01 Jan 2004

Navigation of Mobile Sensors Using PSO and Embedded PSO in a Fuzzy Logic Controller

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Sheetal Doctor

Recommended Citation

G. K. Venayagamoorthy and S. Doctor, "Navigation of Mobile Sensors Using PSO and Embedded PSO in a Fuzzy Logic Controller," *Conference Record of the 39th IAS Annual Meeting of the IEEE Industry Applications Conference, 2004*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2004.
The definitive version is available at https://doi.org/10.1109/IAS.2004.1348565

Navigation of Mobile Sensors Using PSO and Embedded PSO in a Fuzzy Logic Controller

Ganesh K. Venayagamoorthy
Department of Electrical and Computer Engineering
University of Missouri - Rolla, MO 65409, USA
gkumar@ieee.org

Sheetal Doctor
Department of Electrical and Computer Engineering
University of Missouri - Rolla, MO 65409, USA
skdkv6@umr.edu

***Abstract* -- This paper presents novel structures for optimization and communication of a swarm of mobile sensors or robots for maximizing local and global tasks such as firefighting, landmine detection, radioactivity detection, etc. The navigation of the sensors is carried out using two strategies. The first strategy is based on Particle Swarm Optimization (PSO) and the second strategy is based on a swarm of fuzzy logic based controllers. In addition, the membership functions and the rules of the Fuzzy Logic Controller (FLC) are optimized using the PSO algorithm. Navigation of mobile sensors is considered in this paper to locate desirable target sources in a given sensing area. Both approaches presented do not depend on the number of target sources. Results are provided for target locations based on a PSO, a swarm of fuzzy logic controllers and a swarm of optimized fuzzy logic controllers.**

Keywords - Mobile sensors; particle swarm optimization; fuzzy logic; membership functions, rules, optimization

I. INTRODUCTION

Mobile sensors are used for various applications because of their area coverage and simplicity of implementation. A number of these sensors are used to explore and exploit environments which are inhospitable to humans. These include remote areas, hazardous or toxic locations, planetary exploration, seismic activity detection, military surveillance etc. The advantage of using this method is that when an event occurs, all the sensors in the given space try to flock to the source of the event. Since the number of the sensors is large it accommodates for the failure of any of the sensors thereby increasing the redundancy of the system and in turn the reliability. There are various methods used so far for unmanned navigation. Most of these are vision based applications. Neural networks, information fusion techniques [1, 2], multi-sensor and computer controlled methods have also been explored.

In order to optimize the effective operation of these sensors, the computation and memory need to be reduced to a minimum. Also the communication between the distributed sensors and a local/central station (if they exist) should be kept at a minimum. The power consumption of the entire system also needs to be reduced to a minimum. Therefore, it is better to execute as many operations as possible locally on each sensor and only the most vital data be sent across to the other sensors *if necessary* via coordinating base. Stochastic - distributed algorithms have proven to be most efficient for these applications. Various algorithms like evolutionary computations, genetic algorithms, adaptive cultural evolutions, etc have also been used to perform these tasks.

Swarm intelligence is based on the social behavior of flocks of birds/schools of fish and the success of the swarm is because of the communication established between them. The division of large networks into swarms comprising of cooperating nodes has several advantages such as increased robustness and security; simplified addressing, routing, and localization; low energy consumption, and lower memory requirements.

This paper explores two novel methods for navigation of mobile sensors. The first method involves the use of the particle swarm optimization algorithm and the second method is based on an embedded PSO in a swarm of fuzzy logic controllers for the navigation of mobile sensors. In the second method, PSO is also used to determine the optimal input and output membership functions and the optimal rules for a swarm of fuzzy logic based controllers. Navigation of mobile sensors has been developed with these two methods for the location of a single, multiple and unknown number of targets in a given sensing area.

The rest of the paper is outlined as follows: Section II describes mobile sensor network architecture; Sections III and IV describe PSO and fuzzy logic respectively. Section V describes the application of PSO and fuzzy logic to the target location problem and presents results with these approaches. And finally, the conclusion is given in Section VI.

II. MOBILE SENSOR NETWORK ARCHITECTURE

Mobile sensor networks are becoming increasingly important to manage unmanned physical systems. Distributed systems are implemented in order to decentralize the computational complexity. The factors for design consideration include energy management, efficient communication with less disturbances, efficient computation, etc. Various methods explored until now have included wireless communication, image processing and vision based application, neural networks, etc [3]. Due to interference, the transmission range for the communicating bodies reduces. Therefore, stochastic algorithms run at the sensor level and only data to be sent out to the others is sent over a communication media.

Fig. 1 shows a three tier architecture for the communication and navigation of mobile sensors presented in this paper. The sensing area has been divided into local neighborhoods based on arbitrary sensor deployment and if known the number of targets. Each sensor node sends it best position with respect to potential target to the neighborhood level. The neighborhood level calculates the local best within each neighborhood and sends back to the sensor nodes. The global level gets the local best positions from the neighborhood level and uses this information for further action if required.

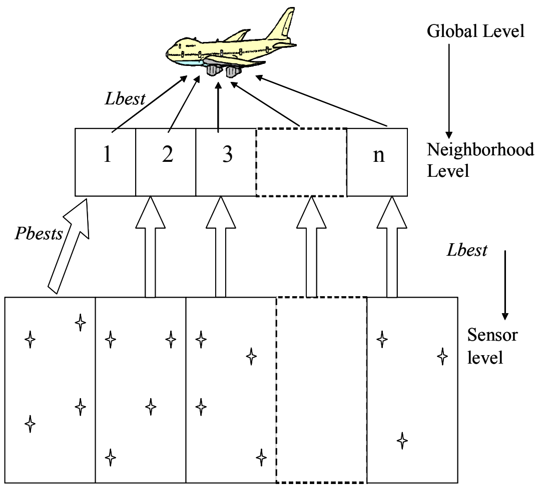

Figure 1. A three tier architecture for the navigation of mobile sensors.

This architecture can be used for various applications such as the landmine detection problem, firefighting, military operations, etc; the sensors can be dropped in the area under surveillance by an airplane. The area is divided into clusters so that the sensor motion can be restricted to a smaller area based on a divide and conquer philosophy [4]. Each sensor calculates its position on the basis of a fitness function with respect to the target. The only parameter it sends is that of its best position and in turn receives the best position of the swarm/cluster. Thus, the communication is kept to a minimum and this reduces transmission losses and congestion overcoming bandwidth limitations.

III. Particle Swarm Optimization

Particle swarm optimization reported by Kennedy and Eberhart is similar to the concept described above [5]. It is relatively a new concept and has been used for target tracing by autonomous communicating bodies [6]. 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 mobile sensor nodes and the local version of the PSO algorithm is considered in the context of this application. [7]

Fig. 2 shows a graphical representation of a single cluster/neighborhood. Within a defined sensing area, the system has a population of mobile sensor nodes. Each node is randomized with a velocity and 'flown' in the problem space. They have memory and they are able to keep track of their previous best position ('P_{best}') with respect to the target. Thus each sensor node has a 'P_{best}'. The best value of all these ''$P_{best's}$' is defined as the best position in the local neighborhood 'L_{best}' with respect to the target. The velocities and positions of these sensors are constantly updated until they have all converged at the target. Thus, in terms of memory requirements, PSO requires only two values (other than the velocity and position from the previous iteration), 'P_{best}' and 'L_{best}'. The basic PSO velocity and position update equations are given by (1) and (2).

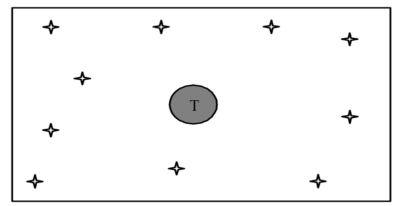

Figure 2. Randomized sensor nodes with a single target shown in the sensing area.

$$V_{new} = w * V_{cur} + c_1 * rand() * (P_{best} - P_{cur}) + c_2 * rand() * (L_{best} - P_{cur}) \quad (1)$$

$$P_{new} = P_{cur} + V_{new} \quad (2)$$

Where

V_{new} - New velocity calculated for each sensor node
V_{cur} - Velocity of the mobile sensor node from the previous iteration
P_{new} - New position calculated for each mobile sensor node
P_{cur} - Position of the mobile sensor node from the previous iteration
w - Inertial weight constant
c_1 & c_2 - cognitive and social acceleration constants

The population responds to the factors 'P_{best}' and 'L_{best}' in order to find a position whose fitness gets better over iterations finally reaching a stage where all the swarm members achieve the highest fitness. The procedure for the implementation of PSO involves the following basic steps:

(i) Define the sensing area with its boundaries. Initialize an array of sensors 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 studies only).

(ii) Evaluate the fitness function (e.g. Euclidean distance, intensities). Select the L_{best} from P_{best} .

(iii) Compute the new velocities and positions of the sensor nodes using (1) and (2) above respectively.

(iv) Check if the sensors' positions are within the problem space. Also check if the velocity exceeds the predefined limits and if they do then the velocity is set to the maximum velocity and the new position is set to its previous best position.

(v) Calculate the new fitness function for all the sensors' 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 sensors converge at the target(s).

IV. Fuzzy Logic Controller

From the inception of fuzzy logic by Zadeh in the 1960's, its foundations and applications have grown stronger and wider over the years [8]. 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 non linear techniques.

Fuzzy logic techniques are primarily applied to systems that cannot function well with the conventional analytical model-based control techniques [9]. Fuzzy logic introduces a realistic situation into a system. It differs from binary logic by allowing an 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 mostly depends on the expert defining the rules and the membership functions. Therefore, there is a need to find optimal rules and membership functions for improved performance of the system and this is also addressed in this paper.

Fig. 3 shows the block diagram of a typical fuzzy system. The fuzzification process is an interface between the real world parameters and the fuzzy system. 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 a reverse process of fuzzification. It maps the fuzzy output variables to the real world variables that can be used in the controlling a real world application.

V. Implementation And Results

As shown in Fig. 1, the mobile sensors are randomly deployed on the ground. The area/zone is assumed to have an unknown number of targets that needs to be located by sensor nodes. For simulation purposes, a sensing area of 300 by 200 (60000 sq units) is considered. Fig. 4 shows the distribution of the sensors (represented by the blue asterisk) and targets (represented by the red circles) in the sensing area. The sensing area is the same for both PSO and fuzzy logic based mobile sensor navigation approaches. The targets are chosen to be light sensors of different intensities (I_1, I_2, I_3, I_4, etc.).

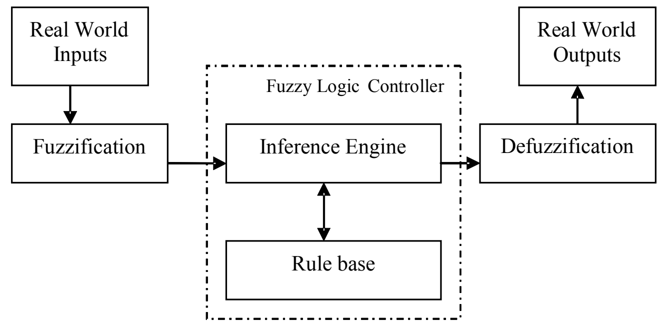

Figure 3. Block diagram of a fuzzy system.

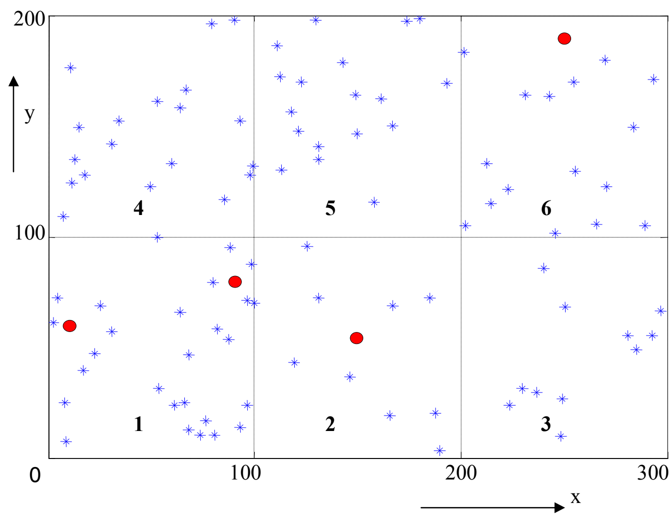

Figure 4. Division of the search space with four targets into six neighborhoods.

Apart from solving the above mentioned problem by the two methods, this paper also looks at the effect different parameters on the performance of the system. The parameters for PSO are the constants the inertial weight, w and the cognitive and social acceleration constants, c_1 & c_2. The parameters for fuzzy logic are the input and output membership functions, and the rules. The system performance is measured in terms of time, number of iterations taken for all the sensor nodes to converge at the targets and the rate of convergence on a given number of trials. Convergence refers to the success of the swarm in finding all the targets within the given sensing area.

A. Application of PSO

The simulation was carried out considering two separate scenarios, namely:

- The sensor nodes have single intensity readings.
- The sensor nodes have four directional intensity readings.

1) Sensor nodes with single intensity readings: The sensor nodes are assumed to be able to pick up intensity readings from all directions. The intensity of a target read by a sensor is given by (3). Fig. 5 shows a graphical representation of a sensor node reading the intensities from two target sources.

Table I shows how the variations of w, c_1 and c_2 in the PSO algorithm affect the performance of the swarm search for targets. The average number of iterations and time taken for the search is computed over 100 trials. The convergence expressed in percentage shows the number of times over 100 trials the swarm of mobile sensors converge at the targets.

$$\textit{Intensity reading at the sensor node} = I_1/d_1^2 + I_2/d_2^2 \qquad (3)$$

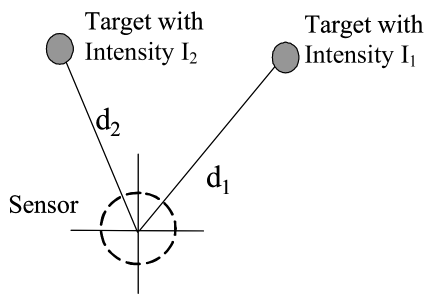

Figure 5. Intensity of the targets seen at a sensor node with a single directional reading.

TABLE I

EFFECT OF W, C_1 AND C_2 ON THE SINGLE DIRECTIONAL SENSOR NODES

w	c_1	c_2	Number of iterations	Time (sec)	Convergence
0.8	2	2	1200.00	3.31	0%
	0.5	2	1200.00	2.90	0%
	2	0.5	1190.69	2.90	10%
	0.5	0.5	1098.33	2.42	76%
0.6	2	2	1100.49	2.36	83%
	0.5	2	1056.82	2.18	98%
	2	0.5	1194.40	2.88	4%
	0.5	0.5	1193.90	2.91	4%

2) Sensor nodes with four directional readings: The sensor nodes have four directional sensors located in the: 'North', 'East', 'South' and 'West'. Fig. 6 shows a graphical representation of the sensor and two targets. Equations (4), (5), (6) and (7) give the relative intensities read at the East, the North, West and South sensors respectively.

$$\textit{Intensity reading in the East direction} = I_1/d_1^2 * (\cos\theta_1) \qquad (4)$$

$$\textit{Intensity reading in the North direction} = I_1/d_2^2 * (\cos\theta_2) + I_2/d_3^2 * (\cos\theta_3) \qquad (5)$$

$$\textit{Intensity reading in the West direction} = I_2/d_4^2 * (\cos\theta_4) \qquad (6)$$

$$\textit{Intensity reading in the South direction} = 0 \qquad (7)$$

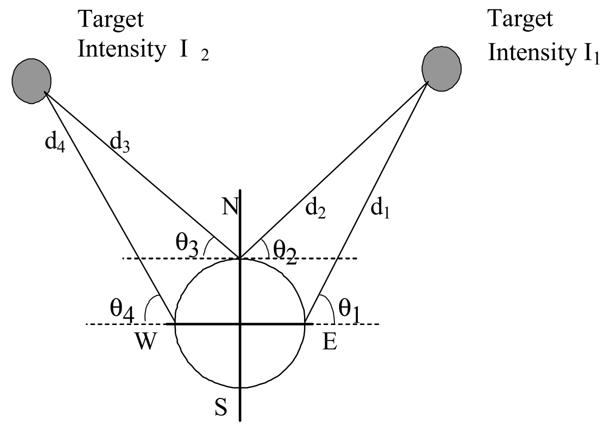

Figure 6. Intensity of the targets read by the four-directional sensor node ($d_1 \cong d_2$ and $d_3 \cong d_4$).

Table II shows how the variations of w, c_1 and c_2 in the PSO algorithm affect the performance of the swarm search for targets.

TABLE II

EFFECT OF W, C_1 AND C_2 ON THE FOUR DIRECTIONAL SENSOR NODES

w	c_1	c_2	Number of iterations	Time (sec)	Convergence
0.8	2	2	1181.58	6.35	0%
	0.5	2	1106.09	5.88	0%
	2	0.5	985.39	4.98	26%
	0.5	0.5	924.10	3.99	93%
0.6	2	2	910.60	4.90	94%
	0.5	2	859.40	4.79	99%
	2	0.5	889.36	5.07	32%
	0.5	0.5	841.85	5.24	8%

As can be seen from Tables I and II, the performance of the swarm is better in the case where four-directional sensors are used. But for the specific values of the constants, for example w=0.6, c_1=0.5 and c_2=2, there isn't much difference in the performance of the two cases. More the number of sensors better will be the performance of the swarm, but at the expense of increased cost.

The swarm network performs a faster and better search for with inertia 'w' of 0.6 than 0.8. The number of iterations for convergence is lower in the case of w=0.6 and the percentage of convergence is also higher in the same case. The w, c_1 and c_2 parameters of the PSO can be optimized using another PSO as described by the authors' previous work [10].

B. Application of Fuzzy Logic

A swarm of fuzzy logic controllers have been used for the target location problem for guiding the swarm of the sensor nodes to the target.

Mobile sensor navigation has been dealt with by two methods. The first method is conventional fuzzy logic method where the membership functions and the rule base are developed based on heuristics and the second method is using optimal FLCs. When designing an optimal fuzzy controller, there are two primary considerations: finding optimal membership functions for input and output variables; and finding an optimal set of rules between input and output variables. Herein, PSO is used to find the optimal membership functions and rules.

The FLC approaches have been implemented for the target location problem in Fig. 4 just like with the PSO explained above but now the position of the mobile sensors are given by (8).

$$P_i = (X_i + \Delta X_i\,,\ Y_i + \Delta Y_i) \tag{8}$$

where

$$\Delta X_i = f(I_i,\ Gdx) \tag{9}$$

$$\Delta Y_i = f(I_i,\ Gdy) \tag{10}$$

Here, $Gdx = L_{best}x$ - Px and $Gdy = L_{best}y$ – Py where Px and Py are the x and y coordinates of the sensors current position respectively and, $L_{best}x$ and $L_{best}y$ are the x and y coordinates of the L_{best} of the swarm respectively. The L_{best} is calculated in the same way as in the PSO implementation discussed above.

In the fuzzy logic application, the sensors have been considered to read a single intensity values as shown in Fig. 5. The inference engine used for this simulation is Mamdani's product inference engine and the defuzzification is based on the center average.

There are three input variables to the fuzzy system namely:

- The intensity reading at the sensor nodes (I_i). This variable has four triangular membership functions - zero (Z), small (S), medium (M) and large (L).
- The difference of the x coordinates of the current position of the sensor node with respect to the sensor node having the best position in the swarm given by 'Gdx'. This variable has five triangular membership functions - zero, very small (VS), small (S), medium (M) and large (L).
- The difference of the y coordinates of the current position of the sensor node with respect to the sensor node having the best position in the swarm given by 'Gdy'. This variable has five triangular membership functions - zero, very small, small, medium and large.

There are two output variables from the fuzzy system namely:

- The displacement amount to be added to the x coordinate ΔX_i which has four triangular membership functions - very small, small, medium and large.
- The displacement amount to be added to the y coordinate ΔY_i which has four triangular membership functions - very small, small, medium and large.

The membership functions for the input and the output variables developed by heuristics are given in Figs. 7 and 8.

In the both FLC approaches, there are two different sets of the rules as shown in Fig. 9. The first set of rules referred to as the coarse set, causes the sensor nodes to take larger steps. This helps in exploration of the sensing area for possible targets and once a target has been identified, the second set of rules referred to as the fine rule set is used that causes the sensor nodes to take smaller steps towards the target(s). This brings about precision in the movement of the sensor nodes.

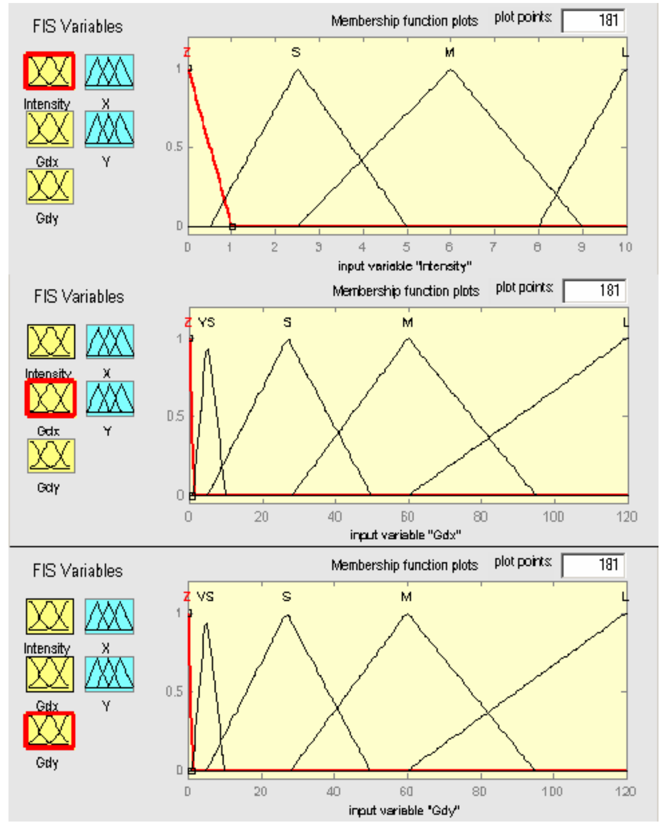

Figure 7. Input membership functions developed by heuristics (not optimized).

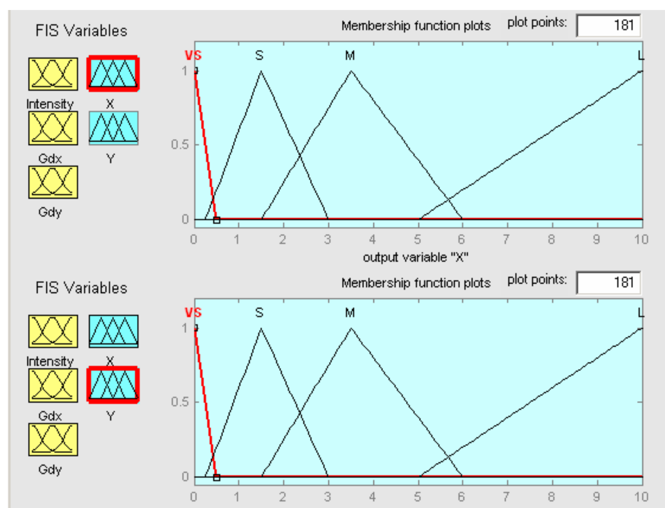

Figure 8. Output membership functions developed by heuristics (not optimized).

Equations (11) and (12) give an example of the IF-THEN statement used in the conventional fuzzy logic. As can be seen, the antecedent of both (11) and (12) are the same but the consequents are different. Equation (11) is from the coarse rule set whereas (12) is from the fine rule set. 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.

$$\text{IF 'Intensity' is Small and 'Gdx' is Large THEN } \Delta X_i \text{ is Large} \quad (11)$$

$$\text{IF 'Intensity' is Small and 'Gdx' is Large THEN } \Delta X_i \text{ is Medium} \quad (12)$$

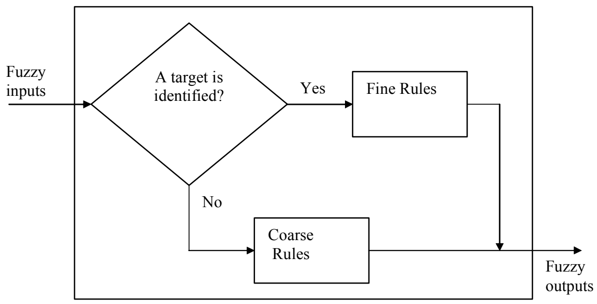

Figure 9. Switching logic between the coarse and fine rules.

Table III shows the performance of the swarm of sensor nodes when the membership functions and rules are optimized independent of each other. It also shows the time taken and the number of iterations required by the sensor nodes to find the target. The simulation was carried out on a range of [0, 100] in both x and y directions, which is a single cluster from the whole space as shown in Fig. 4. The simulation covered all the clusters successively. The results shown above are obtained over 100 trials. As can be seen the time and number of iterations taken by the swarm of fuzzy controllers using all the optimized parameters has been reduced. PSO was used to find the optimal parameters for the fuzzy logic controllers. Figs. 10 and 11 show the optimal membership functions obtained with PSO.

TABLE III

RESULTS OF OPTIMIIZING THE FUZZY LOGIC CONTROLLER PARAMETERS (MEMBERSHIP FUNCTIONS, COARSE AND FINE RULE SETS) SEPARATELY

Case Study	Membership Function	Coarse Rules	Fine Rules	Itera-tions	Time (sec)
1	Unoptimized	Unoptimized	Unoptimized	355.80	349.97
2	Optimized	Unoptimized	Unoptimized	308.41	328.24
3	Unoptimized	Optimized	Unoptimized	356.96	356.18
4	Unoptimized	Unoptimized	Optimized	307.44	325.65
5	Optimized	Optimized	Optimized	306.15	326.78

As can be seen from Figs. 8 and 11, the membership functions are different before and after optimization. In Fig 8, the membership functions have been assigned names according to the author's initialization. When PSO was used, it came up with its own membership functions starting from random initial values. Though the terminology (VS, S, M, and L) has been kept the same, PSO found membership functions independent of the terminology. Therefore, looking at the membership functions in Figs. 10 and 11, it can be observed that PSO finds an optimal set by swapping around the memberships irrespective of the names of each. The optimal parameters, membership functions and the rules are problem dependent and need to be recalculated every time the application changes.

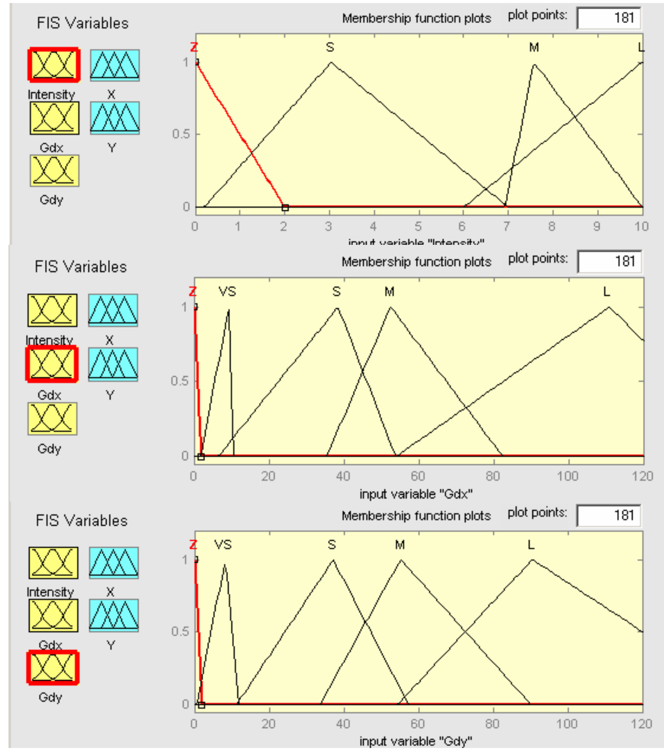

Figure 10. Optimized input membership functions with PSO.

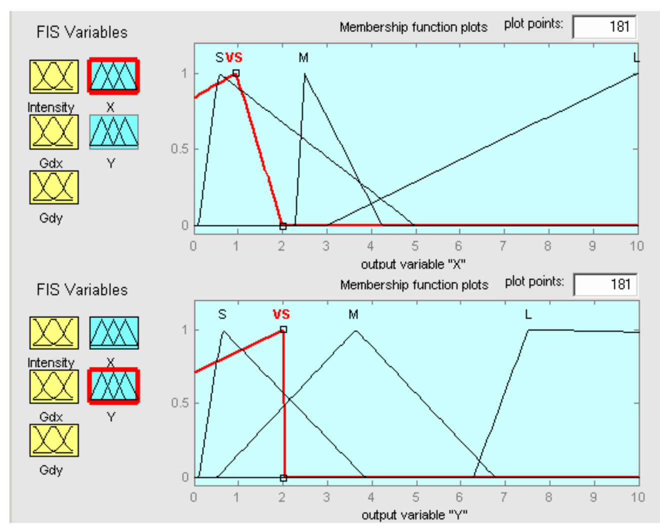

Figure 11. Optimized output membership functions with PSO.

Table IV shows the difference in the performance before and after optimization of the fuzzy membership functions and rules for the target location problem of Fig. 4 with six clusters. From Table IV, it can be observed that the time taken and the number of iterations required by the controllers are reduced with optimal swarm of fuzzy logic controllers.

TABLE IV

COMPARISON OF PERFORMANCE OF SWARM OF FUZZY CONTROLLERS WITH UNOPTIMIZED AND OPTIMIZED FUZZY MEMBERSHIP FUNCTIONS AND RULES

	Without Optimization	With Optimization
Time (sec)	349.97	326.78
Iterations	355.80	306.15
Convergence	100 %	100%

C. Comparison of PSO and Fuzzy Logic

Tables I, II, III and IV show that the PSO and fuzzy logic control methods have been successfully implemented on the for the target location problem of Fig. 4. Results show that though the rate of convergence is slightly higher in the case where fuzzy logic has been implemented, the time taken for convergence is much less in the case with PSO. The time taken for convergence with PSO implementation is ten times faster than the time taken by fuzzy logic. The PSO algorithm comprises of random functions which causes convergence of sensor nodes uncertain unless all the parameters (w, c_1 and c_2) are chosen carefully [10]. With the swarm of fuzzy logic controllers, this uncertainty is removed at the expense of more search time.

VI. CONCLUSION

This paper has presented two novel structures for optimal navigation of a swarm of mobile sensors to achieve local and global tasks such as firefighting, landmine, radioactive detection, etc. These structures are based on collective intelligence implemented using the PSO algorithm and a swarm of fuzzy logic controllers. Three-tier hierarchical navigation architecture has been presented. This paper has also shown that it is possible to optimize the input and output membership functions and the rules of fuzzy systems using particle swarm optimization. The results in this paper show it is possible to carry out optimal navigation of mobile sensors based using these strategies in an efficient, economic and reliable manner.

ACKNOWLEDGEMENT

The support from the National Science Foundation under CAREER Grant: ECS # 0348221 is gratefully acknowledged for this work.

REFERENCES

[1] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks", *Pervasive Computing, IEEE*, and Volume: 2, Issue: 4, Oct.-Dec. 2003, pp. 34 – 42.

[2] V. Di Gesu, G. Gerardi and D. Tegolo, "M-VIF: A machine-vision based on information fusion", *Proceedings of the Computer Architectures for Machine Perception*, 15-17 Dec. 1993, pp. 428 – 435.

[3] M. Haenggi, "Mobile sensor-actuator networks: opportunities and challenges", *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and their Applications*, 22-24 July 2002, pp. 283 – 290.

[4] S. Mulder and D. Wunsch, "Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks", *Advances in Neural networks research-IJCNN*, Vol. 16, numbers 5-6, June - July 2003.

[5] J. Kennedy and R. Eberhart., *Swarm Intelligence,* Morgan Kauffman Publishers, San Francisco, CA. ISBN 1-55860-595-9, 2001

[6] V. Di Gesu, B. Lenzitti, G. Lo Bosco and D. Tegolo, "A distributed architecture for autonomous navigation of robots", *Fifth IEEE International Workshop on Computer Architectures for Machine Perception,* 11-13 Sept. 2000, pp.190 – 194.

[7] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance", *Proceedings of congress on Evolutionary Computation, 1999. CEC 99.* Volume: 3, 6-9 July 1999, pp. 1938.

[8] L. A Zadeh, "Fuzzy Logic" *IEEE Computer*, Volume: 21, Issue:4, April 1988, pp. 83 – 93.

[9] A. Sayeed, D. Zinger and M. Elbuluk, "Fuzzy controller for inverter fed induction machines", *IEEE Transactions on Industry Applications*, vol. 30, no. 1, January/February 1994, pp. 78 – 84.

[10] S. Doctor, G. Venayagamoorthy and V. Gudise, "Optimal PSO for collective robotic search applications", *IEEE Congress on Evolutionary Computations*, June 19-23, 2004, Portland OR, USA.

[11] H. Yim and A.C. Butler, " Motion planning using fuzzy logic control with minimum sensors", *Proceedings of the 1995 IEEE International Symposium on Intelligent Control* , 27-29 Aug. 1995 pp. 558 – 564.

[12] E. P. Dadios and O. A. Maravillas Jr. "Cooperative mobile robots with obstacle and collision avoidance using fuzzy logic" *Proceedings of the 2002 IEEE International Symposium on Intelligent Control,* 27-30 Oct. 2002, pp.75 – 80.