Artificial Life and Robotics (2019) 24:127–134
https://doi.org/10.1007/s10015-018-0466-6

ORIGINAL ARTICLE

Autonomous task allocation by artificial evolution for robotic swarms in complex tasks

Yufei Wei[1] · Motoaki Hiraga[1] · Kazuhiro Ohkura[1] · Zlatan Car[2]

Received: 13 May 2018 / Accepted: 14 August 2018 / Published online: 17 September 2018

Abstract
Swarm robotics is a field in which multiple robots coordinate their collective behavior autonomously to accomplish a given task without any form of centralized control. In swarm robotics, task allocation refers to the behavior resulting in robots being dynamically distributed over different sub-tasks, which is often required for solving complex tasks. It has been well recognized that evolutionary robotics is a promising approach to the development of collective behaviors for robotic swarms. However, the artificial evolution often suffers from two issues—the bootstrapping problem and deception—especially when the underlying task is profoundly complex. In this study, we propose a two-step scheme consisting of task partitioning and autonomous task allocation to overcome these difficulties. We conduct computer simulation experiments where robotic swarms have to accomplish a complex collective foraging problem, and the results show that the proposed approach leads to perform more effectively than a conventional evolutionary robotics approach.

Keywords Robotic swarm · Evolutionary robotics · Autonomous task allocation · Task partitioning

1 Introduction

Swarm robotics [1] studies how systems composed of large numbers of autonomous robots can be used to accomplish tasks that are beyond the capabilities of a single robot. The robots are relatively simple compared to the task they are dealing with, that their communication is usually local and sensory capabilities are limited. A robotic swarm operates in a distributed and self-organizing manner, that is, there is neither a leader dictating to the other robots, nor are the robots informed of global information. On the contrary, each robot follows simple rules and acts autonomously on the basis of local observation. Therefore, the emergence of collective behaviors can be regarded as a result of the numerous local interactions between the robots and between robots and the environment [2].

Designing control software for a robotic swarm is a challenging task. The difficulty resides in the fact that the relationship between simple local rules and complex swarm behaviors is indirect [3]. One approach to designing a robot controller is tuning a finite state machine by trial and error until expected collective behaviors are acquired [4]. However, the design process is guided only by experience and intuition, which requires expertise in the undertaken task. A promising alternative is evolutionary robotics [5], in which the design problem is transformed into an optimization problem to reduce human intervention [6]. In evolutionary robotics, the control software is conventionally represented by a single artificial neural network [7], in which synaptic weights are optimized through the use of artificial evolution.

Relative to these design methods, several collective behaviors have been developed, such as aggregation [8], chain formation [9], collective transport [10] and task allocation [11]. Among these behaviors, task allocation is the one resulting in robots being distributed into different subtasks while dealing with a complex task. The allocation changes dynamically based on local observations of robots, whose goal is to maximize the performance of the whole swarm [1]. Conventionally, evolutionary robotics often fails to develop

✉ Kazuhiro Ohkura
kohkura@hiroshima-u.ac.jp

Zlatan Car
zlatan.car@uniri.hr

[1] Graduate School of Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashi-Hiroshima, Hiroshima 739-8527, Japan

[2] Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia

useful autonomous task allocation mechanisms. The reason is that the artificial evolution is more likely to get stalled in complex tasks due to two issues: the bootstrapping problem [12] and deception [13]. The bootstrapping problem is often caused by the gap between the design objective and primitive capabilities of the controller, which makes it a challenging task to devise fitness functions applying selective pressure towards better solutions in early generations, and therefore, prevents the evolutionary process from starting. Deception is due to the lack of gradient to global optimum, resulting in the evolution being trapped in local optima and generates uninteresting controllers.

On the other hand, task partitioning is the study of how a given task can be decomposed into simpler subtasks, which can be used to reduce the complexity of tasks as well as the difficulty of designing fitness functions. In this study, we propose a two-step scheme which consists of task partitioning and autonomous task allocation to address these issues. In the first step, the original task is partitioned into simpler subtasks to reduce the complexity of designing fitness functions. In the second step, evolutionary approaches are adopted to synthesize a composite artificial neural network-based controller to generate autonomous task allocation for the robotic swarm.

The remainder of this paper is organized as follows. Section 2 reviews previous studies relative to task allocation in robotic swarms, as well as studies that address the bootstrapping problem and deception. In Sect. 3, we describe the proposed two-step scheme and how it can be combined with the evolutionary robotics approach. Section 4 explains the experimental setting. The performance comparison between the proposed approach and a conventional evolutionary robotics approach is analyzed in Sect. 5. Finally, we conclude the paper and discuss our future work in Sect. 6.

2 Related work

In swarm robotics, most task allocation mechanisms are developed for scenarios such as foraging [4], transportation [11] and object clustering [14], in which robots have to search for objects scattered in the environment and then perform further operations on these objects. Typically, task allocation is obtained by either thresholds or probabilistic methods. In threshold-based methods, robots change their activities when an observed variable exceeds the threshold. The observed variable can be either local, e.g., time spent on current activity [15], or global, e.g., energy level of the nest [16]. The value of thresholds may also change all the time based on the perception of the environment [17]. In probabilistic-based methods, the activities of robots are determined randomly with a probability value relative to internal states of robots or environment observations [18].

On the other hand, evolutionary robotics has the advantage to synthesize robot control software given only a fitness function based on a high-level description of the task [19]. However, researchers have to overcome the bootstrapping problem and deception especially when the undertaken task is difficult. A typical approach to overcoming these issues is to promote diversity of individuals, so that the evolutionary process may avoid being deceived by maintaining multiple paths in the search space. In [20], Lehman and Stanley proposed a behavior diversity approach—novelty search, in which the evolution always searches for novel behaviors without an explicit objective describing the quality of behaviors. Although a number of studies reported that novelty search is less affected by the bootstrapping problem and deception [21, 22], defining the behavior characterization that indicates the novelty of behaviors is still a challenging task.

An alternative solution is to assist the evolutionary process directly with human knowledge based on experience. In this context, several approaches have been developed, including incremental evolution, behavioral decomposition, and human-in-the-loop [19]. The key idea of incremental evolution is to train the controller in a simplified task first, then increase the difficulty gradually until the final objective is accomplished. In [23], incremental evolution is applied to a highly integrated task where the robots have to search for light source while avoiding holes in the environment. The evolution is started in an environment with a simple layout and few holes, then more holes and different layouts are used as the evolution finds high-quality controllers for the current stage. Behavioral decomposition focuses on dividing the overall behavior of the robot into sub-behaviors which are trained sequentially or independently. Togelius [24] addressed a goal seeking task, in which three sub-behaviors: "conditional phototaxis", "obstacle avoidance" and "non-reactive learning" are evolved sequentially. In [25], the authors synthesized a hierarchical controller for a rescue task in a T-maze environment, where the overall behavior of the robot is divided into primitives and arbitrators. The primitives are trained independently, representing basic capabilities of the robot, such as "turn left", "follow wall", "turn right" and "exit the room". Then upper arbitrators combine these primitives and lower arbitrators to accomplish the given task. Differently, human-in-the-loop requires the designer to interact with the evolutionary process, guiding the evolution to avoid local optima. In [26], Celis et al. demonstrated this method in a locomotion task with an obstacle, results show that the robot successfully avoided the obstacle with user demonstration and low-level control.

3 The two-step scheme

In this section, we propose a two-step scheme consisting of task partitioning and autonomous task allocation to overcome the bootstrapping problem and deception, and describe how it can be combined with the evolutionary robotics approach to synthesize composite controller for robotic swarms.

3.1 Task partitioning

Task partitioning is a topic highly related to task allocation, which studies how a given task can be decomposed into multiple subtasks. While task allocation studies the dynamics of robots over different subtasks, task partitioning focuses on the way of how the task itself is organized. Applying task partitioning to a complex task benefits the robotic swarm at both individual-level and collective-level. At individual-level, the partitioned subtasks are easier for the robots to achieve, which also reduce the complexity of designing fitness functions. At collective-level, decomposing the given task into simpler subtasks reduces the difficulty of management, allowing the adoption of task allocation mechanisms to improve the performance of the whole swarm.

It is worthy to note the relation between behavior decomposition and task partitioning. In behavior decomposition, the target of decomposition is the behavior of a single robot, that is, since the task is too difficult to develop a simplex controller exhibiting all required behaviors, one may divide and achieve these individual-level behaviors separately. However, in most cases of swarm robotics, the relationship between individual-level behaviors and the resulting collective-level behavior is not straightforward. By contrast, task partitioning works on the task itself rather than robots, in which the designer performs the decomposition given only task description. Task partitioning and behavior decomposition are not exclusive alternatives, that behavior decomposition can be applied to the robots while executing partitioned tasks.

Conventionally, task partitioning is conducted by a human designer based on his expertise [11, 25]. Although progress on the management of partition [27] and automatic task partitioning [28] has been reported, the results are confined to specific tasks, and a generalized guideline for task partitioning is still missing. In this study, we perform task partitioning manually in a recursive and hierarchical manner, where the given task is divided into subtasks which can be organized as a tree-like graph of which the root node represents the original task, the internal nodes are intermediate subtasks that can be further divided, and the leaf nodes stand for the simplest subtasks that should be solved directly.

3.2 Autonomous task allocation

To accomplish the given task, a robot controller that generates autonomous task allocation behavior over the partitioned subtasks is required. In this work, we propose a composite controller architecture in which sub-controllers are organized similarly to the subtasks, where the sub-controller on the top selects and activates one of its lower sub-controller recursively until it reaches a leaf node sub-controller which takes control of the robot. Therefore, the sub-controllers can be divided into two types: sub-controllers in the root node and internal nodes act as "arbitrator", deciding which subtask to perform; leaf node sub-controllers act as "primitive", controlling the robot to accomplish the corresponding subtask with the others cooperatively.

The development of the controller follows a bottom-up procedure, in which related sub-controllers in lower level must be evolved first before developing an upper sub-controller. This process is repeated until the root node sub-controller is obtained. Importantly, task partitioning reduces the interference between design objectives of the partitioned tasks, allowing the designer to develop sub-controllers with different fitness functions.

4 Experiments

In this section, the proposed approach is examined in a complex variation of a typical collective foraging problem.[1] We also perform comparison experiments with different fitness function settings, in which a conventional evolutionary robotics approach is adopted.

4.1 A complex collective foraging problem

To demonstrate the effectiveness, the proposed approach is applied to a complex collective foraging problem in which robots have to search the field for resources and transport them back to the nest as many as possible. Figure 1 shows the experiment environment, which contains three areas: the nest, the resource field, and the decomposition area. Resources of two types can be found on the field: individual resource and resource package. An individual resource is light enough to be transported back to the nest by a single robot. By contrast, each resource package contains seven individual resources, which is so heavy that the cooperation among robots is required to move it. Furthermore, resource packages cannot be transported to the nest directly due to the barriers at the nest entrance. Instead, these resource

[1] All experiments are conducted with an open-source 2D physics engine—Box2D, http://box2d.org.

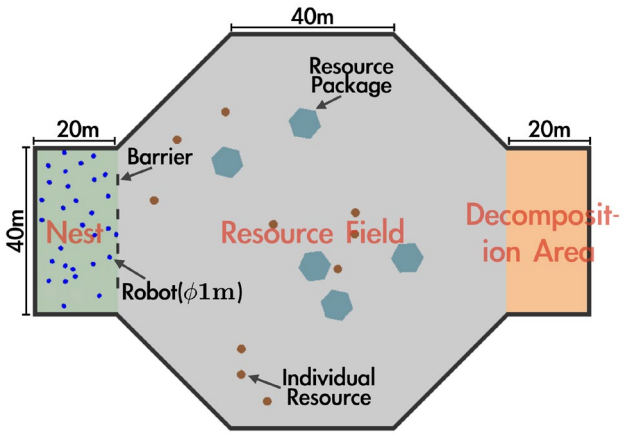

Fig. 1 A complex collective foraging problem

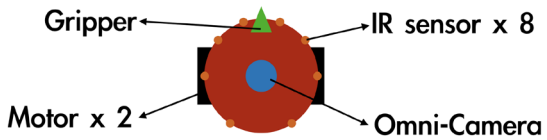

Fig. 2 Specification of the robots

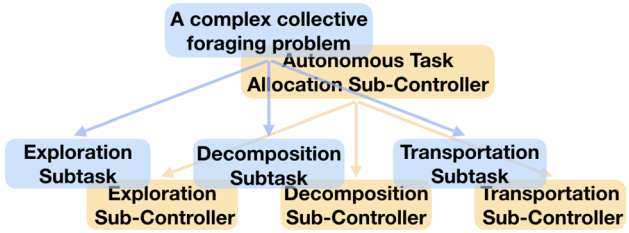

Fig. 3 Partitioned subtasks (blue part) and robot controller (orange part). (Color figure online)

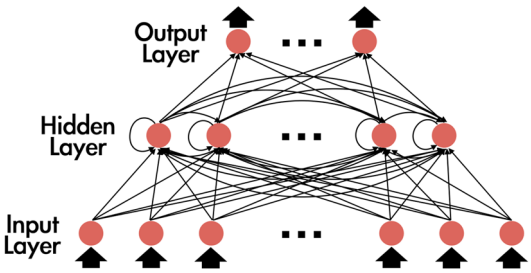

Fig. 4 Artificial neural network architecture, which is adopted for all sub-controllers

packages should be moved to the decomposition area first, in which they will be unpacked into seven individual resources automatically. At the beginning of the task, there are thirty robots placed in the nest with random position and direction. Ten individual resources and five resource packages are randomly located in the resource field.

The specification of robots is as shown in Fig. 2. Each robot is composed of an Omni-camera, eight IR sensors, a gripper, two motors and a composite artificial neural network-based controller. The range of the Omni-camera is set to 8 m, gathering information in its sight, including (1) the number of robots, individual resources and resource packages, and (2) the distance and direction of the nearest robot, the nearest individual resource, and the nearest resource package. The robots are also informed of the direction of the nest and the decomposition area. IR sensors are set to detect the distance between the robot and other objects within 2 m in eight directions. The gripper is a pre-programmed component, which can be turned on to catch an individual resource in front of it, or turned off to drop the catching resource. The max speed of two motors is limited to 5 m/s.

4.2 Experimental setting

To synthesize the composite robot controller, we perform task partitioning in the first step. As discussed in the last section, resources of two types exist and need to be operated separately. Considering the fact that the range of sensory components of the robots is limited compared with the size of the field, robots also have to explore the environment. Therefore, we divide the complex collective foraging problem into three subtasks: exploration, decomposition, and transportation. The partitioned subtasks and the corresponding robot controller architecture are as shown in Fig. 3.

In the second step, we develop sub-controllers for the partitioned subtasks first and then combine these sub-controllers together by evolving the autonomous task allocation sub-controller in the upper layer. All sub-controllers are trained in the same environment as shown in Fig. 1, and are represented by typical three-layered artificial neural networks of which the hidden layer is recurrent (see Fig. 4). The details of the development of sub-controllers are described as below.

Exploration Since the experiment field is large compared with the range of sensory components, the robots have to explore the field for resources collectively. Additionally, to mitigate the congestion in the environment while performing exploration, the robots should also be capable of avoiding other objects. The inputs of the sub-controller include the distance information from eight IR sensors, as well as the distance and direction of the nearest robot. The hidden layer has ten nodes (the same setting is adopted for the decomposition, transportation, and autonomous task allocation sub-controller). The outputs control two motors of the robot.

The sub-controller gets reward proportionated to the coverage rate of the robots in the field, and is punished if the

robots collide with other objects. To calculate the coverage rate, we randomly sample 100 positions in the field at each time-step and count how many positions are in at least one robot's sight. The fitness function is defined as Eq. (1):

$$F_{\text{ex}} = \frac{\sum P_s}{M} - C, \tag{1}$$

where P_s is the number of positions covered by the robots in the sth time-step, M denotes the max time-step of the simulation, and C is the number of collisions during the simulation.

Decomposition Due to the existence of the barriers, the robots have to cooperatively move the resource packages to the decomposition area first to unpack them. Since we want to concentrate on the development of behaviors related to the decomposition subtask, the robots are set to work in the following manner: activates the decomposition sub-controller if there is at least one resource package in the sight of the robot, otherwise performs exploration using the sub-controller obtained in the last section. The inputs consist of the distance information from eight IR sensors, the direction of the decomposition area, and the distance and direction of the nearest resource package and the nearest robot. Similarly, the outputs of the sub-controller control the two motors.

Equation (2) shows the fitness function which rewards the sub-controller for moving resource packages towards the decomposition area. Additionally, there is a bonus encouraging the robots to achieve the subtask as fast as possible.

$$F_{\text{de}} = \sum D^j_{\text{rp}} + R, \tag{2}$$

where D^j_{rp} is the distance shortened between the jth resource package and the decomposition area, and R is the remaining time-steps after all resource packages are unpacked.

Transportation An individual resource is light enough for a single robot to transport to the nest. As with the development of the decomposition sub-controller, the robots perform exploration if no individual resource can be observed, and switch to train the transportation behavior after they find an individual resource. The gripper of the robot is turned on only when performing this subtask. The inputs of the sub-controller include the distance information from eight IR sensors, the direction of the nest, and the distance and direction of the nearest individual resource, as well as the status of the gripper (catching an individual resource or not). The outputs control the two motors.

The evaluation of the sub-controller is based on the distance reduced between individual resources and the nest. The sub-controller will also get a bonus proportionated to the remaining time-steps after the subtask is accomplished. The fitness function is illustrated in Eq. (3).

$$F_{\text{tr}} = \sum D^i_{\text{ir}} + R, \tag{3}$$

where D^i_{ir} is the distance reduced between the ith individual resource and the nest, and R is the time-steps left when no individual resource exists in the field.

Autonomous task allocation After the sub-controllers above have been evolved, we combine them together by evolving the autonomous task allocation sub-controller. The sub-controller takes the number of individual resources, resource packages and robots in the range of the Omni-camera, and its outputs in the last time-step as inputs. The outputs indicate the "necessity" of performing each subtask, and the sub-controller corresponding to the subtask with the highest output is activated and takes control of the robot at each time-step.

To examine the evolvability, we perform two experiments with different fitness function settings. In the first experiment, we concentrate on the original objective, where the sub-controller is only rewarded for transporting individual resources to the nest. By contrast, the sub-controller is also rewarded for exploring the field and for decomposing resource packages in the second experiment to help bootstrap the artificial evolution. Equations (4) and (5) show the fitness functions.

$$F_1 = \sum D^i_{\text{ir}} \tag{4}$$

$$F_2 = \sum D^i_{\text{ir}} + 0.1 \cdot \frac{\sum P_s}{M} + 0.1 \cdot \sum D^j_{\text{rp}}, \tag{5}$$

where $\sum D^i_{\text{ir}}$ is the reward for achieving the original objective—transporting individual resources to the nest. $\sum P_s/M$ and $\sum D^j_{\text{rp}}$ are bootstrapping rewards for exploring the field and decomposing resource packages, respectively. The meaning of each notation is as described in the previous sections.

4.3 Comparison experiments

To make a comparison, we also perform experiments in which a conventional evolutionary robotics approach is adopted, where the controllers are based on a single artificial neural network. We develop robot controllers of two types, namely SingleA and SingleB. Both types adopt the architecture illustrated in Fig. 4, with the same inputs including all inputs used in the previous sections. The hidden layers of SingleA and SingleB have 20 nodes. The outputs of SingleA only control the two motors, and the gripper is set to be turned on all the time. In SingleB, the two motors and the gripper are all controlled by the outputs. For each type, we perform two experiments with Eqs. (4) and (5).

Table 1 Parameters for the (μ, λ) evolution strategy

Parameter	Set 1	Set 2
Parent μ	15	60
Offspring λ	100	400
Max generation	500	
Synaptic weight	$\in [-1.0, 1.0]$	
Mutation step size	$\in [0.0001, 0.2]$	
Initial mutation step size	0.05	
Trials	10	

4.4 Evolutionary algorithm settings

Considering the fact that synaptic weights of artificial neural networks are represented by real-value vectors, the (μ, λ) evolution strategy [29] is adopted. Taking into account the fact that the single artificial neural network controllers approximately have four times as many weights as the sub-controllers have, we evolved SingleA and SingleB controllers with four times larger population size. Table 1 illustrates the parameter settings, where the proposed approach employ Set 1 and the conventional evolutionary robotics approach adopt Set 2. Each candidate sub-controller (or controller) is evaluated with the average of five simulation runs because the task difficulty might be greatly changed by the randomly allocated resources. The simulation lasts for 3000 time-steps (of length 0.02 s each, henceforth) for the exploration, decomposition and transportation sub-controllers, and 6000 time-steps for the autonomous task allocation sub-controller and the controllers in the comparison experiments.

5 Results and discussion

The left part of Fig. 5 shows the fitness trajectories of the single artificial neural network controllers (the conventional approach) and autonomous task allocation sub-controllers (the proposed approach). As it can be observed in the figure, the proposed approach achieved much higher fitness than the conventional evolutionary robotics approach. All experiments of the conventional evolutionary robotics approach reached lower fitness plateaus within 150 generations, which can be considered that they were trapped in local optima. Additionally, as the fitness trajectories of the proposed approach show, compared to the fitness function concentrating on the final objective (see F_1), adding a bootstrapping part (see F_2) helped the fitness to grow smoothly and achieve the fitness plateau with fewer generations. It is worthy to note that the weight of the bootstrapping part in F_2 was selected through preliminary experiments, and we believe that better results can be obtained by careful fine-tuning.

Since the experiments had different fitness scale, we examined the performance quantitatively in terms of the number of individual resources collected and resource packages unpacked. As it can be seen in the middle and right part of Fig. 5, all controllers developed by the conventional evolutionary robotics approach obtained similar results where nearly ten individual resources were collected and no resource package was unpacked, which implies that the conventional evolutionary robotics approach failed to develop behaviors decomposing the resource packages. On the contrary, the composite controllers developed by the proposed approach with F_1/F_2 collected 22.96/25.64 individual resources, and unpacked 2.32/2.62 resource packages in average, respectively.

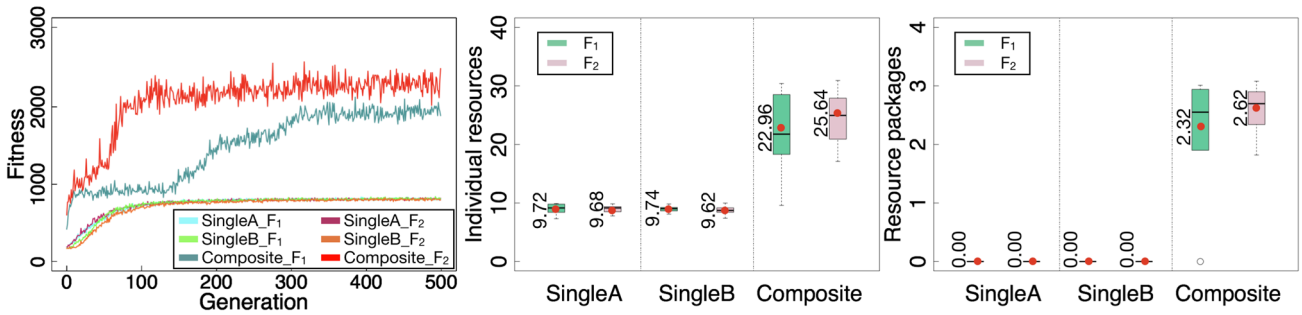

Fig. 5 Left: the fitness trajectories of all experiments, where each trajectory is the average of all ten trials. Composite denotes the composite artificial neural network-based controllers developed by the proposed approach. SingleA and SingleB are the single artificial neural network-based controllers obtained by the conventional evolutionary robotics approach. F_1 and F_2 denote the fitness function used during the artificial evolution. Middle and right: performance comparison between the proposed approach and the conventional evolutionary robotics approach, in terms of individual resources collected (middle) and resource packages unpacked (right), where each data point represents the result of the best controller of each trial

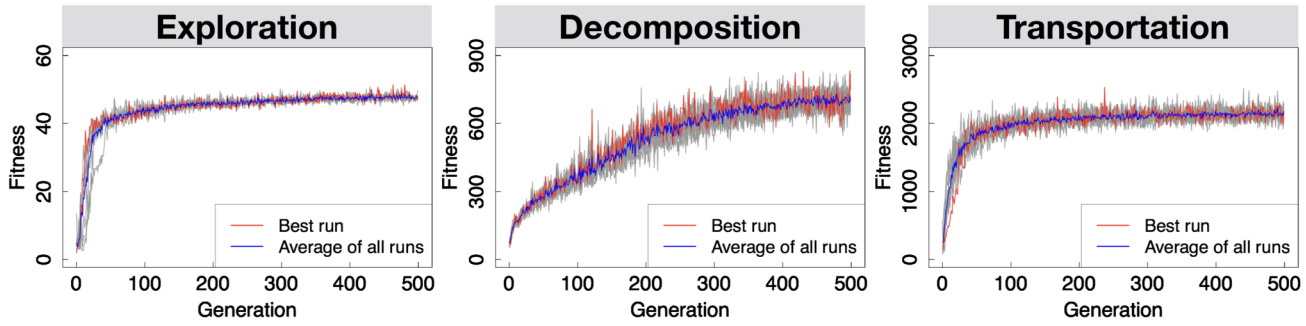

Fig. 6 Trajectories of the fitness in each trial of the partitioned subtasks

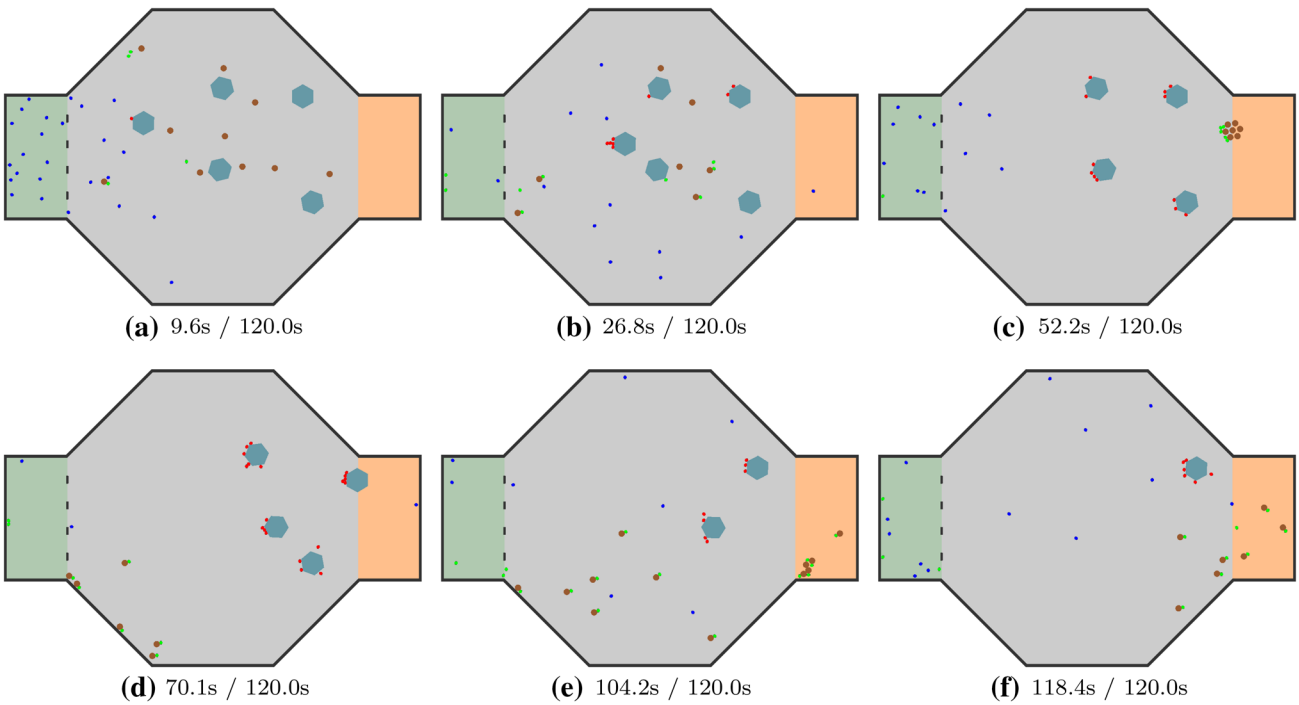

Fig. 7 Simulation snapshots of the best controller developed by the proposed approach. The color of the robot indicates whether it is performing the exploration subtask (blue), the decomposition subtask (red), or the transportation subtask (green). **a** 9.6 s/120.0 s, **b** 26.8 s/120.0 s, **c** 52.2 s/120.0 s, **d** 70.1 s/120.0 s, **e** 104.2 s/120.0 s, **f** 118.4 s/120.0 s. (Color figure online)

In our approach, task partitioning is adopted to reduce the difficulty of designing fitness functions. Figure 6 shows the fitness trajectories of the partitioned subtasks. As it can be observed in the graph, the fitness in the exploration and the transportation subtasks grew rapidly in the early stage and achieved the fitness plateau within 300 generations. The fitness in the decomposition subtask grew slow but smoothly and finally achieved the fitness plateau in the last 50 generations, which implies that the decomposition subtask was relatively difficult than the other subtasks for the artificial evolution. Additionally, considering the fact that all trials in each subtask produced similar solutions (in terms of fitness), it can be said that the fitness functions successfully built gradient to functional solutions.

Figure 7 shows the simulation snapshots of the best controller developed by the proposed approach. It can be clearly observed that an autonomous task allocation mechanism over the partitioned subtasks was acquired successfully.

6 Conclusion

In this study, we proposed a two-step scheme consisting of task partitioning and autonomous task allocation to address the bootstrap problem and deception. The proposed approach was demonstrated in a complex collective foraging problem by means of computer simulation. The given task was partitioned into three subtasks, and we evolved three sub-controllers to solve them, respectively, then the sub-controllers were combined together by evolving the autonomous task allocation sub-controller. We also performed comparison experiments in which a conventional evolutionary robotics approach is adopted, and the results show that the proposed approach leads to perform more effectively.

As future scope, we plan to examine the scalability and flexibility of the proposed method in more complex tasks and analyze the evolutionary acquisition of the autonomous task allocation. We are also interested in developing techniques enabling automatic task partitioning.

References

1. Brambilla M, Ferrante E, Birattari M, Dorigo M (2013) Swarm robotics: a review from the swarm engineering perspective. Swarm Intell 7(1):1–41
2. Şahin E (2004) Swarm robotics: from sources of inspiration to domains of application. International workshop on swarm robotics. Springer, Berlin, Heidelberg, pp 10–20
3. Trianni V, Nolfi S, Dorigo M (2008) Evolution, self-organization and swarm robotics. In: Blum C, Merkle D (eds) Swarm intelligence. Springer, Berlin, pp 1–41
4. Liu W, Winfield A (2010) Modelling and optimisation of adaptive foraging in swarm robotic systems. Int J Robot Res 29(14):1743–1760
5. Nolfi S, Floreano D (2000) Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines. MIT Press, Cambridge
6. Francesca G, Birattari M (2016) Automatic design of robot swarms: achievements and challenges. Front Robot AI 3:29
7. Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evol Intell 1(1):47–62
8. Soysal O, Şahin E (2005) Probabilistic aggregation strategies in swarm robotic systems. In: Proceedings of the 2005 IEEE swarm intelligence symposium, pp 325–332
9. Nouyan S, Campo A, Dorigo M (2008) Path formation in a robot swarm. Swarm Intell 2(1):1–23
10. Groß R, Dorigo M (2009) Towards group transport by swarms of robots. Int J Bio-Inspired Comput 1(1–2):1–13
11. Pini G, Brutschy A, Frison M, Roli A, Dorigo M, Birattari M (2011) Task partitioning in swarms of robots: an adaptive method for strategy selection. Swarm Intell 5(3–4):283–304
12. Gomez F, Miikkulainen R (1997) Incremental evolution of complex general behavior. Adapt Behav 5(3–4):317–342
13. Whitley LD (1991) Fundamental principles of deception in genetic search. Found Genet Algorithms 1:221–241
14. Agassounon W, Martinoli A, Goodman R (2001) A scalable, distributed algorithm for allocating workers in embedded systems. IEEE Int Conf Syst Man Cybern 5:3367–3373
15. Parker LE (1998) ALLIANCE: an architecture for fault tolerant multirobot cooperation. IEEE Trans Robot Autom 14(2):220–240
16. Krieger MJ, Billeter JB (2000) The call of duty: self-organised task allocation in a population of up to twelve mobile robots. Robot Auton Syst 30(1–2):65–84
17. Agassounon W, Martinoli A (2002) Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In: Proceedings of the first international joint conference on autonomous agents and multiagent systems: part 3. ACM, Bologna, pp 1090–1097
18. Brutschy A, Pini G, Pinciroli C, Birattari M, Dorigo M (2014) Self-organized task allocation to sequentially interdependent tasks in swarm robotics. Auton Agents Multi-agent Syst 28(1):101–125
19. Silva F, Duarte M, Correia L, Oliveriram SM, Christensen AL (2016) Open issues in evolutionary robotics. Evol Comput 24(2):205–236
20. Lehman J, Stanley KO (2011) Abandoning objectives: evolution through the search for novelty alone. Evol Comput 19(2):189–223
21. Lehman J, Stanley KO, Miikkulainen R (2013) Effective diversity maintenance in deceptive domains. In: Proceedings of the 15th annual conference on genetic and evolutionary computation. ACM, pp 215–222
22. Lehman J, Miikkulainen R (2014) Overcoming deception in evolution of cognitive behaviors. In: Proceedings of the 2014 annual conference on genetic and evolutionary computation (GECCO '14). ACM, pp 185–192
23. Christensen AL, Dorigo M (2006) Incremental evolution of robot controllers for a highly integrated task. In: International conference on simulation of adaptive behavior, pp 473–484
24. Togelius J (2004) Evolution of a subsumption architecture neurocontroller. J Intell Fuzzy Syst 15(1):15–20
25. Duarte M, Oliveira SM, Christensen AL (2015) Evolution of hybrid robotic controllers for complex tasks. J Intell Robot Syst 78(3–4):463–484
26. Celis S, Hornby G.S, Bongard J (2013) Avoiding local optima with user demonstrations and low-level control. In: Proceedings of the IEEE congress on evolutionary computation, pp 3403–3410
27. Von HE (1990) Task partitioning: an innovation process variable. Res Policy 19(5):407–418
28. Pini G, Brutschy A, Pinciroli C, Dorigo M, Birattari M (2013) Autonomous task partitioning in robot foraging: an approach based on cost estimation. Adapt Behav 21(2):118–136
29. Beyer HG, Schwefel HP (2002) Evolution strategies: a comprehensive introduction. Nat Comput 1(1):3–52