International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 11, November-2015

Robust Swarm Robotics System using CMA-Neuroes with Incremental Evolution

Tian Yu, Toshiyuki Yasuda, and Kazuhiro Ohkura
Graduate School of Engineering, Hiroshima University
Higashi-hiroshima, Hiroshima, Japan

Yoshiyuki Matsumura
Graduate School of Science and Technology,
Shinshu University
Tokida, Ueda, Nagano, Japan

Masanori Goka
Graduate School of Engineering,
Fukuyama University
Fukuyama City, Japan

***Abstract*—Swarm robotics (SR) is a novel approach to the coordination of large numbers of homogeneous robots; SR takes inspiration from social insects. Each individual robot in an SR system (SRS) is relatively simple and physically embodied. Researchers aim to design robust, scalable, and flexible collective behaviors through local interactions between robots and their environment. In this study, a simulated robot controller evolved by a recurrent artificial neural network with the covariance matrix adaptation evolution strategy, i.e., CMA-NeuroES, is adopted for incremental artificial evolution. Cooperative food foraging is conducted by our proposed controller as one of the most complex simulation applications. Since a high level of robustness is expected in an SRS, several tests are conducted to verify that incremental artificial evolution with CMA-NeuroES generates the most robust robot controller among the ones tested in simulation experiments.**

Keywords— Swarm Robotics System, Covariance Matrix Adaptation, Robust Robotic System, Evolutionary Robotics, Evolutionary Algorithms, Genetic Algorithm

I. INTRODUCTION

Swarm robotics (SR) [1] [2] [3] is a novel approach inspired by the observation of social insects, such as ants and wasps. These examples of social in- sects show that simple individuals can successfully accomplish difficult tasks when they coordinate as a group. This kind of system-level behavior, which appears to be robust, scalable and flexible, is impressive to researchers working on robotics. Similarly to these social insects, SR systems (SRSs) are expected to accomplish tasks beyond the capabilities of a single robot. By definition, an SRS comprises a number of relatively simple and typically homogeneous robots that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment, similar to the social insects.

The expression *swarm intelligence* was first conceived by Beni to denote a class of cellular robotic systems in 1980s. These works used many simple agents occupy one-or two-dimensional environment to generate patterns and self-organize there nearest- neighbor interactions. At that time, the definition swarm intelligence only marginally covers works on cellular robotic systems, which does not take the inspiration from social insect behavior. Recently, the expression ”swarm intelligence” moved on to cover a wide range of researches from optimization to social insect studies, losing its robotics context in the meantime. Nowadays, the term SR has started to be used as the application of swarm intelligence to multi-robot systems. Sahin first explicitly started this concern in 2005 [2].

As previously mentioned before, an SRS must have three functional properties at the system level that are observed in natural swarms:

Robustness is the ability to operate despite disturbances resulting from the malfunctioning of its individuals. For instance, lost individuals can be immediately replaced by others, with the operation will continuing smoothly. This is seen as the key advantage of the SRS approach

Flexibility is the ability of an SRS to generate modularized solutions to various tasks, meaning that an SRS must be able to adapt their behaviors to different environments.

Scalability is the ability of an SRS to operate with a wide range of group sizes and support a large number of individuals.

The concept of swarm engineering was introduced by Kazadi [6] in 2000 and the first formal introduction of swarm engineering was released by Winfield et al. in 2004 [7]. Researchers indicated that finding a predictable and controllable design methodology for swarms is the core research direction of swarm engineering [8] [9]. Today, although swarm engineering is still in a very early stage, core topics of swarm intelligence, the design, and analysis have already received attention from SR re- searchers. The notable swarm-bots project [10] was begun in 2001 and terminated in 2005, and was followed by the swarmanoid project in 2006 [11]. New approaches to the design and implementation of self-organizing and the self-assembling problem of autonomous robots were studied in the project [12].

In this study, a cooperative food foraging problem with obstacles in the environment is investigated. We have augmented the covariance matrix adaptation evolution strategy (CMA-ES) with an artificial neural network to create an

efficient approach, CMA-NeuroES, for an SRS to solve simple food foraging problems [30]. However, when an evolutionary approach meets a complex task, it is typical that a simple ER strategy will face a situation that all individuals of the first generation are scored with the same null value; moreover, the selection process cannot operate as expected. This *bootstrap problem* occurs very often with difficult tasks. To avoid this kind of failure, an incremental evolution approach with staged evolution and environmental complexification for a cooperative food foraging task is adopted.

In addition, an SRS can work dynamically as the individual robots are deployed respectively. This kind of decentralized control ensures that SRS has no common failure point. The failure of individual robots will hardly affect SRS performance. The resulting high-level robustness contrasts with the high engineering cost of fault tolerance in conventional robotics systems, and this is free as a basic property of an SRS. Consequently, we compare the best controllers evolved by CMA-NeuroES with two other evolutionary algorithms, fast evolution strategies (FESs) [31] [32] and real-coded genetic algorithms (GAs), [33] through random breakdown tests in computer simulations. As an intrinsic property of SRS, we want to find out whether the loss of some individuals can be compensated for by others or not and also whether the destruction of a particular part of the swarm is will stop its operation or not.

This paper is organized as follows: in Section 2, the related work to SRS and the benchmarks of SRS are presented. Section 3 introduces the cooperative food foraging problem we use in our experiments. Section 4 explains neuroevolution based on a CMA-ES, and a pre-experiment on a CMA-NeuroES controller for cooperative food foraging problem is described. In Section 5, we explain why and how we apply incremental evolution to a cooperative food foraging problem. Section 6 discusses the computer simulation setup and the results of our proposed method. Section 7 draws conclusions and describes our future studies.

II. RELATED WORKS

SRS's design methods can be divided into two categories [13]. (i) Behavior-based design, where in the individual behaviors of robots are designed by hand, is the most commonly used design method. Researchers in this field have proposed various behavior-based design approaches for controlling an SRS. A typical example can be found in Kube and Zhang [14]. They used a design method, i.e., task modeling, where in a robot controller with a finite state machine is carefully designed by a human programmer without a global controller. In this case, individual behaviors were iteratively adjusted until the desired collective behavior was obtained. They demonstrated its effectiveness in box-pushing problems, wherein the collective behavior was obtained after individual behaviors were iteratively adjusted and tuned [15]. However, this approach might have a limitation in problem complexity since no other applications have been published since then. (ii) Automatic design methods are famous for applying evolutionary robotics (ER) that add artificial evolution to robotic systems with a sensory-motor interface to the environment, i.e., evolving a robot controller represented as an artificial neural network [16] [17] [18].

Although the evolutionary robotics approach had been successfully applied to the single robot domain, it recently has been used for evolving group collective behaviors. And the performance of an evolutionary computation approach is strongly dependent on the performance of the artificial neural network optimization [19]. Reynolds [69] was among the first to apply evolutionary robotics techniques to collective behavior making in 1993. He improved on his early work on the simulation of the flocking behavior of birds, i.e., *the boids*. Visual apparatus and the control system was evolved to avoid collisions and to escape from the predator. Based on the experiment of *the boids*, Ward et al. evolved *e-boids* that groups of artificial fish capable of displaying schooling behavior in 2001. In these studies above, the author reported that the creatures were not explicitly rewarded for coordinated motion. On the other hand, Quinn explored two ways of evolving controllers for a coordinated motion behavior by using two simulated Keeper robots. The first approach called *clonal* emphasize the member of the group are homogeneous and share the same genotype. The second approach called *aclonal* requires each member of the group with different genotypes, which means a heterogeneous group. The results indicate that alcohol evolution got better performance than *clonal* evolution. However, the authors report the heterogeneous approach may not be suit- able when the group size becomes larger and the role allocation in the group may be not clear. Perez Uribe et al. successfully evolved small groups of artificial ants for a simple foraging task by simulation to prove homogeneous groups achieve a better performance in 2003.

The collective behaviors of an SRS can be divided into four main categories: spatially organizing, navigation, collective decision-making, and other collective behaviors. SRS researchers could use these basic collective behaviors to work on complex problems, for example, cooperative food foraging problems [25]. Spatially-organizing behaviors focus on how to organize and distribute robots and objects in space, which an SRS could organize and distribute in several ways: for instance, aggregation is the simplest spatial organization of robots in an SRS that are spatially close to each other in an environment. Navigation behaviors focus on how to organize and coordinate the movements of an SRS. These behaviors include collective exploration, coordinated motion, and collective transport, which allow an SRS to explore an environment, coordinate similarly to a flock of birds, or cooperate to transport an object that is too heavy for a single robot. Collective decision-making behaviors focus on how a group of robots influence each other when making choices. For example, to maximize an SRS's performance, task allocation can be specialized by the robots themselves over different tasks. The behaviors that cannot be categorized are called other collective behaviors [26] [27].

These collective behaviors are basic behaviors that can be combined to take over complex real-world applications [28]. In cooperative food foraging problems, an SRS requires the most basic behaviors among benchmark problem in SRS. Self-organization of swarm behavior is needed to cooperate and to move heavy food sources, cooperatively. It also requires navigation behavior to search for food sources and to find a way back to the nest. Last but not least, decision-making behaviors allow robots in a cooperative food foraging problem to change their roles in seconds [29].

III. METHODOLOGY

A. Problem Formulation

The cooperative food foraging problem was inspired by the behavior of ants searching for food sources and bringing the food to the nest. The task is to find better search strategies that maximize the ratio of bringing food to the nest in a specified environment [34] [35]. Fig. 1 shows the food foraging problem we investigate in this paper. The field is a 5,000×5,000 length square unit. The nest, a 1,000×1,000 square unit goal area, is located at the center of the field. One hundred autonomous mobile robots are randomly placed in the nest as the initial condition. Three food sources, F, are randomly placed in the field. Every robot is set to be able to move a food source up to a five-unit weight. However, all of the food sources are 24- unit weight, which means that one food source requires at least five robots to move it cooperatively in a specific direction. A new food source appears soon after one food source is collected during 5,000 time steps. Three obstacles are fixed in the field at a given point that we set in the field. The large static friction we set for each obstacle are impossible for robots to move it, which means the SRS must avoid these obstacles and maximize the ratio of bringing food sources to the nest. The goal of cooperative food foraging task is that SRS should collect as many food sources as possible.

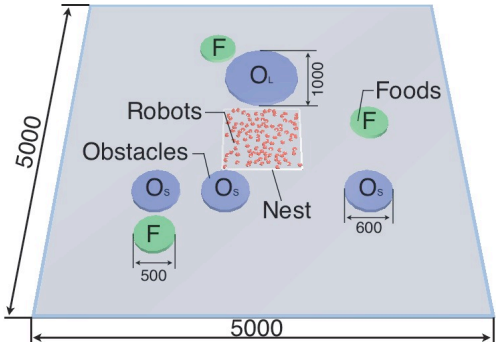


Fig. 1. The cooperative food foraging problem

B. Robot Setup

The SRS in this paper is assumed to be homogenous, i.e., all the robots in the system are assumed to have the same specifications, as shown in Fig. 2. Each robot is 50 length units in diameter and has two types of sensors: eight infrared (IR) sensors and an omni-Vision camera. The eight IR sensors are arranged around a robot. 4 IR sensors equally distributed in the front of the robot, and 2 IR sensors are equally distributed in the back of robot. The other 2 IR sensors are set at two sides of the robot, separately. Each IR sensor provides a value that is inversely proportional to the distance to an object, which might be a food source, an obstacle, a wall, or other robots within the sensor range of 64 length units. The values are normalized between zero and one. The omni-Vision camera is located at the center of each robot.

The robot's sensor abilities are summarized as follows:

- Distance from an IR sensor to objects: $O_i (i = 0, 1, \cdots, 7)$.

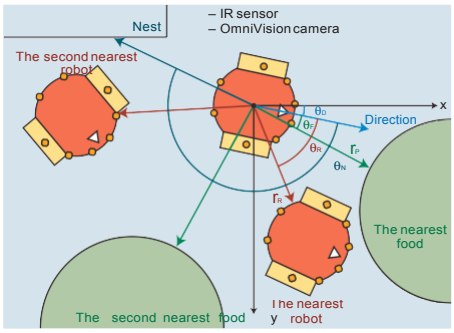


Fig. 2. Robot information

- Distance and direction to the nearest robot: r_{R1}, $\sin\theta_{R1}$ and $\cos\theta_{R1}$.
- Distance and direction to the second nearest robot: r_{R2}, $\sin\theta_{R2}$ and $\cos\theta_{R2}$.
- Distance and direction to the nearest food source: r_{F1}, $\sin\theta_{F1}$ and $\cos\theta_{F1}$.
- Distance and direction to the second nearest food source: r_{F2}, $\sin\theta_{F2}$ and $\cos\theta_{F2}$.
- Direction to the nest: $\sin\theta_N$ and $\cos\theta_N$.
- Global direction of the robot: $\sin\theta_D$ and $\cos\theta_D$.

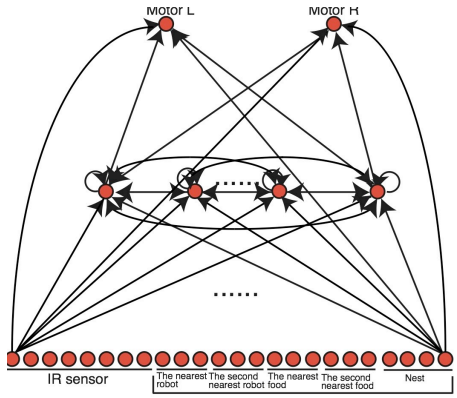


Fig. 3. Artificial neural networks for robot controller

Information obtained by the two types of sensor forms an input layer of a robot controller comprising 24 inputs connected to a motor on the right and another motor on the left that controls two differential driven wheels, enabling robot to move forward or to turn left or right using the rotational difference between wheels. In addition, each input neuron receives Gaussian noise, whose mean and standard deviation (SD) are 0 and 0.03, respectively. Four fully inter-connected hidden layers are adopted from our preliminary experiments for computer simulation. Because the two motor wheels are controlled by EANN output, the output layer consists of two neurons. The neurons of recurrent artificial neural networks (RANN) are connected as shown in Fig. 3, as in our previous study [35]. Therefore, the number of synaptic connections is 162. All robots are assumed to have the same RANN controller.

C. CMA-NeuroES

CMA-ES is a stochastic, iterative method for difficult nonlinear and no convex optimization problems. It has been proven to be a powerful evolutionary optimization algorithm for a variety of test functions, and benchmark problems, and it performs especially well in searching landscapes with discontinuities, noise, and local optima [36] [37].

CMA-ES was introduced by Hansen Ostermeier in 1996 [38], and its use of covariance matrix adaptation made this evolution strategy a highly elaborate optimization algorithm. After weighted recombination was introduced to CMA-ES in 2001 [39], the so-called rank-μ-update greatly reduced time complexity [40] in 2003. The performance of CMA- ES was improved after researchers found that increasing the population size can enhance global search characteristics [41]. In 2008, Raymond and Hansen presented a new approach that reduces evaluation time and space complexity for CMA-ES [42].

Algorithm 1 CMA-ES

procedure CMA–ES
  Initialize:
  $\mathbf{x}_k^{(g+1)} \leftarrow \mathbf{0}$, $\sigma \leftarrow \sigma_{init}$, $\mathbf{C} \leftarrow \mathbf{I}$, g=0
  while Stop condition is not satisfied **do**
    for k=1 to λ **do**
      $x_k^{(g+1)} = m^{(g)} + N_k^{(g)}(0, \sigma^{(g)^2} C^{(g)})$
    end for
    select μ solution points from offspring λ
    adapt mean value $m^{(g)}$ accordingly
    adapt step size σ accordingly
    adapt covariance matrix $\mathbf{C}$ accordingly
  end while
end procedure

In CMA-ES, the offspring for the next generation (g + 1) are generated by sampling a multivariate normal distribution with mean $m \in \mathsf{R}^n$ and covariance $C \in \mathsf{R}^{n \times n}$ [43]. Each solution point $x^{(g+1)}$ at generation $(g + 1)$ in this algorithm presents an n-dimensional real-valued decision variable vector. These variables are altered by recombination and mutation, which correspond to the calculation of the mean value of μ solution points selected from offspring λ. In this algorithm, mutation is used to add a normally distributed random vector with zero mean, and the covariance matrix is updated during evolution to improve searching. Formally, solution points $x^{(g+1)}$ of offspring $k = 1,..., \lambda$ created in generation g are calculated as Algorithm 1.

This is realized by adding a zero-mean random vector drawn from a multivariate normal distribution specified with step size $\sigma^{(g)}$ and covariance matrix $C^{(g)}$. $m^{(g)}$ is the mean value of the population in generation g, and $N^{(g)}(0, \sigma^{(g)} C^{(g)})$ is a multivariate normal distribution with zero mean and covariance matrix C in the g-*th* generation.

CMA-ES efficiency is provided by self-adaptation of C and σ. This allows CMA-ES to search efficiently in a highly correlated search space. For details, see reference [44] [38] [45].

CMA-NeuroES is a weight-evolving artificial neural network that applies CMA-ES to weight optimization. Since the adaptation of C allows efficient searching in the existence of correlation between parameters, we expect that CMA-NeuroES will show good performance on the optimization of the synaptic weights for our robot controller [36] [46] [47]. Every robot in our SRS has the same type of CMA-NeuroES controller. Each robot receives 16-input information from the environment. Swarm behavior fitness is calculated from a fitness table to evaluate a robotic swarm and update the mean value m, covariance matrix C and global step-size σ. CMA-NeuroES operates in five steps:

Step 1: Set all synaptic neural network weights randomly at the initial generation. If it is not the first generation, create offspring from (1).

Step 2: Start the simulation, and then evaluate the fitness of λ offspring by using the fitness table.

Step 3: Send fitness and μ parents to CMA- NeuroES to create new offspring, and update all synaptic weights.

Step 4: Choose synaptic weights with higher fitness as parents for the next generation.

Step 5: Repeat Step 1 to start a new generation until the terminal condition is met.

D. Incremental Evolution

In evolutionary robotics approaches, the situation where no initial search pressures exist can occur when solving highly complex tasks. The result of our experiment on CMA-NeuroES with conventional evolution for the cooperative food foraging problem shows that there are three runs wherein the SRS collected nothing. This situation, the bootstrap problem in ER, occurs when all of the individuals in the initial generation are scored with null fitness prohibiting the progress of evolution. Over-coming the bootstrap problem is one of the difficulties in the ER approach. Incremental evolution is an approach for solving bootstrap problems in highly complex tasks with evolutionary approaches. Mouret and Doncieux [48] categorized incremental evolution into four main approaches: staged evolution, environmental complexification, behavioral decomposition, and fitness shaping. Staged evolution is an approach in which an objective task is divided into ordered sub-tasks, with every sub-task having a corresponding fitness function. A navigation task performed with staged evolution was presented by Bajaj and Ang Jr. [49]. A mobile robot was placed in a simple environment wherein only one obstacle existed. The fitness value was calculated using a straight navigation component and an avoiding obstacles component. At a later stage, the robot was placed in a more complex environment, in which closer walls and sharp turns had been added to the environment. The fitness value was calculated in the same manner as that in the first stage. In the third and final stages, the fitness value was calculated as the product of the value calculated in the previous stage and the wall-following factor. The final result was that the robot acquired the wall-following behavior.

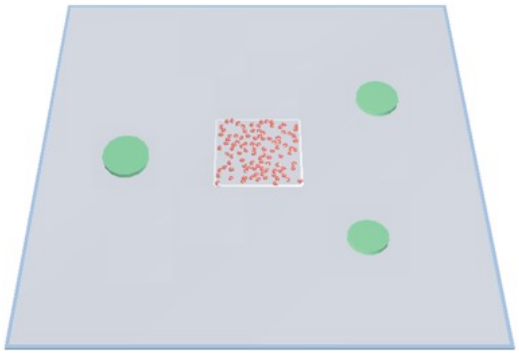
Environmental complexification works on a fitness value calculation in which the task complexity can be continuously modified by operating on certain parameters. A typical example was presented by Gomez and Miikkulainen [50] in 1997. The task was for a predator whose behavior was controlled by an

TABLE I. EVALUATION OF SRS BEHAVIOR

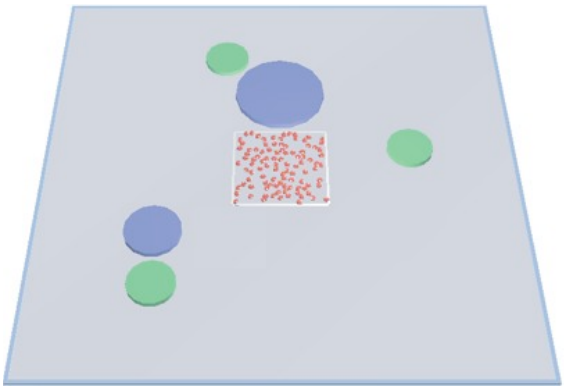
f_1	Touching a food source	+0.0015 × [time steps]
f_2	A food source reaches the nest	+3000
f_3	A food source is moved toward the nest	+1500 × (1 - *drem/dinit*)
f_4	All foods reach the nest	+1.0 × [remaining time steps]

evolving artificial neural network to capture a prey within a fixed number of time steps.

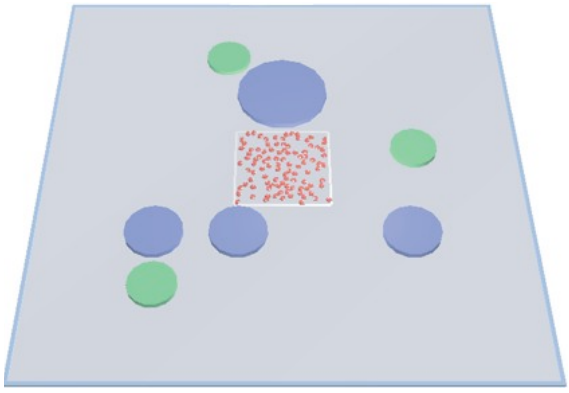
Behavioral decomposition is an approach in which the robot controller is divided into sub- controllers. Every robot controller is evolved separately to solve a sub-task. Nardi et al. [51] evolved a position controller for an autonomous helicopter with three phases of incremental evolution. In the first phase, a simple yaw controller was evolved. In the next phase, the rest of the controller, comprising three modules, specifically, guidance, pitch, and role modules, were evolved independently. In the final phase, these modular controllers were simultaneously evolved to enable them to adapt to each other.



(a) Sub-Task 1



(b) Sub-Task 2



(c) Goal Task 3

Fig. 4. Three-stage incremental evolution for CFFT

Fitness shaping uses a weighted sum of multiple evaluation criteria to create a fitness gradient for artificial evolution to follow. Nolfi and Parisi [52] evolved an autonomous robot that picks up objects. They used a fitness formula with five components, which correspond to the following scenarios: the robot is approaching the target object, the target object is in front of the robot, the robot tries to pick up the object, the robot has the object in its grasp, and the robot releases the object outside the area.

To improve the performance of the SRS in solving a complex cooperative food foraging problem, we proposed staged evolution with environmental complexification using the CMA-NeuroES approach. In our cooperative food foraging task, we assume that three basic behaviors, (1) food-exploration, (2) food-transportation, and (3) obstacle avoidance, are required to solve our problem. Therefore, three-stage incremental evolution was provided, as shown in Fig. 4.

Sub-Task 1 is a very simple problem, in which all three food sources are placed in the field without any obstacles in the environment. Every food source in Sub-task 1 requires at least three robots to move it (The dynamical friction for every food sources is 14 power units). The expectation is that SRS will acquire the basic behavior of food-exploration and food-transportation to the nest, i.e., collect three food sources. When the SRS solved Sub-Task 1, Sub-Task 2, in which two obstacles are added to the field and the positions of food are changed is given to the SRS. The third basic behavior of obstacle avoidance will be acquired after Sub-Task 2. At that time, every food sources needs at least four robots to move it (We increased the dynamical friction to 19 power units). Our simulation will then randomly add a source after the first food source is collected. When the SRS has solved Sub-Task 2, a final task, Goal Task, in which two new obstacles are placed into the field with a narrow path between them, is posed. In that case, food sources are too large to be moved through the narrow path. This trap makes our cooperative food foraging task much more difficult. The SRS learns more advanced food-transportation through obstacle avoidance behavior. In Goal Task, new food sources are randomly created after each food source has been collected. Every food source in Goal Task requires at least five robots to move it. In our cooperative food foraging task, task-transitions to the next sub-task occur only when the SRS has solved the current sub-task continuously for ten generations. The number of generations for task-transition has been optimized in our preliminary experiment.

The performance of SRS was also evaluated using the four components shown in Table 1. In the case of Sub-Task 1, the fitness value f of the SRS is calculated as $f_1 + f_2 + f_3 + f_4$.

In the case of Sub-Task 2, the fitness value f is calculated as $f_2+f_3+f_4$. The f_1 is omitted, because the SRS has already learned the aggregation behavior for food sources through Sub-Task 1. In the case of Goal Task, the simulation will run 5,000 time steps to see how many food sources the SRS can collected. The fitness value of Goal Task f is calculated as f_2+f_3, because the SRS has already learned to touch food sources, and food sources will be added to the field continually during the 5,000 time step simulation.

IV. EXPERIMENTS

A. CMA-NeuroES controller for cooperative food foraging problem

The performance of SRS is depicted by four components shown in Table 1. The SRS collects 0.0015 at each robot and each time step when a robot touches one of the food sources. The sum of the points is set at the f_1 component. The SRS collects a bonus point each time the swarm successfully returns to the nest with a food source. The sum of the points is set at the f_2 component. However, since there can be cases wherein they cannot finish bringing the food source to the nest within the time limit, partial evaluation for moving a food source is considered. For each food source, the points awarded are calculated as $1{,}500 \times (1 - d_{rem}/d_{init})$ at the end of the run, where d_{rem} and d_{init}, the remaining distance to the nest and the initial distance from the nest, respectively, are produced as points. The sum of these points is set as the f_3 component. When all the food sources have been moved to the nest, the f_4 component is calculated as $1.0 \times$ [remaining time steps] when the task is achieved. Otherwise, f_4 is evaluated as zero. The CMA-NeuroES parameter setting is as follows. The offspring λ are set at 100, and the initial SD is set at 0.2, with the initial co-variance matrix $C = I$. The computer simulations' last generation is set at 500, and 10 independent experimental runs are conducted.

B. Applying incremental evolution to CMA-NeuroES for the cooperative food foraging problem

In our compearation computer simulation, (μ, λ)-FastES (FES) [53] [54] and a real-coded GA [55]

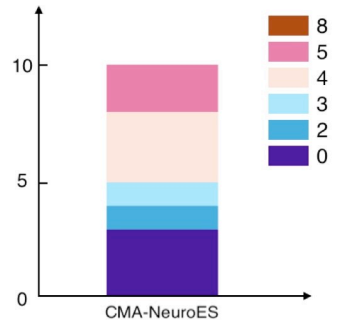


Fig. 5. Food sources that SRS collected

[56] [57] [58] were also used to evolve the synaptic connection weights of the artificial neural network that generated the robots' actions. To make the experiment comparable, four approaches are proposed to solve the cooperative food foraging problem: CMA-NeuroES with conventional evolution, CMA-NeuroES, FES, and real-coded GA with incremental evolution. The parameter settings of the other evolutionary algorithms are as follows. The real-coded GA's population size is also set at 100. Tournament selection with sizes two and elite preservation with size one are adopted. The mutation rate is set at 1.0. This means that all the synaptic connections are mutated for each generation by adding Gaussian noise, who's mean and SDs are 0 and 0.05, respectively. No crossover was used. These parameter tunings had been performed in our preliminary experiments. All the last generations of the artificial evolution are set at 500, and ten independent experimental runs were conducted.

TABLE II. AVERAGE NUMBER OF GENERATIONS IN WHICH THE SWARM SUCCEEDED IN SOLVING SUB-TASKS FOR INCREMENTAL EVOLUTION

		Average	SD
Sub-Task 1	FES	31.8	31.41
	Real Coded GA	41.7	15.2
	CMA-NeuroES	23.9	11.5
Sub-Task 2	FES	149.4	66.3
	Real Coded GA	136.6	46.8
	CMA-NeuroES	115.4	38.8

C. Robustness test

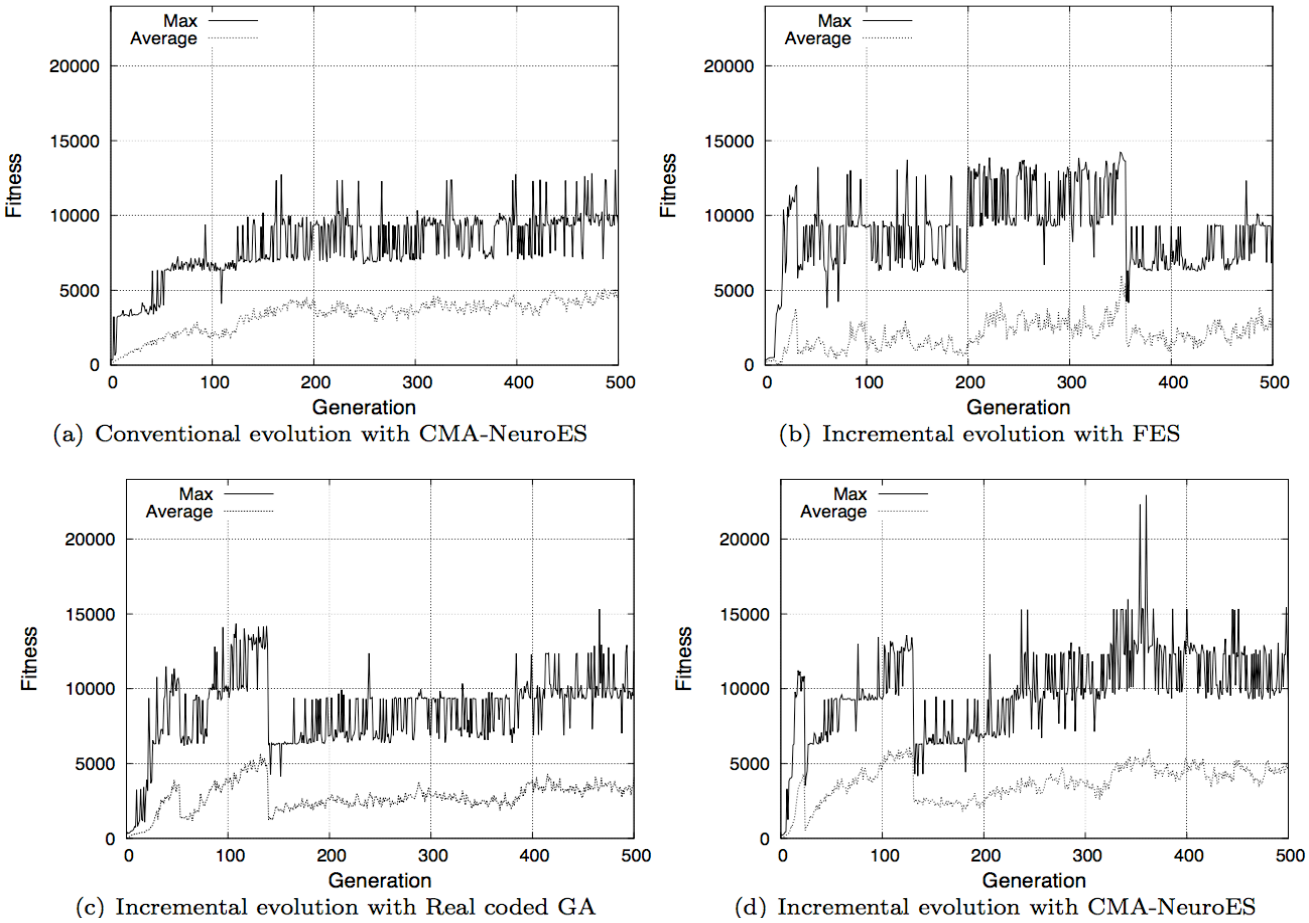
The robustness of the best robot controllers with each approach was measured by conducting a breakdown test with the Goal Task (Fig. 4 (c)). In this test, the robots with the best controllers of each approach were selected. The fact that an SRS can work dynamically as individual robots are deployed has the advantage that the failure of individual robots will hardly affect the performance of an evolved SRS. In our test, the SRS continued its collective behavior for searching food sources and returning food sources to the nest, even after a few robots had stopped working.

Every robot is tested to determine whether it is broken at every time step. The breakdown coefficient (B_c) is calculated as follows:

$$B_c = \frac{S_r}{R_s N} \tag{1}$$

In this equation, Sr is the stop rate, the Rs are the random steps from 0 to 5,000, and N is the number of robots. The test system will decide if any robot is broken by comparing Bc with a uniformly distributed double value between 0.0 and 1.0 from a random number generator's through at every time step. If B_c is larger than the random number, then the robot will stop. All the broken robots remain in the field and can be detected by robot sensors. Stop counter S_c will count one after a robot is stopped to control the number of broken robots.

Our breakdown simulation runs 5,000 time steps for ten iterations. Moreover, we consider the limitation of the stop counter by 10, 20, 30, 50, meaning that 10, 20, 30, 50 robots in the SRS, respectively, will be stopped randomly during the simulation.



(a) Conventional evolution with CMA-NeuroES

(b) Incremental evolution with FES

(c) Incremental evolution with Real coded GA

(d) Incremental evolution with CMA-NeuroES

Fig. 6. Fitness transitions of the best(solid line) and the average (dotted line) for each controller's best performed run

V. RESULTS

Fig. 5 shows the result of the CMA-NeuroES controller for the cooperative food foraging problem. Our SRS successfully collected food sources in seven of ten runs and the maximum number of collected food sources was five.

However, three runs performed very poorly; in them the SRS collected nothing at all.

Fig. 6 shows the fitness transitions of the best individuals and the averages of each individual in the best run. Fig. 7 shows that the incremental evolution approaches with evolutionary algorithms collected at least two food sources, indicating that they were successful in solving the Sub- Tasks for all the runs. The best run of CMA-NeuroES collected eight food sources in Goal Task, whereas conventional evolution had three runs that collected nothing. It is clear that not only the maximum fitness but also the average fitness of CMA-NeuroES with incremental learning is higher than those of others. Table. 2 shows the average number of generations and corresponding SDs required by the swarm to succeed in solving the sub-tasks. The incremental evolution approach with CMA-NeuroES required approximately 23 generations to solve Sub-Task 1 and approximately 115 additional generations to solve Sub-Task 2. Conversely, the FES and real-coded GA required approximately 31 generations and 41

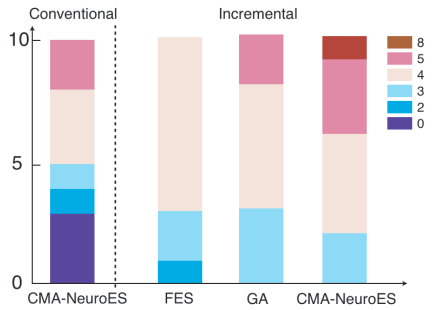


Fig. 7. Food sources that SRS collected

generations to solve Sub-Task 1, respectively, and approximately 149 generations and 136 generations to solve Sub-Task 2,respectively. Therefore, incremental evolution with CMA-NeuroES exhibits better search ability to find better solutions. Table 3 shows the results of the average returned food source numbers, and the SD of four approaches for ten iterations. As a result, we see that conventional evolution performs poorly. When 50 robots stopped during our simulation, only one food source could be returned to the nest. In the incremental approach, the number of returned food sources of FES decreased to two, when the stopped robots number increased from 10 to 50. Real-coded GA shows its robustness, because the returned food source numbers decreased from four to three and the SD

shows its stability. However, CMA-NeuroES with incremental evolution performed best overall. When ten robots in the SRS stopped, the SRS could still return at least five food sources to the nest. The robust of CMA-NeuroES enables it to return four food sources even when half of the robots in the SRS are stopped. Typical behavior observed in robustness tests for an SRS with a CMA-NeuroES controller is shown in Fig. 8, wherein 50 robots are stopped during the simulation. Some robots immediately find a food source (Marked 2) after leaving the nest (Fig. 8(a)-8(b)). At the same time, some robots are stopped at the beginning of our robustness tests and become obstacles in the field. In Fig. 8(c), another food

TABLE III. NUMBER OF BROUGHT BACK FOOD SOURCES WHILE THE ROBOTS BREAKDOWN

	Conventional Evolution				FES				Real Coded GA				CMA-NeuroES			
Breakdown	10	20	30	50	10	20	30	50	10	20	30	50	10	20	30	50
Average	2.3	1.0	1.3	1	3.5	2.8	3.3	2.8	4.4	4.2	3.9	3.7	5.1	4.5	4.9	4.3
SD	1.1	0.9	0.6	0.7	1.3	1.0	1.4	0.9	1.8	1.2	0.9	0.8	1.0	0.5	0.5	0.8

source (Marked 1) is found by another group of robots, when the first group of robots is trying to return the food sources to the nest. The food source (Marked 1) is collected after our SRS successfully bypassed the narrow pass in the field (Fig. 8(c)-8(e)) while two food sources (Marked 2, 3) are already near the nest. An additional food source (Marked 4) is added to the field as soon as the first food source is collected (Fig. 8(f)). Nearly half of the robots are stopped at this moment, after the food sources (Marked 4 and 5) are collected ((Fig. 8(h)-8(i))). At the end of the simulation, 50 robots in our SRS have been stopped. Two groups of robots are still trying to collect the food sources (Marked 7 and 8) as the simulation ends (Fig. 8(j)).

VI. CONCLUSIONS

In this paper, we point out that the definition of swarm intelligence was extended in 2004 and there is almost no relationship between the cellular robotic systems with SRS. After that we success- fully applied the ER approach to a specific complex cooperative food foraging problem with a large SRS including one hundred homogenous robots. Swarm behavior emerged as a result of CMA-NeuroES with incremental evolution. The incremental evolution approach of staged evolution and environmental complexification helps ER avoid the bootstrap problem. The result of incremental evolution out- performs the conventional evolution approach for cooperative food foraging. In addition, a robustness test confirmed that the incremental evolution approach with CMA-NeuroES is robust, because it can solve the same cooperative food foraging problem, even when half of the robots are stopped. We expect that our proposed method and robustness test will also hold for other SRS benchmarks.

In the future, we will focus primarily on the analysis of SRSs [59] [60] [61] [62]. As an SRS can be considered as a large network with interactions among robots, we investigated finding subgroups in a robotic swarm [63] using the technology of complex networks. The next step will be to describe how subgroups in an SRS develop and change in a large robotic swarm using a duration table. That will enable researchers to understand the detail of task allocation in an SRS from a macroscopic viewpoint.

REFERENCES

[1] Murphy R. Introduction to AI robotics. MIT press; 2000.

[2] Şahin E. Swarm robotics: From sources of inspiration to domains of application. In: Swarm robotics. Springer; 2005:10–20.

[3] Dorigo M, Roosevelt AF. Swarm robotics. In: Special Issue, Autonomous Robots. Citeseer; 2004.

[4] Winfield A. Special issue on swarm robotics. Swarm Intelligence 2008;2(2):69–72.

[5] Sahin E, Girgin S, Bayindir L, Turgut AE. Swarm robotics. Swarm Intelligence 2008;1:87–100.

[6] Kazadi ST. Swarm engineering. Ph.D. thesis; California Institute of Technology; 2000.

[7] Winfield AF, Harper CJ, Nembrini J. Towards dependable swarms and a new discipline of swarm engineering. In: Swarm robotics. Springer; 2005:126–42.

[8] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: from natural to artificial systems. 1; Oxford university press; 1999.

[9] Beni G. From swarm intelligence to swarm robotics. In: Swarm Robotics. Springer; 2005:1–9.

[10] Dorigo M, Tuci E, Groß R, Trianni V, Labella TH, Nouyan S, Ampatzis C, Deneubourg JL, Baldassarre G, Nolfi S, et al. The swarm-bots project. In: Swarm Robotics. Springer; 2005:31–44.

[11] Dorigo M, Tuci E, Groß R, Trianni V, Labella TH, Nouyan S, Ampatzis C, Deneubourg JL, Baldassarre G, Nolfi S, et al. The swarm-bots project. In: Swarm Robotics. Springer; 2005:31–44.

[12] SoysalO,ŞahinE. A macroscopic model for self-organized aggregation in swarm robotic systems. In: Swarm robotics. Springer; 2007:27–42.

[13] Crespi V, Galstyan A, Lerman K. Top-down vs bottom-up methodologies in multi-agent system design. Autonomous Robots 2008;24(3):303–13.

[14] Kube CR, Zhang H. Task modelling in collective robotics. In: Robot colonies. Springer; 1997:53–72.

[15] Yim M, Zhang Y, Lamping J, Mao E. Distributed control for 3d metamorphosis. Autonomous Robots 2001;10(1):41–56.

[16] Floreano D, Urzelai J. Evolutionary robotics: The next generation. Tech. Rep.; AAI Books; 2000.

[17] Kaelbling LP, Littman ML, Moore AW. Reinforcement learning: A survey. Journal of artificial intelligence research 1996;:237–85.

[18] Fogel DB, Fogel LJ. An introduction to evolutionary programming. In: Artificial Evolution. Springer; 1996:21–33.

[19] Ferber J. Multi-agent systems: an introduction to distributed artificial intelligence; vol. 1. Addison-Wesley Reading; 1999.

[20] Harvey I, Husbands P, Cliff D, et al. Issues in evolutionary robotics. School of Cognitive and Computing Sciences, University of Sussex; 1992.

[21] Meyer JA, Husbands P, Harvey I. Evolutionary robotics: A survey of applications and problems. In: Evolutionary Robotics. Springer; 1998:1–21.

[22] Storn R, Price K. Differential evolution–a simple and ef- ficient heuristic for global optimization over continuous spaces. Journal of global optimization 1997;11(4):341– 359.

[23] Ohkura K, Yasuda T, Kawamatsu Y, Matsumura Y, Ueda K. Mbeann: mutation-based evolving artifi- cial neural networks. In: Advances in Artificial Life. Springer; 2007:936–45.

[24] Ohkura K, Yasuda T, Kotani Y, Matsumura Y. A swarm robotics approach to cooperative package- pushing problems with evolving

recurrent neural net- works. In: SICE Annual Conference 2010, Proceedings of. IEEE; 2010:706–11.

[25] Camazine S, Deneubourg JL, Franks NR, Sneyd J, Bonabeau E, Theraulaz G. Self-organization in biolog- ical systems (2001).

[26] Cliff D, Husbands P, Harvey I. Explorations in evolu- tionary robotics. Adaptive behavior 1993;2(1):73–110.

[27] Ohkura K, Yasuda T, Matsumura Y. Coordinating the adaptive behavior for swarm robotic systems by using topology and weight evolving artificial neural net- works. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE; 2010:1–8.

[28] Brambilla M, Ferrante E, Birattari M, Dorigo M. Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 2013;7(1):1–41.

[29] Ohkura K, Yasuda T, Sakamoto T, Matsumura Y. Evolving robot controllers for a homogeneous robotic swarm. In: System Integration (SII), 2011 IEEE/SICE International Symposium on. IEEE; 2011:708–13.

[30] Martinoli A, Easton K, Agassounon W. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. The International Journal of Robotics Research 2004;23(4-5):415–36.

[31] BDack T, Hoffmeister F, Schwefel H. A survey of evolution strategies. In: Proceedings of the 4th international conference on genetic algorithms. 1991:2–9.

[32] Ba ̈ck T, Schwefel HP. An overview of evolutionary algorithms for parameter optimization. Evolutionary com- putation 1993;1(1):1–23.

[33] Goldberg DE. Genetic algorithm in search. Optimization and Machine Learning 1989;.

[34] Sugawara K, Sano M. Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system. Physica D: Nonlinear Phenomena 1997;100(3):343–54.

[35] Yu T, Yasuda T, Ohkura K, Matsumura Y, Goka M. Cooperative transport by a swarm robotic system based on cma-neuroes approach. JACIII 2013;17(6):932–42.

[36] Hansen N. The cma evolution strategy: A tutorial. Vu le 2005;29.

[37] Igel C. Neuroevolution for reinforcement learning using evolution strategies. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on; vol. 4. IEEE; 2003:2588–95.

[38] Hansen N, Ostermeier A. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Evolutionary Computation, 1996., Proceedings of IEEE International Conference on. IEEE; 1996:312–7.

[39] Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 2001;9(2):159–95.

[40] Hansen N, Mu ̈ller S, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary Computation 2003;11(1):1–18.

[41] Hansen N, Kern S. Evaluating the cma evolution strategy on multimodal test functions. In: Parallel problem solving from nature-PPSN VIII. Springer; 2004:282–91.

[42] Ros R, Hansen N. A simple modification in cma-es achieving linear time and space complexity. In: Parallel Problem Solving from Nature–PPSN X. Springer; 2008:296–305.

[43] Hansen N, Auger A, Ros R, Finck S, Po ̌s ́ık P. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: Proceedings of the 12th annual conference companion on Genetic and evolutionary computation. ACM; 2010:1689–96.

[44] Hansen N, Ostermeier A, Gawelczyk A. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In: ICGA. Citeseer; 1995:57–64.

[45] Moriguchi H, Honiden S. cma-tweann: efficient optimization of neural networks via self-adaptation and seamless augmentation. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation. ACM; 2012:903–10.

[46] Hansen N, Ostermeier A. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The Eufit 1997;97:650–4.

[47] FloreanoD,DürrP,MattiussiC. Neuroevolution: from architectures to learning. Evolutionary Intelligence 2008;1(1):47–62.

[48] Mouret JB, Doncieux S. Incremental evolution of animats behaviors as a multi-objective optimization. In: From Animals to Animats 10. Springer; 2008:210–9.

[49] Bajaj.D, Ang Jr MH. An incremental approach in evolving robot behavior. In: Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision. 2000:.

[50] Gomez F, Miikkulainen R. Incremental evolution of complex general behavior. Adaptive Behavior 1997;5(3- 4):317–42.

[51] De Nardi R, Togelius J, Holland OE, Lucas SM. Evolution of neural networks for helicopter control: Why modularity matters. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. IEEE; 2006:1799– 806.

[52] Nolfi S, Parisi D. Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects. In: Topics in artificial intelligence. Springer; 1995:243–54.

[53] Rechenberg I. Evolution strategie: Optimerung technischer systeme nach prinzipien des biologischen evolution. Friedrich Fromann? erlag, Stuttgart; 1(9):7.

[54] Yao X, Liu Y. Fast evolution strategies. In: Evolutionary Programming VI. Springer; 1997:149–61.

[55] Eshelman LJ, Schaffer JD. Real–coded genetic algorithms and interval-schemata 1992;.

[56] Holland JH. Genetic algorithms and the optimal allocation of trials. SIAM Journal on Computing 1973;2(2):88–105.

[57] Koza JR. Genetic programming: on the programming of computers by means of natural selection; vol. 1. MIT press; 1992.

[58] Koza JR. Hierarchical genetic algorithms operating on populations of computer programs. In: IJCAI. Citeseer; 1989:768–74.

[59] Everitt B. Cluster analysis. 1993. Edward Arnold and Halsted Press, 1993;.

[60] Girvan M, Newman ME. Community structure in social and biological networks. Proceedings of the National Academy of Sciences 2002;99(12):7821–6.

[61] Newman ME. Detecting community structure in networks. The European Physical Journal B-Condensed Matter and Complex Systems 2004;38(2):321–30.

[62] Newman ME. Fast algorithm for detecting community structure in networks. Physical review E 2004;69(6):066133.

[63] Ohkura K, Yasuda T, Matsumura Y. Analyzing macroscopic behavior in a swarm robotic system based on clustering. In: SICE Annual Conference (SICE), 2011 Proceedings of. IEEE; 2011:356–61.

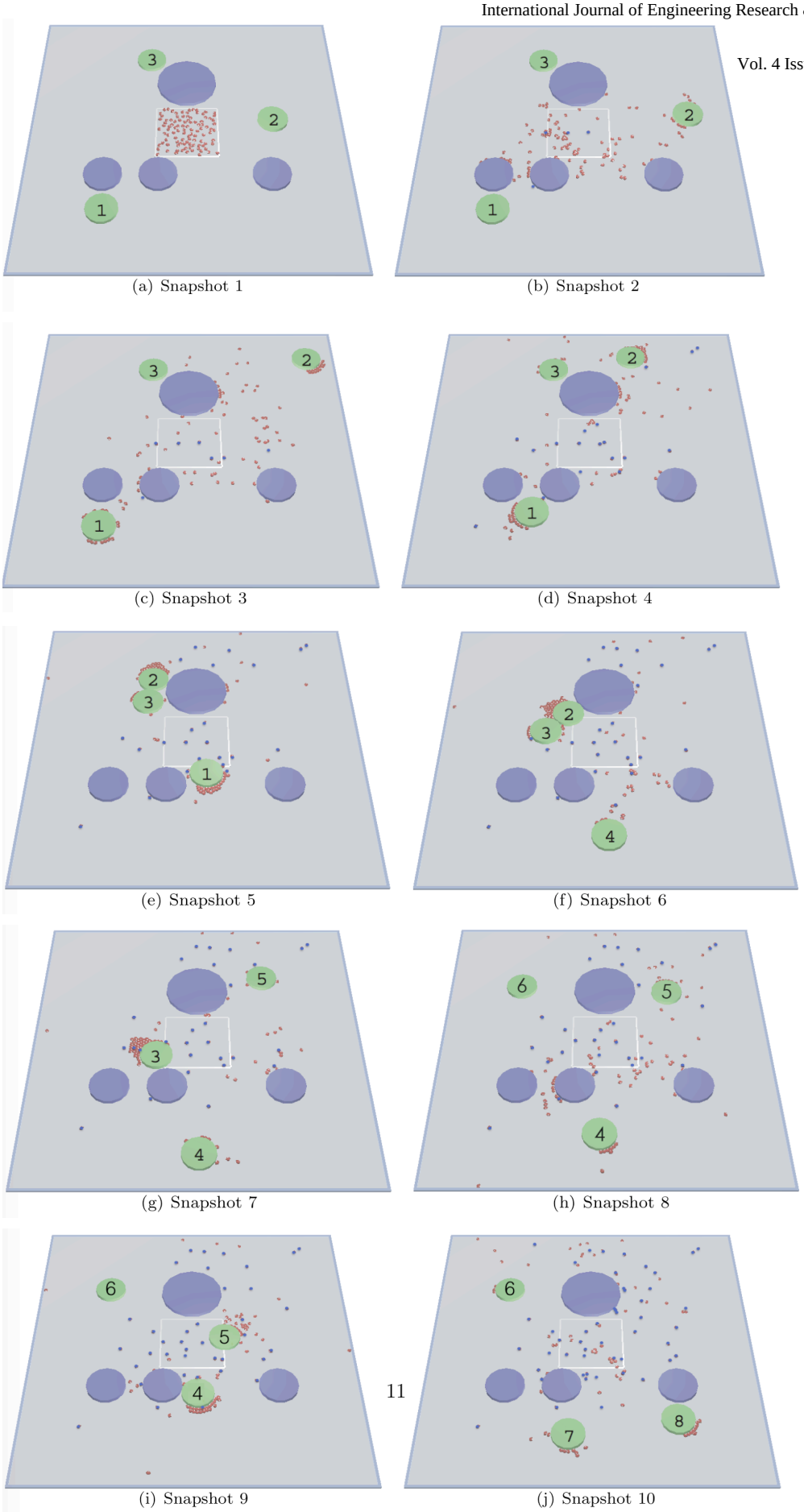


Fig. 8. Snapshots of the best cooperative collective behavior found by CMA-NeuroES with incremental evolution approach of breakdown rate 50%