A Survey of Non-Spatial and Spatial Modeling Techniques and Applications for Robot Swarms

Anshul Kanakia
December 16, 2013

1 Introduction

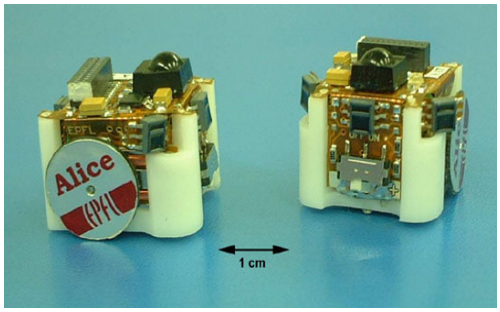
Over the past two decades the field of swarm intelligence has evolved from intertwined branches of mathematics, physics, biology, traditional robotics, and artificial intelligence; attempting to apply observations of physical and biological systems to artificial ones. This has led to novel approaches in the design and analysis of multi-agent systems and the algorithms associated with them. It has spawned an entirely new field of research called *swarm robotics*. A definition of swarm robotics as given by Şahin and Spears is,

> *The study of how a swarm of relatively simple physically embodied agents can be constructed to collectively accomplish tasks that are beyond the capabilities of a single one.*[4]

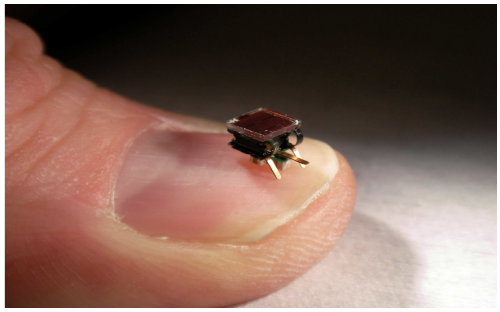
This literature survey paper attempts to collate the research done so far in this relatively new field, while focusing on mathematical and computational modeling techniques that have been developed to help describe the temporal and spatial evolution of a swarm of robots programmed to perform specific, collaborative tasks. While literature reviews for swarm robotics models already exist[55, 36], this paper attempts to include both, non-spatial and spatial modeling techniques as well as discussions on parameter discovery and optimization which is an area I am particularly interested in.

Swarm systems have many benefits over traditional, centralized robot systems. The robots used in swarm applications are generally many orders of magnitude smaller (centimeters vs. meters, grams vs. kilograms) and simpler in design (< 10 actuators vs. 100s of actuators) than conventional robots. Also, most swarm systems are homogeneous—robots with identical software/hardware are used to complete the assigned task. This makes swarm systems easily scalable while simultaneously keeping manufacturing and maintenance costs of the hardware low. Though, perhaps their greatest advantage is system stability and robustness to error. Many well known swarm algorithms such as Ant Colony Optimization, Virtual Pheromone Tracking [48], and Particle Swarm Optimization rely on the fact that most agents (ants/particles) are on a non-optimum trajectory so as to maximize coverage over the search space. This concept also carries over to a hardware swarm system. Most swarm systems consist of small, relatively simple robots that are only capable of limited and noisy sensing, communication and actuation. This means that while no single robot alone is capable of performing the task assigned, the system as a whole is resilient to individual errors from agents and is capable of completing the task.

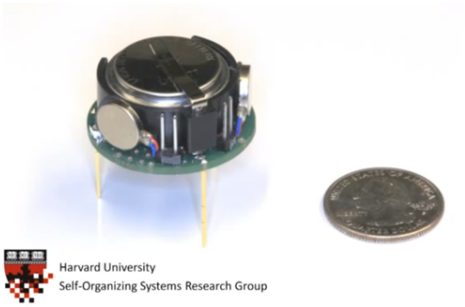
So then, why do we want to model swarm systems? While much progress has been made in the development and production of miniature swarm robots such as, Alice, Khep-



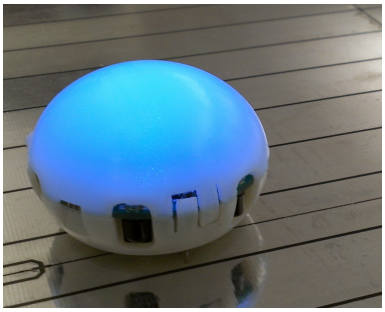
(a) Alice robot



(b) I-SWARM robot concept



(c) Kilobot robot



(d) Droplet robot

Figure 1

era, Kilobots, etc.[20, 52, 46, 12, 50] there is still no viable large scale ($> 10,000$ units) implementation available. It is also difficult to observe and measure large scale system properties during physical experiments without the use of more sophisticated tools like machine vision and global communication. Thus, many algorithms in the field have been developed and formally proven using mathematical models and computer simulation, generally backed up by small scale physical experiments. While physics based simulators such as Webots[44] are indeed helpful for rapid experimentation and observation, they take time to develop and are computationally intensive processes. Simulations are generally faster than physical experiments but slower than mathematical models. They are also not exact—by their very nature—and therefore could miss emergent behavior or other macroscopic properties as they tend to simulate systems on the individual agent or microscopic level.

Methods for modeling a swarm system are therefore extremely important as they provide a formal understanding of the relationship between control parameters, physical properties, and performance. These models can then be used to provide formal guarantees on performance as well as system identification and parameter optimization[16, 14].

With this motivation for modeling swarm systems in mind, the rest of the paper is structured as follows. The next section describes and defines the technical terms that will

be used for the remainder of this paper. While many of the terms, such as, spatial, non-spatial, robot, etc. can have very broad and general definitions, the section aims to define them more specifically in the context of this survey paper and it's purpose; to explain the different modeling and optimization techniques in use for studying swarm robotics algorithms. Section 3 outlines a general methodology often used in the swarm robotics community for designing and optimizing a robot controllers and developing swarm system models. Section 4 covers non-spatial modeling applications using an array of well known swarm robot experiments. The end of this section also describes my original work done in designing a novel robot controller for multi-robot collaboration based on a sigmoid threshold response function. In the next section we begin to explore the less understood realm of spatial models in swarm robotics. The models discussed in this section are heavily inspired by physical models of Brownian motion and involve the use of the Langevin equation and the Fokker–Planck equation (also known as the Kolmogorov forward equation). Finally, the conclusion provides a short summary of open questions in the field of swarm system modeling and areas of research I am interested in pursuing further.

2 Terms and Definitions

2.1 Agent/Robot

In the book, *Artificial Intelligence: A Modern Approach* by Russel and Norwig[51], an agent or robot is defined as, "Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors." While this definition is sufficient in most cases, it excludes one important capability of robots required in most robot swarms—communication. Therefore, an enhanced definition for a *swarm robot* could be:

> Any mechanical automation capable of sensing it's surroundings, processing sensory inputs via internal computation, actuating itself or other objects in the environment based on the inputs, and communicating information with other robots around it, either directly or via *stigmergy*.

2.2 Stigmergy

Stigmergy is a term used to describe indirect communication in robot swarms. The word was first coined by Pierre-Paul Grasse, in his 1959 paper[23], while studying insect behavior and has become a commonly used term in the swarm robotics community.

While most robots are capable of communicating amongst themselves explicitly via infrared, *Bluetooth*$^{\text{TM}}$, and other wired or wireless means, most swarm algorithms try to keep such explicit communication to a minimum. This is done to maintain scalability of the system (e.g., prevent message flooding) and keep the underlying algorithm simple.

Indirect methods of communication are thus preferred when designing controllers for robot swarms, such as changing one's color or moving in a particular pattern or even just waiting at a specific position in the environment (i.e. sensory feedback). The process of adding information to the swarm system by affecting or altering the environment rather than explicitly communicating with other agents is referred to as a stigmergic process[5]. It is used in many of the swarm algorithms discussed in this paper.

2.3 Swarm System

A myriad of definitions are available for systems and models that exhibit "swarming" phenomena—the term "swarming" itself being re-defined in numerous cases. An interesting discussion of the nomenclature of swarm systems is available in [7, 8].

Since our primary purpose in modeling robot swarms is to study and understand properties of the system as a whole, the term *swarm system* or *multi-agent system* is not used just as a collective noun for a group of robots but instead describes the complex relationship between agents, the environment, and the tasks they are attempting to accomplish.

Multi-agent systems can be modeled at different levels of abstraction depending on the system properties we attempt to expose. When these models are used in tandem we

gain the ability to both, verify and enhance the original swarm system via optimization of system parameters. We see the deployment of this strategy in the next section but first, we define the different abstraction levels used in robot swarm modeling.

2.4 Microscopic Level

The micro-level of a robot swarm treats the individual agent as the fundamental unit of the model[34]. Though not a requirement for this form of agent-based modeling, we generally assume that the swarm is homogeneous, i.e. every agent is running the same robot controller within it and all hardware (sensors, actuators, processors and communication devices) between robots is identical. The microscopic level then helps describe direct agent-to-agent interactions as well as agent-to-environment interaction.

A simple method used for micro-level modeling consists of writing down the dynamics equations (equations of motion) for an individual robot and solving them to study system-level behavior. As one can imagine, these dynamics equations can become very tedious and difficult to solve for more complex swarm systems due to the high number of agents, inelastic collisions between agents and obstacles, sensor and actuator noise, etc. Therefore a more common approach to micro-level modeling involves stochastic simulation of individual robot controllers in parallel. This micro-level modeling method has been derived from the popular *Gillespie* method[21, 22] used to model coupled chemical reactions.

2.5 Macroscopic Level

While micro-level models deal with system on an individual agent level, macro-level models consider the system as a whole entity and are used to describe collective group behavior. Macro-models for robot swarms are often phenomenological in nature. The system's parameters are derived from observing and measuring real physical phenomena and extrapolating such properties as may be deemed useful for understanding said phenomena. Macro-level models are generally represented as a system of ordinary (for non-spatial models) or partial (for spatial models) differential equations and as such are good at describing the temporal (and spatial) evolution of the system. They are often also called population dynamics models and/or rate equations in literature.

3 Methodology for Designing a Swarm Robot Control Algorithm

This section describes a particular methodology for designing and modeling robot controllers that is widely used in the swarm robotics community. While this is by no means the only modeling method available for robot swarms—e.g., other (related) modeling and control methods are described in [6, 10, 11, 54]—it has the important property of being an iterative method that integrates multiple levels of abstraction of the multi-agent system. This opens up the opportunity for parameter discovery and optimization, which I am particularly interested in and is covered in section 3.7.

Some early work related to modeling swarm robot systems was done by Martinoli et al.[43, 40] outlining collective group behaviors and the associated system dynamics by studying two distinct swarming tasks. The first task involved the aggregation of sticks by teams of robots in a closed environment while the second experiment was a team-based, collaboration task where groups of two robots were used to pull long sticks out of the ground. The second experiment has been widely studied and extended upon by numerous researchers like Martinoli, Mondada, Ijspeert, Lerman, and Correll in the past decade and is well known in the community as the "Stick Pulling Experiment". We will use this experiment as a recurring example throughout this section to introduce new concepts, beginning with a general procedure for designing the robot controller. If the reader is not familiar with the stick pulling experiment, a brief description is available in section 4.2 and is recommended before continuing with this section.

3.1 Experiment Setup

The first step in the robot controller design methodology is to describe the swarming task being studied. The general strategy used by each individual agent in the swarm is defined and later translated into a viable microscopic model. A hypothesis for the observed, collective behavior of the swarm is also supplied, which is later quantified into a mathematical macroscopic model.

Physical characteristics of the swarm system are generally described in the experiment setup as well. These may include environmental variables such as arena size, agents properties such as speed, communication and sensing radii, the computation power of each individual in the swarm, etc. This is an important step in identifying the important system parameters that affect the outcome of the experiment, versus the environmental and agent based values that can be abstracted away when designing micro and macro-level models.

3.2 Designing the Controller Construct

With this experiment setup in mind we will now discuss designing the robot controller. The first step in this process is to create a logic construct—a flowchart, state-machine or algorithm that describes the desired robot behavior for the given task.

The original robot controller for the stick pulling experiment was designed as a flowchart seen in Figure 2. The rectangular boxes in the flowchart are states that the robot controller can be in while the rhomboid shapes signify decision processes within the controller. The result of each decision process is dependent on several factors, for example, the *object detected?* decision depends on feedback from the robot's IR proximity sensors. The *obstacle?* decision depends on the output produced by the sensory neural net of the agent (see Figure 11b). The *obstacle?* decision happens when the robot has an object in front of it and must decide whether it is an obstacle such as another robot or wall, or a target such as a stick. As one can imagine, the output from the neural net is not always correct due to factors like the distance and angle of incidence between the detecting robot and the object in front of it, as well as sensor noise.

Given this stochastic nature of decision processes in the flowchart, it makes sense for us to assign them probabilities of traversal. We therefore convert a physical 2D arena space into a 1D probability space via a transformation function, $\boldsymbol{\tau} : \mathbb{R}^2_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, (top of Figure 3). The probabilities, p_{robots}, p_{walls}, etc., seen in the flowchart are computed using simple geometric properties of the environment. For example, p_{walls} is computed as the ratio between the area where a robot can sense the wall (the dark gray perimeter of the square seen in top Figure 3) and the area of the whole arena.

You may notice that in Figure 3 there are still some deterministic decision processes, such as *time out?*. The *time out?* decision process is special as it deals with an internal state variable in the agent, namely, its timer value. Such deterministic decision processes make the mathematical system more complex because of the need to maintain the current state of each agent instead of the collective state of the whole system. As we will discuss later, there are ways to stochastically model these deterministic decision processes (thereby simplifying the mathematical model) without loss of generality and without affecting the solution of the system in equilibrium, if one exists.

The use of a flowchart makes it straightforward to describe the robot controller but doesn't do a very good job of exposing the true stochastic nature of the system, nor does it allude to the possibility of a mathematical model for the system. And so, we exchange this simple construct for a more mathematically robust one—a finite state machine (FSM).

When studying non-spatial models, the robot controller can be characterized by an FSM with a discrete number of states under urgent, time-step driven semantics, as seen in Figure 4. Just like the flowchart, the states in the FSM (A, B, C & D) represent physical states that the robot can be in, such as *searching*, *waiting*, etc. One can think of each state as being an *action* that the robot is currently performing based on stimulus from the environment and other robots. These stimuli can cause a robot to transition from

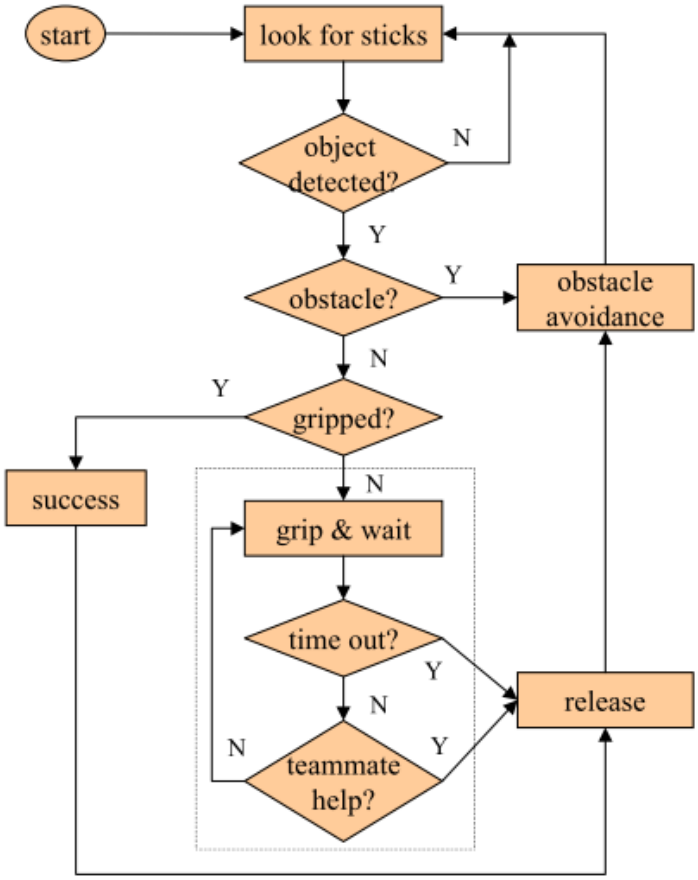


Figure 2: Flowchart for an individual robot controller in the stick pulling experiment.

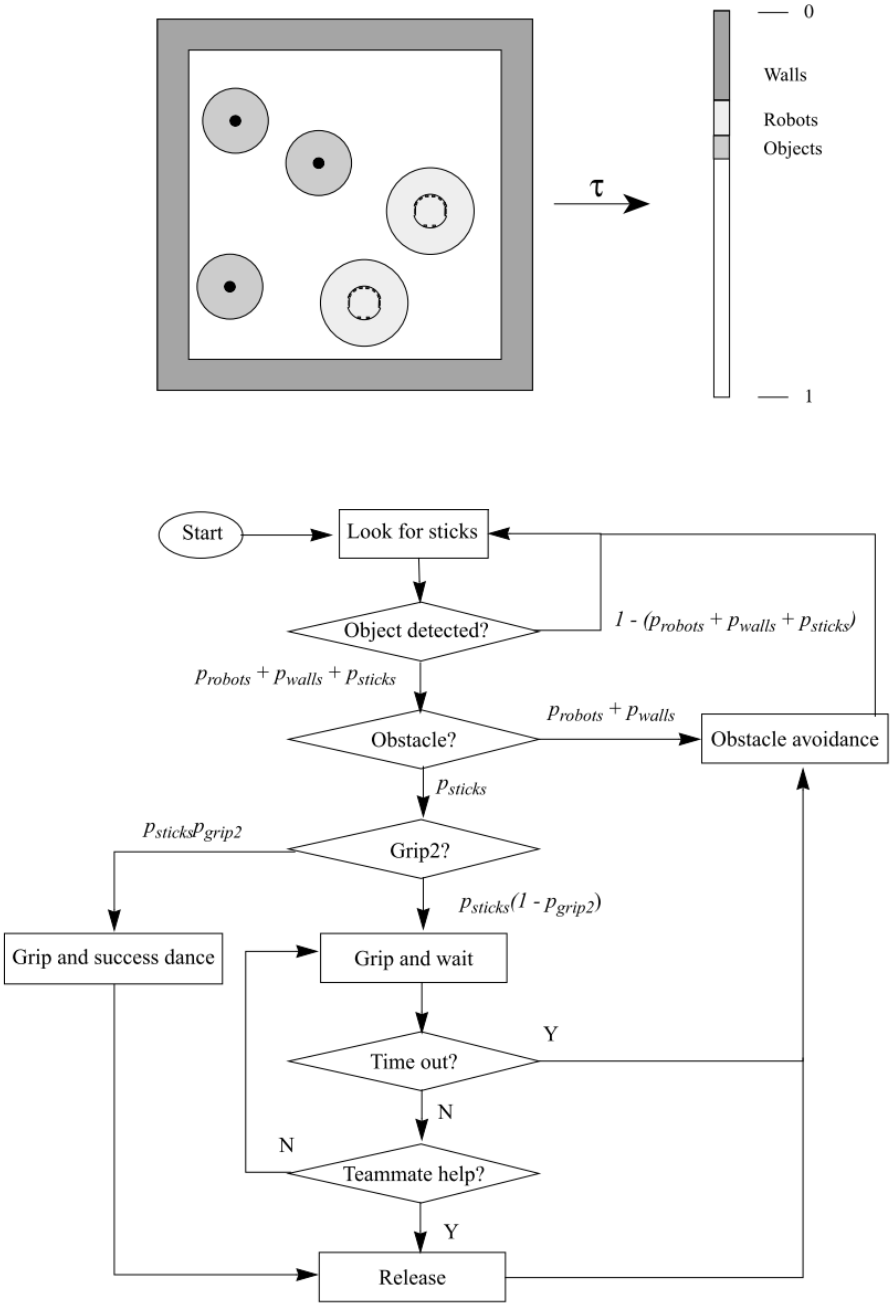


Figure 3: Top: A visualization of the conversion function $\boldsymbol{\tau} : \mathbb{R}^2_{\geq 0} \to \mathbb{R}_{\geq 0}$, going from a 2D physical space to a 1D probability space. **Bottom:** Flowchart of a robot controller in the probabilistic model of the stick pulling experiment. p_{robots} = probability of encountering teammates; p_{walls} = probability of encountering walls; p_{sticks} = probability of encountering one of the sticks; p_{grip2} = probability to make a grip2

one state to another and are represented as *conditionals* on the edges of the FSM, c_i. These conditionals are equivalent to the decision process blocks in the flowchart and can be derived from:

1. Sensor readings (or stigmergy) and explicit communication, e.g., seeing red light through an rgb sensor or seeing a certain number of robots around you,
2. internal timers, e.g., transition back to search after waiting for 3 seconds,
3. or a combination of both, e.g., transition back to search after waiting for 3 seconds *iff* you see no other robots in your vicinity, otherwise, reset your timer.

We can now extend this modeling framework of robot behavior as an FSM to construct a Probabilistic Finite State Machine (PFSM), where the conditionals in the FSM are no longer *true/false* values but instead are probabilities of transitioning from one state to another based on external stimulus or internal state.

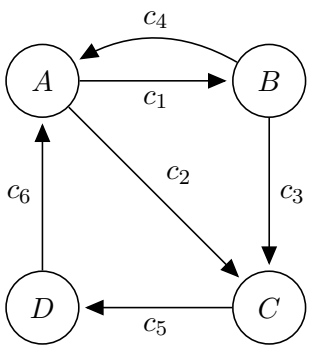
As alluded to earlier, the case of a state transition based on an internal timer is especially interesting. Let c_1 in Figure 4a be the condition $t_A \geq 5$, i.e. time in state A is greater than or equal to 5 time steps (or ticks). This conditional is true when the robot has remained in state A for at least 5 ticks and consequently transitions to state B. Thus, the conditional c_1 says a robot may remain in state A for no more than 5 ticks. The equivalent transition probability for this condition would be $p_1 = 1/5$. Therefore, at each time step of the PFSM simulation, there is a $1/5$ chance that the robot will transition from state A to state B. The expected number of ticks before a transition happens is then equal to 5. This transition from deterministic FSM models to probabilistic PFSM models for swarm robot algorithms is discussed in further detail in[14].

3.3 Mathematical Description of the System

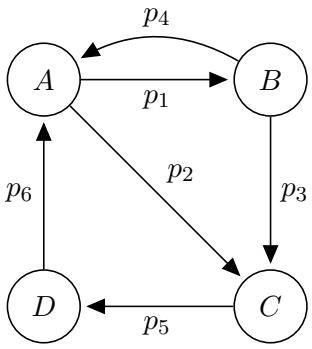
Given a discrete set of states and conditions for transitions between them, usually in the form of probabilities of transition, a *master equation* defines a set of ordinary differential equations that describe the time evolution of a physical system. So far, we have used logical constructs like the FSMs to represent the robot controller running within each individual agent of the swarm system. We could instead look as these constructs as a model for the entire system, in which case the vertices of the FSM become accumulators of robots currently in a state and the edges define fractions of agents entering or leaving a given state at time t. The PFSM now becomes a macroscopic definition of the robot swarm and can be used to define a mathematical model for the time evolution of the system.

We will shortly discuss the conditions for linearity in a swarm system but for now, assume a simple master equation of the form,

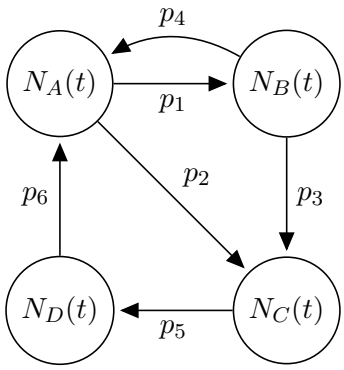
$$\frac{d\vec{P}(t)}{dt} = \mathbf{A}\vec{P}(t) \tag{1}$$



(a) FSM representing a single robot controller with conditional edge transitions that depend on internal and environmental factors such as timers, sensor readings, etc. Vertices represent physical or internal states that a robot can be in.



(b) PFSM of a robot controller with probabilistic edge transitions derived from simple geometric properties of the system.



(c) A macroscopic model for the swarm system as a whole. Vertices, $N_{s_i}(t)$, represent the number of robots in state s_i at time t. Edges are still transition probabilities between states but can also be thought of as proportions of agents entering or leaving a state at time t.

Figure 4: Transitioning from, **(a)**: micro-model FSM that describes a single robot controller, to **(c)**: macro-model PFSM that characterizes the entire swarm system.

$\vec{P}$ is a vector containing $p_i(t)$, $i = [1, \ldots, |P|]$, the time-dependent probability of being in state-i in the corresponding PFSM. $\mathbf{A}$ is a matrix containing transition rates of going from state-i to state-j in the PFSM. When we multiply both sides of equation (1) by the total number of agents, N_0, we get the modified master equation that gives a macroscopic description of the system.

$$\begin{aligned} N_0\vec{P}'(t) &= \mathbf{A}\left(N_0\vec{P}(t)\right) \\ \vec{S}'(t) &= \mathbf{A}\vec{S}(t) \end{aligned} \tag{2}$$

where $\vec{S}$ is a state vector containing the number of agents in each state, N_{s_i}, at time t. Here, $|\vec{S}|$ is equal to the number of unique states of the system, e.g. $|\vec{S}| = 4$ in our previous PFSM example from Figure 4b. The matrix $\mathbf{A}$ contains transition probabilities between the states in the PFSM. There a two types of elements, a_{ij} in matrix $\mathbf{A}$.

1. The non-diagonal entries, a_{ij} s.t. $i \neq j$, are equal to $p(c_{ij})$ (shortened to p_{ij}), the probability of transitioning from state s_i to s_j via the edge with conditional c_{ij} in the FSM.

2. The diagonal entries, a_{ii}, are equal to the negative sum of all edge probabilities p_{ij} leaving state s_i.

If an edge does not exist between two states s_i, s_j $(i \neq j)$ in the FSM, then entry $a_{ij} = 0$, e.g., the master equation for the swarm system described in Figure 4b is,

$$\begin{pmatrix} N'_A(t) \\ N'_B(t) \\ N'_C(t) \\ N'_D(t) \end{pmatrix} = \begin{pmatrix} -(p_1 + p_2) & p_4 & 0 & p_6 \\ p_1 & -(p_3 + p_4) & 0 & 0 \\ p_2 & p_3 & -p_5 & 0 \\ 0 & 0 & p_5 & -p_6 \end{pmatrix} \begin{pmatrix} N_A(t) \\ N_B(t) \\ N_C(t) \\ N_D(t) \end{pmatrix} \tag{3}$$

In most of the scenarios being discussed in this paper, we assume that agents are neither removed nor added to a swarm system once an experiment has begun and therefore add the following constraints to the model,

$$N_0 = \sum_{i=1}^{|\vec{S}|} N_i(t) \tag{4}$$

$$\forall i \leftarrow 1 \ldots |\vec{S}|, \sum_{j=1}^{|\vec{S}|} a_{ij} = 1 \qquad \text{(in matrix } \mathbf{A}\text{)} \tag{5}$$

Due to this constraint a simplification can be made to any one (but no more than one) of the states s_i in $\vec{S}$ so that,

$$N_{s_i}(t) = N_0 - \sum_{|\vec{S}|}^{j=1, j\neq i} N_{s_j}(t) \tag{6}$$

In swarm robotics literature, the master equation is often expanded to a set of difference equations (DEs) or continuous ODEs called *rate equations* of the form,

$$N'_{s_i}(t) = \sum_{j=1}^{|\vec{S}|} p_{ji} N_{s_j}(t) - \sum_{k=1}^{|\vec{S}|} p_{ik} N_{s_i}(t) \tag{7}$$

along with a set of initial conditions that define the number of robots in each state, s_i, at time $t = 0$. Rate equations are the preferred method for describing a macro-model of a swarm system because, unlike the master equation, they can represent probability values that could be complex, non-linear functions of environment variables, control variables, as well as time. These are also referred to as population dynamics models, or PDMs.

3.4 Microscopic Simulation of the System

One of the advantages of using macroscopic, mathematical models for describing robot swarms is their ability to predict the state of the system at equilibrium, if it exists. But given the phenomenological approach to designing macro-models, it may not always be intuitive to construct the math equations to accurately describe the system. This is especially true in the case of spatial macro-models, which we will discuss later. Even if the rate equations are defined, the system may not be easily solvable, either analytically or numerically. Fortunately there is another modeling tool that comes to our aid in such situations.

The Microscopic model (or micro-model) of a swarm system can be simulated using the *Gillespie* simulation technique[21, 22]. Here, each agent is simulated individually using dice rolls and probability. Gillespie developed this simulation algorithm in the 1970s to model the time evolution of reactant and product volumes in a chemical reaction. The individual agents in his chemical system were single molecules of the reactant and the micro-model was derived from the dynamics of molecule interactions. The probability of two reactant molecules colliding was computed using simple physical properties such as the radius and velocity of the molecules in the reaction medium[21].

We can modify Gillespie's original modeling approach for use with swarm systems—instead of modeling simultaneous molecule interactions we model simultaneous interactions of robot controllers. Martinoli outlines this process in detail in chapter 4.2 of his Ph.D. dissertation[39] and provides the following concise description,

> "*[Figure 5] shows a general overview of the probabilistic model. The whole simulation consists in running several probabilistic processes in parallel, with one process per robot, while keeping track of the state of the environment. The environment can be seen as a shared memory area (or blackboard) to which all probabilistic processes have access. The state of the robots is defined by a program with exactly the same structure as that of the controllers of the real robots, but, instead of computing the detailed sensory information and trajectories of the robots, the change of states is determined by the throwing of dice (probabilistic blocks). This is an interesting feature of the method because any extension of the real robot controllers can be implemented easily in the probabilistic model and vice versa.*"

Micro-level simulations are generally not as computationally intensive as solving the rate equations for a macro-model and provide similar results which can, in turn, be used

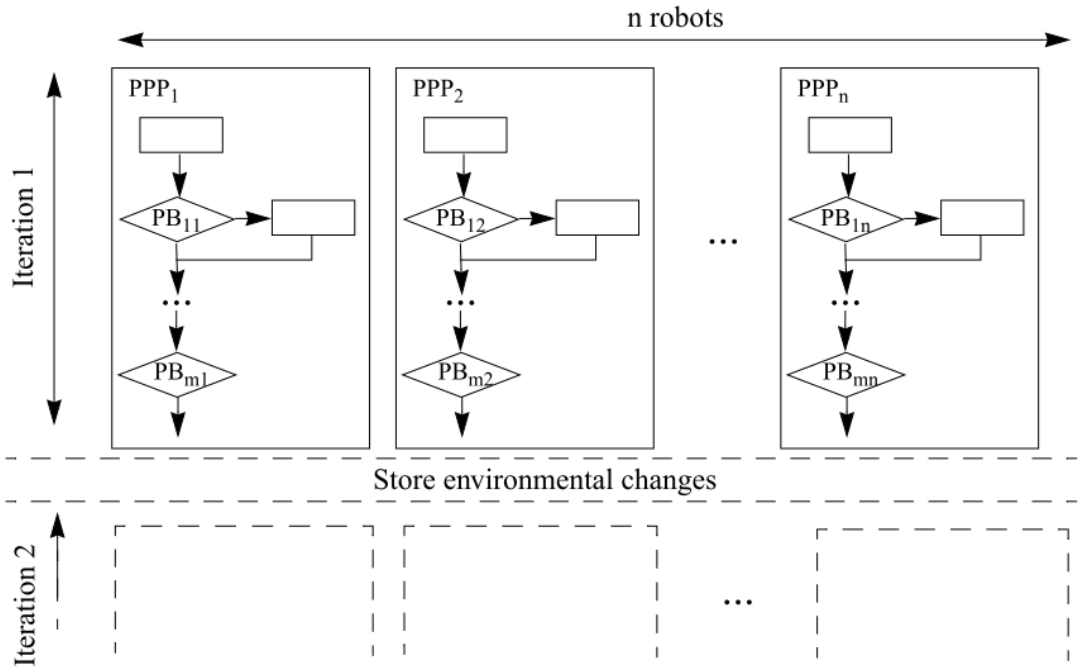


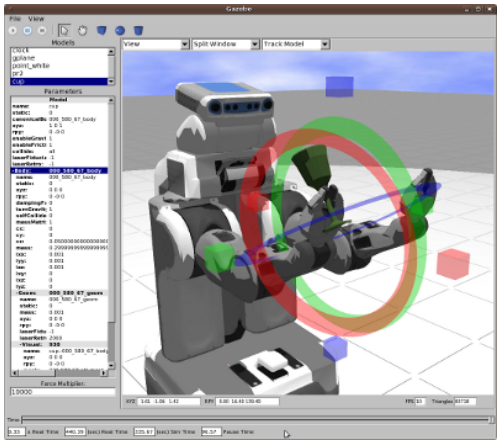
Figure 5: Gillespie simulation of a swarm system. Each controller, PPP_i, describes the independent behavior of a single robot in the swarm. Environmental variables are updated at the end of each iteration.

to verify the correctness of the more mathematically and computationally heavy macro-models. This is why micro-level simulations are both, a valuable tool and important step in the design and analysis of swarm systems.

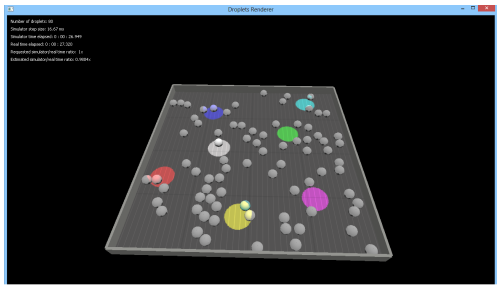
3.5 Verification of System Properties Using Real Experiments and Physics-Based Simulation

An important step in any modeling process is verification by comparing model results to real experiment data. Given the relatively abstract approach we have seen so far for designing models of robot swarms, this step is made even more crucial. The micro and macro-models in swarm robotics have conventionally been designed using observed phenomena from other processes seen in biological and chemical systems and adapted to fit the swarming task being studied. Many swarm algorithms show emergent behavior where the observation of complex properties at the system level cannot be trivially inferred from studying the individual agent behavior. The generalizations and simplifications made in the robot controller design when developing the micro and macro-models can, and in many cases do, suppress the interesting emergent properties seen in real physical systems.

To be able to accurately recreate a task on a swarm system without investing the substantial time and resources required to develop and deploy real robots, we use physics-based simulators. The point of these simulators is to remain as true to the real world as possible while maintaining an order of magnitude improvement in speed and simplicity over real robot experiments. Unlike micro-models that abstract away physical and environmental issues such as wheel slip, sensor noise, communication delays, etc., using probability and dice rolls, physics-based simulators make the added effort to accurately and dynamically model every minute aspect of the swarm system.



(a) The Gazebo robot simulator with the PR2 robot.[1]



(b) The Droplet swarm robot simulator

Figure 6

Many robot simulators are currently available today as either standalone programs like Webots, OpenRAVE and Gazebo (see Figure 6a) as well as plug-ins and toolkits for popular scientific and engineering software like SolidWorks (Assembly Toolbox), Matlab (Robotics toolbox), Mathematica (SystemModeler) etc. Webots is a widely used simulator in swarm robotics due to its capability to simulate multiple agents and agent-agent/agent-environment interactions in real time. The Webots API also allows for cross-compilation of programs from the simulation environment, right on to real robots without the need for reprogramming and supports a wide range of commercially available robot platforms such as Kilobots, Khepera, Alice, etc. There are also in-house implementations of physics simulators for specific robot platforms, such as the Droplet simulator at Correll Lab seen in Figure 6b.

In almost all swarm robot literature, the results being discussed about a swarming algorithm are backed by simulation data and real experiment data to reinforce the correctness of the algorithm and to verify that it behaves as expected. While this may not sound like a formal verification technique, it is widely used in the robotics community and is generally

[1]Image credit: `http://ftp.isr.ist.utl.pt/pub/roswiki/simulator_gazebo.html`

accepted as a good indication that an algorithm works as described and is often supplied as added conformation along with a more robust mathematical verification of the model.

3.6 Repeating the Design Loop

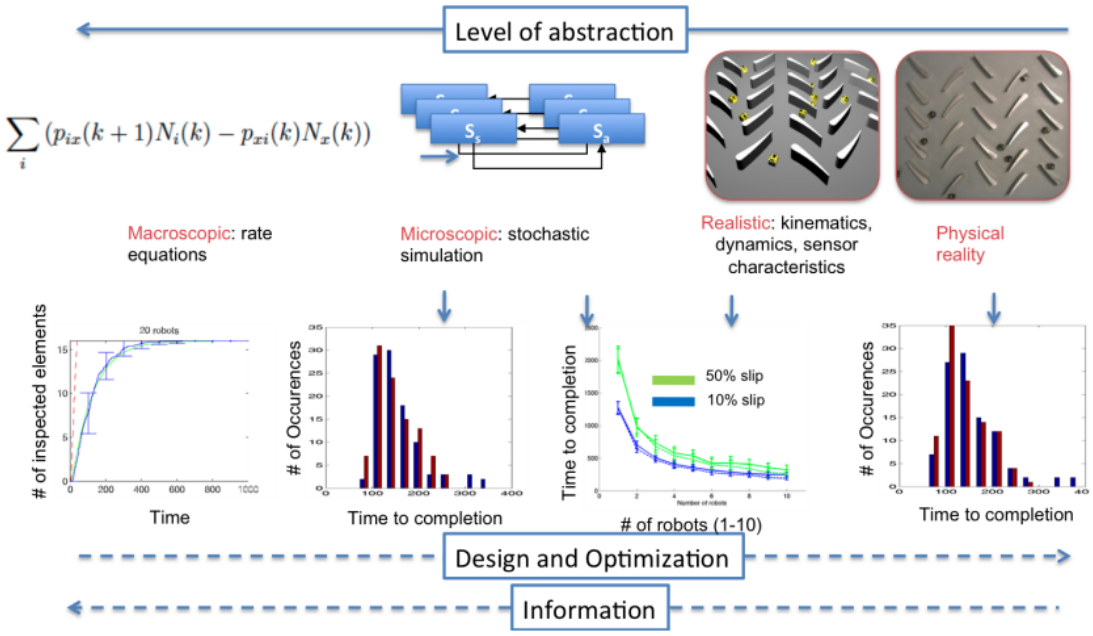


Figure 7

We have now discussed all the of the steps needed to design a robot controller and its corresponding non-spatial models for a particular swarming task. Figure 7 gives us a visual representation of the entire process including the modeling methods used at different levels of abstraction of the swarm system.

Lower abstraction levels (right to left in the figure) inform higher abstraction levels by means of calibration and system identification. In turn, higher abstraction levels can be used for designing and optimizing (as we will see in the next section) lower level models and systems. The bottom row of the figure shows selected results (from left to right) for comparison of rate equation prediction (green) with real robot experiments (blue), the distribution (histogram) for the same data based on stochastic simulation for coordination with (red) and without (blue) communication, the impact of varying amounts of wheel-slip in the Webots simulator (solid lines) and stochastic simulation (dashed lines), and the performance distribution of the real system that closely resembles that obtained from stochastic simulation.

This design methodology is a closed loop feedback system. Setting up the required

swarming task as a real physical experiment or physical simulation allows us to discover and measure the different free parameters in the system. We then use these variables in lower resolution models (micro-level) as well as in the development of mathematical models (macro-level) for our system. These, more abstract models allow us to rapidly experiment with system parameters and optimize them. We then use these optimized parameters back in the real physical experiments to improve the desired behavior of the system and repeat the cycle till the desired level of accuracy and satisfaction of model behavior is achieved. Finally, we hope to deploy these swarm systems in the real world to tackle real problems using the novel yet algorithmically simple approach that swarm robotics lends us.

3.7 Optimization

One of the major advantages of the methodology described in this section is the ability to identify and optimize important system parameters. We can re-write the PDM equations in (7) as,

$$S'(\theta, \mu, t) = \Gamma\left(\vec{S}, \theta, \mu, t\right) \tag{8}$$

So the temporal evolution of a model is characterized by its state in the past, $\vec{S}(t)$, partially known system parameters (θ), robot control parameters (μ), and time. In other words, the probabilities in eq.(7) and eq.(2) need not be constant values but functions of θ, μ, and t.

Two situations commonly arise when designing macro-level models for swarm systems. Firstly, there is the problem of identifying the important system parameters, θ. As pointed out in[14], *"In a real robotic system, not all of the model parameters can be known beforehand, either due to uncertainty of measurements or because the chosen model oversimplifies the system (e.g., ignoring friction or sensor noise)."* As such, these parameters are estimated from observations of real experiments and require solving the following optimization problem.

$$\theta^* = \arg\min_{\overline{\theta}} \left(S^*(\overline{\theta}, \mu) - \hat{S}^*(\theta, \mu)\right)^2 \tag{9}$$

$S^*(\overline{\theta}, \mu)$ is the model prediction of the state vector at steady-state,
i.e. $S^*(\overline{\theta}, \mu) = \lim_{t\to\infty} S(\overline{\theta}, \mu, t)$, and $\hat{S}^*(\theta, \mu)$ is the experimentally observed state vector at steady-state.

The second situation involves finding optimal control parameters, μ, for a known set of system parameters, θ. The goal is to drive the system towards a desired steady-state distribution, $\tilde{S}^*$, by tuning the model's control parameters. The is accomplished by solving the following optimization problem.

$$\mu^* = \arg\min_{\overline{\mu}} \left(\tilde{S}^* - S^*(\theta, \overline{\mu})\right)^2 \tag{10}$$

It is evident from eq.(9) and eq.(10) that these separate optimization problems are not independent. Starting with initial guesses $\mu(0)$ and $\theta(0)$ alongside simulation results from

these guesses, we generate increasingly better values of θ and μ for our desired final state distribution $\tilde{S}^*$ using the following recurrence relations.

$$\overline{\theta}(n+1) = \arg\min_{\overline{\theta}} \sum_{i=0}^{n} \left(S^*(\overline{\theta}, \overline{\mu}(i)) - \hat{S}^*(\theta, \overline{\mu}(i), i) \right)^2 \tag{11}$$

$$\overline{\mu}(n+1) = \arg\min_{\overline{\mu}} \left(\tilde{S}^* - S^*(\overline{\theta}(n+1), \mu) \right)^2 \tag{12}$$

The index n corresponds to the number of the experiment that led to an observation of the steady state $\hat{S}^*(\theta, \mu, n)$. Further discussion and applications of this parameter optimization approach are discussed in[16, 14].

4 Non-Spatial Model Applications

This section provides a brief run-down of some well studied swarm robot applications and the important results ascertained from them. While this list is by no means a comprehensive evaluation of every single swarm robot algorithm ever developed, the featured algorithms share a development model similar to the one described in the previous section, consisting of a PFSM and the ODE (or Difference Equation, DE) system that defines their macroscopic behavior. The final model described in this section is based on our ongoing research work on multi-robot coordination using a response threshold function.

4.1 Object Clustering

Object clustering, or aggregation, is one of the early applications for robot swarms inspired from transport techniques seen in ant colonies. It was introduced in [43] as an example of collective, non-cooperative behavior. The task involves moving objects that have been scattered over a region and collecting them in a pile without explicit communication or global positioning. It is an example of non-cooperative behavior since every agent is capable of moving objects on its own and thus no cooperation is required between individuals to successfully complete the task.

Agassounon, Kalra and Martinoli introduce worker allocation via threshold functions as a variant to the original object clustering experiment in [30, 3] and expand the results in [1]. Here, agents decide whether or not to help with the global task (stick collecting) based on their local perception of the degree of completion of the task. While earlier attempts at threshold based collective algorithms for robots suffer from scalability issues arising from explicit communication [47] between agents or the need for a centralized supervisor [31], the collaborative approach presented in [3] employs a wholly distributed strategy where agents use only local observations to make work vs. rest decisions.

Figure 8: Webots simulation of Khepera robots performing object clustering. Left image shows the initial placement of robots/sticks in the arena while the right image shows the successful completion of the task.

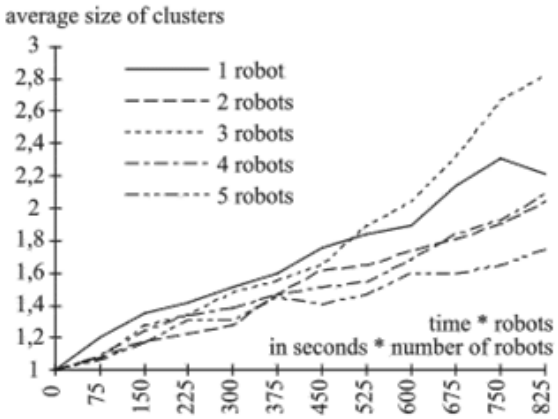


Figure 9: Results from the original object clustering task without threshold based worker allocation emphasizing the non-cooperative nature of the task, i.e. more robots do not necessarily complete the task faster.

Results

It is shown via physical experiments in [43] that adding agents to the task only speeds up the rate of collection to a certain threshold, and that overall efficiency is not necessarily maximized by adding more agents (as seen in Figure 9).

For the sake of brevity, results from the threshold based worker allocation variant of this experiment have been excluded from this discussion but can be found in [3, 1, 2].

4.2 The Stick Pulling Experiment

Figure 10: A real experimental setup for the 2-robot stick pulling experiment showing a team of two robots successfully pulling a stick out of the ground.

Introduced alongside object clustering in [43]—later extended and studied in further detail [41, 42, 28, 35, 40]—the aim of this experiment is to pull long sticks out of the ground using pairs of collaborating robots. Each stick is placed deep enough into the floor that

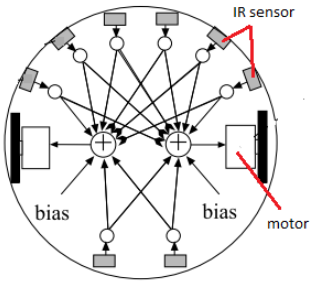
a single robot cannot successfully pull it out. Therefore, the agent's goal is to first find a stick and then either wait at that stick if another robot is not already at it, or to help the robot currently at the stick.

The robots have an internal timer that begins to tick down as they wait at a stick. This parameter is referred to as the *gripping time* or *wait time* in the experiment. If a fellow robot does not come to help before the gripping time expires then the robot lets go of the stick it is currently at and tries to find another stick. Adjusting the gripping time for each agent non-trivially affects the overall outcome the collaborative task. Once a stick is successfully pulled out of the ground by a coalition of two robots, they perform a *success dance* and then break their coalition. Each robot then returns to the search state.

Sticks that are pulled out of the ground are replaced by a human so that the total number of sticks to pull does not change during the course of the experiment. Likewise, the number of agents is constant over course of the experiment. The goal of the experiment is to maximize the rate of collaboration between robots or, in other words, maximize the number of sticks being pulled out of the ground per unit time.

The robots used for this task are called *Khepera* robots. The technical specification of the robot are given in Table 1. The only modification made to the original hardware design for this experiment is that each robot is outfitted with a single-joint gripper arm so that it can grip and lift sticks, as seen in Figure 11a.

(a) The Khepera-II robot used in the *Stick Pulling Experiment* by Martinoli et. al.



(b) Neural net that controls sensory feedback on the Khepera-II robot for the stick pulling experiment.

Figure 11

Results

A particularly interesting insight derived from the classical stick pulling experiment and studied further in [28] is the problem of wait-time parameter optimization. As mentioned earlier, in section 3.7, parameter discovery and optimization are recurring tasks associated with modeling swarm systems. The sick pulling experiment provides a good practical

Diameter	Height	Mass	Velocity
55mm	30mm	80g	0.02 to 1.0ms^{-1}
Battery Life	CPU	RAM	EEPROM
45min	Motorola 68331 @ 16MHz	256KB	512KB
OS	Motors	Sensors	
μKOS (RTOS)	2 DC brushed servos	8 IR proximity and ambient light	

Table 1: Hardware specifications of the Khepera-II robots used in the object clustering and stick pulling experiments.

experiment where optimization of swarm systems can be studied—the important system parameter being gripping-time of the agents.

It is found that for the case where two robots are needed to pull out a stick, an optimum wait-time exists for maximizing the rate of sticks pulled when the number of robots is less than or equal to the number of sticks in the arena (see Figure 12). Intuitively, if we have more robots than sticks, the best strategy for a robot at a stick is to just wait forever since there will be at least one robot still searching at all times. On the other hand, if there are fewer (or equal) robots than sticks, the strategy of waiting forever could cause a deadlock situation in the arena where every robot is at an individual stick waiting for a partner to arrive.

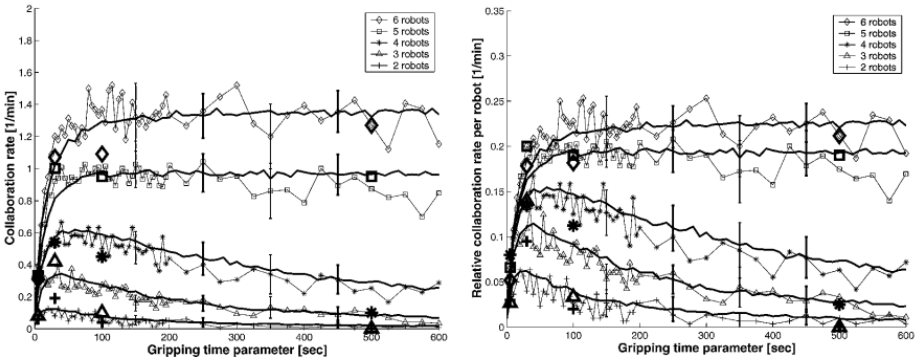


Figure 12: Left: Collaboration rate as a function of the gripping time in homogeneous groups of robots. The large single markers correspond to the results with the real robots, the linked small markers to those with the Webots simulator, and the underlying continuous lines to those with the probabilistic simulation. Right: Relative collaboration rate per robot (i.e., the average number of collaborations over time to which one robot participates by either making a first or second grip). Error-bars correspond to $\pm$ standard deviations of the results with the probabilistic model (thicker bars) and with Webot (thin lines). For reasons of clarity, only some errorbars are shown and results are only shown for gripping time parameters up to 600s.

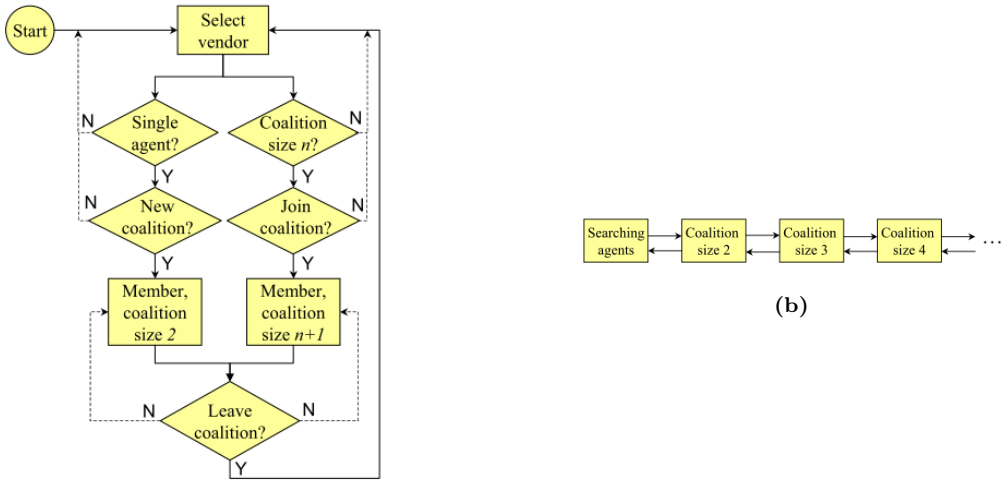


Figure 13: The flowchart and associated FSM used to describe the robot controller for coalition formation experiments are discussed by Lerman et al.

4.3 Coalition Formation

A practical swarm optimization strategy for e-commerce applications was introduced by Lerman et al. in [37, 34, 33]. Given a set of sellers for a specific product, the goal of the agent (or buyer) is to minimize cost per unit by forming coalitions with other buyers to get group discounts.

A coalition between two or more buyers is formed if they are all purchasing the same product from the same seller. Agents are free to join or leave a coalition on their own accord but entire coalitions are not allowed to switch sellers. Discounts given to buyers are directly proportional to the size of the buying coalition. Due to storage or other constraints, sellers only keep a certain amount of the product for sale which, in turn, restricts coalition sizes. A product is kept on sale for a pre-determined period of time buy a seller after which all transactions are completed and the product is taken off the market. The flowchart and FSM for the process described above is seen in Figure 13.

Results

This application has been mainly used to illustrate the use of differential equation, macro-level models for studying multi-agent systems[34, 33]. The analysis provided explains how even a "greedy" agent based approach can lead to maximal utility gains for the entire system. This is a recurring property of many swarm systems—decisions are often made by an individual agent based only on local information but the effect of the decision is felt globally throughout the system.

Figure 14 above shows the results obtained from solving the macro-level equations

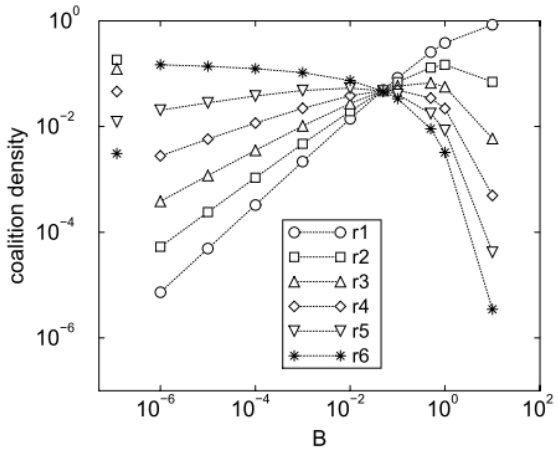


Figure 14: Plot showing utility gain in the coalition experiment with group sizes ranging from 1 to 6 per coalition and varying attachment/detachment rates

associated with this experiment for coalition sizes of 1–6 robots. Discussion of this graph in [37] suggests that there is only a small utility gain in the case where agents are allowed to form coalitions but not leave them. However, there is a very large utility gain in the case where agents can both, form and leave coalitions at will. The caveat is that attachment and detachment rate must be kept very low. For large detachment rates, there is again virtually no utility gain, as the system is composed mainly of unaffiliated agents.

Utility gain is a measure of profit per agent computed by,

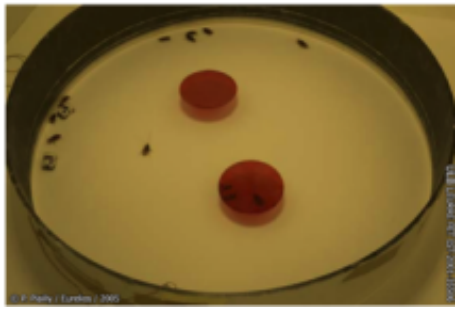
$$G = Np - \sum_{n=1}^{m} p_n n r_n$$

where N is the total number of agents, p_n is the coalition price dependent on the coalition size ($p_n = p - \Delta p(n-1)$, Δp = price decrement, p = base price), and r_n is the number of coalitions of size n.

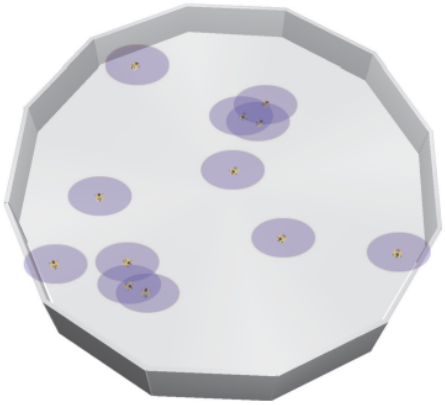
4.4 Cockroach Aggregation

In [29], Jeanson et al. showed that a species of cockroach, *Blattella germanica*, exhibited clustering mechanics to form groups using only local information and without knowledge of the global structure of the environment or group sizes. An agent based, biological model is developed and described in the above paper where simple action rules are extrapolated from watching real cockroach aggregation behavior in a closed environment under different lighting conditions.

The data collected from real cockroach aggregation inspired the development of a robotic model for aggregation, as seen in Figure 16 [17]. The probabilities for leaving



(a) A snapshot of the real cockroach aggregation experiment conducted by Jeanson et al. The cockroaches cluster under red protective covers that block light.



(b) A simulation of the aggregation experiment using robots with communication ranges shown as blue circles surrounding each agent.

Figure 15

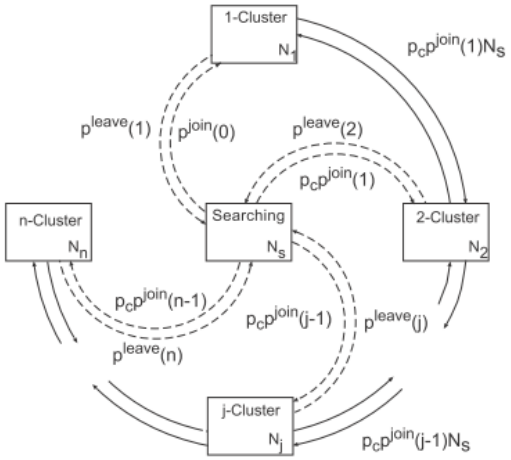


Figure 16: The PFSM the describes the robot controller for the aggregation experiment. The probabilities shown are probabilities of leaving a cluster of size $n-1$ to join a cluster of size n and vice-versa, as well as the probability to leave a cluster entirely and go back to searching.

and joining individual clusters are based exclusively on perceived cluster sizes and the time elapsed since joining a cluster. Global information such as actual cluster sizes are not available to any individual agent. Experiments using miniature *Alice* robots were conducted where the robot controller was developed using observed cockroach behavior in [29].

Results

In [18] the authors theorize that the size of clusters formed by robots can be controlled simply by adjusting their communication range. Specifically, "For the system with dynamics found in [... eqns representing the robot controller as seen in Figure 16], the system behavior can change between dispersion and aggregation simply by tuning p_c [encountering probability], as long as the inequalities for $p^{join}(j)$ and $p^{leave}(j)$ [probability to join/leave a cluster, respectively] are satisfied."

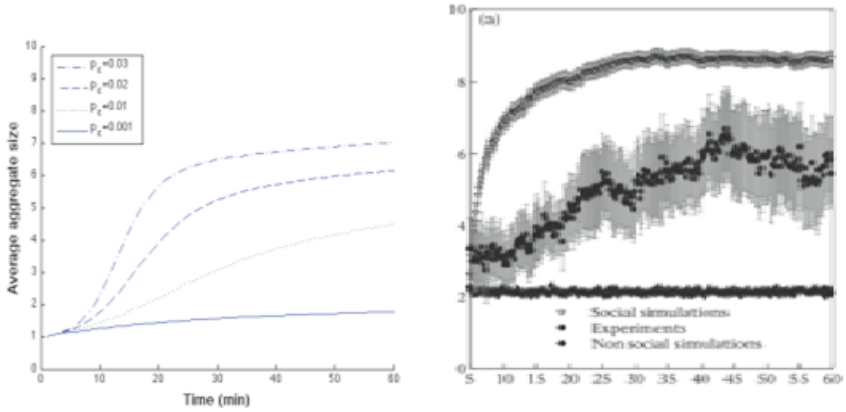


Figure 17: Left: Results of varying communication range and solving macro-model simulation of cockroach aggregation experiment using robotic agents. Right: Results of real cockroach aggregation obtained from experiments and simulation by Jeanson et al. The similarities between both plots are apparent.

Figure 17 shows plots of average cluster sizes observed over time for both, the robotic experiment (left) as well as the experiment involving real cockroaches (right). The robot experiment contains a range of plots for p_c between 0.001 and 0.03. We can easily see that the cluster sizes can be made to match the real cockroach experiment just by adjusting the value of encountering probability, which is in turn a function of the communication range of the robots. Conversely, there is the interesting parameter estimation/optimization problem of finding the right p_c to get the desired global behavior (cluster size) from the swarm.

4.5 Collective Inspection Of Regular Structures

A practical application for swarm systems is proposed by Correll and Martinoli in[15, 13]. The inspection of airplane jet turbine blades has, so far, been performed manually using borescopes. This procedure is both time and cost intensive and is subject to human error. Automating the task has proven difficult since the shielded, complex, narrow structure of a turbine imposes not only strong miniaturization constraints on the design but also prevents the use of any traditional global positioning and communication system. Furthermore, a limited on-board energy budget might prevent computation of a sophisticated deliberative

Figure 18: A simplified (unfolded) jet turbine inspection arena with marked blades and Alice robots performing inspection.

planning strategy and dramatically narrows the sensor and communication range of our robots. [15]

These constraints on the inspection task make it well suited for a miniature robot swarm/multi-agent approach. Since jet turbine blades are regular, repeating structures they provide a good inspection target for robot vision systems.

Results

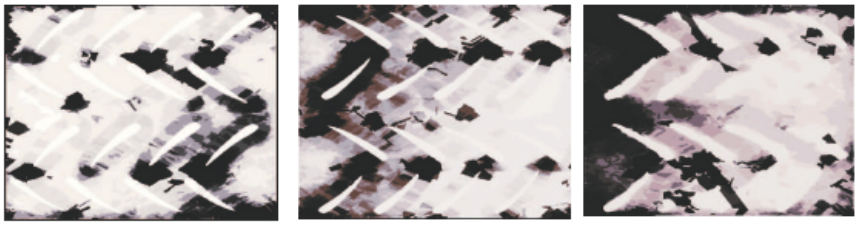


Figure 19: Distribution of the robots on the arena (dark areas correspond to frequently visited regions). Each experiment lasted 2h with 20 robots. Left: Default setup with the default inspection algorithm. Center: Set-up rotated of 180 degrees. Right: Rotated setup and robots characterized by a random walk behavior.

Simulation results seen in Figure 19 show that the probability distribution (agent dispersion and coverage pattern) is non-uniform both, due to the shape of the blades, and border-following algorithm used by robots. It is evident that simply rotating the orientation of the blades results in a different coverage pattern of the arena, by the robots.

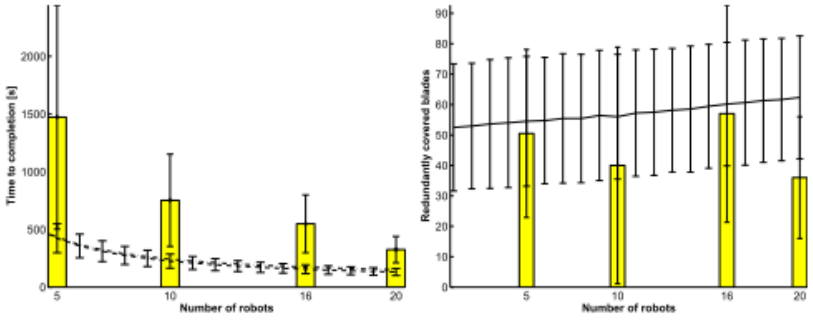


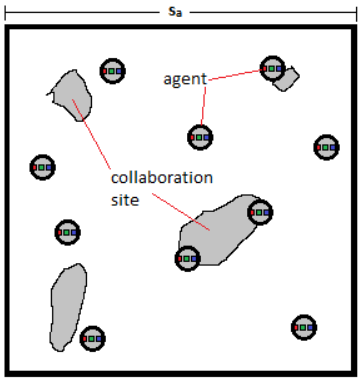
Figure 20: Comparison of microscopic and macroscopic model predictions (basically superimposed) and experimental results. Experimental results are represented by the median and standard deviation because of long tail distributions while microscopic models by mean and standard deviation. Left: Time to completion (16 blades) vs. team size (1 to 20 robots, 5, 10, 16, 20 robots respectively). Right: Number of blades redundantly covered until inspection completion vs. team size.

As a result of the inherent spatial nature of the experiment, it is found that both micro-level simulations and macroscopic equations derived for this setup (using the non-spatial methods we have discussed so far) only prove to be qualitatively correct for small robot teams (see Figure 20). A more detailed analysis of the model vs. real experiment mismatch is provided by the authors in [15, 19]. We shall see a spatial modeling based approach for the same experiment in the next section.

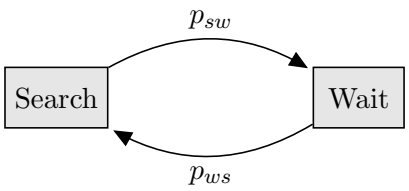
4.6 Multi-Robot Collaboration Using A Response Threshold Function

The work by Lerman et. al [35] reinforces results obtained for stick pulling in [28, 41, 42]. It also extends the original experiment to include the case where more than two robots can be used to pull sticks of variable lengths out of the ground. We build upon this work to introduce a more general algorithm for multi-agent collaboration based on sigmoid response threshold functions, motived from task allocation in insects.

All the collaborative tasks discussed so far are characterized by a binary outcome—success or failure to perform the group task assigned, e.g. pulling a stick out of the ground, or failing to do so. We wish to extend this model to include a more general case of collaboration tasks that involve only approximate number of robots for successful collaboration. Examples of such tasks include collective transport [53, 54], pattern recognition [9], real-time mapping of oil spills [7], determining coverage area of forest fires [32] aerial swarm surveillance, and many others that require a subset of a swarm to coalesce to tackle a task collaboratively. It is important to note that we do not study any particular task in detail but instead, outline a general approach to modeling scenarios with the aforementioned properties. In such cases, so long as we have at least one robot, the task may be accomplished with varying degrees of success. The result is now an accuracy function of the number of robots collaborating on the task, among other environmental factors.



(a) A visual representation of the collaboration experiment being studied. Objects in the image are not to scale.



(b) Robot controller used to drive group collaboration. p_{sw} is the probability of going from the search state to the wait state, while p_{ws} is the probability of going to wait to search.

We consider a generic collaboration task with M uniformly distributed collaboration sites within a flat arena with side, s_a. A swarm of individually simple robots is deployed within the arena, also uniformly and at random as shown in Figure 21a.

The number of robots being used per experiment varies, as we discuss results for a number of different scenarios. The collaboration sites in the arena can be of various sizes and configurations—the only value that is of importance to the model is the total area of all collaboration sites within the arena.

The objective of each agent in the robot swarm is to find a collaboration site in the arena and perform a collective task with other agents at that site. Once this collaboration is complete, the robot detaches itself from its current collaboration site and repeats the search-wait cycle. The precise details of the collective task are not important for the purpose of understanding the coordination mechanism, and we assume collaboration to happen instantaneously for the purpose of the model studied in this paper. The robot controller that describes individual agent behavior for this swarm system is shown in Figure 21b.

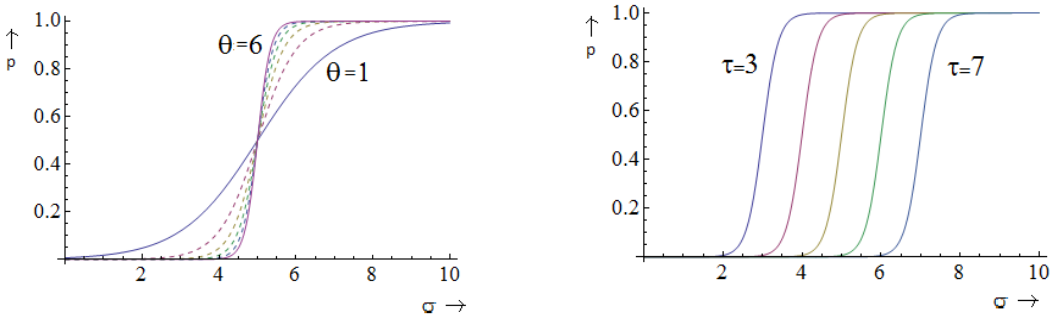
The population dynamics equations and the threshold function used for the model are described below. The associated PFSM is seen in Figure 21b

$$p_{ws}(\sigma) = \frac{1}{1 + e^{\theta(-\sigma+\tau)}} \tag{13}$$

$$N_s'(t) = N_w(t)p_{ws}(\sigma) - N_s(t)p_{sw} \tag{14}$$
$$N_w(t) = N_0 - N_s(t) \tag{15}$$
$$N_s(0) = N_0 \tag{16}$$
$$N_w(0) = 0 \tag{17}$$

In equation (13) control parameter θ controls the slope of the sigmoid function, while τ controls its offset along the σ axis, as seen in Figure 22. The input-σ to the probability

function p_{ws} within an individual robot controller is the number of agents currently at the same collaboration site as the robot. But group sizes at collaboration sites are not tracked in the macro-model and so we re-define σ as the average group size at time t, i.e. $\sigma(t) = N_w(t)/M$. Within the context of the swarming task, τ is the desired collaboration threshold value, while θ controls the variance or degree of flexibility we want the group sizes to have.



(a) Increasing θ in the sigmoid function increases it's slope at the point $\sigma^* = \tau$, $p_{ws}(\sigma^*) = 0.5$.

(b) Changing the τ parameter in the sigmoid function, offsets the curve along the σ axis.

Figure 22

Results

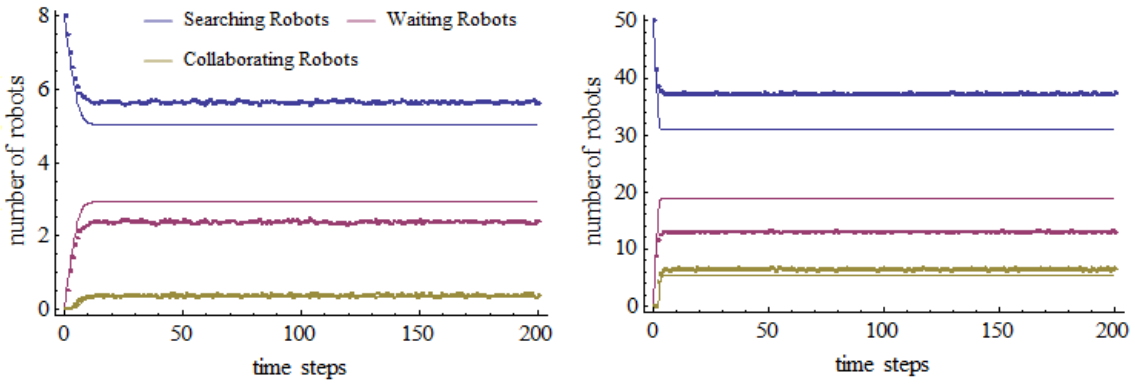


Figure 23: Macro-level numerical solutions and micro-level Gillespie simulation data plots for the proposed collaboration model. Left: $N_0 = 8$, $M = 2$, $\theta = 4$, and $\tau = 2$. Right: $N_0 = 50$, $M = 5$, $\theta = 5$, and $\tau = 4$. Data plotted for 200 seconds of simulation time and 100 runs each.

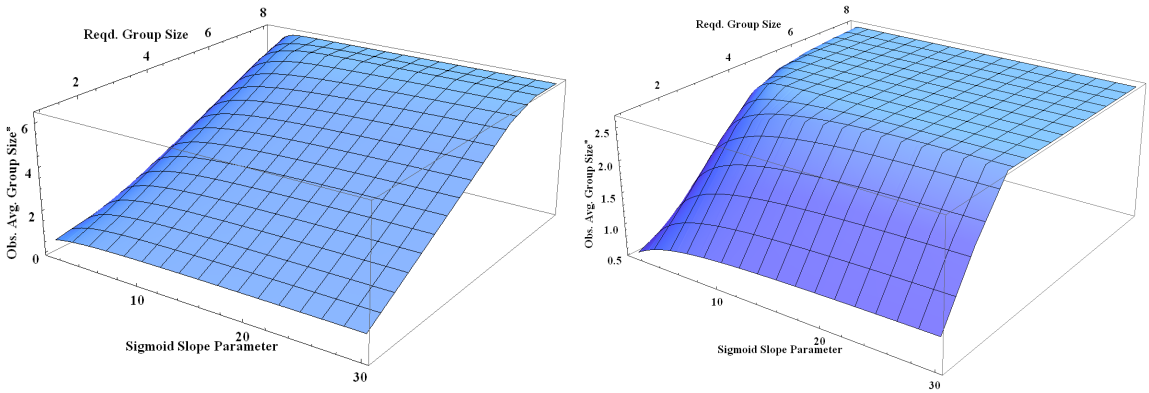
We numerically solve the rate equations with the given initial condition that all robots start off in the search state((14) to (17)), for $N_s(t)$, $N_w(t)$ and $R_c(t)$, from $t = [0, 200]$. Here, $R_c(t)$ is not a state variable of the robot controller but is an aggregated value that

measures the collaboration rate of the system at each time step and is mathematically described by

$$R_c(t) = N_w(t) p_{ws}(\sigma) \tag{18}$$

i.e., the number of robots going from the wait state to the search state at time t.

Numerical solutions for the rate equation are shown in Figure 23, along with data from the microscopic simulation for the same quantities. The two plots show that the system approaches a steady state distribution of agents for each of the two states. We perform the macro-level solution and the micro-level experiments with two sets of system parameters. The left plot in Figure 23 shows results from experiments conducted using 8 robots, 2 collaboration sites, expected group sizes $\tau = 2$, and the sigmoid slope parameter $\theta = 4$. The right plot in Figure 23 shows results using $N_0 = 100$ robots, $M = 5$ collaboration sites, $\tau = 5$ expected group size, and $\theta = 4$. The data points for the microscopic model are averaged over a hundred runs per point.



(a) $M = 1$ **(b)** $M = 3$

Figure 24: Numerical solutions for the rate equations (14) to (17) with $N_0 = 8$ and $t = [0, 200]s$ as we sweep the sigmoid control parameters, θ and τ.

Figure 24 showcases the macro-level behavior of the swarm system when we sweep across θ and τ, numerically solving the rate equations. As the slope of the sigmoid at σ^* gets higher, we expect to see system behavior and average group sizes resembling that as if a step function was used instead of the sigmoid probability function, i.e. for $\theta \gg 1$,

$$p_{ws}(\sigma) \approx \Theta(\sigma) = \begin{cases} 1 & : \sigma \geq \tau \\ 0 & : \text{o/w} \end{cases}$$

The sloped portion of the graphs reinforces this assertion that setting higher values for θ does indeed result in average observed group sizes being close to the requested collaboration threshold using the τ parameter. As the slope parameter, θ, is reduced, we see a marked drop in the correspondence between observed vs. requested group sizes. We also see that

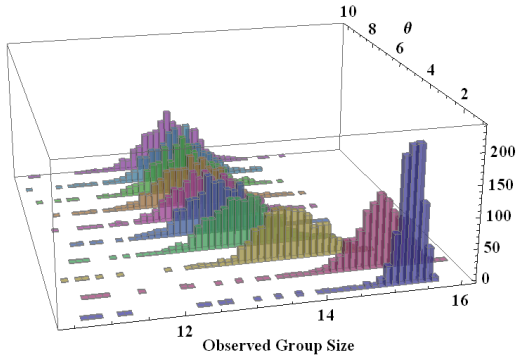


Figure 25: Relative distribution of robot group sizes as θ is varied, while keeping all other parameters constant.

increasing θ past a certain point results in no major observed vs. requested group size shift.

Finally, we are interested in studying the distribution of group sizes during the course of the experiment, as the sigmoid slope parameter is varied. Figure 25 shows a 3D histogram of observed group sizes for increasing θ values, from 1 to 10, over 200 seconds of Gillespie simulation time, using a 100 robots, 5 collaboration sites, and a threshold value of 20. All values a averaged over 100 simulation runs.

5 Spatial Models

So far we have discussed characterization robot controllers and their corresponding PFSMs using rate equations. As Hamann and Wörn[26] point out, "A drawback of these rate equations is the limited representation of space—either homogeneous space is assumed or a coarse discretization by states is done." Indeed, during our discussion of macro-model development we assumed a uniform starting distribution of robots and interaction objects (such as sticks) in an arena. We also defined random walk as being the process by which this uniform distribution of agents was maintained during the course of the experiment and completely ignored physical properties such as the maximum distance a robot could move per time step due to it's velocity. The process of random walk effectively allowed us to *teleport* robots within the arena at any time, thereby simplifying our mathematical model to a system of ODEs. While these assumptions are justified because our aim in designing the macro-model is to study the overarching properties of the swarm system without worrying about the dynamics of individual agents, such non-spatial descriptions of the swarm system are bound to introduce unintended side-effects (such as *teleportation*) in our results. Non-spatial models also make modeling of agent interactions such as collisions and ranged communication more complicated. To account for these inaccuracies in non-spatial models, the concept of spatial modeling of swarm robot systems has evolved into a research topic in the last decade.

Spatial models for studying robot swarms have been adopted from older, well-studied models in physics. Specifically, spatial models of swarm systems are heavily influenced from models of Brownian motion of particles and share a lot of the same mathematics and theory. A brief history of the mathematical solution of Brownian motion follows as it sheds light on the evolution of spatial models over time.

Robert Brown first discovered the phenomenon of Brownian motion in 1828 when he observed seemingly random movements of particles within vacuoles of pollen grains. The term "Brownian Motion" was coined later, in his honor, and is used to describe the motion of larger particles due to their constant collisions with smaller molecules of the medium within which they are suspended, e.g. The random movement of dust particles in still air, crushed tea leaves in a glass of water, etc. It wasn't until the turn of the next century when, in 1905, Einstein mathematically solved the phenomenon of Brownian motion. Independent solutions were subsequently discovered by Smoluchowski (1906) and Langevin (1908). The Langevin equation is used as the basis for swarm system micro-models as it is a simpler and more adaptable approach than the other two. Then in 1914 and 1917, Fokker and Planck respectively analyzed the probability distribution of the velocity of diffusing particles by deriving a PDE, now well known as the Fokker-Planck equation. This equation is used as the basis for a macro-model in swarm system analysis. The Fokker-Planck equation is also sometimes known as the Kolmogorov forward equation as it is a special case of a more general equation used to describe time and state continuous Markov processes,

invented by Kolmogorov in 1931. Finally, in 2003 Schweitzer introduced the concept of free agents or active particles experiencing Brownian motion, capable of producing energy internally and using it for self propulsion; much like swarm robots. For these reactive agents, generalized Langevin and Fokker-Planck equations have been subsequently developed and we will discuss them next.

5.1 Langevin Equation: A Microscopic Model

Like non-spatial models, spatial models for robot swarms can be split up into two system resolutions — the micro-level and the macro-level. We will begin our discussion of spatial modeling at the micro-level using a Langevin equation because the macro-level Fokker-Planck equation can be derived from it. It is important to note that a Langevin equation is any differential equation that describes the temporal and spatial evolution of a subset of the degrees of freedom of a particle, i.e. it is not a single equation but a class of equations that all describe the same phenomenon. The original equation for a Brownian agent by Langevin was of a different form than the one we will be using to study robot swarms as it did not take into consideration the case where an agent could move due to some internal driving force, as robots do. A derivation from Langevin's original equation to the modified form seen below is available in Hamann's dissertation work[24].

$$\mathbf{R}'(t) = -\mathbf{A}(\mathbf{R}(t), t) + B(\mathbf{R}(t), t)\mathbf{F}(t) \tag{19}$$

In Eqn. (19), $\mathbf{R}(t)$ represents the position of the agent at time-t, $\mathbf{A}(\mathbf{R}(t), t)$ is a direction function describing the agent's deterministic motion, while $B(\mathbf{R}(t), t)\mathbf{F}(t)$ describes the agent's random motion based on $\mathbf{F}$, a stochastic process (white noise). $\mathbf{A}$ is commonly referred to as the *drift* term in the equation and B, the *diffusion coefficient*. $\mathbf{F}(t)$ is a normalized noise term ($\|\mathbf{F}(t) = 1\|$) with 0 mean and is uncorrelated in time ($\langle \mathbf{F}(t)\mathbf{F}(t') = \delta(t - t') \rangle$). Using constant values for drift and diffusion in 1 spatial dimension, Figure 26 shows the movement of a robot (or particle) solved using Eqn. (19).

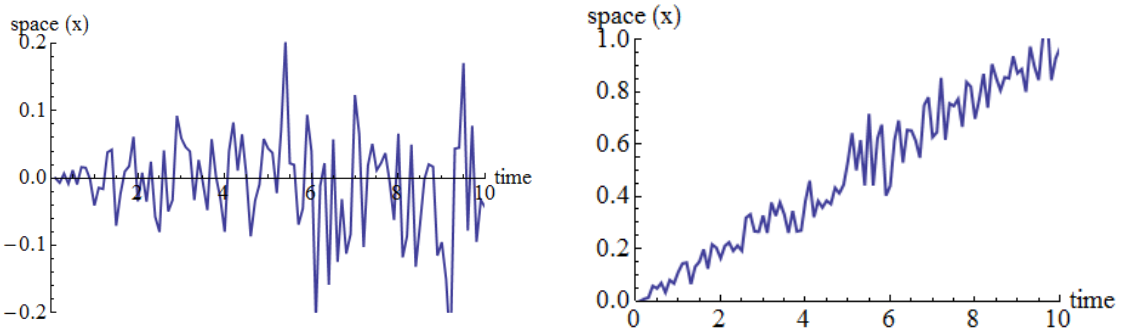


Figure 26: Left: Solution to Langevin equation for a particle in 1D with $\mathbf{A} = 0$, i.e. no drift. Right: Solution with $\mathbf{A} = 0.1$. In both cases the diffusion coefficient is constant, $B = 0.3$, and $\mathbf{F}$ uses an uncorrelated normal distribution with mean $\langle \mathbf{F} \rangle = 0$.

5.2 Fokker-Planck Equation: A Macroscopic Model

From the Langevin equation used in the previous section, we can derive the Fokker-Planck equation as stated below,

$$\frac{\partial \rho(\mathbf{r},t)}{\partial t} = -\nabla(\mathbf{A}(\mathbf{r},t)\rho(\mathbf{r},t)) + \frac{1}{2}Q\nabla^2(B^2(\mathbf{r},t)\rho(\mathbf{r},t)) \tag{20}$$

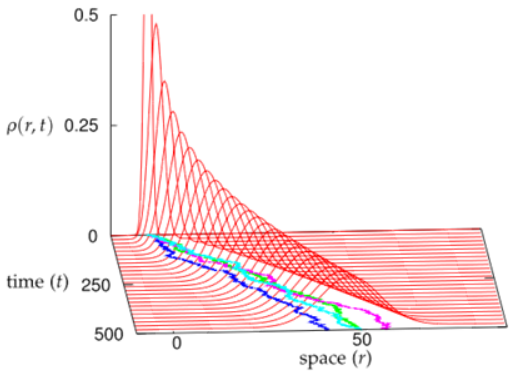


Figure 27: A visual comparison between the spatial micro vs. macro-level equations. The colored lines on the r-t plane show the paths taken by individual particles, found via the Langevin equation, while the overarching red curve is the probability distribution for the entire ensemble as described by the Fokker-Planck equation.

A clear and concise derivation of the above equation from the Langevin equation is provided in section 4.1 of [24], which in turn follows closely from the book *Synergetics* by Haken (1977). The terms $\mathbf{A}$, B and $\mathbf{F}$ reprise their earlier roles from the Langevin equation, while Q is a new displacement term due to collisions between particles. $\rho(\mathbf{r},t)$ is the probability of encountering a particle at position $\mathbf{r}$ and time t. As one can see, the Fokker-Planck equation gives us a *probability distribution* of the positions of particles in space, making it a viable macroscopic model compared to the micro-level Langevin equation which gives us the *position* of a *single* particle in space (see Figure 27).

5.3 Applications

The Fokker Planck equation is a relatively new tool in the field of swarm robot modeling and has notably been used by Hamann[26, 24], as well as Prorok, Correll and Martinoli[49] in their work on boundary coverage and inspection of jet-turbine engines using swarms of miniature robots. They used a hybrid approach for modeling collective inspection that involves a probabilistic non-spatial model coupled with spatial micro-macro diffusion equations for modeling collective inspection of regular structures to improve consensus with observed experimental values.

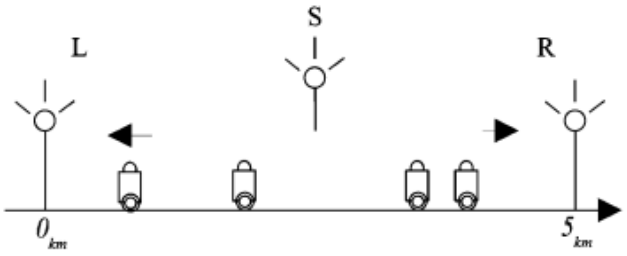


Figure 28: Experiment setup for centralized robot population control experiment proposed by Milutinovìc and Lima in[45].

A spatial model is used in[45] to control a large scale multi-agent system using centralized controller and three signal sources. The goal of this experiment is to show successful control of a large population of robots by collectively moving them along a 1-dimensional arena using *left*, *right* and *stop* signals from a moving aerial controller. The model accounts for the fact that not all robots receive the signal or change direction/move promptly upon successfully receiving a signal. A visual representation of this setup can be seen in Figure 28.

Although the Fokker-Planck equation is not mentioned explicitly in[45], the authors employ the use of a system of PDEs to predict the position of the swarm in the arena via a probability density function (PDF). Their approach is therefore rather similar to the macro-level spatial modeling approach described earlier in this paper. The goal of moving the entire swarm from point A to point B in a 1D arena just through the use three signals controlled by a centralized controller is formulated as an optimal control problem by the authors. It is solved by introducing appropriate weighting and cost functions to the PDEs and computing the PDF at any given time to predict the collective motion/position of the swarm. The control signals are then switched accordingly to move the swarm in a desired direction. Two particularly insightful plots (Figures 4 & 5) are provided in[45] that showcase this result, i.e. the PDF evolution of stopped and moving robot populations as time progresses.

While the non-spatial modeling approaches we have discussed so far use a phenomenological approach to construct the macro-model equations, this is not a viable option in the spatial model case. Typically, the drift and diffusion coefficient functions in the Fokker-Planck equation are adapted based on the swarming task we attempt to model but constructing these functions is not as straightforward as defining the rate equations from a robot controller, as seen in the non-spatial case. The $\mathbf{A}$ and B functions of the Fokker-Planck equation serve to bridge the gap between micro-level and macro-level observations of a swarm system and hence provide quite a challenge to design accurately.

6 Conclusion

This paper has attempted to brief the reader on non-spatial and spatial modeling techniques in swarm robotics, at different levels of abstraction. While non-spatial models such as Gillespie simulations and PDMs are widely used in the field today, there exists no "One method to rule them all" when it comes to modeling any and all real robot applications. There are always robot systems and multi-agent tasks that can be designed where the primary assumptions made in modeling them using the above mentioned methods break down completely. Whether it is due to the spatial nature of the task or other complex underlying processes caused by having to deal with non-perfect hardware, environmental factors, etc. there needs to exist a strong, mathematically provable basis upon which to decide whether or not it makes sense to use the modeling techniques discussed in this paper. Such a basis does not exist as yet which is perhaps why designing accurate models for particular tasks is still so difficult and grounded in phenomenological observations versus cleaner algorithmic approaches for tackling problems commonly seen in other fields of computer science.

That said, robotics and other cyber physical systems have unique property that other fields of computer science do not—direct interaction with the real world and the laws of physics that rule supreme here. The question then becomes, what properties of the real physical system can be safely abstracted away when designing models without sacrificing accuracy and concurrency with the real world? This question is especially important in swarm robotics as so many of its applications rely on emergent behavior within the system to accomplish complex tasks using simple algorithms and interactions between thousands of individual agents[25], e.g., self assembly of robot swarms into regular structures like bridges and honeycombs may not be realized in models that ignore the effects of friction.

With this motivation in mind, system identification and parameter discovery/estimation are areas within swarm robotics that I wish to explore further. Closely related to this is the task of robot control parameter optimization. As we saw in section 3.7, knowing the system parameters, control parameters and the relationship between them (see equation (11)) allows us to fully describe a swarm robot model. Optimization of the control parameters allows us to then control the behavior of the swarm, which is the end goal of this entire endeavor.

I am also interested in studying spatial models for robot swarms that we discussed briefly in section 5. Their non-reliance on the assumption of spatial uniformity of the swarm makes them immensely useful for studying tasks where the positions of agents are important, such as self-assembly, pattern formation [27], shape recognition [38], boundary coverage, swarm surveillance, etc. While system parameter estimation and control parameter optimization have not been widely explored for spatial models, I think it is an area of research that can be provide very useful inferences to the entire field of swarm robotics and perhaps result in real world applications of robot swarms.

References

[1] Agassounon, W., and Martinoli, A. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3* (2002), p. 1090–1097.

[2] Agassounon, W., Martinoli, A., and Easton, K. Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes. *Autonomous Robots 17*, 2-3 (2004), 163–192.

[3] Agassounon, W., Martinoli, A., and Goodman, R. A scalable, distributed algorithm for allocating workers in embedded systems. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on* (2001), vol. 5, p. 3367–3373.

[4] Şahin, E. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 10–20.

[5] Balch, T. Communication, diversity and learning: Cornerstones of swarm behavior. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 21–30.

[6] Bayazit, O. B., Lien, J.-M., and Amato, N. M. Swarming behavior using probabilistic roadmap techniques. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 112–125.

[7] Beni, G. From swarm intelligence to swarm robotics. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 1–9.

[8] Beni, G. Order by disordered action in swarms. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 153–171.

[9] Beni, G., and Wang, J. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

[10] Berman, S., Halász, d., Kumar, V., and Pratt, S. Algorithms for the analysis and synthesis of a bio-inspired swarm robotic system. In *Swarm Robotics*. Springer, 2007, p. 56–70.

[11] Billard, A., Ijspeert, A. J., and Martinoli, A. A multi-robot system for adaptive exploration of a fast-changing environment: Probabilistic modeling and experimental study. *Connection Science 11*, 3-4 (1999), 359–379.

[12] Caprari, G., Balmer, P., Piguet, R., and Siegwart, R. The autonomous micro robot "alice": a platform for scientific and commercial applications. In *Micromechatronics and Human Science, 1998. MHS'98. Proceedings of the 1998 International Symposium on* (1998), IEEE, pp. 231–235.

[13] Correll, N. *Coordination schemes for distributed boundary coverage with a swarm of miniature robots: synthesis, analysis and experimental validation.* PhD thesis, Ecole Polytechnique Fédérale, Lausanne, Switzerland, 2007.

[14] Correll, N. Parameter estimation and optimal control of swarm-robotic systems: A case study in distributed task allocation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (2008), p. 3302–3307.

[15] Correll, N., and Martinoli, A. Collective inspection of regular structures using a swarm of miniature robots. In *Experimental Robotics IX*. Springer, 2006, p. 375–386.

[16] Correll, N., and Martinoli, A. System identification of self-organizing robotic swarms. In *Distributed Autonomous Robotic Systems 7*. Springer, 2006, pp. 31–40.

[17] Correll, N., and Martinoli, A. 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] Correll, N., and Martinoli, A. Modeling and designing self-organized aggregation in a swarm of miniature robots. *The International Journal of Robotics Research 30*, 5 (Apr. 2011), 615–626.

[19] Correll, N., Rutishauser, S., and Martinoli, A. Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures. In *Experimental Robotics* (2008), Springer, pp. 471–480.

[20] Dorigo, M., Tuci, E., Gross, R., Trianni, V., Labella, T. H., Nouyan, S., Ampatzis, C., Deneubourg, J.-L., Baldassarre, G., Nolfi, S., Mondada, F., Floreano, D., and Gambardella, L. M. The swarm-bots project. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 31–44.

[21] Gillespie, D. T. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics 22* (1976), 403–434.

[22] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry 81*, 25 (1977), 2340–2361.

[23] Grasse, P.-P. La reconstruction du nid et les coordinations inter-individuelles chez bellicostitermes natalensis et cubitermes . sp. la theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs. *Insectes Soc 61* (1959), 41–81.

[24] Hamann, H. *Space-Time Continuous Models of Swarm Robotic Systems: Supporting Global-to-Local Programming*, vol. 9. Springer, 2010.

[25] Hamann, H., Schmickl, T., Wörn, H., and Crailsheim, K. Analysis of emergent symmetry breaking in collective decision making. *Neural Computing and Applications 21*, 2 (2012), 207–218.

[26] Hamann, H., and Wörn, H. A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intelligence 2*, 2-4 (2008), 209–239.

[27] Hsieh, M. A., and Kumar, V. Pattern generation with multiple robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (2006), p. 2442–2447.

[28] Ijspeert, A. J., Martinoli, A., Billard, A., and Gambardella, L. M. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots 11*, 2 (2001), 149–171.

[29] Jeanson, R., Rivault, C., Deneubourg, J.-L., Blanco, S., Fournier, R., Jost, C., and Theraulaz, G. Self-organized aggregation in cockroaches. *Animal Behaviour 69*, 1 (Jan. 2005), 169–180.

[30] Kalra, N., and Martinoli, A. Comparative study of market-based and threshold-based task allocation. In *Distributed Autonomous Robotic Systems 7*. Springer, 2006, p. 91–101.

[31] Krieger, M. J., and Billeter, J.-B. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems 30*, 1 (2000), 65–84.

[32] Krishnanand, K., Amruth, P., Guruprasad, M., Bidargaddi, S. V., and Ghose, D. Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (2006), IEEE, pp. 958–963.

[33] Lerman, K. Design and mathematical analysis of agent-based systems. In *Formal Approaches to Agent-Based Systems*. Springer, 2001, pp. 222–234.

[34] Lerman, K., and Galstyan, A. A general methodology for mathematical analysis of multi-agent systems. *ISI-TR-529, USC Information Sciences Institute, Marina del Rey, CA* (2001).

[35] Lerman, K., Galstyan, A., Martinoli, A., and Ijspeert, A. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life 7*, 4 (2001), 375–393.

[36] Lerman, K., Martinoli, A., and Galstyan, A. A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm robotics*. Springer, 2005, p. 143–152.

[37] Lerman, K., and Shehory, O. Coalition formation for large-scale electronic markets. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on* (2000), IEEE, pp. 167–174.

[38] Liu, L., Fine, B., Shell, D., and Klappenecker, A. Approximate characterization of multi-robot swarm "shapes" in sublinear-time. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), p. 2886–2891.

[39] Martinoli, A. *Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies*. PhD thesis, 1999.

[40] Martinoli, A., Easton, K., and Agassounon, W. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research 23*, 4-5 (2004), 415–436.

[41] Martinoli, A., Ijspeert, A. J., and Gambardella, L. M. A probabilistic model for understanding and comparing collective aggregation mechanisms. 575–584.

[42] Martinoli, A., Ijspeert, A. J., and Mondada, F. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems 29*, 1 (1999), 51–63.

[43] Martinoli, A., and Mondada, F. Collective and cooperative group behaviours: Biologically inspired experiments in robotics. In *Proceedings of the Fourth International Symposium on Experimental Robotics* (1995), Springer Verlag, pp. 3–10.

[44] Michel, O. Webots: Symbiosis between virtual and real mobile robots. In *Virtual Worlds* (1998), Springer, pp. 254–263.

[45] Milutinovi, D., and Lima, P. Modeling and optimal centralized control of a large-size robotic population. *IEEE Transactions on Robotics 22*, 6 (Dec. 2006), 1280–1285.

[46] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions* (2009), vol. 1, pp. 59–65.

[47] Parker, L. E. Alliance: An architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on 14*, 2 (1998), 220–240.

[48] Payton, D., Estkowski, R., and Howard, M. Pheromone robotics and the logic of virtual pheromones. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 45–57.

[49] Prorok, A., Correll, N., and Martinoli, A. Multi-level spatial modeling for stochastic distributed robotic systems. *The International Journal of Robotics Research 30*, 5 (Apr. 2011), 574–589.

[50] Rubenstein, M., Ahler, C., and Nagpal, R. Kilobot: A low cost scalable robot system for collective behaviors. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on* (2012), IEEE, pp. 3293–3298.

[51] Russel, S., and Norwig, P. *Artificial Intelligence: A Modern Approach*. No. Chapter 2. Prentice-Hall Inc, 1995.

[52] Seyfried, J., Szymanski, M., Bender, N., Estaña, R., Thiel, M., and Wörn, H. The i-swarm project: Intelligent small world autonomous robots for micro-manipulation. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 70–83.

[53] Sugawara, K., Correll, N., and Reishus, D. Object transportation by granular convection using swarm robots. In *Proc. of the Int. Symposium on Distributed Autonomous Robotic Systems (DARS)* (2012).

[54] Sugawara, K., Correll, N., and Reishus, D. Object transportation by granular convection using swarm robots.

[55] Winfield, A. F. T., Harper, C. J., and Nembrini, J. Towards dependable swarms and a new discipline of swarm engineering. In *Swarm Robotics*, E. Şahin and W. M. Spears, Eds., no. 3342 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2005, pp. 126–142.