Noname manuscript No.
(will be inserted by the editor)

Evolutionary Design Method of Probabilistic Finite State Machine for Swarm Robots Aggregation

Yoshiaki Katada

Received: date / Accepted: date

Abstract This paper proposes to use evolutionary computations to determine the parameters of probabilistic finite state machine controllers for swarm robots. The robots are evolved to perform an aggregation task. This problem was formulated as an optimization problem and solved by the PSO. 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 evolved controllers are feasible as well as 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. Şahin [3] enumerated several criteria[1] for distinguishing swarm robotics as follows:

– autonomy: Each robot should be physically embodied and situated.
– redundancy: Minimum group size 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 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, individuals in SR systems are required to 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: 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 design for collective behavior requires more effort than that for single robot behavior does 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. The optimality is

Y. Katada
Setsunan University, 17-8 Ikeda-Nakamachi, Neyagawa, Osaka 572-8508, JAPAN
Tel.: +81-72-839-9148
Fax: +81-72-839-9148
E-mail: katada @ ele.setsunan.ac.jp

[1] Şahin [3] claimed that these criteria should be used as a measure of the degree of SR in a particular study.

guaranteed based on Markovian decision process, e.g., for Q-learning[4]. 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 (ER)[5], evolutionary computation (EC) is generally applied to design artificial neural networks (ANNs) for robot control. In the case of SR [1], controllers are commonly designed for all the robots or a controller is individually designed for each robot in the context of cooperative coevolution. 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[6–10]. This is due to uncertainty derived from real environment: noise, complexity and nonlinear dynamics of ANNs. When we adjust the parameters of ANNs due to such uncertainty, we find it difficult 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 a neural controller on swarm robotics tasks in the context of ER. That is, we apply evolutionary computations to design a probabilistic finite state machine for swarm robot control. Such automatic design of probabilistic finite state machines would drastically decrease the effort for the design of collective behavior of swarm robots.

The paper is organized as follows. The next section describes an EC, particle swarm optimization applied to determine the parameters of probabilistic finite state machines. Section 3 describes probabilistic finite state machines adopted in a swarm robots control problem. Section 4 defines the swarm robot control problem, *aggregation*[11–13], where the probabilistic finite state machines designed by the PSO are evaluated. Section 5 gives the results of our computer simulations. Section 6 studies scalability of the obtained probabilistic finite state machines. Conclusions are given in the last section.

2 Particle Swarm Optimization (PSO)

Kennedy and Eberhart [14] 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 $\boldsymbol{x} = (x_1, x_2, \cdots, x_L)^T \in \Re^L$.

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:

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

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.

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 machines with probabilistic transitions. Brambilla[2] classified those controllers as probabilistic finite state machines (PFSMs). There are several variants of the PFSMs employed in SR.

In the PFSMs employed in this study, we set several states for an individual robot. Let $S = \{S_1, S_2, \cdots, S_m\}$ be a finite set of m states. Actions in each state may correspond to a module[15], a primitive of the domain ontology[16] or a basic behavior[12]. 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:

$$\boldsymbol{P} = \{p_{ij} \in \Re \mid 0 \le p_{ij} \le 1, \ \sum_{j=1}^{m} p_{ij} = 1, \quad i, j \in \{1, 2, \cdots, m\}\}, \tag{4}$$

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

3.2 Formulation as an optimization problem

In the proposed method, the elements of a state transition probability matrix $\boldsymbol{P}$ are optimized as the design

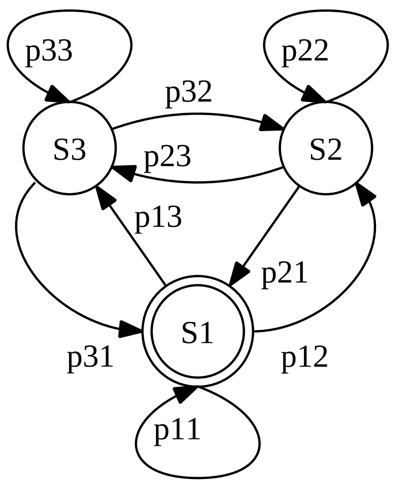


Fig. 1 State transition diagram of the PFSM

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

$$\min \quad f(p_{ij}), \tag{5}$$

$$\text{subject to} \quad 0 \le p_{ij} \le 1, \quad i, j = 1, 2, \cdots, m, \tag{6}$$

$$\sum_{j=1}^{m} p_{ij} = 1, \quad i = 1, 2, \cdots, m. \tag{7}$$

According to Eq. (4), the number of design variables is $L = m \times m$. In the process of the PSO, the variables of the 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. (6) or (7). 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.

$$0 \le p_{ij}, \tag{8}$$

$$p_{ij} \le 1 \ \cap \ \sum_{j=1}^{m} p_{ij} = 1, \quad i, j = 1, 2, \cdots, m. \tag{9}$$

In order for p_{ij} to satisfy Eq. (8), we apply *mirroring* to p_{ij} as follows:

$$p'_{ij} = \begin{cases} -p_{ij} : & \text{if } p_{ij} < 0 \\ \ \ p_{ij} : & \text{otherwise.} \end{cases} \tag{10}$$

p'_{ij} is kept in the design variables for next update. After that, to satisfy Eq. (9), we normalize p'_{ij} as follows:

$$p''_{ij} = \frac{p'_{ij}}{\sum_{j=1}^{m} p'_{ij}}. \tag{11}$$

p''_{ij} is used only for evaluations of the objective function 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[11] 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 an aggregation task is often used as a case study [12][13].

In this paper, two performance measures were employed according to the reference[13]: *dispersion metric* and *cluster metric*. The dispersion metric to be minimized in this paper is as follows:

$$f = \frac{1}{4r^2} \sum_{i=1}^{N} ||\boldsymbol{p}_i^{(t)} - \overline{\boldsymbol{p}}^{(t)}||^2, \tag{12}$$

where r is the radius of a robot, N is the number of robots, $\boldsymbol{p}_i^{(t)}$ is the position of the i-th robot in environment at time t, $\overline{\boldsymbol{p}}^{(t)}$ is the center of the positions of all the robots. In two dimensional case, the minimum value of f for each number of robots is given geometrically in [17].

The cluster metric is as follows:

$$c = \frac{\text{number of robots in the largest cluster at time } t}{N}, \tag{13}$$

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 $4r$.

The above two metrics 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 work using the swarm mobile robots [18], the setting of computer simulations was as follows. Differential wheeled robots (Fig. 2) were used in this experiment. The robot's diameter (D) and height are 0.17 [m] and 0.075 [m], 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)), four light sensors located at the center for detecting other robots' light source (Fig. 3 (left)) and an omni-directional light source located at the center. The light source is always on. The maximum detection range of the infrared sensor is 0.3 [m]. The maximum detection range of the light sensor is R [m], which has directional characteristics (Fig. 4). The light sensor outputs the following value:

$$I_h = \begin{cases} \cos\theta_h \cdot (R - d_h)/(R - D/2) & (d_h \le R) \\ 0 & (d_h > R) \end{cases} \tag{14}$$

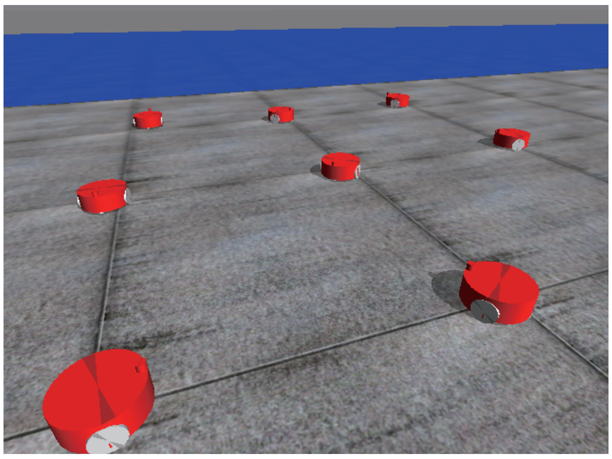


Fig. 2 Setup for swarm mobile robots

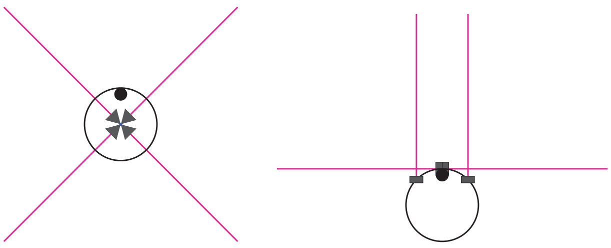


Fig. 3 Sensor configurations: a triangle shows a sensor for light sources, a rectangle shows a distance sensor for robots and walls and a red line shows the sensor direction.

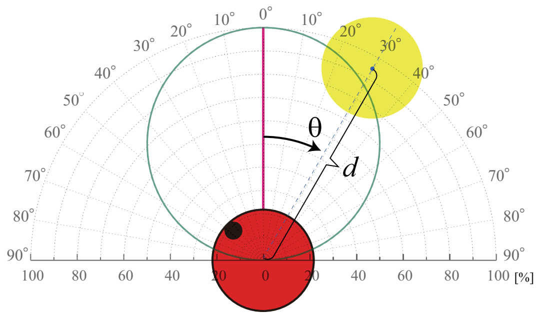


Fig. 4 Relative sensitivity of the light sensor: a red line shows the light sensor direction, a green curve shows directional characteristics and a yellow circle shows a light source.

$$I_s = \sum_{h \in H} I_h \tag{15}$$

where d_h is the distance from the light sensor to the center of the h-th light source, θ_h is a relative angle from the light sensor direction to the h-th light source (Fig. 4) and H is the number of light sources.

The simulated environment is a square arena with walls (Fig. 5). The length of the wall was set to 10 [m]. A series of computer simulations have been conducted varying the number of robots[2] $N_o \in \{10, 20\}$ and the light sensor range $R \in \{1.0, 1.5, 2.0, \ldots, 5.0\}$. At the beginning of each trial, swarm robots were always placed 1 [m] apart at random orientations. One trial ends when 300 steps (300 sec) are performed. We conducted 10 independent runs varying initial orientations.

[2] N_o denotes the number of robots when the PSO was applied to design the PFSM.

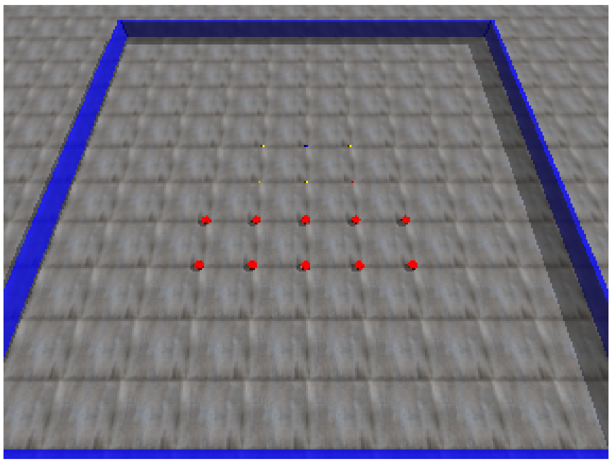


Fig. 5 Set up for computer simulation: Initially, swarm robots are always placed at the center.

Open Dynamics Engine (ODE) [19] was employed in order to consider dynamics of robots and the interaction between robots and environment. Control cycle was set to 1 sec and the physics is updated every 0.01 sec.

5.1.2 Controller

Fig. 6 shows the controller employed in this experiment. This controller is composed of the PFSM, exploration layer, avoidance layer, approach layer and stop layer.

Some layers have the following modules: *forward*, *turn right* or *turn left*, where *forward* means moving forward and *turn right* (*left*) rotating clockwise (counter clockwise) at the position. In the approach layer, the *approach* module sends messages to one of the following modules: *forward*, *turn right* or *turn left* according to the sensory inputs from light sensors described in Section 5.1.1 in order to approach to light sources. In the obstacle avoidance layer, the *avoidance* module sends messages to either *turn right* or *turn left* according to the sensory inputs from infrared distance sensors in order to avoid the walls or other robots which the robot faces. The exploration layer can be explained as the following; In the random walk module, the whole steps are divided into the rotation phase and the move-forward phase [18]. 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 for a few seconds.

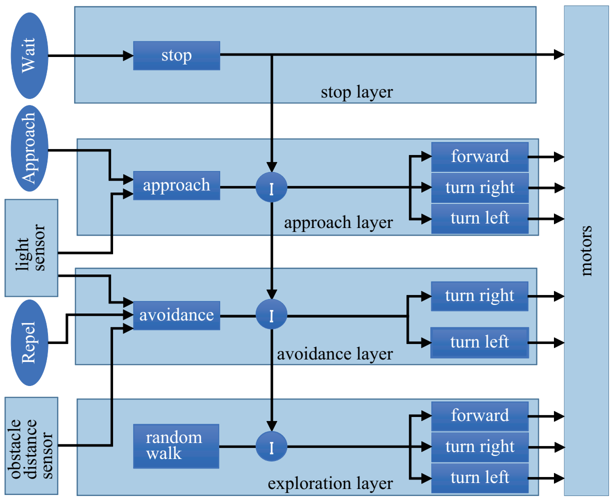


Fig. 6 Subsumption architecture[15] with the PFSM

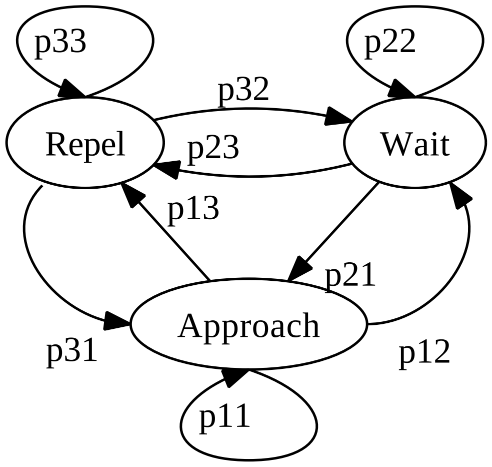


Fig. 7 State transition diagram of the PFSM for aggregation

In the PFSM (Fig.7), we have three states: {*Approach*, *Wait*, *Repel*}. This takes its inspiration from the reference [12]. Thus, $L = 3 \times 3 = 9$. The details are as follows;

- *Approach*:
  - If the controller detects lights emitted by other robots, the robot moves to the brightest direction based on Eq.(15).
  - If the controller detects not other robots but walls, the avoidance module becomes active.
  - If the controller detects neither robot or wall, the random walk module becomes active.
- *Wait*: the robot stops immediately.
- *Repel*:
  - If the controller detects other robots or walls, the avoidance module becomes active.
  - If the controller detects neither robot or wall, the random walk module becomes active.

5.1.3 Setting of the PSO

Computer simulations were conducted using particles of size 20. The number of design variables was $L = 9$ as mentioned above. The PSO was employed to determine 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). Each run lasted 200 iterations. f (Eq.(12)) was employed as an objective function. $1/c$ (Eq.(13)) was measured as the cluster metric. We conducted 10 independent runs for each condition. All results were averaged over 10 runs.

5.2 Experimental Results

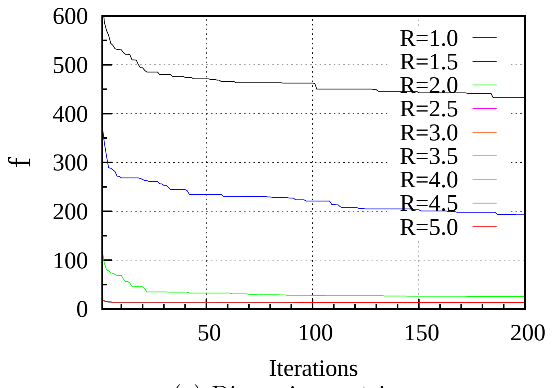
Table 1 shows the best objective function value at the final iteration for each light sensor range, where *theory* denotes the geometrical minimum values of f given in [17]. Thus, the nearly optimal values were obtained when $R \geq 2.5$ for $N_o = 10$ and $N_o = 20$.

Fig. 8 shows the results for each iteration with $N_o = 10$. In Fig. 8(b), the inverse of c is plotted for the cluster metric according to the decrease of f in Fig. 8(a). In Fig. 8, the lines for $R \geq 2.5$ are overlapped so that only the upper line, which is for $R = 5.0$, can be seen. fs decreased early iterations when $R \geq 2.0$. fs were not improved when $R = 1.0$ and 1.5. The cluster metric, $1/c$ converged toward 1.0 when $R \geq 2.5$. That is, robots could aggregate into one cluster when $R \geq 2.5$. These tendencies are also confirmed for $N_o = 20$ in Fig. 9.

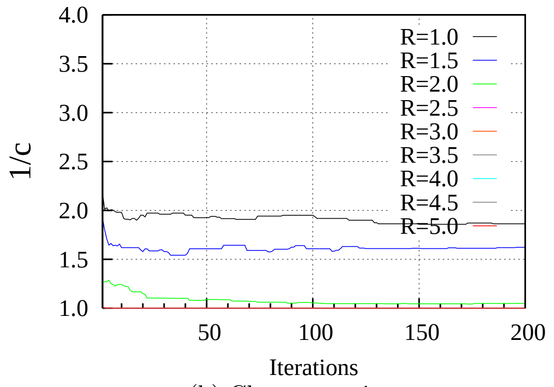
For R, the lower bound for good performance was found to be 2.5. Robots aggregate into a few clusters through observing the behavior when $R < 2.5$. After those aggregations, robots rarely change into one cluster due to the detection range limitation of the sensor.

Table 1 f at the final iteration for each N_o and R in 10 runs

	$N_o = 10$	$N_o = 20$
$R = 1.0$	432.47	1640.34
$R = 1.5$	192.95	725.30
$R = 2.0$	25.78	69.40
$R = 2.5$	13.61	55.35
$R = 3.0$	13.63	55.36
$R = 3.5$	13.67	55.34
$R = 4.0$	13.76	55.19
$R = 4.5$	13.66	55.23
$R = 5.0$	13.67	55.28
theory	13.50	54.65

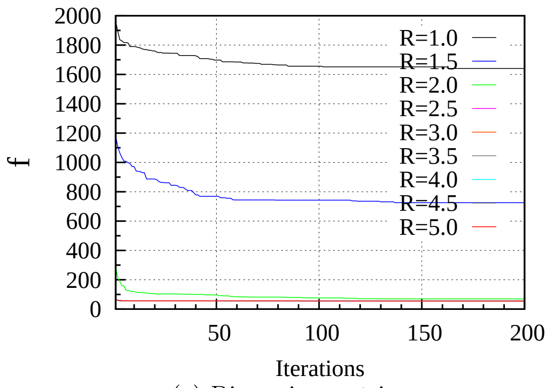


(a) Dispersion metric

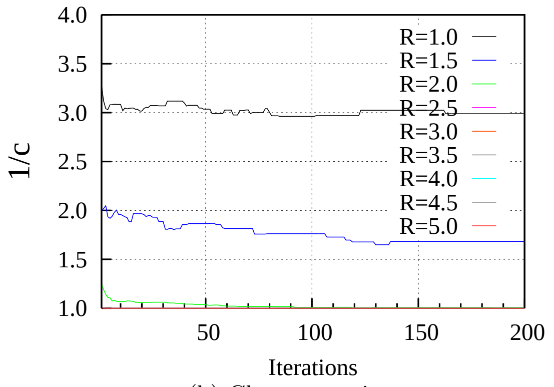


(b) Cluster metric

Fig. 8 Objective function value and cluster ratio for each iteration with $N_o = 10$

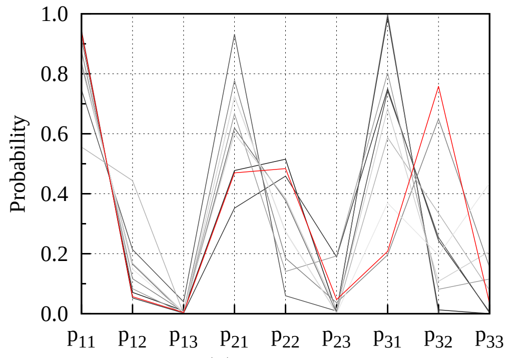


(a) Dispersion metric

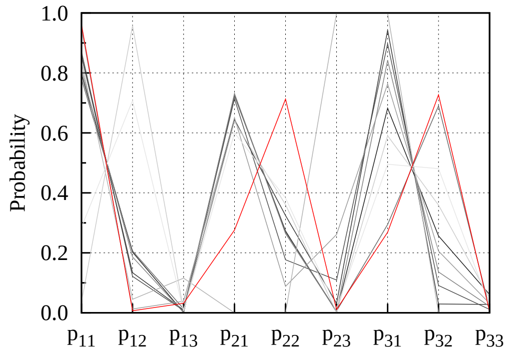


(b) Cluster metric

Fig. 9 Objective function value and cluster ratio for each iteration with $N_o = 20$

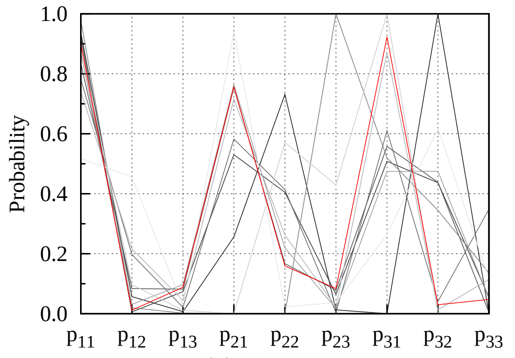


(a) $R = 2.5$

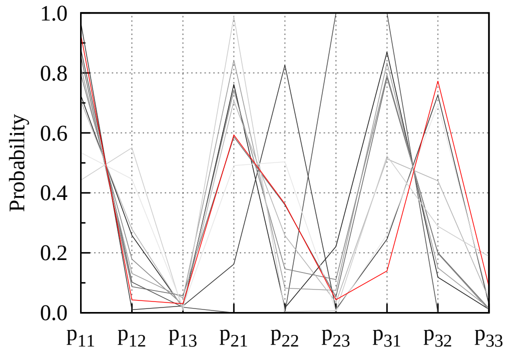


(b) $R = 5$

Fig. 10 State transition probabilities obtained for $N_o = 10$ in 10 runs



(a) $R = 2.5$



(b) $R = 5$

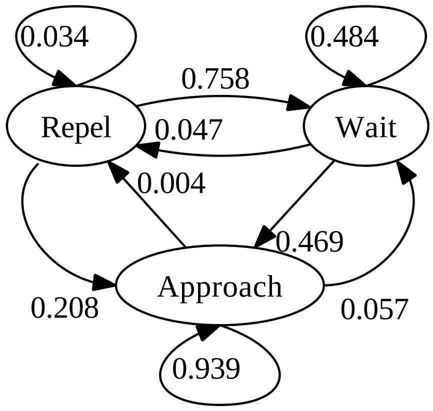
Fig. 11 State transition probabilities obtained for $N_o = 20$ in 10 runs

In the rest of the paper, $R = 2.5$ and 5 were employed for further analyses. Figs. 10 and 11 show the state transition probabilities of the PFSM obtained for $N_o = \{10, 20\}$ and $R = \{2.5, 5\}$ in 10 runs. The indices of p correspond to those in Fig. 7. The darker the line is, the smaller its f is. The red line shows those of the PFSM in the best run. We can find a tendency among those graphs: p_{i1} is high. We explain this from the viewpoint of the aggregation task because p_{i1} is a probability at which a state transits to the state, *Approach*. For the aggregation task, *Approach* is executed frequently. Additionally, p_{22} and p_{32} in some runs are relatively high compared with the other probabilities, p_{23}, p_{33}. The PFSMs obtained in all the runs are not the same for each (N_o, R) although there were no large differences among the runs in the objective function values (Table 2). This means that the nearly optimal PFSM is not unique.

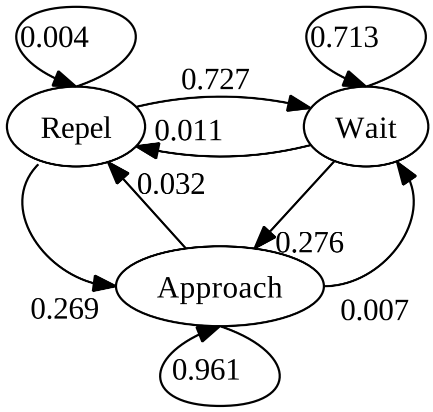
Figs. 12 and 13 show the state transition diagram of the best evolved PFSMs for $N_o = \{10, 20\}$ and $R = \{2.5, 5\}$, respectively. These correspond to the red lines in Figs. 10 and 11. Commonly, the PFSMs have the high probability p_{11}, where *Approach* transits to *Approach*. Besides this, they have the low probabilities p_{12} and p_{13}, where *Approach* rarely transits to *Wait* or *Repel*. This would make it possible to aggregate more closely. These PFSMs have their unique characteristics and no common structure was found for N_o with the same R. By seeing the detail of the PFSM obtained by evolutionary design, we can explain the characteristics of the controller and be convinced by the collective behavior of the swarm.

Table 2 f at the final iteration for each N_o and $R = 2.5, 5.0$ in 10 runs

	$N_o = 10$			$N_o = 20$		
statistics	best	avg	worst	best	avg	worst
$R = 2.5$	13.56	13.61	13.67	55.12	55.35	55.69
$R = 5$	13.54	13.67	13.92	55.00	55.28	55.91

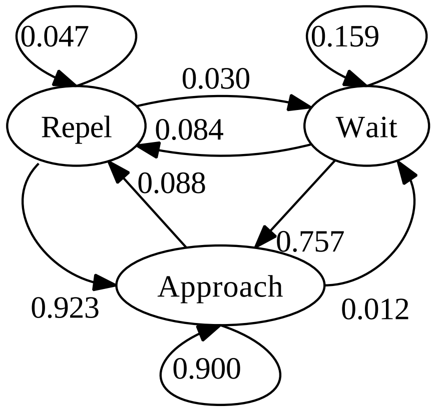


(a) $R = 2.5$

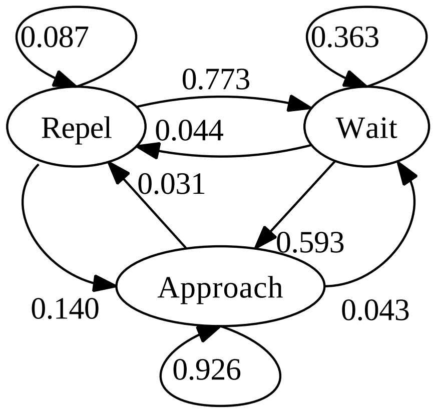


(b) $R = 5$

Fig. 12 State transition diagram of the best evolve PFSM with $N_o = 10$



(a) $R = 2.5$



(b) $R = 5$

Fig. 13 State transition diagram of the best evolved PFSM with $N_o = 20$

6 Scalability of the best evolved controller

This section investigates the scalability of the best evolved controllers obtained in the previous section. Additional computer simulations were conducted using the same setting as the one in Section 5.1.1. The best evolved controllers for $N_o \in \{10, 20\}$ were evaluated varying the number of robots $N \in \{10,\ 20, 30, 40\}$. We conducted 20 independent runs for each parameter setting. Tables 3 and 4 show the average values of f in 20 trials. In each table, *theory* denotes the geometrical minimum value for each N [17], as referred in Section 5.2. Robots aggregated into a single cluster and then fs show the nearly minimum values for almost all N except for $(N, R) = (40, 2.5)$. Even when N is more than N_o, the controller shows good performances. Surprisingly, there were no large differences between when $N_o = 10$ and 20 in fs. When $R = 2.5$ and $N = 40$, robots aggregated into a few clusters.

Table 3 Performance of the best evolved controllers with $N_o = 10$ on f

	N			
	10	20	30	40
$R = 2.5$	13.86	57.44	131.8	4625.97
$R = 5.0$	13.82	56.39	127.7	241.14
theory	13.50	54.65	123.5	220.33

Table 4 Performance of the best evolved controllers with $N_o = 20$ on f

	N			
	10	20	30	40
$R = 2.5$	14.29	56.44	129.4	4868.77
$R = 5.0$	13.95	56.71	128.1	239.71
theory	13.50	54.65	123.5	220.33

7 Conclusions

This paper proposed to use evolutionary computations to determine the parameters of a probabilistic finite state machine for a swarm robot controller. 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 and the best evolved controllers are feasible.

The proposed method depends much on primitive actions initially prepared for states of the PFSMs. Generally, determination or selection of such primitives is important to achieve robot control tasks in the PFSM approach. The concept called *domain ontology and low level specification* by Pfeifer in [16] would be helpful. For basic collective behavior [2, 20–22], several primitive actions have been proposed. These could be also useful for the proposed method.

As confirmed in the last two sections, the best evolved PFSM is interpretable as well as scalable. 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 mentioned above, there are several swarm robotics control tasks where the controller with the PFSM was employed [20–22]. The proposed method will be applied to those control tasks.

References

1. Trianni V (2008) Evolutionary swarm robotics. Springer-Verlag, Berlin
2. Brambilla M, Ferrante E, Birattari M, Dorigo M (2013) Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 7(1):1-41
3. Şahin E (2005) Swarm robotics: from sources of inspiration to domains of application. In: E. Şahin, WM. Spears (Eds.) Swarm Robotics, Springer, pp.10-20 (Lecture Notes in Computer Science 3342)
4. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
5. Nolfi S, Floreano D (2000) Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines. MIT Press, Cambridge
6. Brooks RA (1992) Artificial life and real robots. In: F.J. Varela and P. Bourgine (Eds.) Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life, MIT Press, pp.3-10
7. Jakobi N (1997) Half-baked ad-hoc and noisy: minimal simulation for evolutionary robotics. In Proceedings of the Fourth European Conference on Artificial Life:348-357
8. Miglino O, Lund HH, Nolfi S (1995) Evolving mobile robots in simulated and real environments. Artificial Life 2:417-434
9. Keymeulen D, Iwata M, Konaka K, Suzuki R, Kuniyoshi Y, Higuchi T (1998) Off-line model-free and on-line model-based evolution for tracking navigation using evolvable hardware. In: P. Husbands, JA. Meyer (Eds.) Evolutionary Robotics, First European Workshop EvoRobot 98, Springer, pp.211-226 (Lecture Notes in Computer Science 1468)
10. Katada Y, Ohkura K (2006) An update method of computer simulation for evolutionary robotics. In: T. Arai, et al (Eds.) Intelligent Autonomous Systems 9 IOS Press, pp.357-364
11. Garnier S, Jost C, Jeanson R, Gautrais J, Asadpour M, Caprari G, Theraulaz G. (2005) Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In: M.S. Capcarrere, A.A Freitas, P.J. Bentley, C.G. Johnson, J. Timmis (Eds.) Advances in Artificial Life, Springer, pp.169-178 (Lecture notes in Computer Science 3630)

12. Soysal O, Bahçeci E, Şahin E (2007) Aggregation in swarm robotic systems: evolution and probabilistic control. Turkish Journal of Electrical Engineering and Computer Sciences 15(2):199-225
13. Gauci M, Chen J, Li W, Dodd TJ, Groß R (2014) Self-organised aggregation without computation. The International Journal of Robotics Research 33(9):1145-1161
14. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks, pp.1942-1948
15. Brooks R (1986) A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation 2(1):14-23
16. Pfeifer R, Scheier C (1999) Understanding intelligence. MIT Press, Cambridge
17. Graham RL, Sloane NJA (1990) Penny-packing and two-dimensional codes. Discrete & Computational Geometry 5(1):1-11
18. Katada Y, Nishiguchi A, Moriwaki K, Watakabe R (2016) Swarm robotic network using Lévy flight in target detection problem. Artificial Life and Robotics 21(3):295-301
19. Open Dynamics Engine (ODE). http://ode.org/.
20. Nouyan S, Campo A, Dorigo M (2008) Path formation in a robot swarm – self-organized strategies to find your way home. Swarm Intelligence 2(1):1-23
21. Liu W, Winfield A, Sa J (2009) A macroscopic probabilistic model of adaptive foraging in swarm robotics systems. In Proceedings of 6th Vienna International Conference on Mathematical Modelling
22. Labella TH, Dorigo M, Deneubourg JL (2006) Division of Labor in a group of robots inspired by ants' foraging behavior. ACM Transactions on Autonomous and Adaptive Systems 1(1):4-25