A PROBABILISTIC GEOMETRIC MODEL OF SELF-ORGANIZED AGGREGATION IN SWARM ROBOTIC SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

LEVENT BAYINDIR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

SEPTEMBER 2012

Approval of the thesis:

A PROBABILISTIC GEOMETRIC MODEL OF SELF-ORGANIZED AGGREGATION IN SWARM ROBOTIC SYSTEMS

submitted by **LEVENT BAYINDIR** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Erol Şahin
Supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Göktürk Üçoluk
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Erol Şahin
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Buğra Koku
Mechanical Engineering Dept., METU _____

Assist. Prof. Dr. Cengiz Toğay
Computer Engineering Dept., Çanakkale Onsekiz Mart University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: LEVENT BAYINDIR

Signature :

ABSTRACT

A PROBABILISTIC GEOMETRIC MODEL OF SELF-ORGANIZED AGGREGATION IN SWARM ROBOTIC SYSTEMS

Bayındır, Levent

Ph.D., Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Erol Şahin

September 2012, 125 pages

Self-organized aggregation is the global level gathering of randomly placed robots using local sensing. Developing high performance and scalable aggregation behaviors for a swarm of mobile robots is non-trivial and still in need, when robots control themselves, perceive only a small part of the arena, and do not have access to information such as their position, the size of the arena or the number of robots.

In this thesis, we developed a non-spatial probabilistic geometric model for self-organized aggregation as a tool to analyze aggregation. The model consists of four formulas for predicting the probabilities of aggregation events: creation, growing, shrinking and dissipation of an aggregate. The creation probability is derived mathematically using kinetic theory of gases. In order to derive formulas for growing, shrinking and dissipation probabilities, first, it is assumed that aggregates formed by robots are circular. Then, these formulas are derived geometrically using circle packing theory.

We proposed an aggregation behavior and implemented this behavior in the Stage multi-robot simulator. The behavior consists of four sub-behaviors: search, wait, leave and change

direction. The wait sub-behavior is specially designed to force aggregates to be circular so that our assumption for the model holds in simulation experiments.

We verified each formula using simulation experiments conducted in the Stage multi-robot simulator. Through systematic experiments, we showed that model predictions and simulation results match well and the formulas proposed for growing and shrinking probabilities predict these probabilities better for larger aggregates compared to predictions of previous self-organized aggregation models.

We also conducted experiments, in which certain aggregation events are disabled systematically, in order to verify the model further and show that our model can be used to predict the steady-state performance of generic simulation experiments. We use two different methods to predict the steady state performance with our model: microscopic model execution and steady state analysis. It is shown that the largest aggregate size, the number of aggregates, the number of searching robots and the aggregate distributions at the steady state-obtained from microscopic model execution, steady state analysis and simulation experiments are close to each other and our model can be used to predict steady-state performance of aggregation experiments.

Keywords: aggregation, model, modeling, aggregation model, swarm, swarm robotics

ÖZ

SÜRÜ ROBOT SİSTEMLERİNDE BİRARAYA GELME DAVRANIŞI İÇİN OLASILIKSAL GEOMETRIK BİR MODEL

Bayındır, Levent
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Erol Şahin

Eylül 2012, 125 sayfa

Biraraya gelme, rastgele yerleştirilen robotların yerel algılama ile global olarak bir araya gelmesidir. Robotların içinde bulundukları ortamın küçük bir kısmını algılayabildiği, kendi pozisyonları, ortam alanı veya robot sayısı gibi bilgilere sahip olmadıkları varsayıldığında, bir gezer robot sürüsü için üstün performanslı ve ölçeklenebilir biraraya gelme davranışının geliştirilmesi kolay değildir ve bu özelliklere sahip biraraya gelme davranışlarına ihtiyaç vardır.

Biraraya gelmeyi analiz edebilmek amacıyla, uzaysal-olmayan olasılıksal ve geometrik bir biraraya gelme modeli geliştirilmiştir. Model, biraraya gelme olaylarının olasılıklarını tahmin etmek için kullanılan dört formülden oluşmaktadır: topluluğun oluşması, büyümesi, küçülmesi ve yok olması. Oluşma olasılığı, gazların kinetik teorisi kullanılarak matematiksel olarak türetilmiştir. Büyüme, küçülme ve yok olma olasılıkları için, robot topluluklarının daire şeklinde olduğu varsayılmıştır. Daha sonra, bu formüller daire sıkıştırma teorisi kullanılarak geometrik olarak türetilmiştir.

Bir biraraya gelme davranışı Stage çoklu-robot benzetimcisinde gerçekleştirilmiştir. Bu dav-

ranış şu alt-davranışları içermektedir: arama, bekleme, ayrılma ve yön değiştirme. Bekleme alt-davranışı, toplulukların dairesel olmasını sağlamak amacıyla özel olarak tasarlanmıştır. Bu sayede model varsayımları ile davranış gerçekleştirimi örtüşmektedir.

Modeldeki tüm olasılık formülleri Stage benzetimcisinde gerçekleştirilen deneylerle doğrulanmıştır. Model tahminleri ile benzetim sonuçlarının birbiriyle uyuştuğu ve büyüme ile küçülme olasılıkları için önerilen formüllerin, önceki modellerin tahminlerine göre daha doğru olduğu gösterilmiştir

Son olarak da, modelimizin, herhangi bir biraraya gelme deneyinin sonucunu tahmin etmek için kullanılabileceğini göstermek amacıyla ek deneyler gerçekleştirilmiştir. Belirli biraraya gelme olaylarının sistematik olarak devre dışı bırakıldığı bu deneylerde, kararlı durum performanslarını model aracılığıyla elde etmek için iki ayrı yöntem kullanılmıştır: mikroskopik model yürütmesi ve kararlı durum analizi. Benzetim sonuçları ile bu yöntemlerden elde edilen sonuçların şu performans kriterleri cinsinden örtüştüğü gösterilmiştir: en büyük topluluk boyutu, toplulukların sayısı, arayan robotların sayısı ve topluluk dağılımları.

Anahtar Kelimeler: biraraya gelme, modelleme, biraraya gelme modeli, sürü, sürü robotlar

To my family

ACKNOWLEDGMENTS

First of all, I would like to express my sincerest thanks to my supervisor Assoc. Prof. Dr. Erol Şahin for his guidance, motivation, and continuous support throughout this study. Without his valuable ideas and support, this study would have never been possible.

I am grateful to Prof. Dr. Göktürk Üçoluk and Assoc. Prof. Dr. Afşar Saranlı for their valuable comments during my thesis monitoring committee meetings. I also would like to express my gratitude to Assist. Prof. Dr. Buğra Koku and Assist. Prof. Dr. Cengiz Toğay for their review of my thesis and their valuable comments.

This work was partially funded by the TUBİTAK "KARİYER: Kontrol Edilebilir Robot Oğulları" project with number 104E066. The simulations are performed on the High Performance Computing Center of the Department of Computer Engineering, METU.

Being currently enrolled in Faculty Development Program (ÖYP) in Middle East Technical University on behalf of Atatürk University, I am indebted to thank all people involved in Faculty Development Program.

I would like to thank Kovan alumni, Ali Emre Turgut, Fatih Gökçe, Hande Çelikkanat, Onur Soysal, Emre Uğur, and Erkin Bahçeci for their support. I would give a special thank to Ali Emre Turgut for supporting me during tough times.

I would also like to thank to my close friends in the department. I am grateful to my friends, Alev Mutlu, Fatih Titrek, Gülşah Tümüklü Özyer, and Serdar Çiftçi. We spent many years together in the department and they have been always there when I need help.

Finally, a special thank you goes to all members of my family. They have always given me their unconditional love and supported me in my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

Introduction

Swarm robotics [19] is a new approach to the coordination of large numbers of relatively simple robots. The approach takes its inspiration from the system-level functioning of social insects which demonstrate three desired characteristics for multi-robot systems: robustness, flexibility and scalability.

While *robustness* is the degree to which a system can still function in the presence of partial failures individuals or other abnormal conditions, *flexibility* is the capability of the system to adapt to new, different, or changing requirements of the environment and *scalability* is the ability of the same system to support larger numbers of individuals without impacting performance considerably. Replicating these properties observed in self-organized systems with mobile robots lead robotics researchers to study on small scale problems such as *pattern formation*, *aggregation*, *chain formation*, *hole avoidance* and *flocking*.

Among these problems, self-organized aggregation is one of the mostly studied problems in swarm robotics [60] [55] [8] [56] [17]. It is the global level gathering of randomly placed robots using local sensing. It can be considered as the basis of other collective tasks such as flocking and pattern formation and used in the nature from unicelular organisms to animals in order to protect themselves against predators better and conserve energy [13].

Developing high performant aggregation behaviors with mobile robots is easy when a centralized control approach is used or robots have global information such as the location of other robots. However, the problem becomes non-trivial when robots control themselves and 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 position, the size of the arena or the number of robots. The use of a trivial approach such as moving to the closest robot fails to be a solution since

it generates several small number of aggregates. Under these constraints, the development of scalable high performant aggregation solutions, as well as models, that provide insight to the dynamics of self-organized aggregation are in need.

In this thesis, we developed a non-spatial probabilistic *geometric model for self-organized aggregation*. The model consists of four formulas for calculating probabilities of aggregation events: *creation*, *growing*, *shrinking* and *dissipation* of an aggregate. While *creation* probability is derived using *mean free path* derivation in kinetic theory of ideal gases [3], *growing* and *shrinking* probabilities are derived geometrically with the help of circle packing theory.

In order to verify our model, we proposed an *aggregation behavior* which is realistic but carefully designed in order to match model predictions with simulation results. The behavior is implemented on Stage multi-robot simulator [63].

Unlike previous aggregation models, a *detailed verification* is performed for our aggregation model. Each formula being proposed is verified for varying parameters related to the corresponding formula using Stage simulation experiments. For example, creation probability formula is verified by varying *number*, *speed* and *sensing range* of searching robots.

In the next three sections, we briefly review existing *self-organized aggregation* and *non-spatial geometric modeling* studies and describe our *contributions* in more details subsequently.

1.1 Self-organized Aggregation

In order to develop high performant scalable aggregation, two approaches are used in existing studies. In the first approach [60] [55] [8], an omnidirectional speaker which continuously produces a tone that can be perceived from a certain distance and a couple of directional microphones to determine the loudest sound direction are attached to robots. The main idea is to use cumulative nature of the loudness of sound to guide the searching robots toward areas with larger aggregates.

Trianni et al. [60] used genetic algorithms to evolve aggregation behaviors for a group of simulated robots. The robots are equipped with three microphones and a speaker to communicate with each other, infrared sensors to detect other robots or environment boundaries and

a gripper which allow them to grab each other and move together.

The controller of the robots is a simple perceptron which connects the state of sensors to motor and gripper control parameters. The weights of the perceptron are evolved using a fitness function which computes average distance of the robots from their group's center of mass.

A *static* and a *dynamic* aggregation behavior are evolved. While the aggregates are immobile in the static behavior, the aggregates in the dynamic one move due to interactions of the waiting robots who grabbed each other inside the aggregate. The authors show that dynamic behavior is more scalable, since moving aggregates find each other in time and form larger aggregates. However the decline of the fitness of the dynamic behavior (See Figure 5 on page 5) when the number of robots are increased shows that the dynamic solution is not fully scalable.

Later, Bahçeci and Şahin [8] use a similar approach to develop perceptron based aggregation behavior for the same simulated robot model without a gripper. The perceptron whose weights are to be evolved has twelve input neurons and three output neurons. While the input neurons encode states of the infrared sensors and three directional microphones, the output neurons are used to control the wheels and the speaker. The fitness function is defined as the ratio of the size of the largest aggregate to the total number of robots. The evolved behaviors are shown to be unscalable when the number of robots is increased.

Soysal and Şahin [55] proposed an aggregation behavior for the same simulated robot used by Bahçeci and Şahin [8]. There are four behaviors in the controller: *approach*, *repel*, *wait* and *obstacle avoidance*. The former three behaviors are combined using a three-state finite state automata. The robots start in *approach* state which moves the robots toward to the loudest sound direction. When approaching robots detect other robots with their infrared sensors, they switch to the *wait* state. The waiting robots stay immobile until they probabilistically switch to *repel* state with a predefined *leave probability*. The *repel* behavior drives the robot in the opposite direction of the loudest sound. A repelling robot can switch to approach state with predefined *return probability*. Obstacle avoidance behavior is used as an emergency and takes control when a robot is too close to other robots or the environment boundaries.

Soysal and Şahin proposed four different strategies by setting different values to *leave* and

return probabilities. Two of the strategies are similar to the ones found in [60]. While static aggregates are formed in the first strategy, the second strategy generated dynamic aggregates that can move in the environment and find other aggregates. As in [60], the second strategy is shown to be more successful.

Although, successfull results are reported in all of the above studies, the solutions are only tested up to 40 robots. Due to limitations for the ranges for the speaker and microphone, it is expected to obtain more than one aggregates which can not "hear" each other in large environments. Although the strategy that generate dynamic aggregates [60] [55] can potentially solve this problem, the scalability of this strategy is still questionable.

The problem of joining multiple aggregates into one aggregate deserves further attention. If we consider each searching robot as a separate moving aggregate, the problem becomes the aggregation problem itself: How can we join multiple aggregates together?

The second approach is one possible answer to the above question. In this approach, robots are assumed to have the capability to estimate the size of an aggregate they detect. The robots use the aggregate size estimation in order to keep or grow larger aggregates while dissipating smaller ones. This is achieved by setting leaving probability *inversely proportional* with the aggregate size.

The origin of this approach is the studies performed by biologists working on aggregation of ants [20] and cockroaches [6] [38] [39]. These studies modeled the aggregation behavior of ants and cockroaches which shows that the leaving probability of ants/cockroaches is inversely proportional with the aggregate size.

Jeanson et al. [38] developed a numerical model of *individual* cockroach movements in a bounded circular arena. Due to tendency of cockroaches to aggregate close to the walls, the cockroach behavior is split into two sub-behaviors: *wall following* in the peripheral zone and *diffusive random walk* in the central zone. While the peripheral zone is defined as the area where cockroaches can establish antennal contact with the wall, the central zone is defined as the remaining area. The correlated random walk behavior in the central zone is characterized with transport mean free path and the probability per unit time to stop and exit central and peripheral zones are calculated from the experimental data. There are two different waiting behavior of cockroaches: short and long waits. The difference between these two waiting

Table 1.1: The probabilities of stopping and resting times for different aggregate sizes 1.1. While P_{short} is the probability for stopped cockroaches to display a short stop, τ_{short} and τ_{long} are the duration of the short and long stops respectively.

Size of aggregate	P_{short}	$\tau_{short}(s)$	$\tau_{long}(s)$
$N = 1$	0.93	5.87	700
$N = 2$	0.66	16	1248
$N = 3$	0.34	18.5	1062
$N = 4$	0.24	34.1	1719

behaviors is the duration. The probabilities and average durations of these waits are estimated from the experiments. Since angle of entrance to the central zone affects the time spent in the central zone, the frequency distribution of angles to the central zone is also calculated from cockroach experiments. Finally, the predictions of the numerical model which uses above quantities are compared against the results of the cockroach experiments and it is shown that cockroach behavior can be replicated using these quantities.

Later Jeanson et al. [39] extended the numerical model described above by making probabilities of stopping and resting times dependent on the number of neighbors in the aggregates. The hypothesis is that *the probabilities of stopping and resting times are higher when the number of cockroaches in the aggregates are greater*. This hyphothesis is verified by calculating the probabilities of stopping and resting times for different aggregate sizes in cockroach experiments. Table 1.1 summarizes calculated values in this study. The results of the numerical model and cockroach experiments are also compared in terms of size of the largest aggregate over experiment duration.

Ame et al. [6] proposed that the probability of leaving an aggregate of size x as (equation 1 in [6])

$$p = \frac{\theta}{k + x^n},$$

where k, n and θ are constants ($\theta = 0.06$, $k = 6$ and $n = 2$) estimated from experiments with cockroaches. In these experiments, two shelters are used as the aggregation sites for the cockroaches. The experiments start with equal number of cockroaches in each shelter and

Table 1.2: The leaving and joining probabilities used as a function of the number of neighbors (n) within sensing range during one time step k [17].

n	P_{join}	P_{leave}
0	0.03	n.a
1	0.42	1/49
2	0.5	1/424
3	0.51	1/700
4+	0.51	1/1306

each experiment ends when majority of cockroaches select one of the shelters. Although the main purpose of this study is to analyse the effects of two shelters on cockroach behavior, one of the figures they produced using the corresponding Monte Carlo simulation of this experiment shows that time needed to aggregate majority of the population on one site increases exponentially when the population size is increased. Although the study does not discuss the scalability of this approach as in other studies, this figure gives the most powerful evidence showing the scalability problem of this approach. Later, the studies performed by Jeanson et al. and Ame et al. are also implemented on real robots by Garnier and his colleagues [26], [27], [25] [24] in order to compare the previous experimental results with the ones obtained from real robot experiments.

Corell and Martinoli [17] also use this approach while proposing a non-spatial model for aggregation. The model is tracking the average number of robots in a certain aggregate size similar to the model developed in [56]. Table 1.2 shows the leaving and joining probabilities used as a function of the number of neighbors within sensing range. Simulation results and model predictions of the average number of robots in a certain aggregate size are shown to match for 12 robots.

1.2 Non-spatial Geometric Modeling

The main approach that have been used to model aggregation and some other problems is mainly performed by Martinoli and his colleagues and can be called “non-spatial modeling”. [46] [5] [35]. In non-spatial modeling approach, the trajectories of the robots or the relative

positions of the aggregates with respect to other aggregates or to the walls of the environment are ignored and robots are assumed to occupy random positions *at each time step*.

Although assumptions for non-spatiality above might look realistic at first look, it gives robots the maximumum motion flexibility and consider robots as jumping objects in the environment. Unfortunately, due to lack of clarfication of assumptions, low level implementation details or verifications of low level probabilities in existing studies, it is *not* easy to reach the above conclusion and this conclusion can only be supported by presenting quotes from the existing studies. The following quote is taken from Martinoli et al. [46].

> "In nonspatial models, robots' positions on the arena are assigned randomly at each new *iteration* (or *time step*). At each iteration, the probability that a robot in the search mode will encounter a wall, a stick, or another robot is determined by their corresponding detection area divided by the whole arena area A_a. For instance, the probability of finding a stick can be computed as $p_s = A_s/A_a$, A_s being the detection area of a stick." (page 8)

Another quote from Agassounon's Ph.D. thesis [4] (advised by Martinoli) is as follows.

> "Nonspatial models do not take into account the trajectories of the individuals, instead, agents are assumed to occupy *independent positions (randomly assigned) during consecutive time steps*. This is equivalent to assuming that the agents hop around randomly with equal probability of occupying any position during any time step." (page 14)

In an older study [47], the same limitation is stated in a different way by Martinoli et al.

> "There are three fundamental approximations in our simulation:
>
> - *Robots are not moving in the environment* (i.e. no trajectories are calculated): the simulation calculates the global probability of finding a cluster or another teammate based on their detection area and the arena surface.
> - Boundary effects are only taken into account in a limited way: ..." (page 7)

In non-spatial modeling approach, three different methods are used to calculate the model probabilities: *geometric probability*, *sweeping probability* and *encountering probability*. Although the same name (encountering probability) is used for second and third methods in existing studies, we chose to use different names for these methods since using the same name for two different things creates confusion.

Geometric probability is the simplest method among three. In this method, growing probability of an aggregate is calculated by dividing the area of the aggregate to the area of the environment. This method is appropriate when searching robots hop around randomly on the environment. However, the growing probabilities does not include any terms for random walk behavior or the speed of the robots and sensing range of the robots which makes the method less realistic compared to other two methods. It is generally used when the aim is to introduce new modeling approaches or to analyse a proposed algorithm.

Soysal and Şahin proposed a macroscopic model for aggregation in [56]. In this model, authors use number theory to track the changes in average aggregate distributions over time. As in previous aggregation studies, the events that can change the aggrgate distributions are growing and shrinking of aggregates. In calculating growing probability of aggregates, authors use the geometric probability.

Sweeping probability is a more advanced method for calculating growing probabilities. In this method, the area swept by a searching robot, instead of the aggregate area in geometric probability, is divided by the area of the environment. A typical equation for growing probability of an aggregate used by Corell and Martinoli [17] [18] is shown below.

$$P_g = \frac{1}{A} v_r w_d t, \tag{1.1}$$

where A is the area of the arena, v_r is the average speed of a searching robot, w_d is a robot's detection width (the width it sweeps with its sensors while moving) and t is the time discritization. $v_r w_d t$ term in this equation corresponds to the area swept by a searching robot in one time step and P_{grow} is the growing probability in one step.

Using this growing probability in addition to a simple shrinking probability, Corell and Martinoli [17] develop a non-spatial macroscopic model for aggregation. The model is iterated by a difference equation which tracks the average number of robots in all cluster sizes.

Lerman and Galstyan [44] propose a model for tracking the number of pucks collected to a home region in a bounded environment using differantial equations. Although, the parameters needed to iterate the model such as the rate at which a robot encounters a puck and another robot are obtained from simulation experiments, the authors made the following comment regarding the calculation of these parameters which shows their reliance on sweeping proba-

bility method.

> "In principle, these parameters can be computed ab initio by taking into account the details of the robots dimensions and sensing capabilities in the following way: as a robot travels through the arena, it sweeps out some area during time interval dt and will detect objects that fall in that area. This detection area is $vw_i dt$, where v is robot's speed, and w_i is robot's detection width for object of type i . This number is the sum of the sizes of the robot and the object it is trying to detect, and the detection distance associated with the sensing hardware it is using to detect that object (e.g. sonar, camera resolution, etc.). If the arena radius is R with N_i objects of type i distributed uniformly around it, a robot will detect these objects at a rate $\alpha_i = vw_i N_i/\pi R^2$. This idealization is useful for roughly estimating model parameters, but because it omits all the details of the experiment (such as sensor errors and failures), it does not get them right. A better way is to estimate them by fitting the model to experimental data ..." (page 130)

Encountering probability is another more advanced way of calculating growing probabilities in non-spatial modeling. In this method, encountering probability is defined as the multiplication of geometric probability and a constant which is said to convert model iterations into simulation time. Following quote from [35] describes how this constant is calculated.

> "**Time-iterations transformation** Similarly to the methodology proposed in (Martinoli et al., 1999b), the correspondence between iterations in the probabilistic simulation and time in the real experiment is obtained by linking the number of iterations and the time needed to systematically cover the whole arena in the probabilistic simulation and in the real experiment, respectively. In the probabilistic simulation $N = A_A/A_S$ iterations would be needed for the systematic search for sticks (i.e. without passing twice for the same position) while this would take a duration of $T = A_A/(V_R.W_R)$ in the real experiment, where V_R and W_R are the robot's mean forward speed and detection width. The duration of one iteration therefore corresponds to $A_S(V_R.W_R)$. Using this correspondence factor, it is possible to translate the different durations appearing in the real experiment, such asa the gripping time, the duration of obstacle avoidance, and the duration of the detection procedure into number of iteration." (page 11)

One pioneering application of non-spatial modeling to swarm robotics is performed by Martinoli et al. [47] for the clustering problem. In clustering problem, robots collect objects scattered around a bounded environment. In [47], Martinoli et al. present a clustering implementation on a simulator and real robots. The change of the cluster sizes in these implementations is compared with the change reported by a non-spatial microscopic model proposed in this study. The model is run by iterating states of each robots using state transitions probabilities calculated using *encountering probability* method.

Table 1.3: The leaving and joining probabilities used as a function of the number of neighbors (n) within sensing range during one time step [17].

Object	Measured enc. rate	Computed enc. rate
Half-wall	$0.0125s^{-1}$	$0.0148s^{-1}$
Full-Wall	$0.0423s^{-1}$	$0.042s^{-1}$
Blade	$0.0059s^{-1}$	$0.0085s^{-1}$
Robot	$0.0021s^{-1}$	$0.0017s^{-1}$

Later, Agassounon et al. [5] used same probabilities to develop a non-spatial *macroscopic* model. The model is used to track the average number of robots inside different states (e.g. obstacle avoidance, search) of the PFSA based controller with difference equations.

Ijspeert et al. [35] applied non-spatial modeling to stick pulling problem which requires to pull sticks randomly placed in a bounded environment by two robots collaboratively. The probabilistic finite state automata (PFSA) of the robot controller is tranformed into a probabilistic model by assigning certain probabilities and timeouts for each state of the PFSA. The probabilities are estimated using *encountering probability* and the timeouts are calculated from simulation experiments. In both of the studies, it is shown that the non-spatial model predicts the collaboration dynamics.

The *encountering probability* is also used by Corell and Martinoli to develop non-spatial models (both microscopic and macroscopic) for the tribune inspection problem [15] [16]. In tribune inspection problem, the aim is to inspect jet turbine engines by a swarm of robots collectively in order to minimize failures of the engines. There are two interesting points of these studies. First, unlike other studies of the authors, the verification results of the encountering probability calculated with the swept area method is presented. The results show that there are around %30. error compared to experimental results as shown at Table 1.3.

Second, contrary to other non-spatial modeling studies, the results of the non-spatial models do not match to experimental results acceptably. Authors claim that discrepancies between models and experimental results are mainly due to spatial effects of blade shape and the configuration of the blades.

In overall, above discussion indicates that *geometric*, *sweeping* and *encountering* probabilities

are approximations to actual probabilities. Lack of details of impementations prevents us to comment or analyze further about these studies.

1.3 Contributions

We have developed a probabilistic geometric model for self-organized aggregation in swarm robotic systems. The model consists of probabilities derived for four main aggregation events: *creation*, *growing*, *shrinking* and *dissipiation* of an aggregate.

Our model is significant in several aspects. First, this model is the first aggregation model which includes an equation for aggregate creation probability. In [56] and [17], creation event and creation probability are ignored. Aggregates are being created when one robot was waiting and a searching robot finds this waiting robot. In other words, aggregate creation is simplified to growing of an aggregate-1. However, aggregate creation event is a fundamental/natural component of aggregation behavior, preventing creation event requires a non-trivial implementation and delays solution of aggregation problem undesirably. These issues and implementation details of prevention of aggregate creation are not discussed in these studies.

Second, the equation corresponding to shrinking probability is more realistic compared to equations in previous studies [56] and [17] [39] [6] [26] [27] [25] [24] [17]. When an aggregate is created and grows multiple times, latest searching robots that joins to the aggregate surrounds previously joined robots and prevent them from leaving the aggregate. In other words, only robots that are at the periphery of an aggregate can leave that aggregate. In previous aggregation models, it is assumed that any robots that belog to an aggregate can leave that aggregate. This wrong assumption becomes a serious obstacle for analyzing aggregation dynamics when aggregates get larger.

Third, we proposed a behavior for self-organized aggregation in a bounded arena. The behavior consists of four sub-behaviors: *searching*, *waiting*, *leaving* and *change direction*. Sub-behaviors are connected to each other in order to perform actual aggregation behavior. Transitions between sub-behaviors are triggered by either robot detection event o a timeout. In this aggregation behavior, *waiting* sub-behavior is designed carefully so that aggregates are circular and model assumptions regarding growing and shrinking probabilities hold in the simulator. Additionally, each sub-behavior has some internal parameters to perform its task

correctly or efficiently. Default values for these parameters are also selected carefully so that the model assumptions and the overall aggregation behavior match. As the implementation platform, Stage multiple robot simulator [63] is used. Since, Stage is a widely used simulator in robotics community, a general purpose sensor and kinematic models that are available in Stage are used, our implementation can be easily ported to real robot platforms.

Fourth, we performed a detailed verification of our proposed aggregation model. Detailed verification of aggregation models is lacking in the existing literature. In existing aggregation modeling studies, proposed aggregation models are only used to show that simulation and model results match for a proposed aggregation solution strategy using some basic high level metrics and for low number of robots. However, using low number of robots with basic metrics can hide the limitations of these probabilistic models. In our approach, we use seperate Stage simulation experiments for verifying each equation we proposed. In these experiments, we enabled only one aggregation event (e.g. creation event), calculated the probability of this event and compared it with the model prediction.

This detailed verification approach is continued by performing experiments in which only two aggregation events (e.g. growing and shrinking) are enabled. For these experiments, the performance for a certain performance metric (e.g. the number of searching robots or aggregates) in the steady state is compared for the model and the simulation experiment. The model prediction is calculated in three different ways: microscopic model, simplified microscopic model and steady state analysis. We believe that this detailed verification approach is valuable for our research area and our results can be used as a reference for future aggregation modeling studies. Additionally, these 2-events experiments (experiments which have only two aggregation events) are also good examples of how our model can be used to predict the steady state of aggregation easily by just solving an equation numerically without running time consuming simulations (steady state analysis).

In the next chapter, we present a literature survey for swarm robotics. Then, in chapter three our experimental framework, which consists of a multiple robot simulator, a probabilistic aggregation controller implemented on this simulator and aggregation performance metrics, is described. In the fourth chapter, our probabilistic geometric model for aggregation which consists of four formulas to calculate probabilities corresponding to aggregation events, is desbribed. In chapter five, we verified proposed aggregation model with simulation experi-

ments performed in our multi robot simulator. In each experiment, the formula used to calculate probability for one of the aggregation events is verified by enabling only that aggregation event in the simulator and calculating the probability from the number of occurences of that event in the simulator. In chapter six, the verification is continued by performing experiments in which only two aggregation events are enabled. In these experiments, steady state performance and time are predicted using microscopic models, simplified microscopic models and steady state analysis. The thesis ends with conclusions.

CHAPTER 2

Literature Survey

In this chapter, we categorize and review swarm robotics studies. Most of the contents of this chapter are taken from our previous survey [11] and are slightly modified. In order to categorize swarm robotics studies, previous literature surveys related to swarm robotics are investigated and summarized below.

Dudek et al. [21] classified the swarm robotics literature in terms of swarm size, communication range, communication topology, communication bandwidth, swarm reconfigurability and swarm unit processing ability. They prepared a taxonomy instead of a survey on swarm robotics and describe a limited number of sample publications inside this taxonomy.

Cao et al. [14] presented the survey of cooperative robotics in a hierarchical way. They evaluated studies along five main axes: group architecture, resource conflicts, origins of cooperation, learning and geometric problems. Group architecture is further divided into centralization/decentralization, differentiation (denotes the homogeneous or heterogeneous robot groups), communication structure and modeling of other agents categories. The dimension of modeling of other agents contains studies which models the intentions, beliefs, actions, capabilities, and states of other agents to obtain more effective cooperation between robots.

Iocchi et al. [37] presented a taxonomy of multi-robot systems and address some multi-robot system studies in their taxonomy. They presented their taxonomy hierarchically using levels. First level is *cooperation level* which is divided into *aware* and *unaware* categories as the lower knowledge level. Aware category is divided into three more categories namely *strongly-coordinated*, *weakly-coordinated* and *not-coordinated* as the coordination level. Strongly-coordinated category is divided into *strongly-centralized*, *weakly-centralized* and *distributed*

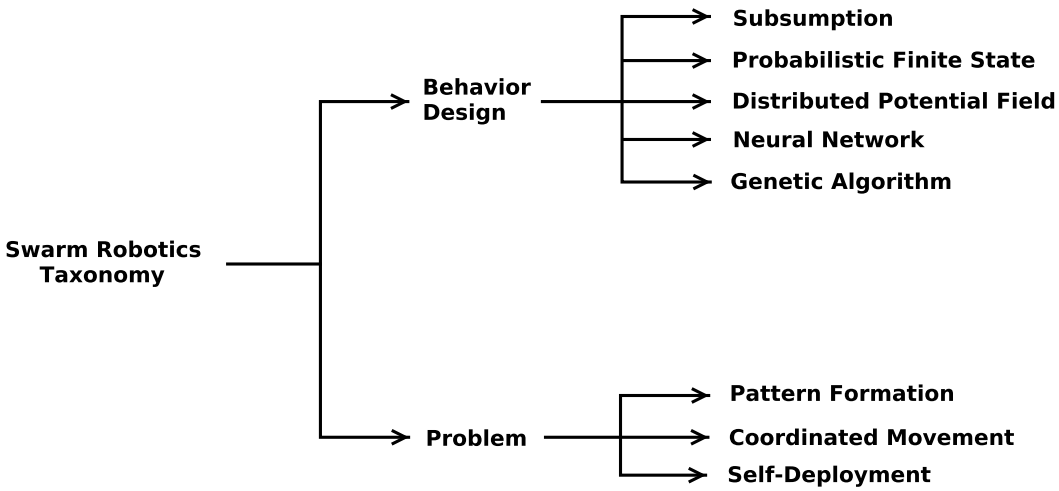


Figure 2.1: Taxonomy of swarm robotics literature. The taxonomy is divided into two main categories: *behavior design* and *problem*. Six behavior design methods and eight swarm robotic problems are identified under these categories.

categories as the organization level. Moreover, the study described a set of application domains of multi-robot systems.

Gazi and Fidan [28] presented a review of multi-agent systems from the system dynamics and control perspective. The authors focused on agent dynamics models. Then, they presented some swarm coordination and control problems and different approaches to modeling, coordination and control of swarms.

In this survey, existing studies are divided into two main categories: *behavior design* and *problem*. In *behavior design* category, existing studies are classified according to behavior design method and summarized. Five behavior design methods are identified: subsumption, probabilistic finite state, distributed potential field, neural network and genetic algorithm.

In *problem* category, existing studies are classified according to problems being worked on. Three swarm robotic problems are reviewed: pattern formation, coordinated movement and self-deployment. Figure 2.1 shows our taxonomy for swarm robotics studies.

2.1 Behavior Design

According to the behavior design category, existing swarm robotics studies can be categorized into four subcategories: subsumption, probabilistic finite state, distributed potential field, neu-

ral network and genetic algorithm methods. Studies that fit into these categories are described in the following subsections respectively.

Payton et al. [53] [54] proposed an approach called *pheromone robotics* and use this approach to solve map generation and shortest path solving problems using a group of robots. Three main concepts are defined in this approach: *virtual pheromone*, *world embedded computation* and *world embedded display*.

The *virtual pheromones* are 10-bit messages which mimic the biological pheromones used by insects. Each virtual peromone consists of message type, hop-count and data fields. The actual message transmitted to other robots is kept in the data filed and the opponent interprets this message according to the message type. Hop-count is used to simulate the decay of pheromone message while being transmitted between robots. The virtual pheromones are signaled between robots with a turret attached at the top of the robots that has eight radially-oriented, directional infrared receivers and transmitters.

World embedded computation corresponds the capability of robots to process computationally expensive operations using the personal digital assistant attached at the top of them. Using this feature, robots can collect data and solve problems like generating the map of a field or solving the shortest path problem.

An external observer can also be informed about the data collected by the robots with the help of a video camera mounted on the observer's head which receives and displays coded infrared signals from each robot. This feature is named as *world embedded display*.

2.1.1 Subsumption

Subsumption architecture [12] is a well-known behavior design method in behavior-based robotics [7]. The method allows efficient coordination of behaviors by using a simple inhibition mechanism between the behaviors and incremental building of robot controllers by considering each behavior as a separate module which can inhibit other behaviors.

In [49], Mataric presented design of some behaviors from simple to complex using subsumption method. The behaviors are *collision avoidance*, *following* (inverse of collision avoidance), *dispersion* (used in order to balance goal-directed behavior against interference), *ag-*

gregation, *homing* and *flocking*. Although the author claimed that the behaviors are tested on a herd of physical mobile robots, the implementations details or evaluation of the success of behaviors were lacking in the study.

Nouyan and Dorigo [51] implemented a *chain formation* behavior in a sensor-based simulator. The robots had two states: *explorer* and *chain member*. In explorer state, the robots search for other chain members or the nest. Whenever a robot finds the nest or other chain members, it switches to the *chain member* state and tries to keep permanent visual contact with the nest or chain other member using an omni-directional camera. With this behavior, robot moves toward the end of the chain and stays there when the explorer timeout is reached. Robots in this study are distinguishing chain members and the nest based on the color of ring, consisting of light emitting dioides around their body.

Systematic experiments are performed in order to understand effect of modifying number of robots and the explorer timeout on the chain formation speed and the shape of chains. It is observed that short explorer timeout values lead to fast formation of many chains and long explorer timeout values lead to slow formation of fewer chains.

Nouyan extended this work with additional experiments in his master thesis [52]. In this thesis, the same behavior is used to establish a path towards a goal location from the nest too.

2.1.2 Probabilistic Finite State

In probabilistic finite state method, robot behavior is represented as a group of connected states. Each state corresponds to a sub-behavior and transitions between states are controlled with external events or user defined probabilities.

In [55], Soysal and Şahin used probabilistic finite state method to develop an aggregation behavior. Four sub-behaviors are used: *obstacle avoidance*, *approach*, *repel* and *wait*. Robots start in *approach* state and switch to the *wait* state when they sense another robot. The switches between *repel* and *approach* states, and *wait* and *repel* states are determined by predefined *returning* and *leaving* probabilities respectively. Four different aggregation behaviors are obtained by using different combinations of *returning* and *leaving* probabilities. Experimental results show that the behavior whose returning and leaving probabilities are 1 produces largest aggregates. However, as stated in Section 1.1, it is shown that this behavior

is *not* scalable.

Labella et al. developed *prey retrieval* behavior using the probabilistic finite state method [41] [43] [42]. The behavior consists of five sub-behaviors: *search*, *retrieve*, *deposit*, *rest* and *give up*. All transitions between states are triggered by external events except the transition between *rest* and *search* states which is triggered probabilistically. The probability of triggering rest-search transition is updated depending on the number of consecutive foraging successes or failures. The behavior is implemented on Lego Mindstorm robots. Experimental results show that two different roles (*foragers* and *loafers*), which has low and high rest-search transition probabilities respectively, has emerged due to minor mechanical differences exist in robots.

2.1.3 Distributed Potential Field

The distributed potential field method represents interactions of a robot with other objects/robots in the environment as virtual force vectors. These vectors can be attractive (e.g. moving towards a goal) or repulsive (e.g. moving away from obstacles). The vectorial summation of these forces are computed and mapped to actions of the robot. Unlike the potential field method used by Khatib [40] and Arkin [7], virtual force vectors are calculated by each robot locally based on the robot's sensing data.

Spears et al. [57] proposed a distributed potential method, called *artificial physics*, for the control of large number of robots. In artificial physics, the virtual force of a neighbor robot or object is calculated using its distance and bearing. Authors used the method for forming hexagonal lattices in two and three dimensions using simulators. Additionally, obstacle avoidance, surveillance and perimeter defense behaviors are implemented on real robots. Detailed analysis and robustness tests for these behaviors are performed successfully.

Balch and Hybinette [10] proposed a behavior for achieving various formations while navigating to a goal location. The behavior consists of several motor schemas [7]: avoid-static-obstacles, avoid-robots, noise, move-to-unit-center and maintain-formation. Each motor schema generates a virtual force vector from local sensing data available to a robot and the resultant force vector is mapped to motor commands.

Avoid-static-obstacles and avoid-robots motor schemas avoid obstacles and robots respec-

tively. Noise motor schema generates a random noise vector to escape from local minimas in the system.

Move-to-unit-center motor schema generates a force toward the center of the group in order to keep robots together. Maintain-formation is used to generate formations with different shapes using virtual attachement sites located around robots. Depending on the nuber of attachement sites and contact angles with these attachement sites, formations with various shapes are obtained with this motor schema. The proposed behavior is shown to be scalable using simulation experiments.

2.1.4 Neural Network

Neural networks [32] [31] are powerful learning mechanisms inspired from nervous systems. There are two types of swarm robotics studies performed using neural networks. The first type uses genetic algorithms to evolve the weights of a neural network to obtain a desired behavior with a fitness function appropriate for the problem. This type of studies [8] [60] [62] [59] will be discussed under Section 2.1.5.

The second type of studies with neural networks considers the neural networks as a *generalization mechanism* and do *not* use its learning capabilities. The remaining part of this section summarizes this type of studies.

Grob et al. [29] used a perceptron neural network for obtaining self-assembly behavior. The perceptron connects sensory inputs to the output neurons used to drive the robot. The task of robots is to locate, approach and connect with an object that acts as a seed or connect to other robots already connected to the seed. The seed and the robots connected to the seed are discriminated based on the color of the ring around them. The self-assembly behavior is tested on flat and rough terrains using real robots and it is shown thet robots achieve self-assembly in a scalable way.

Martinoli and Mondada [48] used two different neural networks with hand-coded weights to obtain *object clustering* and *stick pulling* behaviors. While sensory inputs are used to feed the neural network, the output of the output of the neural network is directly used to drive the robot. For object clustering behavior, authors reported that the performance of the behavior decreased when number of robots is increased. They have also reported that the stick pulling

behavior is sucessfull without presenting any quantitative results.

2.1.5 Genetic Algorithm

Genetic algorithms are one of the mostly used offline optimization algorithms in robotics because of their ability to escape from local optimum and successful results reported by studies in various research fields.

In swarm robotics, genetic algorithms are generally used to evolve weights of the neural networks to obtain a desired behavior. This approach is very powerful since it combines the generalization ability of the neural networks with the ability to escape from local optimums of genetic algorithms.

Bahçeci and Şahin [8] used a neural network, with twelve input and three output neurons, to obtain aggregation behavior. Genetic algorithm is used to generate random weights for the neural network. Each chromosome in the genetic algorithm corresponds to a candidate aggregation behavior. The performance of a chromosome is calculated by running corresponding candidate behavior in a simulation and calculating the fitness function at the end of the simulation run. The fitness function calculates the ratio of the number of robots forming the largest aggregate to the total number of robots. Authors evaluated the performance and the scalability of evolved aggregation behaviors systematically.

Trianni et al. [60] used the same method in order to obtain aggregation behavior. However, wights of the perceptron are evolved using a fitness function which computes average distance of robots from their group's center of mass. Two types of aggregation behavior is obtained in this study: *static* and *dynamic*. Experimental results obtained from simulation runs showed that the dynamic behavior creates larger aggregates in a more scalable way by joining small aggregates by time.

Trianni et al. [62] [59] applied genetic algorithm based behavior design method on *hole avoidance* problem too. In hole avoidance problem, robots have to perform coordinated movement in an environment which has holes too large to be traversed alone and robots have to connect to each other in order to pass the holes. While performing the hole avoidance behavior, if a robot detects hole using its ground sensors, it should move toward the opposite direction of the hole. The force generated with this movement is detected by the traction sensors of

other robots connecting to this robot and the challenge becomes learning to react to these forces in an appropriate way so that the group can still move together while avoiding the hole independent from relative position of robots and size of the hole or group.

Trianni et.al used a genetic algorithm to evolve weights of a simple perceptron which connects sensory inputs to the motor outputs of the robot as studies discussed previously. The fitness of perceptrons are calculated using simulation experiments. The fitness function being used has three components. While the first component is measuring how well the coordinated movement is performed, second one is measuring how much the arena is explored and the third one is using fast reaction of the group for detecting a hole. Obtained behaviors are tested on environments with varying hole sizes and it is shown that behaviors are successful and scalable.

2.2 Problems

In this section, studies performed on *pattern formation*, *coordinated movement* and *self-deployment* problems are reviewed from a problem perspective in following subsections respectively.

2.2.1 Pattern Formation

Pattern formation is the emergence of global patterns from interactions of individuals.The pattern formation is important in swarm robotics since any kind of coordinated behavior performed by a group of robots forms a pattern when viewed globally. And all problems investigated in swarm robotics can be seen as creating these patterns with local interactions of individuals.

Bahçeci and Soysal [9] reviewed pattern formation and adaptation studies in multi-robot systems. Pattern formation studies are divided into two categories: *centralized* and *decentralized* pattern formation. Authors claimed that having a central unit assumption in centralized pattern formation makes the approach more costly, less robust to failures and less scalable compared to the decentralized approach. Adaptation studies are divided into two categories too: *individual level* and *group level* adaptation. While individuals adapt themselves to the

problem in individual level adaptation studies, group level adaptation emerges in group level adaptation studies.

Fredslund and Mataric proposed a general method for developing robot formation behaviors using local sensing and minimal communication in [22] and [23]. The algorithm works for a particular class of formations specifically the ones that can be folded from an open bicycle chain, keeping either the middle or the end of the chain in front.

The algorithm requires that each robot has a unique identification number[1] and a *friend sensor* which is used to track the friend robot. The friend sensor is defined as a sensor that can measure the relative direction and distance of the friend robot and the friend robot is defined as the robot which has either the lowest or the highest identification number depending on the target formation.

There are two disadvantages of the method. First, robots should know the number of robots that will be used in the experiment. Second, one robot should be selected as the conductor robot which specifies the movement direction for the group. However, the method has two valuable features. First, it can be applied to different formations. Second, robots in the formation can move with the help of the conductor robot without breaking the formation.

In [61], Trianni et al. proposed a method for solving the pattern formation problem based on a concept called *context*. The concept is an abstraction for sensor readings of a robot. In this method, at each step, robots identify their contexts from its sensor readings and decide their actions from a set of possible actions available for their contexts probabilistically. The probabilities of actions are predefined by authors.

The method is shown to work with *aggregation* and *chain formation* problems using a simple simulator which moves robots on a large hexagonal grid. However, the attempt to predict simulation results with a mathematical model is failed. Authors claimed that possible reasons for this failure are the lack of spatial information in the mathematical model, carrying out the simulation in discrete time and the lack of interaction dynamics in the model.

[1] Identification numbers are only used for numerical comparison purpose. For example, the robot turns only to the robot with the lowest identification number when more than one robot exist around it.

2.2.2 Coordinated Movement

Coordinated movement problem requires a group of individuals to move with a certain pattern and observed in many different animal species [13]. The movement of flocks of birds or schools of fish are examples of coordinated movement behavior in nature.

Hayes and Dormiani-Tabatabaei proposed a leaderless distributed coordinated movement behavior for a group of robots in in [30]. Robots use two sub-behaviors to perform coordinated movement behavior:

They used two behaviors to obtain flocking behavior: *collision avoidance* and *flock centering*. Collision avoidance is activated when a robot detects an obstacle or another robot and mediates the robot to turn away from the obstacle or other robot. Flock centering sub-behavior is active when the collision avoidance is not active and it is used to move the robot towards the center of mass of the robot group using local relative distance and bearing data.

Internal parameters of the behaviors are found using an offline optimization method. After the optimization, the method is shown to work on real robots which use overhead cameras to obtain distance and bearing data.

2.2.3 Self-deployment

In the self-deployment problem, robots should deploy themselves to the environment in such a way that they cover the environment as much as possible. When robots are initialized randomly in the environment and they have limited sensing capabilities, the problem becomes non-trivial.

Howard et al. [33] proposed a solution to self-deployment problem which aims to maximize the area covered while simultaneously ensuring that each individual can be seen by at least one other individual. The algorithm of the solution is estimated to have a polynomial computation time complexity of order n^2 where n is the number of individuals.

The solution has four phases: *initialization*, *selection*, *assignment* and *execution*. In the initialization phase, one individual is deployed to a random location in the environment. In the selection phase, deployed individual sends what it sense to a base station and the base sta-

tion combines sensor data obtained from deployed individuals into a map. Then, this map is analyzed in order to find unexplored locations to deploy new individuals.

In the assignment and execution phases, selected locations are assigned to individuals and assigned individuals are sent to new locations respectively. The algorithm iterates through the selection, assignment and execution phases and terminates when all individuals are deployed.

Four different selection policies are tested using two performance metrics: total coverage of the area by the individuals and total deployment time. The best policies are found to be between 70% and 85% of the performance obtained for a greedy algorithm which has the environment map.

Howard et al. [34] also used a distributed potential field method [40] [7] in order to solve self-deployment problem. The individuals are treated as virtual particles subject to virtual forces. The virtual particles repels from each other and from obstacles. A virtual friction force is also added to the system to be able to reach static equilibrium. The performance of the method is tested with the same metrics used in [33] and found that the solutions shows promising results. However, performance comparisons with existing solutions are left as future work.

CHAPTER 3

Experimental Framework

In this chapter, we present the details of the experimental framework used in this study, namely a multiple robot simulator based on Stage, probabilistic aggregation behavior and performance metrics.

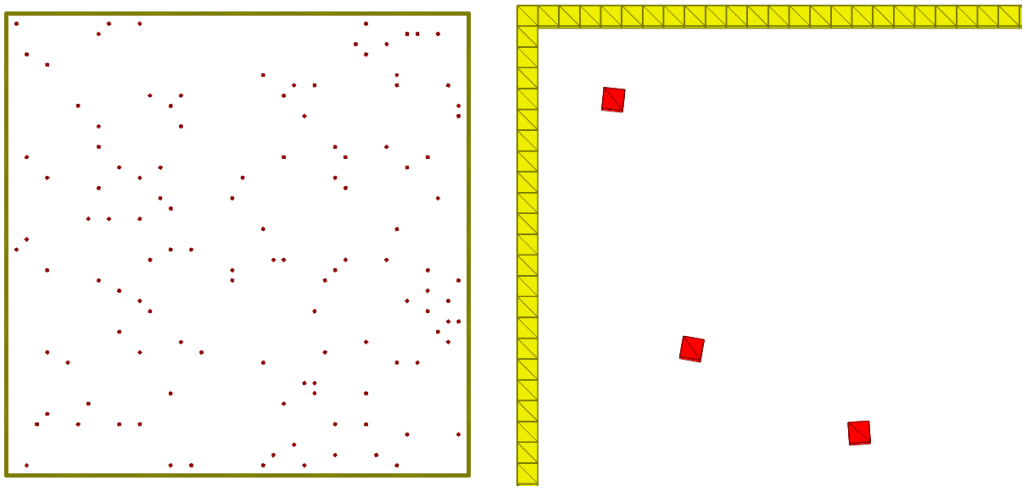
3.1 Stage-based Simulator

We used the Stage simulator to simulate a population of mobile robots in a two-dimensional bitmapped environment [63]. Figure 3.1-a shows screenshot of the Stage simulator with default environment settings (120 robots in a 180×180 arena) in which robots are randomly placed in the environment surrounded by walls.

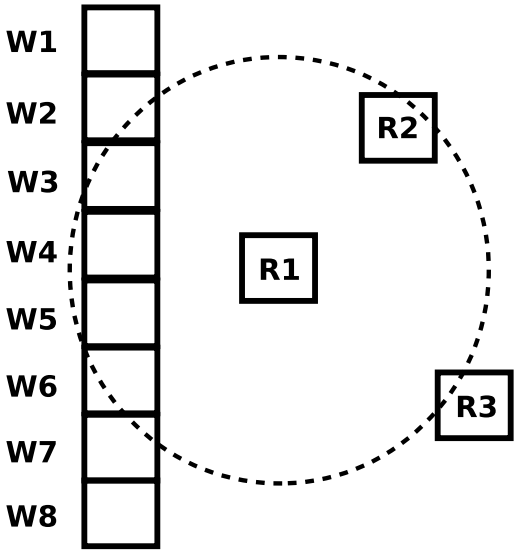
Robots are 1×1 m^2 square blocks which two wheels and have kinematic constraints of the Pioneer robot [50] model implemented in Stage. The motion of robots are controlled with forward (v) and angular (w) velocities calculated by the active behavior and passed to the Stage API.

Each robot has two fiducial sensors [1] located at the center of the robot, with 360° field of view, and a range of s. While one of them is used to detect wall blocks, the other one is used to detect other robots. Both sensors have detection zone around the robot as shown in Figure 3.1-c. All wall blocks or robots inside this zone are detected and the *distances* and *bearings* to detected robots or wall blocks are known.

The walls consisted of fixated 1×1 m^2 square blocks. The wall blocks and robots are shown more clearly in Figure 3.1-b which shows the zoomed out view of top left corner of the



(a) (b)



(c)

Figure 3.1: (a) Screenshot of the Stage simulator with 120 randomly placed robots in a 180 × 180 arena. (b) Zoomed out view of top left corner of the environment on the left. Red and yellow squares are robots and wall blocks subsequently. (c) Illustration for the detection zone of a robot. Three robots ($R1$, $R2$ and $R3$) and eight wall blocks ($W1$, $W2$, ... $W8$) are drawn. The dotted circle represent the detection zone of $R1$ robot. In order to sense a robot or wall block, the center of the square representing the robot or wall block should be inside the detection zone. In this case, only $R2$ robot and $W3$, $W4$, $W5$ and $W6$ wall blocks can be detected by $R1$ robot.

environment shown in Figure 3.1-a. The world file that we used for executing Stage simulator is shown in Appendix A.

3.2 Probabilistic Aggregation Behavior

All robots are controlled by the same aggregation behavior which has four states: *search*, *wait*, *leave* and *change direction*. These states and transitions between them are illustrated in Figure 3.2. [1]

The robots start in *search* state and look for other robots by moving straight in the environment. When a searching robot detects another robot, it switches to the *wait* state and becomes part of an aggregate.

Waiting robots (that is robots in the wait state) can wait inside an aggregate or leave the aggregate. They can leave in two different ways. First, if a waiting robot detects that there are no robots around, it switches to the *change direction state* immediately. Second, a waiting robot can switch to leave state probabilistically with probability p.

Robots in leave state, avoid other robots for a certain time period (leave timeout) or until they no longer detect any other robots. When one of these events occur, they switch back to *search* state.

In *change direction* state, the robots turn right for a random number of steps. The turning angle is random in $[0, 2\pi]$ radian range. When robots finish turning, they switch back to *search* state.

Behaviors are implemented using the bearing (b_i) and range (r_i) for each neighbor robot i and the bearing (b_j) and range (r_j) for each neighbor wall block j obtained. With the exception of the change direction behavior, all behaviors calculate a virtual force for each neighbor robot ($\mathbf{f_i}$) and/or wall block ($\mathbf{f_j}$), calculate resultant virtual force ($\mathbf{f}$) and map it to forward (v) and angular (w) velocities of the robot. The details of behavior implementations are described in following subsections.

[1] The control method is *not* within the scope of this thesis but is left as a future work.

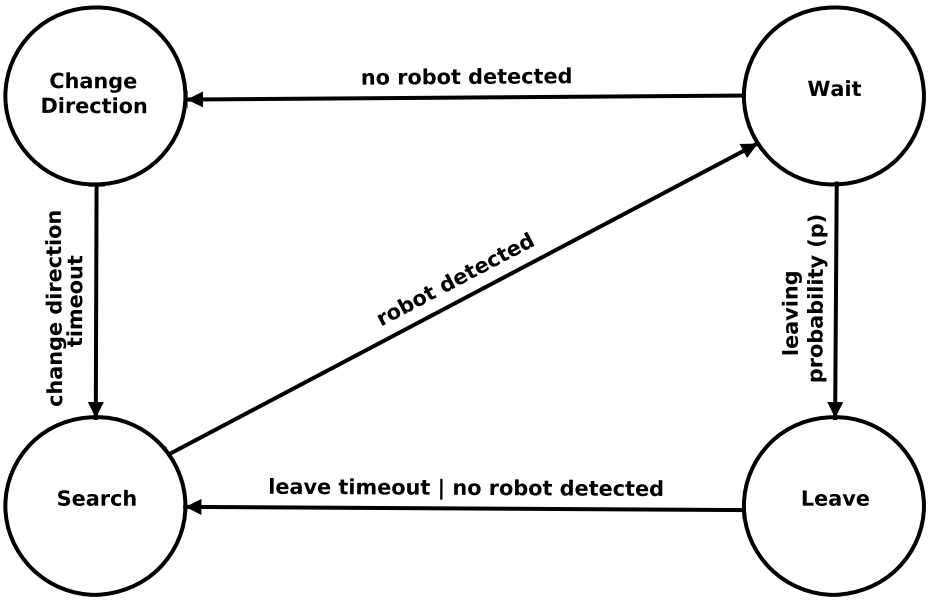


Figure 3.2: The probabilistic aggregation behavior has four possible states: *search*, *wait*, *leave* and *change direction*. Robots starts in search state. The transitions between these states are controlled with leaving probability (p), leave and change direction timeouts, "robot detection" and "no robot detection" events.

3.2.1 Change Direction Behavior

In the *change direction* behavior, robots turn right for a random amount of time. The duration of turning (t_{rand}) is determined randomly in $[t_{min\_cd}, t_{max\_cd}]$ range which roughly corresponds to a random turn in $[0, 2\pi]$ radian range. When leaving robots finish turning, they switch back to the *search* state. The forward and angular velocities of robots in this state are set as:

$$v = 0 \ , \ w = v_s \ .$$

3.2.2 Search Behavior

In the *search* behavior, robots move at the maximum forward searching speed (v_s) until they sense other robots or wall blocks. If they sense another robot, they switch to the wait state. If they sense wall blocks, they change their forward and angular velocities to avoid collisions. Trace of a searching robot moving alone in the default environment is shown in Figure 3.3.

The virtual force $\mathbf{f_j}$ for each wall block j is calculated as:

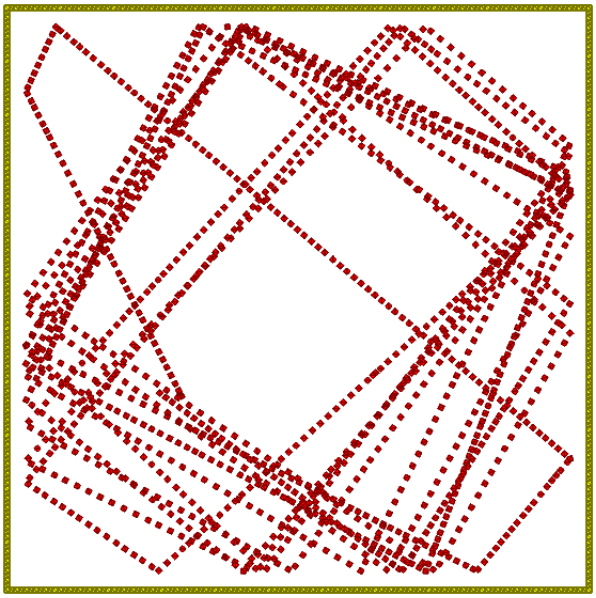


Figure 3.3: Trace of a searching robot moving alone in the default environment. Dotted lines represent the trace.

$$\mathbf{f_j} = \begin{bmatrix} r_j \cos(b_j) \\ r_j \sin(b_j) \end{bmatrix} . \tag{3.1}$$

The resultant virtual force (**f**) of a searching robot is calculated by vectorially summing up virtual forces calculated for all detected wall blocks as:

$$\mathbf{f} = \sum_{j \in N_j} \mathbf{f_j} ,$$

where N_j denotes set of wall blocks detected by the searching robot.

In order to map the resultant virtual force (**f**) to forward and angular velocities, the *length* (l) and *angle* (ϕ) of the resultant virtual force is calculated as:

$$l = |\mathbf{f}| ,$$

$$\phi = \begin{cases} 0, & \text{if } f(1) = 0 \text{ or } f(2) = 0, \\ atan2(-f(2), -f(1)), & \text{otherwise,} \end{cases}$$

where $f(1)$ and $f(2)$ are the components of $\mathbf{f}$ vector. Afterwards, the angular (w) and forward (v) velocities of the robot are set as:

$$w = \phi \, ,$$

$$v = k_1 v_s \, ,$$

where k_1 is the proportional speed multiplier and v_s is the maximum forward search behavior. k_1 is calculated as:

$$k_1 = \begin{cases} 1, & \text{if } l = 0, \\ \max(0, \pi/2 - |\phi|), & \text{otherwise.} \end{cases} \tag{3.2}$$

3.2.3 Leave Behavior

In the *leave* behavior, the robots avoid other robots and wall blocks for certain duration called leave timeout (t_{leave}). When the leave timeout ends, the leaving robots switch to search behavior. The virtual force for wall blocks being sensed ($\mathbf{f_j}$) is calculated as in equation 3.1.

In order to prevent collisions between the leaving robot and waiting robots around it, the virtual forces for the leaving robot is calculated depending on the distance to its waiting neighbors. If a neighbor robot is relatively far away from the leaving robot, more than $s/2$ units away from the leaving robot, the force is calculated as:

$$\mathbf{f_i} = \begin{bmatrix} r_i \cos(b_i) \\ r_i \sin(b_i) \end{bmatrix} .$$

However, if a neighbor robot is closer than $s/2$ units to the leaving robot, the force is calculated as:

$$\mathbf{f_i} = \begin{bmatrix} k_2(r_i + (s/2)) \cos(b_i) \\ k_2(r_i + (s/2)) \sin(b_i) \end{bmatrix} , \tag{3.3}$$

where k_2 is the collision prevention constant. Using "$(r_i + (s/2)$" instead of "r_i" is to scale the distance so that we don't have larger forces for robots more than $s/2$ units away.

Multiplying the force with collision prevention constant (k_2), helps to balance the opposite force being exerted on the waiting robot. When this constant is *not* used and a robot inside an aggregate leaves, waiting robots in the opposite direction exert a cumulative force which is larger than the force generated by the robot in the leaving direction and leaving robot collides with the waiting robot in the leaving direction.

The resultant virtual force (**f**) for each robot is calculated by vectorially summing up virtual forces calculated for all neighbor robots and wall blocks of a robot as:

$$\mathbf{f} = \sum_{i \in N_i} \mathbf{f_i} + \sum_{j \in N_j} \mathbf{f_j} \ ,$$

where N_i denotes set of robots and N_j denotes set of wall blocks within the robot's sensing range s. Mapping of the resultant virtual force (**f**) to forward and angular velocities is the same except the equation for setting forward velocity. In this behavior, forward velocity is calculated as:

$$v = k_1 v_l \ ,$$

where v_l is the maximum forward speed for leave behavior and k_1 is the proportional speed multiplier calculated with Equation 3.2.

3.2.4 Wait Behavior

In the *wait* behavior, each waiting robot tries to stay at a desired distance ($s/2$) from its neighbors and avoid the walls. The virtual force for wall blocks being sensed ($\mathbf{f_j}$) is calculated as in Equation 3.1.

In order to prevent collisions with other waiting robots in close contact, the virtual force for a robot being sensed is calculated depending on the distance with it. If a neighbor robot is closer than $s/2$ units to the leaving robot, the force is calculated as in Equation 3.3.

If a neighbor robot is more than $s/2$ units away from the waiting robot, the force vector is calculated as:

$$\mathbf{f_i} = \begin{bmatrix} -r_i \cos(b_i) \\ -r_i \sin(b_i) \end{bmatrix} .$$

Note the minus sign at the right of the equation which makes f_i an attraction force instead of a repulsive force. The calculation of the resultant virtual force (**f**) for each robot and mapping of the resultant virtual force (**f**) to forward and angular velocities is same with ones in leave behavior except the equation for setting forward velocity. In this behavior, forward velocity is calculated as:

$$v = k_1 v_w ,$$

where v_w is the maximum forward speed for wait behavior and k_1 is the proportional speed multiplier calculated with Equation 3.2.

In previous wait behavior implementations [39] [55], waiting robots stay motionless during waiting. In this implementation, each waiting robot try to stay away from each of its neighbors at a desired distance ($s/2$) instead of staying motionless. This behavior creates more compact circular aggregates and allows to estimate growing and creation probabilities for aggregates better. Table 3.1 shows default numerical values of simulation parameters.

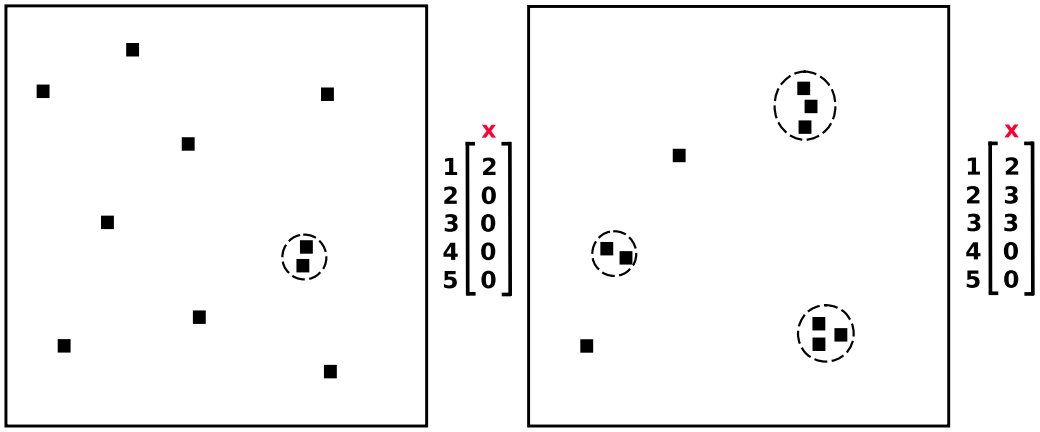
Table 3.1: Default settings for the simulation parameters. Timeout values are written in terms of number of simulation steps. w is the length of arena walls.

Parameter	Default Value
Maximum forward speed in search state (v_s)	0.5
Maximum forward speed in wait state (v_w)	0.05
Maximum forward speed in leave state (v_l)	0.10
Leave timeout (t_{leave})	800
Minimum change direction timeout (t_{min_cd})	65
Maximum change direction timeout (t_{max_cd})	140
Sensing range (s)	4.0
Simulation step (t_s)	0.10
Collision prevention constant (k_2)	6.0
Arena size ($w \times w$)	180×180
Number of searching robots (n)	120
Number of runs	10

3.3 Swarm State Vector

During performance metric calculations and model iteration, the number of aggregates and sizes of these aggregates are kept in a vector called *swarm state vector* (x). The size of the swarm state vector is $n/2$. This is half of the number of robots in the environment and corresponds to maximum possible number of aggregates in the environment. Each vector position contains either the size of an aggregate in the environment or zero which means no record of an aggregate. The mapping of a swarm state to x vector is demonstrated with two example swarm states in Figure 3.4.

The **CreateNewAggregate** function which is used to update the swarm state vector when a new aggregate is formed, is shown in Algorithm 1. In this function, a vector position with zero value is sought sequentially. When this position is found, the value of that vector position is set to 2.



(a) Swarm State 1 (b) Swarm State 2

Figure 3.4: Representation of two swarm states with corresponding swarm state vectors. Large squares represent the arenas. Filled black boxes represent robots. The robots surrounded by dotted circles are waiting robots. In other words, dotted circles represent aggregates. Robots that are not inside dotted circles are searching robots. The contents of swarm state vectors (x) are shown at the right of the environments. Number of searching robots and state of aggregates in these swarm states are as follows (a) eight searching robots and one 2-aggregate (b) two searching robots, one 2-aggregate and two 3-aggregates.

Algorithm 1: The **CreateNewAggregate** function.

Data: Swarm state vector (x)

Result: Updates swarm state vector (x).

for $i \leftarrow 0$ **to** $n/2$ **do**

  if $(x_i == 0)$ **then**

    $x_i \leftarrow 2$;

    `break`;

  end

end

For *dissipation*, *growing* and *shrinking* of an existing aggregate, the value at the position of this aggregate in the swarm state vector is set to zero, incremented or decremented subsequently.

3.4 Performance Metrics

In aggregation literature, three kinds of metrics for measuring the performance of aggregation are used: *distance*, *swarm state* and *time* based metrics.

Distance based metrics are calculated based on the distance of robots from each other and is useful for analyzing effects related to spatial distribution of aggregates. *Total distance* (TD) is a distance based metric which calculates the total of distances between each robot pair [55] as:

$$TD = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} dist(R_i, R_j)\ ,$$

where n is the number of robots and $dist(R_i, R_j)$ denotes the distance of i^{th} to j^{th} robot.

Trianni et. al [60] used a variation of this metric which calculates the average of distances of robots to the center of mass of the robots. The metric can be formally rewritten as:

$$AD = \frac{1}{n}\sum_{i=1}^{n}(1 - \frac{d_i}{d_{max}})\ ,$$

where d_i is the distance of i^{th} robot from the center of mass of robots and d_{max} is the upper bound for d_i in order to have the metric in the interval $[0, 1]$. Since spatial factors are not analyzed in this study, distance based metrics are *not* used.

Swarm state based metrics are used for analyzing the aggregate size distribution for a swarm of robots. In these metrics, the spatial distribution of aggregates are ignored, a neighborhood relationship is defined, robots in the same aggregate are determined using this neighborhood relationship and the aggregate size distribution representing the swarm state is reduced to a single number.

In our study, the following neighborhood relationship that is commonly used in the aggregation literature [55] [8], is used:

$$Neigh(R_i, R_j) = \begin{cases} 1, & \text{if } dist(R_i, R_j) < s \\ 0, & \text{otherwise,} \end{cases}$$

where s is the sensing range of the robots. According to this relationship, robots which are closer than the sensing range are considered as part of the same aggregate. And the transivity of this neighborhood relation is used to determine all aggregates in the environment.

The following *swarm state based metrics* are used in aggregation literature:

- Number of aggregates (NOA),
- Largest aggregate size (LAS),
- Largest aggregate size ratio (LAR),
- Number of searching robots (NS),
- Expected cluster size (ECS).

Number of aggregates (NOA) metric is defined as the number of aggregates in the environment and is used by Garnier et. al [25]. Algorithm 2 shows the algorithm of **CalcNOA** function which calculates the NOA metric from the given swarm state vector.

Algorithm 2: The **CalcNOA** function.

Data: Swarm state vector (x)
Result: Returns number of aggregates (NOA).
for $i \leftarrow 0$ **to** $n/2$ **do**
    $noa = 0$;
    if $x_i > 0$ **then**
        $noa = noa + 1$;
    end
    `return` *noa*;
end

Alternatively, *number of aggregates (NOA)* metric can be calculated for a specific aggregate size. In this version, the number of aggregates with a specified size, instead of number of all aggregates, is calculated. The **CalcNOA** function which calculates the NOA metric for a specific aggregate size, is shown in Algorithm 3.

Algorithm 3: The modified **CalcNOA** function which calculates the *number of aggregates (NOA)* metric for a *specific* target aggregate size.

Data: Swarm state vector (x) and the target aggregate size (t)

Result: Returns number of aggregates (NOA) whose size is t.

for $i \leftarrow 0$ **to** $n/2$ **do**

    $noa \leftarrow 0$;

    if $x_i == t$ **then**

        $noa \leftarrow noa + 1$;

    end

    `return` noa;

end

The *largest aggregate size (LAS)* metric equals to the size of the largest aggregate in the environment and heavily used in the literature [39], [25]. The **CalcLAS** function which calculates the LAS metric from the given swarm state vector, is shown in Algorithm 4.

Algorithm 4: The **CalcLAS** function.

Data: Swarm state vector (x)

Result: Returns the largest aggregate size (LAS).

$las \leftarrow 0$;

for $i \leftarrow 0$ **to** $n/2$ **do**

    if $x_i > las$ **then**

        $las \leftarrow x_i$;

    end

end

`return` las;

The *largest aggregate size ratio (LAR)* is the normalized version of LAS metric and calculated by dividing the number of robots forming the largest aggregate to the total number of robots. It is previously used in [8] and [6]. **CalcLAR** function which calculates the LAR metric, is shown in Algorithm 5.

Algorithm 5: The **CalcLAR** function.

Data: Swarm state vector (x) and number of robots (n)

Result: Returns the largest aggregate size ratio (LAR).

$las \leftarrow$ `CalcLAS(`x`);`

$lar \leftarrow las/n$;

`return` lar;

Another swarm state based metric is the *number of searching robots (NS)* which is equal to the number of free robots that do *not* belong to any aggregate. **CalcNS** function which calculates the NS metric from the given swarm state vector and the number of robots, is shown in Algorithm 6.

Algorithm 6: The **CalcNS** function.

Data: Swarm state vector (x) and number of robots (n)

Result: Returns the number of searching robots (NS).

$total \leftarrow 0$;

for $i \leftarrow 0$ **to** $n/2$ **do**

  $total \leftarrow total + x_i$;

end

$ns = total/n$;

`return` ns;

The last swarm state based metric described in this section, *expected cluster size (ECS)*, is proposed and used by Soysal and Şahin [55] in order to obtain an average aggregate size from diverse aggregate sizes in a swarm state. It is calculated by summing the aggregate sizes and dividing the total by number of robots as:

$$ECS = \frac{1}{n} \sum_{i=1}^{n} x_i \ .$$

Time based metrics are used for performing temporal analysis on aggregation experiments. They are useful to test whether an aggregation solution is scalable in terms of the time required to obtain the solution or not. To the best of our knowledge, the only time based metric used by aggregation related studies is in [6]. In this study, Ame et. al plotted time needed to aggregate %80 of the robots on an aggregate.

In order to perform temporal analysis on our aggregation experiments, steady state of these experiments are found automatically with methods described in the next section. In order to show distribution of aggregatates at a certain time, NOA, LAS and NS metrics are used. Distance based metrics are *not* used in this thesis.

3.5 Automatic Steady State Detection

Steady state is the performance and the time step of an experiment in which the experiment performance according to a performance metric is stabilized. In this section, we describe how the steady state is detected automatically for experiments presented in this chapter. The method is applied to both simulation and microscopic model experiments in four stages.

In the first stage, *performance metric* and *sampling period* ($t_{sampling}$) is chosen for the experiment. Recording and using *performance samples* instead of performance for all steps is important in order to obtain experiment results in a reasonable time and *does* not effect the steady state being found. During the experiment, robot positions are recorded at every $t_{sampling}$ step. By recording robot positions instead of performance results and delaying calculation of performance, we prevent running the same experiment multiple times for analyzing the same experiment for different metrics.

In the second stage, performance values for a chosen or particular metric is calculated for each recorded sample. This is performed for all runs and the result is saved to a $n_r \times n_s$ matrix called **samples** where n_r is the number of runs and n_s is the number of samples in a run. n_s is calculated as:

$$n_s = \frac{t_{max} \times (\frac{t_{sim}}{t})}{t_{sampling}} .$$

In the third stage, mean performance values for each recorded sample over all runs is calculated. **CalcMeanPerf** function which calculates the mean performance values and save them into the vector called **mean**, is shown in Algorithm 7.

Algorithm 7: The **CalcMeanPerf** function.

Data: Performance samples (*samples*), number of runs (n_r) and number of samples (n_s).

Result: Mean performance values are calculated and saved into the vector named *mean*.

```
for i ← 1 to n_s do
    total ← 0;
    for j ← 1 to n_r do
        total ← total + samples_{i,j};
    end
    mean_i ← total /n_r;
end
```

In the fourth stage, the steady state detection algorithm is chosen and executed. Two steady state detection methods are used in this thesis: *maximum performance* and *sliding window* steady state detection methods.

The *maximum performance* steady state detection method works by finding the maximum of the mean performance values. The maximum value found and the step corresponding to the position of this value in the mean performance vector are set as the steady state value and staedy state time subsequently. **FindSteadyWithMax** function which implements the maximum performance steady state detection method, is shown in Algorithm 8.

Algorithm 8: The **FindSteadyWithMax** function.

```
Data: Mean performance values (mean).
Result: Calculates steady state time (steadytime) and steady state value
        (steadyperf).
mean ← CalcMeanPerf();
max ← 0;
maxt ← 0;
for i ← 0 to n_s do
    if mean_i < max then
        max ← mean_i;
        maxt ← i;
    end
end
steadytime ← maxt;
steadyperf ← max;
```

In *sliding window* steady state detection method, a sliding window is moved on the **mean** vector and the mean value for the values inside this window is compared to the average performance of all samples. If the *percentage error* between these two values is below a threshold (ϵ), the average performance of the window and the step corresponding to the starting position of the current sliding window in the mean performance vector are set as the steady state value and steady state time subsequently. **FindSteadyWithSW** function which implements the sliding window steady state detection method, is shown in Algorithm 9.

Algorithm 9: The **FindSteadyWithSW** function.

Data: Mean performance values (*mean*), window size (w) and target error rate (r_e).

Result: Calculates steady state time (steadytime) and steady state value (steadyperf).

```
mean ← CalcMeanPerf();
for i ← w to n_s do
    total ← 0;
    for j ← 0 to w − 1 do
        total ← total + mean_{i−j};
    end
    avg ← total /w;
    diff ← abs(avg − mean_i);
    targeterror ← mean_i r_e;
    if diff < targeterror then
        steadytime ← i − 1;
        steadyperf ← mean_i;
        break;
    end
end
```

CHAPTER 4

Aggregation Model

In this chapter, we describe a probabilistic geometric model for self-organized aggregation. The chapter consists of three sections. In the first section, we describe the assumptions of our aggregation model with the key assumption that any aggregation experiment is a random sequence of four possible aggregation events: *creation*, *growing*, *shrinking* and *dissipation*.

The model estimates the occurence probabilities of these events in a certain duration. Step by step derivation of these equations are described in the second section. Finally, in the third section, we present the results of simulation experiments used to verify the event probabilities of the model.

4.1 Assumptions and Issues

In this section, assumptions of our aggregation model are described. First, we describe assumptions about *robots* such as sensing characteristics, initial state and positions of robots. Secondly, model's assumptions related to *aggregates* are described. Specifically, we describe how an aggregate consists of three circles geometrically in our model's perspective and how we calculate radius of these circles using circle packing theory [36]. Next, *aggregation events* and related issues are described. Finally, *model iteration* and *spatiality* related issues are discussed subsequently. Related to model iteration, selecting a model time step and effect of leave timeout on model iteration are discussed. Regarding spatiality, spatial issues that make our model semi-spatial are discussed.

4.1.1 Robots

We assumed that robots are initially placed in a bounded arena. Using a bounded arena removes the possibility that some robots may get lost during searching and is common in the aggregation studies [60] [8] [56] [17].

Initial positions of searching robots are random. Robots do *not* have any knowledge regarding the size of the arena or the number of robots. These assumptions rule out any solutions that may be specific for a particular number of robots, arena size or initial robot positions.

Another assumption about robots is the *limited sensing range*. Robots can *only* sense the presence of other robots or the environment walls within their sensing range and measure the distance to them. This assumption is one of the fundamental assumptions that makes our model realistic and general purpose.

4.1.2 Aggregate

An aggregate is defined as a group of two or more waiting robots. In our model, it is assumed that waiting robots inside an agregate are $s/2$ away from each other.

An illustration of an aggregate with seven robots is shown in Figure 4.1. The waiting robots are shown as filled black squares. Three circles with radiuses of $R_0 = s/4$, R_1 and R_2 exists around the waiting robots.

The reason for drawing these circles around waiting robots is to have a connection to *circle packing theory* [58] and use this connection to derive probabilities of aggregation events. 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 robot as circle with a radius of $R_0 = s/4$ and assume the given boundary is the circle with radius of R_1. Using this formulation, R_1 can be calculated approximately using the radius of inner circles ($s/4$). For an outer circle with radius R_1, the following relation is known to hold from circle packing research.

$$\frac{R_1}{R_0} \approx am^b \ , \tag{4.1}$$

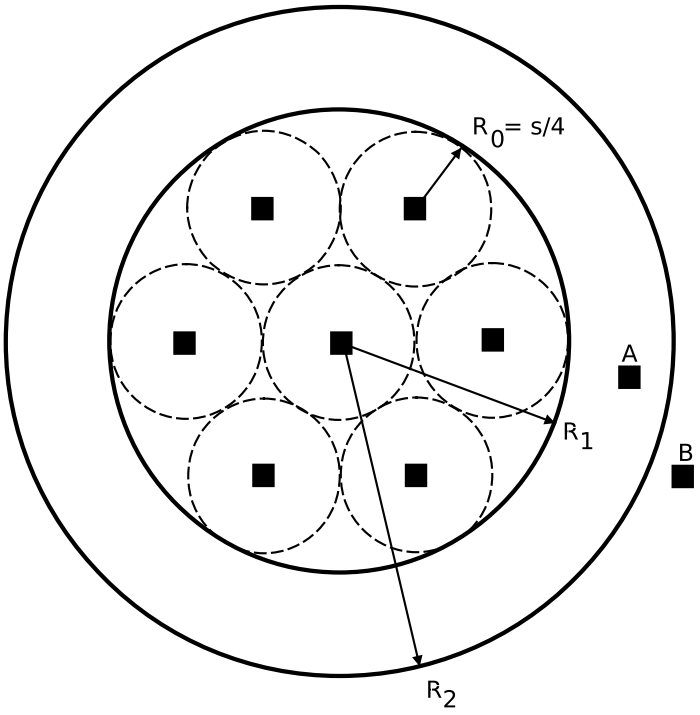


Figure 4.1: The illustration of an arbitrary aggregate with seven robots. Robots are represented as filled black squares. Three types of circles are drawn around the robots with following radiuses: $s/4$, R_1 and R_2. First, we draw a circle with radius of $s/4$ units around each robot. Then, we draw another circle outside of these circles which is tangent to these circles. The radius of this circle is defined as R_1 and can be estimated with circle packing theory. Finally, one more concentric circle around this circle whith radius of R_2 is drawn. R_2 is calculated using R_1 and the circle with this radius corresponds to the *attraction circle* of the aggregate. Searching robots inside the attraction circle of the aggregate are detectable by the waiting robots while searching robots outside of this region is undetectable by the waiting robots. For example, robot A is detectable and robot B is undetectable by the waiting robots in this aggregate.

where m is the aggregate size, a (1.20) and b (0.48) are constants. These constants are obtained by fitting a curve on top of the best known packings of equal circles in the unit circle [58] as shown in Figure 4.2.

After replacing R_0 with $s/4$ and rearranging Equation 4.1, we can calculate R_1 as:

$$R_1 \approx \frac{sam^b}{4} .$$

When b is assumed to be 0.5, R_1 can be rewritten as:

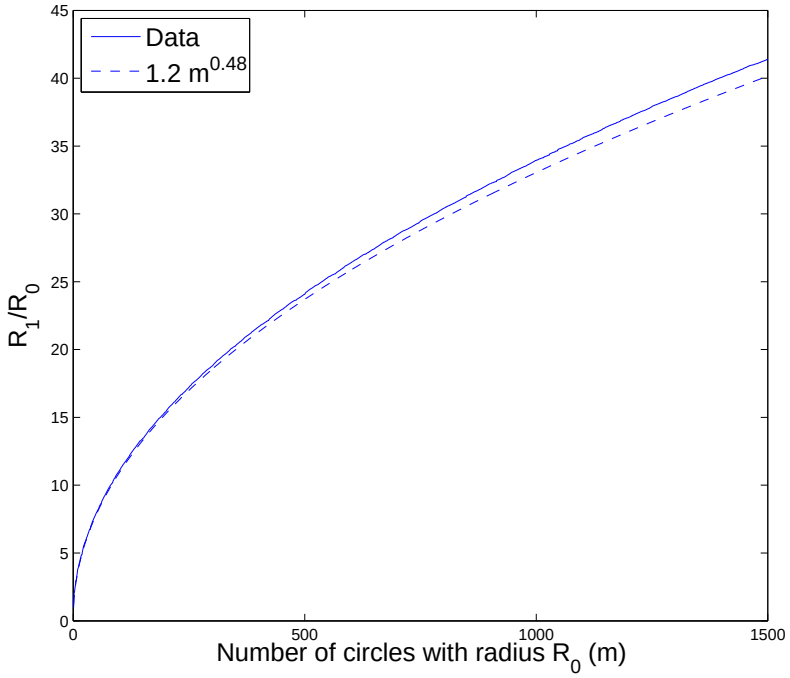


Figure 4.2: Curve fitting for R_1/R_0 data obtained from the best known packings of equal circles in the unit circle ($R_1 = 1$) [58]. Regular curve shows the R_1/R_0 from the data and dotted curve shows the am^b curve when a and b are 1.20 and 0.48 subsequently.

$$R_1 \approx \frac{sa\sqrt{m}}{4} . \tag{4.2}$$

The area of circle with radius R_2 corresponds to the maximum area which can be sensed by the waiting robots. Only searching robots inside this area are detected by waiting robots. This area is called as the *attraction circle* of the aggregate. When a searching robot enters into this area, it is attracted toward the aggregate. R_2 is derived from R_1 as:

$$R_2 = R_1 - s/4 + s = R_1 + 3s/4 .$$

Using Equation 4.2, R_2 can be rewritten as:

$$R_2 \approx \frac{sa\sqrt{m}}{4} + 3s/4 = \frac{sa\sqrt{m} + 3s}{4} = \frac{s(a\sqrt{m} + 3)}{4} . \tag{4.3}$$

CalcAggrRadius function which calculates R_2 of the circle with given size, is shown in

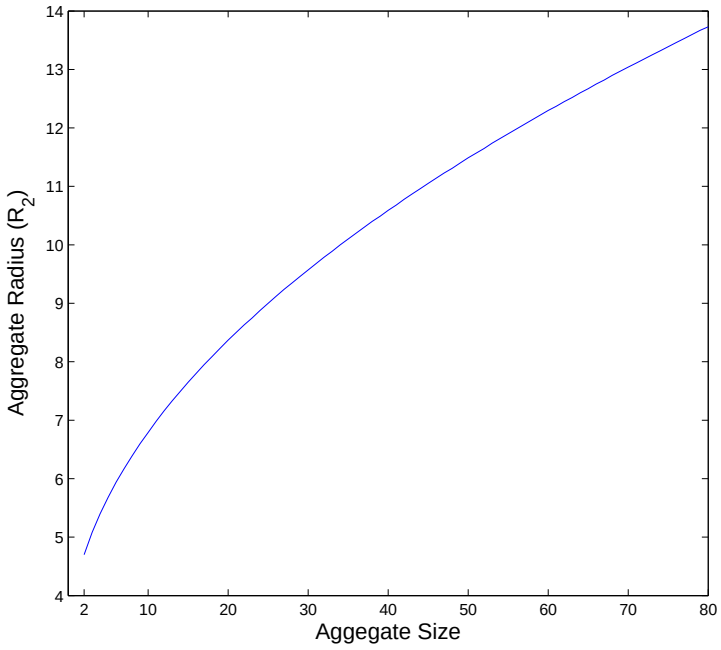


Figure 4.3: Radius of the *attraction circle* (R_2) of aggregates with varying sizes calculated using **CalcAggrRadius** function shown in Algorithm 10.

Algorithm 10.

Algorithm 10: The **CalcAggrRadius** function.

Data: Aggregate size (m) and sensing range (s).

Result: Returns aggregate radius (R_2).

$a \leftarrow 1.20$;

$r1 \leftarrow sa\sqrt{m}/4$;

$r2 \leftarrow r1 + (3s/4)$;

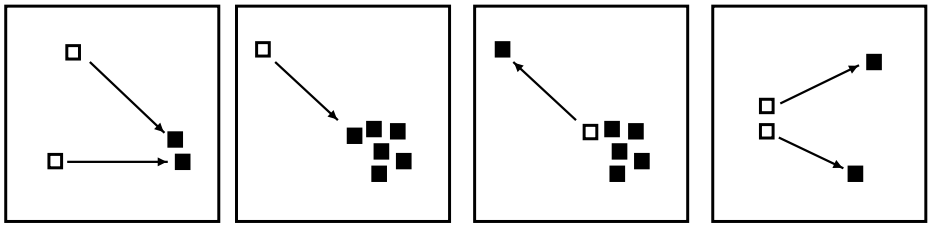
`return` $r2$;

e

Figure 4.3 shows R_2 for aggregates with varying sizes calculated using Algorithm 10.

4.1.3 Aggregation Events

There are four probabilistic events that can occur in aggregation: *creation of an aggregate*, *growing of an aggregate*, *shrinking of an aggregate* and *dissipation of an aggregate* as illustrated in Figure 4.4.



(a) Creation (b) Growing (c) Shrinking (d) Dissipation

Figure 4.4: Four events that can occur in aggregation: (a) creation of an aggregate, (b) growing of an aggregate, (c) shrinking of an aggregate and (d) dissipiation of an aggregate. Arrows show direction of moving robots during the event. Filled squares represent the positions of robots in the current model time step. Empty squares represent the positions of robots in the previous model time step. Filled squares that are close to each other are robots that are part of an aggregate.

The *creation event* (Figure 4.4-a) occurs when *two* searching robots find each other in one model step. The rendezvous of more than two robots in a model time step is ignored due to its low probability.

The *growing event* (Figure 4.4-b) occurs when *one* searching robot finds an existing aggregate. The situation in which the aggregate is found by more than one searching robot in the same model time step is ignored due to its low probability and to simplify the growing probability formula.

The *shrinking event* (Figure 4.4-c) occurs when a waiting robot leaves its aggregate. The situation in which more than one waiting robot leaving the aggregate in the same model time step is ignored due to its low probability and to simplify the shrinking probability formula.

The *dissipation event* (Figure 4.4-d) occurs when one of the waiting robots in an aggregate of size two leaves the aggregate. The situation where both of the waiting robots leaving the aggregate within the same model time step is ignored due to its low probability.

4.1.4 Model Iteration

Iterating the model is necessary to analyze different aggregation experiments. In order to iterate a model, a time step for the model should be selected. The same time step can be used for both simulation and the model but this has the disadvantage of having a slower iteration speed compared to selecting a larger time step for the model. However, the model time step should *not* be chosen too large. Otherwise, same aggregation event may happen

more than once in one model step. As discussed in the previous section, these situations are not considered in our model and may cause errors in model predictions. In order to find a good value for model time step, we checked the number of these multiple events during model verification and set the model step as ten times the simulation step to minimize the occurrence of multiple events of the same type in one model step.

Another issue related to the iteration of the model is the *leave timeout* (t_{leave}). During model iteration, the time that is spent in leaving is ignored and a robot leaving its aggregate is assumed to leave the aggregate immediately. Since this can effect model's prediction capacity, t_{leave} in the simulation experiments has to be selected as low as possible.

4.1.5 Spatial Issues

Our geometric model is based on non-spatial modeling approach which relies on the assumption that spatial effects are ignored.

We assume that free robot density is uniform within the environment and the position of an aggregate does not change the probability of events. For instance, an aggregate has the same growing probability whether the aggregate is placed in the center, at the periphery of the environment or close to other aggregates. This assumption holds for environments with low robot density.

Another spatiality issue is related to the t_{leave}. If it is set to small values, leaving robots may not go far enough from the aggregate in the selected t_{leave} duration and return back to the aggregate. In that case, model probabilities would be biased and results of simulation experiments may not match well with model results. In order to prevent this, t_{leave} should be set to values that are high enough.

Another important spatiality issue is the effect of position of waiting robots to the shrinking probability. Although previous studies assume that any waiting robot can leave its aggregate, this is *not* a correct assumption. When aggregates get larger, many of the waiting robots are surrounded by other waiting robots and only waiting robots close to the periphery of aggregates will be able to leave their aggregates. This is depicted in Figure 4.5 in which arbitrary aggregates with sizes from two to seven are depicted.

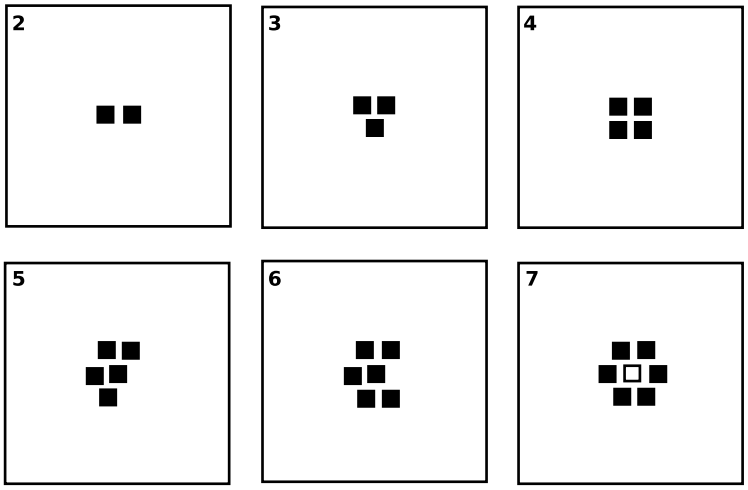


Figure 4.5: Illustration of arbitrary aggregates with 2, 3, 4, 5, 6 and 7 robots subsequently. Up to aggregates of six robots, any robot can leave the aggregate. However, the robot at the center (distinguished as an empty rectangle) of the aggregate with seven robots, can *not* find a path to leave the aggregate. The problem becomes more serious when aggregates get larger and more waiting robots are not be able to leave their aggregates.

In order to minimize and control this spatial effect, we used a sophisticated waiting behavior implementation which creates circular aggregates, estimated the number of waiting robots at the periphery of the aggregate from circle packing theory and use this estimation to calculate the shrinking probability.

4.2 Model Probabilities

In this section, we derive *creation*, *growing*, *shrinking* and *dissipation* probabilities through geometric analysis. However, before starting their derivations, we should introduce *free area* and *searching robot density* concepts.

Free area (A_f) is the area that is *not* occupied by aggregates and searching robots in the environment. A_f is calculated as:

$$A_f = w^2 - \sum_{i=1}^{n/2} \pi R_2(x_i)^2 \ ,$$

where x is the swarm state vector, w is the wall length, R_2 is the radius of an aggregate and n is the total number of robots. **CalcFreeArea** function which calculates the free area in the

environment, is shown in Algorithm 11.

Algorithm 11: The **CalcFreeArea** function.

Data: Swarm state vector (x), arena wall length (w) and total number of robots (n).

Result: Returns the free area (A_f).

$A_f \leftarrow w * w$;

for $i \leftarrow 0$ **to** $n/2$ **do**

  if $x_i \neq 0$ **then**

    $r \leftarrow$ `CalcAggrRadius(`x_i`);`

    $A_f \leftarrow A_f - \pi r^2$;

  end

end

`return` A_f;

The *searching robot density (d)* is defined as the number of searching robots per unit area and calculated as:

$$d = \frac{y}{A_f},$$

where y is the number of searching robots. **CalcDensity** function which calculates the searching robot density, is shown in Algorithm 12.

Algorithm 12: The **CalcDensity** function.

Data: Swarm state vector (x), arena wall length (w), total number of robots (n) and number of searching robots (y).

Result: Returns the searching robot density (d).

$A_f \leftarrow$ `CalcFreeArea(`x, w, n`);`

$d \leftarrow y/A_f$;

`return` d;

4.2.1 Creation Probability

The creation probability is defined as the probability of two searching robots meeting and forming a 2-aggregate among all searching robots in one model time step and calculated

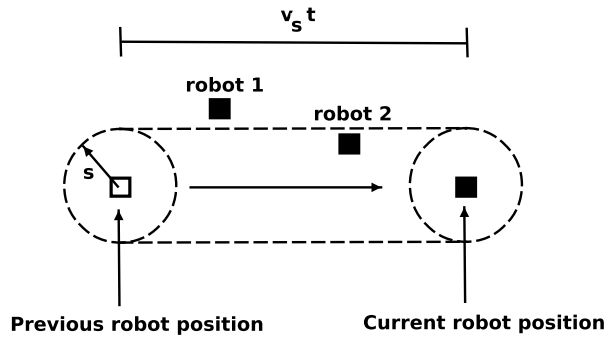


Figure 4.6: The sketch representing the movement of a searching robot with speed v_s for a duration of a model step t where s is the sensing range from center of a robot to the center of another robot. The searching robot sweeps an area of size $2sv_st$ in one model time step. Robots are indicated with filled squares and the previous position of the searching robot is indicated with an empty square. Robots whose center is inside the swept area are detected by the searching robot. While robot 1, whose center is outside the swept area, will not be detected by the searching robot; robot 2, whose center is on the surface of the swept area, will be detected by the searching robot.

using *mean free path* derivation in kinetic theory of ideal gases [3]. The derivation of the corresponding equation is performed in four steps. First, we calculate the area swept out by one searching robot in one model time step. Then, we find the number of collisions betwen this searching robot and other robots assuming that other robots are *immobile* inside this area. In the third step, we extend this equation to find total number of collisions among all robots. In the final step, we remove the assumption that the robots inside the swept area are immobile and obtain the final equation.

The area swept by one searching robot in one model time step, depicted in Figure 4.6, is calculated as:

$$\text{Swept area} = 2sv_st \ .$$

If any other searching robots are inside this area, they are detected by this searching robot assuming that other robots are immobile (mobility of other robots will be considered in later steps of derivation). The number of robots in this swept area is calculated by multiplying the swept area with the searching robot density as:

$$\text{Number of robots in the swept area} = 2sv_std \text{ (Multiply with } d) \ ,$$

where d is the searching robot density. This is also equal to the number of collisions encountered by this searching robot in one model time step. In order to find the number of collisions among all searching robots, assuming that the robots that are inside their swept area are immobile, we multiply this result with number of searching robots (y) as:

$$\text{Total number of collisions} = 2sv_s tdy \text{ (Multiply with } y\text{)} . \tag{4.4}$$

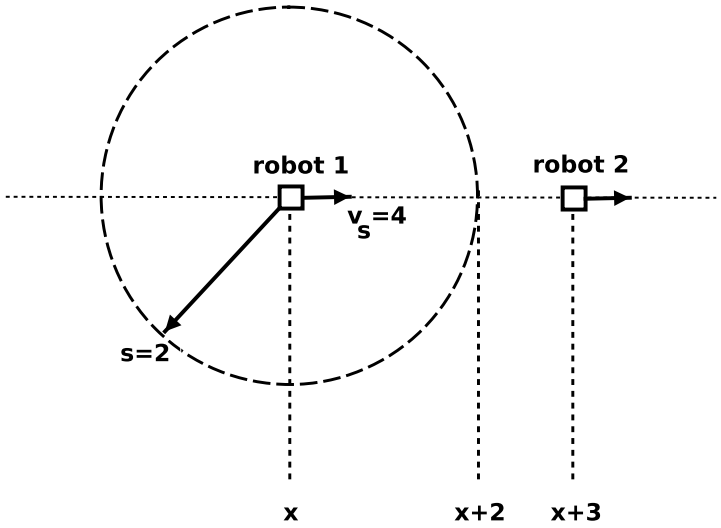
However, the collisions are counted twice in this formula. For example, the collision between robot-1 and robot-2 is counted as robot-1's collision. But the same collision is counted as robot-2's collision with robot-1 too. Hence the total number of collisions calculated with Equation 4.2.1 needs to be divided into two as:

$$\text{Total number of collisions (1st update)} = sv_s tdy \text{ (Divide by 2)} .$$

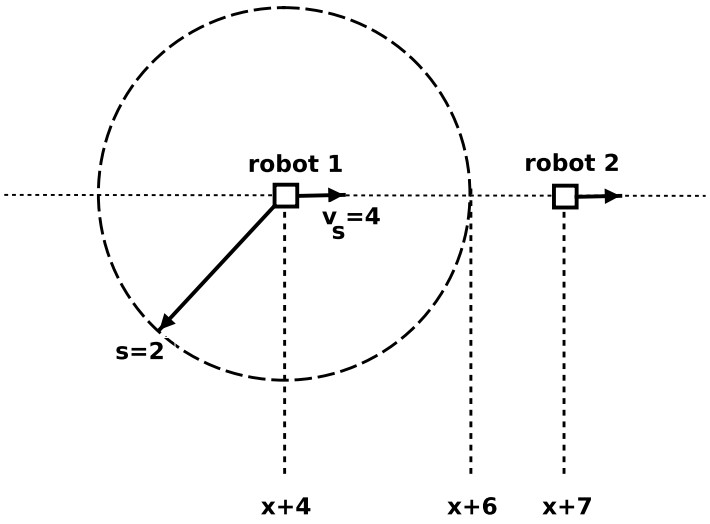
So far, we assumed that other searching robots that are inside the swept area are *immobile*. However, these are searching robots too and they have their own speeds and their speeds effect the total number of collisions. The issue is exemplified in Figure 4.7-a where two robots with the same *velocity* are shown. Although, the second robot is inside the swept area of the first one, it is *not* inside the sensing range of the first robot. When the second robot is assumed to be *immobile*, two robots will collide in the next step. However, they will *not* collide in the next step if the second robot is assumed to have a speed with the same direction. Figure 4.7-b shows new positions of these robots in the next step. As can be seen, the second robot is still inside the swept area of the first one. However, it is still *not* inside the sensing range of the first robot.

This problem is *not* specific to robots with the same velocity. Depending on velocities of robots, even if the second robot is inside the swept area of the first robot in the current step, it may *not* be in the sensing range of the first robot in the next step. The solution to this problem is replacing the *average speed* in above equation with *relative speed*. However, as shown in Figure 4.8, there are several variations of relative speeds and an *average relative speed* should be found.

When the average speed is v_s, average relative speed is shown to be $\sqrt{2}v_s$ in [3]. When we replace average speed with average relative speed, we obtain:



(a) Previous robot positions at time t



(b) Current robot positions at time $t + 1$

Figure 4.7: Illustration of movements of two searching robots by showing robot positions in two consecutive time steps. The previous robot positions are shown on (a) and the current robot positions are shown on (b). The sensing range (s), searching speed (v_s) and model time step (t) is assumed to be 2, 4 and 1 subsequently. In this settings, searching robots moves 4 units and sweeps 8 unit^2 area in a model time step. Although robot 2 is in the swept area of robot 2 at both step t and $t + 1$, it is outside of the sensing range of robot 1 and can *not* be detected by robot 1.

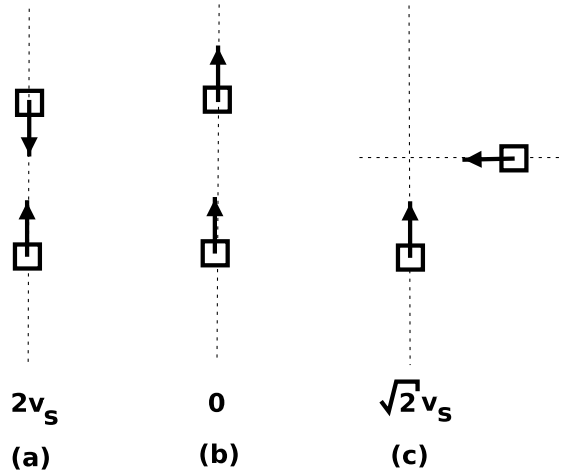


Figure 4.8: Three variations of relative speeds for two searching robots assuming that searching speed is v_s.

$$\text{Total collisions (2nd update)} = \sqrt{2}sv_s tdy \text{ (Replace } v_s \text{ with } \sqrt{2}v_s) \,.$$

The total number of collisions above equals to the *creation probability* and can be rewritten in terms of number of searching robots and free area subsequently as:

$$P_c = \sqrt{2}sv_s tdy = \frac{\sqrt{2}sv_s ty^2}{A_f} \,. \tag{4.5}$$

CalcPcreate which calculates the creation probability, is shown in Algorithm 13.

Algorithm 13: The **CalcPcreate** function.

Data: Number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s) and model step (t)

Result: Returns aggregate creation probability.

$P_c \leftarrow \sqrt{2}sv_s tdy$;

`return` P_c;

4.2.2 Growing Probability

The growing probability of an aggregate is defined as the probability of a searching robot joining that aggregate in one model time step. It is calculated geometricaly in our model. First, we find the area in which searching robots can grow the aggregate in one model time

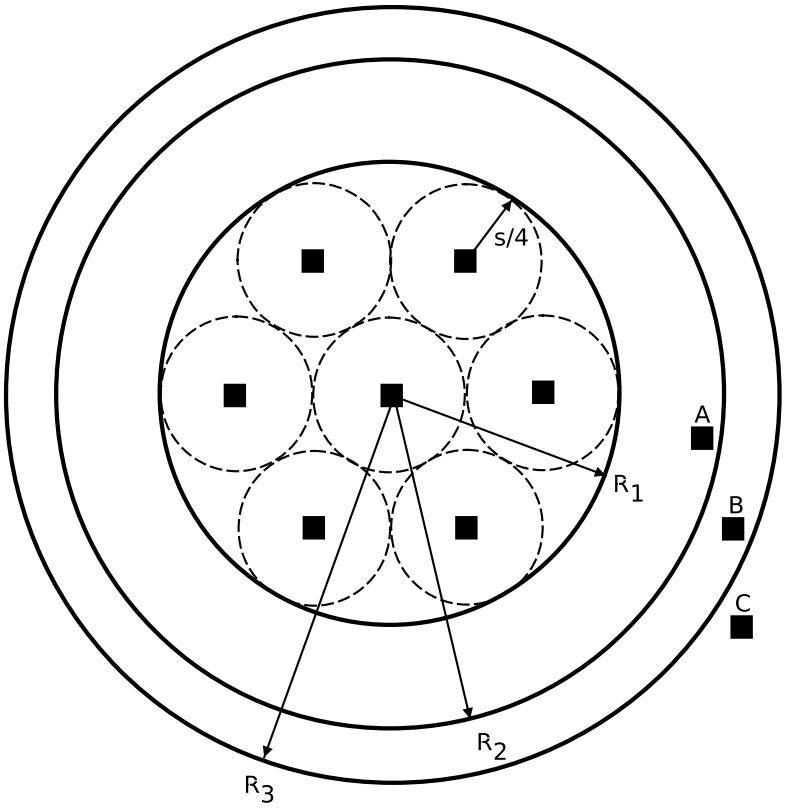


Figure 4.9: The illustration of an arbitrary aggregate with seven waiting robots. Four circles are drawn around the robots with following radiuses: $s/4$, R_1, R_2 and R_3. First, we draw a circle with radius of $s/4$ units around each robot. Then, we draw another circle outside of this circle which is also tangent to these circles. The radius of this circle is defined as R_1. Then, we draw two more concentric circles around this circle whith radiuses of R_2 and R_3. R_2 and R_3 is calculated using R_1, and R_1 is approximately calculated from circle packing theory.

step. Then, we find the average number of robots in that area which can move in the correct direction and join the aggregate.

Figure 4.9 is a modified version of the aggregate figure that is presented in section 4.1.2. As discussed before, the circle with radius R_2 is called as the attraction circle and robots inside this circle is part of the aggregate. Searching robots can only be outside of this circle.

The concentric circle with radius R_3 around the aggregate is drawn to indicate the area in which searching robots can grow the aggregate within one model time step. R_3 is calculated as:

$$R_3 = R_2 + v_s t \ .$$

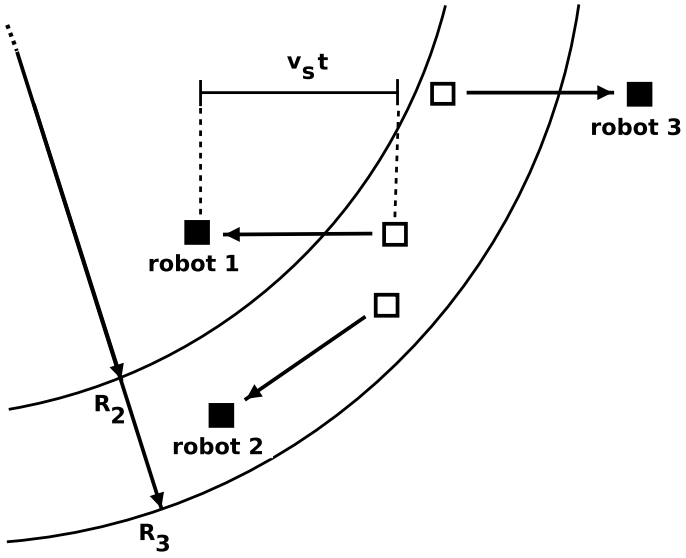


Figure 4.10: Illustration of movements of three searching robots in the R_2-R_3 band of an aggregate for one model time step. All searching robots move $v_s t$ units in one model step where v_s is the searching robot speed and t is the model time step. Searching robots are indicated with filled squares and previous positions of searching robots are indicated with empty squares. Although all three robots start moving while they are inside the R_2-R_3 band, only robot 1 enters to the attraction circle of the aggregate and grow the aggregate. This is due to direction of robots' velocities. Only a certain ratio of searching robots (k_3) in the R_2-R_3 band of an aggregate can grow the aggregate and this portion should known in order to calculate the growing probability of the aggregate.

Since the searching robots can move $v_s t$ units in one model time step, only searching robots that are less than $v_s t$ units away from attraction circle can grow the aggregate in one model time step. This corresponds to the R_2-R_3 band and can be calculated as:

$$\text{Ring Area} = \pi(R_3^2 - R_2^2) \, .$$

In order to find number of searching robots in the R_2-R_3 ring, we multiply the area with the searching robot density (d) as follows

$$\text{Number of searching robots in the ring area} = \pi(R_3^2 - R_2^2)d \, .$$

Now, we know average number of searching robots that can grow the aggregate. However, depending on the velocity of searching robots, only a certain ratio of these searching robots can grow the aggregate. This problem is depicted in Figure 4.10.

We call this ratio as k_3 and the derivation of k_3 is described in the next section. Finally, the equation for growing probability is obtained by multiplying the average number of searching robots in the ring area with k_3 as follows.

$$P_g = k_3\pi(R_3^2 - R_2^2)d\ . \tag{4.6}$$

CalcPgrow function which calculates the growing probability for an aggregate of size m, is shown in Algorithm 14.

Algorithm 14: The **CalcPgrow** function.

Data: Aggregate size (m), searching robot density (d), sensing range (s) and searching robot speed (v_s) and model time step (t) and .

Result: Returns growing probability for the aggregate.

$R_2 \leftarrow$ `CalcAggrRadius(`m`);`

$R_3 \leftarrow R_2 + v_s t$;

$k_3 \leftarrow$ `CalcAvgK3(`R_2, R_3`);`

$A_r \leftarrow \pi(R_3^2 - R_2^2)$;

$P_g \leftarrow k_3 A_r d$;

`return` P_g;

CalcAggrRadius function calculates the radius of the attraction circle of the aggregate and its algorithm is previously shown in Algorithm 10. **CalcAvgK3** function which calculates the average k_3 parameter for the aggregate is presented in Algorithm 16 in the next section.

4.2.3 Calculation of k_3

k_3 parameter of the growing probability formula is calculated in two steps. In the first step, we find a formula for calculating k_3 for any point in the R_2-R_3 band of an aggregate. In the second step, we find the average k_3 over all points in the R_2-R_3 band.

The formula for calculating k_3 for any point in the R_2-R_3 band of an aggregate is derived geometrically using *law of cosines* [2]. Figure 4.11 illustrates initial steps of this geometric the derivation.

In Figure 4.11-a, an aggregate is shown at the origin and a searching robot is placed at position

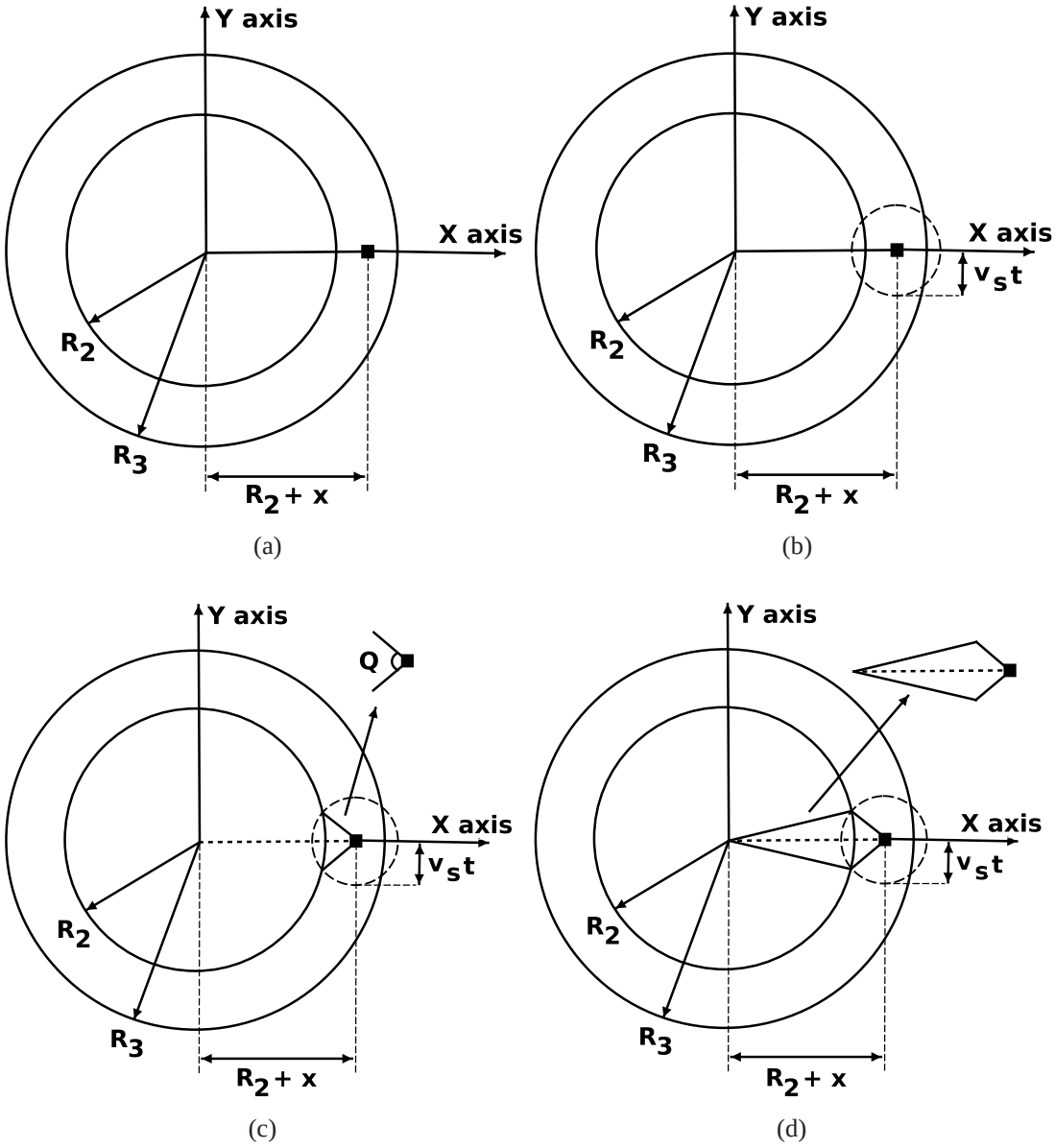


Figure 4.11: Illustration of initial steps of geometric derivation of k_3 for any point in the R_2-R_3 of an aggregate. The aggregate is placed at the origin of a cartesian coordinate system. (a) A searching robot, denoted with a filled rectangle, is placed at position (R_2+x) along the x axis where x is a random real number in $[0, R_3\text{-}R_2]$ range. (b) A circle, called *movement circle*, with radius $v_s t$ is added around the robot. The periphery of the movement circle contains all possible points this searching robot can be in the next step. (c) Two lines are drawn from the current position of the robot to the intersection points of the movement circle and the R_2 circle. The ratio of the angle between these two lines (Q) to 2π corresponds to the ratio of successfull movements, movements which grow the aggregate, to all possible movements of the searching robot. (d) Two new lines are drawn between the intersection points of the lines added on (c) and the center of the aggregate. With addition of these lines, two identical triangles are formed. One of these triangles will be used to derive k_3.

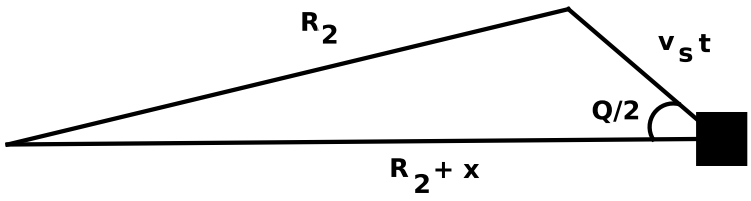


Figure 4.12: One of the identical triangles obtained in Figure 4.11-d. When law of cosines is used on one of these triangles, it is possible to calculate Q. And using Q, k_3 is calculated using Equation 4.7.

(R_2+x) along the x axis where x is a random real number in $[0, R_3\text{-}R_2]$ range. A circle, called *movement circle*, with radius $v_s t$ is added around the robot in Figure 4.11-b. The periphery of the movement circle contains all possible points this searching robot can be in the next step. Then, in Figure 4.11-c, two lines are drawn from the current position of the robot to the intersection points of the movement circle and the R_2 circle. The angle between these two lines are called Q in radians. If the searching robot moves within Q angle, it will enter to the attraction circle and join the aggregate. If it moves in the remaining (2π-Q) angle, it can *not* grow the aggregate.

k_3 for displacement x is calculated by dividing Q to 2π which corresponds to the ratio of successfull movements to all possible movements of the searching robot as:

$$k_3(x) = \frac{Q}{2\pi} \, . \tag{4.7}$$

In order to calculate Q, in Figure 4.11-d, two new lines are drawn between the intersection points discussed above and the center of the aggregate. This forms two identical triangles in that region and one of these triangles are redrawn in Figure 4.11-e. Now using this triangle, we can derive the following equation using law of cosines as:

$$(R_2)^2 = (R_2 + x)^2 + (v_s t)^2 - 2(R_2 + x)(v_s t)\cos(Q/2) \, . \tag{4.8}$$

When this equation is solved for Q, we obtain:

$$Q = 2\arccos\left(\frac{x^2 + 2R_2x + 0.25}{R_2 + x}\right).$$

When Q in Equation 4.8 is replaced with this formula, we obtain the final formula for k_3 for displacement x as:

$$k_3(x) = \frac{\arccos\left(\frac{x^2+2R_2x+0.25}{R_2+x}\right)}{\pi}. \tag{4.9}$$

CalcK3 function which implements Equation 4.9 in order to calculate k_3 for displacement x, is shown in Algorithm 15.

Algorithm 15: The **CalcK3** function.

Data: The radius of the attraction circle of the aggregate (R_2) and the displacement in R_2-R_3 band (x).

Result: Returns k_3 for displacement x.

$k_3 = \frac{\arccos\left(\frac{x^2+2R_2x+0.25}{R_2+x}\right)}{\pi}$;

`return` k_3;

Figure 4.13 shows k_3 values calculated using Equation 4.7 for varying displacements (x) in R_2-R_3 band of a 20-aggregate.

In order to calculate the average k_3 for all possible displacements in R_2-R_3 band, R_2-R_3 band is divided into n sub-bands. This is depicted on Figure 4.14 in which $(n-1)$ concentric circles are drawn inside R_2-R_3 band. The width of sub-bands (w_b) are calculated as:

$$w_b = \frac{R_3 - R_2}{n}.$$

The radius of concentric circle which specifies the outer border of band i ($r_b(i)$) is calculated as:

$$r_b(i) = R_2 + (i+1)w_b,$$

and the distance of the middle points of the band i to the attraction circle ($c_b(i)$) is calculated as:

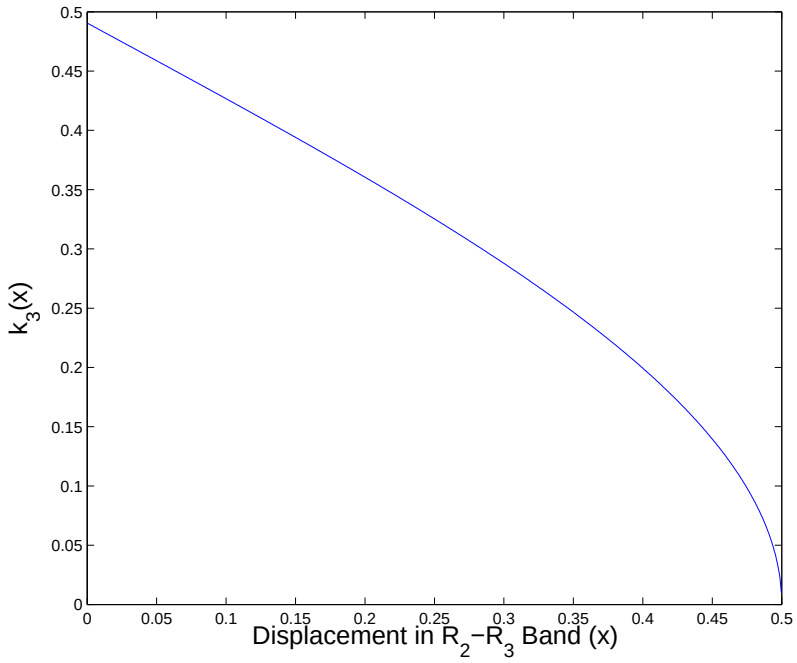


Figure 4.13: k_3 values calculated using Equation 4.7 for varying displacements (x) in R_2-R_3 band of a 20-aggregate

$$c_b(i) = r_b(i) - (w_b/2)\ ,$$

where w_b, $r_b(i)$ and $c_b(i)$ are depicted in Figure 4.15.

The area of sub-band, annulus formed by two circles of radii $r_b(i-1)$ and $r_b(i)$, is calculated as:

$$a_b(i) = \pi(r_b(i)^2 - r_b(i-1)^2)\ .$$

The average k_3 is calculated by calculating a sub-band-area weighted average of k_3 over all sub-bands as:

$$\text{Average } k_3 = \frac{\sum_{i=0}^{n} a_b(i) \times k_3(c_b(i) - R_2)}{\pi(R_3^2 - R_2^2)}\ , \tag{4.10}$$

where the numerator is weighted total of the k_3 for all bands and the denominator is the total area of R_2-R_3 band.

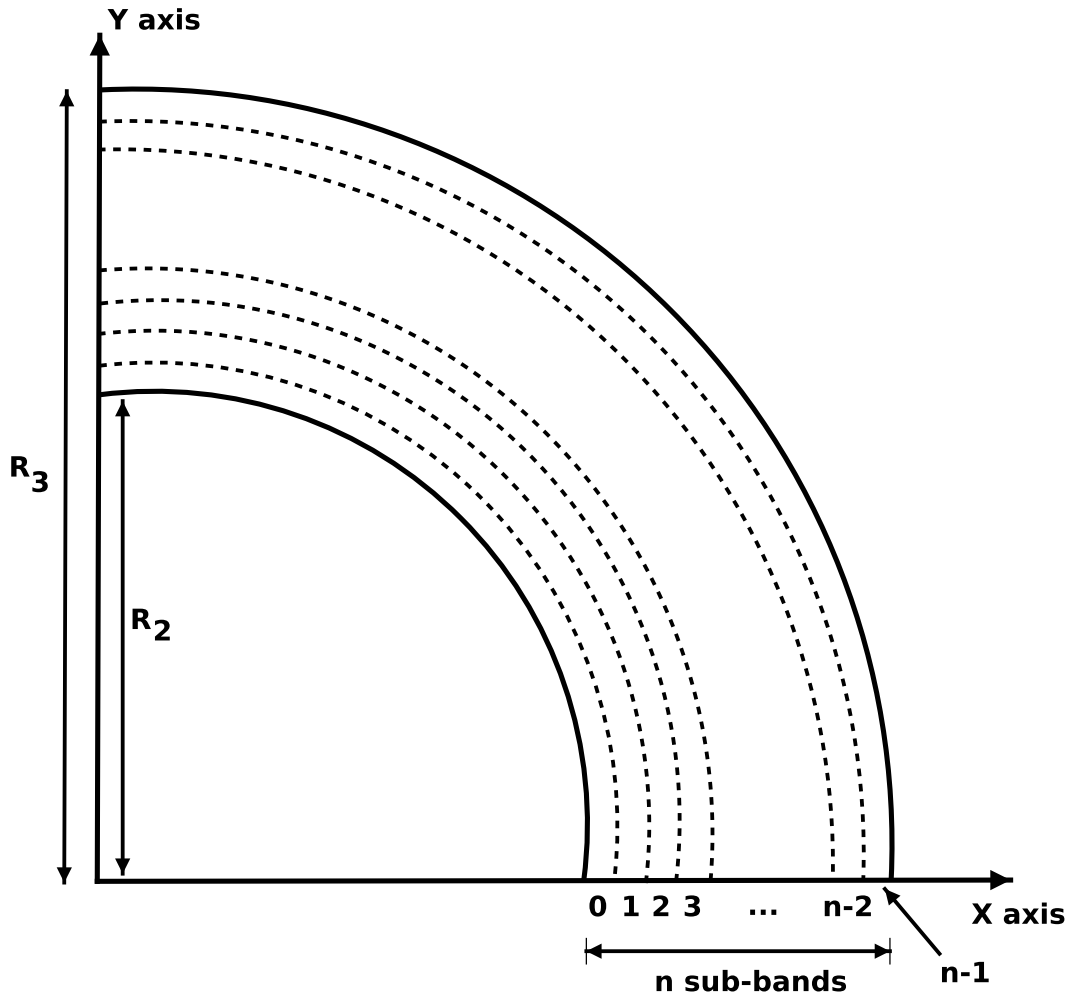


Figure 4.14: The illustration of sub-bands in the R_2-R_3 band of an aggregate. Dotted concentric circles represent the outer border of sub-bands $0 \ldots n-2$.

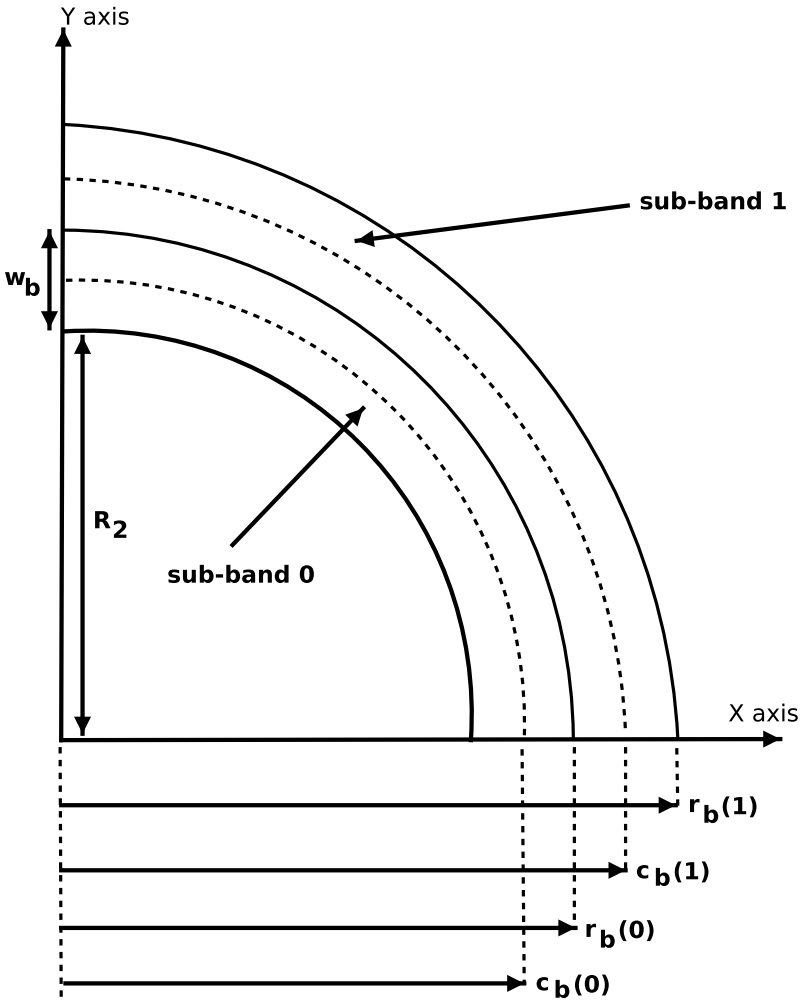


Figure 4.15: The first two sub-bands in R_2-R_3 band of an aggregate. Equally spaced concentric circles represent the borders of sub-bands. Dotted concentric circles represent middle points of these sub-bands. The average k_3 for all possible displacements is calculated using the radius of concentric circles (r_b), width of sub-bands (w_b), distance of sub-band centers to the aggregate center (c_b) and area of sub-bands (a_b). w_b of sub-bands and r_b and c_b of first two sub-bands are illustrated on the figure.

CalcAvgK3 function which calculates average k_3 for R_2-R_3 band using Equation 4.10, is shown in Algorithm 16.

Algorithm 16: The **CalcAvgK3** function.

Data: R_2 and R_3 of the aggregate.

Result: Returns average k_3.

$total = 0$;

$w_b = \frac{R_3 - R_2}{n}$;

for $i \leftarrow 0$ **to** n **do**

  $r_b = R_2 + (i + 1)w_b$;

  prev $= R_2 + iw_b$;

  $c_b = r_b - (w_b/2)$;

  $a_b = \pi(r_b^2 - \text{prev}^2)$;

  $k_3 = \texttt{CalcK3}(R_2, c_b(i) - R_2)$;

  $total = total + (a_b k_3)$;

end

`return` $\frac{total}{\pi(R_3^2 - R_2^2)}$;

Figure 4.16 shows average k_3 values calculated using Algorithm 16 for varying aggregate sizes.

4.2.4 Shrinking Probability

Shrinking probability is the probability one robot leaving from an aggregate of size m in one model time step. The shrinking probability is calculated geometricaly in our model. First, we find the number of robots that can leave from the aggregate. Then, we find shrinking probability by multiplying the number of robots that can leave the aggregate with the leaving probability.

Aggregates that are obtained with our implementation are circular and compact due to the *wait* behavior. This allows us to assume that robots at the periphery of the aggregate stay $(s/4)$ away from the periphery of the circle with radius R_1 towards the center of the aggregate. Expected positions of the robots at the periphery of an aggregate are illustrated as a circle in Figure 4.17-a for an aggregate with seven robots. The periphery of this circle is calculated as:

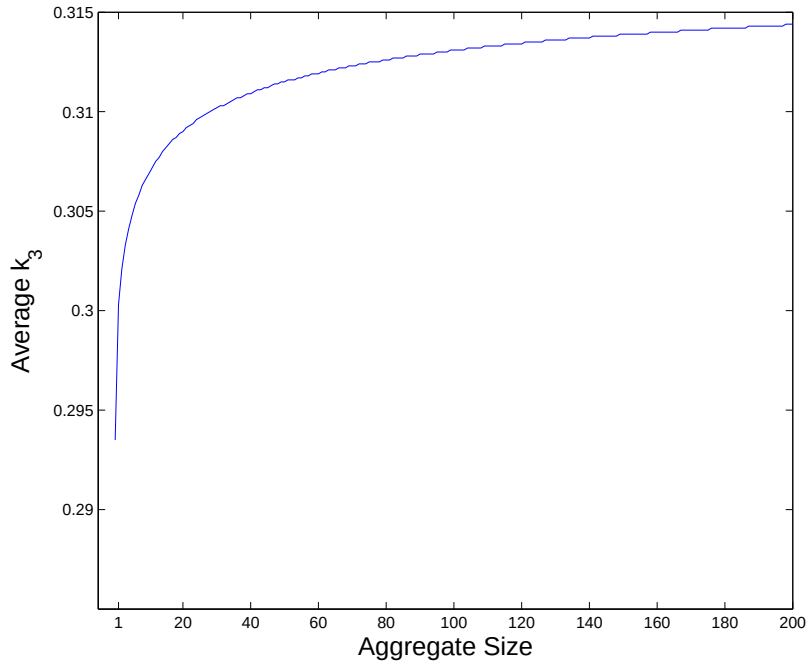


Figure 4.16: Average k_3 for varying aggregate sizes. Average k_3 values are calculated using Algorithm 16.

$$\text{Periphery of the inner circle} = 2\pi(R_1 - s/4)\ .$$

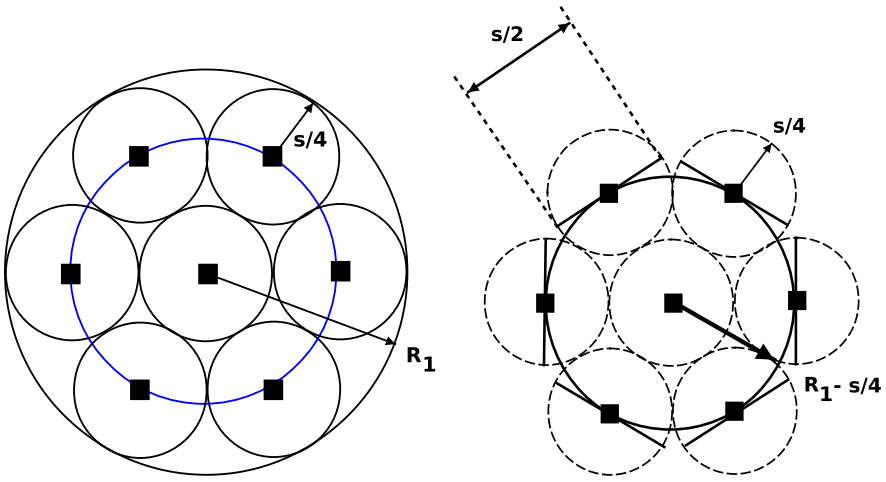
Figure 4.17-b shows the circle with radius $R_1 - s/4$ and diameters of small circles that are adjacent to this circle.

Our assumption is that when the aggregate gets larger, these diameters connect to each other and form a polygon with many edges and our estimation gets more realistic.

With this assumption, we can estimate the number of waiting robots at the periphery of an aggregate by dividing periphery of the circle with radius $(R_1 - s/4)$ by $(s/2)$ as:

$$\text{Number of robots at the periphery} \approx \frac{2\pi(R_1 - s/4)}{s/2}\ .$$

The larger the aggregate, the better this estimation is. When we substitute R_1 with the right side of Equation 4.2, we obtain:



(a) (b)

Figure 4.17: Illustration for estimation of the number of waiting robots at the periphery of an aggregate by dividing periphery of the circle with radius $(R_1 - s/4)$ by $(s/2)$. (a) A circle with radius $(R_1 - s/4)$ is drawn on top of a regular aggregate with seven robots. (b) Diameters of small circles that are adjacent to the larger circle are drawn. The number of waiting robots at the periphery of an aggregate is estimated as the periphery of the larger circle divided by the diameter of the smaller circle.

$$\text{Number of robots at the periphery} \approx \pi(a\sqrt{m} - 1)\ .$$

Finally, the leaving probability is obtained by multiplying the number of waiting robots at the periphery of the aggregate by the leaving probability (p).

$$P_s \approx \pi(a\sqrt{m} - 1)p\ . \tag{4.11}$$

CalcPshrink function which calculates the shrinking probability for an aggregate of size m, is shown in Algorithm 17.

Algorithm 17: The **CalcPshrink** function.

Data: Aggregate size (m) and leaving probability (p).

Result: Returns shrinking probability for the aggregate.

$a = 1.20$;

$P_s \leftarrow \pi(a\sqrt{m} - 1)p$;

`return` P_s;

4.2.5 Dissipation Probability

The dissipation event occurs when at least one of the robots in an aggregate of size two leaves the aggregate. Since both robots can leave the aggregate at the same model time step, it is calculaed as:

$$P_d = 2p \ . \tag{4.12}$$

CalcPdissipate function which calculates the dissipation probability, is shown in Algorithm 18.

Algorithm 18: The **CalcPdissipate** function.

Data: Leaving probability (p).

Result: Returns dissipation probability for a 2-aggregate.

$P_d \leftarrow 2p$;

`return` P_d;

CHAPTER 5

Experimental Model

In this chapter, we verify the creation, shrinking and growing probability formulas derived in the previous section using simulation experiments conducted in the Stage simulator. In each experiment, the probability that will be verified is calculated by finding the average number of these events occuring in one model step. Table 5.1 summarizes the experiments permormed in this chapter.

Table 5.1: Summary of experiments performed in Chapter 5. The first column shows the probability being verified and the section of the experiments. Following three columns specify which aggregation event is verified in this section. Aggregation event being tested in each section contains "+" sign in its corresponding column. Remaining columns show parameters being varied in experiments. "+" and "-" sign in a column means that parameter corresponding to that column is and is *not* varied respectively. SR, SP and DE are sensing range, speed and density of the searching robots respectively. AS is the aggregate size.

Probability (Section)	**Creation**	**Shrinking**	**Growing**	**SR**	**SP**	**DE**	**AS**
Creation (5.1)	+	-	-	+	+	+	-
Shrinking (5.2)	-	+	-	-	-	-	+
Growing (5.3)	-	-	+	-	-	+	+

5.1 Creation Probability

In this section, we verify creation probability equation (Equation 4.5) that is proposed for estimating the creation probability. We present our results as three interesting cases. In the first case, the creation probability is verified for five different searching robot densities. For

each density, different number of searching robots are placed in an environment with default arena size and the effect of number of robots on the creation probability is analyzed.

The second and third cases are designed to show the effect of *sensing range* and *speed* of searching robots on the creation probability. Sensing range and speed of searching robots are varied in an environment with default wall length and effects of these changes are analyzed.

Experiments for all cases are initiated with searching robots at random locations and velocities. All experiments are performed 10 times for 24000 simulation steps. During runs, aggregate creation is disabled by setting leaving probability of searching robots to 1. In order to calculate the creation probability during a run, the number of collisions between searching robots are counted and divided to the number of steps in a run. Creation rate for a run is obtained by dividing the creation probability to the free area in that run.

Figure 5.1 plots the creation probability with respect to the number of searching robots. Each box represents the interquartile interval. The line in the middle of box is the median, and whiskers are minimum and maximum values of creation probability. The line drawn on top of boxes connects the estimated creation probability (calculated using Equation 4.5) for each value of number of searching robots. In this case, it can be seen that model estimations for creation probability match well with simulation results and creation probability is proportional with the square of number of searching robots.

Figure 5.2 plots the creation probability for varying *sensing ranges* of searching robots in the default environment. Plus signs under the last box represent outliers in that data. It can be seen that model and simulation estimations for this experiment match well even though the sensing range is varied considerably in both directions and the creation probability is linearly proportional with the sensing range of searching robots.

Insignificant under-estimations of the model for large values of sensing range and number of searching robots can be attributed to *sensing range effect*. In kinetic theory of gases, it is assumed that the gas particles collide with each other and wall blocks elastically. However, in our simulation experiments, searching robots turn away from other searching robots or wall blocks when they sense them. This reduces the free area in practice and is *not* included to the model. Since increasing both number of searching robots and sensing range increase this effect, the model calculates relatively smaller creation probability values for these settings.

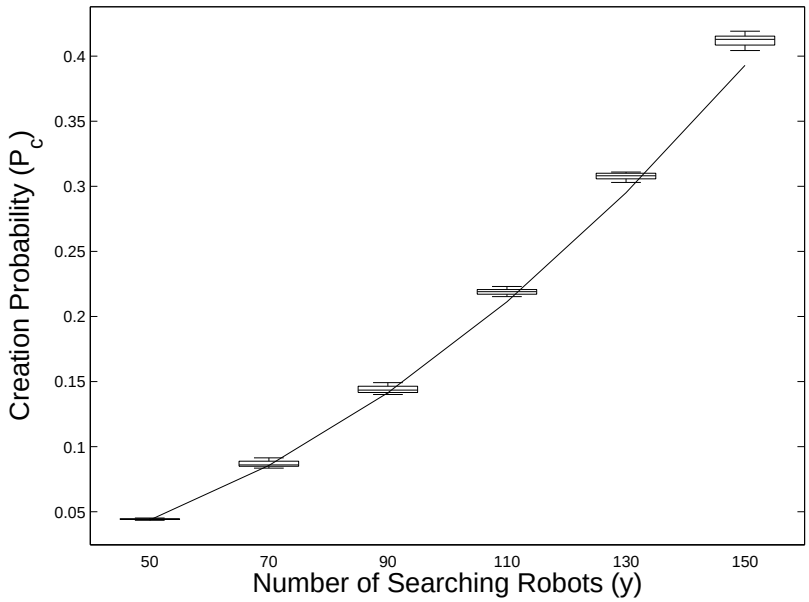


Figure 5.1: The creation probability with respect to the *number of searching robots*. The number of searching robots changes along *x* axis and *y* axis shows the creation probability. Creation probabilities calculated with simulation experiments are shown as box plots. The line plots the model predictions for the creation probability. Boxes in the middle represents interquartile interval. The line in the middle of boxes are the median, and whiskers are minimum and maximum values of creation probability. Default speed and sensing range are used for searching robots. Unless otherwise stated, all experiments presented in this chapter were run using default arena size, number of searching robots, speed, sensing range and number of runs.

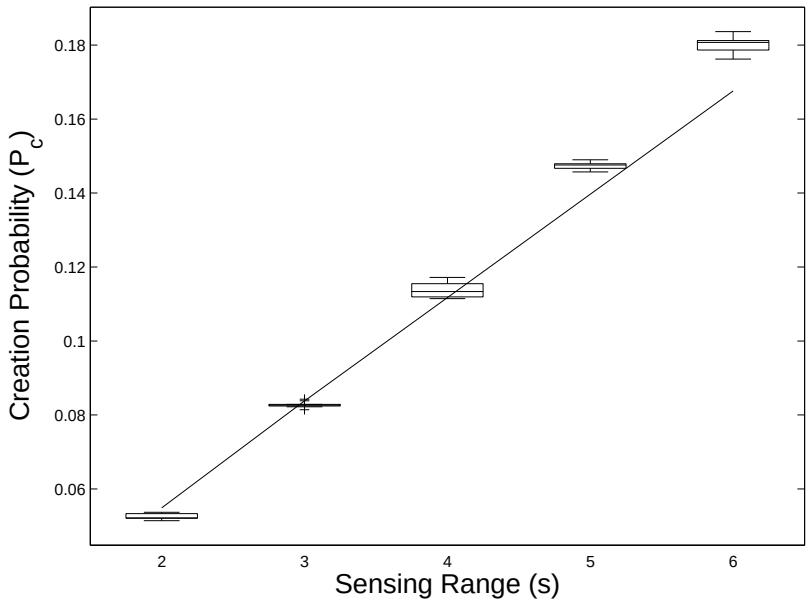


Figure 5.2: Change of creation probability for different *sensing ranges* for searching robots. Creation probabilities calculated with simulation experiments are shown as box plots. The line plots the model predictions for the creation probability. Plus signs under the last box represent outliers in that data. 80 searching robots are used in these experiments.

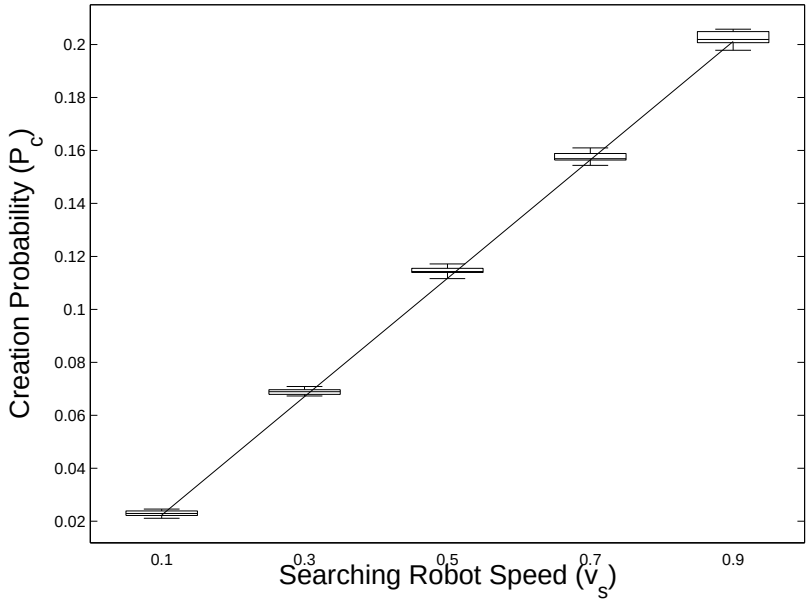


Figure 5.3: Change of creation probability for different *searching robot speeds*. Searching robot speed changes along *x* axis and *y* axis shows the creation probability. Simulation results are presented as box plots. The model predictions are drawn as a line. 80 searching robots are used in these experiments.

In Figure 5.3, creation probability for searching robots with varying *speeds* are shown. It can be seen that model and simulation estimations for creation probability match well even though the searching robot speed is decreased to $1/5$th and increased to approximately double of the default speed value.

5.2 Shrinking Probability

In this section, shrinking probability prediction of our model, shown in Equation 4.11, is verified using simulation experiments conducted in the Stage simulator. Assuming that the aggregate whose shrinking probability will be verified has a size of m, simulation experiments are initiated with an aggregate of size m at the center of the environment.

In order to obtain an aggregate of size of m, a seperate experiment, called *aggregate forming experiment*, is conducted. In this experiment, one robot, called the seed robot, is placed at the center of an environment. In order to make the seed robot stationary, its state and leaving probability are set to "wait" and zero subsequently. In addition to the seed robot, $(m-1)$ searching robots are placed randomly in the same environment. During the experiment, searching robots are prevented to form new aggregates with other searching robots while they

are allowed to join to the the seed robot or its neighbors. The leaving probability is set to zero, so that they can *not* leave the only aggregate in the environment. When an aggregate of size m is obtained, the experiment run is terminated and positions of the waiting robots in the aggregate are saved to be used in the shrinking experiment.

For shrinking probability verification experiment, an aggregate is placed by placing waiting robots to positions that are obtained from the aggregate forming experiment. A predefined and constant leaving probability is assigned to each waiting robot in the aggregate. As in creation probability verification, multiple runs are made for each experiment. The runs are terminated when a waiting robot leaves the aggregate *successfully* or when the maximum number of steps for the experiment is reached. If a successful shrinking happens, the shrinking time for that run is recorded.

Before describing calculation of shrinking probability from shrinking times, we will describe the possible outcomes of the leaving behavior. When a waiting robot starts to leave its aggregate, there are three possible outcomes of this action. If the robot is inside the aggregate and surrounded by other robots, the robot fails to leave the aggregate and return back to the wait state. This is the *unsuccessful leaving*. If the robot is at the periphery of the aggregate and leaves the aggregate before the leave timeout (t_{leave}), this is *successful leaving* and has two different types: *multiple* or *single* shrink. *Multiple shrink* occurs when multiple waiting robots leave the same aggregate within the same model step. Otherwise, *single shrink* occurs.

In order to calculate the shrinking probability, (1) number of single shrinks (n_{sh}), multiple shrinks (n_{msh}) and remaining runs (n_{run}) are tracked for each model time step, (2) shrinking probability is calculated for each model step and (3) median of shrinking probabilities is calculated.

The remaining runs are the runs in which single or multiple shrinks did *not* occur yet and calculated as follows for model step t where $t > 0$:

$$n_{run}(t) = \begin{cases} n_{run}(0), & \text{if } t = 1 \\ n_{run}(t-1) - (n_{sh}(t-1) + n_{msh}(t-1)), & \text{otherwise,} \end{cases}$$

where $n_{run}(t-1)$ is the number of remaining runs in the previous model step and $n_{run}(0)$ is the initial number of runs.

Shrinking probability for model step t is calculated as:

$$P_s(t) = \frac{n_{sh}(t) + 2n_{msh}(t)}{n_{run}(t)} \, . \tag{5.1}$$

Table 5.2 demonstrates shrinking probability calculation for selected model steps. It is seen that the shrinking probability for the initial model steps is zero. This is because of the time required to go outside of the attraction circle of the aggregate. We observe that the time required to leave an aggregate is in 15-60 model step range depending on the size and compactness of the aggregate and the position of the waiting robot. This effect can be observed further by investigating the shrinking probabilities in subsequent model steps. It can be seen that the shrinking probability in model steps 20-22 are still unstable and increasing. The shrinking probability is being stabilized only after model steps 60-62.

The median of shrinking probabilities for all steps is calculated as:

$$P_s = \frac{\sum_{t=t_{start}}^{t_{max}} P_s(t)}{\sum_{t=t_{start}}^{t_{max}} n_{run}(t)} \, , \tag{5.2}$$

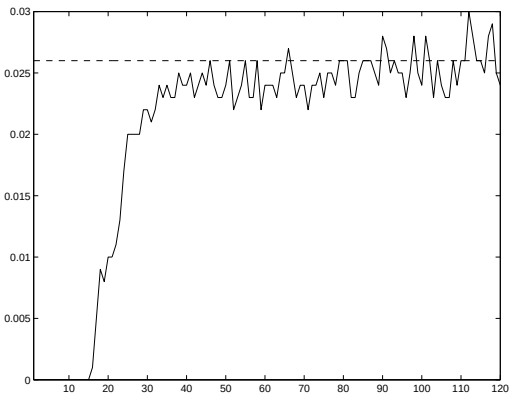
where t_{max} is the maximum number of model steps for a run and t_{start} is the begining model step to calculate the median of shrinking probability. The value of t_{start} is selected as 60 for shrinking verification experiments in order to exclude unstable values of shrinking probabilities at the begining of the experiment.

Figure 5.4 plots shrinking probability with respect to aggregate size. The dotted horizontal line represents the median of shrinking probabilities of all steps calculated using equation 5.2. It can be seen that fluctations of leaving probability estimations increase when model step and aggregate size are increased. This is due decreasing number of remaining runs dring the experiment. When few samples are left for estimating leaving probability, fluctations increase. Since larger aggregates shrink more quickly than smaller aggregates, remaining runs decrease more quickly and fluctations are observed earlier for larger aggregates.

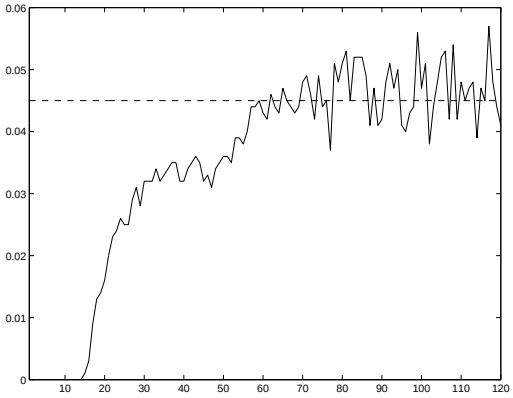
Figure 5.5 plots the median of shrinking probabilities for varying aggregate sizes and model predictions calculated using Equation 4.11. It can be seen that model and simulation estimations for shrinking probability match. The difference between the simulation results and model predictions is less than %3 of simulation results even for the largest aggregate size (80)

Table 5.2: Demonstration of shrinking probability calculation for selected model steps.

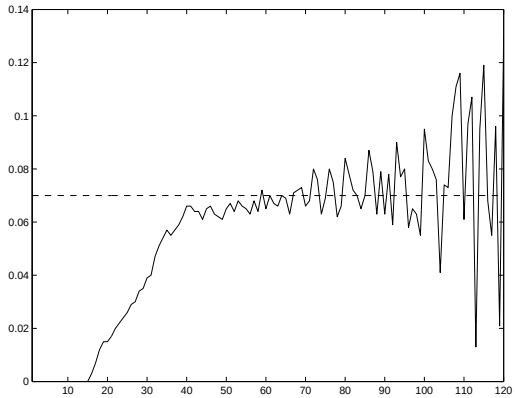
Model Step (t)	**Remaining runs (n_{run})**	**Single shrinks (n_{sh})**	**Multiple shrinks (n_{msh})**	**P_{shrink}**
1	37620	0	0	0
2	37620	0	0	0
3	37620	0	0	0
...	...	...	...	...
20	36385	361	0	0.0099
21	36024	398	1	0.0111
22	35625	464	4	0.0131
...	...	...	...	...
60	14767	351	3	0.0240
61	14413	349	1	0.0243
62	14063	327	1	0.0233
...	...	...	...	...
80	9007	236	0	0.0262
81	8771	201	4	0.0234
82	8566	196	3	0.0232
...	...	...	...	...



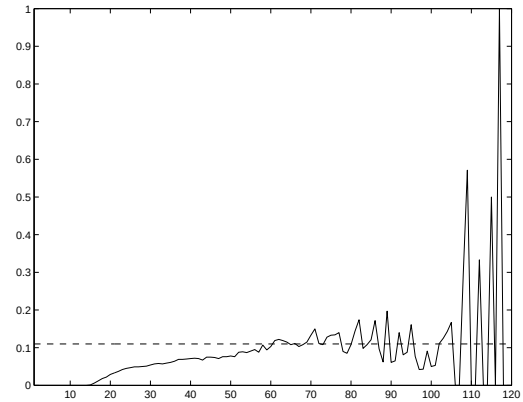
(a) Aggregate size: 5



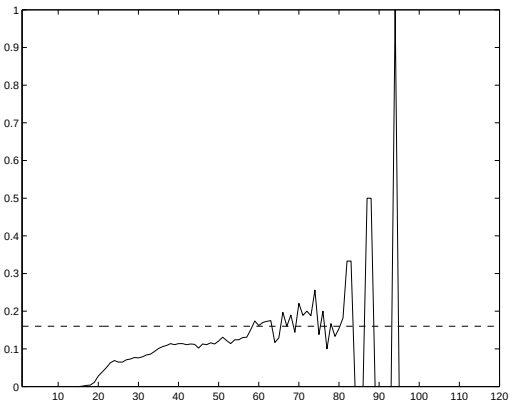
(b) Aggregate size: 10



(c) Aggregate size: 20



(d) Aggregate size: 40



(e) Aggregate size: 80

Figure 5.4: Median of shrinking probabilities for different model steps. Each plot shows shrinking probabilities for a different aggregate size: (a) 5, (b) 10, (c) 20, (d) 40 and (e)80. While x axis shows the model step (t), y axis shows the growing probability (P_g). 40000 runs are performed for each experiment. Leaving probability (p) is 0.0005. For finding the median of growing probabilities, $t_{start} = 60$ and $t_{max} = 120$.

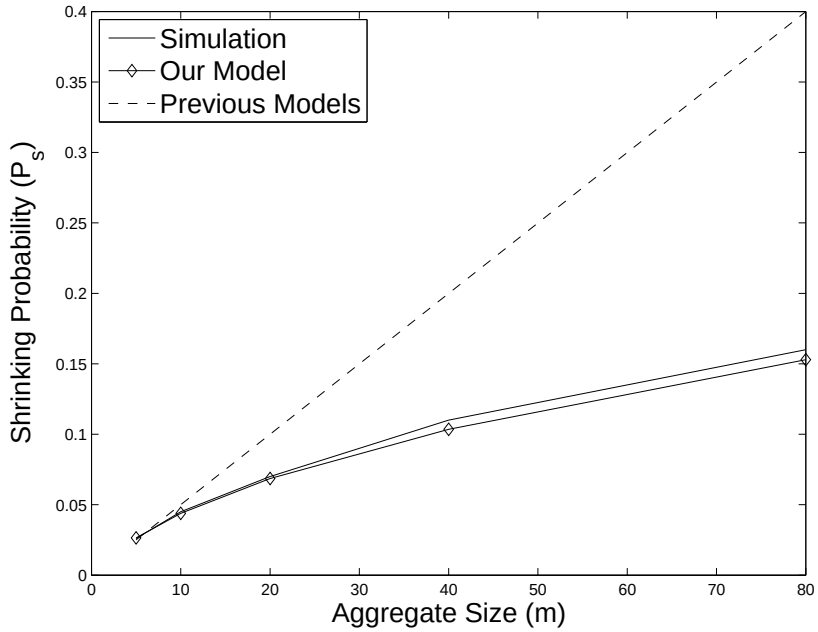


Figure 5.5: Shrinking probability predictions for varying aggregate sizes. This is the summary of experimental results shown in Figure 5.4. Simulation results, predictions of our model and predictions of previous models are shown. Previous models assume that shrinking probability is linearly proportional with the aggregate size.

being tested. As shown on the same figure, this error would be more than %100 of simulation result if shrinking probability predictions in previous studies were used.

5.3 Growing Probability

In this section, growing probability prediction of our model, shown in Equation 4.6, is verified using simulation experiments conducted in the Stage simulator. Assuming that the aggregate whose growing probability will be verified has a size of *m*, simulation experiments are initiated with an aggregate of size *m* at the center of the environment and *n* randomly placed searching robots.

The same *aggregate forming experiment* described in the previous section is used to obtain an aggregate of size *m*. The leaving probability of waiting robots are set to 0 so that they can not leave from the aggregate during the experiments. As in shrinking probability verification, multiple runs are made for each type of experiment. Each run is terminated when a searching robot enters to the attraction circle of the aggregate or the maximum number of steps for the experiment is reached. If the aggregate grows, the growing time for that run is recorded.

Similar to shrinking probability verification, in order to calculate the growing probability, (1) number of single grows (n_{gr}), multiple grows (n_{mgr}) and remaining runs (n_{run}) are tracked for each model time step, (2) growing probability is calculated for each model step and (3) median of growing probabilities is calculated.

The remaining runs are the runs in which single or multiple grows did *not* occur yet and calculated as follows for model step t where $t > 0$:

$$n_{run}(t) = \begin{cases} n_{run}(0), & \text{if } t = 1 \\ n_{run}(t-1) - (n_{gr}(t-1) + n_{mgr}(t-1)), & \text{otherwise,} \end{cases}$$

where $n_{run}(t-1)$ is the number of remaining runs in the previous model step and $n_{run}(0)$ is the initial number of runs.

Similar to Equation 5.1, the growing probability for model step t is calculated as:

$$P_g(t) = \frac{n_{gr}(t) + 2n_{mgr}(t)}{n_{run}(t)} \, . \tag{5.3}$$

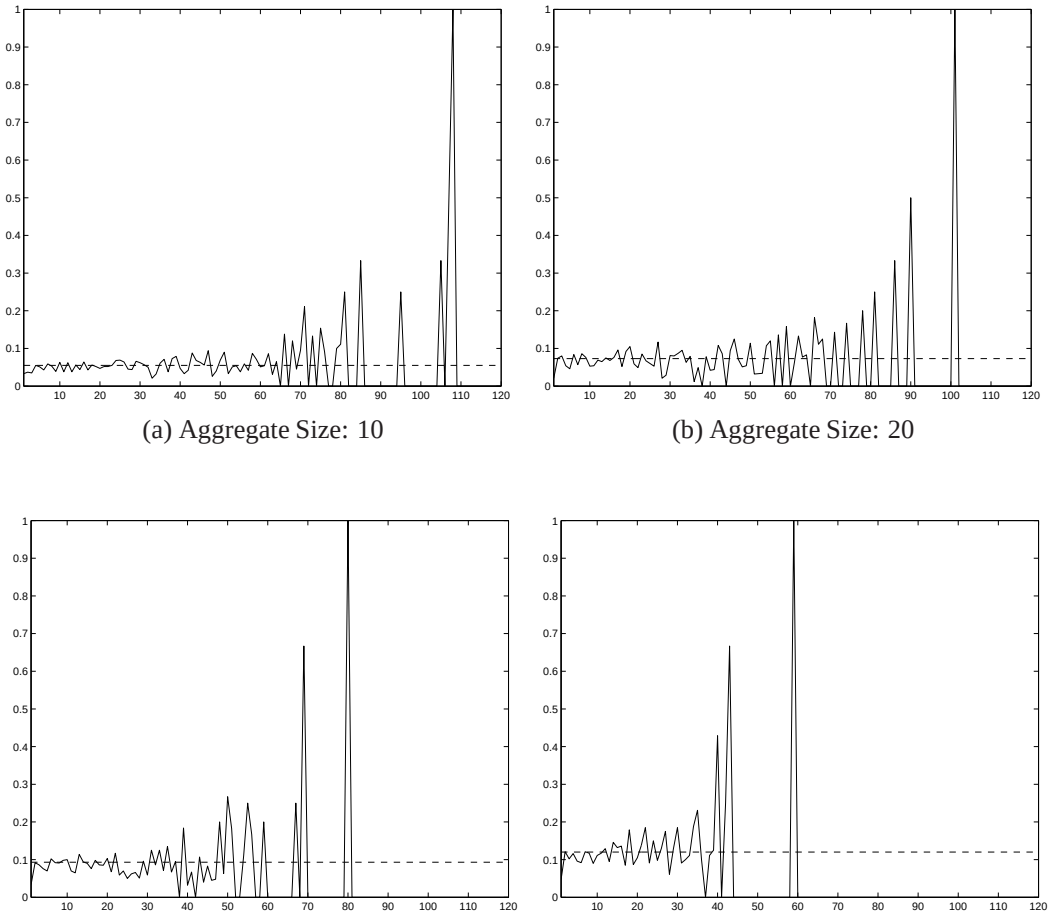
Similar to Equation 5.2, the median of shrinking probabilities for all steps is calculated as:

$$P_g = \frac{\sum_{t=t_{start}}^{t_{max}} P_g(t)}{\sum_{t=t_{start}}^{t_{max}} n_{run}(t)} \, , \tag{5.4}$$

where t_{max} is the maximum number of model steps for a run and t_{start} is the begining model step to calculate the median of growing probability. The value of t_{start} is selected as 60 for growing verification experiments in order to exclude unstable values of shrinking probabilities at the begining of the experiment.

Three different experiments are conducted to verify the growing probability. In first two experiments, two different searching robot densities are used. In both of the experiments, aggregate size is increased and the model's prediction for the growing probability is tested. In the third experiment, the aggregate size is kept constant and the model's prediction for the growing probability is tested for varying searching robot densities.

Figure 5.6 and Figure 5.7 plot the growing probability with respect to aggregate size in two



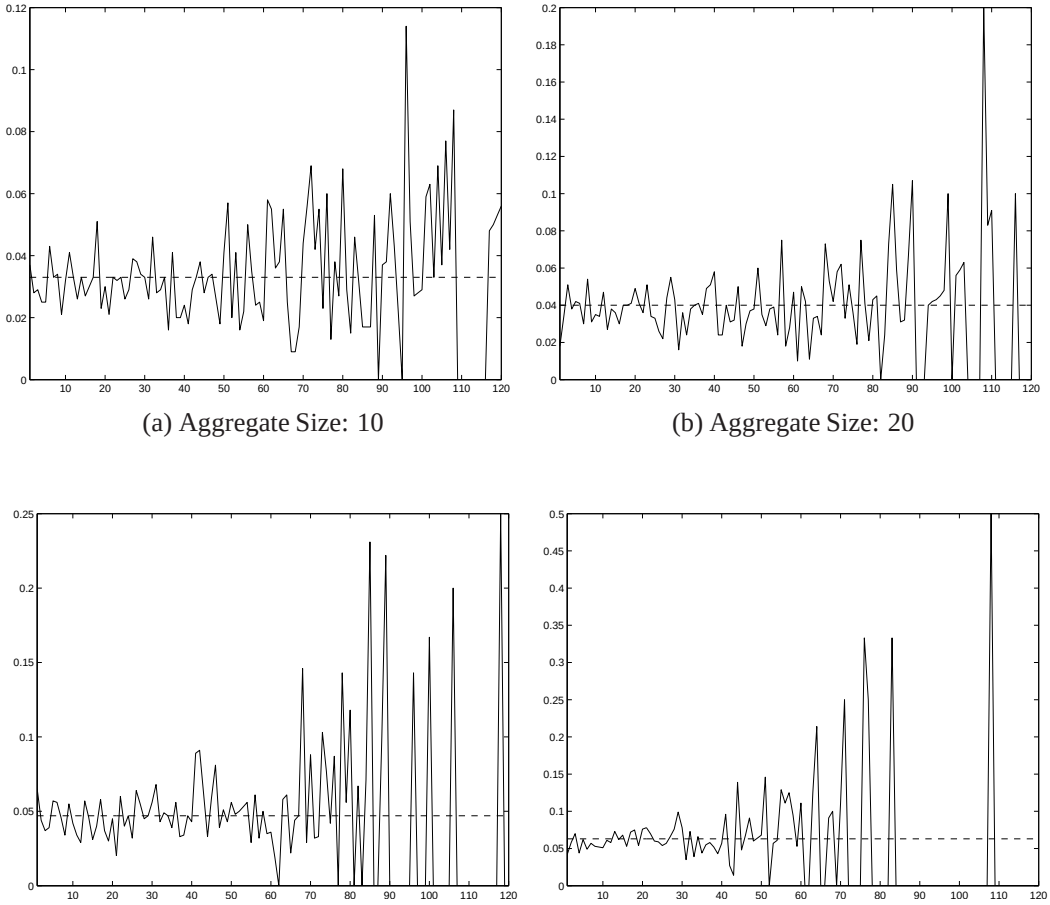
(a) Aggregate Size: 10 (b) Aggregate Size: 20

(c) Aggregate Size: 40 (d) Aggregate Size: 80

Figure 5.6: Median of growing probabilities for different model steps. Each plot shows growing probabilities for a different aggregate size: (a) 10, (b) 20, (c) 40 and (d) 80. While x axis shows the model step (t), y axis shows the growing probability (P_g). 1000 runs are performed for each experiment. The wall length (w) is 150 and the number of searching robots (n) is 150. For finding the median of growing probabilities, $t_{start} = 2$ and $t_{max} = 120$.

different searching robot densities. Aggregate size changes along x axis and y axis shows growing probabilities for each model step calculated using Equation 5.3. The dotted horizontal line represents the median of growing probabilities of all steps calculated using Equation 5.4. Fluctuation behavior that is observed in shrinking probability verification results exist in these results too and the reasons are the same.

In Figure 5.8, the median of growing probabilities for varying aggregate sizes and model predictions calculated using equation 5.4 are plotted. It can be seen that model and simulation estimations for growing probability match well. The difference between the simulation results and model predictions is less than %20 of simulation results even for the largest aggregate size (80) being tested. This difference is attributed to the *sensing range effect* discussed in Sec-



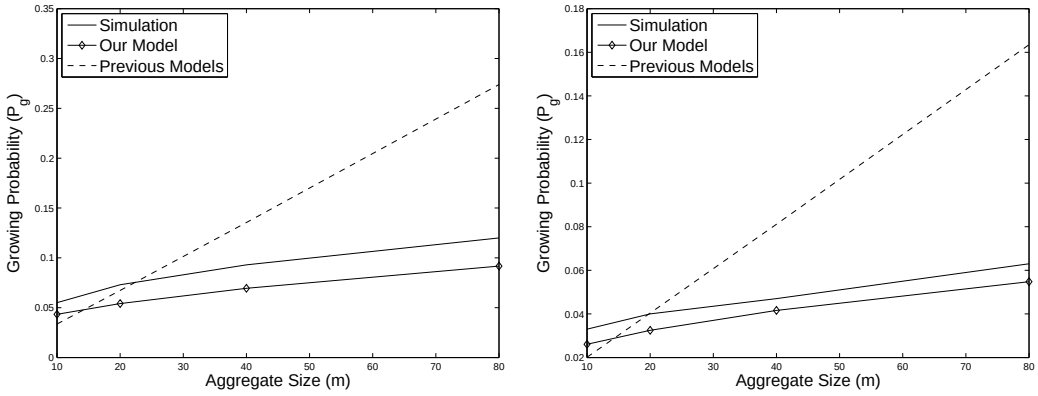
(a) Aggregate Size: 10

(b) Aggregate Size: 20

(c) Aggregate Size: 40

(d) Aggregate Size: 80

Figure 5.7: Median of growing probabilities for different model steps. Each plot shows growing probabilities for a different aggregate size: (a) 10, (b) 20, (c) 40 and (d) 80. While x axis shows the model step (t), y axis shows the growing probability (P_g). 1000 runs are performed for each experiment. The wall length (w) is 180 and the number of searching robots (n) is 130. For finding the median of growing probabilities, $t_{start} = 2$ and $t_{max} = 120$.



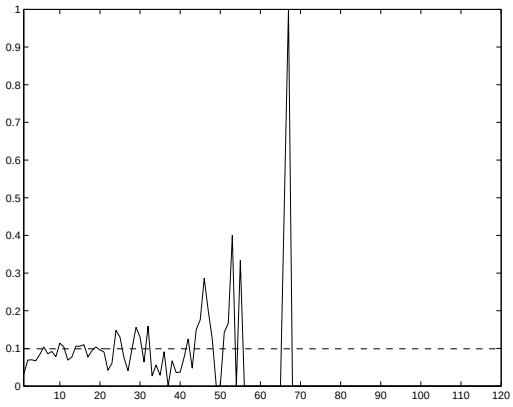
(a) (b)

Figure 5.8: Growing probability predictions for varying aggregate sizes. This is the summary of experimental results shown in Figure 5.6 and Figure 5.7. Simulation results, predictions of our model and predictions of previous models are shown. As the representative of previous models, the sweeping probability is used. Sweeping probability assumes that growing probability is linearly proportional with the aggregate size.

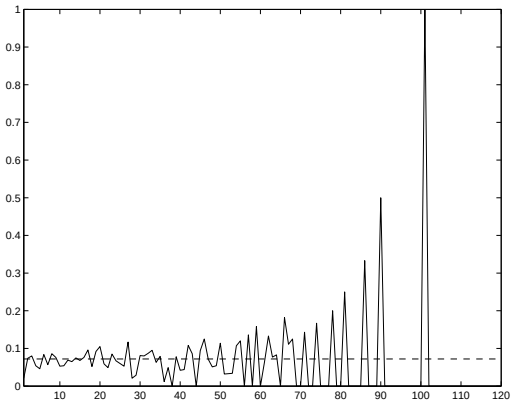
tion 5.1. It is also seen that this error gets smaller when the searching robot density is smaller and this supports our hypothesis about the sensing range effect. Additionally, it can be seen that the predictions of older models (sweeping probability is selected as the representative of older models) for growing probability is far from realistic for larger aggregates.

Figure 5.9 presents growing probability for varying searching robot densities for an aggregate of size 20. Model step is shown in x axis and y axis shows growing probabilities for each model step calculated using Equation 5.3. Dotted horizontal lines represent the median of growing probabilities of all steps calculated using Equation 5.4.

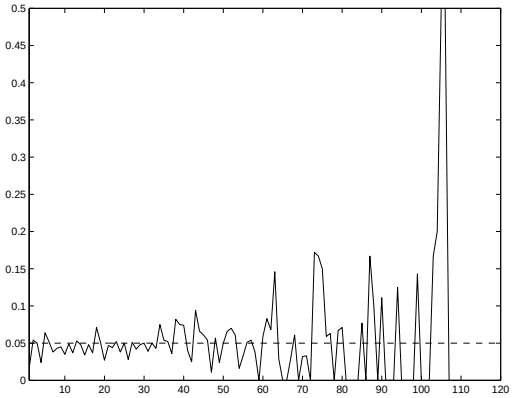
Figure 5.10 plots the median of growing probabilities for varying searching robot densities. This is a summary of experimental results shown on Figure 5.9. It is seen that (1) the difference between simulation results and model predictions is less than %20 of simulation results, (2) this difference decreases when the searching robot density is decreased and (3) older models (sweeping probability is used) predict the growing probability well for this aggregate size.



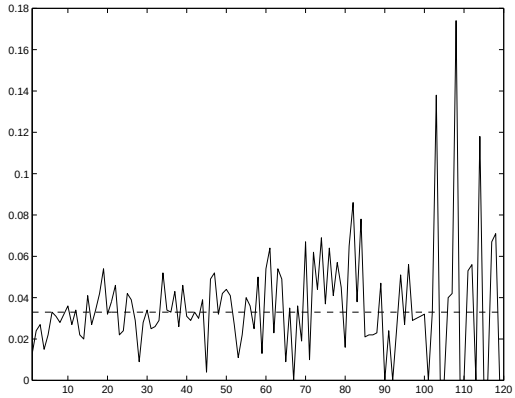
(a) Searching robot density: 5d/4



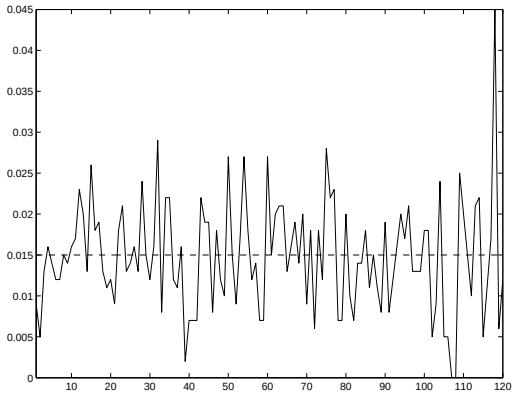
(b) Searching robot density: 4d/4 = d



(c) Searching robot density: 3d/4



(d) Searching robot density: 2d/4 = d/2



(e) Searching robot density: d/4

Figure 5.9: Median of growing probabilities for different model steps. Each plot shows growing probabilities for a different searching robot density: (a) $5d/4$, (b) $4d/4 = d$, (c) $3d/4$, (d) $2d/4 = d/2$ and (d) $d/4$. While x axis shows the model step (t), y axis shows the growing probability (P_g). 1000 runs are performed for each experiment. For the searching robot density d, the wall length (w) is 150 and the number of searching robots (n) is 150. For other searching robot densities, the wall length is modified while the number of searching robot is kept constant. For example, for the searching robot density $d/4$, the wall length (w) is 300 and the number of searching robots (n) is 150. For finding the median of growing probabilities, $t_{start} = 2$ and $t_{max} = 120$.

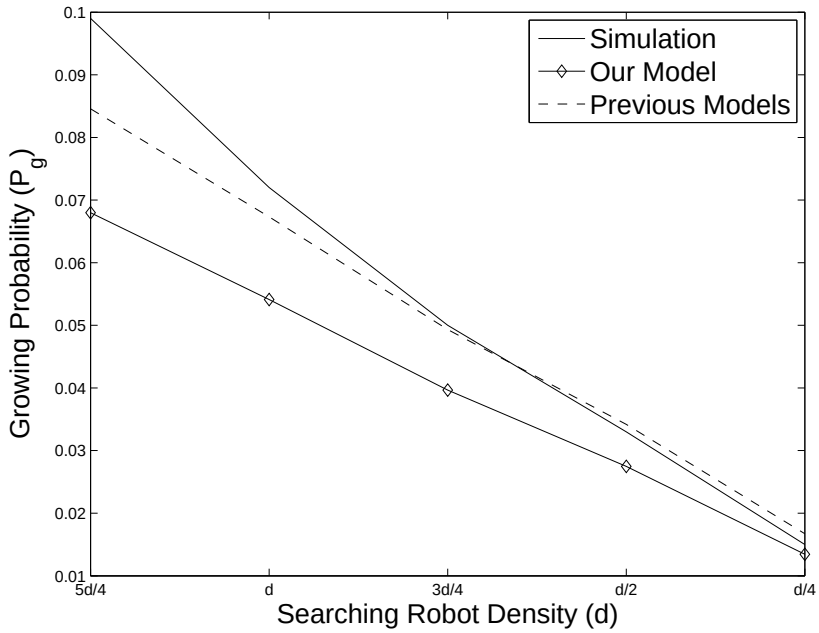


Figure 5.10: Growing probability predictions for varying searching robot densities. This is the summary of experimental results shown in Figure 5.9. Simulation results, predictions of our model and predictions of previous models are shown. As the representative of previous models, the sweeping probability is used.

CHAPTER 6

Experimental Results

In the previous chapter, the probability formulas for basic aggregation events are verified experimentally using Stage simulations. However, a probabilistic model is especially useful if it can be used to predict the results of generic simulation experiments. In this chapter, we used our model to predict results of various Stage-based simulation experiments.

Two different methods are used to predict simulation results: *microscopic model execution* and *steady state analysis*. Microscopic model execution is a common way for predicting results of simulation experiments for aggregation [35] [39] [45] [46]. In this method, the model is iterated similar to simulation experiments. However, unlike simulation experiments, iterations are performed probabilistically using the probabilities corresponding to all possible events that can happen in the current model step. The details of micrsocopic model execution are described in Section 6.1.

Steady state analysis is performed by finding an equation in terms of experiment parameters which describes the steady state and solving this equation for the parameter being interested. However, finding such an equation is only possible for simple cases.

While predicting results of simulation experiments, another issue is finding the steady state time step and performance of the experiment. In this chapter, steady state time step and performance of experimentars are found automatically using algorithms described in Section 3.5. Finally, Section 6.2 presents details and results of experiments that predict various simulation results.

6.1 Microscopic Model Execution

The function used to execute *one run* of the microscopic model is shown in Algorithm 19.

Algorithm 19: The function for executing *one run* of the microscopic model.

Data: Swarm state vector (x), number of searching robots (y), arena wall length (w), sensing range (s), searching robot speed (v_s), model time step (t), leaving probability (p) and maximum number of time steps (t_{max}).

Result: Updates swarm state vector (x).

`step` = 0;

while `step` < t_{max} **do**

  d = `CalcDensity(`x, y, w`);`

  `CheckAggrEvents(`x, y, d, s, v_s, p, t`);`

  `step` = `step` +1;

end

During the run, the model is iterated until the current model time step is greater than or equal to the maximum number of model time steps in a run (t_{max}). In each iteration, the searching robot density is calculated using the **CalcDensity** function and aggregation events (creation, growing etc.) are checked using the **CheckAggrEvents** function.

The algorithm of **CalcDensity** function is previously shown in Algorithm 12. The default algorithm for **CheckAggrEvents** function is shown on Algorithm 20.

Algorithm 20: Default algorithm for the **CheckAggrEvents** function.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

`CheckCreation(`x, y, d, s, v_s, t`);`

`CheckGrowing(`x, y, d, s, v_s, t`);`

`CheckShrinking(`x, y, p, t`);`

`CheckDissipation(`x, y, p, t`);`

The **CheckAggrEvents** function uses four different functions to check occurence of aggrega-

tion events: **CheckCreation**, **CheckGrowing**, **CheckShrinking** and **CheckDissipation**.

In **CheckCreation** function, a uniform random number between 0 and 1 is generated using *rand* function and this random number is compared against the creation probability calculated using **CalcPcreate** function. If the random number is lower than the creation probability, a new aggregate is created by calling **CreateNewAggregate** function which finds an empty slot in swarm state vector and fills it with the size of the new aggregate. The algorithms for **CalcPcreate** and **CreateNewAggregate** functions are previously shown in Algorithm 13 and Algorithm 1 subsequently.

Algorithm 21: The **CheckCreation** function.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s) and model time step (t).

Result: Updates swarm state vector (x).

p_c = `CalcPcreate`(y, d, s, v_s, t);

tmp = `rand()`;

if $p_c < tmp$ **then**

    `CreateNewAggregate`(x);

end

Similar algorithms are performed for **CheckGrowing**, **CheckShrinking** and **CheckDissipation**: the probability for corresponding event is calculated, the calculated probability is compared against a uniform random number between 0 and 1 and the swarm state vector is updated based on the result of this comparison. Algorithms for **CheckGrowing**, **CheckShrinking** and **CheckDissipation** functions are shown in Algorithm 14, Algorithm 17 and Algorithm 18 subsequently.

Algorithm 22: The **CheckGrowing** function.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

```
for i ← 0 to n/2 do
    if x_i ≠ 0 then
        p_g = CalcPgrow(x_i, d, s, v_s, t);
        tmp = rand();
        if tmp < p_g then
            x_i = x_i + 1;
            y = y − 1;
        end
    end
end
```

Algorithm 23: The **CheckShrinking** function.

Data: Swarm state vector (x), number of searching robots (y), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

```
for i ← 0 to n/2 do
    if x_i > 2 then
        tmp = rand();
        p_s = CalcPshrink(x_i);
        if tmp < p_s then
            x_i = x_i − 1;
            y = y + 1;
        end
    end
end
```

Algorithm 24: The **CheckDissipation** function.

Data: Swarm state vector (x), number of searching robots (y), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

```
1  for i ← 0 to n/2 do
2      if x_i = 2 then
3          tmp = rand();
4          p_d = CalcPdissipate();
5          if tmp < p_d then
6              x_i = 0;
7              y = y + 2;
8          end
9      end
10 end
```

6.2 Experiments

Experiments presented in this section are designed to analyze effects of aggregation events on aggregation dynamics by isolating aggregation events in a systematic way. Model predictions for the steady state performance of experiments are presented alongside with results of simulation experiments. This allows us to verify our model further especially in the time dimension.

Five different experiments and their results are presented in five subsequent sections below. In the first experiment, all aggregation events except the *creation* event is disabled. It is shown that the change of number of aggregates during the experiment for three different searching robot densities matches with model's results.

In the subsequent experiment, we disable all aggregation events except *creation* and *dissipation* events. The change of number of aggregates is analyzed when leaving probability of robots is varied.

In the third experiment, both *growing* and *shrinking* events are enabled while remaining events are disabled. In this experiment, the change of the size of the aggregate that exist in the

environment during the experiment run is compared with the model's prediction.

In the fourth experiment, we enabled both *creation* and *growing* events while disabling *shrinking* and *dissipation* events. In this experiment, we analyzed the chnage of number of aggregates and aggregate sizes while chaning the searching robot density and leaving probability of robots.

In the fifth experiment, all aggregation events are enabled. The model's prediction for the largest aggregate size and the number of aggregates is tested while leaving probability is varied. Table 6.1 summarizes the experiments performed in this chapter.

Table 6.1: Summary of experiments performed in Chapter 6. First column shows title and section of the experiments. Following four columns specify which aggregation events (C: Creation, G: Growing, S: Shrinking, D: Dissipation) are enabled in the experiments performed in this section. Aggregation events being enabled contains "+" sign in its corresponding column. Remaining columns show whether microscopic model (MM), simplified micrscopic model (SMM) or steady state analysis (SSA) is used (denoted with + in corresponding column) or *not* subsequently.

Title (Section)	**C**	**G**	**S**	**D**	**MM**	**SMM**	**SSA**
Creation (6.2.1)	+	-	-	-	+	+	+
Creation vs. Dissipation (6.2.2)	+	-	-	+	-	+	-
Growing vs. Shrinking (6.2.3)	-	+	+	-	-	+	+
Growing vs. Growing (6.2.4)	+	+	-	-	-	+	+
Creation, Growing and Shrinking (6.2.5)	+	+	+	+	-	+	+

6.2.1 Creation Experiment

In this experiment, searching robots with varying densities are placed into the enviornment. The searching robots are allowed to create new aggregates when they detect each other but growing an existing aggregate and leaving from an aggregate are *disabled*. In simulation experiments, this is achieved by modifying the default controller so that when searching robots detect other robots, they check the size of the aggregate being formed and if the aggregate has more than two robots (including the searching robot itself), it leaves the aggregate.

In order to implement this experiment as a microscopic model, growing, shrinking and dissipation events should be disabled while creation event is still enabled. This can be achieved by replacing the default **CheckAggrEvents** function with the one provided in Algorithm 25.

Algorithm 25: Modified **CheckAggrEvents** function for the **Creation Experiment**.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

`CheckCreation(`x, y, d, s, v_s, t`);`

In addition to simulation and microscopic model, a simplified microscopic model is used for this experiment. In this simplified microscopic model, the number of aggregates is directly increased by two times of creation probability at each time step. Algorithm 26 shows the algorithm of the simplified microscopic model.

Algorithm 26: The algorithm for the execution of *one run* of the *simplified* microscopic model for the **Creation Experiment**.

Data: Number of searching robots (y), arena wall length (w), sensing range (s), searching robot speed (v_s), model time step (t) and maximum number of time steps (t_{max}).

Result: Updates number of searching robots (y) and number of aggregates (noa).

`step` = 0;

`noa` = 0;

while `step` < t_{max} **do**

  d = `CalcDensity(`x, y, w`);`

  if $y \geq 2$ **then**

    p_c = `CalcPcreate(`y, d, s, v_s, t`);`

    `noa` = `noa` + p_c;

    $y = y - 2p_c$;

  end

  `step` = `step` +1;

end

Figure 6.1 presents time evolution of number of aggregates (NOA) for varying initial searching robot densities. Three conclusions can be reached from these experiment results:

1. **The simulation results and model predictions match well.**

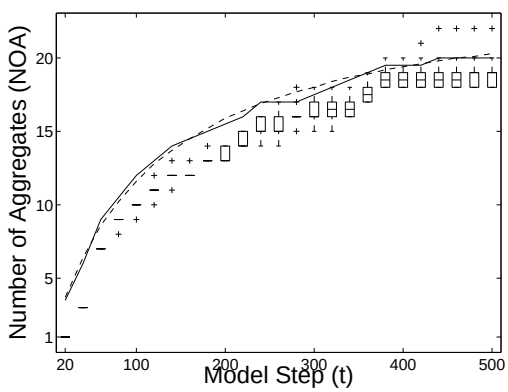
2. **The experiments seem to reach to the steady state earlier than expected.** For example, the steady states seem to be around 20, 43 and 68 when plots are evaluated manually. However. the actual steady state values should be 25, 50 and 75. When time passes, few searching robots left in the environment, the creation probability becomes too slow and NOA changes occurs very rarely. This is why the experiment seem to reach to the steady state earlier than expected.

3. **The speed of reaching steady state is faster for higher densities compared to lower ones.** Creation probility is higher for higher densities. Furthermore, the creation probability is proportional with the square of number of searching robots. This allows the experiments with higher densities move faster toward the steady state.

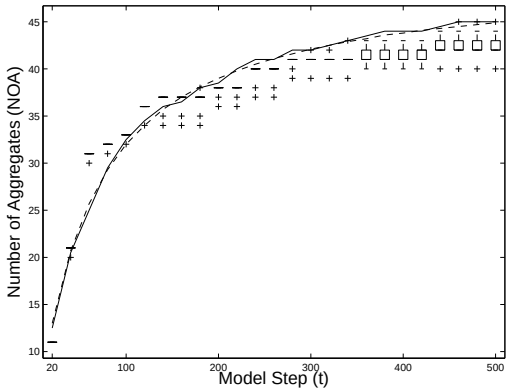
4. **The time to reach the steady state is close to each other for all initial searching robot densities even though the highest density is three times of the lowest density.** This result can be explained with the nonlinear relation between the number of searching robots and the creation probability. Due to the creation probability being proportional with the square of the number of searching robots, swarm states with higher densities do *not* spend much time to aggregate creation and most of the time is being spent on the creation of aggregates in lower searching robot densities.

Figure 6.2 shows NOA and NS for varying initial searching robot densities (d) at the steady state for the same experiments. The values at the last model step are taken as the steady state values for both simulation and model experiments. At the steady state, it is seen that (a) NOA is linearly proportional with d and (2) NS does *not* depend on d.

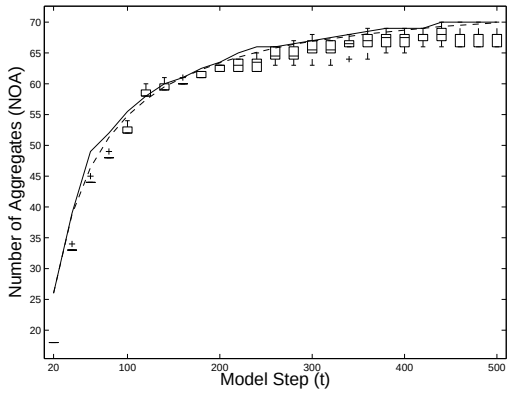
The second observation is interesting since it shows that NS at the steady state is more like a parameter of the environment size instead of searching robot density.



(a) 50 searching robots (d)



(b) 100 searching robots (2d)



(c) 150 searching robots (3d)

Figure 6.1: Time evolutions of number of aggregates (NOA) for varying initial searching robot densities in an environment with default wall length. The density is varied by changing the initial number of searching robots and 50 searching robots are used for density *d*. The plot shows simulation, microscopic model and simplified microscopic model results. 10 runs are made for simulation and microscopic model. Simulation results are represented as boxes. The line in the middle of boxes are the median, and whiskers (if exists) are minimum and maximum values for NOA. Plus sign represents outliers. *Flat* and *dotted* lines connect median of *microscopic model* and *simplified microscopic model* predictions respectively.

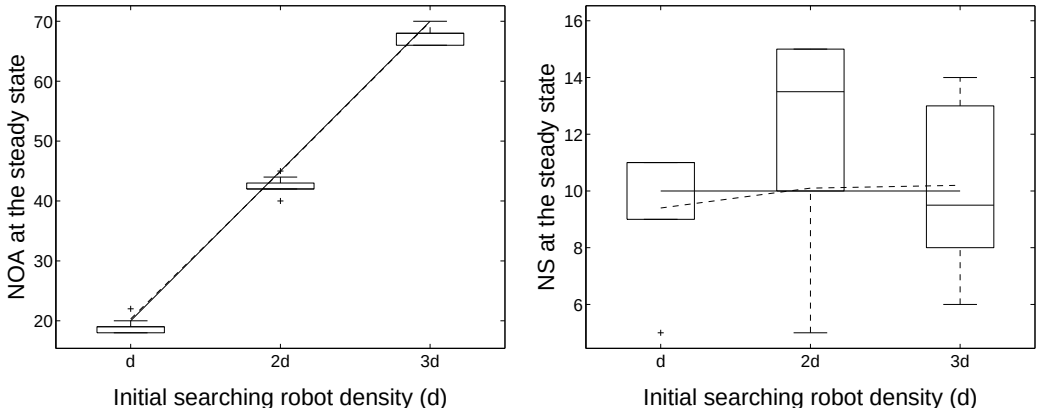


Figure 6.2: (a) Number of aggregates (NOA) and (b) number of searching robots (NS) at the steady state for the experiments whose time evolutions are presented on Figure 6.1. The median of values at the last model step (500) are taken as the steady state values for both simulation and model experiments. Simulation results are represented as boxes. *Flat* and *dotted* lines connect median of *microscopic model* and *simplified microscopic model* predictions respectively.

6.2.2 Creation vs. Dissipation Experiment

This experiment is the same as the experiment in the previous section (creation experiment) except that waiting robots can leave their aggregates. In other words, *creation* and *dissipation* events are allowed in this experiment but searching robots can *not* grow existing aggregates as in the previous experiment. The experiment is conducted in simulation by modifying the controller used in the creation experiment in the previous section so that waiting robots can leave their aggregates as in the default controller.

In order to conduct this experiment as a microscopic model, default **CheckAggrEvents** function is modified so that only creation and dissipation events are enabled as shown in Algorithm 27.

Algorithm 27: Modified **CheckAggrEvents** function for the **Creation vs. Dissipation Experiment**.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

`CheckCreation(`x, y, d, s, v_s, t`);`

`CheckDissipation(`x, y, p, t`);`

A simplified microscopic model is developed for this experiment too. In this simplified microscopic model, the number of aggregates is increased by the amount of creation probability and decreased by the amount of dissipation probability at each time step. Algorithm 28 shows the algorithm of the simplified microscopic model for this experiment.

Algorithm 28: The algorithm for the execution of *one run* of the *simplified* microscopic model for the **Creation vs. Dissipation Experiment**.

Data: Number of searching robots (y), arena wall length (w), sensing range (s), searching robot speed (v_s), model time step (t) and maximum number of time steps (t_{max}).

Result: Updates number of searching robots (y) and number of aggregates (noa).

```
step = 0;
noa = 0;
while step < t_max do
    d = CalcDensity(x, y, w);
    if y ≥ 2 then
        p_c = CalcPcreate(y, d, s, v_s, t);
        noa = noa + p_c;
        y = y − 2p_c;
    end
    if noa > 0 then
        p_d = CalcPdissipate();
        noa = noa − ( p_d × noa );
        y = y + 2 ( p_d × noa );
    end
    step = step +1;
end
```

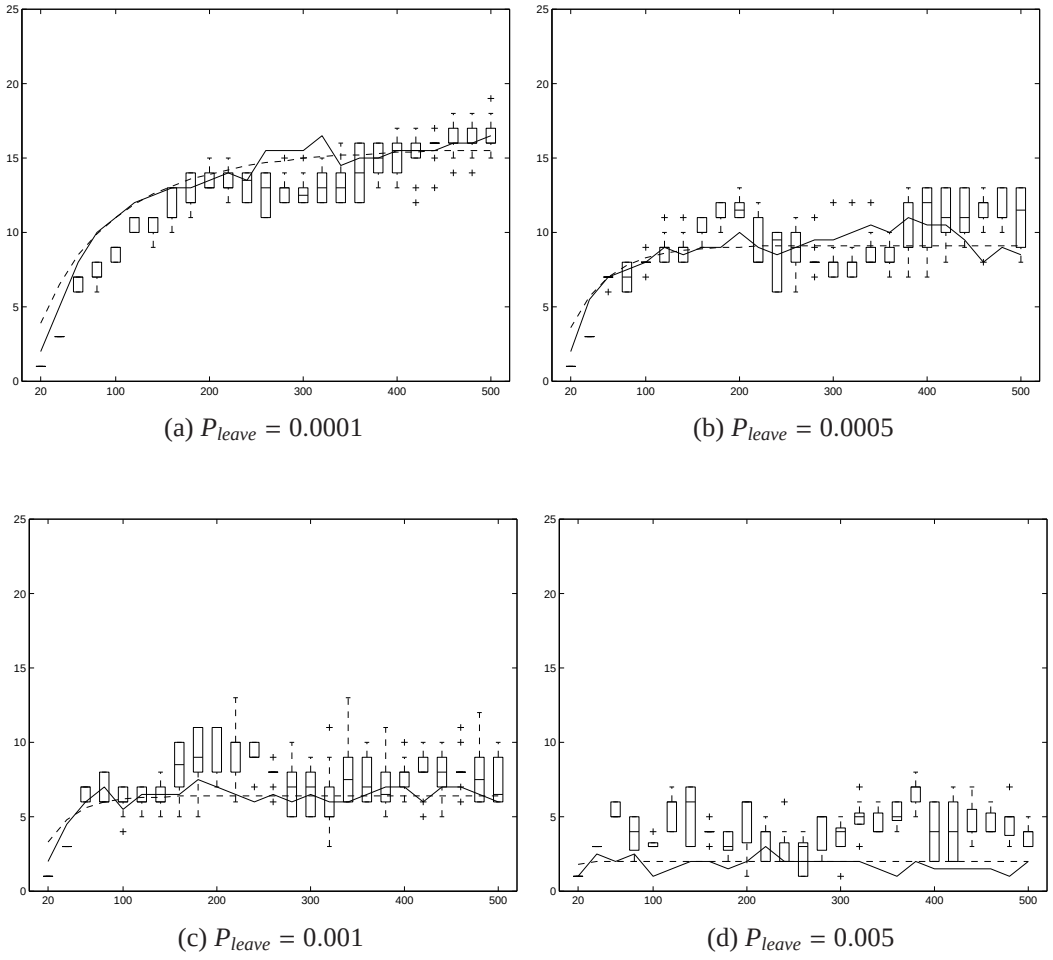
In this experiment, both leaving probability (p) and number of searching robots are varied in order to analyze the effects of p and searching robot density (d) on the time evolution of number of aggregates (NOA). Figure 6.3, Figure 6.4 and Figure 6.5 show these results for 50, 100 and 150 robots respectively. By investigating the plots, it can be seen that following changes happen for the steady state when p is increased:

1. **The time required to reach the steady state decreases.** This can be better seen in Figure 6.6-a which plots the steady state time for the experiment with searching robot density $3d$. The result can be explained as follows. There are two components of the steady state time in this experiment: creation time and dissipation time. Decreasing any of these will decrease the steady state time. Increasing leaving probability decreases the dissipiation time and this decreases the steady state time consequently.

2. **The error between the model and simulation results increases.** This can be better seen on Figure 6.6-b which plots NOA at the steady state for the experiment with searching robot density $3d$. The result can be attributed to the *leave timeout*. In the Stage simulation experiments, leaving robots leave their aggregates after a certain duration. The maximum value for this duration is the leave timeout. However, this delay is *not* considered in the probabilistic model and leaving robots switch to random walk state immediately. For Stage simulation experiments, these leaving robots are still in the sensing range of their partners in the aggregate they left and they are counted as aggregates. However, they are considered as searching robots in the probabilistic model. This causes reporting higher number of aggregates in the simulation compared to the model. For example, when the leaving probability is 0.005, the average number of leaving robots during the leave timeout (800 simulation steps) is 5. Depending on which robots left [1], this can cause 3-5 dissipations in the environment. Since these leaving robots are still considered as part of an aggregate, simulation experiments will report 3-5 more number of aggregates as shown in Figure 6.3-d.

3. **The number of aggregates decreases.** This observation can be explained better when the steady state of this experiment is analyzed as follows. There are two aggregation events in this experiment: aggregate creation and aggregate dissipation. These two events affect the performance of aggregation in opposite directions. For example, while aggregate creation increases the number of aggregates and decreases the number of searching robots, aggregate dissipation decreases the number of aggregates and increases the number of searching robots. Therefore, steady state occurs when these two events have equal probabilities.

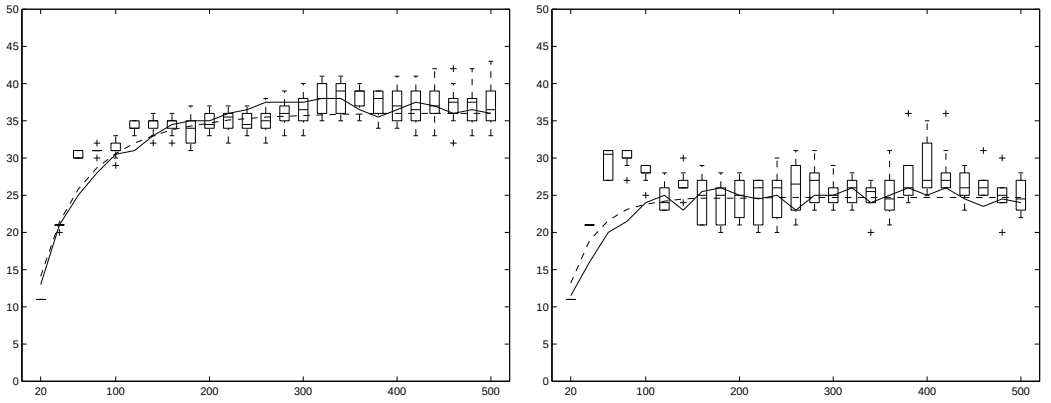
[1] If two robots leave from the same aggregate, one aggregate will be dissipated. If two robots are not from the same aggregate, two aggregates will be dissipated.



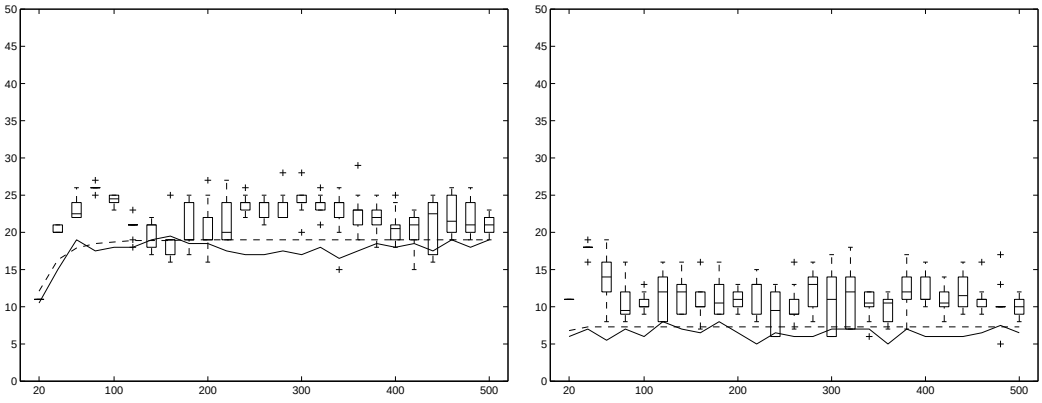
(a) $P_{leave} = 0.0001$ (b) $P_{leave} = 0.0005$

(c) $P_{leave} = 0.001$ (d) $P_{leave} = 0.005$

Figure 6.3: Time evolutions of NOA for varying leaving probabilities with 50 robots in an environment with default wall length. The plot shows simulation, microscopic model and simplified microscopic model results. 10 runs are made for simulation and microscopic model. Simulation results are represented as boxes. *Flat* and *dotted* lines connect median of *microscopic model* and *simplified microscopic model* predictions subsequently.

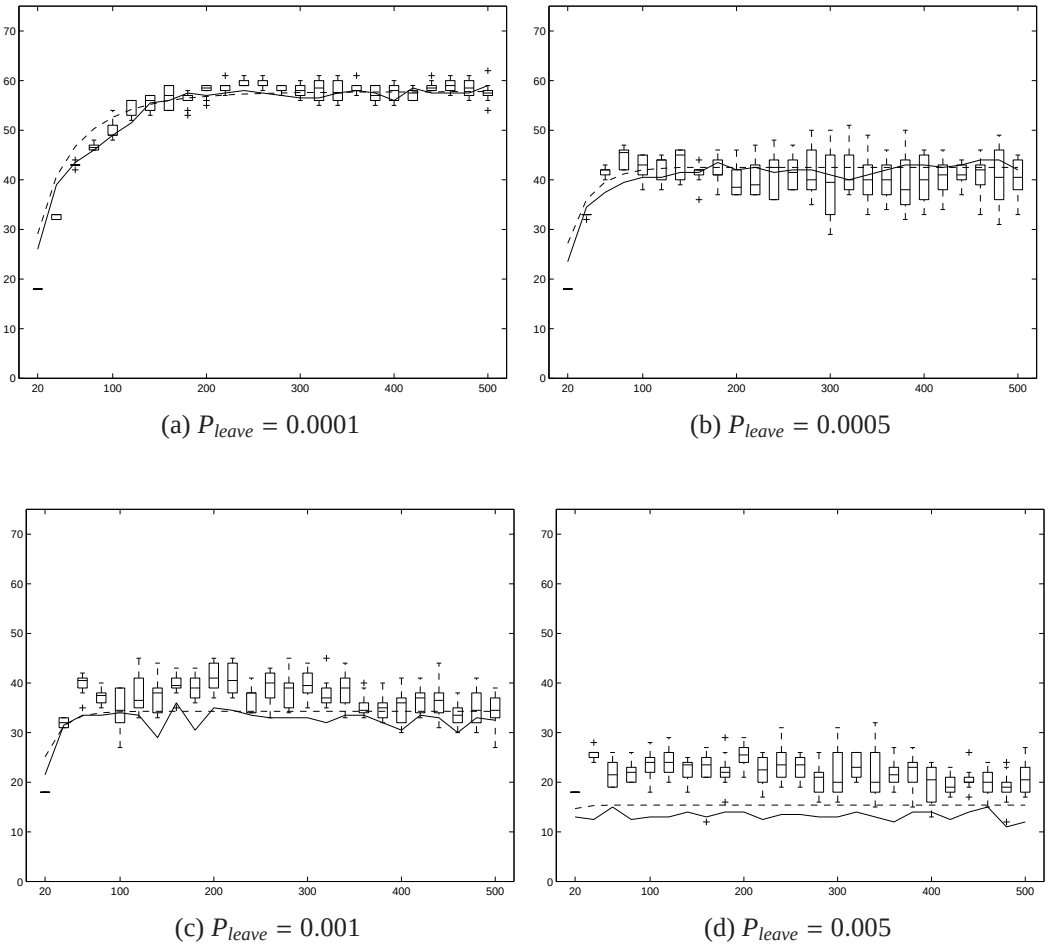


(a) P_{leave} = 0.0001 (b) P_{leave} = 0.0005



(c) P_{leave} = 0.001 (d) P_{leave} = 0.005

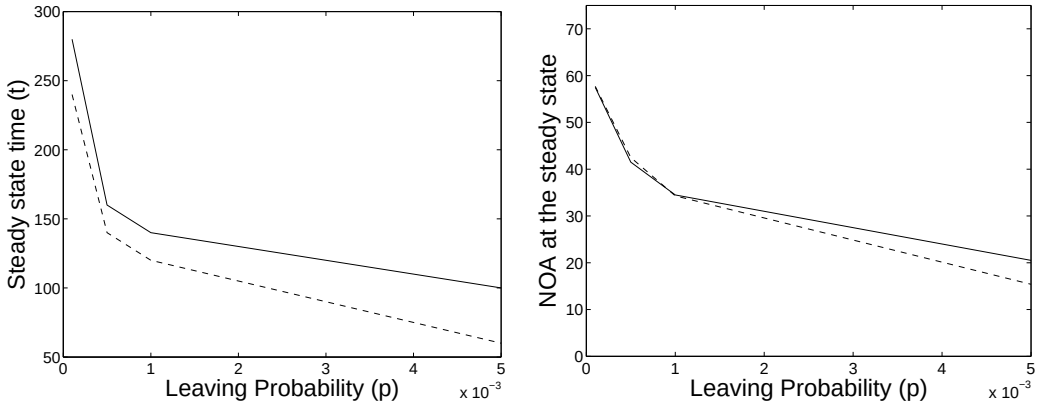
Figure 6.4: Time evolutions of NOA for varying leaving probabilities with 100 robots in an environment with default wall length. The plot shows simulation, microscopic model and simplified microscopic model results. 10 runs are made for simulation and microscopic model. Simulation results are represented as boxes. *Flat* and *dotted* lines connect median of *microscopic model* and *simplified microscopic model* predictions subsequently.



(a) P_{leave} = 0.0001 (b) P_{leave} = 0.0005

(c) P_{leave} = 0.001 (d) P_{leave} = 0.005

Figure 6.5: Time evolutions of NOA for varying leaving probabilities with 150 robots in an environment with default wall length. The plot shows simulation, microscopic model and simplified microscopic model results. 10 runs are made for simulation and microscopic model. Simulation results are represented as boxes. *Flat* and *dotted* lines connect median of *microscopic model* and *simplified microscopic model* predictions subsequently.



(a) (b)

Figure 6.6: (a) Steady state time and (b) number of aggregates (NOA) at the steady state of the experiment with 150 robots in an environment with default wall length. *Flat* and *dotted* lines connect median of *simulation* and *simplified microscopic model* results subsequently.

If we assume that the number of aggregates at the steady state is m, it can be derived by equating the creation probability with the total dissipiation probability of all aggregates as:

$$P_c = mP_d \, ,$$

where P_c is the creation probability and P_d is the dissipation probability of a 2-aggregate. P_c and P_d can be expanded using Equation 4.5 and 4.12 subsequently as:

$$\frac{\sqrt{2}sv_s t y^2}{A_f} = 2mp \, ,$$

where y is the number of searching robots and A_f is the free area. For the steady state, number of searching equals to "$n-2m$" and free area equals to "$w^2 - m(\pi R_2^2)$". When these are replaced in the above equation, we obtain:

$$\frac{\sqrt{2}sv_s t(n-2m)^2}{w^2 - m(\pi R_2^2)} = 2mp \, .$$

R_2 for a 2-aggregate is $\frac{s(a\sqrt{2}+3)}{4}$. When R_2 is replaced with this, we obtain:

$$\frac{\sqrt{2}sv_s t(n-2m)^2}{w^2 - m(\pi(\frac{s(a\sqrt{m}+3)}{4})^2)} = 2mp \, .$$

When this equation is solved with respect to the variable m using MATLAB Symbolic Math Toolbox, Equation 6.1:

$$m = \frac{\sqrt{2}(\ \text{Eq6.2} + \ \text{Eq6.3}\ )}{128 s t v_s}\ , \tag{6.1}$$

which consists of two parts: Eq6.2 and Eq6.3, is obtained. These parts and their sub-parts are written below.

$$16pw^2 + 32\sqrt{2}nstv_s - 6\pi\sqrt{2}aps^2 - 6\pi a^2 ps^2 - 9\pi ps^2 \tag{6.2}$$

$$6\sqrt{2}\pi as^2\ \text{Eq6.4} + 2\pi a^2 s^2\ \text{Eq6.4} + 9\pi s^2\ \text{Eq6.4} - 16w^2\ \text{Eq6.4} \tag{6.3}$$

$$\sqrt{\frac{\text{Eq6.5} + \ \text{Eq6.6} + \ \text{Eq6.7}}{\text{Eq6.8} + \ \text{Eq6.9}}} \tag{6.4}$$

$$p(4pa^4\pi^2 s^4 + 24\sqrt{2}pa^3\pi^2 s^4 + 108pa^2\pi^2 s^4 - 128\sqrt{2}ntv_s a^2\pi s^3 - 64pa^2\pi s^2 w^2 \tag{6.5}$$

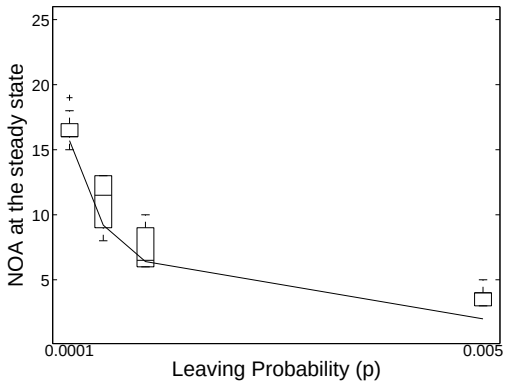
$$108\sqrt{2}pa\pi^2 s^4 - 768ntv_s a\pi s^3 - 192\sqrt{2}pa\pi s^2 w^2 + 81p\pi^2 s^4 \tag{6.6}$$

$$-576\sqrt{2}ntv_s\pi s^3 - 288p\pi s^2 w^2 + 1024\sqrt{2}ntv_s s w^2 + 256pw^4 \tag{6.7}$$

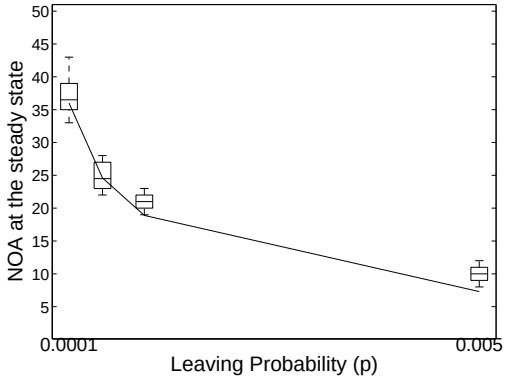
$$4a^4\pi^2 s^4 + 24\sqrt{2}a^3\pi^2 s^4 + 108a^2\pi^2 s^4 - 64a^2\pi s^2 w^2 \tag{6.8}$$

$$108\sqrt{2}a\pi^2 s^4 - 192\sqrt{2}a\pi s^2 w^2 + 81\pi^2 s^4 - 288\pi s^2 w^2 + 256w^4 \tag{6.9}$$

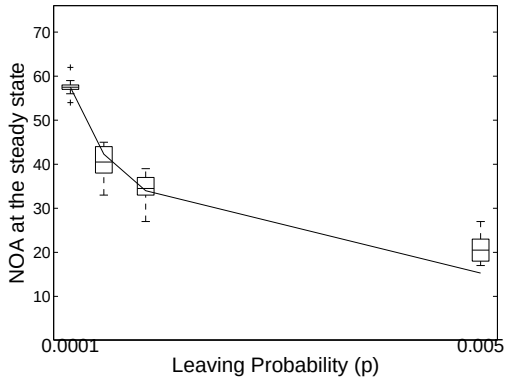
Figure 6.7 shows NOA predictions at the steady state calculated using Equation 6.1. Predictions are plotted on top of simulation results for varying leaving probabilities and three



(a) 50 robots [d]

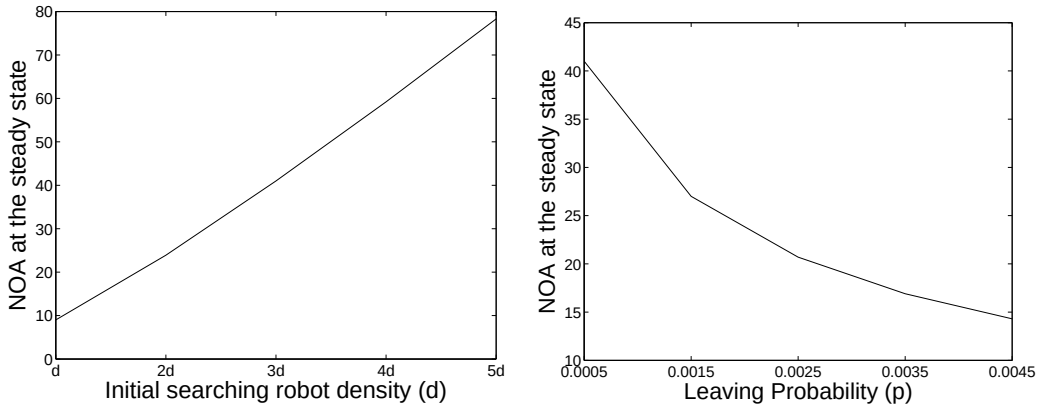


(b) 100 robots [2d]



(c) 150 robots [3d]

Figure 6.7: Number of aggregates (NOA) for varying leaving probabilities and three different searcing robot densities. Density *d* corresponds to 50 searching robots in an environment with the default wall length. Both simulation results steady state predictions, that are calculated with Equation 6.1, are shown. Simulation results are presented as boxes and the predictions are shown as a solid line.



(a) (b)

Figure 6.8: Predictions of number of aggregates (NOA) at the steady state for varying (a) initial searching robot densities and (b) leaving probabilities. Predictions are calculated using Equation 6.1. Density d corresponds to 50 searching robots in an environment with the default wall length. For (a), leaving probability (p) is selected as $p = 0.001$. For (b), initial searching robot density $3d$ is used.

different initial searching robot densities. It is seen that simulation results and steady state predictions match well.

Figure 6.8-a shows predictions of NOA at the steady state calculated using Equation 6.1 for varying initial searching robot densities. It is seen that NOA at the steady state is linearly proportional with the initial searching robot density.

Figure 6.8-b plots predictions of NOA at the steady state for varying leaving probabilities. It is seen that NOA at the steady state is inversely and nonlinearly proportional with the leaving probability

Now, the third observation at the begining of this section ("When leaving probability is increased, the number of aggregates decreases.") can be easily seen on Figure 6.8-b. The reason of this result is that increasing leaving probability makes aggregates easier to dissipate.

6.2.3 Growing vs. Shrinking Experiment

In this experiment, a waiting robot and multiple searching robots are placed in an environment. While searching robots are randomly placed, waiting robot is placed at the center of the environment and is prevented to switch to leave state. Searching robots are prevented to form aggregates with each other but are allowed to form an aggregate with the waiting robot

at the center or to join to the aggregate being formed with this waiting robot. In this way, *growing* and *shrinking* events are allowed but *creation* and *dissipation* events are disabled in this experiment.

In order to implement this experiment as a microscopic model two things are performed. First, the swarm state vector, that is given as a parameter to the model execution function presented in Algorithm 19, is initiated with one aggregate of size 1 in addition to the searching robots. Secondly, the default **CheckAggrEvents** function is modified as shown in Algorithm 29.

Algorithm 29: Modified **CheckAggrEvents** function for the **Growing vs. Shrinking Experiment**.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

`CheckGrowing(`x, y, d, s, v_s, t`);`

`CheckShrinking(`x, y, p, t`);`

As in the **Creation vs. Dissipation Experiment**, a steady state analysis is performed by deriving an equation describing the steady state. In this experiment, there are two aggregation events: growing and shrinking of an aggregate. These two events affect the performance in opposite directions. For example, while aggregate growing increases the aggregate size and decreases the number of searching robots, aggregate shrinking decreases the aggregate size and increases the number of searching robots. Therefore, steady state occurs when these two events have equal probabilities.

The derivation starts by equating the growing probability with the shrinking probability of the aggregate in the environment as:

$$P_g = P_s \, ,$$

where P_g is the growing probability and P_s is the shrinking probability of the aggregate in the environment. When the aggregate size at the steady state is assumed to be m, P_g and P_s can be extended using Equation 4.6 and Equation 4.11 subsequently as:

$$k_3\pi[R_3(m)^2 - R_2(m)^2]d = \pi(a\sqrt{m} - 1)p \ ,$$

where d is the searching robot density in the environment and for this experiment it is calculated as:

$$d = \frac{n - m}{w^2 - \pi R_2^2} \ ,$$

where $n - m$ is the number of searching robots and $w^2 - \pi R_2^2$ is the free area. When d is expanded, we obtain:

$$\frac{k_3\pi(R_3^2 - R_2^2)(n - m)}{w^2 - \pi R_2^2} = \pi(a\sqrt{m} - 1)p \ .$$

When R_2 and R_3 are expanded using Equation 4.3 and Equation 4.2.2 respectively, we obtain:

$$\frac{k_3\pi((\frac{s(a\sqrt{m}+3)}{4} + vt)^2 - (\frac{s(a\sqrt{m}+3)}{4})^2)(n - m)}{w^2 - \pi(\frac{s(a\sqrt{m}+3)}{4})^2} = \pi(a\sqrt{m} - 1)p \ .$$

When this equation is solved with respect to the variable m using MATLAB Symbolic Math Toolbox, we obtain the equation:

$$m = \left(\text{Eq6.11} - \frac{\text{Eq6.12}}{\text{Eq6.11}} + \frac{3.\overline{33} \times \text{Eq6.15}}{10 \times \text{Eq6.16}}\right)^2 \ , \tag{6.10}$$

which consists of seven parts (Eq6.11, Eq6.12, Eq6.13, Eq6.14, Eq6.15, Eq6.16 and Eq6.17) which are written below subsequently.

$$\sqrt[3]{\sqrt{(\text{Eq6.14} - \text{Eq6.17} + \text{Eq6.13})^2 + (\text{Eq6.13})^3} - \text{Eq6.14} + \text{Eq6.17} - \text{Eq6.13}} \tag{6.11}$$

$$\frac{0.\overline{33}a(3\pi ps^2 + 8k_3ntvs - 1.6pw^2)}{\text{Eq6.16}} - \frac{0.\overline{1} \times (\text{Eq6.15})^2}{(\text{Eq6.16})^2} \tag{6.12}$$

$$\frac{0.16a \times \text{Eq6.15} \times (3\pi ps^2 + 8kntvs - 16pw^2)}{(\text{Eq6.16})^2} \tag{6.13}$$

$$\frac{0.5(-9\pi ps^2 + 24k_3nstv + 16knt^2v^2 + 16pw^2)}{\text{Eq6.16}} \tag{6.14}$$

$$16k_3t^2v^2 + 24k_3stv - 5\pi pa^2s^2 \tag{6.15}$$

$$\pi pa^3s^2 - 8k_3tvas \tag{6.16}$$

$$\frac{0.0\overline{3703} \times (\text{Eq6.15})^3}{(\text{Eq6.16})^3} \tag{6.17}$$

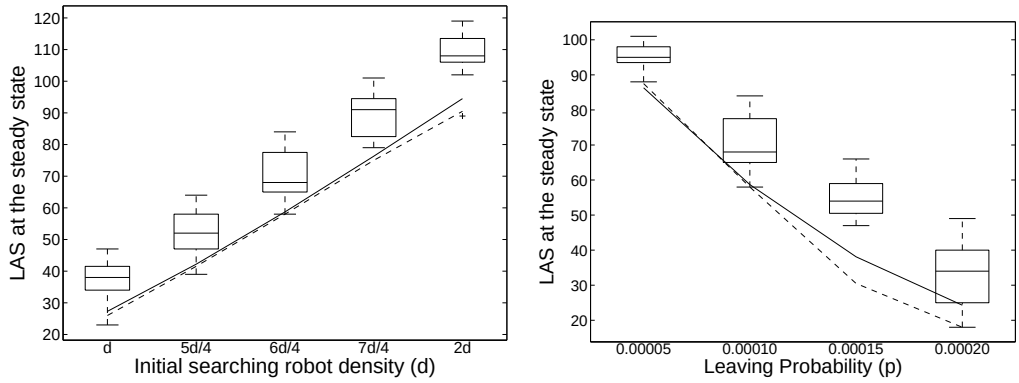
Figure 6.9 shows the aggregate size at the steady state (m) for (a) varying initial searching robot densities and (b) leaving probabilities. While box plots represent the simulation results, regular line connects the median of aggregate sizes calculated using Equation 6.10 and dotted line connects median of aggregate sizes obtained from microscopic model runs. Following conclusions can be drawn from these results.

1. **Steady state and model predictions match.** The steady state values calculated with Equation 6.10 match well with values obtained from the microscopic model.

2. **Model predictions are lower than the simulation results.** This result can be attributed to error in growing probability equation (Equation 4.6). In Section 5.3, it was shown that growing probability predicted with this equation is lower than the simulation results. Since the same equation is used in Equation 6.10, obtaining lower predictions than the actual results is expected.

3. **Aggregate size is linearly proportional with the initial density.** The size of the aggregate at the steady state depends on growing and shrinking probabilities of this aggregate. However, the density effects only the growing probability. According to Equation 4.6, growing probability is linearly proportional with the initial searching robot density. Consequently, the aggregate size at the steady state increases linearly when the density is increased linearly.

4. **Aggregate size is is linearly and inversely proportional with the leaving probability.** As mentioned in the previous conclusion, the size of the aggregate at the steady



(a) (b)

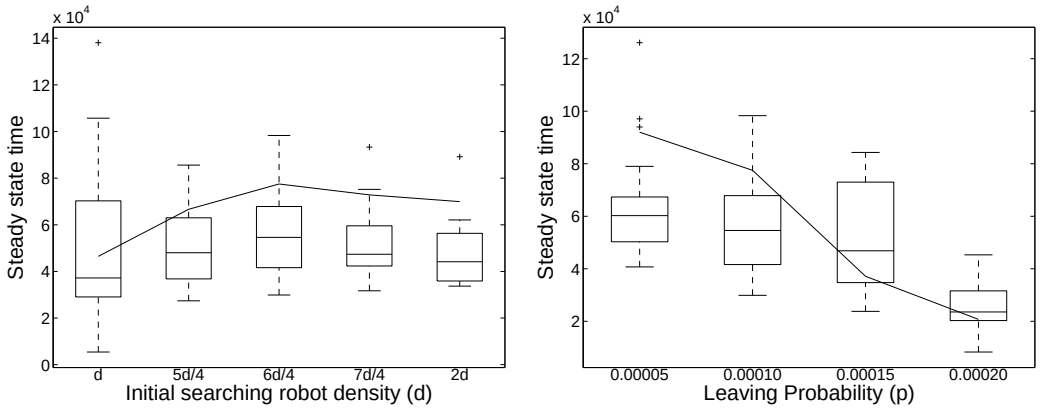
Figure 6.9: Largest aggregate size (LAS) at the steady state for varying (a) initial searching robot densities and (b) leaving probabilities. Simulation results are represented as boxes. Flat and dotted lines connect median of *steady state analysis* and *microscopic model* predictions respectively.

state depends on growing and shrinking probabilities of this aggregate. However, leaving probability (p) effects only the shrinking probability. Since, shrinking probability is linearly proportional with p (Equation 4.11) and shrinking probability is inversely proportional with m, m is inversely and linearly proportional with p.

Figure 6.10-a and Figure 6.10-b show the steady state time for the same experiments. It can be seen that steady state time (1) is *not* changing considerably when the initial searching robot density is increased and (2) decreases linearly when the leaving probability (p) is increased. These two conclusions can be explained as follows.

There are two components of the steady state time in this experiment: growing time and shrinking time. Effects which decrease any of these will decrease the steady state time and effects which increase any of these will increase the steady state time. Additionally, these components are inversely proportional with growing and shrinking probabilities subsequently. Therefore, when the growing probability is increased, growing time is decreased and when the shrinking probability is increased, shrinking time is decreased.

Regarding the first conclusion, when the initial searching robot density is increased, growing probability increases according to Equation 4.6. This will speed up the growing of the aggregate. However, when the growing probability is increased, the aggregate size at the steady state increases which requires to spend more time for growing. Since, growing probability is linearly proportional with both aggregate size and searching robot density, these two effects



(a) (b)

Figure 6.10: Steady state time for varying (a) initial searching robot densities and (b) leaving probabilities. Simulation results are represented as boxes. Flat line connects median of microscopic model predictions.

wipe out each other and the aggregate reaches to its steady state at the same time even the initial searching robot density is increased.

Regarding the second conclusion, when the leaving probability is increased, shrinking probability increases according to Equation 4.11. Increased shrinking probability forces the system to reach the balance between growing and shrinking probabilities at a smaller aggregate size and reduces the time to reach the steady state.

6.2.4 Creation vs. Growing Experiment

In this experiment, searching robots with a certain initial density are placed randomly in an environment. Searching robots are allowed to create aggregates and grow aggregates that is formed by other searching robots. However, waiting robots are prevented to leave their aggregates. In this way, *creation* and *growing* events are allowed but *shrinking* and *dissipation* events are disabled in this experiment.

In order to conduct this experiment as a microscopic model, **CheckAggrEvents** function shown in Algorithm 30 is used.

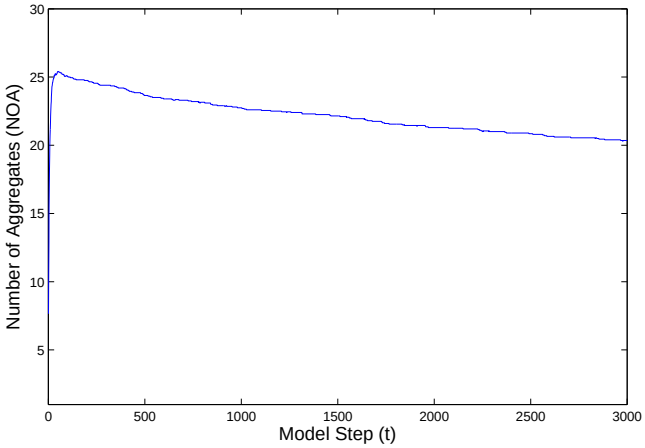


Figure 6.11: Time evolution of number of aggregates (NOA) in the **Creation vs. Growing Experiment**. NOA decreasing by time, forces us to use the maximum performance steady state detection method.

Algorithm 30: Modified **CheckAggrEvents** function for the **Creation vs. Growing Experiment**.

Data: Swarm state vector (x), number of searching robots (y), searching robot density (d), sensing range (s), searching robot speed (v_s), leaving probability (p) and model time step (t).

Result: Updates swarm state vector (x) and number of searching robots (y).

`CheckCreation(`x, y, d, s, v_s, t`);`

`CheckGrowing(`x, y, d, s, v_s, t`);`

In order to find the steady state time and performance for this experiment, the maximum performance steady state detection method is used instead of the sliding window method due to the problem shown in Figure 6.11. The figure shows time evolution of this experiment for a certain initial searching robot density. As can be observed, the number of aggregates is increasing for a certain time period and start to decrease aftwerwards. Due to creation of aggregates, having increasing number of aggregates at the initial stages is normal. However, since shrinking and dissipation events are disabled in this experiment, observing decreasing number of aggregates after some period is *not* expected. After analyzing the situation, it can be seen that this occurs due to the *waiting behavior*. Pushing-pulling behavior of waiting robots can move relatively small aggregates and cause them to join together. Since we can *not* use sliding window steady state detection method for a decreasing steady state, the maximum performance method is used for steady state detection of this experiment.

Figure 6.12-a and Figure 6.12-b show the number of aggregates (NOA) and the aggregate size at the steady state (m) for varying initial searching robot densities subsequently. While the box plots represent the simulation results, lines plot the median of microscopic model runs. It can be seen that when the initial searching robot density is increased,

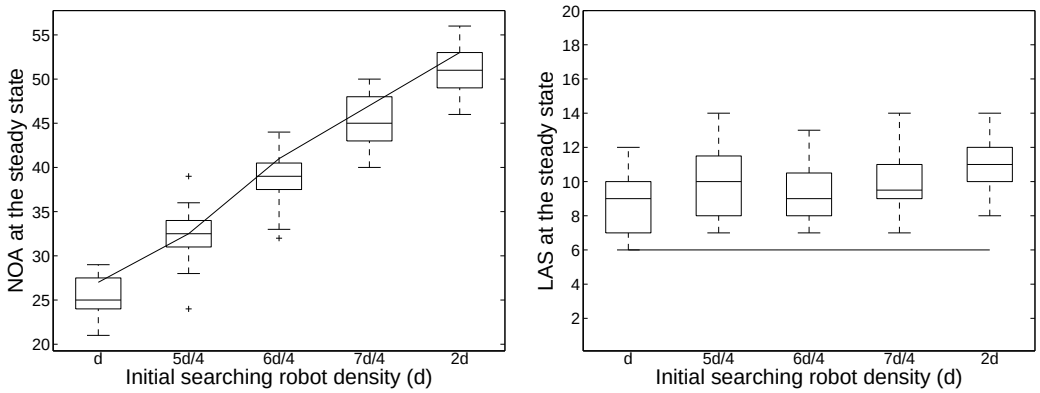
1. **Steady state values match well with the microscopic model predictions.**
2. **NOA is linearly proportional with the searching robot density**
3. **Aggregate size at the steady state does *not* change significantly.**

Second and third results above can be best explained by making an analogy with "Creation Experiment" results in Section 6.2.1. Because the only difference between "Creation Experiment" and this experiment is the addition of growing event. Since aggregate creation event is only related to the creation probability, the linear relation of NOA with density that is shown in the "Creation Experiment" still exist here.

However, since searching robots can either create a new aggregate or grow an existing aggregate, NOA at the steady state is lower in this experiment compared to NOA in the in "Creation Experiment". This can be verified by comparing NOA for the density with 150 searching robots for both experiments. However, creation probability is proportional with square of the searching robot density while growing probability is linearly proportional with the searching robot density. This limits the effect of growing probability and causes very limited changes on m as observed.

Figure 6.13 shows the steady state time for varying initial searching robot densities. It is seen that steady state time decreases slowly when the initial searching robot density is increased. There are two components of the steady state time in this experiment: creation time and growing time. Any effects which decrease/increase any of these will decrease/increase the steady state time. Additionally, these components are inversely proportional with creation and growing probabilities subsequently. When the initial searching density is increased, both creation and growing probabilities increase. This decrease the steady state time.

Figure 6.14 shows aggregate size distribution at the steady state for two different initial searching robot densities. It can be seen that simulation and microscopic model results match well.



(a) (b)

Figure 6.12: (a) Number of aggregates (NOA) and (b) largest aggregate size (LAS) at the steady state for varying initial searching robot densities. Simulation results are represented as boxes. Flat line connects median of microscopic model predictions.

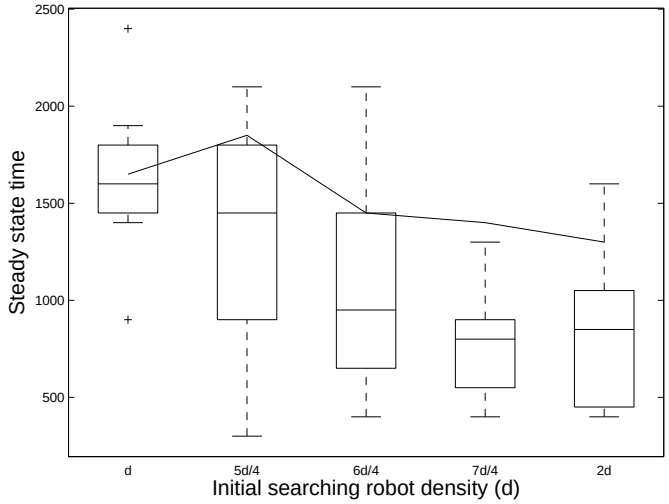
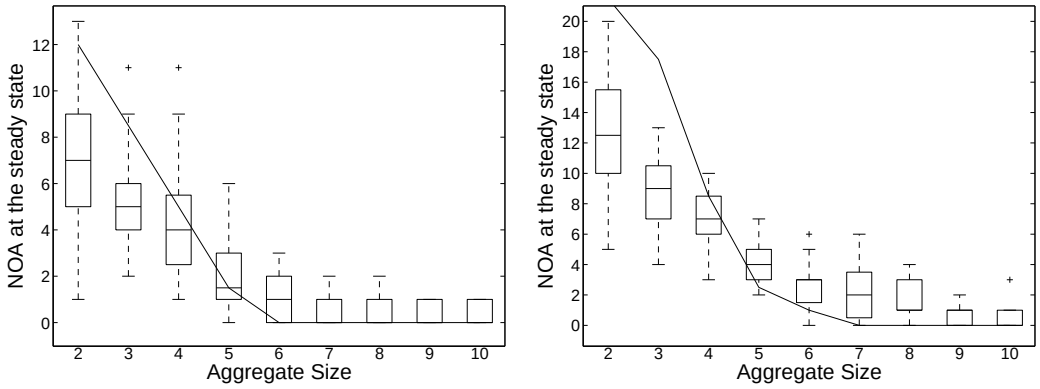


Figure 6.13: Steady state time for varying initial searching robot densities. Simulation results are represented as boxes. Flat line connects median of microscopic model predictions.



(a) 80 searching robots [d] (b) 160 searching robots [2d]

Figure 6.14: The number of aggregates with a certain size at the steady state is plotted for two different initial searching robot densities. Box plots represent the simulation results and the flat line connects the median of number of aggregates predicted with the microscopic model.

6.2.5 Creation, Growing and Shrinking Experiment

In this experiment, none of the aggregation events is disabled and the regular aggregation behavior described in Section 3.2 is used. While searching robots can form new aggregates or grow existing aggregates, waiting robots can leave their aggregates. Default algorithm described in Section 6.1 is used to implement the microscopic model of this experiment.

Figure 6.15 shows largest aggregate size, number of aggregates and number of searching robots at the steady state and the time to reach steady state while leaving probability (p) is increased. By investigating the plots, it can be seen that following changes are happening for the steady state when p is increased:

1. **Model and simulation results match well.**

2. **Microscopic model estimates less NOA and more NS compared to simulation results.** This can be explained with leave timeout effect discussed before. When p is increased, the number of leaving robots in leave timeout duration increases. In the microscopic model, these robots switch to the search state immediately. In simulation, these robots are in leave state but due to being still in aggregate sensing range, they are counted as part of the aggregate. This causes larger NS values to be reported for the microscopic model when p is increased. Since most of the aggregates in the environment are 2-aggregates (See Figure 6.16), majority of leaving robots would leave 2-aggregates and this also causes reporting of more NS by the microscopic model.

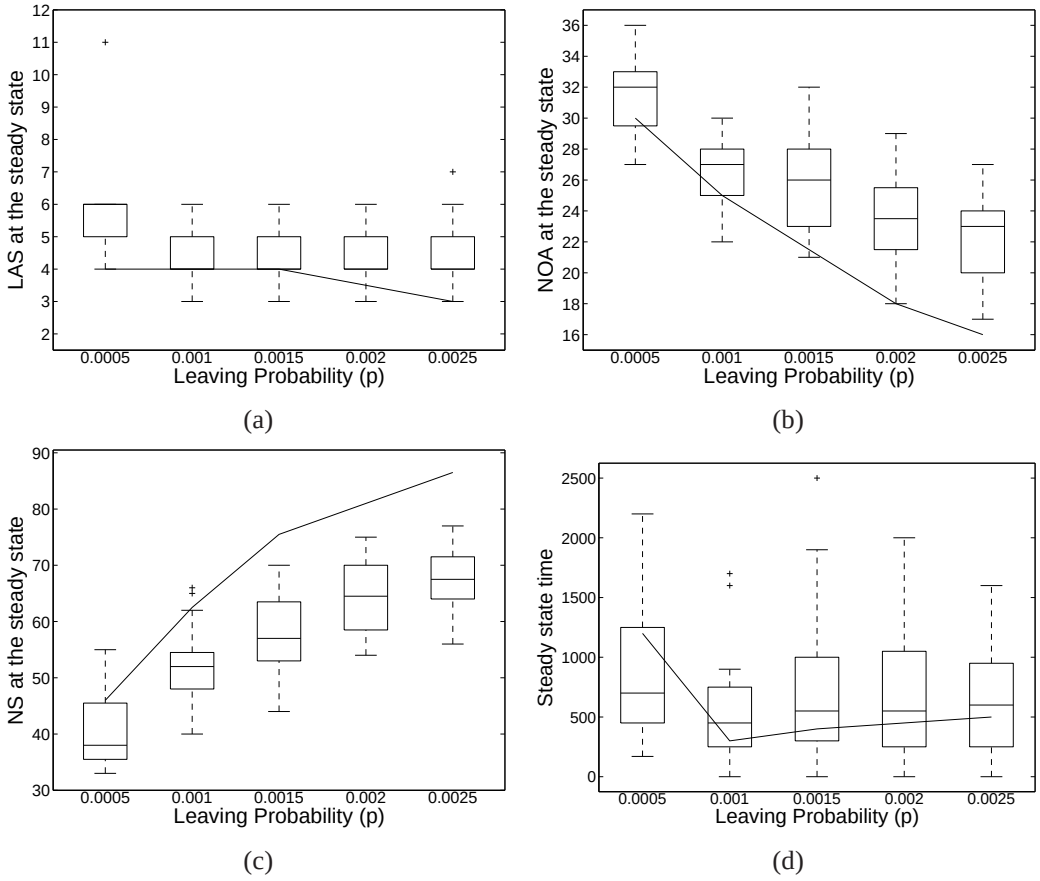
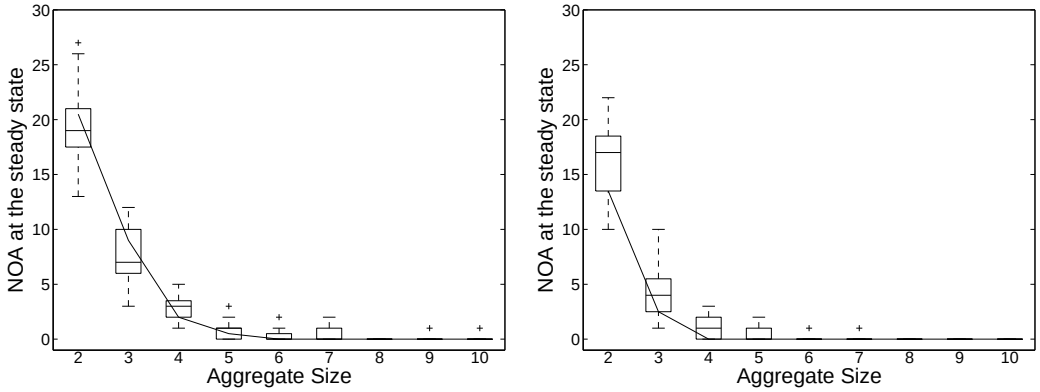


Figure 6.15: (a) Largest aggregate size (LAS), (b) number of aggregates (NOA), (c) number of searching robots (NS) at the steady state and (d) the steady state time for varying leaving probabilities. While box plots represent the simulation results, the flat line connects the median of aggregate sizes predicted by the microscopic model.

3. **NS increases and NOA decreases.** When leaving probability is increased, the size of aggregates at the steady state decreases. This increases NS and decreases NOA.

4. **LAS does *not* change significantly** Although increasing leaving probability makes all aggregates easier to shrink, dissipation of 2-aggregates and shrinking of smaller aggregates is more probable. This can be seen by looking at the aggregate distributions at the steady state shown in Figure 6.16. While there are approximately nineteen 2-aggregates one 7-aggregate in the steady state for $p = 0.0005$ case, there are approximately seventeen 2-aggregates one 7-aggregates in the steady state for $p = 0.0025$. The reason for this is that total dissipation/shrinking probability of many smaller aggregates is bigger than total shrinking probability of few relatively bigger aggregates.

5. **Steady state time is *not* changing significantly.**



(a) P_{leave} = 0.0005 (b) P_{leave} = 0.0025

Figure 6.16: The number of aggregates with a certain size at the steady state is plotted for two different leaving probabilities. Box plots represent the simulation results and the flat line connects the median of number of aggregates predicted with the microscopic model.

Figure 6.16 shows aggregate size distribution at the steady state for two different leaving probabilities. While box plots represent the simulation results, regular line connects the median of aggregate sizes obtained from microscopic model. It is seen that simulation and microscopic model results match well.

CHAPTER 7

Conclusion

In this thesis, we study self-organized aggregation in a swarm of mobile robots that have myopic sensing abilities which allow robots to perceive only a small part of the arena, and do *not* have access to global information such as the size of the environment or number of robots in the environment.

Under these constraints, we proposed an *aggregation behavior* and a probabilistic geometric model for self-organized aggregation. The aggregation behavior consists of four sub-behaviors: *search*, *wait*, *leave* and *change direction*. These behaviors are implemented on a scalable multi-robot simulator.

A probabilistic geometric model is proposed as a tool to analyze aggregation experiments easily without time consuming simulation or real robot experiments. The model consists of four formulas for predicting the probabilities of aggregation events: *creation*, *growing*, *shrinking* and *dissipation* of an aggregate. Our model is significant in several aspects.

First, to the best of our knowledge, this model is the first aggregation model which includes a formula for the creation probability. In previous aggregation modeling studies, creation event, which occurs when two searching robots detect each other, is ignored. In these studies, when two searching robots detect each other, they avoid each other and continue searching. A 1-aggregate is created when a searching robot randomly decides to wait and a 2-aggregate is only formed if a searching robot finds a waiting robot. In this way, the creation event is being reduced to a waiting probability in these studies. However, creation event is a fundamental component of aggregation problem, ignoring creation event requires a non-trivial implementation and delays the solution of the aggregation problem. In previous studies, these issues

are *not* discussed and implementation details are *not* presented.

Second, a specially designed *wait* behavior is proposed and implemented. This wait behavior forces aggregates to be circular so that (1) we derive formulas for growing and shrinking probabilities from circle packing theory and (2) model assumptions regarding growing and shrinking probabilities hold in the simulator.

Third, unlike previous aggregation models, a detailed verification is performed for our aggregation model. In previous aggregation modeling studies, proposed aggregation models are only used to show that simulation and model results match for a proposed aggregation solution strategy using some basic high level metrics and for low number of robots. However, using low number of robots with basic metrics can hide the limitations of these models and a detailed verification is necessary. In our thesis, each formula is verified for varying parameters related to the corresponding formula using simulation experiments. For example, creation probability formula is verified by varying *number, speed* and *sensing range* of searching robots. It is found that model predictions and simulation results match.

Fourth, detailed verification experiments showed that the formulas proposed for growing and shrinking probabilities in this thesis predicts these probabilities for larger aggregates much better than the formulas used in previous studies. Specifically, while previous studies assume that growing and shrinking probabilities are proportional with the aggregate size, detailed experiments in which aggregate size is increased showed that the growing and shrinking probabilities are proportional with the square root of the aggregate size as in our formulas.

In addition to detailed verification experiments, additional experiments, in which certain aggregation events are disabled systematically, are performed in order to verify our model further and use our model as a tool to analyze the main dynamics in aggregation problem. In order to verify that our model can be used to predict the steady state of generic simulation experiments, two different methods are used: *microscopic model execution* and *steady state analysis*. It is shown that largest aggregate size, number of aggregates, number of searching robots and the aggregate distributions at the steady state obtained from microscopic model execution, steady state analysis and simulation experiments are close to each other. The minor difference between model predictions and simulation results are attributed to the “sensing range effect”. Our preliminary experiments to include sensing range effect into model fomulas (e.g. by subtracting the sensing range of searching robot from the free area) in order to

have a better match between the model predictions and simulation results are failed. Further analysis of sensing range effect and including this effect to the model are left as future works.

REFERENCES

[1] Fiducial detector model. http://playerstage.sourceforge.net/doc/stage-2.0.0a/group__model__fiducial.html. Accessed: 20/09/2012.

[2] Law of cosines - wikipedia. http://en.wikipedia.org/wiki/Law_of_cosines. Accessed: 20/09/2012.

[3] Mean free path, molecular collisions. http://hyperphysics.phy-astr.gsu.edu/hbase/kinetic/menfre.html. Accessed: 20/09/2012.

[4] W. Agassounon. *Modeling Artificial, Mobile Swarm Systems*. PhD thesis, 2003.

[5] W. Agassounon, A. Martinoli, and K. Easton. Macroscopic Modeling of Aggregation Experiments using Embodied Agents in Teams of Constant and Time-Varying Sizes. *Autonomous Robots*, 17(2-3):163–191, 2004. Special issue on Swarm Robotics, M. Dorigo and E. Sahin, editors.

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

[7] R. Arkin. *Behavior-Based Robotics*. The MIT Press, 1998.

[8] E. Bahçeci and E. Şahin. Evolving aggregation behaviors for swarm robotic systems: A systematic case study. In *Proc. of the IEEE Swarm Intelligence Symposium*, Pasadena, California, June 2005.

[9] E. Bahçeci, O. Soysal, and E. Şahin. A review: Pattern formation and adaptation in multi-robot systems. Technical report, Carnegie Mellon University, 2003.

[10] T. Balch and M. Hybinette. Behavior-based coordination of large-scale robot formations. In *Proc. of the International Conference on Multiagent Systems (ICMAS-2000)*, pages 363 – 364, Boston, July 2000.

[11] L. Bayındır and E. Şahin. A review of studies in swarm robotics. *Turkish Journal Electrical Engineering and Computer Sciences*, 15:115–147, 2007.

[12] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[13] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2001.

[14] Y. U. Cao, A. S. Fukunaga, and A. B. Kahing. Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, 4:226–234, 1997.

[15] N. Correll and A. Martinoli. Collective Inspection of Regular Structures using a Swarm of Miniature Robots. In *In Proceedigns of the Ninth International Symposium on Experimental Robotics (ISER-04)*, volume 6, pages 375–385. Springer Verlag, 2004.

[16] N. Correll and A. Martinoli. Modeling and Optimization of a Swarm-Intelligent Inspection System. In *Proceedings of the 7th Symposium on Distributed Autonomous Robotic System (DARS)*, 2004.

[17] 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.

[18] N. Correll and A. Martinoli. Modeling and designing self-organized aggregation in a swarm of miniature robots. *The International Journal of Robotics Research*, 30:615–626, 2011.

[19] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In Erol Şahin and William Spears, editors, *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342 in Lecture Notes in Computer Science, pages 10–20, Berlin Heidelberg, 2005. Springer-Verlag.

[20] S. Depickère, D. Fresneau, and J. L. Deneubourg. Dynamics of aggregation in lasius niger (formicidae): influence of polyethism. *Insectes Sociaux*, 51:81–90, 2004.

[21] G. Dudek, E. Jenkin, and Wilkes D. A taxonomy for swarm robots. In *IEEE International Conference on Intelligent Robots and Systems*, pages 441–447, 1993.

[22] J. Fredslund and M. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[23] J. Fredslund and M. Mataric. Robots in formation using only local sensing and control. In *The 7th International Conference on Intelligent Autonomous Systems (IAS-7)*, 2002.

[24] 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*, 17(2):109–133, March 2009.

[25] 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 microrobots. *Artificial Life*, 14:387–408, 2008.

[26] 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.

[27] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, and G. Theraulaz. Collective decision-making by a group of cockroach-like robots. In *In Proceedings of the 2nd IEEE Swarm Intelligence Symposium*, 2005.

[28] V. Gazi and B. Fidan. *Proceedings of the Second International Workshop on Swarm Robotics at SAB 2006, volume 4433 of Lecture Notes in Computer Science*, chapter Coordination and Control of Multi-agent Dynamic Systems: Models and Approaches, pages 71–102. Springer Verlag, 2006.

[29] R. Groß, M. Bonani, F. Mondada, and M. Dorigo. *Proceedings of the Third International Symposium on Autonomous Minirobots for Research and Edutainment*, chapter Autonomous Self-assembly in a Swarmbot, pages 314–322. Springer Verlag, 2006.

[30] A. T. Hayes and P. Dormiani-Tabatabaei. Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3900–3905, Washington, DC, May 2002.

[31] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[32] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Perseus Books Group, 1991.

[33] A. Howard, M. Mataric, and G. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13:113–126, 2002.

[34] A. Howard, M. Mataric, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, 2002.

[35] A.J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11:149–171, 2001.

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

[37] L. Iocchi, D. Nardi, and M. Salerno. *Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From RoboCup to Real-World Applications (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence)*, chapter Reactivity and Deliberation: A Survey on Multi-Robot Systems, pages 9–34. Springer, 2008.

[38] R. Jeanson, S. Blanco, R. Fournier, J. L. Deneubourg, V. Fourcassie, and Theraulaz G. A model of animal movements in a bounded space. *Journal of Theoretical Biology*, 225:443–451, 2003.

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

[40] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[41] T. Labella. Prey retrieval by a swarm of robots. Technical report, IRIDIA, Université Libre de Bruxelles, 2003.

[42] T. Labella, M. Dorigo, and J. Deneubourg. *Proceedings of the First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-ADIT2004), Lecture Notes in Computer Science*, chapter Efficiency and Task Allocation in Prey Retrieval, pages 32–47. Springer Verlag, Heidelberg, Germany, 2004.

[43] T. Labella, M. Dorigo, and J. Deneubourg. Self-organised task allocation in a group of robots. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS04)*, 2004.

[44] K. Lerman and A. Galstyan. Mathematical Model of Foraging in a Group of Robots: Effect of Interference. *Autonomous Robots*, pages 127–141, 2002.

[45] A. Martinoli and K. Easton. Modeling swarm robotic systems. In *Proceedings of the Eight Int. Syrup. on Experimental Robotics ISER-02*, pages 285–294, 2003.

[46] A. Martinoli, K. Easton, and W. Agassounon. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research*, 23:415–436, 2004.

[47] 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.

[48] A. Martinoli and F. Mondada. *Proceedings of the Fourth International Symposium on Experimental Robotics ISER-95*, chapter Collective and cooperative group behaviours: Biologically inspired experiments in robotics, pages 3–10. Springer Verlag, 1995.

[49] M. Mataric. *Proceedings, From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior (SAB-92)*, chapter Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, pages 432–441. The MIT Press, 1992.

[50] Adept MobileRobots. Pioneer robot. http://www.mobilerobots.com/researchrobots/pioneerp3dx.aspx, 09 2012.

[51] S. Nouyan and M. Dorigo. Chain formation in a swarm of robots. Technical Report TR/IRIDIA/2004-18, IRIDIA, Université Libre de Bruxelles, March 2004.

[52] S. Nouyan and M. Dorigo. Path formation in a robot swarm. Technical report, IRIDIA, Université Libre de Bruxelles, 2007.

[53] D. Payton, M. Daily, R. Estkowski, M. Howard, and C. Lee. Pheromone robots. *Autonomous Robots*, 11(3):319–324, 2001.

[54] D. Payton, R. Estkowski, and M. Howard. Pheromone robotics and the logic of virtual pheromones. In Erol Sahin and William Spears, editors, *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342, pages 45–57, Berlin Heidelberg, 2005. Springer-Verlag.

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

[56] O. Soysal and E. Şahin. A macroscopic model for probabilistic aggregation in swarm robotic systems. In E. Şahin and A. F. T. Spears, W. M.and Winfield, editors, *Second International Workshop on Swarm Robotics at SAB 2006*, volume 4433 of *Lecture Notes in Computer Science*, Berlin, Germany, 2006. Springer-Verlag.

[57] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3):137–162, 2004.

[58] 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.

[59] V. Trianni and M. Dorigo. Emergent collective decisions in a swarm of robots. In *Proceedings of the IEEE Swarm Intelligence Symposium*, Pasadena, California, June 2005.

[60] V. Trianni, R. Groß, T. H. Labella, E. Şahin, and M. Dorigo. Evolving aggregation behaviors in a swarm of robots. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 865–874. Springer Verlag, Heidelberg, Germany, 2003.

[61] V. Trianni, T. H. Labella, R. Groß, E. Şahin, M. Dorigo, and Deneubourg J.-L. Modeling pattern formation in a swarm of self-assembling robots. Technical report, IRIDIA, Université Libre de Bruxelles, 2002.

[62] V. Trianni, S. Nolfi, and M. Dorigo. Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, 54(2):97–103, 2006.

[63] R. Vaughan. Massively multiple robot simulations in stage. *Swarm Intelligence*, 2:189–208, 2008.

APPENDIX A

Stage World File

```
resolution 0.02
show_clock 1
paused 1

# THE ENVIRONMENT
define wall model
(
  color "gray30"
  obstacle_return 1
  ranger_return 1
  fiducial_key 1
  fiducial_return 1
  boundary 1
  gui_move 0
)

define wall_cube wall
(
  size [1 1 1]
)

# THE ROBOT
define base_fiducial fiducial
(
```

```
  ignore_zloc 1
  range_min 0
  fov 360.0
  size [0 0 0]
)

define base_robot position
(
  color "red" # Default color.
  drive "diff" # Differential steering model.
  gui_nose 1            # Nose shows which way robot points
  obstacle_return 1     # Can hit things.
  ranger_return 0.5     # Reflects sonar beams
  blob_return 1         # Seen by blobfinders
  fiducial_key 0

  localization "gps"
  localization_origin [0 0 0 0] # Start odometry at (0, 0, 0).

  # four DOF kinematics limits
  # [ xmin xmax ymin ymax zmin zmax amin amax ]
  # x,y,z in meters per second, a in degrees per second
  velocity_bounds [-1 1 0 0 0 0 -90.0 90.0 ] # -2 2
  acceleration_bounds [-0.5 0.5 0 0 0 0 -90 90.0 ]

  size [1 1 1]
  origin [0 0 0 0]
  mass 23.0

  base_fiducial( name "robot fiducial" fiducial_key 0 range_max 4 )
  base_fiducial( name "wall fiducial" fiducial_key 1 range_max 4 )
)
```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name	: Bayındır, Levent
Nationality	: Turkish (TC)
Date and Place of Birth	: 10 December 1979 , Ankara, TURKEY
Email	: levent@ceng.metu.edu.tr, levent.bayindir@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
Ph.D.	METU, Computer Engineering Dept.	2012
BS	Ege University,	
	Computer Engineering Dept.	2002
High School	Yalova High School, Yalova	1997

WORK EXPERIENCE

Year	Place	Enrollment
2002–Present	METU, Department of Computer Eng.	Research and Teaching Assistant
2009–2010	IRIDIA, Belgium	Visiting Scholar

FOREIGN LANGUAGES

Turkish: Native

English: Advanced

PUBLICATIONS

1. A. E. Turgut, F. Gökçe, H. Çelikkanat, L. Bayındır, and E. Şahin. Kobot: Sürü robot çalışmaları için tasarlanmış gezgin robot platformu. Otomatik Kontrol Ulusal Toplantısı (TOK'07), 259-264, Sabancı Üniversitesi, 2007.

2. L. Bayındır and E. Şahin. A Review of Studies in Swarm Robotics. Turkish Journal Electrical Engineering and Computer Sciences, 15, pages 115-147, 2007.

3. E. Şahin, S. Girgin, L. Bayındır, and A. E. Turgut. Swarm Robotics. Swarm Intelligence - Introduction and Applications (editors: C. Blum and D. Merkle), Springer, pages 87-100, 2008.

4. L. Bayındır and E. Şahin. Modeling Self-Organized Aggregation in Swarm Robotic Systems. Proc. of the IEEE Swarm Intelligence Symposium (SIS'09), pp. 88-95, 2009.