# An Algorithm for Self-Organized Aggregation of Swarm Robotics using Timer


Xinan Yan
School of Software and Shanghai Key Laboratory of Scalable Computing and Systems
Shanghai Jiao Tong University
Shanghai, China
yanxinan@sjtu.edu.cn

Alei Liang
School of Software and Shanghai Key Laboratory of Scalable Computing and Systems
Shanghai Jiao Tong University
Shanghai, China
liangalei@sjtu.edu.cn

Haibing Guan
Department of Computer Science and Shanghai Key Laboratory of Scalable Computing and Systems
Shanghai Jiao Tong University
Shanghai, China
hbguan@sjtu.edu.cn



***Abstract*—Aggregation is a basic collective behavior in biology, and is a prerequisite for many applications of swarm robotics. This paper proposes a new distributed algorithm for aggregation of swarm robotics under the constraints of no central control, no information about positions, and only local interaction among robots. Our control strategy contains two states, Search and Wait, for individual robot, as in the model of probabilistic finite state automata (PFSA). The main difference from PFSA is the way of how individual robot decides to leave from an aggregate. In our approach, each robot in an aggregate has a timer as its lifetime in the aggregate instead of having a leaving probability in PFSA. Further, to make all robots aggregate together, the key idea of our algorithm is that robots in small aggregate have shorter lifetime than those in large aggregate. The lifetime of individual robot in an aggregate is approximated as the lifetime of the aggregate, by making all the timers of the robots in the same aggregate have the same time settings, where the time is proportional to aggregate size and is updated only with the increase of aggregate size. By choosing linear function as the time setting of the timer, experiments based on simulator have been done. The results show that our algorithm is successful and scalable for large scale of robots, and indicate that the performance of the aggregation of all robots is improved by decreasing the lifetime of uncompleted aggregate in high robot density or increasing it in low robot density.**



***Keywords-aggregation; swarm robotics; swarm intelligence; distributed; self-organized***


## I. INTRODUCTION

In this paper, we study the aggregation of swarm robotics to understand the underlying coordination mechanism of large scale of robots inspired by the collective behaviors in biology. Aggregation is one of the basic researching problems in the field of swarm intelligence and swarm robotics, since it is the fundamental emergence of various forms of many collective tasks in biological systems and might be the prerequisite for many applications of swarm robotics that rely on local interaction, such as navigation, exploration, pattern formation, and etc. [1][2][3].

To develop a scalable algorithm for aggregation, this paper proposes a new algorithm for self-organized aggregation of swarm robotics under three conditions: (1) there is no central control that knows and controls all robots; (2) the robot has no idea about any information of its own or other robots' positions; (3) there is only local interaction among robots, that is, robots in long distance cannot detect or communicate with each other. In our approach, the robots have myopic abilities of detecting objects using infrared sensors and exchanging information through wireless communication, of which both are limited in short range.

In our control strategy, there are two states, Search and Wait, as in the model of probabilistic finite state automata (PFSA) [4]. In the beginning, all the randomly placed robots start in Search state. Then, the robots randomly traverse the arena to look for others. The encountered robots change to be in Wait state, in which the robots keep still. The group of encountered robots forms an aggregate. And the main difference between our approach and the PFSA model is the way of how robots in Wait state decide to leave from the aggregate they are in, after which the robots become searching. In PFSA model, each robot in an aggregate keeps calculating a probability, and decides to depart from the aggregate if the probability is smaller than a threshold value that is fixed or based on the feedback of neighbor count, after which the robot switches to be in Search state. Different from PFSA, in our approach, each robot has a timer, which starts timing when it joins in an aggregate, as its lifetime in the aggregate. When its timer is timeout, the waiting robot has to leave from the aggregate it is in. Another condition of robot's sate switching from Wait to Search is when the waiting robot detects no other robots around it in our algorithm as well as in PFSA.

Furthermore, to make all the distributed robots aggregate together, the key idea of our algorithm is that robots in small aggregate have shorter lifetime than those in large aggregate. All the timers of the robots in the same aggregate have the same time settings, where the time is proportional to aggregate size and is updated only when aggregate size increases. That is to say: if one or more searching robots join in an aggregate, the information of aggregate size increasing will be propagated in


This work was partly supported by the National Natural Science Foundation of China under Grant 60773093, 60970107, 60970108 and the Science and Technology Commission of Shanghai Municipality under Grant 09510701600, 10DZ1500200.



978-1-61284-052-9/11/$26.00 ©2011 IEEE

the aggregate based on local interaction. According to the new aggregate size, all the timers of the robots in the aggregate restart timing with a longer time than the time before the increase of aggregate size. On contrary, the departure of robots from an aggregate would not have influence on the lifetime of the rest robots in the aggregate. As a result, the lifetime of individual robot in an aggregate is approximated as the lifetime of the aggregate, since the timers of the robots in the same aggregate are almost simultaneously timeout, where the aggregate tries to dismiss in whole. With robot knowing the total number of robots, when all the robots aggregate together, they know that the aggregate is the final goal aggregate, where the timers of the robots are disabled so that the final aggregate does not dismiss any more. And the process of the whole aggregation is over. Except for the final aggregate, any formed aggregate during the course of aggregation is called uncompleted or unfinished aggregate in this paper.

By choosing the linear function, $T(x) = k \times x$, as the time calculation of the timers of the robots in uncompleted aggregates, our algorithm is examined by the experiments based on the Player/Stage simulator [5]. The results show that our control strategy is successful and scalable for large scale of robots. Moreover, the impact of that how long the lifetime of uncompleted aggregate is on the performance of the whole aggregation is analyzed by changing the constant parameter k of T(x). And the results indicate that the performance is improved by reducing the lifetime of uncompleted aggregate (decreasing k) in high robot densities or adding the lifetime of uncompleted aggregate (increasing k) in low density of robots.

The rest of this paper is organized as follows. Section II gives a brief review of the research in aggregation of swarm robotics. And our approach is described in detail in Section III. Moreover, section IV shows the detailed experiments and the results. Finally, this paper is concluded in section V.

## II. Related Work

In the field of aggregation of swarm robotics, a lot of outstanding effort has been made to study the control strategy. Two kinds of the existing algorithms for aggregation are probabilistic finite state automata (PFSA) and potential fields.

As a prior study of PFSA, Jeanson et al. [4] studied the aggregation behavior of real cockroaches. A model of PFSA was proposed and the dynamics of aggregation was explained in collective level. The PFSA model included two states, moving and waiting, for single robot. In the beginning, the robots were moving randomly in the environment until they encountered with others, after which the robots became waiting with a joining probability. Waiting robots had a leaving probability to be moving. Both the joining and leaving probability depended on the number of neighbors. In larger aggregate, robots had lower leaving probability and higher joining probability. With real micro-robots Alice, Garnier et al. [6] implemented a model of a self-enhanced aggregation based on the experiments in [4]. And the solution was applied to the "shelter selection" problem, where robots collectively selected an aggregation site among two identical or different predefined shelters. Later, a detailed description of the part of aggregation in [6] was given in [7]. In this paper, the faithful reproduction of the self-organized aggregation behavior of real cockroaches with a group of robots was reported. Based on [6], Garnier et al. [8] also demonstrated that the emergence of a collective decision at the level of the group could be explained by the self-amplified aggregation behavior of distinct zones in heterogeneous environment. Moreover, Soysal and Sahin [9] presented a PFSA-based aggregation strategy which combines three basic behaviors, approach, repel and wait, as a finite state machine. Correll and Martinoli [3] developed a probabilistic model for self-aggregation by keeping track the number of robots in clusters of specific size. In [10], Bayindir and Sahin modeled the self-organized aggregation of swarm robots based on a simple PFSA. The results of experiments showed that PFSA-based aggregation behaviors with fixed leaving probabilities or probabilities that were inversely proportional to neighbor count were unable to generate scalable aggregations in low robot densities.

Another kind of algorithm for aggregation is based on the study of potential functions (field). Gazi and Passino [11] presented a class of repulsion and attraction functions based on potential fields for swarm aggregations. The functions were odd functions where the repulsion and attraction acted in opposite directions. Moreover, the stability analysis was studied in this paper, which allowed an unbounded repulsion that prevented individuals from moving too close to each other. In later study, Gazi [12] extended the model in [11] as a general model by adding a sliding mode control to force agents to obey the dynamics of swarms. And more detail about the model in [12] and [11] were given in [13]. Moreover, based on potential field, a distributed control law for aggregation of a team of multiple kinematic agents was developed in [14][15]. The results of simulation indicated that a better size of swarm aggregation was got in the case of dynamic edge addition than in static case. Haghighi et al. [16] presented a new interactive force for self-aggregation, which was distinguished by four areas: separation area, neutral area, attractive area, and inactive area. Simulation showed that robots could pass obstacles to move toward the desired regions.

There are also many other distinguished studies for aggregation, of which only some are listed here. Trianni et al. [17] evolved the aggregation behaviors in a swarm of robots using artificial evolution. Bahceci and Sahin [18] analyzed the impact of parameters of evolutionary methods on the performance and scalability of aggregation behavior in swarm robotics. Yalcin [19] studied the cost of distinct fitness functions for evolving aggregation behavior for robot swarms. In addition, [20][21] studied the control and scalability of multi robots. In [22], a new control strategy for aggregation of multi robots using emotional learning was presented.

## III. Algorithm

The goal of our algorithm is to make all the randomly distributed robots aggregate together somewhere, not predefined, in the arena. To be scalable, our algorithm takes effect under the constraints of no central control, no information about positions for robots, and only local interaction among robots. The skeleton of our algorithm in global view is given as follows. With unique IDs and knowing the total number of robots (the size of the final aggregate),

randomly placed robots take a random walk in the arena to look for other robots. Once two robots encounter with each other, they stop as a start of an aggregate based on IR sensing and wireless communication. The aggregate is identified by the larger ID of the two robots, and both the two robots start the same timers. Robots in the same aggregate have the same timers, of which the time is calculated based on and proportional to the aggregate size. Any freely moving robot joins in the aggregate when it meets with any robot of an aggregate, after which all the robots in the aggregate are notified with the information of the growth of the aggregate, and they restart their timers with the same time, of which the time is longer than the time before the increase of the aggregate size. When the timer is timeout, the robot tries to leave from the aggregate it is in. Since timers of the robots in the same uncompleted aggregate are always timeout almost at the same time, the lifetime of an individual in an aggregate is approximated as the lifetime of the aggregate, where the aggregate tries to dismiss when timeout. Because the small aggregate has shorter lifetime than the large aggregate, robots in small aggregate are easy to become free moving and can participate in large aggregate. Once all the robots aggregate together, the timers of robots are disabled and the whole process of aggregation is over. Moreover, our algorithm is described in detail in the rest of this section from the view of an individual robot.

### A. *Robot Model*

In our algorithm, all the robots are identical. The robot is mobile with limited ability of interaction including IR sensing for detecting objects and wireless communication for communicating with other robots. The model of robot is illustrated in Fig. 1, where the most important component is the model of interaction. Total 12 IR sensors are evenly placed around the robot with a max detecting distance D. The range of wireless communication is approximated as a standard circle, where the max range is R that is calculated as the distance from the outside of robot to the boundary of the circle. Since the range of wireless communication is limited to be short by lowering the power supply, both the IR sensing and the wireless communication are in short range, where R is not more than D, so that the "Request-ACK" mechanism can be used for individual robot distinguishing other robots from obstacles or the boundary of arena.

### B. *Initialization*

Each robot is randomly placed in the arena with the total number of robots (GOAL-NUMBER), a unique number as identity (R-ID), and other properties as follows. Robot also has a list, called ID-List, to store the R-IDs of all the robots in the same aggregate. Robots in the same aggregate, as a group, have the same group-identity number, G-ID, which is defined in the creation of the aggregate. And G-Size is defined as the aggregate size, the number of robots in an aggregate, which is known by each robot in the aggregate and is updated while the aggregate grows or shrinks. Initially, the ID-List only contains the R-ID of the robot itself; G-ID equals to R-ID; and G-Size is one. This initialization means that the single robot can be seen as a mobile aggregate with size 1.

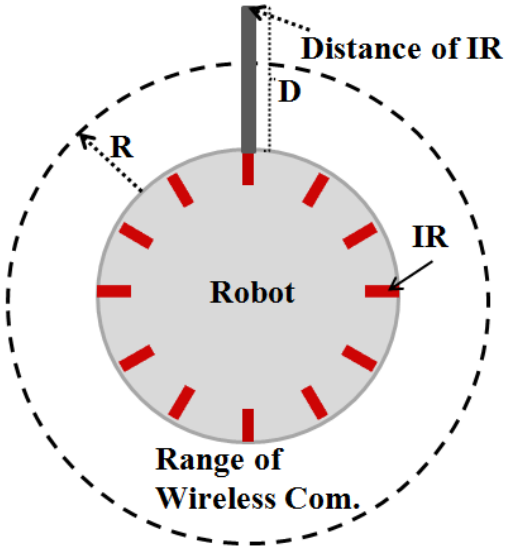


Fig. 1 The gray circle is the body of the robot in overhead view, on which the short rectangles represent the IR sensors. There are 12 IR sensors evenly arranged around the robot. D is the max sensing distance of IR. The outside dashed circle stands for the range of wireless communication that is approximated as a standard circle, and the max range, R, is limited not more than D by lowering the power supply.

Table I. CONTENT OF MESSAGES

| Msg. Type | Content |
|---|---|
| HELLO | R-ID of the sender |
| ACK | R-ID, G-ID, G-SIZE, ID-List, and State of the sender; |
| PROPAGATE | R-ID and G-ID of the first sender; Tag: 'joining' or 'leaving' |

Table II. SUMMARY OF MESSAGES

| Msg. Type | Sending Mode | Sender State | Receiver State |
|---|---|---|---|
| HELLO | Broadcast | Search | Search or Wait |
| ACK | P to P | Wait | Search |
| PROPAGATE | Broadcast | Wait | Wait |

### C. *Interaction Messages*

During the process of aggregation, there are three types of messages used in wireless communication: HELLO, ACK, and PROPAGATE. The main content of these messages are given in Table I. PROPAGATE message is a kind of message that will be propagated in the aggregate till all robots in the aggregate receive it at least once. And the PROPAGATE message contains the R-ID and G-ID of the first sender that starts the propagation of this PROPAGATE message. Different types of messages are sent and received by robots in different states including Search and Wait (see next section). The messages' sending mode and the corresponding states of the sender and receiver are summarized in Table II, where the sending mode includes '*P to P*' and '*Broadcast*'. '*P to P*' means that the sender sends the message to one specified receiver, while '*Broadcast*' refers to that the sender broadcasts the message without specifying who the receiver is. If one

message is received by a robot, of which the state is not the corresponding receiver state of the message, the robot will just ignore the message.

### D. States and Corresponding Behaviors

There are two states, Search and Wait, and corresponding behaviors for individual robot during the course of aggregation. The two states and corresponding behaviors of robot are respectively described in detail as follows.

**Search** – Each robot starts in this state. A searcher keeps moving forward and uses all its front IR sensors to detect objects including robots, obstacles, and boundary of the arena. If one object is detected, the searcher does not stop moving toward to the object until the distance between the searcher and the object is less than R that is the max range of wireless communication. Then, the "Request-ACK" mechanism is used to check whether the detected object is a robot. The searcher broadcasts a HELLO message and waits for an ACK message in a period time $T_{waiting}$. If there is no ACK message received by the searcher after $T_{waiting}$, the searcher will regard the detected object as an obstacle, and take the strategy of avoidance that it rotates to look for a free view in front and randomly chooses a free direction to go. The process of avoidance takes time $T_{avoiding}$, during which the searcher does not try to recognize other robots, that is, everything it detects in front is considered as obstacles. The way of randomly choosing free direction in the avoiding procedure makes the searching be a way of random walk.

**Wait** – A waiter keeps stopping with a timer, called W-Timer, as a count of its lifetime in the aggregate. The time of W-Timer is set according to formula T(G-SIZE), which is proportional to G-SIZE. W-Timer starts after a robot transfers into Wait state. Before W-Timer is timeout, the waiter uses all its IR sensors to realize the existence of its neighbors, and can receive messages through wireless communication. While ignoring any ACK message, the waiter deals with the HELLO and the PROPAGATE messages, of which the detail is illustrated in Fig. 2 and described as follows. (1) If a HELLO message is received, the waiter will send an ACK message back to the sender of the HELLO message; (2) If a PROPAGATE message is received, the waiter will check the R-ID and G-ID of the first sender in the PROPAGATE message; if the G-ID does not equal to the G-ID of the waiter, the waiter will do nothing; otherwise, if the R-ID in the PROPAGATE message does not exist in the ID-List of the waiter and the Tag of the PROPAGATE message is 'joining', the waiter will adds the R-ID of the first sender of the message to ID-List and increases its G-SIZE by 1; then, the waiter resets the time of the W-Timer using the updated G-SIZE, and broadcasts the received PROPAGATE message without changing anything, which can be received by all its neighbors; if the R-ID in the PROPAGATE message exists in the ID-List of the waiter and the Tag of the PROPAGATE message is 'leaving', the waiter just broadcasts the received PROPAGATE message after it removes the R-ID of the PROPAGATE message from its ID-List and decreases its G-SIZE by 1; and the waiter does nothing in any other situations.

```
Algorithm: Waiter processes received message
1:  switch (type of received message)
2:    case HELLO :
3:      sends an ACK message to the sender of HELLO;
4:      break;
5:    case PROPAGATE :
6:      if G-ID == G-ID of the first sender of the message
7:        if (R-ID of the first sender of the message is not in the
          ID-LIST) && (tag of the message is 'joining') then
8:          adds R-ID of the first sender of the message to ID-
      LIST;
9:          G-SIZE += 1;
10:         restart W-Timer with time T(G-SIZE);
11:         broadcast the received PROPAGATE message;
12:       else
13:         if (R-ID of the first sender of the message is in the
              ID-LIST) && (tag of the message is 'leaving')
              then
14:             removes R-ID of the first sender of the
      message from ID-LIST;
15:             G-SIZE -= 1;
16:             broadcast the received PROPAGATE message;
17:       end if
18:     end if
19:   end if
20:   break;
21:   default:
22:     do nothing;
```

Fig. 2 Robot in Wait state only deals with the HELLO and the PROPAGATE messages, and ignores any other type of messages.

### E. State Transitions

The transitions between the states, Search and Wait, are illustrated in Fig. 3. And the details are given as follows.

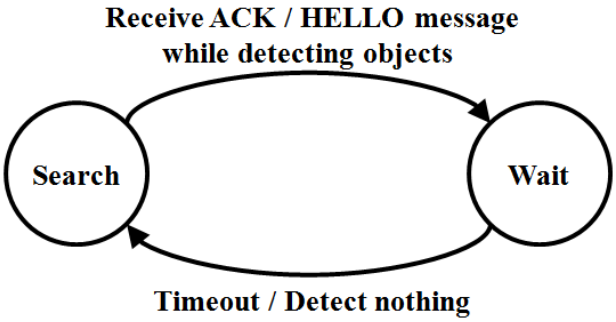


Fig. 3 Transition between the states: Search and Wait

**Search → Wait**: While one or more objects are detected by the IR sensors, a searcher will change to be a waiter when (1) if one or more ACK messages are received; (2) if one HELLO message is received, and the R-ID of the HELLO message is lower than the R-ID of itself. Notice that here are some actions the robot should take just after finishing the state transition. In (1), only the first ACK message is legal; and the robot changes its G-ID to equal to the received G-ID, sets its G-SIZE with the received G-SIZE + 1, and adds all the elements of the received ID-List to its own ID-List, after which the robot setups the W-Timer and broadcasts a PROPAGATE message. And in (2), the robot immediately sends an ACK message back to the sender of the HELLO message after it changes its state into Wait.

**Wait → Search**: A waiter becomes a searcher when (1) W-Timer is timeout, and the waiter broadcasts a PROPAGATE

message with 'leaving' tag; (2) Nothing is detected around by the IR sensors. Notice that to prevent oscillation between aggregates and searchers, the robot takes the strategy of avoidance (see the state of Search) in a time period $T_{avoiding}$ after it becomes a searcher, where the transition to Wait state is disabled. After the transition, the robot immediately does the following reset operations. The ID-List only contains the R-ID of the robot itself; G-ID = R-ID; and G-SIZE = 1.

Moreover, there are three points to be noticed. (1) Conflict in the creation of an aggregate. This problem means that when two searchers encounter with each other and try to be an aggregate through communication, there is one or more new coming robots meet with any of them. In our algorithm, the two robots, the start of an aggregate, ignore the new coming searcher before they finish the state transition. (2) Partial aggregation. This problem refers to that there are always several uncompleted aggregates occurring during the course of the aggregation of all robots, which is a general problem in aggregation. By providing the uncompleted aggregate with lifetime that is proportional to the aggregate size, the problem of partial aggregation is solved through our algorithm. (3) It is possible that the timers of all the robots in the same aggregate may not be almost synchronously timeout because of the delay of wireless communication, slightly difference of the timer frequency, or any other exception. When the error is small, all the timers are timeout only with little time difference in order. Since there is $T_{avoiding}$ time that prevents robots from doing the state transition from Search to Wait, all the robots still can leave from the aggregate together. When the error is large, what happens is that some robots tries to leave from the aggregate while others might keep staying in the aggregate. This kind of problem does not have influence on the success of the whole aggregation, because the leaving robots have no effect on the lifetime of the waiting robots, and the left aggregate dismisses with the time goes.

## IV. EXPERIMENT

The goal of experiments is to examine the success of our approach, by choosing the linear function, T(G-SIZE) = k × G-SIZE, as the time setting of the W-Timer of the robot in Wait state, where k is a constant parameter. Moreover, we study the scalability and the impact of the parameter k of T(G-SIZE), as a factor of the lifetime of uncompleted aggregate, on the performance of the whole aggregation. And all experiments are done based on Player/Stage simulator [5].

### A. Setup

In all experiments, the arena is set as 20 × 20 $m^2$ without any obstacles. The robot has a body shape looked as a circle in overhead view, and keeps constant velocity while moving. The parameters of the simulated robot are illustrated in Table III. The settings of the time $T_{waiting}$, $T_{avoiding}$ and the function T(G-SIZE) of W-Timer are given in Table IV. We respectively use total 10, 20, 30, 40 and 50 robots to examine our approach, where the total number of robots is the size of the final aggregate, called GOAL-NUMBER. And the constant parameter k is respectively chosen as 5, 10, 15, 20, 25 and 30. For each pair (GOAL-NUMBER, k), 10 experiments are done,

Table III. PARAMETERS OF SIMULATED ROBOT

| Parameter Name | Value |
|---|---|
| Body Size | radius = 0.24 m |
| Velocity | 0.5 m/s |
| Distance of IR Sensing | 0.8 m |
| Range of Wireless Communication | 0.65 m |

Table IV. TIME SETTINGS

| Time Name | Value (second) |
|---|---|
| $T_{waiting}$ | 3 |
| $T_{avoiding}$ | 5 |
| T(G-SIZE) | k × G-SIZE, k is constant |

and the average time cost is approximated as the performance. The time of the Player/Stage simulator is set to be synchronized with the real time. If the aggregation of one experiment is not finished in 5 hours, the aggregation is considered as a failure. Each experiment starts with random placement of robots.

### B. Result and Analysis

**Overall Success** – All the experiments of aggregation are successful only except for one failure that the aggregation of 10 robots with k = 5 does not succeed in 5 hours. The performance of all the experiments respectively with total 10, 20, 30, 40 and 50 robots and different k is shown in Table V. As an instance, the process of the aggregation of 30 robots with k = 20 is illustrated in Fig. 4. Moreover, to examine the success of our algorithm in higher robot density, additional experiments using total 100 robots with k = 5 are done. The aggregation is successfully achieved, and the result is illustrated in Fig. 5, where the total time cost of the aggregation of all the robots is 25 minutes and 4 seconds.

Table V. PERFORMANCE OF AGGREGATION

| Time of Aggregation (minutes) | | | | | |
|---|---|---|---|---|---|
| **k value** | **Total Number of Robots** | | | | |
| | **10** | **20** | **30** | **40** | **50** |
| **5** | – | 132.0 | 18.5 | 11.6 | 15.2 |
| **10** | 158.4 | 19.0 | 18.7 | 18.8 | 28.7 |
| **15** | 83.8 | 21.2 | 22.6 | 23.8 | 24.5 |
| **20** | 138.0 | 14.0 | 15.9 | 21.1 | 34.0 |
| **25** | 31.4 | 17.2 | 22.3 | 35.0 | 41.8 |
| **30** | 23.8 | 18.3 | 31.1 | 39.7 | 45.3 |

The reason why the aggregation of 10 robots with k = 5 fails contains two points: (1) the density of robots in the area is so low that the robots have less probability to encounter with other robots or an uncompleted aggregate; (2) uncompleted aggregate has too short lifetime and is easy to dismiss, since the time calculated by T(G-SIZE) with k = 5 is short. Both these two reasons make the uncompleted aggregate dismiss before any other searcher joins in it. So it is difficult for the aggregation of 10 robots with k = 5 to be achieved. In other experiments, the aggregation are all achieved, because the encountering probability among robots increases with the rise of the robot density, or the lifetime of uncompleted aggregate becomes long by increasing k.

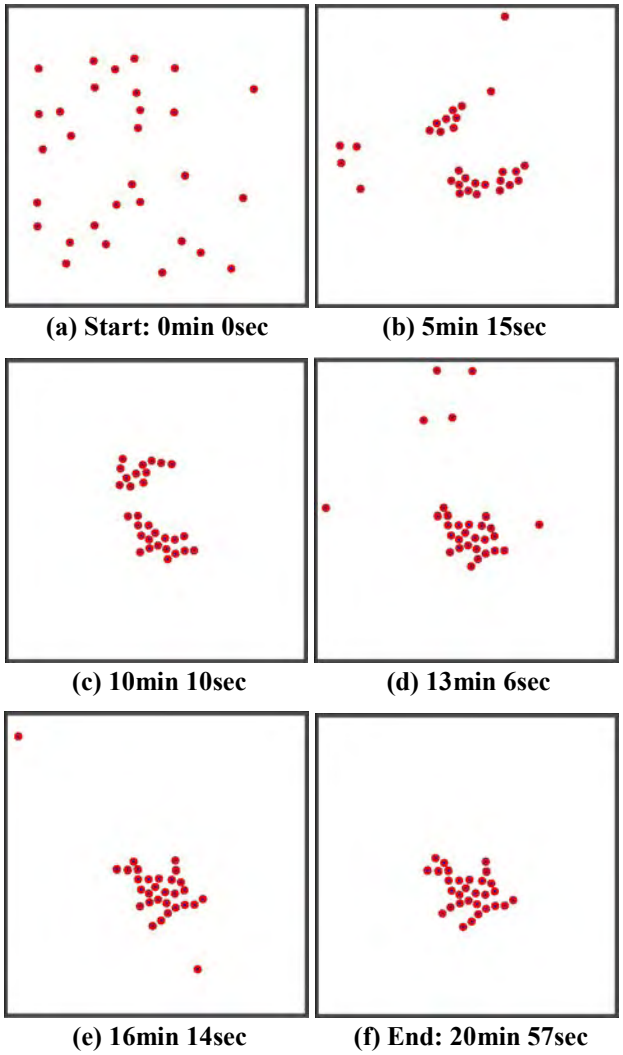


Fig. 4 Process of the aggregation of 30 robots with k = 20 of T(G-SIZE). The points stand for the robots, while the outside black square is the boundary of the arena. (a) is the start where 30 robots are placed randomly. (b) – (e) are the selected snaps of the process of aggregation in order. Firstly, the robots aggregate as several independent small groups. Later, some aggregates dismiss, and the robots join in other aggregates to produce large ones. And (f) is the final result of aggregation that all the robots aggregate together.

**Scalability** – To measure the scalability of our algorithm, we choose the ratio of the time cost of aggregation to the total number of robots as the metric, since it is reasonable that large number of robots may take longer to aggregate together than small number of robots. This kind of time cost ratio stands for the average time cost for each individual robot joining in the final aggregate, which is more reasonable than the total time cost on analyzing the scalability. Fig. 6 shows the comparison of the time cost ratio of different total number of robots under different fixed k. The time cost ratio of the aggregation of total 10 robots is always the highest including one failure when k = 5, compared with other aggregations of more total number of robots with the same k. Except for the aggregation of total 10 robots, when k is the same, with the increase of total number of robots, the time cost ratio keeps low, even is slightly improved

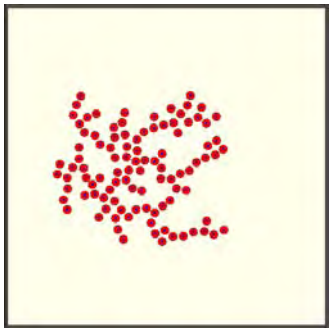

Fig. 5 This figure shows the final aggregation of 100 robots with k = 5 in the area of $20 \times 20$ m$^2$. The average time cost of 10 experiments is 25 minutes and 4 seconds, where the starting positions of robots are random in each experiment.

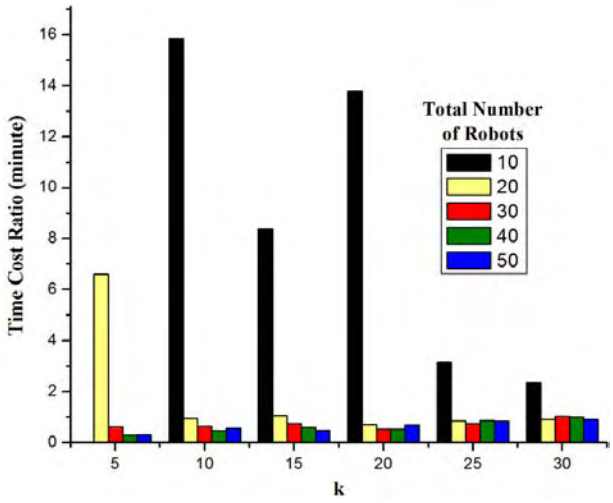


Fig. 6 With the same k, the comparison of time cost ratio of different total number of robots is illustrated. There is no time cost ratio for the aggregation of 10 robots with k = 5, since the aggregation is a failure. Lower the bar is, less average time cost for individual robot takes to participate in the final aggregate.

in some cases, such as that when k is 15. This illustrates the point that our algorithm is scalable for large scale of robots in high robot density.

**Performance** – Let's see the impact of the lifetime of uncompleted aggregate on the total time cost of the aggregation of all robots. Since the lifetime is a linear function T(G-SIZE) = k × G-SIZE, increasing k increases the lifetime, whereas the lifetime is decreased by lowing k. So k is used to represent the lifetime. Fig. 7 shows the impact of k on the performance of aggregation in low robot density, while Fig. 8 illustrates the impact of k on the performance of aggregation in high robot density. Though there are some shakes, the trends of lines in both the two figures indicate that the performance of aggregation can be improved by decreasing the lifetime of uncompleted aggregate in high robot densities or increasing it in low density of robots. This is reasonable. Because the encountering probability among robots is high in crowded arena, the uncompleted aggregate needs less time to wait for a searching robot joining in high robot density than the uncompleted aggregate does in low robot density.

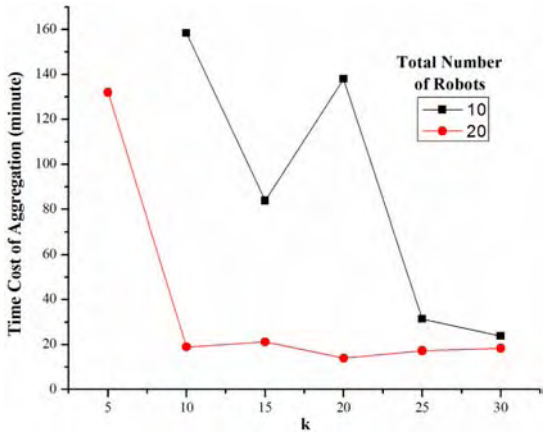


Fig. 7 Impact of k on performance in low robot density

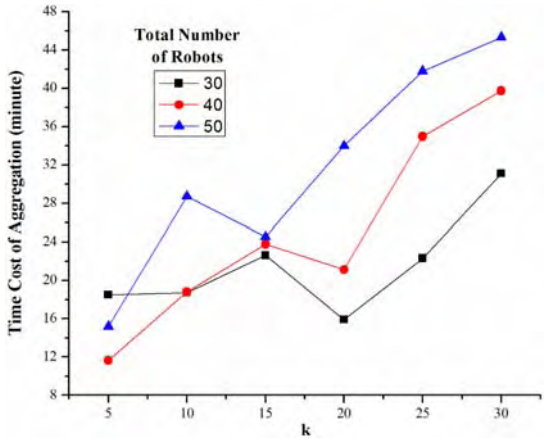


Fig. 8 Impact of k on performance in high robot density

## V. CONCLUSION

The main contribution of this paper includes threefold. (1) A new distributed control strategy for the aggregation of swarm robotics using timer is proposed, under the conditions of only local interaction among robots, no information of positions, and no central control unit. (2) The success and scalability of our approach is examined by experiments in simulation, by choosing the linear function as the calculation of the lifetime of the robots in uncompleted aggregates. The results show that our algorithm is successful and scalable for large scale of robots. (3) The impacts of both the robot density and the lifetime of uncompleted aggregate on the performance are analyzed, and the results of experiments indicate that the performance of the aggregation of all robots is improved by decreasing the lifetime of uncompleted aggregate in high robot densities or increasing it in low density of robots.

## REFERENCES


[1] G. Beni. “From swarm intelligence to swarm robotics”, Swarm Robotics, Lecture Notes in Computer Science, vol. 3342, 2005, pp. 1-9.

[2] L. Bayindir and E. Sahin. “A review of studies in swarm robotics”, Turkish Journal of Electrical Engineering & Computer Sciences, vol. 15, no. 2, 2007, pp. 115-147.

[3] N. Correll and A. Martinoli. “Modeling self-organized aggregation in a swarm of miniature robots”, In Proceedings of International Conference on Robotics and Automation Workshop on Collective Behaviors inspired by Biological and Biochemical Systems, 2007, Rome, Italy.

[4] R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost and G. Theraulaz. “Self-organized aggregation in cockroaches”, Animal Behaviour, vol. 69, no. 1, Jan. 2005, pp. 169 – 180.

[5] Player/Stage. Online: http://playerstage.sourceforge.net/

[6] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari and G. Theraulaz. “Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots”, Advances in Artificial Life, Lecture Notes in Computer Science, vol. 3630, 2005, pp. 169-178.

[7] S. Garnier, C. Jost, J. Gautrais, M. Asadpour, G. Caprari, R. Jeanson, A. Grimal and G. Theraulaz. “The embodiment of cockroach aggregation behavior in a group of micro-robots”, Artificial Life, MIT Press, vol. 14, no. 4, 2008, pp. 387-408.

[8] S. Garnier, J. Gautrais, M. Asadpour, C. Jost and G. Theraulaz. “Self-organized aggregation triggers collective decision making in a group of cockroach-like robots”, Adaptive Behavior: Animals, Animats, Software Agents, Robots, Adaptive Systems, vol.17, no. 2, Jun. 2009, pp. 109-133.

[9] O. Soysal and E. Sahin, “Probabilistic aggregation strategies in swarm robotic systems”, In Proceedings of the IEEE Swarm Intelligence Symposium, 8-10 Jun. 2005, Pasadena, California, USA, pp. 325 – 332.

[10] L. Bayindir and E. Sahin. “Modeling self-organized aggregation in swarm robotic systems”. In Proceedings of IEEE Swarm Intelligence Symposium, Mar. 30 2009-Apr. 2 2009, Nashville, TN, pp. 88-95.

[11] V. Gazi and K. M. Passino. “A class of attraction/repulsion functions for stable swarm aggregations”. In Proceedings of Conference on Decision and Control, 10-13 Dec. 2002, Las Vegas, USA, vol. 3, pp. 2842-2847.

[12] V. Gazi. “Swarm aggregations using artificial potentials and sliding mode control”. In Proceedings of 42nd IEEE Conference on Decision and Control, 9-12 Dec. 2003, Maui, Hawaii, USA, vol. 2, pp. 2041-2046.

[13] V. Gazi. “Swarm aggregations using artificial potentials and sliding-mode control”. IEEE Transactions on Robotics, vol. 21, no. 6, Dec. 2005, pp. 1208-1214.

[14] D. V. Dimarogonas and K. J. Kyriakopoulos. “Connectedness preserving distributed swarm aggregation for multiple kinematic robots”, IEEE Transactions on Robotics, vol. 24, no. 5, Oct. 2008, pp. 1213-1223.

[15] D. V. Dimarogonas and Kostas J. Kyriakopoulos. “Connectivity preserving distributed swarm aggregation for multiple kinematic agents”, In Proceedings of 46th IEEE Conference on Decision and Control, 12-14 Dec. 2007, New Orleans, LA, USA, pp. 2913-2918.

[16] R. Haghighi and C. C. Cheah. “Self-aggregation in multi-agent shape control”, In Proceedings of 2010 IEEE Conference on Robotics Automation and Mechatronics, 28-30 Jun. 2010, Singapore, pp. 212-217.

[17] V. Trianni, R. Grob, T. H. Labella, E. Sahin and M. Dorigo. “Evolving aggregation behaviors in a swarm of robots”. Advances in Artificial Life, Lecture Notes in Artificial Intelligence, vol. 2801, 2003, pages 865-874.

[18] E. Bahceci and E. Sahin. “Evolving aggregation behaviors for swarm robotic systems: a systematic case study”, In Proceedings of Swarm Intelligence Symposium, 8-10 Jun. 2005, Pasadena, USA, pp. 333 – 340.

[19] C. Yalcin. “Evolving aggregation behavior for robot swarms: a cost analysis for distinct fitness functions”, In Proceedings of 23rd International Symposium on Computer and Information Sciences, 27-29 Oct. 2008, Istanbul, Turkey, pp. 1 – 4.

[20] X. Chen and Y. M. Li, "Smooth Formation Navigation of Multiple Mobile Robots for Avoiding Moving Obstacles", International Journal of Control, Automation, and Systems, Vol. 4, No.4, pp.466-479, 2006.

[21] Y. M. Li and X. Chen, "Control and Stability Analysis on Multiple Robots" , The 2nd International Conference on Autonomous Robots and Agents(ICARA04), New Zealand, Dec. 13-15, 2004, pp.158-163.

[22] S. Etemadi, R. Vatankhah, A. Alasty and G. Vossoughi. “Swarm aggregation using emotional learning based intelligent controller”, In Proceedings of 6th International Symposium on Mechatronics and its Applications, 23-26 Mar. 2009, Sharjah, UAE, pp. 1-6.