Engineering Applications of Artificial Intelligence 22 (2009) 431–441

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Collective robotic search using hybrid techniques: Fuzzy logic and swarm intelligence inspired by nature

Ganesh K. Venayagamoorthy *, Lisa L. Grant, Sheetal Doctor

Real-Time Power and Intelligent Systems Laboratory, Missouri University of Science and Technology, Electrical and Computer Engineering, 132 Emerson Electric Co. Hall, Rolla, MO 65409, USA

ARTICLE INFO

Article history:
Received 24 April 2005
Received in revised form
7 August 2007
Accepted 4 October 2008
Available online 26 November 2008

Keywords:
Collective robotic search
Fuzzy logic
Fuzzy swarm
Greedy search
Particle swarm optimization
Swarm fuzzy
Swarm intelligence

ABSTRACT

This paper presents two new strategies for navigation of a swarm of robots for target/mission focused applications including landmine detection and firefighting. The first method presents an embedded fuzzy logic approach in the particle swarm optimization (PSO) algorithm robots and the second method presents a swarm of fuzzy logic controllers, one on each robot. The framework of both strategies has been inspired by natural swarms such as the school of fish or the flock of birds. In addition to the target search using the above methods, a hierarchy for the coordination of a swarm of robots has been proposed. The robustness of both strategies is evaluated for failures or loss in swarm members. Results are presented with both strategies and comparisons of their performance are carried out against a greedy search algorithm.

1. Introduction

In recent years, a lot of research is focused on collective robotic applications (Van Dyke Paranuk, 2005; Zhang and Wunsch, 2003a, b). Robots have been used for numerous applications where human intervention is not feasible. The building of intelligent robots with a number of sensors for various parameters is expensive. In applications such as radioactivity detection, firefighting and landmine detection, the robots need to be dispensable. A large number of robots allow for redundancy and increase the robustness of the swarm. An expensive robot may be able to achieve the task but its failure can prove to be costly and dangerous in mission critical applications. By building a swarm of robots with elementary features, the same task can be achieved for a lower cost and increased reliability. Multiple robots used for search applications also have an advantage of larger coverage of the search space and its simplicity of implementation.

Planning the motion of these robots in a search space differs from application to application. Motion planning of the robots differs in obstacle-laden environments (Foux et al., 1993; Kim and Butler, 1995). The navigation of the robots from the start to the destination in an unknown environment that requires the use of sensors and a method to explore the area with the inputs of these sensors (Kim and Butler, 1995). Fuzzy logic has been a widely used method for these applications (Kim and Butler, 1995; Zhang and Wunsch, 2003a, b; Dadios and Maravilla Jr., 2002). Other methods explored in literature include navigation based on intelligent data carrier systems (Konishi et al., 1999) or geometric algorithms (Sharir, 1989), etc. In the case of multiple robot navigation, basic algorithms like particle swarm optimization (PSO), evolutionary strategies, genetic algorithms, etc, have been explored for this purpose.

Swarm intelligence is a method derived from the study of a flock of birds, school of fish, a colony of ants or a swarm of bees. These methods take into account the natural behavior of social animals that carry out a certain task collectively. Fuzzy logic has been used in a number of ways for robotic navigation. This paper explores the performance of a fuzzy system by introducing the social interaction exhibited in swarm intelligence. This paper presents the integration of fuzzy logic with swarm intelligence specifically the PSO.

The integration of fuzzy logic and swarm intelligence has been carried using two different approaches. The first method is based on the canonical PSO algorithm that uses a fuzzy term to replace a part of the PSO dynamics. This method is referred to as the *fuzzified swarm of robots* in this paper. The second method is based on a swarm of robots that use fuzzy logic controllers, referred to as

* Corresponding author. Tel.: +1 573 3416641; fax: +1 573 3414532.
E-mail addresses: gkumar@ieee.org (G.K. Venayagamoorthy), lisagrant@ieee.org (L.L. Grant), skdkv6@mst.edu (S. Doctor).

0952-1976/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2008.10.002

swarm-fuzzy controllers in this paper. In this method, the robots use fuzzy logic and are guided towards the desired target locations by trying to move towards the robot with the best fitness in the swarm.

The rest of the paper is outlined as follows: Section 2 describes the collective robotic search architecture and the search problem considered in this paper. Sections 3 and 4 describe PSO and fuzzy logic, respectively. Section 5 describes the experimental setup for the implementation of the collective robotic search of Section 2. Sections 6 and 7 describe the implementation of the two different approaches. And finally, Section 8 presents the results for the collective search using the hybrid swarm-fuzzy approaches compared with a greedy search.

2. Collective robotic search

There are various applications as mentioned in the previous section, which require the use of multiple robots or a team of robots. This paper proposes a three-tier architecture for the implementation of a collective robotic search as shown in Fig. 1. The aim of the robotic search application is to identify target locations. The nature of the targets varies in different applications. For the study in this paper, the targets are taken to be stationary light sources. The robots are assumed to have light sensors and they sense the total light intensity at their current locations irrespective of the location and number of light sources.

The robots are randomly deployed in the problem search area. Since collective robotic search is targeted to applications where human intervention is not possible, the random deployment can be carried out for example by dropping the robots to the search area of interest by an aircraft.

As seen in Fig. 1, the search area is divided into a number of swarms and the operation of each swarm is independent of each other. In environments with extreme topology/terrain, this division of the search area into multiple swarms makes sure that the entire search space is covered and independently the targets are searched out, thus performing a fast, efficient and parallel search. Not all robots may have access to the entire search space depending upon their initial position of deployment and the severity of terrain, which they will have to traverse, such as being restricted to a certain area by a natural boundary such as a body of water or mountain range. In addition, with divided search space, the communication range of the robots does not have to be too long. Thus, transmission power can be minimized. The concepts of the neighborhood level and global level are explained in the next section with respect to PSO.

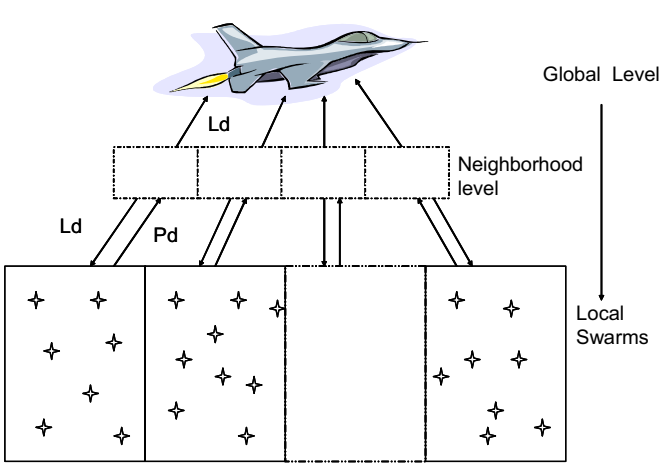

Fig. 1. A three-tier architecture for the collective robotic search. Robots are randomly deployed from an airplane.

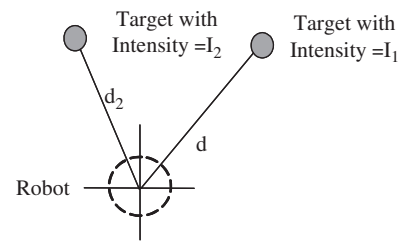

Fig. 2. Graphical representation of sensor reading at the robot.

At the ground/local level, all the robots can pick up sensor readings from all the targets within a certain range irrespective of the location of targets. This means that a robot can pick up a sensor reading from a target lying in another neighborhood swarm. Targets are assumed to be stationary for this application. Fig. 2 shows a graphical representation of the sensor readings. Eq. (1) shows the total sensor reading at the robot as considered for this paper. The sensor readings of each robot are used to locate the target. In the first few iterations of the search, exploration of the swarm area is carried out and then the search is focused gradually towards locating a primary target and readings from secondary targets fade away or shadowed:

$$\text{Intensity reading at the sensor node} = I_1/d_1^2 + I_2/d_2^2 \tag{1}$$

3. Particle swarm optimization

PSO is relatively a new concept reported by Kennedy and Eberhart (2001), in 1995. PSO has been applied for target tracing by autonomous communicating bodies (Gesu et al., 2000). A problem space is initialized with a population of random solutions in which it searches for the optimum over a number of generations/iterations and reproduction is based on prior generations. The concept of PSO is that each particle randomly searches through the problem space by updating itself with its own memory and the social information gathered from other particles. In this paper, the PSO particles are referred to as robots and the local version of the PSO algorithm is considered in the context of this application (Kennedy, 1999).

Within a defined sensing area or a swarm as in Fig. 1, there is a population of robots. Each robot is randomized with a velocity and 'flown' in the problem space. They have memory and they are able to keep track of the position that resulted in the highest sensor readings. This position is referred to as 'P_{best}'. Thus, each robot has a 'P_{best}'. The best of all these P_{best} values is defined as the local best position 'L_{best}' with respect to the target. The velocities and positions of these robots are constantly updated until they have all converged at the target location. Thus, in terms of memory requirements, PSO requires only 2 values (other than the velocity and position from the previous iteration), P_{best} and L_{best}.

Fig. 3 gives the vector representation of the PSO in a two-dimensional search space. The stars represent the particles/robots and the circle represents the target. In Fig. 3, vectors V_{pd} and V_{gd} represent the effect of P_{best} and L_{best} on the robots, respectively. The basic PSO velocity and position update equations are given by (2) and (3) respectively. To correlate Fig. 3 and velocity equation, (2), the term V_{pd} in the figure represents '$c_1 \times \text{rand} \times (P_{\text{best}} - P_{\text{cur}})$' in (2) and the term V_{gd} in the figure represents '$c_2 \times \text{rand} \times (L_{\text{best}} - P_{\text{cur}})$' in (2). The constants w, c_1 and c_2 in (2) are called the quality factors:

$$V_{\text{new}} = w \times V_{\text{cur}} + c_1 \times \text{rand} \times (P_{\text{best}} - P_{\text{cur}}) + c_2 \times \text{rand} \times (L_{\text{best}} - P_{\text{cur}}) \tag{2}$$

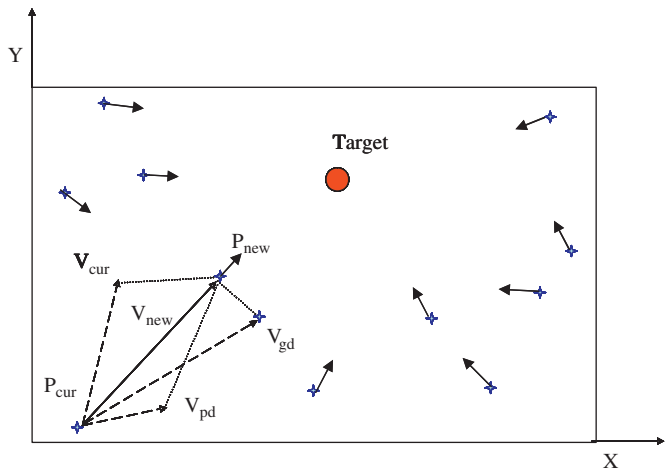

Fig. 3. Vector representation of PSO.

$$P_{new} = P_{cur} + V_{new} \quad (3)$$

where V_{new} is the new velocity calculated for each robot, V_{cur} the velocity of the robot from the previous iteration, P_{new} the new position calculated for each robot, P_{cur} the position of the robot from the previous iteration, w the inertia weight, c_1 and c_2 the cognitive and social acceleration constants, respectively and 'rand' generates a uniform random value in the range [0, 1]

The robots respond to the factors 'P_{best}' and 'L_{best}' in order to find a new position closer to the target. The basic steps for the implementation of PSO involve the following:

(i) Define the sensing area with its boundaries. Initialize an array of robots with random positions and velocities. These random positions are initially assigned to be the P_{best}. Also initialize the target(s) position(s) randomly in the sensing area (for the simulation case).
(ii) Evaluate the fitness function (based on Euclidean distance of the robot from the target(s) based on sensory readings). Select the L_{best} from all P_{best}.
(iii) Compute the new velocities and positions of the robots using (2) and (3) above, respectively.
(iv) Check if the robots' positions are within the problem space. Also check if the velocity exceeds the predefined limits. If they are not then the velocity is set to the maximum velocity (V_{max}) and the new position is set to its previous best position.
(v) Calculate the new fitness function for all the robots' new positions. Determine the new P_{best}. Compare with the previous P_{best} and update the value with the new one if necessary.
(vi) Calculate the new local best position L_{best} among all the new best positions, P_{best}. Compare with the previous best and update the local best before the next iteration.
(vii) The procedure is repeated from step (iii), until all the robots converge at the target(s).

4. Fuzzy logic controller

From the inception of fuzzy logic by Zadeh in the 1960s, its foundations have grown stronger with the years (Zadeh, 1988). There have been numerous applications using fuzzy logic. Conventional control techniques that have been used over the ages rely on linear models. They do not accurately model real world systems but are only approximations. Most real world systems are far too complicated for linear approximators and they require nonlinear techniques.

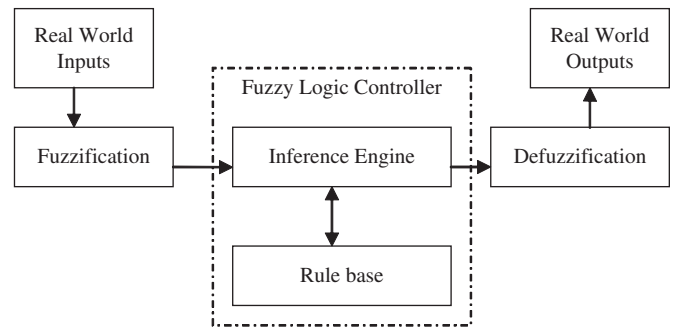

Fig. 4. Block diagram of a fuzzy system.

Fuzzy logic techniques are primarily applied to systems that cannot function well with the conventional analytical model-based control techniques or where mathematical models are difficult to obtain. In fuzzy systems, quick control strategy is needed where imprecise and qualitative definition of the actions is available instead of precise data sets (Sayeed et al., 1994).

Fuzzy logic introduces a realistic situation into a system. It differs from binary logic by allowing the addition of a degree of truthfulness or falsehood into the system. A fuzzy logic system contains linguistic variables that define the parameters of the real world. It has membership functions that define the degree of the input and output variables. And lastly, it has a knowledge base or a set of rules that define the input–output relationships. These rules are developed by heuristics and the performance of the system many a times depends on the expert defining the rules and the membership functions.

Fig. 4 shows the block diagram of a fuzzy system. The fuzzification process is an interface between the real world parameters and the fuzzy controller. It performs a mapping that transfers the input data into linguistic variables and the range of these input data forms the fuzzy sets. The inference engine uses the rules defined and it develops fuzzy outputs from the inputs. The defuzzification is the reverse process of the fuzzification. It maps the fuzzy controller output variables to the real world values that can be applied to the robots in this paper.

This paper uses fuzzy logic in two different ways for the collective robotic search applications. In the first method, it embeds a fuzzy term in the PSO equation and in the second method, a swarm of fuzzy logic controllers is developed.

5. Implementation

Fig. 5 shows a graphical representation of the search space. For this application, the search space considered is divided into swarms or local neighborhoods and each neighborhood is treated as an independent swarm. It is assumed that the robots within each neighborhood do not communicate with those in other neighborhoods.

For simulation purposes, a sensing area (200×200 units) is divided into four local neighborhoods/swarms according to the architecture described by Fig. 1. Fig. 5 shows the distribution of the sensors (represented by the asterisk) and targets (represented by the circles) in the sensing area. The sensing area is the same for both the fuzzy swarm and swarm-fuzzy-based robot navigation.

There is no clustering algorithm that is used for the division but instead the area is fixed and the robots deployed within each neighborhood constitute a local swarm. The total number of robots randomly deployed is 40 and the number of robots per swarm could be different.

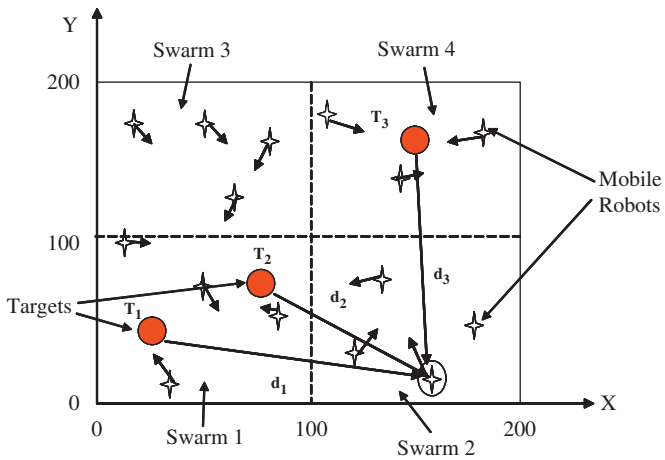

Fig. 5. Randomized robots and targets in the search space divided into neighborhoods and light intensity read by the robot.

For the simulation studies, the targets are chosen to be stationary light sources of different intensities. The robots are assumed to read a single intensity value. Fig. 5 also shows the sensor reading at a robot under the influence of all the three targets in the entire search space, irrespective of the location the target(s) fall into. Eq. (4) gives the value of intensity read by a robot. This value is used as the fitness function in both approaches: fuzzy swarm and a swarm of fuzzy controllers:

$$\text{Intensity read bya robot at } P_n = (I_1/d_1^2) + (I_2/d_2^2) + (I_3/d_3^2) \tag{4}$$

where I_1, I_2 and I_3 are absolute intensities of the light sources T_1, T_2 and T_3, respectively, and d_1, d_2 and d_3 are Euclidean distances of robot at P_n from the light sources T_1, T_2 and T_3, respectively.

6. Fuzzy swarm controllers

PSO is a stochastic search algorithm, as evident from the velocity update equation's dependence on random terms; $c_1 \times \text{rand} \times (P_{\text{best}} - P_{\text{cur}})$ and $c_2 \times \text{rand} \times (L_{\text{best}} - P_{\text{cur}})$. The randomness from the two terms helps the robots in exploring the search space, but it can also cause the robots to diverge at times and as a result convergence of the robots cannot be guaranteed.

Fuzzy logic has been explored in this application to replace the randomness of the velocity equation with a target-oriented motion. A fuzzy system is designed to modify PSO's equation (2). The inference engine used for the fuzzy system is Mamdani's Product inference engine and the defuzzification is based on center average. 'P_d' and 'L_d' are the inputs to the system. P_d has an X and Y component namely Pdx and Pdy. Likewise, L_d has an X and Y component namely $L\,dx$ and $L\,dy$. There are four input variables to the fuzzy system.

- The difference of the X coordinates of the current position of the robot, P_{cur} from the robot having the best position, L_{best} in the swarm given by '$L\,dx$'.
- The difference of the Y coordinates of the current position of the robot, P_{cur} from the robot having the best position, L_{best} in the swarm given by '$L\,dy$'.
- The difference of the X coordinates of the current position of the robot, P_{cur} from the robot's own previous best position, P_{best} in the swarm given by 'Pdx'.
- The difference of the Y coordinates of the current position of the robot, P_{cur} from the robot's own previous best position, P_{best} in the swarm given by 'Pdy'.

There are two output variables

- The change in velocity along the X-axis 'ΔVx'.
- The change in velocity along the Y-axis 'ΔVy'.

The membership functions for all the input variables defined as negative large ($-L$), negative medium ($-M$), negative small ($-S$), zero (Z), positive small (S), positive medium (M), large (L). The output variables have seven membership functions. They are the same as the input membership functions except for zero (Z). The membership function is denoted by very small (VS) instead.

The structure of the fuzzy system still adheres to the structure of PSO. As described above, the inputs to the fuzzy system are 'P_d' and 'L_d', therefore still maintaining the effect of the social and cognitive components of PSO. Therefore, the terms "$c_1 \times \text{rand} \times (P_{\text{best}} - P_{\text{cur}})$" and "$c_2 \times \text{rand} \times (L_{\text{best}} - P_{\text{cur}})$" are now represented by (4) and (5) for X- and Y-axis, respectively.

Table 1 gives the general set of rules used by this fuzzy system. The rules for the X and Y coordinates are independent of each other but are the same. Eqs. (5) and (6) show each output as a function of two of the four input variables:

$$\Delta Vx = f(P\,dx,\ L\,dx) \tag{5}$$

$$\Delta Vy = f(P\,dy,\ L\,dy) \tag{6}$$

As seen in Table 1, in the case where L_d is either negative large (row 1) or positive large (row 7) the output of the fuzzy system, 'Δv', follows the L_d value instead of a relationship between 'P_d' and 'L_d'. Thus, there are 37 rules each for calculating 'ΔV_x' and 'ΔV_y'. Therefore, there are a total of 74 rules for this system. From previous studies, it has been seen that the PSO algorithm functions better when the effect of the social component is greater than that of the cognitive component. In other words, the value of c_2 is greater than that of c_1 (Doctor et al., 2004).

The functioning of the system after adding the fuzzy logic component has been explored with different variations. These variations have been described below

Case 1—*Canonical implementation*: In this case, (2) is now replaced by

$$V_{\text{new}} = w \times V_{\text{cur}} + \Delta v \tag{7}$$

where

$$\begin{aligned} \Delta v &= f(P_d, L_d) \\ P_d &= P_{\text{best}} - P_{\text{cur}} \\ L_d &= L_{\text{best}} - P_{\text{cur}} \\ P_d &= [P\,dx, P\,dy] \\ L_d &= [L\,dx, L\,dy] \\ \Delta v &= [\Delta Vx, \Delta Vy] \end{aligned} \tag{8}$$

Table 1
Rule set for fuzzy swarm controllers.

$L\,dx/L\,dy$	$P\,dx/P\,dy$						
	$-L$	$-M$	$-S$	Z	S	M	L
$-L$							
$-M$	$-L$	$-M$	$-M$	$-M$	$-S$	$-M$	$-L$
$-S$	$-M$	$-S$	$-S$	VS	$-S$	$-S$	M
Z	$-S$	$-S$	VS	VS	VS	S	S
S	$-M$	$-S$	S	VS	S	S	M
M	$-L$	M	S	M	S	M	L
L	L	L	L	L	L	L	L

Case 2—*Adding a uniform random term to the fuzzy output*: In this case, (7) is now replaced by

$$V_{\text{new}} = w \times V_{\text{cur}} + rand \times \Delta v \tag{9}$$

Case 3—*Adding a Gaussian random term to the fuzzy output*: In this case, (7) is now replaced by

$$V_{\text{new}} = w \times V_{\text{cur}} + \text{gaussrand} \times \Delta v \tag{10}$$

Here, the 'gaussrand' function generates a random number with a Gaussian distribution with a mean of 0.5 and values ranging between 0 and 1.

Case 4—*Varying the value of w in velocity equation:* In this case, (7) is now replaced by

$$V_{\text{new}} = wi \times V_{\text{cur}} + \text{rand} \times \Delta v \tag{11}$$

Two methods of varying the value of 'w_i' have been explored and are given by (12) and (13):

(a) Decreasing 'w_i'

$$w_i = 0.9 - \frac{(0.5 \times i)}{\text{max\_}i} \tag{12}$$

Here, i is the iteration number and max_i is the iteration limit set for the search process. Eq. (11) continuously reduces the value of w_i starting from a value less 0.9 depending on the value of max_i'.

(b) Random 'w_i'

$$w_i = 0.5 + \frac{\text{rand}}{2} \tag{13}$$

Eq. (13) generates a random value between 0.5 and 1.0.

Case 5—*Adding mutation*: The robot's fitness value is tracked through the process and if this value does not change over a predefined number of iterations, it is assumed to be stuck in a local minimum. In this case, a perturbation is added to a robot's position if it gets stuck in a local minimum. The probability of mutating a certain robots position has been varied and the results are presented for three different mutation probabilities of 0.2, 0.5, 0.7 (cases 5a, 5b and 5c, respectively).

$$P_i = P_i \pm (\text{rand} \times 20) \tag{14}$$

As seen in (14), the ith robot is perturbed by adding/subtracting a value no greater than 20 units (20% of the search area's width/length in each swarm) to its current position.

Case 6—*Fault tolerance study*: In order to test the robustness of the search strategy, a 20% loss in the number of robots in the swarm is inflicted intentionally during the search process. This study is performed on case 1 implementation. Therefore the exact number of robots lost for example, for an initial population of 3–7 is 1 robot; for a population of 8–12, 2 robots are lost and so on. In the neighborhoods 2 and 3 in Fig. 5, where there are no targets, a loss makes no difference. For this case study, the 20% loss is inflicted in two steps in neighborhood 1. At iteration number 5, a 10% loss is inflicted and then a further 10% at iteration 10. Whereas, for neighborhood 4, the 20% loss is inflicted after 5 iterations of search.

Case 7—*Greedy search study*: A greedy search utilizing the same environment and basic system parameters is presented for comparison to the fuzzy swarm method. Greedy search is explained in detail in Section 8.

The results for all the above case studies are all provided in Section 8.

7. Swarm of fuzzy controllers

A swarm of fuzzy logic controllers have been used for the target location problem for guiding the robots to the target. The identical fuzzy controller on each robot has been developed based on heuristics. That is, the membership functions and the rules are developed based on heuristics. Fig. 4 shows the space in which the robots are randomly deployed.

Eq. (15) is implemented by the fuzzy systems to find the robots' now positions

$$P_i = (X_i \pm \Delta X_i, Y_i \pm \Delta Y_i) \tag{15}$$

where

$$\Delta X_i = f(I_i, L\,\text{d}x)$$
$$\Delta Y_i = f(I_i, L\,\text{d}y) \tag{16}$$

where $L\,\text{d}x = L_{\text{best}}x - Px$ and $L\,\text{d}y = L_{\text{best}}y - Py$. Px and Py are the X–Y coordinates of the robot's current position and $L_{\text{best}}x$ and $L_{\text{best}}y$ are the X–Y coordinates of the L_{best} of the swarm. This L_{best} is calculated in the same way as in the fuzzy swarm implementation discussed above. If $L\,\text{d}x$ is negative then the X coordinate in (15) becomes ($X_i - \Delta X_i$) and if it is positive then it becomes ($X_i + \Delta X_i$). The same logic is applied to the Y coordinates as well. The robots in this approach have been considered to read a single intensity values (I_i) as shown in Fig. 6. The inference engine used for this simulation is Mamdani's product inference engine and the defuzzification is based on center average.

There are three input variables to each robot's fuzzy controller:

- The intensity reading at the sensor on each robot (I_i) with zero (Z), small (S), medium (M) and large (L) membership functions.
- The difference of the X coordinates of the current position of the robot from the robot having the best position in the swarm given by '$L\,\text{d}x$' with zero (Z), very small (VS), small (S), medium (M) and large (L) membership functions.
- The difference of the Y coordinates of the current position of the robot from the robot having the best position in the swarm given by '$L\,\text{d}y$' with zero (Z), very small (VS), small (S), medium (M) and large (L) membership functions.

There are two output variables from each fuzzy logic controller

- The displacement amount ΔX_i to be added to the robot's X-axis's current position. ΔX_i has very small (VS), small (S), medium (M) and large (L) membership functions.
- The displacement amount ΔY_i to be added to the robot's Y-axis's current position. ΔY_i has very small (VS), small (S), medium (M) and large (L) membership functions.

There are two different sets of the rules used as shown in Fig. 6. The first set of rules referred to as the *coarse set*, makes the robots

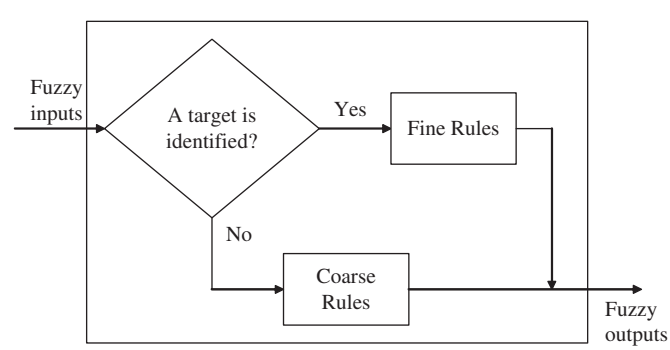

Fig. 6. Rule base switching logic.

Table 2
Coarse rule set for the swarm of fuzzy logic controllers.

Intensity (I)	Ldx/Ldy				
	Z	VS	S	M	L
Z					
S	S	M	M	L	L
M	S	S	M	M	M
L	VS	VS	S	S	M

Table 3
Fine rule set for the swarm of fuzzy logic controllers.

Intensity (I)	Ldx/Ldy				
	Z	VS	S	M	L
Z	VS	VS	S	M	M
S	VS	VS	S	S	M
M	VS	VS	VS	S	M
L	VS	VS	VS	S	S

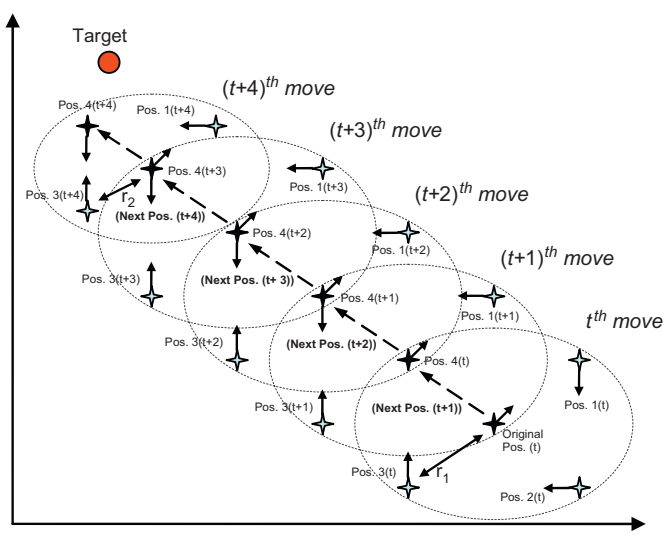

Fig. 7. A robot scanning four positions to take the next move in the direction of highest fitness using the greedy ordinal method.

move in larger steps. This helps in exploration of the search area and once a target has been identified, the second set of rules referred to as the *fine rule* set is used to make the robots move towards the target in smaller steps. This ensures a more target-oriented motion. Tables 2 and 3 show the coarse and fine rule sets, respectively. Each set has a total of 40 rules. Twenty rules each for X- and Y- axis.

Eqs. (17) and (18) give an example of the IF-THEN statement used in the coarse and fine rule sets, respectively. As can be seen, the antecedent of both (17) and (18) are the same but the consequents are different. As can be seen the coarse rule set gives a larger step as the output as compared to the fine rule set, for the same input conditions.

IF 'Intensity' is small and '$L\,dx$' is large THEN ΔX_i is Large (17)

IF 'Intensity' is small and '$L\,dx$' is large THEN ΔX_i is Medium (18)

Three case studies have been carried out as described in the following:

Case 1—*Canonical implementation*: This is a direct implementation of the equations described above in (15) and (16).

Case 2—*Fault tolerance study*: In order to test the robustness of the search strategy, a 20% loss in the number of robots in the swarm is inflicted intentionally during the search process as in the case of the fuzzy swarm controllers described in Section 6.

Case 3—*Greedy search study*: A greedy search utilizing the same environment and basic system parameters is presented for comparison to the swarm of fuzzy method. Greedy search is explained in detail in Section 8.

8. Results

8.1. Greedy swarm

For a comparison with traditional search techniques, the results for the fuzzy methods are compared with greedy search.

Table 4
Results for all the cases of fuzzy swarm controllers.

Case no.	Time (s)	Iterations	Convergence (%)
1	301.72	318.88	94
2	342.83	363.58	95
3	336.98	347.00	91
4a	306.18	322.34	86
4b	391.42	392.69	72
5a	436.17	491.43	29
5b	341.79	386.56	87
5c	300.52	334.38	94
6	262.53	311.25	93
7	622.81	662.56	87

Table 5
Number of robots in each swarm for failure cases.

No.	Swarm 1	Swarm 2	Swarm 3	Swarm 4	Failure to identify targets
1	2	16	8	14	Swarm 1 (2)
2	12	11	11	6	Swarm 1 (2)
3	12	7	12	9	Swarm 1 (1)
4	12	9	16	3	Swarm 4 (1)
5	4	11	14	11	Swarm 1 (1)
6	6	8	13	13	Swarm 1 (1)
7	5	12	8	15	Swarm 1 (1)

Table 6
Number of robots in each swarm for unexpected successes.

No.	Swarm 1	Swarm 2	Swarm 3	Swarm 4
1	5	14	13	8
2	5	8	13	14
3	6	15	12	7
4	6	14	10	10
5	6	8	9	17
6	6	12	11	11
7	14	15	7	4

Table 7
Results for all the cases in swarm of fuzzy controllers.

Case study	Time (s)	Iterations	Convergence (%)
1	214.04	256.14	94
2	222.64	305.17	92
7	622.81	662.56	87

Table 8
Number of robots in each swarm for failure cases.

No.	Swarm 1	Swarm 2	Swarm 3	Swarm 4	Failure to identify targets
1	4	15	11	10	Swarm 1 (2)
2	7	9	15	9	Swarm 1 (1)
3	5	9	14	12	Swarm 1 (1)
4	2	16	8	14	Swarm 1 (1)
5	4	11	12	13	Swarm 1 (1)
6	6	12	8	14	Swarm 1 (2)
7	7	12	14	7	Swarm 1 (2)
8	8	9	13	10	Swarm 1 (2)

Table 9
Number of robots in each swarm for unexpected successes.

No.	Swarm 1	Swarm 2	Swarm 3	Swarm 4
1	5	17	8	10
2	5	14	9	12
3	5	9	9	17

In greedy search, the robot chooses its path based solely on which positions will orient it closest to its goal. In the greedy search method, each robot acts as an individual agent. In a search utilizing multiple greedy robots, no communication occurs, so each robot must locate the target on its own. Other similar strategies have been proposed to produce agent convergence where global information and communication are not required but global group behavior is still observed (Tang and Ozguner, 2005; Cortes et al., 2006; Jadbabaie et al., 2003; Lin et al., 2003). To find its next position, the robot maps out a four radial congruent points represented by H_1 through H_4 from its current position (P_x, P_y). The search points are oriented in the ordinal (NE, SE, SW and NW) directions. As the robot visits each of the search points, it scans for the fitness at that position and ultimately chooses the search point with the best fitness for its next position, then the process repeats. The robot is basically used as a mobile sensor.

An initial velocity of 4 units is used for the greedy search. In order for the greedy algorithm to have a target fitness comparable to the fuzzy method, the velocity variable slowly decreases as the relative intensity increases, which can be seen in (19). This

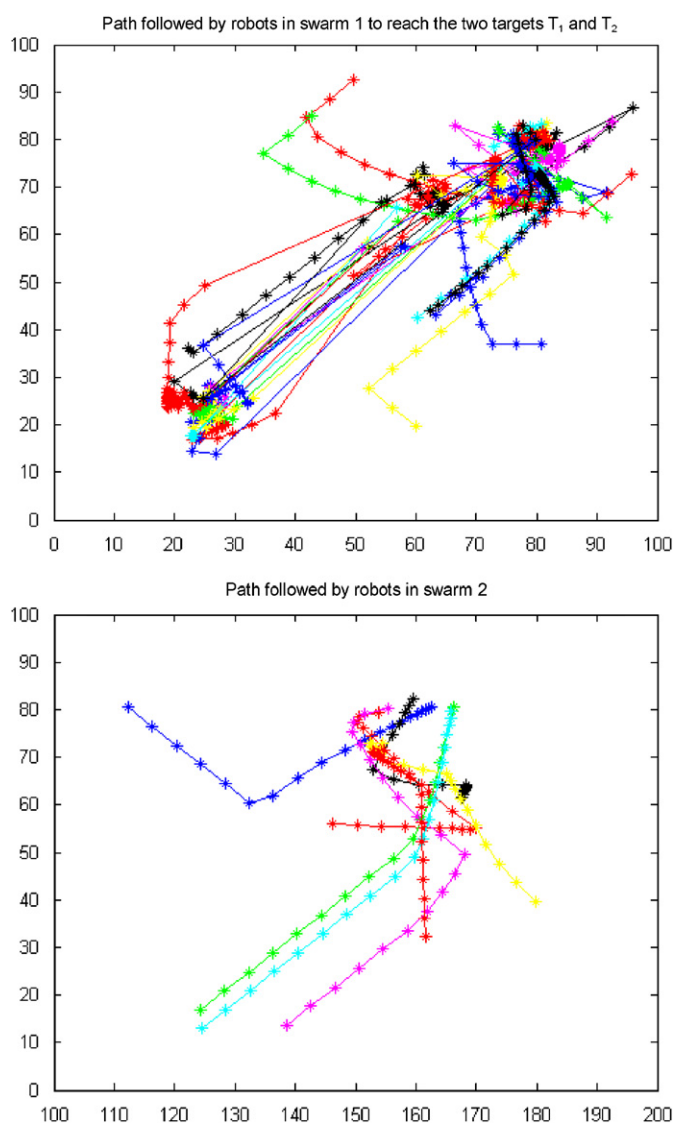

Fig. 8. Graphs for the path of the robots in swarms 1 and 2 in the fuzzy swarm approach.

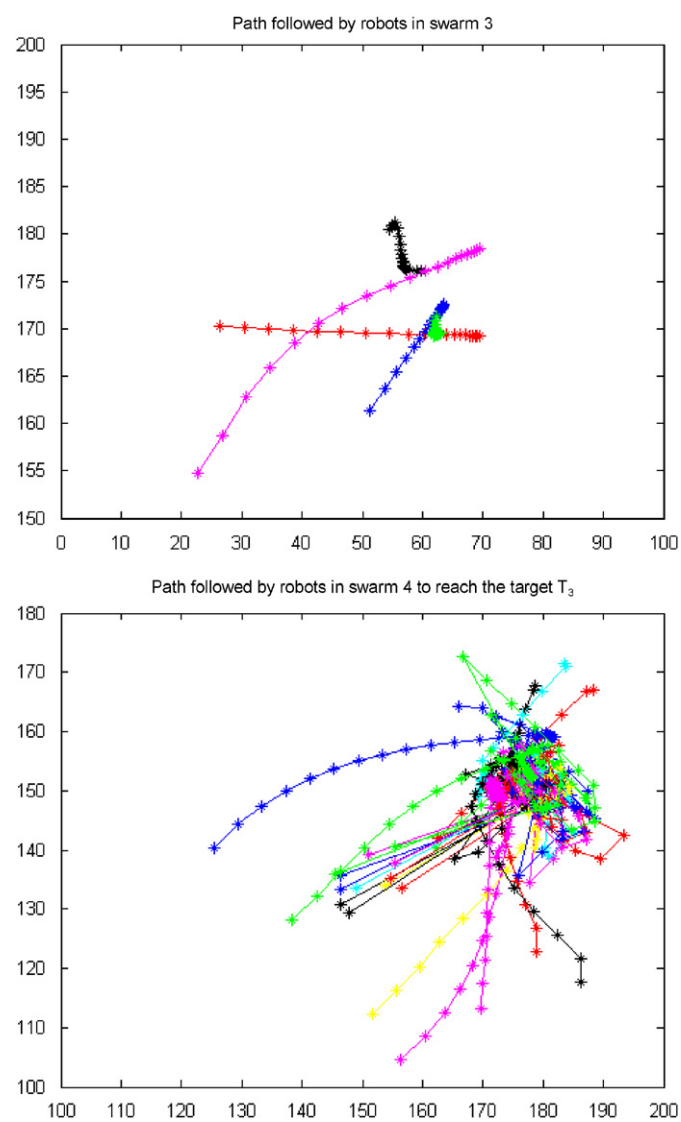

Fig. 9. Graphs for the path of the robots in swarms 3 and 4 in the fuzzy swarm approach.

enables the greedy robots to have a closer proximity to the targets, more aligned with the overall fitness observed in the fuzzy method. The limiting parameters, 0.05 and 1, can be adjusted until the desired performance for the greedy search is observed:

$$v_i = \begin{cases} v_i & \mathrm{RI}_i < 1 \\ v_i - (0.05 \times v_i) & \mathrm{RI}_i \geqslant 1 \end{cases} \tag{19}$$

Fig. 7 provides a detailed example of the mapping of a robot's search points over a number of iterations using the greedy search method.

For the greedy search, a single iteration includes the robot completing the entire process of mapping all of the search points and moving to the best fitness position, which is equivalent to the robot completing one circle in Fig. 7. The robot must travel to each of the search points in order to scan the fitness of all points before moving to the position of best fitness. A fitness evaluation occurs at every position change, including at each of the points visited during the mapping of the search points because the fitness is checked at each point. For a single greedy robot, one iteration is equal to four fitness evaluations.

8.2. Fuzzy swarm controllers

Fuzzy logic is used to replace in PSO the effect of the randomness in the velocity update equation. But the performance of the system still depends on the inertial weight *w*. Table 4 shows the result of the various case studies on the performance of the fuzzy swarm. All the results presented have been averaged over 100 trials. These results have been obtained on a 2.2 GHz, Pentium 4 processor-based PC. As can be seen in case 2 by adding a random term to the fuzzy output of Case 1, gave a slightly improved performance. In general, it can be seen that the direct implementation, adding a uniform random and a Gaussian random term to the fuzzy output gave similar performance. The uniform random term (Case 2) performed better than the rest in terms of convergence. Varying the value of '*w*', the inertia weight, as seen in Case 4 did not improve the performance on the contrary the convergence decreased. For Case 5, it can be seen that the mutation probability affects the convergence of the robots. A mutation of 0.7 gives the same result as no mutation in Case 1.

The robustness of the system has been tested by inducing a loss of robots during the search. Twenty percent of the robots in the swarm are lost. In this case, it can be seen that the system is quite reliable since the convergence dropped from 94% to just 93%.

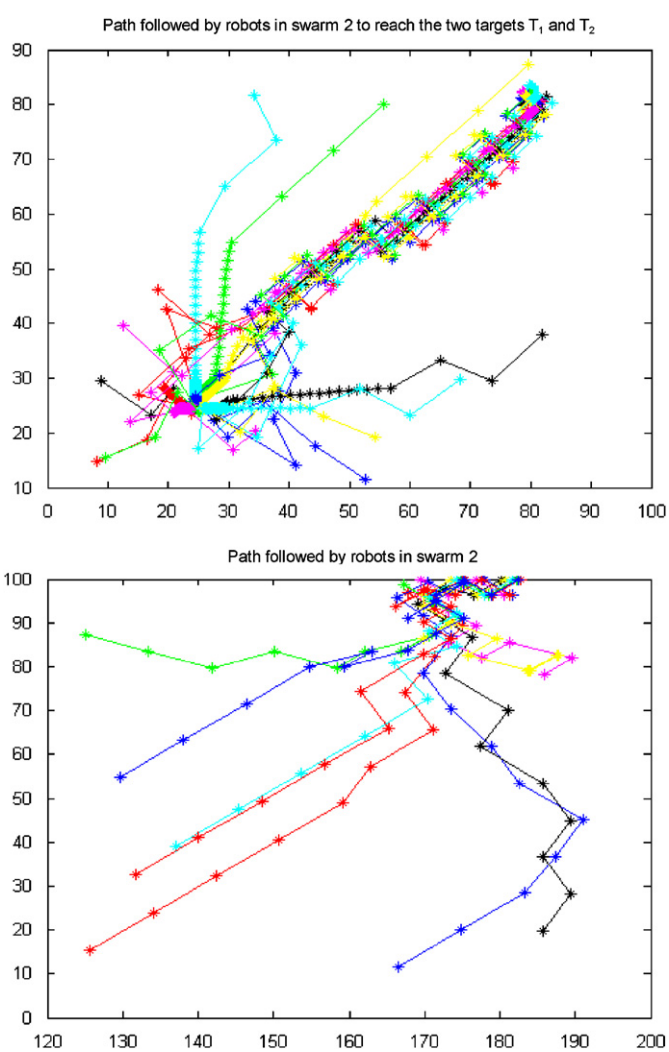

Fig. 10. Graphs for the path of the robots in swarms 1 and 2 in the swarm fuzzy approach.

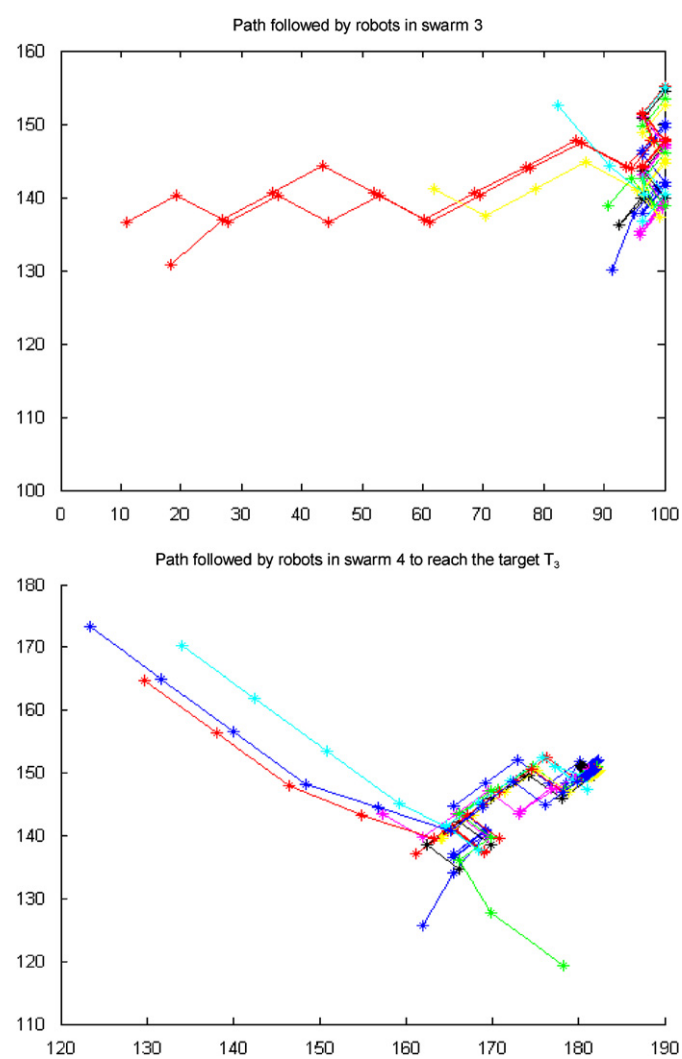

Fig. 11. Graphs for the path of the robots in swarms 3 and 4 in the swarm fuzzy approach.

In Case 6, 20% of the robots are lost during the search process. Case 7 is discussed in Section 8.3. It is observed that if the initial number of robots in the area with the targets is less than 6, the robots are unable to find the targets. Though it was seen that over the 100 trials, for the first swarm having two targets, 49 cases had an initial number less than 10 and of these only 9 are less than 7 (6 or less). The number of failures for this swarm totaled 6, of which 4 are of those less than 7. In swarm 4 where there is only 1 target, there is only 1 failure when the initial number of robots is less than 5. Table 5, lists the number of robots per swarm in the failure cases.

As seen in Table 5, for rows 1 and 4, where the initial numbers of robots are 2 and 3 in swarm 1 and swarm 4, respectively, the search can be expected to fail. Whereas for rows 2 and 3 where the initial number of robots is 12 in swarm 1, a failure is not expected. Table 6 shows cases where a failure could be expected but it turned out to be a successful case. Table 6, shows that though there are 4 failure cases with 6 or less robots, there are also 6 successes with 6 or less robots. Also the minimum number in the success case for finding the target in swarm 4 is 4.

8.3. Swarm of fuzzy logic controllers

In the case of a swarm of fuzzy logic controllers, the performance is not dependent on constants as is the case with the fuzzy swarm method. The performance of the swarm of fuzzy controllers depends on the design of the fuzzy system. Table 7 shows the results of the target location problem using the swarm-fuzzy method obtained over 100 trials. These results are obtained on a 2.2 GHz Pentium 4 processor-based PC. This application of the swarm of fuzzy controllers did not face the problem of the robots getting caught in a local minimum. Therefore, the case where mutation had to be used in the fuzzy swarm approach is not necessary here.

As seen in Table 4, Case 7 and Table 7, Case 3, both fuzzy methods outperform the greedy method with about 50% fewer iterations and a shorter run time. The greedy search also has a smaller convergence percentage at 87% where the swarm of fuzzy method stays above 90% for all cases tested and the fuzzy swarm method has many cases with convergence above 90% also.

The robustness of the system has been tested by inducing a loss of robots during the search. 20% of the robots in the swarm

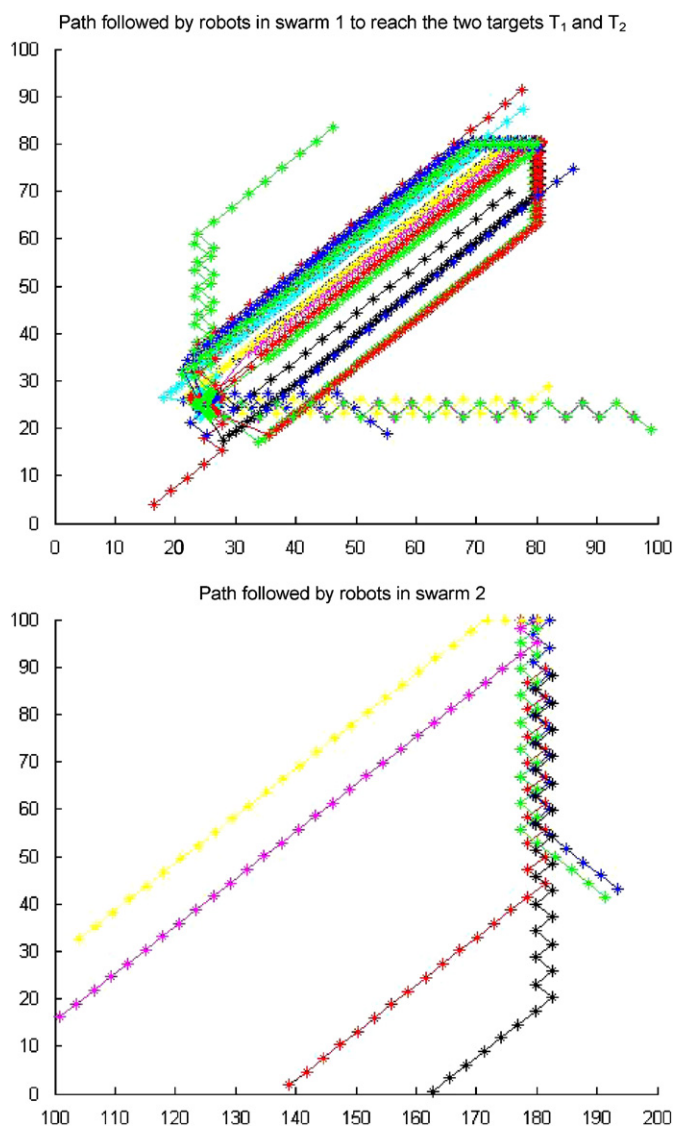

Fig. 12. Graphs for the path of the robots in swarms 1 and 2 in the greedy approach.

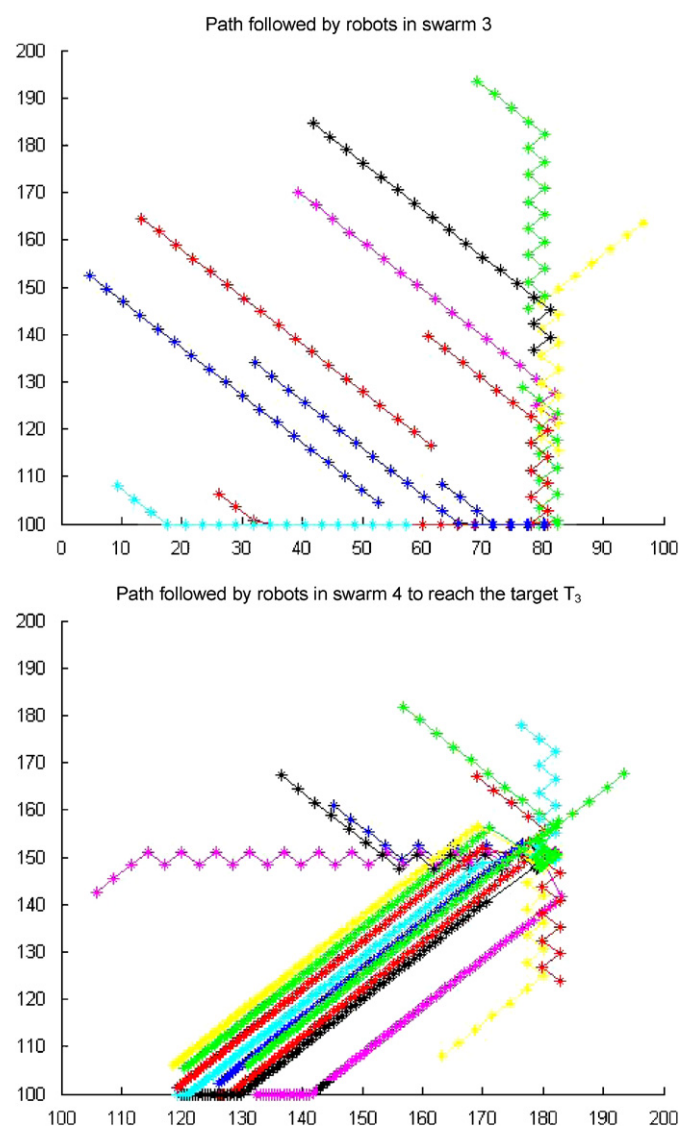

Fig. 13. Graphs for the path of the robots in swarms 3 and 4 in the greedy approach.

are lost. The result of this experiment has also been shown in Table 8. As can be seen, the time taken under the fault tolerance case (Case 2) did not increase drastically though the number of iterations required increased slightly. In terms of convergence, the reliability dropped from 94% to 92%. Table 8 also displays the failure cases. All the failures are in swarm 1, where there are 2

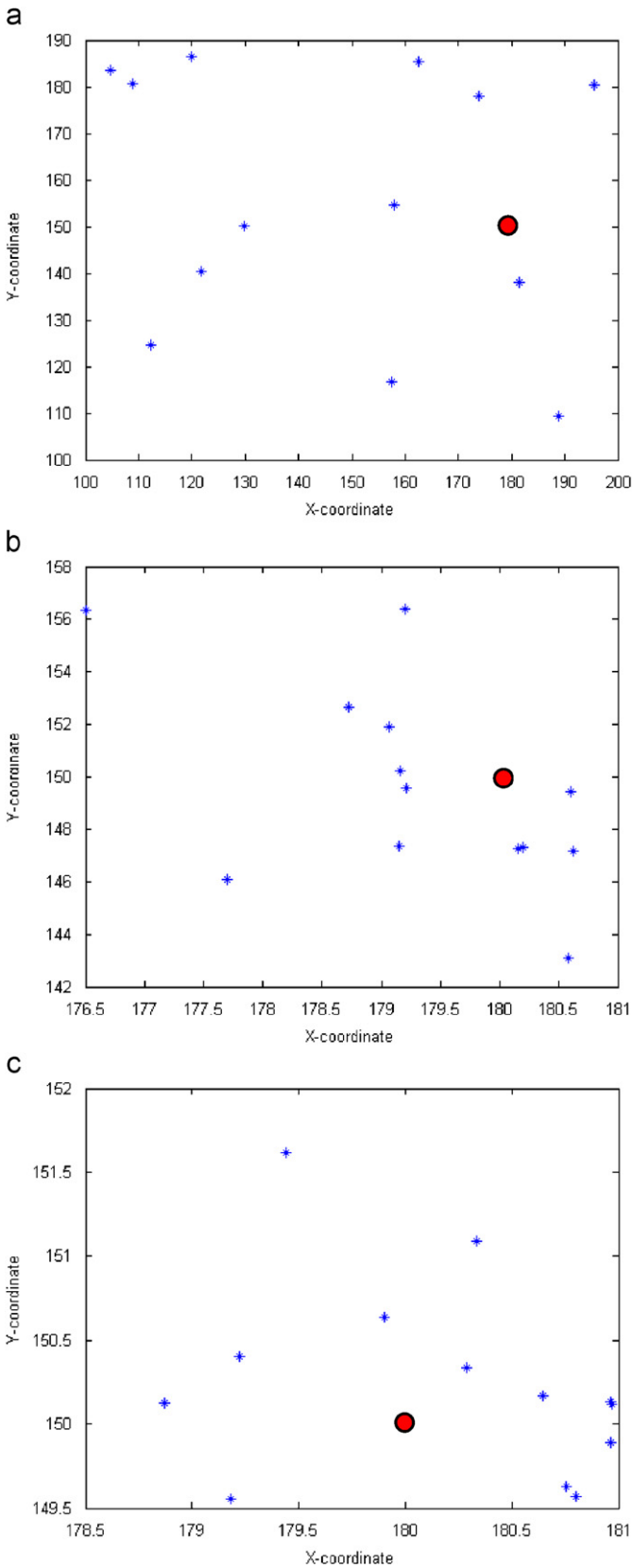

Fig. 14. Plots of the position of the robots in fuzzy swarm at different iterations in the search process in swarm 4 for target T_3: (a) initial position, (b) intermediate position and (c) final position.

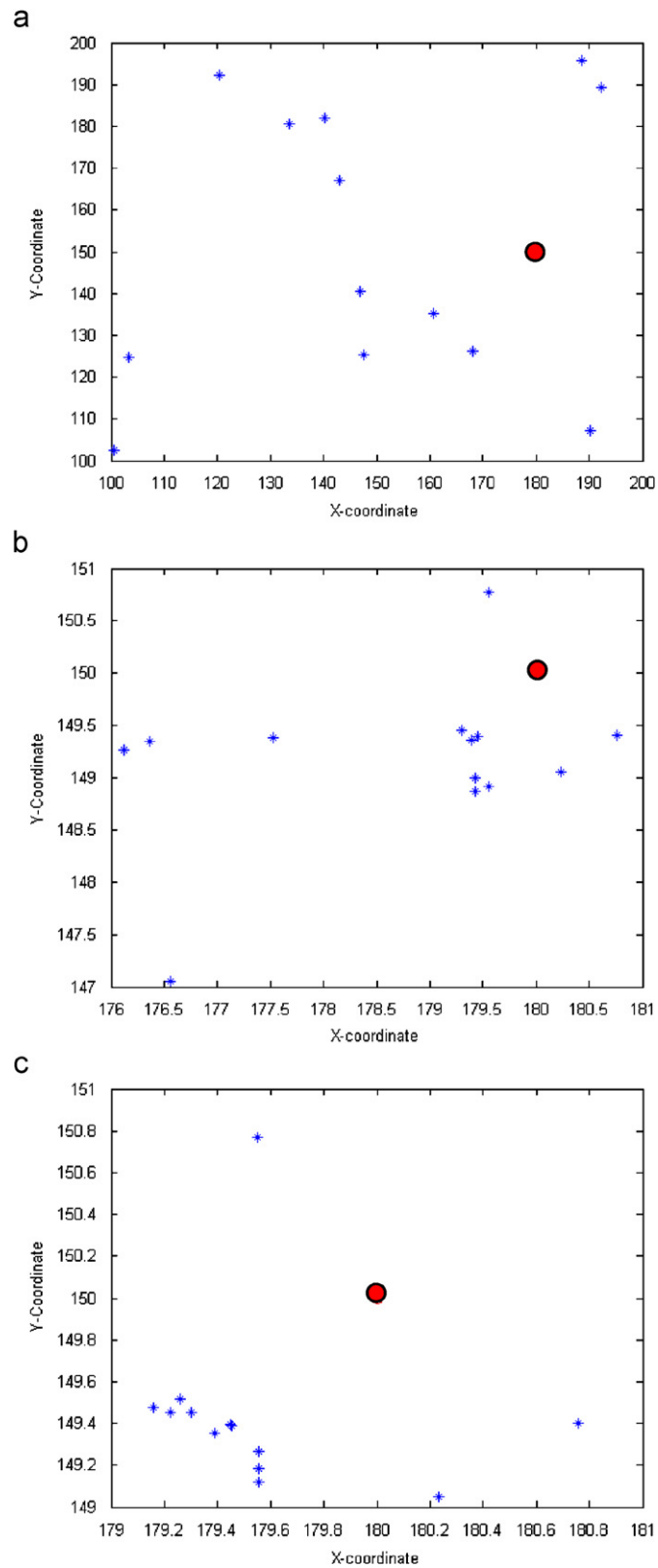

Fig. 15. Plots of the position of the robots in the swarm of fuzzy controllers at different iterations in the search process in swarm 4 for target T_3: (a) initial position, (b) intermediate position and (c) final position.

targets to identify. Out of the 100, trials there were 53 cases in which the initial number of robots is less than 10. All the 8 failures fell under this. Swarm 4 had 1 target and no failure. The minimum number of robots in swarm 4 is 6.

In row 4 of the table above, the initial number of robots is 2 and, therefore, a failure can be expected for this case. There are 3 unexpected cases in which the initial number of robots in swarm 1 is 5 and yet it resulted in a success. Table 9, shows a list of unexpected successful cases. Only 3 cases have been listed below. There are 4 other cases with 6 robots in swarm 1 that are successful too. There are 6 cases in swarm 4 with the initial number of robots as 6.

8.4. Comparison of fuzzy swarm controllers, swarm of fuzzy logic controllers, and greedy search approaches

As seen from Figs. 8–13, the path followed by the robots with the swarm of fuzzy controllers is more random. The path as seen in Figs. 9 and 10 is smoother than the paths in Figs. 10 and 11. Fuzzy swarm has a more random motion, since it still follows the PSO equation. In Figs. 12 and 13, the greedy immediate goal seeking drive can be seen by the fairly straight paths of the robots to the targets.

The performance of the fuzzy swarm, in terms of convergence, is lower than that of the swarm fuzzy. The time taken by the swarm fuzzy is more than the fuzzy swarm giving higher reliability in terms of convergence. Both proposed fuzzy methods perform more efficiently in terms of time, iterations and convergence than the greedy method.

Figs. 14 and 15 show the position of the robots at different time steps during the search process for the fuzzy swarm and swarm of fuzzy methods, respectively. The first plot shows the initial random deployment. The bottom plots in both the figures show the positions of the robots after finding the targets. This can be seen by the compression of the x and y axes. From Fig. 14 it can again be seen as in Figs. 8–13, that the fuzzy swarm method is more random by the different relative position locations of the robots as a swarm as they approach the target. The swarm of fuzzy paths in Fig. 15 are more like the greedy search where the robots relative positions with respect to each other are similar for each of the positions (a–c), the robots just move closer together as a swarm as they directly approach the target.

As can be seen from the Tables 4–9, both approaches are comparable with and without fault conditions.

9. Conclusion

This paper has presented two approaches for collective robotic search. Both the approaches are based on hybrids of fuzzy logic and swarm intelligence. The approaches were also compared to a non-swarm-oriented technique, greedy search.

Both the approaches are comparable in terms of reliability in performance based on convergence and fault tolerance. Swarm of fuzzy controllers takes less time and number of iterations to achieve success. Both fuzzy methods are more efficient than greedy search in terms of time, iterations and convergence. Future work involves deeper understanding of the effect of membership functions and rules on both approaches. Optimization of the membership functions, rules bases and the PSO parameter 'w' (inertial weight) is expected to improve the performance of the search. Also, currently a robotic platform is being developed for use in testing the fuzzy and other navigation methods in hardware for CRS problems in a laboratory setting.

Acknowledgements

The financial support from the National Science Foundation under Grant no. ECS # 0348221 and the University of Missouri Research Board are gratefully acknowledged for this work.

References

Cortes, J., Martinez, S., Bullo, F., 2006. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. IEEE Transactions on Automatic Control 51 (8), 1289–1298.
Dadios, E., Maravilla Jr., O., 2002. Cooperative mobile robots with obstacle and collision avoidance using fuzzy logic. In: Proceedings of IEEE International Symposium on Intelligent Control, pp. 75–80.
Doctor, S., Venayagomoorthy, G., Gudise, V., 2004. Optimal PSO for collective robotic search applications. IEEE Congress on Evolutionary Computations, 1390–1395.
Foux, G., Heymann, M., Bruckstein, A., 1993. Two-dimensional robot navigation among unknown stationary polygonal obstacles. IEEE Transactions on Robotics and Automation 9, 96–102.
Gesu, Di V., Lenzitti, B., Bosco, G.Lo., Tegolo, D., 2000. A distributed architecture for autonomous navigation of robots. In: Fifth IEEE International Workshop on Computer Architectures for Machine Perception, pp. 190–194.
Jadbabaie, A., Lin, J., Morse, A.S., 2003. Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Transactions on Automatic Control 48 (6), 988–1001.
Kennedy, J., 1999. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of IEEE congress on Evolutionary Computations, vol. 3, p. 1938.
Kennedy, J., Eberhart, R., 2001. Swarm Intelligence. Morgan Kauffman Publishers, San Francisco.
Kim, H., Butler, A., 1995. Motion Planning using fuzzy logic control with minimum sensors. In: Proceedings of the 1995 IEEE International Symposium on Intelligent Control, pp. 558–564.
Konishi, K., Kurabayashi, D., Asama, H., Shin, S., 1999. Analysis of emergence by intelligent data carrier system for collective robots based on stochastic models. In: IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics, vol. 4, pp. 769–774.
Lin, Z., Broucke, M., Francis, B., 2003. Local control strategies for groups of mobile autonomous agents. In: Proceedings, IEEE Conference on Decision and Control, pp. 1006–1011.
Sayeed, A., Zinger, D., Elbuluk, M., 1994. Fuzzy controller for inverter fed induction machine. IEEE Transactions on Industry Applications 30 (1), 83–93.
Sharir, M., 1989. Algorithmic motion planning in robotics. IEEE Computer 22 (3), 9–19.
Tang, Z., Ozguner, U., 2005. Motion planning for multitarget surveillance with mobile sensor agents. IEEE Transactions on Robotics 21 (5), 898–908.
Van Dyke Paranuk, H., 2005. Phermone learning for self-organizing agents. IEEE Transactions on Systems, Man, and Cybernetics 35, 316–326).
Zadeh, L.A., 1988. Fuzzy Logic. IEEE Computer 21 (4), 83–93.
Zhang, N., Wunsch II, D., 2003a. Fuzzy logic in collective robotic search. In: The 12th IEEE International Conference on Fuzzy Systems, vol. 2, pp. 1471–1475.
Zhang, N., Wunsch II, D., 2003b. A comparison of dual heuristic programming (DHP) and neural network based stochiastic optimization approach on collective robotic search problem. In: Proceedings of the International Joint Conference on Neural Networks, vol. 1, pp. 248–253.