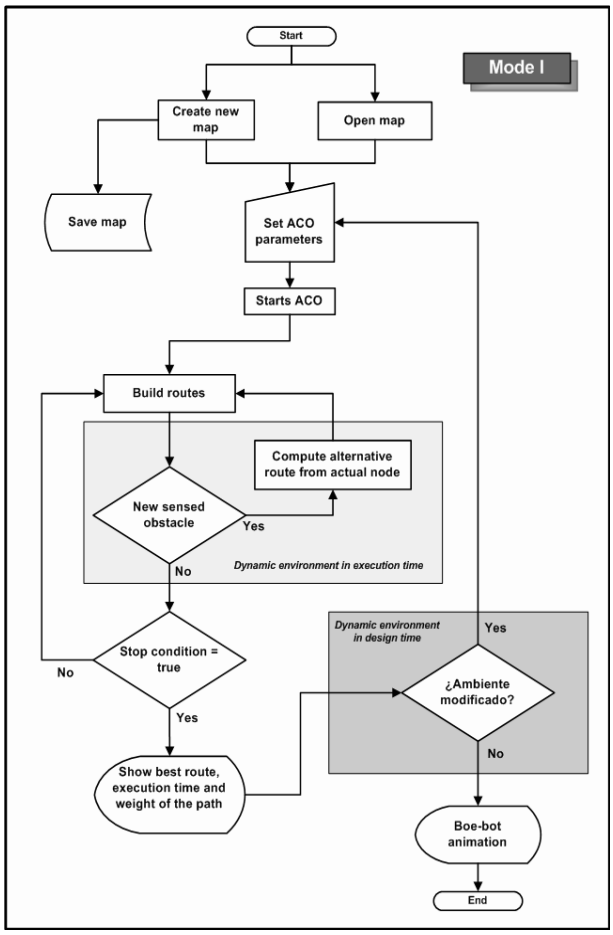

Fig. 5. Flow diagram of ACOTC in Mode I.

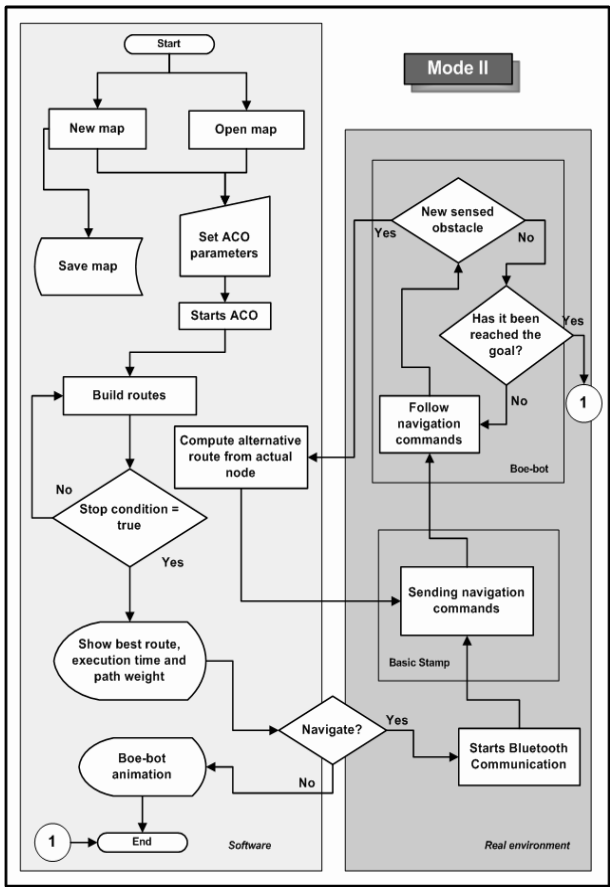

Fig. 6. Flow diagram of ACOTC in Mode II

status of each epoch, showing the weights of the routes of each ant at the moment. The algorithm can be ended when the specified maximum number of epochs has been reached, or by pressing de Stop button. Then, the program shows the solution path that the algorithm found, and now the ACOTC is ready for sending navigation commands to the Boe-bot. Mode II begins when the *Navigate* button is pressed, and the Bluetooth communication is established with the robot and starts navigating. The program translate the sequence of coordinates x and y of each node part of the final route into navigation commands for the mobile robot to move over this nodes as imaginary checkpoints in the real map. Taking advantage of the fact that the map has been discretized, the navigation commands are reduced to:

- 101: Indicates the Boe-bot to go forward one node. From left to right, the first 1 tells the direction, and the second and third digits are reserved for how many nodes the Boe-bot must go in forward direction. For example, to make the robot moves forward ten nodes, the sent command should be 110.
- 200: Turn Left 45°.
- 300: Turn Left 90°.
- 400: Turn Left 135°.
- 500: Turn 180°.
- 800: Turn Right 45°.
- 700: Turn Right 90°.
- 600: Turn Right 135°.
- 000: Gives the order of perform the Bluetooth connection between the two Basic Stamp Modules, the one attached to the computer by the USB port and the one installed in the Boe-bot.
- 900: Finishes the Bluetooth connection.

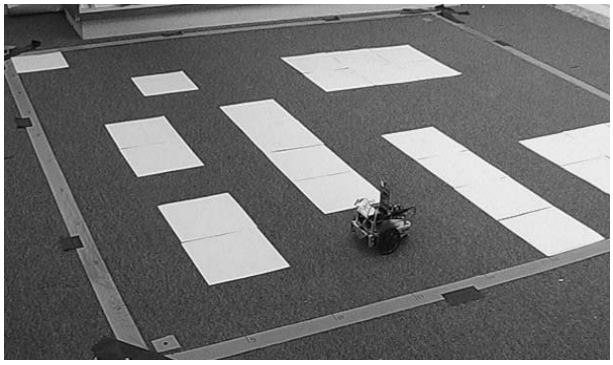

Fig. 7. The real 4m² laboratory area with the Boe-bot.

A carrier board with a Basic Stamp BS2p24 microcontroller module of Parallax is used to make the interface between the Boe-bot and the computer that performs the algorithm processing. The tests were achieved in an HP Athlon64 notebook.

As mentioned before, the mobile robot used in this work is a Boe-bot of Parallax, with a BS2p24 microcontroller. The BS2p24 has a program execution speed of 12,000 instructions per second approximately, with a processor speed of 20 MHz. It also includes a 38 bytes (12 I/O, 26 Variable) RAM size and 8 x 2K Bytes, giving approximately 4,000 instructions. This circuit has 16 I/O pins plus 2 for dedicated serial. The total number of PBASIC commands on the BS2p is 61 [20].

For the wireless communication between the computer and the Boe-bot, it is used two eb500 modules, which provides Bluetooth connectivity for 8/16 bit microcontroller applications. It provides a point to point connection much like a standard serial cable. Connections are made dynamically and can be established between two eb500 modules or an eb500 module and a standard Bluetooth v1.1 device. Devices can be dynamically discovered and connected in an *ad-hoc* manner. It has an open field range of 328 feet (more than 100 meters). The power consumption goes from 5 volt to 12 volt. It supports Bluetooth version 1.1 compliant with profiles L2CAP (Logical Link Control and Adaptation Protocol), RFCOMM (protocol for RS-232 serial port emulation), SDP (Service Discovery Protocol), SPP (Serial Port Profile) [21].

5. Experimental results

5.1 Operation in Mode I

The experiments results reported in this chapter have the same parameter adjustment for $k = 3$, $\tau_{ij} = 0.5$, $\rho = 0.2$ and $\alpha = 2$. The first considered scenario is the one with no obstacles. Figure 8(a) shows the path generated by the first ant in the first iteration of the algorithm, with $\beta = 0$ and $\gamma = 1$. As it can be seen, unnecessary and excessively exploration of the map is made by the ant, since there is no obstacles in the whole area. Figure 8(b) displays how a bigger value of γ reduces considerably the excessive exploration, by adding the memory capability to the ant, but without the help of γ, in Figures 8(c) and 8(d) it can be observed how increasing the value of β is enough to make the algorithm more efficient. For $\beta = 1$, the optimal path (the diagonal between the nodes) can be found in less than three epochs, and it takes approximately 1 second to get the optimal solution, which is shown in Figure 9(a).

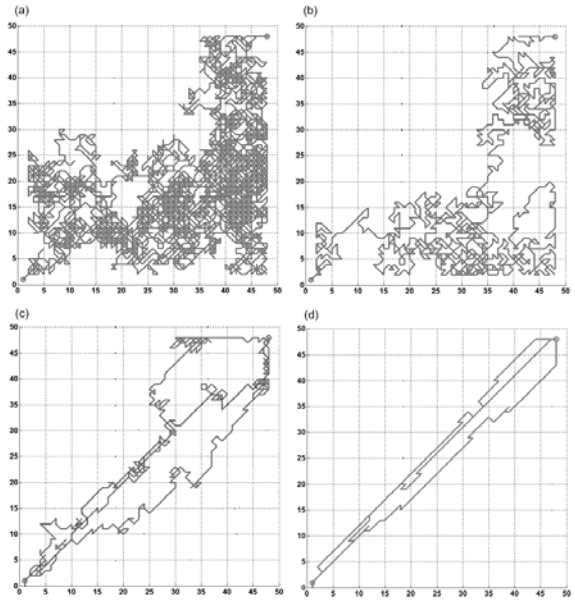

Fig. 8. (a) Route generated by the first ant in the first epoch, with $\beta = 0$ and $\gamma = 1$, (b) same situation but with $\beta = 0$ and $= 100$, (c) The routes of three ants in the first epoch with $\beta = 0.1$ and $\gamma = 1$, (d) same situation but with $\beta = 0.5$ and $\gamma = 1$.

Figure 9(b) shows the same first scenario, but now it has been added 3 new obstacles dynamically in different times t, $t+1$ and $t+2$. The ants were able to surround the blocking object and finally reach the goal.

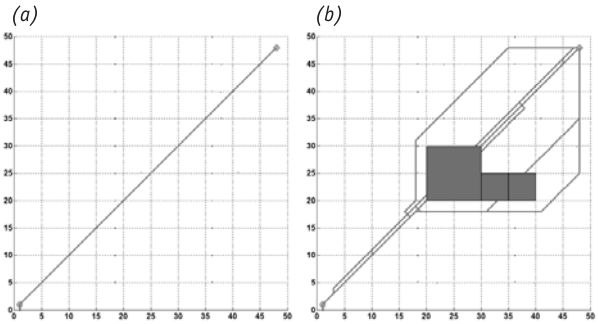

Fig. 9. (a) Optimal path found by the algorithm after one or two epochs with $\beta = 1$ and $\gamma = 1$, (b) generated routes after addition of obstacles dynamically at time t, $t+1$ and $t+2$.

At the moment, an analysis about β and γ has been made. Figure 10(a) presents a more complex map and the optimal path given by the algorithm, where the algorithm is more sensitive to variations of the parameters. Changes in the values of γ, ρ and k mostly, influence the execution time and the effectiveness of the algorithm considerably. In Figure 10(a), with k =3, $\beta = 0$ and $\gamma = 1$, adjusting $\alpha = 0.5$ and $\rho = 0.2$, the algorithm takes approximately 180 seconds and 130 epochs to find the optimal path. This causes more exploration but less exploitation. Making $\alpha = 2$ and $\rho = 0.35$, it takes around 75 seconds to find the optimal path in 20 epochs.

Figure 10(b) shows a frame from the Boe-bot model animation navigating over the path, ensuring the mobile robot dimensions fits in the free space area crossed by the route. Another example of dynamic search map can be observed in Figure 11(a), where the robot model is crossing over the alternative route given by the algorithm.

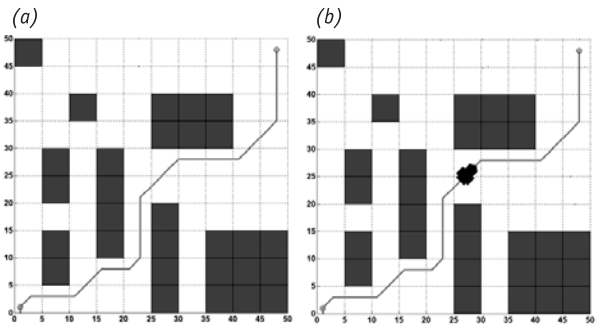

Fig. 10. (a) Optimal path found by the algorithm, (b) A frame of the Boe-bot model animation following the final route.

Figure 11(b) displays a completely different map, with major randomness in the obstacles positioning. At first sight it can be noticed the path is not the optimal one. There are some turns that are not desirable, like the one around the circle. This kind of situations are intended to be corrected by the fuzzy cost function, but the method is not fully tested yet and every different scenario has its own particular features; they may need different parameters settings in order to get optimal solutions at a reasonable time. However, these are satisfactory results that allow seeing the proposed method as an effective path planning tool and it is getting improvements. Using the STA also reduce the number of cases of suboptimal paths like the one in Figure 11(b).

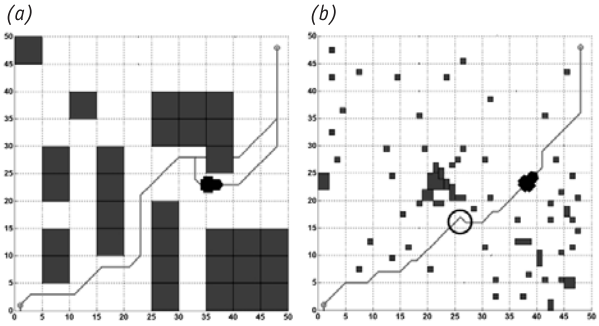

Fig. 11. (a) Alternative route given by the algorithm after a new obstacle appears at time t and the Boe-bot model following the modified path, (b) Another search area, with major randomness in the obstacles positioning. The circle indicates a not desirable movement in the path.

5.2 Operation in Mode II

The map design used for experiments with the Boe-bot is the one shown in Figures 4 and 10, reconstructed over the real search map with the square cardboards as it can be observed in Figure 7. The robot has to move along the course of the optimal path given by the algorithm, like there was a stripe to follow using sensors placed over the floor of the map.

Currently, in this research there are some particular problems in navigation to be solved. The wireless communication and the process of sending navigation commands to the robot had already been successfully established. By the other hand, in order to assure the Boe-bot will navigate following the path delivered by the ACO algorithm over the search map as accurate as possible. For a forward order, it has to move in straight line as many nodes as it was specified in the command; for a turn order is expected to move the precise angle. Then the carpeting of the laboratory area reserved for building the maps increase error.

There are still different choices of solution that can be considered, with their own advantages and disadvantages according to time and resources. An option for the turnings can be the use of a low-cost Vector 2X Compass Module from Precision Navigation, Inc., suitable for OEM applications. This 2 axis compass delivers 2-degree accuracy with 1-degree resolution. The first experiments with the compass revealed that the response time for the readings of the angle is slow for the correct robot motion.

Another topic is the sensing of a new obstacle over the route, like the simulation case shown in Figure 11(a). For this situation, a complete cardboard cube will be placed manually obstructing the robot's path. The sensing may be done with ultrasound or infrared light sensors.

6. Conclusions

The ACO-MH proposed method seems to be a promising path planning system for autonomous mobile robot navigation since the given solutions are not only paths, but the optimal ones. It is a very fast algorithm since it is able to find an optimal path with few cost function evaluations in many cases, regarding addition of β and γ. At present time, the method has been used with a real robot (Boe-bot), and the alternatives for precise navigation are still under evaluation and testing process. It also has the ability of finding optimal paths in dynamic search spaces, considering the physical size of the robot. On the other hand, applying the STA makes a more precise difference between costs of paths with same length but different effort for navigation, causing major preference for straighter paths.

AUTHORS

Miguel Porta-Garcia, **Oscar Montiel**, **Roberto Sepulveda** - CITEDI-IPN. Ave. del Parque 1310 Mesa de Otay, Tijuana B.C. 22510. E-mails: mporta@citedi.mx, o.montiel@ieee.org, r.sepulveda@ieee.org.

References

[1] Víctor Fernando Muñoz Martínez, *Planificación de Trayectorias para Robots Móviles*, Doctoral thesis presented on 5th July, 1995. Available at: http://webpersonal.uma.es/~VFMM/

[2] M. Dorigo, M. Birattari, T. Stützle, "Ant Colony Optimization", *IEEE Computational Intelligence Magazine*, November 2006, pp. 28-39.

[3] A.P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, Wiley, J England, 2005.

[4] A.R. Diéguez, R. Sanz, J.L. Fernández, "A global motion planner that learns from experience for autonomous mobile robots", *Robotics and Computer-Integrated Manufacturing*, vol. 25, issue 5, 2007, Elsevier, pp. 544-552.

[5] Ashraf Elnagar, Leena Lulu, "A visual tool for computer supported learning: The robot motion planning example", *Computers & Education*, vol. 49, no. 2, 2007, Elsevier, pp. 269-283.

[6] Shuzhi Sam Ge, Xue-Cheng Lai, Abdullah Al Mamun, "Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints", *Robotics and Autonomous Systems*, vol. 55, issue 7, 2007, Elsevier, pp. 513-526.

[7] K. Gopalakrishnan, S. Ramakrishnan, *Optimal Path Planning of Mobile Robot with Multiple Targets Using Ant Colony Optimization*. Smart Systems Engineering, 2006, New York, pp. 25-30.

[8] L. Zhishuo, C. Yueting, "Sweep based Multiple Ant Colonies Algorithm for Capacitated Vehicle Routing Problem", *IEEE International Conference on e-Business Engineering (ICEBE'05)*, 2005, pp. 387-394.

[9] H. Chen, Z. Xu, *Path Planning Based on a New Genetic Algorithm*, International Conference on Neural Networks and Brain, 2005. Volume 2, 13-15 Oct. 2005, pp. 788-792. Digital Object Identifier 10.1109/ ICNNB. 2005.1614743

[10] M. Gemeinder, M. Gerke, "An Active Search Algorithm Extending GA Based Path Planning for Mobile Robot Systems". In: *Soft Computing and Industry - Recent Applications*, Roy R. *et al.*, (Eds.) Berlin Heidelberg New York: Springer Verlag, 2002, pp. 589-596.

[11] M. Tarokh, *Path planning of rovers using fuzzy logic and genetic algorithm, World Automation Conf. ISORA-026*, Hawaii, 2000, pp. 1-7.

[12] S. Cardenas O. Castillo, L. Aguilar L, J. Garibaldi, "Intelligent planning and control of robots using genetic algorithms and fuzzy logic", *International Conference on Artificial Intelligence (IC-AI '05)*, 2005, pp. 412-418.

[13] O. Castillo, L. Trujillo, "Autonomous mobile robot path planning optimization using multiple objective genetic algorithms", *International Conference on Artificial Intelligence (IC-AI '04)*, 2004, pp. 71-76.

[14] J. Garibaldi, A. Barreras A., O. Castillo, "Intelligent Control and Planning of Autonomous Mobile Robots using Fuzzy Logic and Genetic Algorithms". In: *Hybrid Intelligent Systems* (Edited by O. Castillo *et al.*), Springer-Verlag, 2007, pp. 255-265.

[15] M. Tarokh, "Genetic Path Planning with Fuzzy Logic Adaptation for Rovers Traversing Rough Terrain". In: *Hybrid Intelligent Systems* (Edited by Castillo *et al.*), Springer-Verlag, 2007, pp. 215-228.

[16] M. Mohamad, W. Dunningan, "Ant Colony Robot Motion Planning", *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, vol. 1, IEEE, 2005, pp. 213-216.

[17] W. Ye, D. Ma, H. Fan, "Path Planning for Space Robot Based on The Self-adaptive Ant Colony Algorithm". *1st International Symposium on Systems and Control in Aerospace and Astronautics*, 19-21 January, 2006 (ISSCAA), pp. 4.

[18] M. Dorigo, T. Stützle, *Ant Colony Optimization*, Bradford, Cambridge, Massachusetts, 2004.

[19] E. Gómez Ramírez E., "Simple Tuning of Fuzzy Controllers". In: *The International Conference on Fuzzy Systems, Neural Networks and Genetic Algorithms (FNG 2005)*, Tijuana, México, 2005, pp. 49-64.

[20] *Parallax Inc. 2006, Basic Stamp Syntax and Reference Manual*. Version 2.2. Available at: http://www.parallax.com/dl/docs/prod/stamps/web-BSM-v2.2.pdf

[21] *A7 Engineering, EmbeddedBlue™ 500 User manual*. Available at: http://www.a7eng.com/products/embeddedblue/downloads/eb500UserManual.pdf