Modeling Self-Organized Aggregation in Swarm Robotic Systems

Levent Bayındır and Erol Şahin

***Abstract*— In this paper, we propose a model for the self-organized aggregation of a swarm of mobile robots. Specifically, we use a simple probabilistic finite state automata (PFSA) based aggregation behavior and analyze its performance using both a point-mass and a physics-based simulator and compare the results against the predictions of the model. The results show that the probabilistic model predictions match simulation results and PFSA-based aggregation behaviors with fixed probabilities are unable to generate scalable aggregations in low robot densities. Moreover, we show that the use of a leave probability that is inversely proportional to the square of the neighbor count (as an estimate of aggregate size) does not improve the scalability of the behavior.**

I. Introduction

In this paper, we study the dynamics of self-organized aggregation of a swarm of mobile robots. Specifically, we assume that the robots have myopic sensing abilities that allow them to perceive only a small part of the arena, and do not have access to information such as their (or other robots') position, the size of the arena or the number of robots. Under these constraints, the development of scalable aggregation behaviors, as well as models, that provide insight to the dynamics of self-organized aggregation are in need.

One basic approach to solve aggregation problem is the use of probabilistic finite state automata (PFSA) based controllers in which probabilistic transitions among a number of primitive behaviors are used.

In one of the pioneering studies, Jeanson et al. [7] modeled the aggregation behavior of cockroaches as a PFSA and validated the model predictions with results of real cockroach experiments. The PFSA-based behavior of cockroaches consisted of two states and their corresponding behaviors: *random walk* and *wait*. The cockroaches started in *random walk* state and randomly traverse the environment until they find another cockroach. The cockroaches detecting other cockroaches switched to *wait* state with a *joining probability*. Similarly, cockroaches in *wait* state returned back to the *random walk* state with a certain *leave probability*. An important property of this study was its reliance on *the neighbor count feedback*. The *joining* and *leaving* probabilities of a cockroach depended on the number of neighbors that cockroach sensed. The cockroaches had higher probability to wait on and lower probability to leave from larger aggregates. In this study, cockroaches also used environmental heterogeneity and aggregated near walls instead of the center of the arena.

This work is funded by TÜBİTAK (Turkish Scientific and Technical Council) through the "KARİYER: Kontrol Edilebilir Robot Oğulları" project with number 104E066.

The simulations are run on the HPC Cluster of the Department of Computer Engineering, Middle East Technical University.

Levent Bayındır is currently enrolled in the Faculty Development Program (ÖYP) at Middle East Technical University on behalf of Atatürk University.

Later, Garnier et al. [5] implemented the solution proposed by Jeanson et al. [7] on real robots and validated obtained results with results of [7]. The solution was also applied to the *shelter selection* problem in which robots are allowed to aggregate only inside predefined shelter locations and are expected to select one of these shelters. The study showed that robots select collectively an aggregation site among two. Later, the aggregation part of this study is presented with more details in [4].

Ame et al. [2] set the probability of leaving a shelter based on a simple formula which makes the probability inversely proportional with the number of neighbors in that shelter. Monte Carlo simulations showed that the formula is adequate for the individuals to select one of two shelters. In a subsequent study [1], Ame et al. developed a detailed model that took more than two shelters and the capacity of shelters into account.

Corell and Martinoli [3] developed a probabilistic macroscopic model for aggregation. The model kept track of the number of robots in aggregates of specific size based on the waiting and leaving probabilities of robots and the encountering probability of searching robots with aggregates. The authors compared the results of a simulation model with the results of the macroscopic model and showed that the model predicts the aggregation dynamics successfully.

Soysal and Şahin [9] extended the controller in [7] by adding an *approach* behavior as a sub step prior to the *wait* behavior in which they can control the distance of a robot to its aggregate while waiting. A simulated robot which has infrared sensors for obtaining local information and a sound sensor/emitter pair for obtaining information from longer distances are used. In this study, robots can emit sound, hear the sound in the environment and estimate the highest sound direction. Despite using a powerful sensing capability, the aggregations obtained were shown to be unscalable when the number of robots is increased.

In this study, we use a simple PFSA-based aggregation behavior and analyzed its performance using a point-mass simulator, a physics-based simulator and a probabilistic model. Our contribution is threefold. First, a probabilistic model for aggregation, which is more realistic and faster than previous microscopic probabilistic model implementations, is proposed. Second, this probabilistic model is validated with simulation experiments in the lack and existence of neighbor count feedback. Third, we show that a previously proposed formula [2] for utilizing the neighbor count feedback is insufficient to solve the aggregation problem in a scalable way.

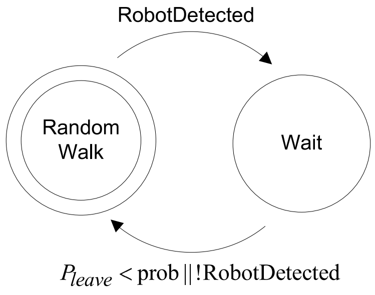


Fig. 1. Probabilistic finite state automata (PFSA) representing the robot controller. PFSA has two states: *random walk* and *wait*. The transitions between states are controlled with P_{leave} control parameter and robot detection event.

II. Aggregation Controller

The aggregation controller used is a variation of the PFSA based controller used in previous studies [7] [9]. The PFSA representing the robot controller has two states and their corresponding behaviors: *random walk* and *wait* (See Figure 1).

The robots start in *random walk* state wandering aimlessly in the environment. When a robot in the *random walk* state (a searching robot) detects another robot, it switches to the *wait* state.

The transition from the *wait* state to the *random walk* state can occur in two ways. First, if a waiting robot no longer detects other robots around itself, it switches back to *random walk* state. Second, if a waiting robot has one or more neighbors, it switches to the *random walk* state with probability P_{leave}.

In previous implementations [7] [9], waiting robots stop when the *wait* state is activated and keep stopping until they switch to the *random walk* state. Compared to previous implementations, our wait behavior is more sophisticated. First, each waiting robot try to keep its distance to its neighbors at a predefined value o_{des}. Second, a small amount of noise is added to the movements of the robots in the *wait* state. These extensions are observed to create more compact aggregates.

In order to prevent oscillations between aggregates and searching robots, when a waiting robot switches to the *random walk* state, the transition to the *wait* state is disabled for certain time period (T_{leave}).

III. Kobot Robot Platform and Physics-based Simulator

The target robot platform for our study is the Kobot mobile robot [12] shown on Figure 2(a). It is a circular, CD-sized (robot radius, $R_r = 6\ cm$) differential-drive robot platform. The robot is equipped with eight infrared sensors around its body 45° apart from each other (Figure 2(b)). Each sensor can sense up to $\sim 20\ cm$ (R_s) and returns an integer pair (o_k, r_k), where $o_k \in \{0, 1, \cdots, 7\}$ denotes the detection level to the object being sensed by the k^{th} sensor ($o_k = 1$ and $o_k = 7$ indicate respectively a far and nearby object, whereas $o_k = 0$ stands for no detection), and $r_k \in \{0, 1\}$ indicates whether

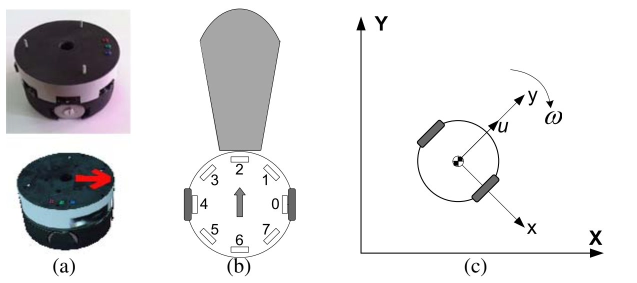


Fig. 2. (a) Photo of a Kobot (top) and its model in physics-based simulator (bottom). (b) The scaled sketch. The circle represents the body. The small rectangles show the placement of the IR sensors placed evenly at multiples of 45° around the body. The blob indicates the sensing range. (c) The reference frame is fixed to the center of the robot where the x-axis coincides with the rotation axis of the wheels. The forward velocity (u) is along the y-axis. ω denotes the angular velocity of the robot.

the sensed object is an obstacle or a kin robot where r_k equals to 1 if k^{th} sensor of the robot detects a robot and 0 otherwise. The infrared sensor value corresponding to the desired distance between robots ($o_{des} = 21\ cm$ - center to center distance between robots) is o'_{des} and equals to 3.

In this study, we use a physics-based simulator instead of real robots in order to verify our results. In physics-based simulator, the physical properties and physical interactions of the robots are implemented using a physics engine. The infrared sensors of Kobot are also modeled realistically in this simulator [12].

The behaviors in the physics-based simulator (which can be directly transferred to real robots) are implemented as follows. An acceleration vector ($\vec{a}$), denoting the desired movement direction for a robot, is calculated as the normalized sum of the virtual forces effected on each robot:

$$\vec{a} = \frac{\sum_{k=0}^{7} f_k e^{i\phi_k}}{\sum_{k=0}^{7} f_k} \tag{1}$$

where $k \in \{0, 1, \cdots, 7\}$ denotes the sensor positioned at angle $\phi_k = \frac{\pi}{4}k$ with the x-axis of the body-fixed reference frame and f_k is the virtual force calculated for sensor k.

In the *random walk* state, the virtual force (f_k) is calculated as

$$f_k = \frac{-(o_k)^2}{C}$$

where o_k is the distance measured with sensor k and C is a scaling constant set to 35.

In the *wait* state, the robots approach to their neighbors to keep them at a desired distance ($o'_{des} = 3$). The virtual force vector for the *wait* state is calculated as

$$f_k = \begin{cases} \frac{-(o_k - o'_{des})^2}{C}, & \text{if } o_k \geq o'_{des} \\ \frac{(o_k - o'_{des})^2}{C}, & \text{otherwise} \end{cases} \tag{2}$$

Figure 3 plots f_k versus o_k for both *random walk* and *wait* behaviors.

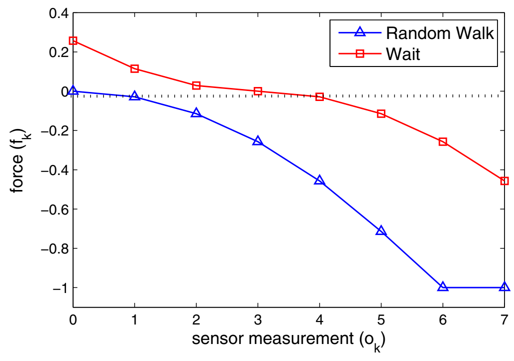


Fig. 3. The magnitude of the virtual force (f_k) is drawn with respect to o_k for both random walk and wait behaviors. Higher values of o_k indicate closer distances. The force function is saturated within [0, 1]. Negative force values indicate repulsive forces and positive force values indicate attractive forces.

In order to apply acceleration vector ($\vec{a}$) to the robots, the acceleration vector $\vec{a}$ of the robot is mapped to forward velocity (u) and angular velocity (ω). u is calculated as

$$u = \begin{cases} (\frac{\vec{a}}{\|\vec{a}\|}.\hat{a}_c)\ u_{max} & \text{if } \frac{\vec{a}}{\|\vec{a}\|}.\hat{a}_c \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\hat{a}_c$ is the current direction of the robot parallel to the y-axis of the body-fixed reference frame and u_{max} (0.07 m/s) is the maximum forward velocity.

The angular velocity (ω) of the robot is controlled by a proportional controller using the deviation of the desired angle from the current direction of the robot.

$$\omega = (\angle\vec{a}_c - \angle\vec{a})K_p + \delta \tag{4}$$

where K_p is the proportionality constant of the controller and $\angle(.)$ computes the argument of a vector. δ is the noise value derived from uniform distribution on the interval $[-0.5, 0.5]$. In the *random walk* state, δ is used to create a random walk pattern while moving. In *wait* state, the noise adds perturbation to make aggregates more compact. The rotational speeds of the right and left motors are calculated and applied to motors as described in [11].

IV. Point-Mass Simulator

In the point-mass simulator, the robots are considered as points which can move any direction they choose immediately with a speed up to u_{max} and can sense other robots up to a distance of o_{max} (sensing range from center of a robot to the center of another: $\sim 32\ cm$). Periodic boundaries are used for the arena in the point-mass simulator. Once the robots reach these boundaries they appear on the other side with the same direction and continue moving. The details of the kinetic equations used to simulate movement of the robots are described as follows.

The behavior of a swarm of M robots in a 2-dimensional Euclidean space is simulated. The position of robot i of the swarm is described by $x_i \in \mathbb{R}^2$ where $1 \leq i \leq M$. Each robot can sense up to a distance of $o_{max} \in \mathbb{R}$. The sensory information obtained by robot i from its sensors are described by

$$s_{ij} = \frac{d_{ij}}{\|d_{ij}\|}\left(8 - \left\lceil\frac{\|d_{ij}\| - 2R_r}{R_s/7}\right\rceil\right) \tag{5}$$

where $\forall j \in N_i$, $\|s_{ij}\| \in \{1, ..., 7\}$, R_r is the Kobot radius and R_s is the sensing range of the Kobot's infrared sensors. N_i is the neighborhood set of robot i and defined as

$$N_i = \{j : 1 \leq j \leq M, \|d_{ij}\| \leq o_{max}, i \neq j\} \tag{6}$$

and d_{ij} is the distance vector between robot i and robot j:

$$d_{ij} = x_j - x_i \tag{7}$$

Similar to the controller implementation in the physics-based simulator, an acceleration vector is calculated as the desired movement direction for each robot in the point-mass simulator. The acceleration vector for *random walk* behavior of robot i, a_i, is calculated as the sum of the virtual forces proportional to the square of the distance to the neighbor robots

$$a_i = -\sum_{j\in N_i}\frac{\|s_{ij}\|s_{ij}}{C}$$

where C is the same scaling constant used in physics-based simulator implementation.

The acceleration vector for the *wait* state, is calculated as

$$a_i = \sum_{j\in N_i} f_{ij}$$

where f_{ij} is calculated as

$$f_{ij} = \begin{cases} \frac{-\|s'_{ij}\|s'_{ij}}{C}, & \text{if } \|s_{ij}\| \geq o'_{des} \\ \frac{\|s'_{ij}\|s'_{ij}}{C}, & \text{otherwise} \end{cases} \tag{8}$$

where $o'_{des} = 3$ and s'_{ij} is defined as

$$s'_{ij} = s_{ij} \cdot \frac{\|s_{ij}\| - o'_{des}}{\|s_{ij}\|}$$

In the point-mass simulator, the acceleration vector is multiplied with the maximum robot speed (u_{max}) to obtain the velocity of robot i

$$v_i = (a_i + \delta)u_{max}, \qquad \|v_i\| < u_{max} \tag{9}$$

where δ is a random vector with a magnitude in interval $[0,\ 0.1]$ which has the same purpose of δ used for physics-based simulator. The velocity is then used to update the robot position as

$$x_i = x_i + v_i\Delta t \tag{10}$$

where Δt is duration of the simulation step.

V. Probabilistic Model

The main idea of our probabilistic model is to represent the aggregation dynamics as a sequence of probabilistic events. Specifically, we assume that there are three types of probabilistic events that can occur: *growing of an aggregate*, *creation of an aggregate* and *shrinking of an aggregate*. The probabilities corresponding to these events are denoted by $P_{grow}(m)$, P_{create} and $P_{shrink}(m)$ subsequently, where m is the size of the aggregate that will grow or shrink. The model is iterated probabilistically using these probabilities which are calculated by considering geometrical aspects, robot characteristics and time step range.

The model proposed here partially builds on the models proposed by Martinoli [8] and Corell [3].

A. Growing and Creation of an Aggregate (P_{grow} and P_{create})

An aggregate of size m grows when one searching robot finds this aggregate in the environment. The simplest form of this event is growing of an aggregate of size one. The growing probability for an aggregate of size one is calculated as

$$P_{grow}(1) \approx \frac{2(o_{max})v\Delta t}{A_{total}} \tag{11}$$

where A_{total} is the area of the arena, v is the average speed of an individual, o_{max} is the sensing range of an individual and Δt is the time discretization of the system.

The derivation of this equation is depicted on Figure 4. The figure shows the change of the position of a searching robot at one time step (Δt) in an environment of size A_{total}. The area swept by the searching robot at that time step equals to $2o_{max}v\Delta t$. o_{max} is the sensing range from center of a robot to the center of another robot. Defining o_{max} in this way allows us to consider the aggregate of size one as a point in the environment.

Since the searching robot can only detect the aggregate if the aggregate is inside the swept region, the growing probability equals to the probability of having a randomly selected point inside the swept area which is the ratio of the swept area to the total area of the environment (Equation 11).

A new aggregate is created when two searching robots find each other. Experiments show us that the creation probability of an aggregate (P_{create}) approximately equals to the growing probability of an aggregate of size one for relatively large environments.

$$P_{create} \approx P_{grow}(1) \tag{12}$$

Calculating growing probability for larger aggregates ($m > 1$) requires to consider the area covered by the aggregates. For a searching robot, the probability of finding an aggregate of size m in the arena is calculated as

$$P_{grow}(m) \approx \frac{2(o_{max} + R_m - (o_{des}/2))v\Delta t}{A_{total}}, \quad m > 1 \tag{13}$$

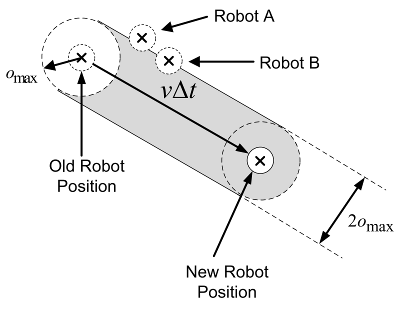


Fig. 4. The sketch representing the movement of a searching robot with an average speed (v) for a duration of a simulation step (Δt) where o_{max} is the sensing range from center of a robot to the center of another robot. The searching robot sweeps an area of size $2o_{max}v\Delta t$ (shown in gray). The waiting robots whose center is inside the swept area are detected by the searching robot. While robot A, whose center is outside the swept area, will not be detected by the searching robot; robot B, whose center is on the surface of the swept area, will be detected by the searching robot. The center of robots are indicated by cross signs.

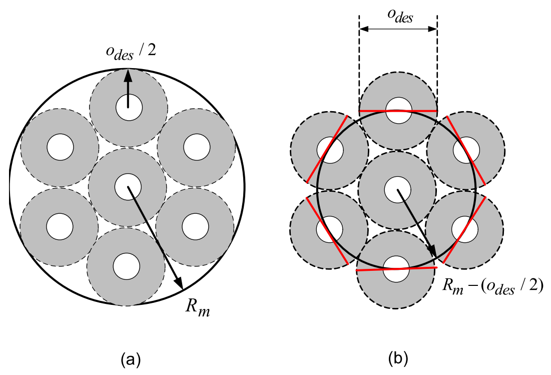


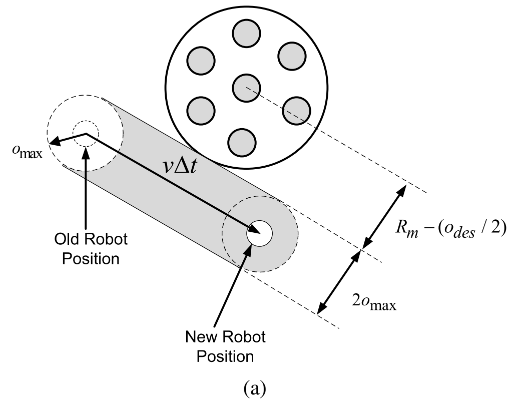
Fig. 5. The sketches of an aggregate with seven robots. (a) The aggregates are assumed to form circles with a radius of R_m. When the robots are o_{des} away from each other, we can calculate R_m with the help of circle packing theory. (b) Both growing probability for large aggregates ($m > 1$) and shrinking probability are calculated using the circle whose radius is $o_{des}/2$ units smaller than R_m.

where o_{des} is the center-to-center desired distance between robots and R_m is the approximate radius for an aggregate of size m (Figure 5-a). In this formula, the area swept by one searching robot is extended by $R_m - (o_{des}/2)$ units (Figure 5-b) to include the regions in which the aggregate with radius R_m can be detected by the searching robot. This is depicted on Figure 6.

R_m is calculated with the help of circle packing theory [6]. Circle packing is the arrangement of circles inside a given boundary such that no two overlap and have a specified pattern of tangencies. In our context, we regard robots as circles with a diameter of o_{des} and assume the given boundary, which correspond to the area of the aggregates, has the shape of a circle. For an aggregate with radius R_m (as depicted on Figure 5-a), following relation is known to hold from the best known packings of equal circles in the unit circle [10]

$$\frac{R_m}{o_{des}/2} \approx am^b \tag{14}$$

where m is the number of robots in the aggregate, a (~ 1.20)



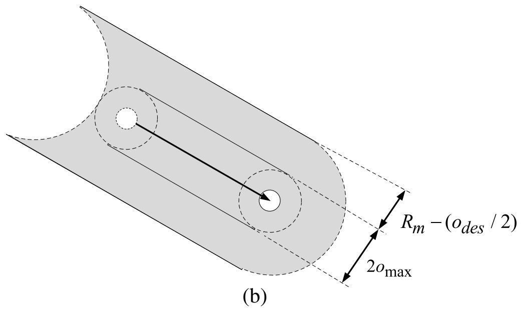


Fig. 6. The sketches represent the calculation of the growing probability for large aggregates. (a) An aggregate with radius R_m is detectable by a searching robot if it is located up to $R_m - (o_{des}/2)$ units away from the surface of the swept area of the searching robot. (b) The area obtained by extending the swept area of a searching robot by $R_m - (o_{des}/2)$ units: $2(o_{max} + R_m - (o_{des}/2))v\Delta t$.

and b ($\sim$ 0.48) are constants. After some rearrangements on equation 14, we can calculate the radius of the aggregate as

$$R_m \approx \frac{a o_{des} m^b}{2} \tag{15}$$

B. Shrinking of an Aggregate (P_{shrink})

An aggregate of size m shrinks when one of the waiting robots belonging to this aggregate leaves from the aggregate. In previous probabilistic model studies (e.g. [3]), it is assumed that any of the robots inside an aggregate can leave their aggregates. However, this is unrealistic since some of the robots inside an aggregate will be covered by other robots which will prevent them to leave.

Our aggregation controller which forces the robots to form compact aggregates, allows us to estimate the number of robots that can leave an aggregate. The probability of an aggregate of size m to shrink is calculated as

$$P_{shrink}(m) \approx N_p(m) P_{leave} \tag{16}$$

where $N_p(m)$ is the approximate number of robots on the periphery of the aggregate and P_{leave} is the leaving probability of a robot.

The number of robots on the periphery of an aggregate with m robots can be estimated by finding how many robots can be placed on the circle with radius $R_m - (o_{des}/2)$

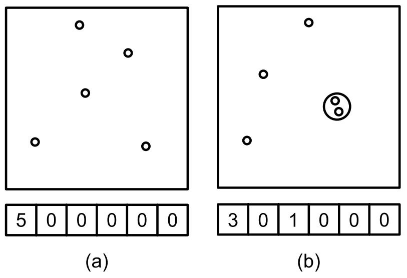


Fig. 7. Demonstration of the iteration of the probabilistic model using an array of size $N + 1$ where N is the number of robots in the environment ($N = 5$ for this example). The content of the array is updated when probabilistic events occur. In this demonstration, the array content (a) is updated when two searching robots meet (b).

(Figure 5-b). This number can be found approximately by dividing the periphery of the circle to o_{des}

$$N_p(m) \approx \frac{2\pi(R_m - (o_{des}/2))}{o_{des}} \tag{17}$$

When we substitute R_m with the right hand side of equation 15 and simplify the result, we obtain $N_p(m)$ as

$$N_p(m) \approx \begin{cases} \pi(am^b - 1), & \text{if } m \geq 6 \\ m, & \text{otherwise} \end{cases} \tag{18}$$

The second part of the equation is introduced for small aggregates ($m < 6$) where any waiting robots belonging to these aggregates can leave their aggregates.

C. Iteration of the probabilistic model

The model keeps an array for tracking the number of aggregates for all possible aggregate sizes. Initially, all elements in this array are assumed to be zero except the position zero which keeps track of the searching robots. This corresponds to the swarm state in which all robots are in random walk state (Figure 7-a). In such a state, the only possible event is the creation of new aggregates by searching robots. When a new aggregate is created, the value of the array at position two, which corresponds to the number of aggregates of size two, is incremented by one and the value of the array at position zero is decremented by two (Figure 7-b).

In subsequent iterations, the number of events that can happen in the environment increases. However, since the shrinking of an aggregate and the events that can happen to a searching robot are independent, the iteration of the model can be divided into two phases: *iteration of each searching robot* and *checking each aggregate for shrinking*. Checking each aggregate for shrinking is done probabilistically by calculating the shrinking probability of an aggregate calculated using equation 16

There are three mutually exclusive events that can occur for searching robots: (1) forming a new aggregate by finding another searching robot, (2) growing any of the existing aggregates or (3) continuing searching in the arena. The event occurred for each searching robot is selected with a mechanism similar to the roulette wheel selection in genetic algorithms. At each time step, a bar is created for deciding

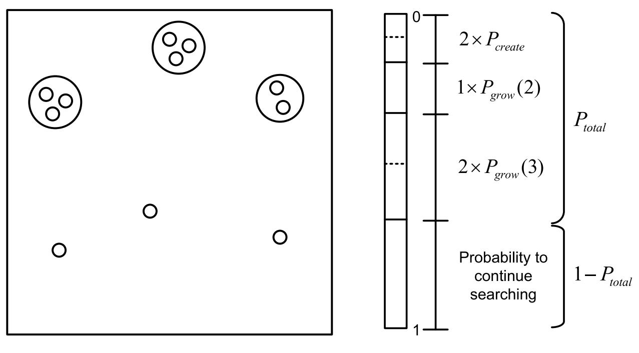


Fig. 8. The action for a searching robot is selected with a mechanism similar to roulette wheel selection. (a) Example swarm state having two aggregates of size three, one aggregate of size two and three searching robots in the environment. (b) The bar created for the swarm state at the left. The event for a searching robot is decided by selecting a random position inside this bar.

on the event occurred for each searching robot. On this bar, each of the possible events occupy space proportional to its probability. The event occurred for each searching robot is decided by randomly picking a position on this bar. The event corresponding to the area containing the selected position is then executed by updating the array used for tracking the swarm state. The bar created for searching robots in an example swarm state is shown on Figure 8.

In order to keep simplicity and obtain faster execution, two assumptions are made in the iteration of the model. First, it is assumed that only one robot can *join to* a specific aggregate within one time step. This assumption usually holds due to the small duration of a step, as long as the robots are not in an overcrowded arena.

Second, it is assumed that only one robot can *leave from* a specific aggregate in one time step. This looks like an invalid assumption for very large aggregates since each robot in the periphery of an aggregate can leave independently and the number of robots on the periphery of aggregates increases when the aggregates get larger. However, if this assumption is shown to be invalid for very large aggregates, it is possible to change the iteration of the model to exclude this assumption by iterating the shrinking part of the iteration by checking leaving probability for each robot in the periphery. Although this will lead to slower execution, the iteration will be still faster than a pure microscopic model since the behaviors of robots inside the aggregate are still ignored at each iteration.

D. Neighbor count feedback

As discussed in Section I, in some of the previous studies, the neighbor count feedback is used in order to obtain better results. In our study, we implemented the update formula proposed in [2]

$$P_{leave} = \frac{a}{1+X_i^2} \tag{19}$$

where P_{leave} is the leaving probability of robot i, X_i is the number of neighbors of robot i and a is constant.

In the point-mass simulator, the number of neighbors of robot i is defined as the cardinality of the neighborhood set of robot i

$$X_i = |N_i| \tag{20}$$

For the physics-based simulator, the number of neighbors of robot i can be estimated as the number of infrared sensors detecting a robot

$$X_i = \sum_{k=0}^{7} r_k \tag{21}$$

where r_k equals to 1 if k'th sensor of the robot detects a robot and 0 otherwise.

However, in order to utilize equation 19 in our probabilistic model, we need a way to determine the number of neighbors of robot i (X_i). Since the robots in an aggregate are not handled individually and the spatial properties of these robots are ignored in our probabilistic model, the estimation of the number of neighbors of a robot is nontrivial. One simple way is to use the aggregate size as the number of neighbors. However, this is unrealistic since robots have limited sensing ranges and may not sense the whole aggregate.

The number of neighbors of a robot depends on three main factors: the aggregate size, the position of the robot in its aggregate and the sensing range of the robot. In this study, we ignore the position of a robot in its aggregate, fix the sensing range to Kobot's sensing range and calculate the probability of having specific number of neighbors for a specific aggregate size experimentally on the point-mass simulator.

Figure 9 shows the results which have two interesting points to note. First, the robots can not discriminate aggregates of different sizes starting from aggregates of size three. Second, majority of robots estimate the size of their aggregate as two or three, no matter what the actual size is.

These probability distribution functions are used in our probabilistic model for finding the number of neighbors sensed (X_i) by a robot in an aggregate of a specific size (m). Once X_i is specified probabilistically, P_{leave} is calculated using equation 19 and afterwards $P_{shrink}(m)$ is calculated using equation 16.

VI. Results

The experiments are conducted using the probabilistic model, point-mass simulator and physics-based simulator. The default environment size is 20x20 m^2. The number of robots are changed in two of the experiments in order to analyze the scalability of the strategies.

Each bar in the figures represents an average value over 25 runs. The standard deviation over the 25 runs is shown as the error bar. Each run in simulators is started with a different placement of robots.

Two performance metrics are used to calculate the performance of the solutions at the end of each run: *largest aggregate size* and *largest aggregate size ratio*. While the

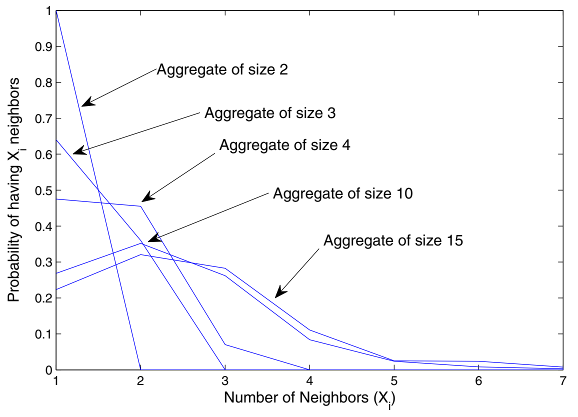


Fig. 9. Result of the point-mass simulator experiment performed to understand how robots sense their aggregates. A probability distribution function is plotted for each aggregate size. Each probability distribution function specifies how the members of an aggregate estimate their number of neighbors.

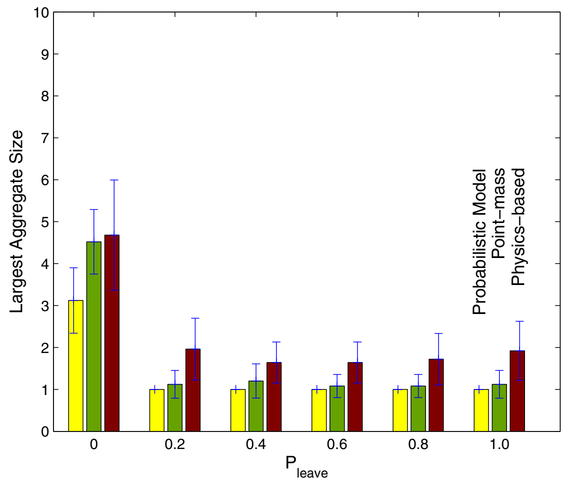


Fig. 10. The average of the largest aggregate size for ten different P_{leave} values. The most trivial solution which has P_{leave} value of zero has the highest performance. 10 robots are used in an arena of size 5x5 m^2.

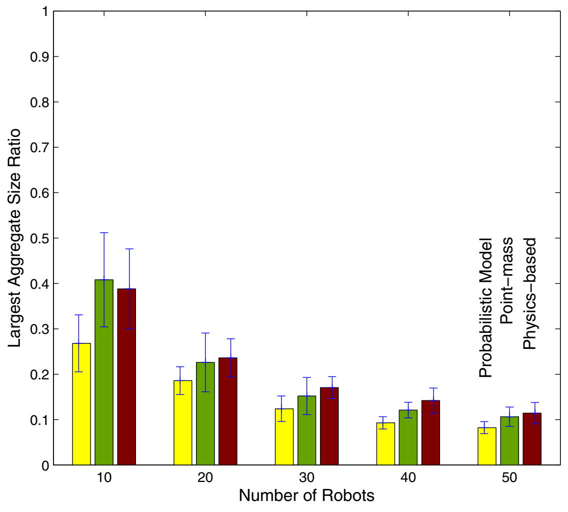


Fig. 11. The change for the ratio of the largest aggregate size to the total number of robots when the number of robots increased in an arena of size 20x20 m^2. The performance of the most performing strategy decreases.

first metric is the size of the largest aggregate at the end of a run, the second metric is the ratio of the first metric to the number of robots.

All runs are executed for 100.000 steps which corresponds approximately to 3 hours of real robot experiments. The searching robots move with maximum speed of 0.07 m/s and T_{leave} is set to 100 seconds for both of the simulators. The duration of the simulation step (Δt) is set to 0.11 seconds for both the probabilistic model and simulators.

A. Aggregation without Neighbor Count Feedback

Figure 10 shows the average of the largest aggregate size for different P_{leave} values. The experiment is performed with 10 robots inside an arena of size 5x5 m^2. The most successful strategy is the one which has P_{leave} value of zero. This corresponds to the strategy of joining the first aggregate observed and never leaving from it. The experiment results for the probabilistic model and simulators match quantitatively for the selected performance metric. Although the figure is generated with limited number of P_{leave} values, extensive number of experiments are performed in order to be make sure that there is no P_{leave} value with a significant performance improvement over the strategy with P_{leave} value of zero.

Figure 11 shows how scalable the strategy with best performance ($P_{leave} = 0$) is when the number of robots is increased in an environment of size 20x20 m^2. The metric used is the ratio of the average of the largest aggregate size to the total number of robots. Although, this ratio is expected to be constant in the ideal case, it decreases when the robot density is increased. The model predictions match to the predictions of the simulators quantitatively for the selected performance metric.

B. Aggregation with Neighbor Count Feedback

Figure 12 shows the change of the largest aggregate size for different a values when the neighbor count formula is integrated to the system. The best performance obtained is slightly better than the performance obtained without the neighbor count integration. Although for higher robot densities (e.g. 10 robots in an arena of size 2.5x2.5 m^2) the performance of the system with neighbor count feedback is almost doubled compared to the system without neighbor count (as reported in [3], [5] and [7]), this result shows that the neighbor count formula proposed in [2] is insufficient to solve the aggregation problem for the environments with low robot densities.

Figure 13 shows how scalable the strategy with best performance ($a = 10^{-5}$) is when the number of robots is increased.

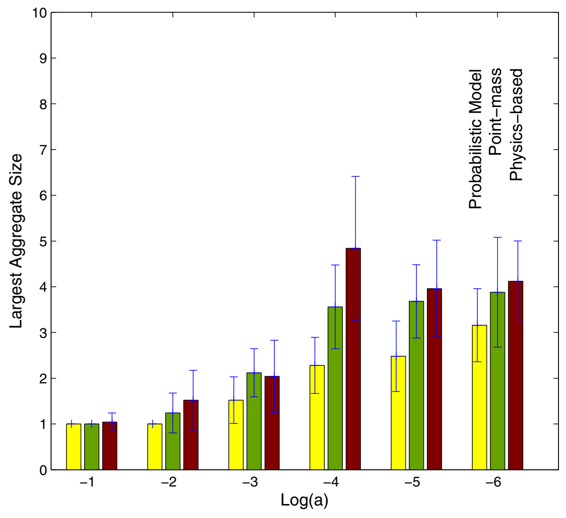


Fig. 12. The change of the largest aggregate size for different a values when the neighbor count formula (Equation 19) is used. The size of the environment is 20x20 m^2 and 10 robots are used in the experiments.

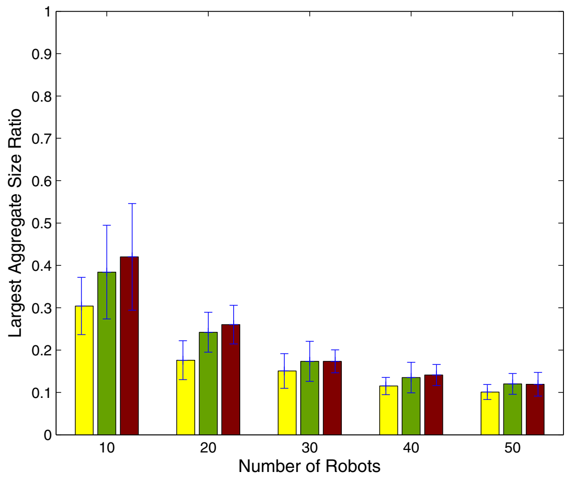


Fig. 13. The change for the ratio of the largest aggregate size to the total number of robots in the environment when the number of robots increased for the most performing strategy ($a = 10^{-5}$). As in the corresponding experiment without neighbor count feedback, the performance decreases.

VII. Conclusions

The contribution of this paper is threefold. First, a probabilistic microscopic model, which is more realistic and faster than previously proposed microscopic probabilistic models, is proposed. The model is more realistic than previous models because it has a better estimation for the shrinking probability of aggregates. It is also faster than previous probabilistic models because aggregates are considered as a whole and are checked once for shrinking at each iteration.

Second, simulators are used to verify the probabilistic model *with* and *without* the neighbor count feedback (a commonly used feedback in previous studies). It is shown that the model predicts the swarm behavior successfully in spite of its assumptions and use of approximated data.

Third, an important question related to existing studies is raised and tried to be answered: Is it possible to obtain optimum and scalable performance for aggregation by using the neighbor count feedback? In order to answer this question, a formula proposed in a previous study [2] for utilizing the neighbor count feedback is used. Although the formula is proposed as a solution to the aggregation problem, it is only tested on the shelter selection problem in the original study.

In existing studies, shelter selection is implicitly considered as a simplified version of the aggregation. Although two problems (aggregation and shelter selection) have the same core dynamics, the latter one simplifies the problem by preventing searching robots to form new aggregates at locations other than predefined shelter locations. The creation of aggregates other than predefined shelter locations and using larger environments in aggregation make the swarm system to reach equilibrium state more slowly. In addition to these two additional difficulties, Ame et al. [2] also reported that the time required to reach equilibrium states even for shelter selection problem increases exponentially with the number of individuals. These reasons make the formula proposed in [2] is incapable of solving the aggregation problem.

References

[1] J. Ame, J. Halloy, C. Rivault, C. Detrain, and J. Deneubourg. Collegial decision making based on social amplification leads to optimal group formation. *PNAS*, 103:5835–5840, 2006.

[2] J. Ame, C. Rivault, and J. Deneubourg. Cockroach aggregation based on strain odour recognition. *Animal Behaviour*, 68:793–801, 2004.

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

[4] 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*, 14:387–408, 2008.

[5] 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. In *8th European Conference on Artificial Life*, 2005.

[6] Wolfram Research Inc. Circle packing – from wolfram mathworld. http://mathworld.wolfram.com/CirclePacking.html, 12 2008.

[7] R. Jeanson, C. Rivault, J. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz. Self-organized aggregation in cockroaches. *Animal Behaviour*, 69:169–180, 2005.

[8] A. Martinoli, A. J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: from probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29:51–63, 1999.

[9] O. Soysal and E. Şahin. Probabilistic aggregation strategies in swarm robotic systems. In *Proc. of the IEEE Swarm Intelligence Symposium*, Pasadena, California, June 2005.

[10] E. Specht. The best known packings of equal circles in the unit circle (up to n = 720). http://hydra.nat.uni-magdeburg.de/packing/cci/, 12 2008.

[11] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2:97–120, 2008.

[12] A. E. Turgut, F. Gökçe, H. Çelikkanat, L. Bayındır, and E. Şahin. Kobot: A mobile robot designed specifically for swarm robotics research. Technical report, METU-CENG-TR-2007-05, 2007.