SWARM 2017: The 2nd International Symposium on
Swarm Behavior and Bio-Inspired Robotics
October 29-November 1, Kyoto, Japan

Automatic Design Method of Probabilistic Finite State Machine Using PSO for Swarm Robots Aggregation

Yoshiaki Katada[1]†

[1]Department of Electrical and Electronic Engineering, Setsunan University, Osaka, Japan
(Tel: +81-72-839-9148; E-mail: katada @ ele.setsunan.ac.jp)

Abstract: This paper proposes to use evolutionary computations to design probabilistic finite state machine for the controller on an aggregation problem of swarm robotics. This problem formulated as an optimization problem was solved by the PSOs. Several computer simulations were conducted to investigate the validity of the proposed method. The results obtained in this paper show to us that the proposed method is useful for the aggregation problem and the best optimized controllers are interpretable. This would be transferable to real swarm robots problems.

Keywords: Evolutionary Swarm Robotics, Aggregation, Probabilistic Finite State Machine.

1. INTRODUCTION

Swarm Robotics (SR) [1, 2] have attracted much research interest in recent years. Generally, the tasks in SR are difficult or inefficient for a single robot to cope with. Thus, SR and multi-robot systems overlap each other. Sahin [3] enumerated several criteria[1] for distinguishing swarm robotics as follows:

- autonomy: Each robot should be physically embodied and situated.
- redundancy: Group sizes accepted as swarms is 10 to 20.
- scalability: SR system should be able to operate under a wide range of group sizes.
- simplicity: Each robot should employ cheap design, that is, the structure of a robot would be simple and the cost for it would be cheap.
- homogeneity: SR system should be composed of homogeneous individuals. This enhances the above 2nd and 3rd criterion.

Following the last criterion, homogeneous controllers for individuals are desirable for SR systems. Additionally, this approach does not assume the existence of an explicit leader in swarm robots due to the above criteria. This results in that a collective behavior emerges from the local interactions among robots and between the robots and the environment. Therefore, SR systems are required for that individuals show various behaviors through those interactions although the individuals are homogeneous.

Following the taxonomies proposed by Brambilla[2], we have the most common design methods to develop swarm robotics systems: *behavior-based design* and *automatic design*. In behavior-based design methods, there are further three categories: subsumption architecture design method, probabilistic finite state machine design method, virtual physics-based design method and others. On behavior-based design methods, such swarm robotics systems are developed by hand of the designer until the desired collective behavior is obtained. Such effort to design collective behavior is larger than those to design single robot behavior because the interactions are so complex among robots and between the robots and the environment.

In automatic design methods, there are two categories: *reinforcement learning* and *evolutionary robotics*. In reinforcement learning (RL), an agent aims at building value functions in the process of learning while getting rewards from the environment. Such optimality is guaranteed based on Markovian decision process, e.g., Q-learning. The existence of other robots may violate such an assumption. From the viewpoint of a search problem, RL is considered as a single point search. The ability to search the optimal solution is not so high.

In evolutionary robotics[4], evolutionary computation (EC) is generally applied to design artificial neural networks (ANNs) as a controller of a robot. In the case of SR [1], an evolved controller is commonly used for each individual robot or the controller for each robot is designed individually in the context of cooperative co-evolution. The best evolved neural controller may show good performance on robot control tasks in simulated environment. However, such controllers suffer from the gap between simulated and real environment when the controllers are transferred to real environment[5-9]. This is due to uncertainty derived from real environment: noise, complexity and nonlinear dynamics of ANNs. When we adjust parameters of ANN due to such uncertainty, we find it hard to interpret the collective behavior emerged from the best evolved neural controller.

In order to overcome these problems, we propose to use probabilistic finite state machines instead of ANNs for the controller on a swarm robot control task in the context of ER. That is, we apply evolutionary computations to design a probabilistic finite state machine for the swarm robot controller. Such automatic design of probabilistic finite state machines would decrease the effort of the design of collective behavior for swarm robot drastically.

The paper is organized as follows. The next section describes an EC, particle swarm optimization applied to design the PFSMs. Section 3 describes probabilistic finite

† Yoshiaki Katada is the presenter of this paper.
[1]Sahin [3] claimed that these criteria should be used as a measure of the degree of SR in a particular study.

state machine adopted in a swarm robots control problem. Section 4 defines the swarm robot control problem, *aggregation*[10-12], where the probabilistic finite state machines designed by the PSO are evaluated. Section 5 gives the results of our computer simulations. Conclusions are given in the last section.

2. PARTICLE SWARM OPTIMIZATION (PSO)

Kennedy and Eberhart [13] introduced particle swarm optimization (PSO) that was based on social behavior of fish schooling or bird flocking. The PSOs have been applied to various fields so far due to the search performance and the simplicity of their algorithms.

The optimization problems solved by the PSOs in this paper are formulated as follows:

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_L)^T \in R^L$.

In the PSOs, a solution in the search space is called, "particle". All the particles in a population have their own positions and velocities. At each iteration, particles are updated. The update method of positions and velocities in the PSO is as follows:

$$\begin{aligned} v_{ij}^{k+1} &= w \cdot v_{ij}^k + c_1 \cdot rand() \cdot (pbest_{ij} - x_{ij}^k) \\ &\quad + c_2 \cdot rand() \cdot (gbest_j - x_{ij}^k), && (2) \\ x_{ij}^{k+1} &= x_{ij}^k + v_{ij}^{k+1}, && (3) \end{aligned}$$

where $rand()$ is a random number between $[0, 1]$, w, c_1 and c_2 are weight factors for the terms. x_{ij}^k is the j-th variable in the position vector of the i-th particle at iteration k, v_{ij}^k is the j-th variable in the velocity vector of the i-th particle. $pbest_{ij}$ is the j-th variable in the best position visited so far by the i-th particle and $gbest_j$ is the j-th variable in the best position visited so far by all the particles. Each particle is evaluated by the objective function $f(\cdot)$ to be optimized. The update method in Eq.(2) is called, "inertia weight model (IWM)", which is employed in this paper.

w, c_1 and c_2 are parameters to tune for good performance of the PSO. The several methods to tune those parameters have been proposed. In this paper, linearly decreasing inertia weight method (LDIWM) was employed as well as IWM. In LDIWM, w is not constant but time-varying according to the increase of iterations:

$$w(k) = w_{\max} - \frac{w_{\max} - w_{\min}}{k_{\max}} \times k, \tag{4}$$

where $w_{\max}$ and $w_{\min}$ are the upper and lower bounds of inertia weight, and $k_{\max}$ is the maximum number of iterations.

3. PROBABILISTIC FINITE STATE MACHINE (PFSM)

3.1. Definition of the PFSM

There are many literatures in swarm robotics employing a controller to be considered as a class of finite state machine with probabilistic transitions. Brambilla[2] classified those controllers as probabilistic finite state machines (PFSMs). There are several variants of the PFSMs employed in swarm robotics [2] .

In the PFSMs employed in SR, we set several states for an individual robot. Let $S = \{S_1, S_2, \cdots, S_m\}$ be a finite set of m states. Each state may correspond to a module[16], a primitive of the domain ontology[17] or a basic behavior[11]. A state transits to the next state with a probability (Fig. 1). We can describe those probabilities as a state transition probability matrix. This is defined as follows:

$$\begin{aligned} \boldsymbol{P} \;=\; \{p_{ij} \in \Re \mid 0 \le p_{ij} \le 1,\ \sum_{j=1}^{m} p_{ij} = 1, \\ i, j \in \{1, 2, \cdots, m\}\}, \end{aligned} \tag{5}$$

where $\boldsymbol{P} \in \Re^{m \times m}$ is a matrix in which each entry p_{ij} is the transition probability from state i to state j.

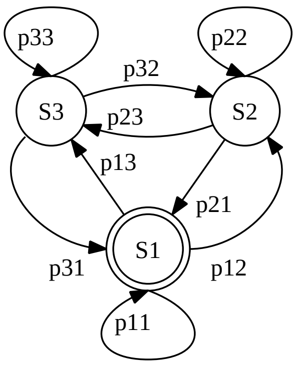


Fig. 1 State transition diagram of the PFSM

3.2. Automatic design of the PFSM

The PFSM is optimized as the design variables in an optimization problem by the PSO. The optimization problem is formulated as follows:

$$\begin{aligned} \min \quad & f(p_{ij}), && (6) \\ \text{subject to} \quad & 0 \le p_{ij} \le 1, \quad i, j = 1, 2, \cdots, m, && (7) \\ & \sum_{j=1}^{m} p_{ij} = 1, \quad i = 1, 2, \cdots, m. && (8) \end{aligned}$$

According to Eq. (5), $|\boldsymbol{P}| = m \times m$. Thus, the number of design variables is $m \times m$. In the process of the PSO, initial particles $\boldsymbol{x}^0$s are generated in the range $[0, 1]$. According to Eqs. (2) and (3) for $k > 0$, $\boldsymbol{x}^k$s would violate the constraints, Eqs. (7) or (8). Thus, we must consider these two constraints in order not to leave $\boldsymbol{x}^k$ unfeasible.

These constraints can be replaced with the following constraints.

$$0 \le p_{ij}, \tag{9}$$

$$p_{ij} \le 1 \;\cap\; \sum_{j=1}^{m} p_{ij} = 1, \quad i, j = 1, 2, \cdots, m. \tag{10}$$

[2]We need discussions whether those are the PFSMs anywhere.

In order for p_{ij} to satisfy Eq. (9), we apply *mirroring* to p_{ij} as follows:

$$p'_{ij} = \begin{cases} -p_{ij} : & \text{if } p_{ij} < 0 \\ p_{ij} : & \text{otherwize.} \end{cases} \tag{11}$$

p'_{ij} is kept in the design variables for next update. After that, to satisfy Eq. (10), we normalize p'_{ij} as follows:

$$p''_{ij} = \frac{p'_{ij}}{\sum_{j=1}^{m} p'_{ij}}. \tag{12}$$

p''_{ij} is used only for evaluations of the objective functions and is not kept in the design variables for next update.

4. SWARM ROBOT CONTROL PROBLEM AND THE OBJECTIVE FORMULA

Swarm robot control problem in this paper is set to be aggregation[10] where robots aggregate to an arbitrary position as closely as possible from their initial positions. Aggregation is considered to be a building block for applications so that the task of aggregation is often used as a case study [11][12].

In this paper, two performance measures were employed according to the reference[12]: *dispersion metric* and *cluster metric*. The dispersion metric to be minimized is as follows:

$$f_1 = \frac{1}{4r^2} \sum_{i=1}^{N} ||\boldsymbol{p}_i^{(t)} - \overline{\boldsymbol{p}}^{(t)}||^2, \tag{13}$$

where r is the radius of a robot, $\boldsymbol{p}_i^{(t)}$ is the position of the i-th robot at time t, $\overline{\boldsymbol{p}}^{(t)}$ is the center of the positions of all the robots.

The cluster metric is as follows:

$$c = \frac{\text{number of robots in the largest cluster at time } t}{N}, \tag{14}$$

where a cluster is defined as a maximal connected subgraph, where two robots are adjacent if the distance between the centers of them is less than $4r$.

Thus, the 2nd objective function to be minimized is as follows:

$$f_2 = \frac{1}{c} \ . \tag{15}$$

The objective functions are evaluated when t is the final time step in a trial.

5. COMPUTER SIMULATION

5.1. Setting of computer simulations

5.1.1. Simulated swarm robot and environment

According to the previous experiment using the swarm mobile robots [14], the setting of computer simulations was as follows. 10 differential wheeled robots (Fig. 2) were used in this experiment. The robot's diameter and height are approximately $D = 170$ mm and $H = 75$ mm, respectively. The robot is equipped with four infrared distance sensors located at the front of the body for measuring the distance to other robots and walls (Fig. 3 (right)), and four distance sensors located at the front of the body for detecting other robots (Fig. 3 (left)). The maximum detection ranges of the former sensor and the latter sensor are 300 mm and 5 m, respectively.

The simulated environment is a square arena with walls (Fig. 4). The length of the wall was set at 10 m. At the beginning of each trial, swarm robots were always placed at the same initial position at random orientations. One trial ends when $30,000$ steps (300 sec) are performed. We conducted 10 independent runs varying initial orientations.

Open Dynamics Engine (ODE) [15] was employed in order to consider dynamics of robots and the interaction

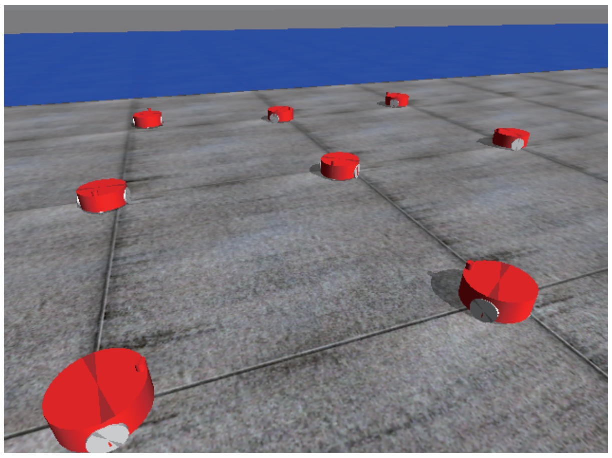


Fig. 2 Setup for swarm mobile robots

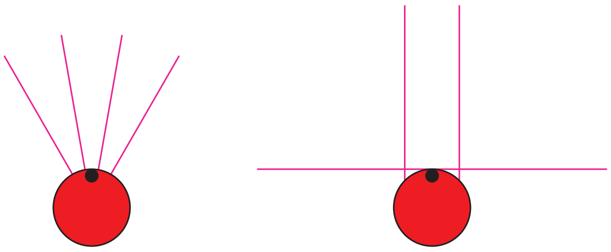


Fig. 3 Distance sensor configurations: (left) sensors for other robots, (right) sensors for obstacles

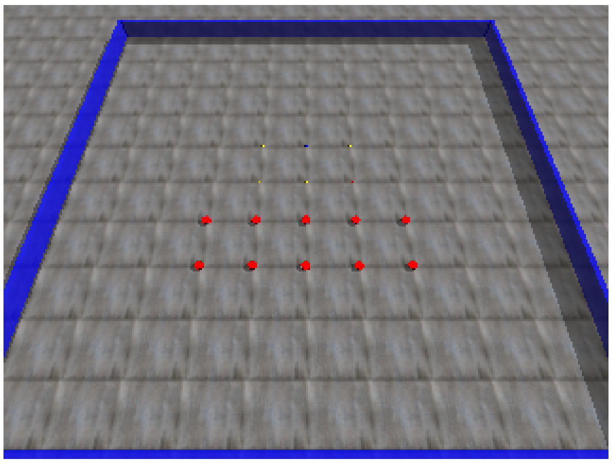


Fig. 4 Set up for computer simulation: Initially, swarm robots are always placed at the center.

between robots and environment.

5.1.2. Controller

Fig. 5 shows the controller employed in this experiment. This takes its inspiration from the reference [11]. This controller is composed of the PFSM, a random walk module, an avoidance module, approach module and stop module. In the PFSM (Fig.6), we have three states: $\{$*Approach*, *Wait*, *Repel*$\}$. Thus, $m = 3$ and $|\boldsymbol{P}| = 9$.

- $Wait$: the robot stops immediately.
- $Approach$:
  - If the controller detects other robots, the robot approaches to the detected robots. When the distance between the robots is less than the threshold[3], the state transits to $Wait$ to stop the robot.
  - If the controller detects not a robot but obstacles, the avoidance module becomes active.
  - If the controller detects neither robot or obstacle, the random walk module becomes active.
- $Repel$:
  - If the controller detects other robots or obstacles, the avoidance module becomes active.
  - If the controller detects no obstacles, the random walk module becomes active.

[3]The threshold value was set to D mentioned in Section 5.1.1.

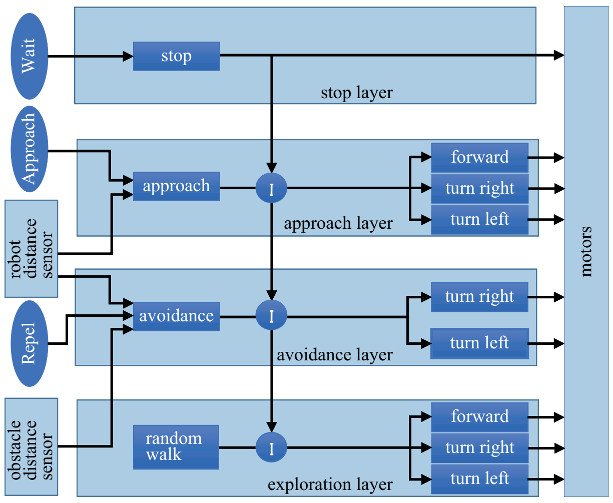


Fig. 5 Subsumption architecture[16] with the PFSM

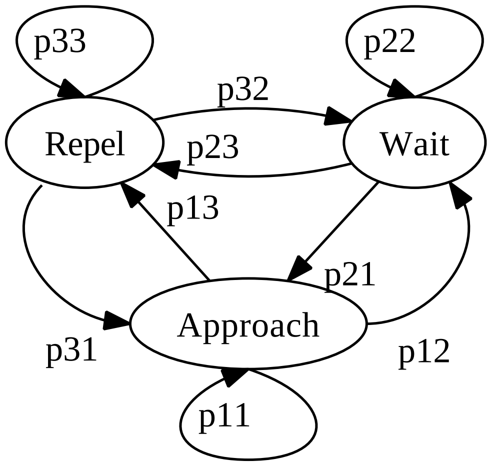


Fig. 6 State transition diagram of the PFSM for aggregation

In the random walk module, the whole steps are divided into the rotation phase and the move-forward phase [14]. In the rotation phase, the controller determines the direction of rotation and selects an angle of rotation randomly from $\{45, 90, 135\}$ degree. Then, a robot rotates until reaches the desired angle. In the move-forward phase, a robot moves forward driving two wheels. Its one step is set in two way: *constant* and *Gaussian*. For *constant random walk*, one step is set at 0.1 sec due to that the physics is updated at a rate of 10 times per control cycle. For *Gaussian random walk*, one step is determined according to Gaussian distribution.

5.1.3. Setting of the PSO

Computer simulations were conducted using particles of size 20. The number of design variables was $L = 9$ mentioned above. The PSO was employed to design the PFSM parameters. The parameters of the PSO adopted in this experiment are as follows: $c_1 = 2.0, c_2 = 2.0$, $w = 0.5$ in Eq.(2) for IWM and $c_1 = 2.0, c_2 = 2.0$, $w_{\min} = 0.4, w_{\max} = 0.9$ in Eq.(4) for LDIWM. Each run lasted 200 iterations. We conducted 10 independent runs for each condition. All results were averaged over 10 runs.

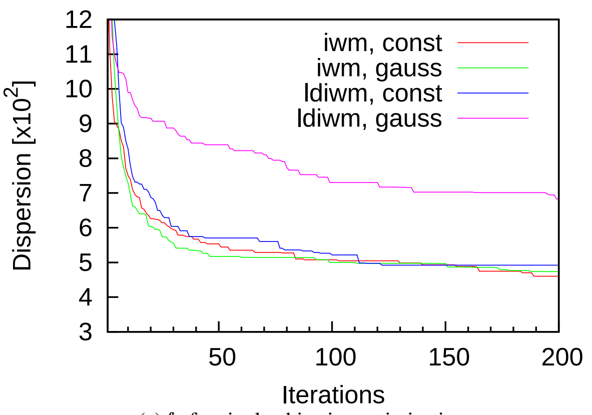
We conduct two types of optimization problem;
- *single objective optimization (single obj)* We employed a single objective function, f_1, in Eq.(13).
-two objective optimization (two obj) In addition to f_1, we employed f_2 in Eq.(15) in order to decrease the number of clusters. For this multiobjective optimization, we applied Lexicographic ordering[18] where f_2 has priority over f_1; When $f_2(\boldsymbol{P}) < f_2(\boldsymbol{P}')$ or $f_2(\boldsymbol{P}) > f_2(\boldsymbol{P}')$, f_1 is not evaluated. When $f_2(\boldsymbol{P}) = f_2(\boldsymbol{P}')$, $f_1(\boldsymbol{P})$ and $f_1(\boldsymbol{P}')$ are compared for updating the position in the memory of the PSO. This priority considers the discreteness of f_2.

5.2. Experimental Results

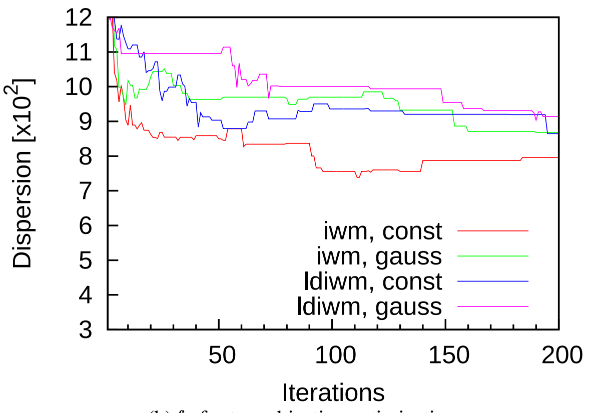
5.2.1. Objective function values

Figs. 7(a) and 7(b) show the best objective function values of f_1 for each iteration for *single obj* and *two obj*, respectively. The IWM shows good performance regardless of the type of random walk. The LDIWM for Gaussian random walk shows the worst performance among them. The objective function values in Fig. 7(a) were better than those in in Fig. 7(b) for each condition.

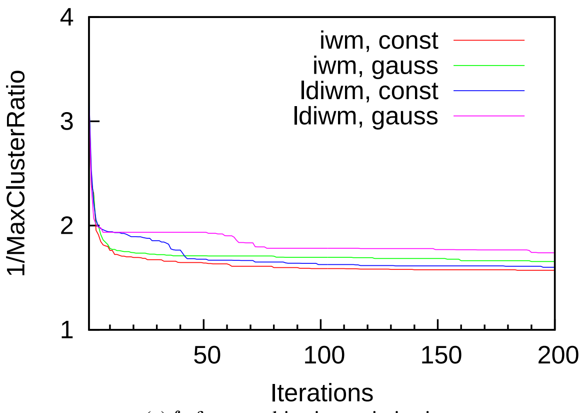
Fig. 7(b) shows *non-elitism*, that is, the objective function values can not keep their best values in the previous iteration. This is due to Lexicographic ordering where f_2 has priority over f_1. Fig. 7(c) shows the best objective function values of f_2 for each iteration for *two obj*. The same tendencies were observed as those in Fig. 7(a). The differences among the conditions were small at the last iteration. No condition was found where f_2 converges to 1. This means that the swarm robots did not aggregate into a single cluster.



(a) f_1 for single objective optimization



(b) f_1 for two objective optimization

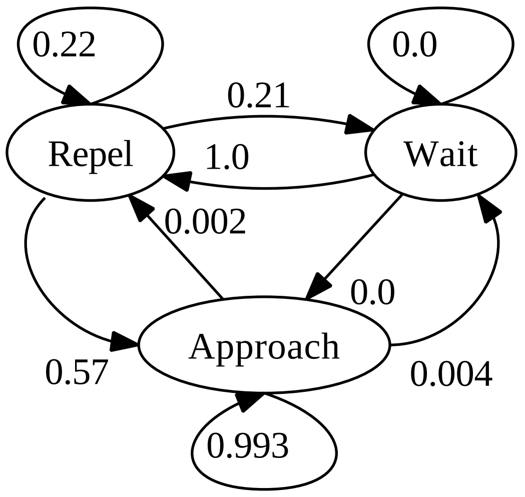


(c) f_2 for two objective optimization

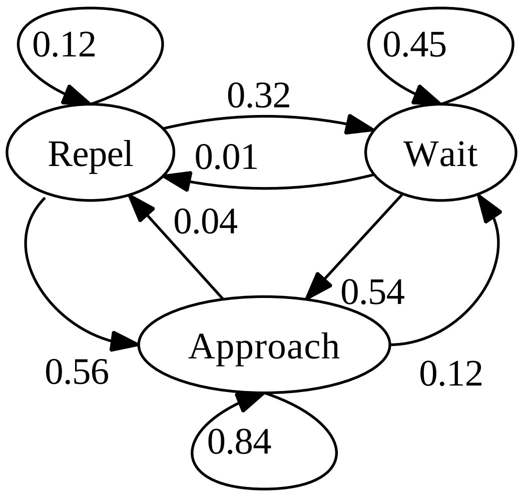
Fig. 7 Objective function values for each iteration

Table 1 Performance of the controllers obtained by the IWM for constant random walk

metric	f_1			c		
statistics	best	avg	worst	best	avg	worst
single obj	139	674	2515	0.8	0.31	0.1
two obj	123	927	2505	0.9	0.58	0.3



(a)For single objective optimization



(b)For two objective optimization

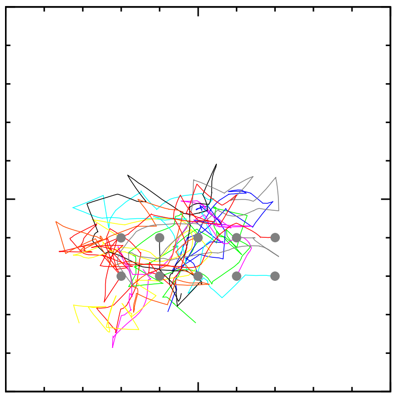
Fig. 8 State transition diagram of the best optimized PFSM

5.2.2. Performance of the best optimized controller

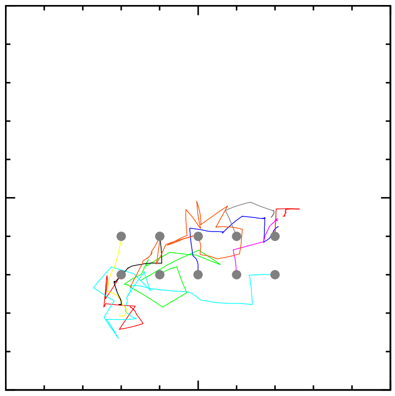
In this section, we investigate the performance of the best optimized controllers obtained in the previous section. The IWM for constant random walk shows the best performance for *single obj* and *two obj*. Table 1 shows the performance in 50 trials on the dispersion (f_1 in Eq.(13)) and the ratio of the number of robots in the largest cluster (c in Eq.(14)). The controller designed in *single obj* shows smaller dispersion than the one designed in *two obj*. The controller designed in *two obj* tends to form the bigger cluster while it keeps relatively large dispersion.

Figs. 8(a) and 8(b) show the state transition diagram of the best optimized PFSMs for *single obj* and *two obj*, respectively. Each PFSM has its unique characteristics; In the PFSM for dense aggregation (Fig. 8(a)), *Wait* always transits to *Repel*. This means an instant start even if a state transits to *Wait*. Thus, swarm robots approach to each other most of the time and they rarely repel. In the PFSM for a large cluster (Fig. 8(b)), a state sometimes transits to *Wait*. *Wait* transits to *Approach* or itself. This tendency might give the ability to form larger clusters.

These tendencies are also confirmed in the trajectories



(a)For single objective optimization



(b)For two objective optimization

Fig. 9 Trajectories of the swarm robots with the best optimized PFSM: Gray points show the initial positions of the robots.

of the swarm robots. Fig. 9(a) shows the trajectories of the swarm robots with the PFSM in Fig. 8(a). The robots frequently moved around to approach the other robots. On the contrary, the robots with the PFSM in Fig. 8(b) stayed at a certain place as well as moved around.

6. CONCLUSIONS

This paper proposed to design a probabilistic finite state machine for a swarm robot controller on an aggregation problem by using the PSO. Several computer simulations were conducted to investigate the validity of the proposed method. The results obtained in this paper show that the proposed method is useful for the aggregation problem. The collective behavior of the swarm reflected the characteristics of the best optimized PFSMs.

The two objective functions employed in this paper are not in the relationship of trade-off which is found in general multi-objective optimization problems. Those objectives can coincide because a single cluster brings small dispersion. In near future, another PSO can be applied to generate selection pressure towards the Pareto front.

As confirmed in Section 5, the best optimized PFSM is interpretable. This means that the PFSM would be transferable to real environment and readjustable after its transfer. Future work will investigate these potential abilities of the propose method in real robot experiments. In literature, there are several swarm robotics control tasks where the controller with the PFSM is employed. I would like to apply the proposed method to those control tasks.

REFERENCES

[1] V. Trianni, *Evolutionary Swarm Robotics*, Springer-Verlag, 2008.

[2] M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo, “Swarm Robotics: A Review from the Swarm Engineering Perspective,” *Swarm Intelligence*, Vol. 7, No. 1, pp. 1–41, 2013.

[3] E. Şahin, “Swarm Robotics: From Sources of Inspiration to Domains of Application,” Lecture Notes in Computer Science, Volume 3342/2005, pp.10–20, 2005.

[4] S. Nolfi and D. Floreano: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press, 2000.

[5] R. A. Brooks, “Artificial Life and Real Robots”, *In Proceedings of the First European Conference on Artificial Life*, pp.3–10, 1992.

[6] N. Jakobi, “Half-baked Ad-hoc and Noisy: Minimal Simulation for Evolutionary Robotics”, *In Proceedings of the Fourth European Conference on Artificial Life*, pp.348–357, 1997.

[7] O. Miglino, H. H. Lund and D. Nolfi, “Evolving Mobile Robots in Simulated and Real Environments”, *Artificial Life* 2, pp.417–434, 1995.

[8] D. Keymeulen, M. Iwata, K. Konaka, R. Suzuki, Y. Kuniyoshi and T. Higuchi, “Off-line Mode-free and On-line Model-based Evolution for Tracking Navigation Using Evolvable Hardware”, *In Proceedings of the First European Workshop on Evolutionary Robotics*, Springer-Verlag, 1998.

[9] Y. Katada and K. Ohkura, “An Update Method of Computer Simulation for Evolutionary Robotics”, *Intelligent Autonomous Systems* 9 (IAS-9), pp. 357–364, 2006.

[10] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, G. Theraulaz, “Aggregation Behaviour as a Source of Collective Decision in a Group of Cockroach-like-robots”, In Lecture notes in Computer Science: Vol. 3630. *Advances in Artificial Life*, Springer, pp. 169–178, 2005.

[11] O. Soysal, E. Bahçeci, E. Şahin, “Aggregation in Swarm Robotic Systems: Evolution and Probabilistic Control”, Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 15, No. 2, pp. 199–225, 2007.

[12] M. Gauci, J. Chen, W. Li, T. J. Dodd, R. Groß, “Self-Organised Aggregation without Computation”, The International Journal of Robotics Research, Vol. 33, No. 9, pp. 1145–1161, 2014.

[13] J. Kennedy and R. C. Eberhart, “Particle swarm op-

timization", *In Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[14] Y. Katada, A. Nishiguchi, K. Moriwaki and R. Watakabe, "Swarm Robotic Network Using Levy Flight in Target Detection Problem", *Artificial Life and Robotics*, Volume 21, Issue 3, pp. 295–301, 2016.

[15] Open Dynamics Engine (ODE), http://ode.org/.

[16] R. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, pp. 14–23, 1986.

[17] R. Pfeifer and C. Scheier, *Understanding Intelligence*, MIT Press, 1999.

[18] C. A.C. Coello, G. T. Pulido and M. S. Lechuga, "Handling Multiple Objectives with Particle Swarm Optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 256–279, 2004.