Simultaneous Evolution of Neural Controllers for Multi Robot Formation Control

Genci Capi, *Member, IEEE* Mitsuki Kitani, Zulkifli Mohamed

Abstract— **This paper presents a new method for adaptive multiple robot formation using multiobjective evolution of neural controllers. An advantage of our proposed algorithm is that in a single run of multiobjective evolution are generated multiple neural networks that control robots to several relative positions to each other. Therefore, based on the environment conditions the robot can switch to different neural controllers resulting in different formations. We compare the performance of two formation algorithms: (a) leader based formation (LBF) and (b) follower based formation (FBF). Results are shown which demonstrate that the proposed method can be applied effectively for generating neural networks for multiple robot formation tasks. In addition, FBF outperformed LBF in terms of the quality of solutions.**

I. Introduction

Multiple robots systems have attracted considerable attention due to their wide range of applications, such as environment exploration, cooperative task performance and target tracking. One important research issue in systems of multiple robots is the motion planning for "formation paradigm". The approaches that address the problem of multiple robots formation range from leader following [1], [2], virtual structures [3], [4] and virtual leaders [5], [6]. In other works, social potentials [7] and formation constrained functions [8] are used to guide robots into formations. In [9] neural network is applied for a vision-based moving in formation by four mobile robots was presented. One robot, the leader, goes first providing moving plans to the other robots that follow the leading robot. In motion control, for each robot a radial basis function network approximated by learning is used.

Most of previous works focus on controlling the robots to a specific formation. However, in real life environments, robots have to switch to different formations, as the environment changes. In this paper, we formulate the multiple robot formation as a multiobjective optimization problem, which in general generates multiple solutions in the Pareto optimal front [10], [11]. In our method, the accumulated error between the target and real position of each robot relative to the leader robot is considered as a separate objective function.

In the proposed method, we evolve the weight connections of the neural controller for each target location. As the number of target locations increase and spread in a wide area, the evolved neural controllers have a poor performance ([12]). This is because the initial robot position influenced the fitness function during evolution. For example, even if good individual (NN) of the population can get a bad fitness if the distance between the initial and the target position is large and vice-versa. To solve this problem, we increase the number of robots during evolution considering as fitness function the average fitness of all robots. The initial positions of multiple robots can be designed based on the target positions, reducing the effect of the initial positions in the fitness function. In addition, in this paper we compare the performance of two formation algorithms: LBF and FBF. In the LBF the leader robot observes each follower robot position relative to the target one. The leader robot calculates and sends the follower robot action wirelessly. In difference from LBF, in the FBF each follower robot determines by itself the best action based on the leader robot position relative to the target one.

An advantage of the proposed method is that in addition to the neural networks controlling the robots to get to the target locations, are generated also neural controllers that position the robots in other locations. This can be utilized in two ways: 1) Switching between neural controllers resulting in different formations; 2) Increasing the number of the robots in the formation. First, the neural controllers are evolved in a simulated environment and than they are evaluated in the hardware experiments, showing a good performance.

The rest of the paper is organized as follows. LBF and FBF are described in Section II. The neural controllers are presented in Section III. Section IV details the formation task as a multiobjective evolution problem. Section V provides the simulation and experimental results. Section VI provides the conclusions and potential future extensions.

II. Formation Task

The robots have to get to their position relative to the leader robot and keep the geometrical formation throughout their travel. Because the robots initial positions are different from the target ones, the robots have to move fast to reach the target positions. In our implementation, we evolved neural controllers for three target locations of the follower robot relative to the leader robot.

A. Leader Based Formation (LBF)

In the LBF, the leader robot utilizes the visual information to measure the distance, angle and orientation of each follower robot (Fig. 1(a)). Each follower robot action (NN output) is sent by the leader robot, wirelessly. From multiple pre-evolved neural controllers of several target locations, the leader robot

G. Capi, M. Kitani and Z. Mohamed are with the Department of Electric and Electronic Systems Eng., University of Toyama, Japan (e-mail: capi@eng.u-toyama.ac.jp).

simulates the appropriate neural network that controls each follower robot to get to the goal location, resulting in the desired multi-robot formation. When the number of follower robots is smaller than the number of target locations (neural controllers), the leader robot can switch between neural controllers resulting in several formations.

B. *Follower Based Formation (FBF)*

In the FBF, each follower robot is equipped with a camera measuring the distance, angle and orientation to the leader robot (Fig. 1(b)). Each follower robot autonomously decides its own action in order to have the leader robot in the desired location throughout the navigation. Each follower robot select one neural controller, controlling the robot to have the leader robot in the target position. During navigation, the robot can switch to multiple neural controllers resulting in different formations.

III. Neural Architecture

Fig. 2 shows the fully connected feed-forward neural network composed of 3 input, 2 hidden and 2 output units. The distance, angle, and orientation of the follower robot relative to the leader robot are the inputs of the neural controller in the case of LBF (leader robot relative to the follower robot in the case of FBF). The discrete integer values of distance sensor varies from 0 to 12 where 0 corresponds to closest distance and 12 to 2.5 m, which is the maximum distance the robot can be detected using the visual sensor. In the real robot implementation, the distance to the robot is calculated based on the pixel number. The camera sensor has a wide visual field (120º). The visual field is divided in 21 sections (each section 5.7º). Therefore, the egocentric angle to the robot varies from 0 to 21 where 0 corresponds to 60° to the right and 21 is 60° to the left. Random noise, uniformly distributed in the range of +/- 5% of sensor readings, has been added to the calculated angle and +/- 10% to the calculated distance. The activation function of hidden and output units is the sigmoid function:

$$y_i = \frac{1}{1+e^{-x_i}} \tag{1}$$

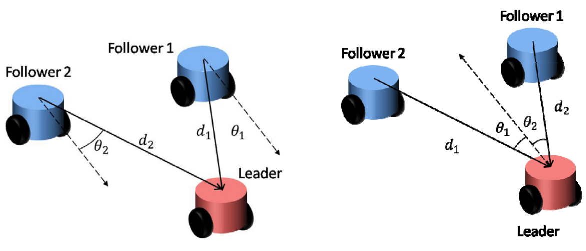


Figure 1. Formation task: (a) LBF and (b) FBF.

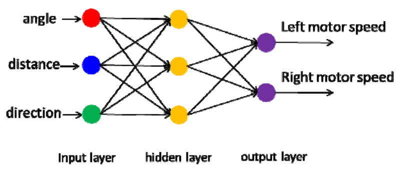


Figure 2. Applied Neural Network.

where the incoming activation for node i is:

$$x_i = \sum_j w_{ji} y_j \tag{2}$$

and j ranges over nodes with weights into node i.

The output units directly control the right and left wheel angular velocities where 0 corresponds to no motion and 1 corresponds to full-speed forward rotation. The left and right wheel angular velocities, w_r and w, are calculated as:

$$\omega_{right} = \omega_{\max} * y_{right}$$
$$\omega_{left} = \omega_{\max} * y_{left} \tag{3}$$

where $\omega_{\max}$ is the maximum angular velocity and y_{right} and y_{left} are the neuron outputs. The maximum forward velocity is considered to be 0.5 m/s.

IV. Robot Formation as a Multiobjective Optimization Problem

A. *Multiobjective Optimization Problem*

In multiobjective optimization problems there are many (possibly conflicting) objectives to be optimized, simultaneously. Therefore, there is no longer a single optimal solution but rather a whole set of possible solutions of equivalent quality. Consider without loss of generality the following multiobjective maximization problem with *m* decision variables, *x* parameters and *n* objectives:

$$y = f(x) = (f_1(x_1, \ldots\ldots x_m), \ldots, f_n(x_1, \ldots\ldots x_m)) \tag{4}$$

where $x = (x_1, \ldots\ldots x_m) \in X$, $y = (y_1, \ldots\ldots y_n) \in Y$ and where *x* is called decision parameter vector, *X* parameter space, *y* objective vector and *Y* objective space. A decision vector $a \in X$ is said to dominate a decision vector $b \in X$ (also written as a $\succ$ b) if and only if:

$$\forall i \in \{1, \ldots, n\} : f_i(a) \geq f_i(b) \wedge$$
$$\exists j \in \{1, \ldots, n\} : f_j(a) > f_j(b) \tag{5}$$

The decision vector *a* is called Pareto-optimal if and only if *a* is nondominated regarding the whole parameter space *X*. Pareto-optimal parameter vectors cannot be improved in any objective without causing degradation in at least one of the other objectives. They represent in that sense globally optimal solutions. Note that a Pareto-optimal set does not necessarily contain all Pareto optimal solutions in *X*. The set of objective vectors corresponding to a set of Pareto-optimal parameter vectors is called "Pareto-optimal front".

Different approaches to relate the fitness function to the objective function can be classified with regard to the first issue. The second problem is usually solved by introducing elitism and intermediate recombination. Elitism is a way to ensure that good individuals do not get lost (by mutation or set reduction), simply by storing them away in an external set, which thereafter only participates in selection. Intermediate recombination, on the other hand, averages the parameter vectors of two parents in order to generate one offspring, which helps to achieve a good distribution of the Pareto-optimal set.

B. *Nondominated Sorting Genetic Algorithm II*

NSGAII was employed to evolve the neural controller where the weight connections are encoded as real numbers. In [13], the authors compared the NSGAII with four other multiobjective evolutionary algorithms using two test problems. The NSGAII performed better than the others did, showing that it can be successfully used to find multiple Pareto-optimal solutions. In NSGAII, before selection is performed, the population is ranked on the basis of domination using Pareto ranking, as shown in Fig. 3. All nondominated individuals are classified in one category with a dummy fitness value, which is proportional to the population size [14]. After this, the selection, crossover, and mutation operators are performed.

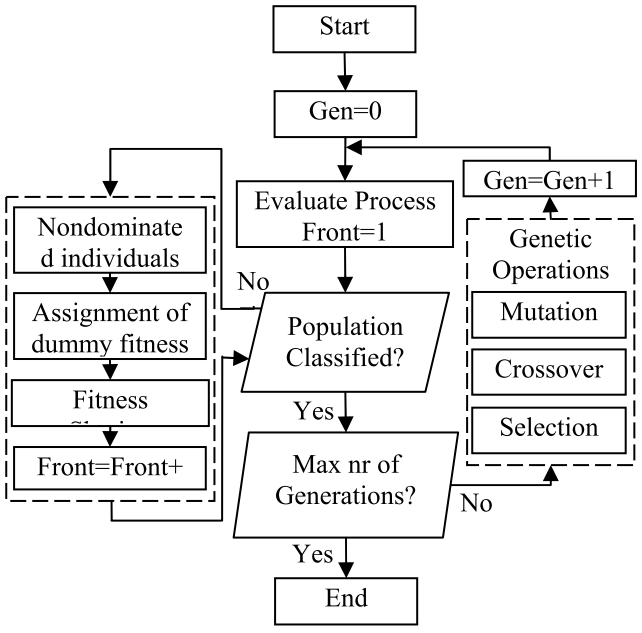


Figure 3. Flowchart of NSGAII.

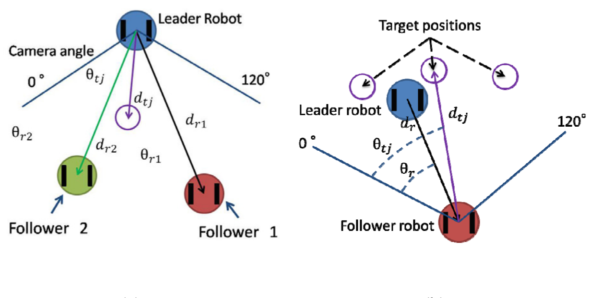


Figure 4. Formation task: (a) LBF and (b) FBF.

C. *Fitness functions*

The selection of the fitness function influences the performance of MOEA. It is important that the objective functions conflict each other as much as possible. In the multirobot formation task, the number of objective functions to be optimized is the same with the number of the follower robots. The objective functions conflict each other because as the robot reach the target location one objective function improves while the other deteriorates.

The leader and follower robot move in the environments 45 s (450 time steps, time step 0.1s). During this time the leader robot moves in the environment. For any evolutionary computation technique, a chromosome representation is needed to describe each individual in the population. The genome of every individual of the population encodes the weight connections of the neural controller. The connection weights range from -5 to 5.

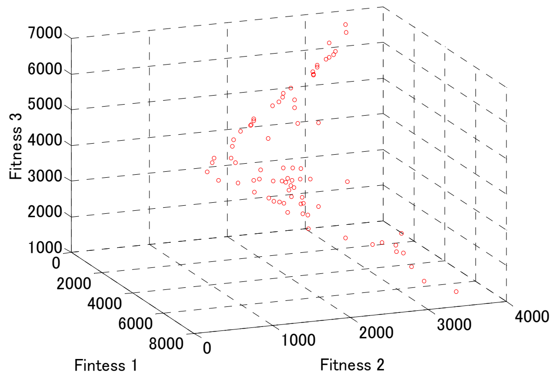
The target distance d_t between the leader and follower robots is considered 1.25m, while the target angles are 28°, 62° and 97°. During evolution each neural of the population controls two robots motion that are initially randomly placed in the environment. At the end of its lifetime the fitness function for each target position is calculated as the average of the fitness functions of both robots. In the case of LBF (Fig. 4(a)), in order to minimize the difference between the target and real position relative to the leader robot, the fitness f_j of each target position (j=1~3), is calculated as follows:

$$f_j = \sum_{k=1}^{2}\left[\sum_{i=1}^{\max\_st}\left(\left|{}_j d_t^i - d_r^i\right| + \left|{}_j\theta_t^i - \theta_r^i\right|\right)\right]/2 \qquad (6)$$

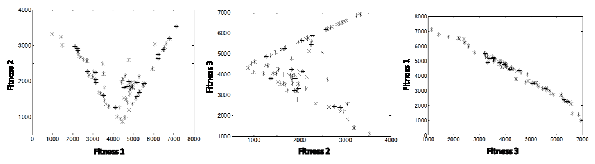
where *max_st* is the maximum number of steps, d_r and ${}_jd_t$ are the real and the target distance, and θ_r and ${}_j\theta_t$ are the real and target angle, and *k* is the number robots during evolution. In FBF, the fitness is the total sum of the difference between the target and real position relative to the leader robot (Fig. 4(b)). If an individual happens to get out of the visual field or hit the leader robot, the trial is terminated and a low fitness is assigned. Therefore, such individuals will have a low probability to survive. The following genetic parameters are used: N_{gen}=100,N_{pop}=1000, σ_{shared}=0.4.

V. Results

In this section, the performance of the best neural controllers obtained from the MOEA for the formation task are discussed. The results of the LBF and FBF are presented where the follower robots have to be positioned in three target locations relative to the leader robot. Fig. 5(a) shows the nondominated optimal front of the last generation, averaged for five different runs for the LBF. Fig. 5(b) shows that the nondominated optimal front has a clear tradeoff between the three objective functions. The extreme solutions represent the best Neural Networks that control each robot to get to the target positions relative to the leader robot and keep this throughout the motion. The neural networks of intermediate solutions control the follower robot to get to positions between the target ones.

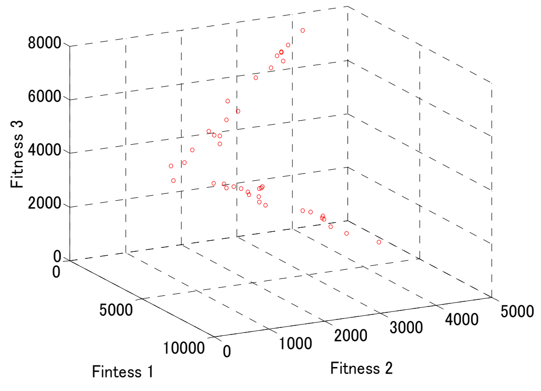


(a)

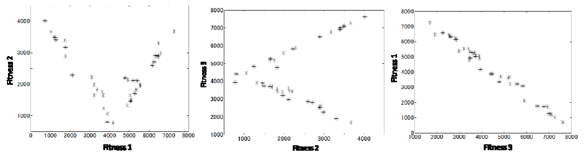


(b)

Figure 5. Pareto front of LBF.



(a)

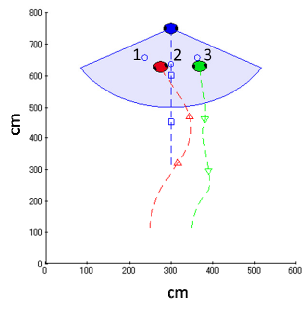


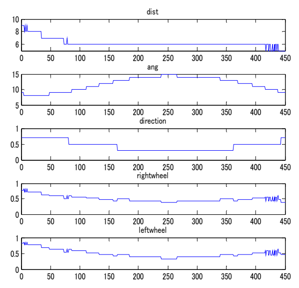
(b)

Figure 6. Pareto front of FBF.

As can be seen from Figure 6, the shape of the Pareto optimal front of LBF and FBF is the same. However, fitness values of FBF are better than LBF.

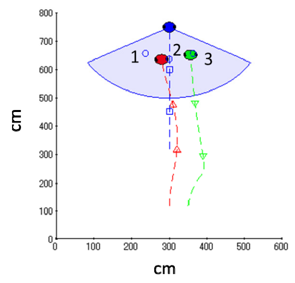
Fig. 7(a) shows the robot motion controlled by the best neural controllers generated by LBF for targets positions 2 and 3. Initially, the distance between the leader and the follower robots is larger than the target one. In addition, the initial angles are different from the target ones. Therefore, the robots move quickly to get into the desired distance relative to the leader robot. The neuron activation (Fig. 7 (b)) shows that the distance between the leader and follower robot is the target one.

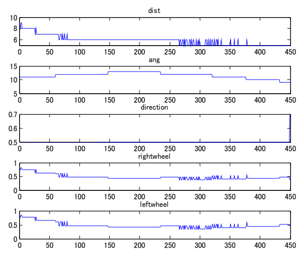




(a) (b)

Figure 7. LBF simulation results. (a) robot trajectory; (b) input and output neuron activation.





(a) (b)

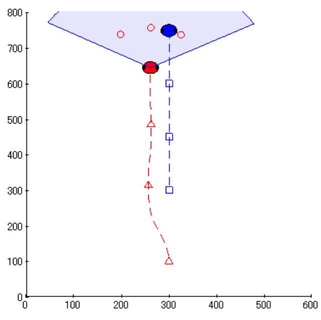
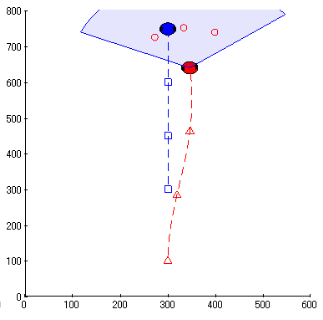
Figure 8. FBF simulation results. (a) robot trajectory; (b) input and output neuron activation.

However, the angle to the follower robot is fluctuating around the target one. Therefore, the follower robot position does not match with the target 2.

In difference, the FBF results (Fig. 8), show a better performance for the same initial positions and orientations of follower robots. A smooth motion of the follower robots to get to the target positions is generated. The error between the robot and target position is smaller throughout the travel.

As already pointed out, an advantage of applying MOEA to evolve neural controllers for multi robot formation is that a large number of optimal neural controllers are generated. In addition to three neural networks that control the robot to get to the three target positions, other neural controllers position the follower robot in other locations relative to the leader robot. Fig. 9 shows the performance of two neural controllers selected from the Pareto front solutions. Fig. 9 (a) and (b) show that the leader robot position is between targets 1 and 2 and targets 2 and 3, respectively

We implemented the evolved optimal neural controllers on the real hardware of the iRobot create, a programmable robot based on Roomba vacuum-cleaning robot. This is a differential drive robot with two powered wheels. Each wheel may be controlled independently with a maximum speed of 0.5m/s. In the case of LBF, the follower robots receive commands through a wireless Bluetooth communication from the leader robot computer. In our experiments, we use a USB wide-view camera (120 degree) to obtain images at a resolution of 640x480 at a refresh rate of 30Hz.

(a) (b)

Figure 9. Performance of Pareto front neural controllers.

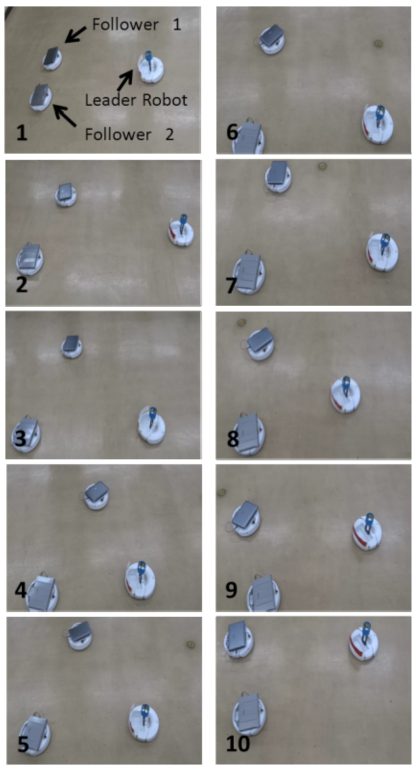


Figure 10. iRobot create implementation.

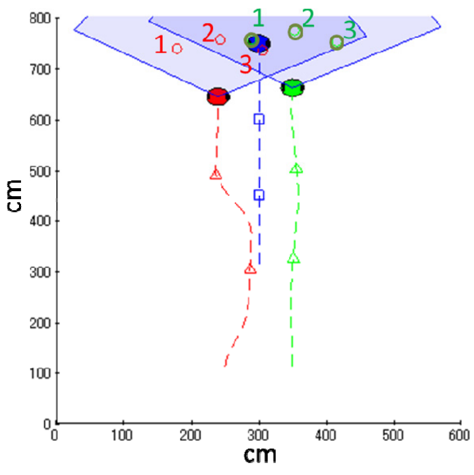


Figure 11. Follower robots switching to different neural controllers..

A two color marker is attached to the top of each robot to calculate the distance, angle and direction of the follower or leader robot.

Because the distance is calculated based on the number of pixels, while the angle based on the blob position, there are some differences between the simulated and real robot performance. However, despite these differences, the robots still performed the formation task well. Even in the experiments, the FBF outperformed the LBF. The results show that two follower robots reached the target positions relative to the leader and kept the formation shape even when the leader robot changed the moving direction, or the moving speed (Fig. 10). One possible reason that FBF performed better than LBF is the time required to process the visual sensor data. In the LBF, the leader robot has to process the visual sensor data to determine the distance, angle and orientation of each follower robot. In addition, the wireless communication makes the follower robot response to state changes slower, resulting in a poor performance.

Simultaneous evolution of multiple neural networks controlling the robots to reach the target position relative to the leader robot, makes it possible that robots switch between neural controllers resulting in different formations. Fig. 11 shows two robots following the leader robot initially in target positions 2 and 1. Then, while one of the follower robots continues to be controlled by the same neural network, the other robot switches to the target 3 position neural controller.

VI. Conclusion

This paper has experimentally investigated the effectiveness of applying MOEAs to address the multiple robot formation problem. In particular, it was demonstrated that in a single run of the MOEA, robust neural controllers for each robot to get to the target position relative to the leader robot and keep it throughout the motion. In addition, the robots can switch to multiple formations by switching to different optimal NNs of Pareto front solutions. The robustness of evolved neural controllers was also tested on the real hardware. The results show that FBF outperformed LBF in terms of the quality of the solution.

For future work, a possible extension is to develop neural controllers for formation task in more dynamic environments, with moving and static obstacles. Therefore, the robots will be equipped also with distance sensors.

References

[1] L. E. Parker, “Alliance: An architecture for fault tolerant multirobot cooperation," IEEE Transactions on Robotics and Automation, vol. 14, no. 2, pp. 220-40, 1998.

[2] J. P. Desai., J. P. Ostrowski, and V. Kumar, “Modeling and control of formations of nonholonomic mobile robots," IEEE Transactions on Robotics and Automation, vol. 17, no. 6, pp. 905-908, 2001.

[3] R. W. Beard, J. Lawton, and F. Y. Hadaegh, “A feedback architecture for formation control," IEEE Transactions on Control Systems Technology, vol. 9, no. 6, pp. 777-90, 2001.

[4] M. A. Lewis, and K-H. Tan, “High precision formation control of mobile robots using virtual structures,” Autonomous Robots, vol. 4, pp. 387-403, 1997.

[5] N. E. Leonard and E. Fiorelli, “Virtual leaders, artificial potentials and coordinated control of groups," in Proceedings of the IEEE Conference on Decision and Control, Orlando, Florida, pp. 2968-2973, 2001.

[6] P. Ögren, E. Fiorelli, and N. E. Leonard, “Formations with a mission: Stable coordination of vehicle group maneuvers,” Proc.15th International Symposium on Mathematical Theory of Networks and Systems, August, 2002.

[7] T. Balch and M. Hybinette, “Social Potentials for Scalable Multi Robot Formations,” Proc. IEEE International Conference on Robotics and Automation, pp. 73–80, April 2000.

[8] M. Egerstedt and X. Hu, “Formation constrained multi−agent control,”IEEE Transactions on Robotics and Automation, vol. 17, no. 6, pp. 947–951, 2001.

[9] K. Hirota, T. Kuwabara, K. Ishida, A. Miyanohara, H. Ohdachi, T. Ohsawa, W. Takeuchi, N. Yubazaki, M. Ohtani, Robots moving in formation by using neural network and radial basis functions, in: Proceedings of the 1995 International Conference on Fuzzy Systems, vol. 5, pp. 91–94, 1995.

[10] C.A. CoelloCoello, D.A. Van Veldhuizen, and G.B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002.

[11] G. Capi, Multiobjective Evolution of Neural Controllers and Task Complexity, IEEE Transactions on Robotics, 23(6), 1225-1234, 2007.

[12] G. Capi, Z. Mohamed, Multiple Robots Formation-A Multiobjctive Evolution Approach, Procedia Engineering 41 (2012), pp. 156-162.

[13] C.M. Fonseca and P.J. Fleming, “An overview of evolutionary algorithms in multiobjective optimization,” *Evolutionary Computation*, vol. 3, pp. 1-16, 1995.

[14] N. Srivinas, and K. Deb, “Multiobjective optimization using non-dominated sorting in genetic algorithms”, *Evolutionary Computation*, vol. 2, no. 3, pp. 279-285, 1995.