AFRL-IF-RS-TR-2005-330
Final Technical Report
September 2005

MATHEMATICAL MODELING OF LARGE MULTI-AGENT SYSTEMS

University of Southern California at Los Angeles

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K359/00

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-330 has been reviewed and is approved for publication

APPROVED: /s/

ROBERT J. PARAGI
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	SEPTEMBER 2005	Final Jul 00 – Jul 05

4. TITLE AND SUBTITLE
MATHEMATICAL MODELING OF LARGE MULTI-AGENT SYSTEMS

5. FUNDING NUMBERS
C - F30602-00-2-0573
PE - 62301E
PR - TASK
TA - 00
WU - 05

6. AUTHOR(S)
Kristina Lerman,
Maja Mataric and
Aram Galstyan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Prime:
University of California at Los Angeles
837 West Downey Way
Los Angeles California 90089-1147

Sub:
University of California at Marina Del Rey
4676 Admiralty Way
Marina Del Rey California 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFSB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-330

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Robert J. Paragi/IFSB/(315) 330-3547/ Robert.Paragi@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
The biologically-inspired swarm paradigm is being used to design self-organizing systems of locally interacting agents. A major difficulty in designing swarms with desired characteristics is understanding the causal relation between individual agent and collective behaviors. Mathematical analysis of swarm dynamics can address this difficulty to gain insight into system design. This project developed a formal framework for mathematical modeling and analysis of multi-agent swarms. Though the behavior of an individual agent can be considered to be stochastic and unpredictable, collective behavior of a swarm can have a simple probabilistic description. We showed that a class of mathematical models that describe the dynamics of collective behavior of multi-agent systems can be written down from the details of the individual agent controller. We have successfully applied this formalism to study collective behavior of distributed robot systems for which a body of experimental and simulations data exists.

14. SUBJECT TERMS
Autonomous Agents, Software Agent System Dynamics, Intelligent Agents

15. NUMBER OF PAGES
77

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Contents

1 INTRODUCTION **1**
- 1.1 Background . . . 2
- 1.2 Stochastic Approach to Modeling Multi-agent Systems . . . 3
- 1.3 Modeling Adaptive Agents . . . 5
- 1.4 Modeling Spatially Correlated Systems . . . 6
- 1.5 Learning Approaches to Distributed Coordination . . . 9

2 SUMMARY OF PROJECT RESULTS **10**
- 2.1 Collective Behavior of Groups of Robots . . . 10
- 2.2 Distributed Resource Allocation . . . 12
- 2.3 Collective Mind Project . . . 13

3 Robotic Applications **13**
- 3.1 Collaboration in a Group of Robots . . . 13
  - 3.1.1 Real Robots, Embodied Simulations and Microscopic Modeling . . . 14
  - 3.1.2 Mathematical Model of the Stick-Pulling Experiments . . . 16
- 3.2 Optimal Group Size for Robot Foraging . . . 21
  - 3.2.1 Interference . . . 22
  - 3.2.2 Mathematical Analysis of Foraging . . . 23
  - 3.2.3 Comparison with simulations . . . 26
- 3.3 Dynamic Task Allocation . . . 29
  - 3.3.1 Mathematical Model of Dynamic Task Allocation . . . 32
  - 3.3.2 Comparison with Simulations . . . 34
- 3.4 Target Localization with Microscopic Robots . . . 34
  - 3.4.1 Mathematical Model of Target Localization Using Chemical Fields . . . 36

4 Distributed Resource Allocation in the Grid **40**
- 4.1 Grid Scheduling Issues . . . 42
- 4.2 Multi-Agent Reinforcement Learning . . . 42
- 4.3 The Model . . . 43
  - 4.3.1 Resources Providers . . . 43
  - 4.3.2 Users . . . 44
  - 4.3.3 Resource Selection . . . 44
- 4.4 Experimental Results . . . 45
  - 4.4.1 Effect of dynamic agent population . . . 46
  - 4.4.2 Significance of Results . . . 48

5 Collective Mind Initiative **48**
- 5.1 Background . . . 50
- 5.2 Collective Reasoning (includes planning and scheduling) . . . 54
- 5.3 Collective Behavior . . . 55
- 5.4 Improvisation . . . 57

5.5 Collective Mind Experiment and Prototype 57
5.5.1 The Experiments 59
5.5.2 Conclusion of the Experiment 61
5.6 Technology Transfer to the Military 62
5.7 Potential Impact 62

6 Discussion of Results and Approach **63**

7 Future Directions **64**

List of Figures

1 Diagram of a robot controller for the simplified foraging scenario 3

2 Collaboration rate per robot vs gripping time parameter for different robot group sizes and 16 sticks. (a) Results of the minimal model for 8 (short dash), 16 (long dash) and 24 (solid line) robots. (b) Results for detailed model (solid lines), embodied simulations (dotted lines), the microscopic model (dashed lines). 11

3 Time it takes the swarm of robots to collect objects in the arena for two difference interference strengths. Symbols are results of embodied simulations, while lines give the model's predictions. . 12

4 Physical set-up of the stick-pulling experiment showing six Khepera robots. 14

5 Flowchart of the robots' controller reported from [16] with overlapped state blocks. 15

6 Macroscopic state diagram of the multi-robot system. The arrow marked 's' corresponds to the transition from the gripping to the searching state after a successful collaboration, while the arrow marked 'u' corresponds to the transition after an unsuccessful collaboration, *i.e.*, when the robots time out. 17

7 Steady state solution vs (dimensionless) gripping time parameter τ: for $\beta = 0.5$ (short dash), 1 (long dash), 1.5 (solid line). Inset shows a typical relaxation to the steady state for $\tau = 5$, $\beta = 0.5$. 20

8 Collaboration rate per robot vs (dimensionless) gripping time parameter τ for $\beta = 0.5$ (short dash), $\beta = 1$ (long dash), $\beta = 1.5$ (solid line). These values of β correspond, respectively, to two, four, and six robots in the experiments with four sticks. 20

9 Diagram of the foraging arena (courtesy of D. Goldberg). 22

10 State diagram of a multi-robot foraging system with homing. . . 24

11 Time evolution of the fraction of searching robots (solid line) and undelivered pucks (dashed line) for $\tau = 3\ s$, $\alpha_p = 0.02$, $\alpha_r = 0.04$, and $\alpha_r' = 0.08$. 25

12 (a) Efficiency of different size robot groups defined as the inverse of the time it takes the group to collect 80% of the pucks in the arena for $\tau = 3\ s$ (solid line) and $\tau = 1\ s$ (dashed line) and $\tau_h^0 = 16\ s$, $\alpha_p = 0.02$, $\alpha_r = 0.04$, $\alpha_r' = 0.08$. (b) Efficiency per robot for different group sizes 26

13 (a) Average time each robot spent in the active behaviors during the time it took the group to deliver all pucks vs robot group size. (b) Percentage of time each robot was homing as a function of group size. 30

14 Snapshots from the simulation environment used. (left) An overhead view of foraging arena and robots. (right) A closeup of robots and pucks. 31

15 Evolution of the fraction of *Red* robots for different history lengths and transition functions, compared to predictions of the model . 35

16 Time evolution of robot densities without communication (a) and with communication (b) 39
17 Time evolution of robot densities at $x = 1$ for three different strategies 40
18 Aggregation time as a function of drift velocity, for three different values of n_0. 41
19 Average load vs time for job arrival rates a)P=0.15 and b)P=0.2 46
20 Average wait time for Least Loaded and RL selection rules ($P = 0.15$) 47
21 Average load vs time for different leaving probabilities P_L ($P = 0.15$) 47
22 Design cycle of Agent-based computing algorithms 64

List of Tables

1 Simulation parameters 28
2 Average time (in seconds) each robot spends in the active behaviors during the foraging task (0: search, 1: collect, 2: home, 3: reverse home, 4: avoid, 5: avoid while homing) as a function of robot group size. The last two columns give, respectively, the average number of avoidance maneuvers per robot while searching/collecting/reverse homing and while homing. 29

1 INTRODUCTION

Future intelligence services and armed forces will be increasingly reliant on distributed swarms of smart devices. While some of these intelligent swarms are already operational — teams of UAVs, groups of robots, networks of sensors — deployment of others, such as self-assembled and self-reconfigurable structures, smart materials, medical nano-robots, is ten, twenty or more years in the future.

Although individual units in the swarms mentioned above have various levels of individual complexity and size, swarms share common characteristics and control issues: namely, ability to function *autonomously* and *robustly* in uncertain dynamic environments with a high probability of component failure. Individual elements in a swarm also have *limited* and *faulty* sensing and communication capabilities, and often need to *coordinate* to achieve a global goal despite the *highly heterogeneous* and *highly distributed* nature of the system. Finally, individual components should be endowed with rudimentary *intelligence* and *learning* abilities in order to enhance the *adaptability* and *capabilities* of the swarm. The UAV example provides an illustration for why these capabilities are necessary. Although some UAVs are individually controlled, as bigger swarms are deployed for wider, more detailed surveillance, remote control is no longer feasible, and UAVs have to operate autonomously. Distributed decision making is thus critical to ensure robust coordination among UAVs, and to ensure mission continuation even as individual UAVs return for refueling or delegate their tasks for other reasons.

The issues listed above pose fundamental challenges to the design of robust, scalable swarm control algorithms. The challenge is made even greater by the fact that one does not control the collective behavior of the swarm directly — rather, it emerges out of interactions among individual components and between components and the environment.

Over the course of this project we have developed a mathematical framework for studying collective behavior of multi-agent swarms (MAS). This framework will allow the MAS designer to rationally specify and optimize individual control algorithms that will lead to the best collective behavior. Our mathematical approach has enabled us to model and quantitatively analyze collective behavior of different classes of agents:

- Simple agents using *reactive* control: agents decide about future actions based solely on input from sensors (including communication with other agents) and the action they are currently executing. Such agents can be represented as stochastic Markov processes [26, 29].

- Next, we extended the formalism to *adaptive* agents that change their behavior based on observations of the environment. These agents can be represented by a generalized Markov process of order m, where m is the number of observations used [24, 25].

- We also created a framework to study spatially inhomogeneous systems where agents are interacting with spatially extended fields, for example,

diffusing pheromones [11].

We showed that an equation, known as the Rate Equation, describes the dynamics of the collective behavior of swarms. The Rate Equation formalism can be derived from theory of stochastic processes [26], although in practice, the models are usually phenomenological. The Rate Equation approach has been applied to study distributed systems of reactive robots [43, 27, 23, 30, 1]. We formalized the approach and extended it to adaptive agents [25, 28] as well agents interactive through external fields [11]. Below we review the elements of the mathematical formalism for reactive (Section 1.2), adaptive(Section 1.3) and spatially interacting (Section 1.4) agents and illustrate with a few results from the robotics domain (Section 3.2, Section 3.1, and Section 3.3).

1.1 Background

Mathematical models can generally be broken into two classes: *microscopic* and *macroscopic*. Microscopic descriptions treat the agent as the fundamental unit of the model. These models describe the agent's interactions with other agents and the environment. Solving or simulating a system composed of many such agents gives researchers an understanding of the global behavior of the system. Examples of such microscopic models are reported in [31, 16]. Rather than compute the exact trajectories and sensory information of individual agents, their behavior is modeled as a series of stochastic events, with probabilities determined by simple geometric considerations and systematic experiments with small groups of agents. Running several series of stochastic events in parallel, one for each agent, allows researchers to study collective MAS behavior.

A macroscopic model, on the other hand, directly describes collective MAS behavior. It is computationally efficient because it uses fewer variables. These models have been successfully applied to a wide variety of problems in physics, chemistry, biology and the social sciences. In these applications, the microscopic behavior of an individual (e.g., a Brownian particle in a volume of gas or an individual residing in US) is quite complex, often stochastic and only partially predictable, and certainly analytically intractable. Rather than account for the inherent variability of individuals, scientists model the behavior of some *average* quantity that represents the system they are studying (e.g., volume of gas or population of US). Such macroscopic descriptions often have a very simple form and are analytically tractable. It is important to remember that such models do not reproduce the results of a single experiment — rather, the behavior of some observable averaged over many experiments or observations. The two description levels are, of course, related: we can start from the Stochastic Master Equation that describes the evolution of a agent's probability density and get the Rate Equation, a macroscopic model, by averaging it [26]. In most cases, however, Rate Equations are phenomenological in nature, *i.e.*, not derived from first principles. However, we have developed a "recipe" that allows one to formulate the Rate Equations describing dynamics of a homogeneous MAS composed of reactive agents simply by examining the details of individual agent controller.

1.2 Stochastic Approach to Modeling Multi-agent Systems

The behavior of individual agents in, for example a robotic, swarm has many complex influences, even in a controlled laboratory setting. Robots are influenced by external forces, many of which may not be anticipated, such as friction, battery power, sound or light signals, *etc.* Even if all the forces are known in advance, the robots are still subject to random events: fluctuations in the environment, as well as noise in the robot's sensors and actuators. A robot will interact with other robots whose exact trajectories are equally complex, making it impossible to know which robots will come in contact with one another. Finally, the designer can take advantage of the unpredictability and incorporate it directly into the robot's behavior: e.g., , the simplest effective policy for obstacle avoidance is for the robot to turn a random angle and move forward. In summary, the behavior of robots in a swarm is so complex, it is best described probabilistically, as a stochastic process.

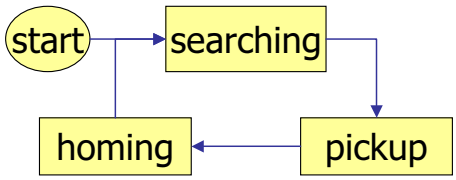


Figure 1: Diagram of a robot controller for the simplified foraging scenario

Consider Figure 1 — a controller for a simplified foraging scenario. Each box represents a robot's state — the action it is executing. In the course of accomplishing the task, the robot will transition from searching to puck pick-up to homing. Transitions between states are triggered by external stimuli, such as encountering a puck. This robot can be described as a stochastic Markov process[1], and the diagram in Figure 1 is, therefore, the Finite State Automaton (FSA) of the controller.

The stochastic processes approach allows us to mathematically study the behavior of robot swarms and other multi-agent systems. Let $p(n,t)$ be the probability a reactive robot is in state n at time t. The Markov property allows us to write change in probability density as [26]

$$\begin{aligned} \Delta p(n,t) &= p(n,t+\Delta t) - p(n,t) \\ &= \sum_{n'} p(n,t+\Delta t|n',t)p(n',t) - \sum_{n'} p(n',t+\Delta t|n,t)p(n,t). \quad (1) \end{aligned}$$

The conditional probabilities define the transition rates for a Markov process

$$W(n|n';t) = \lim_{\Delta t \to 0} \frac{p(n,t+\Delta t|n',t)}{\Delta t}. \quad (2)$$

[1] A Markov process's future state depends only on its present state and none of the past states.

The quantity $p(n,t)$ also describes a macroscopic variable — the fraction of robots in state n, with Eq. (1) describing how this variable changes in time. Averaging both sides of the equation over the number of robots (and assuming only individual transitions between states are allowed), we obtain in the continuous limit ($\lim_{\Delta t \to 0}$)

$$\frac{dN_n(t)}{dt} = \sum_{n'} W(n|n',t)N'_n(t) - \sum_{n'} W(n'|n,t)N_n(t), \quad (3)$$

where $N_n(t)$ is the *average* number of robots in state n at time t. This is the so-called Rate Equation. It is sometimes also written in a discrete form, as a finite difference equation that describes the behavior of $N(kT)$, k being an integer and T the discretization interval: $(N(t+T) - N(t))/T$. Eq. (3) has the following interpretation: the number of robots in state n will increase in time due to transitions *to* state n from other states, and it will decrease in time due to the transitions $from$ state n to other states.

The Rate Equation is a useful tool for mathematical analysis of collective dynamics of robot swarms. To facilitate the analysis, we begin by drawing the macroscopic state diagram of the system. *The collective behavior of the swarm is captured by an FSA that is functionally identical to the individual robot FSA, except that each state of the automaton now represents the number of robots executing that action* [27, 23, 30]. Not every microscopic robot behavior need to become a macroscopic state. In order to keep the model tractable, it is often useful to coarse-grain it by considering several related actions or behaviors as a single state. For example, we may take the searching state of robots to consist of the actions *wander in the arena, detect objects* and *avoid obstacles*. When necessary, the searching state can be split into three states, one for each behavior; however, we are often interested in the *minimal* model that captures the important behavior of the system. Coarse-graining presents a way to construct such a minimal model.

The macroscopic automaton can be directly translated into the Rate Equations. Each state in the automaton becomes a dynamic variable $N_n(t)$, with its own Rate Equation. Every transition will be accounted for by a term in the equation: a positive term for the incident ($W(n|n')N_{n'}$) arrows and negative term for the outgoing ($W(n'|n)N_n$) arrows.

Finding an appropriate mathematical form for the transition rates is the main challenge in studying real systems. The transition is triggered by some stimulus — be it another robot in a particular state, an object to be picked up, *etc.* In order to compute the transition rates, we assume, for simplicity, that robots and stimuli are uniformly distributed. The transition rates then have the following form: $W(n|n') \approx M$, where M is the environmental stimulus encountered (e.g., , number of sticks in the arena). The proportionality factor connects the model to experiments, and it depends on the rate at which a robot detects sticks. It can be roughly estimated from first principles ("scattering cross section" approach [27]), measured from simulations or experiments with one or two robots, or left as a model parameter. There will be cases where the

uniformity assumption fails: e.g., , in overcrowded scenarios where robots, depending on their obstacle avoidance controller, tend to clump, forming "robotic clouds" [30]. If the transition rates cannot be calculated from first principles, it may be expedient to leave them as parameters of the model and obtain them by fitting the model to data. We illustrate the approach in the sections below by applying it to study foraging (Section 3.2) and collaboration (Section 3.1) in multi-robot systems.

1.3 Modeling Adaptive Agents

If each agent had instantaneous global knowledge of the environment and the state of other agents, the system could dynamically adapt to any changes. In most situations, such global knowledge is impractical. However, for sufficiently slow dynamics, agents can correctly estimate the state of the environment through repeated local observations (by storing them in memory). We developed a theory of such a memory-based adaptation mechanism [24, 25], which is outlined here. Let $p(n,t)$ be the probability an agent is in state n at time t. We note that for a homogenous system of independent and indistinguishable agents, $p(n,t)$ describes the macroscopic state of the system, since it is simply the fraction of agents in the state n. Let us assume that the agents use a finite memory of length m of the past of the system in order to estimate the present state of the environment and make decisions about future actions. Then the evolution of the system can be represented as a generalized Markov processes of order m. This means that the state of an agent at time $t+\Delta t$ depends not only on the configuration of the system at time t (as in simple Markov systems), but also on configurations at times $t-\Delta t$, $t-2\Delta t$, ..., $t-(m-1)\Delta t$, which we refer to as history h of the system. In the derivation below we will employ the following identities: $p(n,t+\Delta t|h) = \sum_{n'} p(n,t+\Delta t|n',t;h)p(n',t|h)$ and $\sum_n p(n,t+\Delta t|n',t;h) = 1$.

Let us introduce the probability distribution function over the histories (for a homogenous system this distribution is the same for all the agents): $p(h,t)$, $1 = \sum_{h\in H} p(h,t)$, where H is the set of all feasible histories (if it continuous, one should use integration instead of summation for proper normalization). We can then write for the change in probability density Δp is:

$$\begin{aligned}
\Delta p(n,t) &= p(n,t+\Delta t) - p(n,t) \qquad (4)\\
&= \sum_h \big[p(n,t+\Delta t|h) - p(n,t|h)\big]p(h)\\
&= \sum_h \sum_{n'} p(n,t+\Delta t|n',t;h)p(n',t|h)p(h,t)\\
&- \sum_h \sum_{n'} p(n',t+\Delta t|n,t;h)p(n,t|h)p(h,t)
\end{aligned}$$

In the continuum limit, as $\Delta t \to 0$, $\Delta p/\Delta t$ can be written as

$$\frac{dp(n,t)}{dt} = \sum_h \sum_{n'} W(n|n';h)p(n',t|h)p(h,t) \qquad (5)$$

$$- \sum_{h} \sum_{n'} W(n'|n;h) p(n,t|h) p(h,t) \,,$$

with transition rates

$$W(n|n';h) = \lim_{\Delta t \to 0} \frac{p(n, t+\Delta t|n', t; h)}{\Delta t}. \tag{6}$$

In the most general form Eq.5 is analytically intractable due to strong correlations both in time and state–space. Instead, we average over all agents as in the preceding section and derive the macroscopic equation for the rate of change of $\langle N_n \rangle$, the average number of agents in state n:

$$\frac{dN_n}{dt} = \sum_{n'} \left[\langle W(n|n') \rangle_h N_{n'} - \langle W(n'|n) \rangle_h N_n \right]. \tag{7}$$

Here for notational convenience $\langle \ldots \rangle_h$ denotes average over histories, and we have dropped angle brackets around N, although this variable still denotes an average quantity.

Equation 7 is very similar to the rate equation we used to study Markov-based agent systems Eq. (3), except that transition rates $W(n|n')$ are now replaced by their history-averaged values. We will use the above equation to study how agents can use histories, or memories of past events, to improve the collective behavior of the system. We illustrate the approach in Section 3.3 by examining adaptive task allocation in robots.

1.4 Modeling Spatially Correlated Systems

While the approach outlined above works well for many spatially uniform systems, it is too coarse-grained for systems with a spatial correlation in agents' interactions. Thus, it is not sufficient to describe, for example, an ant-like swarm where agents interact through evolving chemical fields or robots monitoring chemicals released into a fluid. These situations require a generalization of the Master Equation, in which each robot not only has a discrete controller state k but also a continuous coordinate $\mathbf{x}$ (i.e., its spatial location). As with the original formulation, we suppose the number of agents in each state is sufficient to determine the collective behavior of interest. Because $\mathbf{x}$ is a continuous variable, these counts become densities leading us to introduce $\bar{n}_k(\mathbf{x}, t)$ as the average robot fraction density in state k at location $\mathbf{x}$ and time t. Thus a small volume Δx around location $\mathbf{x}$ contains, on average, the fraction $\bar{n}_k(\mathbf{x}, t)\Delta x$ of the robots in the system.

Let us consider a system where agents interact with the environment through a certain external chemical field. Let us also assume that agents are able to interact through stigmergy by releasing a special chemical into the environment that we call *communicative signal*. We denote $\rho(\mathbf{x}, t)$ and $c(\mathbf{x}, t)$ concentration of the chemical and communicative signal, respectively, at point $\mathbf{x}$ at time t.

Then we can write down the generalized rate equation as follows:

$$\begin{aligned} \frac{\partial \bar{n}_k(\mathbf{x},t)}{\partial t} &= \int d\mathbf{x}' \sum_j w_{jk}(\mathbf{x},\mathbf{x}';\rho,c)\bar{n}_j(\mathbf{x}',t) \\ &- \bar{n}_k(\mathbf{x},t)\int d\mathbf{x}' \sum_j w_{kj}(\mathbf{x},\mathbf{x}';\rho,c)\,. \end{aligned} \tag{8}$$

Now the transition rates w_{jk} depend not only the state indices j and k and occupation vector but also on the spatial coordinates and concentration of the chemical at the corresponding points. Note also that we have included the dependence of the transition rates on $\mathbf{x}$ and $\mathbf{x}'$ explicitly to account for agents' kinematics even in the absence of chemical and communicative concentrations (e.g., to describe freely diffusing agent).

The transition rates $w_{jk}(\mathbf{x},\mathbf{x}';\rho(\mathbf{x}),c(\mathbf{x}))$ summarize the behaviors of the individual robots. For example, the robot's internal state could change when it detects a chemical concentration above a predetermined threshold. Communication among nearby robots allows the robots to reduce noise in estimating chemical gradients and hence perform better than individual robots, but at a cost of additional power use for the communication. Robot motion, either moving passively with the fluid or using powered locomotion, e.g., to follow chemical concentration gradients, also contributes to the transitions.

While Equation 8 is a general description of the overall system behavior, it is too complex in its present form to be useful. Fortunately, it can be simplified considerably into a more intuitive form by noting that in many physically realistic situations agents' motion can be decoupled from state transitions, so that the transition rate can be represented as

$$\begin{aligned} w_{jk} &= \delta_{jk} W_k(\mathbf{x},\mathbf{x}';\rho(\mathbf{x}),\rho(\mathbf{x}'),c(\mathbf{x}),c(\mathbf{x}')) \\ &+ \delta(x-x')w_{jk}(\rho(\mathbf{x}),c(\mathbf{x}))\,, \end{aligned} \tag{9}$$

where δ_{jk} is Kroenecker's symbol[2] and $\delta(x)$ is its continuous analogue δ–function. In other words, during a transition between two discrete states we neglect the change in robot's position. In Eq. 9 W_k is an appropriately chosen kernel that describes agents' motion (as index k indicate, it can be different for each state), while the second term describes transition between discrete states.

Equation 9 allows us to separate transition function into terms with purely spatial transitions and terms with purely state transitions. Indeed, using Eq. 9 we can decouple the agents' kinematics from the state transitions between discrete state and rewrite Eq. 8 as follows:

$$\begin{aligned} \frac{\partial \bar{n}_k(\mathbf{x},t)}{\partial t} &= \mathcal{L}_k \bar{n}_k(\mathbf{x},t) + \sum_j w_{jk}(\rho,c)\bar{n}_j(\mathbf{x},t) \\ &- \bar{n}_k(\mathbf{x},t)\sum_j w_{kj}(\rho,c) \end{aligned} \tag{10}$$

[2]Kroenecker's symbol is defined as follows: $\delta_{ij}=1$ if $i=j$ and $\delta_{ij}=0$, $i\neq j$.

Here $\mathcal{L}_k$ is an operator (specified below) that describes the motion of agents in state k. The second and third terms in Eq. 10 describe agents state transitions. Note that now $w_{jk}(\rho, c)$ depends on spatial coordinates indirectly, through concentration $\rho(\mathbf{x}, t)$ and $c(\mathbf{x}, t)$. When the concentrations ρ and c are constants, Eq. 3 is recovered by integrating Eq. 10 over the spatial coordinates $\mathbf{x}$, assuming that $\mathcal{L}_k$ preserves the number of agents in state k (e.g., no absorbing boundaries) so that integral over the first term in Eq. 10 vanishes.

To specify the operators $\mathcal{L}_k$, we note that for the particular environment we are interested in, (i.e., microscopic robots operating in a fluid) robots' motion can be described by a diffusion equation [19]. We studied [11] chemotactic robots that respond to a chemical and signalling fields by propelling themselves in the direction of increasing concentration. This capability is modeled after bacterial chemotaxis which allows these single cell organisms to efficiently move towards food sources and away from noxious sources. Although in some cases the exact derivation from the microscopic transition rates is feasible, if very involved (see, for example, [36, 7] for treatment of bacterial chemotaxis which can be treated as a biased random walk), chemotaxis in a chemical concentration field $\rho(\mathbf{x}, t)$ is usually introduced into the rate equations phenomenologically by postulating a chemotactic velocity as $V_D = \eta_\rho \nabla \rho(\mathbf{x}, t)$, where η_ρ is the so called chemotactic sensitivity (which may itself depend on ρ). One can then write for operators $\mathcal{L}_k$

$$\mathcal{L}_k = D_k \nabla^2 - \mathbf{v} \cdot \nabla - \nabla \cdot [\mathbf{V}_D^\rho(\rho, \nabla\rho) + \mathbf{V}_D^c(c, \nabla c)] \tag{11}$$

Here, D_k is the diffusion coefficient of agents in state k assumed to be a constant, $\mathbf{v}$ is the flow velocity, and $\mathbf{V}_D^\rho$ and $\mathbf{V}_D^c$ are the chemotaxis drift velocities of robots due to concentration gradients of the chemical and the communicative signal, respectively.

To proceed further, we should also define how the chemical and concentration fields evolves in time. As an example relevant for microscopic robots, we consider the evolution of this fields in a moving fluid in which the robots operate. The evolutions of $\rho(\mathbf{x}, t)$ $c(\mathbf{x}, t)$ are governed by the diffusion equation:

$$\frac{\partial \rho}{\partial t} = D_\rho \nabla^2 \rho - \mathbf{v} \cdot \nabla \rho - \gamma_\rho \rho + Q_\rho(\mathbf{x}, t) \tag{12}$$

$$\frac{\partial c}{\partial t} = D_c \nabla^2 c - \mathbf{v} \cdot \nabla c - \gamma_c c + \sum_k q_k \bar{n}_k(\mathbf{x}, t) \tag{13}$$

In Eq. 12 the terms on the right describe, respectively, the diffusion of the chemical (with a diffusion constant D_ρ), the advection of the chemical due to fluid motion with velocity $\mathbf{v}$, the decay of the chemical at rate γ_ρ, and its deposition by sources with intensity profile $Q_\rho(\mathbf{x}, t)$. Terms in Eq. 13 have similar meaning, except the deposition rates of signalling chemical is proportional to the fraction of agents in state k, $\bar{n}_k(\mathbf{x}, t)$ (note that, generally speaking, the coefficients q_k themselves depend on the fraction of agents in state k). The parameters in this equation could, in general, depend on space and time, as well as the location of the robots (e.g., a sufficiently high concentration of the robots could significantly affect the fluid flow). For simplicity, we will treat

them as constants. For microscopic robots, fluid motions will usually be at very low Reynolds number so the fluid flow will be laminar with the velocity $\mathbf{v}$ changing smoothly with location. Viscous forces dominate the motions of such robots with requirements for locomotion mechanisms and power use quite different from experiences with larger robots [39].

In Section 3.4 we motivate the approach by describing a medically relevant scenario that considers a swarm of microscopic robots moving in a fluid to localize a chemical source. We solve a one-dimensional model and analyze different design choices.

1.5 Learning Approaches to Distributed Coordination

The problem of coordination in multi-agent systems, where agents have to achieve a consensus in their actions to receive maximum reward, is an important problem that has attracted much interest. Reinforcement learning and game dynamics have been shown to be a general and robust method for achieving coordination in MAS, even when agents are not directly communicating or sharing information [42]. In the game theory formalism, each agent is characterized by a set of strategies and it seeks to maximize its payoff (i.e., , utility or profit). Game dynamics studies the behavior of agents in response to games that are played many times successively. Over the course of the games, the winning strategies are rewarded, loosing ones are penalized, and the agents maximize their profit or utility by choosing the best performing strategies. It is the extra degree of freedom, characterized by the agents' strategies, that allows the system to adapt in dynamic environments. Game dynamics has a number of appealing properties as a control mechanism for multi-agent systems: it is distributed, flexible and scalable. The agents may vary in complexity from the very simple agents who do not have any information about other players or rules of the game, or even be aware of their existence, to more complex deliberative agents who can strategize and reason about their opponents beliefs and actions. The agents may act independently of one another, or jointly as in some cooperative agent systems, or they may cooperate or act competitively.

The Minority Game (MG) was introduced as a simplification of Arthur's El Farol Bar attendance problem [2]. The MG consists of N agents with bounded rationality that repeatedly choose between two alternatives labelled 0 and 1 (*e.g.*, staying at home or going to the bar). At each time step, agents who made the minority decision win. In the Generalized Minority Game, the wining group is 1 (0) if the fraction of the agents who chose "1" is smaller (greater) than the capacity level η, $0 < \eta < 1$ (for $\eta = 0.5$, the game reduces to the the traditional MG). Each agent uses a set of S strategies to decide its next move and reinforces strategies that would have predicted the winning group. A strategy is simply a lookup table that prescribes a binary output for all possible inputs. In the original version of the game, the input is a binary string containing the last m outcomes of the game, so the agents interact by sharing the same global signal. If the agents choose either action with probability 1/2 (the random choice game), then, on average, the number of agents choosing

"1" (henceforth referred to as attendance) is $(N-1)/2$ with standard deviation $\sigma = \sqrt{N}/2$ in the limit of large N. The most interesting phenomenon of the minority model is the emergence of a coordinated phase, where the standard deviation of attendance, the volatility, becomes smaller than in the random choice game. The coordination is achieved for memory sizes for which the dimension of the reduced strategy space is comparable to the number of agents in the system[4, 40], $2^m \sim N$.

In addition to the original MG, different versions of the game where the agents interact using local information only have been studied. In particular, it was established that coordination still arises out of local interactions, and the system as a whole achieves "better than random" performance in terms of the utilization of resources. Note that reinforcement learning is similar to game dynamics, in that an agent receives a payoff that allows it to determined best actions. Minority Games and reinforcement learning in general can serve as a general paradigm for resource allocation and load balancing in multi–agent systems.

In all previous studies the capacity level has been fixed as an external parameter, so the environment in which the agents compete is stationary. In many situations, however, agents have to operate in dynamic environments. We addressed this problem in our research. Namely, we studied a system of boolean agents playing a generalized minority game, and assumed that the capacity level is not fixed but varies with time, $\eta(t) = \eta_0 + \eta_1(t)$, where $\eta_1(t)$ is a time dependent perturbation. The framework of the interactions was based on Kauffman NK random boolean nets [20], where each agent gets its input from K other randomly chosen agents, and maps the input to a new state according to a boolean function of K variables, which is also randomly chosen and quenched throughout the dynamics of the system. The generalization we made is that agents are allowed to adapt by having more than one boolean function, or strategy, and the use of a particular strategy is determined by an agent based on how often it predicted the winning group throughout the game.

2 SUMMARY OF PROJECT RESULTS

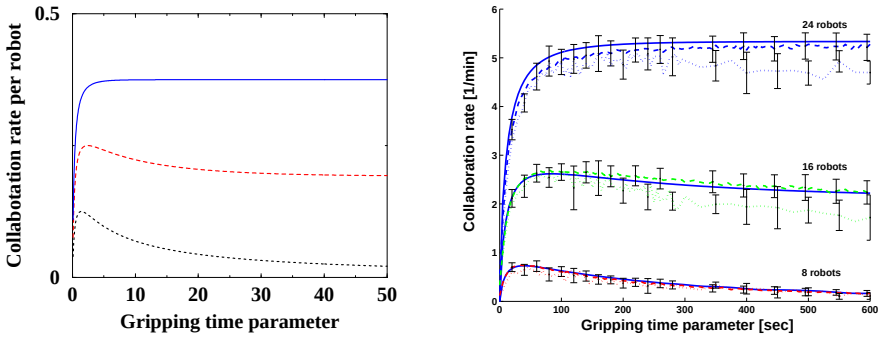
We have achieved great success in applying mathematical formalism outlined above to study collective behavior of distributed systems of mobile robots for which a body of experimental and simulations data exists. In this section we outline some of our successes, going into detail of the particular applications in the later sections.

2.1 Collective Behavior of Groups of Robots

We mathematically studied collective behavior of various distributed robot systems. These studies were inspired and corroborated by experiments and simulations with real robots. For example, Ijspeert et al. [16] studied dynamics of collaboration in groups of robots using stick-pulling experiments as a model of

collaboration. The robots' task was to locate sticks scattered around the arena and pull them out of their holes. A single robot cannot complete the task on its own: rather, when a robot finds a stick, it lifts it partially out of the hole and waits for a period specified by its *gripping time parameter* for a second robot to find it. If a second robot finds the first during this time interval, it will pull the stick out; otherwise, the first robot releases the stick and returns to the searching state.

We found that a minimal model that includes only the salient details of the process [27] reproduced key experimental observations and qualitatively agreed with results of experiments and simulations (see Figure 2(a)). Martinoli & Easton [30] formulated a more detailed model based on our work that accounts for every state in the robot control diagram.



(a) (b)

Figure 2: Collaboration rate per robot vs gripping time parameter for different robot group sizes and 16 sticks. (a) Results of the minimal model for 8 (short dash), 16 (long dash) and 24 (solid line) robots. (b) Results for detailed model (solid lines), embodied simulations (dotted lines), the microscopic model (dashed lines).

Figure 2 depicts the collaboration rate, the rate at which robots pull sticks out, as a function of the individual robot gripping time parameter for the minimal (a) and the detailed (b) models. Figure 2(b) also shows results of embodied and probabilistic numeric simulations for the same set of parameters. One can see quantitative agreement already with swarms as small as 8 robots. The minimal model shows the same qualitative behavior as the more detailed model. See Section 3.1 for details of the application.

In foraging experiments, we studied the influence of physical interference on the swarm performance [23]. Interference is a critical issue in swarm robotics, in particular in foraging experiments where there is a spatial bottleneck at the predefined "home" region where the collected objects must be delivered. When two robots find themselves within sensing distance of one another, they will execute obstacle avoidance maneuvers. Because this behavior takes time, interference decreases robots' efficiency. Clearly, a single robot working alone is

relatively more efficient, because it does not experience interference from other robots (the larger the swarm, the greater the degree of interference). However, parallel work helps speed up the foraging process and increases the system robustness in case of individual robot failures.

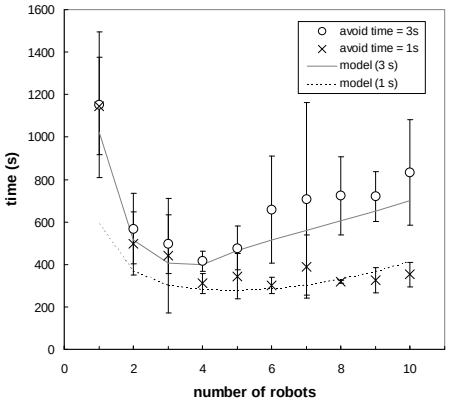


Figure 3: Time it takes the swarm of robots to collect objects in the arena for two difference interference strengths. Symbols are results of embodied simulations, while lines give the model's predictions.

Figure 3 shows the total time required to complete the task for two different interference strengths, as measured by the avoiding time τ. For both cases task completion time is minimized for some swarm size and increases for larger swarms. The greater the effect of interference (larger τ), the smaller the optimal swarm size. Results show good quantitative agreement with embodied simulations with swarms of one to 20 robots. Section 3.2 presents details of the application.

We studied extensions of the basic model outlined in Section 1.2. In Section 3.3 we analyze dynamic task allocation in multi-robot systems. In this application, robots adapt to changing task requirements and environmental conditions by making repeated local observations of the tasks, environment and other robots. Such robots can be described as general Markov processes and studied using the formalism of Section 1.3. We obtained very good agreement between predictions of the model and results of realistic 3-D simulations.

Another refinement of the formalism, as described in Section 1.4, applies to spatially non-uniform systems, for example, systems where robots generate and interact with diffusing chemical fields.

2.2 Distributed Resource Allocation

The problem of coordination in multi-agent systems, where agents have to achieve a consensus in their actions to receive maximum reward, is an important problem that has attracted much interest recently [41]. We studied minority

games and reinforcement learning as a model for resource allocation/load balancing problem in a large scale MAS, where resource capacities are changing in time. We found that reinforcement learning [9] and minority games[13, 12] were efficient and robust mechanisms for achieving coordination in dynamic distributed systems. We applied this mechanism for load balancing in Grid distributed computing environment [10]. Section 4 presents details of this application.

2.3 Collective Mind Project

In addition to this work, *Business Collective Mind for Equipment Reliability* project was funded by DARPA at the level of a study. The Principal Investigator was Norman Sondheimer of University of Massachusetts with William Wallace of Rensselaer Polytechnic Institute and Peter Will of University of Southern California Information Sciences Institute as co-PIs. The tasks were to solicit ideas from the best University, Industry and Military researchers and practitioners on the *Collective Mind* concept to generate support for a research program from the Military and report the results to DARPA. This project is described in Section 5.

3 Robotic Applications

In the sections below we illustrate our approach to modeling and analyzing collective behavior of multi-agent systems with detailed applications from the robotics domain.

3.1 Collaboration in a Group of Robots

The stick-pulling experiments were carried out by Ijspeert *et al.*[16] to study the dynamics of collaboration among locally interacting simple reactive robots. Figure 4 is a snapshot of the physical set-up of the experiments. The robots' task is to locate sticks scattered around the arena and pull them out of their holes. A single robot cannot pull the stick out by itself — a collaboration between two robots is required for the task to be successfully completed. Collaboration occurs in the following way: one robot finds a stick, lifts it partly out of the ground and waits for a second robot to find it and complete the task by pulling the stick out of its hole completely.

The actions of each robot are governed by the same simple controller, outlined in Figure 5. The robot's default behavior is to wander around the arena looking for sticks and avoiding obstacles, which could be other robots or walls. When a robot finds a stick that is not being held by another robot, it grips it, lifts it half way out of the ground and waits for a period of time specified by the *gripping time parameter*. If no other robot comes to its aid during the waiting period (time out), the robot releases the stick and resumes the search for other sticks. If another robot encounters a robot holding a stick, a successful

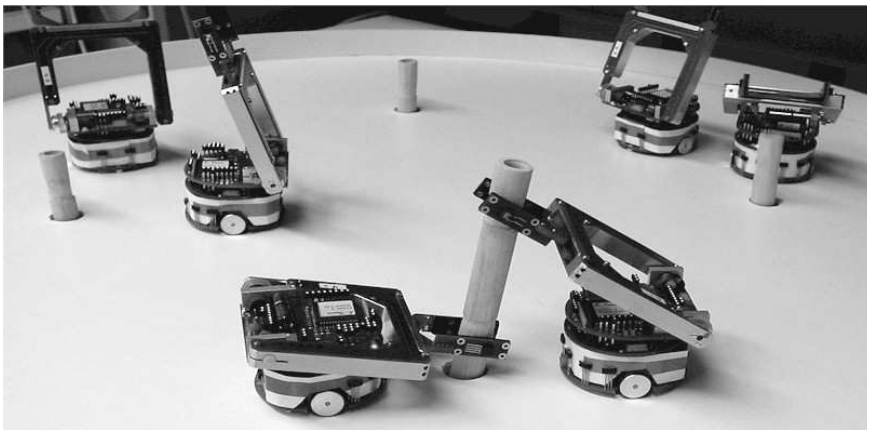


Figure 4: Physical set-up of the stick-pulling experiment showing six Khepera robots.

collaboration will take place during which the second robot will grip the stick, pulling it out of the ground completely, while the first robot releases the stick and resumes the search. After the task is completed, the second robot also releases the stick and returns to the search mode, and the experimenter replaces the stick in its hole.

3.1.1 Real Robots, Embodied Simulations and Microscopic Modeling

Ijspeert *et al.* studied the dynamics of collaboration in the stick-pulling experiment at three different levels: by conducting experiments with physical robots; using a sensor-based simulator of robots; and using a microscopic probabilistic model. The physical experiments were carried out in groups of two to six Khepera robots in an arena containing four sticks. Because experiments with physical robots are very time consuming, Webots, the sensor-based simulator of Khepera robots, was used to systematically explore parameters affecting the dynamics of collaboration. The Webots simulator [32] attempts to faithfully model the environment and replicate the experiment by reproducing the robots' (noisy) sensory input and the (noisy) response of the on-board actuators in order to compute the trajectory and interactions of all the robots in the arena. The probabilistic microscopic model, on the other hand, does not attempt to compute trajectories of individual robots. Rather, it is a numerical model in which the robot's actions — encountering a stick, a wall, another robot, a robot gripping a stick, or wandering around the arena — are represented as a series of stochastic events, with probabilities based on simple geometric considerations and systematic tests with one or two real robots. For example, the probability of a robot encountering a stick is equal to the product of the number of ungripped sticks, and the detection area of the stick normalized by the arena area.

Probabilities of other interactions can be similarly calculated. The microscopic simulation consists of running several processes in parallel, one for each robot, while keeping track of the global state of the environment, such as the number of gripped and ungripped sticks. According to Ijspeert *et al.* the acceleration factor for Webots and real robots can vary between one and two orders of magnitude for the experiments presented here. Because the probabilistic model does not require calculations of the details of the robots' trajectories, it is about 300 times faster than Webots for this experiment.

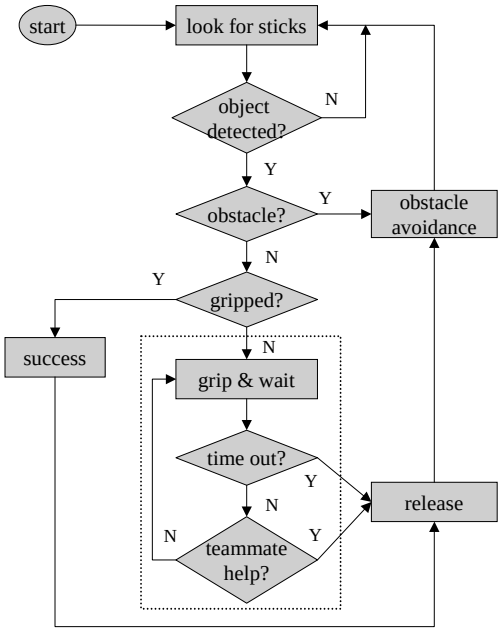


Figure 5: Flowchart of the robots' controller reported from [16] with overlapped state blocks.

Ijspeert *et al.* systematically studied the collaboration rate, *i.e.*, the number of sticks successfully pulled out of the ground in a given time interval, and its dependence on the group size and the gripping time parameter. Though in that work they also investigated the effects of robot heterogeneity and explicit communication, we will focus on a homogeneous system of non-communicating robots. Ijspeert *et al.* report very good qualitative and quantitative agreement between the three different levels of experiments. The main result is that, depending on the ratio of robots to sticks (or workers to the amount of work), there appear to be two different regimes in the collaboration dynamics. When there are fewer robots than sticks, the collaboration rate decreases to zero as the value of the gripping time parameter grows. In the extreme case, when the robot grabs a stick and waits indefinitely for another robot to come and help it, the collaboration rate is zero, because after some period of time each robot ends

up holding a stick, and no robots are available to help. When there are more robots than sticks, the collaboration rate remains finite even in the limit the gripping time parameter becomes infinite, because there will always be robots available to help pull the sticks out. Another finding of Ijspeert *et al.* was that when there are fewer robots than sticks, there is an optimal value of the gripping time parameter which maximizes the collaboration rate. In the other regime, the collaboration rate appears to be independent of the gripping time parameter above a specific value, so the optimal strategy is for the robot to grip a stick and hold it indefinitely. They also found that the system is one of few collaborative systems known to the authors that demonstrates super-linearity, *i.e.*, for some range of robot group sizes and a given number of sticks, adding a robot not only increases the global performance of the system but also the relative performance of the other robots. However, as the robot group size increases, the overcrowding and interference effects cause the relative collaboration rate to saturate and become sub-linear.

3.1.2 Mathematical Model of the Stick-Pulling Experiments

In the following sections we present a macroscopic analytical model of the stick-pulling experiments in a homogeneous multi-robot system. Such a model is useful for the following reasons. First, the complexity of a macroscopic model is independent of the system size, *i.e.*, the number of robots: therefore, the time required to obtain solutions for a system of $5,000$ robots is as long as that to obtain solutions for a system of five robots, whereas for a microscopic description the time required for computer simulation scales at least linearly with the number of robots. Second, our approach allows us to derive analytic expressions for certain important parameters, (*e.g.*, those for which the performance is optimal). It also enables us to study the stability properties of the system, and see whether solutions are robust under external perturbation or noise. These capabilities are important for the design and control of large multi-agent systems.

In order to construct a model of the stick-pulling experiments, it is helpful to write the macroscopic state diagram of the system. During a sufficiently short time interval, each robot can be thought to be in one of two states: *searching* or *gripping*. The state labels several related robot behaviors and it is a useful shorthand for thinking about the system. Using flowchart of the robots' controller, shown in Fig. 5, as a reference, we can consider the search state to be the set of behaviors associated with looking for sticks, such as wandering around the arena ("look for sticks" action), detecting objects and avoiding obstacles; while the gripping state is composed of the decisions and actions inside the dotted box. We assume that actions "success" (pull the stick out completely) and "release" (release the stick) take place on a short enough time scale that they can be incorporated into the search state. While the robot is in the obstacle avoidance mode, it cannot detect and try to grip objects; therefore, avoidance serves to decrease the number of robots that are searching and capable of gripping sticks. We can also include avoidance into the model explicitly [27].

In addition to states, we must also specify all possible transitions between states. When it finds a stick, the robot makes a transition from the search state to the gripping state. After both a successful collaboration and when it times out (unsuccessful collaboration) the robot releases the stick and makes a transition into the searching state, as shown in Fig. 6. These arrows correspond to the arrow entering and the two arrows leaving the dotted box in Fig. 5. We will use the macroscopic state diagram as the basis for writing down the differential rate equations that describe the dynamics of the stick-pulling experiments.

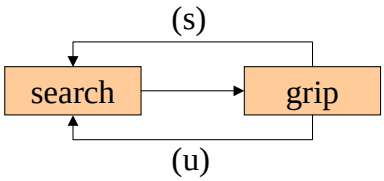


Figure 6: Macroscopic state diagram of the multi-robot system. The arrow marked 's' corresponds to the transition from the gripping to the searching state after a successful collaboration, while the arrow marked 'u' corresponds to the transition after an unsuccessful collaboration, *i.e.*, when the robots time out.

The dynamic variables of the model are $N_s(t)$ and $N_g(t)$, the number of robots in the searching and gripping states respectively. Also, let $M(t)$ be the number of unextracted sticks at time t. The latter variable does not represent a macroscopic state, rather it tracks the state of the environment. We assume that robots and sticks are distributed uniformly around the arena.

A series of differential rate equations govern the dynamics of the stick-pulling system:

$$\begin{aligned} \frac{dN_s}{dt} &= -\alpha N_s(t)\Big(M(t) - N_g(t)\Big) + \tilde{\alpha} N_s(t) N_g(t) \\ &\quad +\alpha N_s(t-\tau)\Big(M(t-\tau) - N_g(t-\tau)\Big)\Gamma(t;\tau) \qquad (14) \\ N_g &= N_0 - N_s \qquad (15) \\ \frac{dM}{dt} &= -\tilde{\alpha} N_s(t) N_g(t) + \mu(t) \qquad (16) \end{aligned}$$

where α, $\tilde{\alpha}$ are the rates at which a searching robot encounters a stick and a gripping robot respectively, τ is the gripping time parameter, and $\mu(t)$ is the rate at which new tasks are added. The parameters α, $\tilde{\alpha}$, and τ connect the model to the experiment. α and $\tilde{\alpha}$ are related to the size of the object, the robot's detection radius, or footprint, and the speed at which it explores the arena. The three terms in Eq. 14 correspond to the three arrows in Fig. 6. The first term accounts for the decrease in the number of searching robots because some robots find and grip sticks. Under the uniform distribution assumption, the rate at which robots encounter ungripped sticks is proportional to the number

of ungripped sticks in the arena, with the proportionality factor given by α. The second term describes the successful collaborations between two robots, and the third term accounts for the failed collaborations, both of which lead to an increase the number of searching robots.

$\Gamma(t;\tau)$, the fraction of failed collaborations at time t, is the probability no robot came "to help" during the time interval $[t-\tau,t]$. To calculate $\Gamma(t;\tau)$ let us divide the time interval $[t-\tau,t]$ into K small intervals of length $\delta t=\tau/K$. The probability that no robot comes to help during the time interval $[t-\tau,t-\tau+\delta t]$ is simply $1-\tilde{\alpha}N_s(t-\tau)\delta t$. Hence, the probability for a failed collaboration is

$$\begin{aligned}\Gamma(t;\tau) &= \prod_{i=1}^{K}[1-\tilde{\alpha}\delta t N_s(t-\tau+i\delta t)]\Theta(t-\tau) \qquad (17)\\ &\equiv \exp\left[\sum_{i=1}^{K}\ln[1-\tilde{\alpha}\delta t N_s(t-\tau+i\delta t)]\right]\Theta(t-\tau)\end{aligned}$$

The step function $\Theta(t-\tau)$ ensures that $\Gamma(t;\tau)$ is zero for $t<\tau$. Finally, expanding the logarithm in Eq.(18) and taking the limit $\delta t\to 0$ we obtain

$$\Gamma(t;\tau)=\exp[-\tilde{\alpha}\int_{t-\tau}^{t}dt' N_s(t')]\Theta(t-\tau) \qquad (18)$$

We do not need a differential equation for N_g, the number of gripping robots, because this quantity may be computed using conservation of robots condition, Eq. 15. The last equation, Eq. 16, says that the number of unextracted sticks $M(t)$ decreases in time at the rate of successful collaborations. The equations are subject to the initial conditions that at $t=0$ the number of searching robots is N_0 and the number of unextracted sticks is M_0.

Dimensional Analysis To proceed further let us introduce $n(t)=N_s(t)/N_0$, $m(t)=M(t)/M_0$, $\beta=N_0/M_0$, $R_G=\tilde{\alpha}/\alpha$, $\tilde{\beta}=R_G\beta$ and a dimensionless time $t\to\alpha M_0 t$, $\tau\to\alpha M_0\tau$. μ' is the dimensionless rate at which new tasks (sticks) are added. $n(t)$ is the fraction of robots in the search state and $m(t)$ is the fraction of unextracted sticks at time t. Due to the conservation of the number of robots, the fraction of robots in the gripping state is simply $1-n(t)$. The equations Eq. 14– 16 can be rewritten in dimensionless form as:

$$\begin{aligned}\frac{dn}{dt} &= -n(t)[m(t)+\beta n(t)-\beta]+\tilde{\beta}n(t)[1-n(t)]+n(t-\tau)[m(t-\tau)\\ &\quad +\beta n(t-\tau)-\beta]\times\gamma(t;\tau) \qquad (19)\\ \frac{dm}{dt} &= -\beta\tilde{\beta}n(t)[1-n(t)]+\mu' \qquad (20)\\ \gamma(t;\tau) &= \exp[-\tilde{\beta}\int_{t-\tau}^{t}dt' n(t')] \qquad (21)\end{aligned}$$

Equations 19–21 together with initial conditions $n(0)=1$, $m(0)=1$ determine the dynamical evolution of the system. Note that only two parameters, β

and τ, appear in the equations and, thus, determine the behavior of solutions. The third parameter $\tilde{\beta} = R_G\beta$ is fixed experimentally and is not independent. Note that we do not need to specify α and $\tilde{\alpha}$ — they enter the model only through R_G (throughout this paper we will use $R_G = 0.35$, the value reported in [16]).[3] Below we provide a detailed analysis of these equations.

Analysis of Results Let is assume that new sticks are added to the system at the same rate that the robots pull them out. This situation was realized experimentally by replacing the sticks in their holes after they were pulled out by robots. Therefore, the number of sticks does not change with time ($m(t) = m(0) = 1$). A steady-state solution, if it exists, describes the long term time-independent behavior of the system. To find it, we set the left hand side of Eq. 19 to zero. Eq. 19 has a non-trivial steady–state solution which satisfies the following transcendental equation:

$$-1 + (\beta + \tilde{\beta})(1 - n) + (1 - \beta(1 - n))e^{-\tilde{\beta}\tau n} = 0 \tag{22}$$

Figure 7 shows the dependence of the fraction of searching robots in the steady state on the gripping time τ for different values of the parameter β. Note, that for small enough β's $n(\tau) \to 0$ as $\tau \to \infty$. The intuitive reason for this is the following: when there are fewer robots than sticks, and each robot holds the stick indefinitely, after a while every robot is holding a stick, and no robots are searching. For $\beta > 1/(1 + R_G)$, however, $n(\tau) \to const \neq 0$ as $\tau \to \infty$. The inset in Fig. 7 shows how a typical solution, $n(t)$, relaxes to its steady state value. The oscillations are characteristic of time–delay differential equations, and their period is determined by τ.

The collaboration rate is the rate at which robots successfully pull sticks out of their holes. The steady–state collaboration rate $R(\tau; \beta)$ is given by the following equation:

$$R(\tau, \beta) = \beta\tilde{\beta}n(\tau, \beta)[1 - n(\tau, \beta)]\,, \tag{23}$$

where $n(\tau, \beta)$ is the number of searching robots in the steady–state for a particular value of τ and β, and $(1 - n(\tau, \beta))$ is the number of gripping robots in the steady–state. Figure 8 depicts the collaboration rate as a function of τ. For $\beta > \beta_c$ the collaboration rate increases monotonically with τ. However, for $\beta < \beta_c$ there is an optimal gripping time, $\tau = \tau_{opt}$, which maximizes the collaboration rate. To understand this behavior note that the maximum collaboration rate for a given β is achieved for $n(\tau, \beta) = 1/2$. For $\beta > \beta_c$, however,

[3]The parameter α can be easily calculated from experimental values quoted in [16]. As a robot travels through the arena, it sweeps out some area during time dt and will detect objects that fall in that area. This detection area is $V_R W_R dt$, where $V_R = 8.0\ cm/s$ is robot's speed, and $W_R = 14.0\ cm$ is robot's detection width. If the arena radius is $R = 40.0\ cm$, a robot will detect sticks at the rate $\alpha = V_R W_R/\pi R^2 = 0.02\ s^{-1}$. According to [16], a robot's probability to grab a stick already being held by another robot is 35% of the probability of grabbing a free stick. Therefore, $R_G = \tilde{\alpha}/\alpha = 0.35$. R_G is an experimental value obtained with systematic experiments with two real robots, one holding the stick and the other one approaching the stick from different angles.

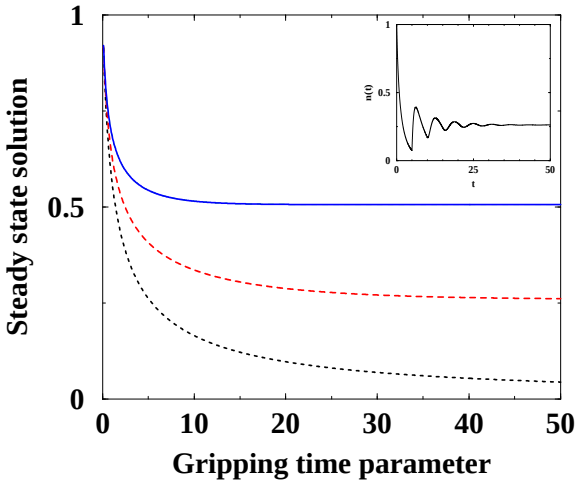


Figure 7: Steady state solution vs (dimensionless) gripping time parameter τ: for $\beta = 0.5$ (short dash), 1 (long dash), 1.5 (solid line). Inset shows a typical relaxation to the steady state for $\tau = 5$, $\beta = 0.5$.

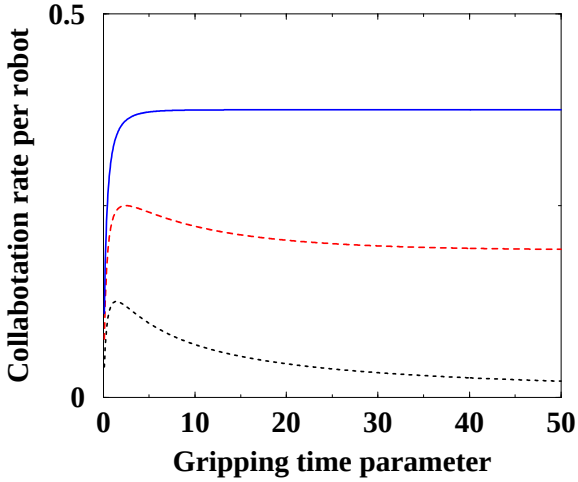


Figure 8: Collaboration rate per robot vs (dimensionless) gripping time parameter τ for $\beta = 0.5$ (short dash), $\beta = 1$ (long dash), $\beta = 1.5$ (solid line). These values of β correspond, respectively, to two, four, and six robots in the experiments with four sticks.

the solution of Eq. 22 is always greater than 1/2, so an optimal solution does not exist. For $\beta < \beta_c$ a simple analysis gives

$$\tau_{opt} = \frac{2}{\tilde{\beta}} \ln \frac{1 - \beta/2}{1 - 1/2(\beta + \tilde{\beta})}, \quad \beta < \beta_c = \frac{2}{1 + R_G} \tag{24}$$

Mathematical analysis of the minimal model reproduces the following conclusions of Ijspeert *et al.*: the different dynamical regimes depending on the value of the ratio of robots to sticks (β) and the optimal gripping time parameter for $\beta < \beta_c$. The three curves in Fig. 8 are qualitatively similar results of simulations in groups of up to six robots. Martinoli & Easton [30] formulated a more detailed model based on our work that accounts for every state in the robot control diagram and agrees quantitatively with simulations of groups of as few as a dozen robots.

3.2 Optimal Group Size for Robot Foraging

Figure 9 is a snapshot of a typical foraging experiment with four robots. The robots' task is to collect small pucks scattered randomly around the arena. The arena itself is divided into a search region and a small "home", or goal, region where the collected pucks are deposited. The "boundary" and "buffer" regions are part of the home region and are made necessary by limitations in the robots' sensing capabilities, as described below. Each robot has an identical set of behaviors governed by the same controller. The behaviors that arise in the collection task are [15]:

Avoiding obstacles, including other robots and boundaries. This behavior is critical to the safety of the robot.

Wandering or searching for pucks: robot moves forward and at random intervals turns left or right through a random arc. If the robot enters the Boundary region, it returns to the search region. This prevents the robot from collecting pucks that have already been delivered.

Detecting a puck.

Grabbing a puck.

Homing : if carrying a puck, move towards the home location.

Creeping : activated by entering Buffer region. The robot will start using the close-range detectors at this point to avoid the boundaries.

Home : robot drops the puck. This activates the exiting behavior.

Exiting : robot exits the home region and resumes search.

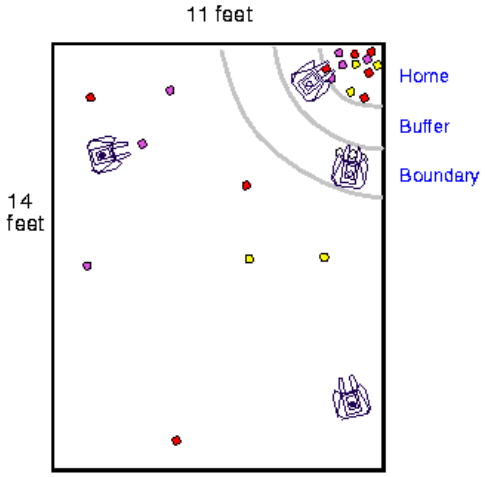


Figure 9: Diagram of the foraging arena (courtesy of D. Goldberg).

3.2.1 Interference

In the foraging scenario outlined above, robots act completely independently, without communicating directly or through the environment. Interference is the only interaction between the robots, and it is caused by competition for space between spatially extended robots. When two robots find themselves within sensing distance of one another, they will execute obstacle avoiding maneuvers in order to reduce the risk of a potentially damaging collision. The robot stops, turns in place by some angle and moves forward. This behavior takes time to execute; therefore, avoidance increases the time it takes the robot to find pucks and deliver them home. Clearly, a single robot working alone will not experience interference from other robots. However, if a single robot fails, as is likely in a dynamic, hostile environment, the collection task will not be completed. A group of robots, on the other hand, is robust to an individual's failure. Indeed, many robots may fail but the performance of the group may be only moderately affected. Many robots working in parallel may also speed up the collection task. Of course, the larger the group, the greater the degree of interference — in the extreme case of a crowded arena, robots will spend all their time avoiding other robots and will not bring any pucks home.

Interference has long been recognized as a critical issue in multi-robot systems [8, 43]. Several approaches to minimize interference have been explored, including communication [37] and cooperative strategies such as trail formation [46] and bucket brigade [8, 35]. In some cases, the effectiveness of the strategy to minimize interference will also depend on the group size [35]. Therefore, it is important to quantitatively understand interference between robots and how it relates to the group and task sizes before choosing alternatives to the default strategy. For some tasks and a given controller, there may exist an optimal group

size that maximizes the performance of the system [34, 8, 35]. Beyond this size the adverse effects of interference become more important than the benefits of increased robustness and parallelism, and it may become beneficial to choose an alternate foraging strategy. We will study interference mathematically and attempt to answer these questions.

3.2.2 Mathematical Analysis of Foraging

As mentioned above, interference is the result of competition between two or more robots for the same resource, be it physical space, a puck both are trying to pick up, energy, communications channel, *etc.* In the collection and foraging tasks, competition for physical space, and the resulting avoidance of collisions with other robots, is the most common source of interference. In order to understand interference quantitatively, we will first examine the simplified foraging task that includes searching and avoiding only. This task can be implemented with a subset of robot behaviors listed in Section 3.2, namely searching, avoiding, detecting a puck and grabbing it. This scenario may be realized experimentally by allowing robots to pick up a puck and store it in a carrying pouch, for instance. Then we will examine the full foraging scenario, where robots are required to deliver collected pucks to a home location.

Above we described a methodology for constructing mathematical models of collective behavior of multi-agent systems. The methodology applies to Markov systems, in which each agent's state at a future time depends only on its present state and none of its past states. While this may seem as a restrictive criterion, it is satisfied by many behavior-based and reactive robot systems. In the context of robotics, *state* labels a set of related robot behaviors required to accomplish a task. Thus, the search state may consist of the *wandering* and *puck detecting* behaviors, or we may simply take each behavior to be a separate state. The mathematical model consists of a series of coupled differential equations, one for each state, each of which describes how the average number of agents in that state changes in time. The equations may be solved analytically or numerically, allowing us to quantitatively study the behavior of the multi-agent system. Below we construct and solve a mathematical model of two foraging scenarios, with an emphasis on analyzing the effects of interference.

Figure 10 shows the state diagram for foraging with homing. Initially the robots are in the search state. When a searching robot encounters a puck, it picks it up and moves toward the "home" region. Execution of the homing behavior requires a period of time τ_h. At the end of this period, the robot deposits the puck at home and resumes the search for more pucks. While a robot is either searching or homing, it will encounter and try to avoid obstacles for a time period τ after which it returns to its previous state. There are two separate avoiding states to preclude robots from moving from the searching to the homing state, or *vice versa*, through the common avoiding state.

Each state in the diagram corresponds to a dynamic variable. Let $N_s(t)$, $N_h(t)$, $N_s^{av}(t)$, $N_h^{av}(t)$ be the number of searching, homing, avoiding while searching and avoiding while homing robots at time t, with the total num-

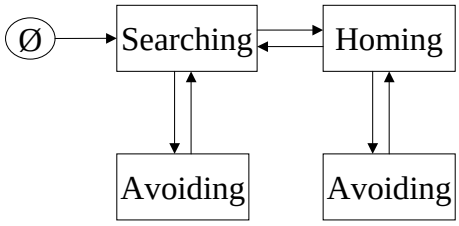


Figure 10: State diagram of a multi-robot foraging system with homing.

ber of robots, $N_0 = N_s(t) + N_h(t) + N_s^{av}(t) + N_h^{av}(t)$, a constant. We model the environment by letting $M(t)$ be the number of undelivered pucks at time t. Also, let α_r be the rate of detecting another robot and α_p the rate of detecting a puck. These parameters connect the model to the experiment, and they are related to the size of the robot and the puck, robot's detection radius and the speed of the robot. It was shown experimentally [15] that interference is most pronounced near the home region, because the density of robots is, on average, greater there. Therefore, we expect the rate of encountering other robots to be greater near the home region and introduce α_r', the rate of detecting another robot while homing. The following equations describe the time evolution of the dynamic variables[4]:

$$\begin{aligned}
\frac{dN_s(t)}{dt} &= -\alpha_p N_s(t)[M(t) - N_h(t) - N_h^{av}(t)] \\
&\quad -\alpha_r N_s(t)[N_s(t) + N_0] + \frac{1}{\tau_h} N_h(t) + \frac{1}{\tau} N_s^{av}(t), \qquad (25) \\
\frac{dN_h(t)}{dt} &= \alpha_p N_s(t)[M(t) - N_h(t) - N_h^{av}(t)] \\
&\quad -\alpha_r' N_h(t)[N_h(t) + N_0] + \frac{1}{\tau} N_h^{av}(t) - \frac{1}{\tau_h} N_h(t), \qquad (26) \\
\frac{dN_h^{av}(t)}{dt} &= \alpha_r' N_h(t)[N_h(t) + N_0] - \frac{1}{\tau} N_h^{av}(t), \qquad (27) \\
\frac{dM(t)}{dt} &= -\frac{1}{\tau_h} N_h(t). \qquad (28)
\end{aligned}$$

The first two terms in Eq. 25 account for a decrease in the number of searching robots when robots find pucks and start homing, or when searching robots encounter and attempt to avoid other robots. The number of available pucks is just the number of pucks in the arena less the pucks held by homing robots. When a searching robot encounters another searching robot, both start executing avoidance maneuvers, decreasing the number of searching robots by two; while when a searching robot encounters a homing or either of the avoiding

[4]For simplicity, we do not include wall avoidance in the equations, but do take it into account when fitting model to the data.

robots, the number of searching robots decreases by one. The total decrease is, therefore, proportional to $2N_s + N_h + N_s^{av} + N_h^{av} = N_s + N_0$. The last two terms in the equation require more explanation. We assume that it takes on average τ_h time for a robot to reach home after grabbing a puck. Then the average number of robots that deliver pucks during a short time interval dt and return to the searching state can be approximated as dtN_h/τ_h. Likewise, in a period of time dt, dtN_s^{av}/τ robots leave the avoiding state and resume searching. Interference will increase the homing time for each robot; therefore, in general, homing time will be a function of N_0, τ and τ_h^0, the average homing time in the absence of collisions with other robots. For low to moderate robot densities, it is reasonable to assume the increase will be linear in the interference strength. The effective homing time can, therefore, be modeled as

$$\tau_h = {\tau_h}^0[1 + \alpha_r'\tau N_0]\,. \tag{29}$$

The remaining equations have similar interpretations. We can take advantage of the conservation of the total number of robots to compute $N_h^{av}(t)$. Equations 25–28 are solved numerically under the conditions that initially, at $t = 0$, there are M_0 pucks and N_0 searching robots.

Figure 11 shows the time evolution of the fraction of searching robots and pucks for $M_0 = 20$, $N_0 = 5$, $\tau = 3\ s$, $\tau_h^0 = 16\ s$. The number of searching robots (solid line) first quickly decreases as robots find pucks and carry them home, but then it increases and saturates at some steady state value as the number of undelivered pucks approaches zero (dashed line). The fraction of searching robots in the steady state is inversely proportional to the avoiding time parameter.

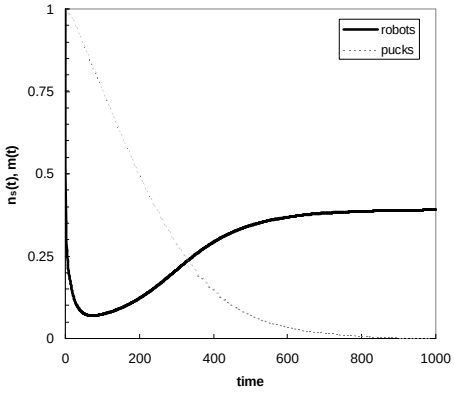


Figure 11: Time evolution of the fraction of searching robots (solid line) and undelivered pucks (dashed line) for $\tau = 3\ s$, $\alpha_p = 0.02$, $\alpha_r = 0.04$, and $\alpha_r' = 0.08$.

In order to compare the performance of different size groups, we define the efficiency of the system as the inverse time required for the group to collect 80% of the pucks ($M(T_{80\%})/M_0 = 0.2$ in Fig. 11(a)). Figure 12(a) shows efficiency

of the group vs. group size for two different interference strengths, as measured by τ. For both cases the efficiency of the group peaks for some group size, indicating an optimal group size for the task. The efficiency is less for the group with a higher interference strength, or larger avoiding time parameter (solid line). Moreover, the greater the effect of interference, the smaller the optimal group size. However, unlike the searching-and-avoiding task, in this case efficiency has a maximum, indicating an optimal group size for the task. Moreover, the greater the effect of interference (larger τ), the smaller the optimal group size.

The final plot (Fig. 12(b)) shows that for this variant of the foraging task interference causes the per-robot efficiency to monotonically decrease with group size — adding a robot to the group decreases the performance of all robots, though if the initial group size was less than the optimal size, adding a robot will increase the overall efficiency of the group.

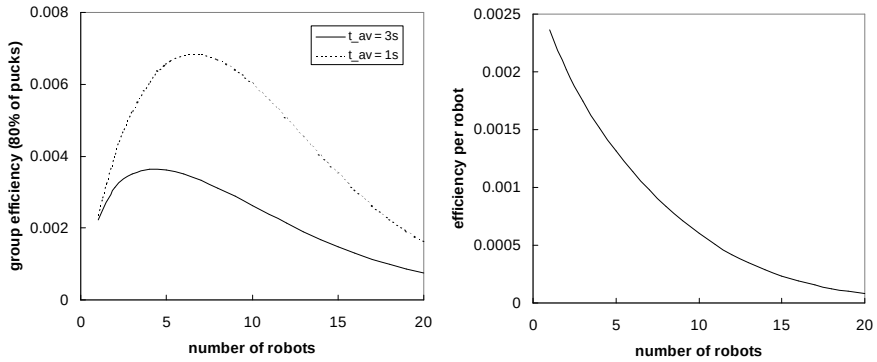


Figure 12: (a) Efficiency of different size robot groups defined as the inverse of the time it takes the group to collect 80% of the pucks in the arena for $\tau = 3\ s$ (solid line) and $\tau = 1\ s$ (dashed line) and $\tau_h^0 = 16\ s$, $\alpha_p = 0.02$, $\alpha_r = 0.04$, $\alpha_r' = 0.08$. (b) Efficiency per robot for different group sizes

3.2.3 Comparison with simulations

We validate the mathematical model by comparing its predictions to the results of foraging simulations. We used Player/Stage to simulate the foraging task with groups of robots. Player/Stage is a client/server-based scalable multi-robot simulator developed at the USC Robotics Lab [14]. Player is a network-based interface to the onboard sensors and actuators that constitute a robot, while Stage supports virtual Player robots, sensing and moving in a two-dimensional bitmapped world, that interact with simulated devices. Available sensor models include sonar, laser rangefinder, pan-tilt-zoom camera with color "blob" detection and odometry.

The Stage world consists of a circular arena, with robots and pucks initially randomly distributed around the arena. Each robot comes equipped with a

ring of 16 sonars, evenly distributed around its perimeter, for the purpose of obstacle avoidance, a color camera and a vision system to locate "colored" pucks, a gripper for picking up the puck, and an odometry system to help robot find "home" and move towards it. We simulated foraging task in groups of one to ten robots, each given a task to collect (or collect and deliver home) 20 pucks. For each group of robots, we averaged results of several, usually ten, simulations. Simulation parameters are listed in Table 1.

Behavior structure The robots' behavior structure closely replicates that of the robots studied in experiments [15]. Behavior-based control governs the actions of the simulated robots. The following behaviors were used:

0 **Search for pucks**: robot executes a random walk around the arena until a puck is found with a camera. The puck is "painted" some bright color, so that it can be seen with a color camera. The size of the puck in the robot's visual field must exceed some minimum detection area (in pixels), before the robot recognizes it as a puck.

1 **Collect pucks**: under this behavior the robot will visually servo towards the puck and collect it with a gripper. The gripper may fail to pick up a puck with some small probability, consistent with failure under experimental conditions due to unreliability of real grippers and sensor update rates.

2 **Go home**: after the puck has been collected, the robot will odometrically servo towards the home location and deposit the puck there. Home is a semicircular region centered on a point at the edge of the arena.

3 **Reverse homing**: the robot moves away from home a specified distance in a random direction.

4 **Avoid collisions**: If a close obstacle (another robot or arena wall) is sensed at any time, the robot will turn away from the obstacle in a random direction at 40 deg/s for a time specified by the avoid time parameter.

For purposes of analysis only, we split behavior **4** into two distinct behaviors: **4**—avoiding collisions while behaviors **0**, **1** and **3** are active, and **5**—avoiding collisions while homing, i.e., , when behavior **2** is active.

A note on calculating parameters In the mathematical models presented below, we will use a set of parameters to connect the model to experiments and simulations. The main parameters we will use are α_p, α_r, the rate at which a robot encounters a puck and another robot respectively. In principle, these parameters can be computed *ab initio* by taking into account the details of the robots dimensions and sensing capabilities in the following way: as a robot travels through the arena, it sweeps out some area during time interval dt and will detect objects that fall in that area. This detection area is $vw_i dt$, where v is robot's speed, and w_i is robot's detection width for object of type i. This number

Parameter	Value	Parameter	Value
# of robots	1 - 10	avoid time	3 s
# of pucks	20	avoid dist	250 mm
robot radius	0.2 m	robot speed	300 mm/s
puck radius	0.05 m	min detect area	200 pixels
arena radius	3 m	rev. homing time	10 s
home radius	0.75 m		

Table 1: Simulation parameters

is the sum of the sizes of the robot and the object it is trying to detect, and the detection distance associated with the sensing hardware it is using to detect that object (*eg.* sonar, camera resolution, *etc.*). If the arena radius is R with N_i objects of type i distributed uniformly around it, a robot will detect these objects at a rate $\alpha_i = vw_iN_i/\pi R^2$. This idealization is useful for roughly estimating model parameters, but because it omits all the details of the experiment (such as sensor errors and failures), it does not get them right. A better way is to estimate them by fitting the model to experimental data, or by calibrating the model by measuring these parameters experimentally or in simulation for a single robot and using this value in the calculations. In order to estimate α_r by calibration, for instance, we have to run the experiment or simulation for two robots in an empty arena, keeping track of the number of times each robot attempts collision avoidance maneuvers. Likewise, to estimate α_p, we have to run the experiment or simulation for a single robot and some pucks scattered around the arena, keeping track of the rate at which the robot picks them up. Although we did not perform these calibrations explicitly, we can estimate the parameters from the simulations data: $\alpha_r = ncollisions/T_{total} = 0.06$ (note that this number includes wall collisions), and $\alpha_p = T_1/(20 \cdot T_{total}) = 0.02$. These numbers are very close to the values we used, which we determined (by eye) to give the best agreement between theory and simulations. Note that this calibration can be done in simulation for a single robot for an environment of arbitrary complexity, and the parameters can be used to study the performance of teams of robots quantitatively in the same complex environment.

We ran foraging simulations for groups of one to ten robots and twenty pucks randomly scattered around the arena. In the results presented below, we split the avoiding behavior into two behaviors: **4**—avoiding while searching, collecting pucks and reverse homing, and **5**—avoiding while the homing behavior is active.

Analysis of Results Table 2 lists the average amount of time (in seconds) each robot spent in the active behaviors during the time it took the group collect the pucks and deliver them home. The last two columns list the average number of times a robot attempted to avoid collisions, both while engaging in the non-homing behaviors and while homing, during the time it took the group to complete the task. Although in all cases all twenty pucks were collected, robots

rbts	0	1	2	3	4	5	colls	hcolls
1	307.64	156.90	265.11	225.84	73.69	21.99	23.7	7.1
2	118.68	81.07	170.02	101.89	46.70	45.08	15.1	13.5
3	94.80	61.48	143.22	61.54	57.65	71.78	17.8	22.1
4	50.98	39.71	131.51	34.06	55.84	99.85	15.9	29.5
5	53.14	29.59	126.84	24.27	69.52	150.66	18.9	41.3
6	67.05	28.89	139.68	20.32	94.26	224.40	22.0	53.3
7	137.90	58.11	111.32	23.69	130.21	184.20	37.0	43.1
8	80.94	32.94	133.35	17.06	123.05	265.56	30.1	62.3
9	74.62	31.36	153.58	15.96	130.10	299.18	33.7	77.7

Table 2: Average time (in seconds) each robot spends in the active behaviors during the foraging task (0: search, 1: collect, 2: home, 3: reverse home, 4: avoid, 5: avoid while homing) as a function of robot group size. The last two columns give, respectively, the average number of avoidance maneuvers per robot while searching/collecting/reverse homing and while homing.

were only able to deliver on average 19.14 ± 0.53 of them. This was caused by excessive crowding near the home location. In the current implementation of the simulator, robots see the already delivered pucks, and if there are no other pucks left in the arena, the robots will all go home. Although reverse homing acts to disperse robots, and eventually all puck should be delivered, we did not run the simulations long enough for this to happen. The total time in the results presented below is, therefore, the time the last of the pucks was delivered.

Figure 13(a) graphically displays the average amount of time each robot spent in the active behaviors while foraging. Fig. 13(b) shows the fraction of the total task time the robot was homing (behaviors **2** and **5** active). Note that the rate of increase in the homing time per robot as a function of group size appears to justify our assumption, Eq. 29, that the homing time increases with the size of the group.

3.3 Dynamic Task Allocation

We studied adaptive task allocation in multi-robot systems [18, 25, 28]. This scenario is based on the foraging task. Consider an arena with some number of pucks scattered about it. The pucks can be of two distinct types, *Red* and *Green*. Each robot can be tasked to collect pucks of a specific type, say *Red*. When the robot's foraging state is set to *Red*, it is searching and collecting *Red* pucks. The robots can also recognize the foraging state of robots that are visible to it. The robots have no *a priori* information about the shape of the arena, the number of pucks left in it or the number of foraging robots. The goal of adaptive task allocation is to design a robot controller that will allow robots to dynamically adjust the division of labor, so that the number of robots searching for *Red* and *Green* pucks will, over time, correctly reflect their prevalence. To achieve this group behavior, each robot must be able to dynamically change its

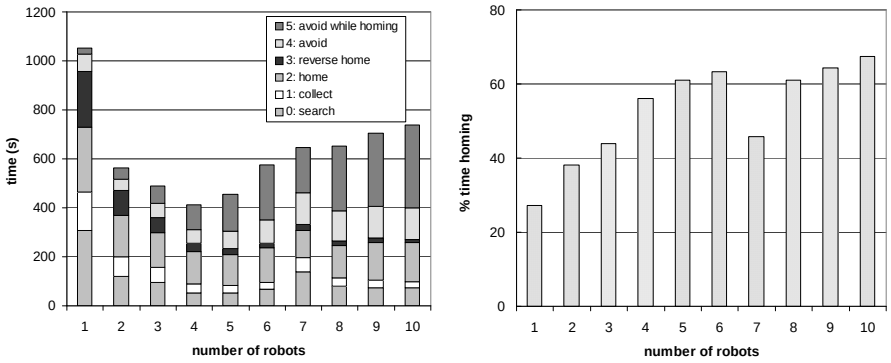


Figure 13: (a) Average time each robot spent in the active behaviors during the time it took the group to deliver all pucks vs robot group size. (b) Percentage of time each robot was homing as a function of group size.

foraging type.

The solution is for each robot to count the number of pucks of each type in the environment as well as the number of robots in each foraging state [18]. It does so by observing pucks and robots that are visible to it and adding these observations to history (memory). At some time interval, the robot uses the history array to estimate the fraction of pucks and robots of each type, and changes its foraging state according to a transition function.

In order to experimentally demonstrate the dynamic task allocation mechanism we made use of a physically-realistic simulation environment. Our simulation trials were performed using Player and Gazebo simulation environments. Player [14] is a server that connects robots, sensors, and control programs over a network. Gazebo [21] simulates a set of Player devices in a 3-D physically-realistic world with full dynamics. Together, the two represent a high-fidelity simulation tool for individual robots and teams that has been validated on a collection of real-robot robot experiments using Player control programs transferred directly to physical mobile robots. Figure 14 provides snapshots of the simulation environment used. All experiments involved 20 robots foraging in a 400m^2 arena.

The robots used in the experimental simulations are realistic models of the ActivMedia Pioneer 2DX mobile robot. Each robot, approximately 30 cm in diameter, is equipped with a differential drive, an odometry system using wheel rotation encoders, and 180 degree forward-facing laser rangefinder used for obstacle avoidance and as a fiducial detector/reader. Each puck is marked with a fiducial that marks the puck type and each robot is equipped with a fiducial that marks the active foraging state of the robot. Note that the fiducials do not contain unique identities of the pucks or robots but only mark the type of the puck or the puck type a given robot is engaged in foraging. Each robot is also equipped with a 2-DOF gripper on the front, capable of picking up a single 8

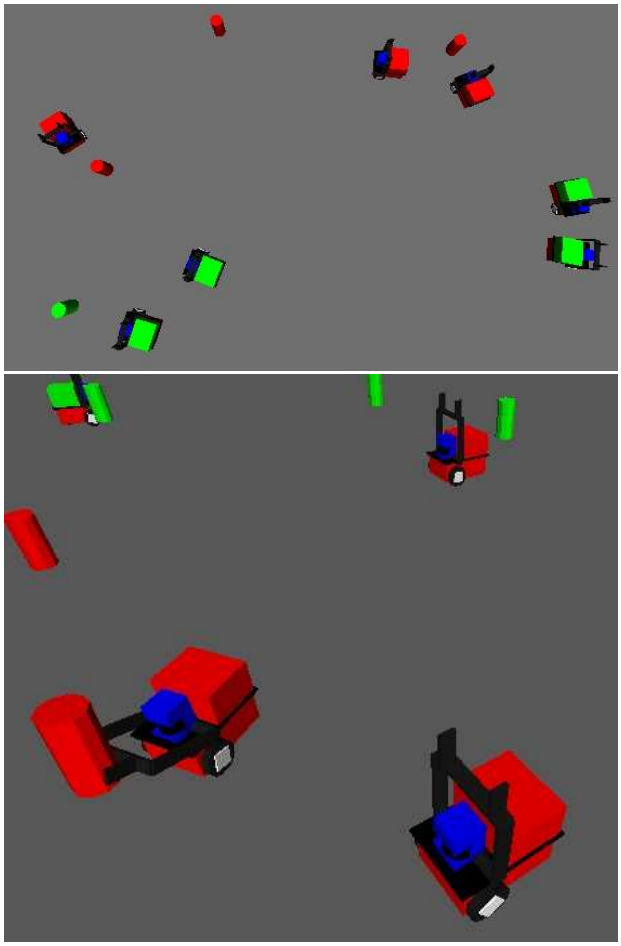


Figure 14: Snapshots from the simulation environment used. (left) An overhead view of foraging arena and robots. (right) A closeup of robots and pucks.

cm diameter puck at a time. There is no capability available for explicit, direct communication between robots nor can pucks and other robots be uniquely identified.

3.3.1 Mathematical Model of Dynamic Task Allocation

At a high level, coarse-grained description, each robot can be considered to belong to either *Green* or *Red* foraging state during a sufficiently short time interval. In reality, each state is composed of several robot actions and behaviors, such as wandering the arena, detecting pucks, avoiding obstacles, *etc.* However, since we want the model to capture how the fraction of robots in each foraging state evolves in time, it is a sufficient level of abstraction to consider only these states. If we find that additional levels of detail are required to explain robot behaviors, we can elaborate the model by breaking each of the high level states into its underlying components.

A robot uses information in its history to make a transition between states. A robot makes a transition to *Red* foraging state according to a transition function that depends on the difference between the estimated fraction of *Red* robots and *Red* pucks; otherwise it makes a transition to the *Green* state.

Let $N_R(t)$ and $N_G(t)$ be the number of robots in *Red* and *Green* foraging states respectively at time t, and $M_R(t)$ and $M_G(t)$ be the number of uncollected *Red* and *Green* pucks in the arena. These dynamic variables correspond to quantities that have been averaged over many experiments or simulations. The following set of differential equations govern how the average numbers of robots and pucks evolve in time. [5]

$$\begin{aligned} \frac{dN_R}{dt} &= \alpha_R(t)N_G(t) - \alpha_G(t)N_R(t) \\ \frac{dM_R}{dt} &= \beta_R M_R(t)N_R(t) + \mu_R \end{aligned}$$

Due to conservation of robots, $N_G = N - N_R$, where N is the total number of robots (likewise, $M_G = N - M_R$). Quantities α_R and α_G govern the rate at which robots switch to *Red* and *Green* states respectively. In an adaptive system, these are time-dependent. Parameter β_R is the rate at which robots encounter *Red* pucks, while μ_R is the rate at which new *Red* pucks are deposited in the arena (likewise for *Green* pucks). For simplicity, μ_R and μ_G are such that the total number of pucks remains constant. Experimentally, this is realized by the replacing a puck in a new random location after a robot picks it up.

It is more convenient to work with the average density, $n_R = N_R/N$, rather than the number of robots. Also, we may safely ignore the equations for pucks,

[5] The differential equations describing the evolution of a dynamical system are usually derived as a continuous limit of discrete time difference equation, for example: $N_R(t+1) = N_R(t) - \alpha_G \Delta t N_R(t) + \alpha_R \Delta t N_G(t)$. The problem with this approach is that it models a *synchronous* system, where all robots make decision at the same time. Although feasible, such a model is not realistic; moreover, most choices of transition rates α_R and α_G lead to severe oscillations in the dynamic variables. The differential equations model we are working with is derived from the stochastic master equation, and is applicable to *asynchronous* systems.

because these quantities do not enter the equations describing time evolution of the number of robots. Dividing both sides of the equation by N, the total number of robots, reduces it to:

$$\frac{dn_R}{dt} = \alpha_R n_G(t) - \alpha_G n_R(t)\,. \tag{30}$$

Likewise, the densities of *Red* and *Green* pucks are $m_R = M_R/M$ and $m_G = M_G/M$.

Transition Rates Equation 30 is a special case of the Rate Eq.7 describing an adaptive system, with α_R and α_G representing the history averaged transition rates $\langle W \rangle_h$. At regular time intervals, the robot looks at the history of observations and estimates the density of *Red* pucks and robots in *Red* state. In general, the transition probability should be a function of $\hat{m}_R - \hat{n}_R$, the difference between the estimated fractions of pucks and robots in a particular state (degenerate to choice of R or G).

At the collective level of Equation 30, the macroscopic transition rates α_R and α_G are in fact simply averaged microscopic transition probabilities:

$$\begin{aligned} \alpha_G &= \alpha \langle f(\hat{n}_R - \hat{m}_R) \rangle_{P(\hat{n}_R, \hat{m}_R)} \\ \alpha_R &= \alpha \langle f(\hat{n}_G - \hat{m}_G) \rangle_{P(\hat{n}_G, \hat{m}_G)} \end{aligned} \tag{31}$$

where α assures the proper time scale, $\langle ... \rangle_P$ stands for averaging over the distribution P, and $P(\hat{n}, \hat{m})$ is the joint probability that a robot has observed the fraction of robots and pucks of a corresponding color to be $\hat{n}$ and $\hat{m}$ respectively. We note that for sufficiently large history lengths can approximate $P(\hat{n}, \hat{m})$ by a sharply peaked distribution around its mean $(\langle n \rangle, \langle m \rangle)$. This suggests that if the microscopic transition functions are smooth enough, then the effect of averaging is to replace the estimated values of densities with their mean values (in the case of the step function, the effect of averaging is to smear out the discontinuity).

A steady state is one in which the densities of robots in *Red* or *Green* states no longer change. Existence of the steady state is of prime interest to the designer, because if a system has one, we can reliably predict its long term behavior. In the adaptive task allocation problem, the desired steady state is one in which the distribution of robots is equal to the distribution of pucks, namely, $n_{R,ss} = m_R$ and $n_{G,ss} = m_G$. In our previous work [25] we showed that in order to achieve the desired steady state, the transition rates must have the following functional form:

$$\alpha_R(\hat{m}_r, \hat{n}_r) = \hat{m}_r g(\hat{m}_r - \hat{n}_r), \tag{32}$$

$$\alpha_G(\hat{m}_r, \hat{n}_r) = \hat{m}_g g(\hat{m}_g - \hat{n}_g) \equiv (1 - \hat{m}_r) g(-\hat{m}_r + \hat{n}_r). \tag{33}$$

Here $g(z)$ is a continuous, monotonically increasing function of its argument defined on an interval $[-1, 1]$. We consider the following forms for $g(z)$:

- *Power:* $g(z) = 100^z/100$

- *Stepwise linear:* $g(z) = z\Theta(z)$.[6]

3.3.2 Comparison with Simulations

Figure 15 shows results of embodied simulations (solid lines) as well as solutions to the stochastic version [28] of the model (dashed lines) for different values of robot history length and forms of transition function (given by Eq. (32) and 33, with $g(z)$ linear or power function). Initially, the *Red* puck fraction (dotted line) is 30%. It is changed abruptly at $t = 500$ s to 80% and then again at $t = 2000$ s to 50%. Each solid line showing *Red* robot density has been averaged over 10 runs. We rescale the dimensionless time of the model by parameter 10, corresponding $\varepsilon = 0.1$. The history length was the only adjustable parameter used in solving the equations. The values of h used to compute the observed fraction of *Red* robots were $h = 2,\ 8,\ 16$, corresponding to experimental history lengths $10,\ 50,\ 100$ respectively. For m_r, the observed fraction of *Red* pucks, we used their actual densities.

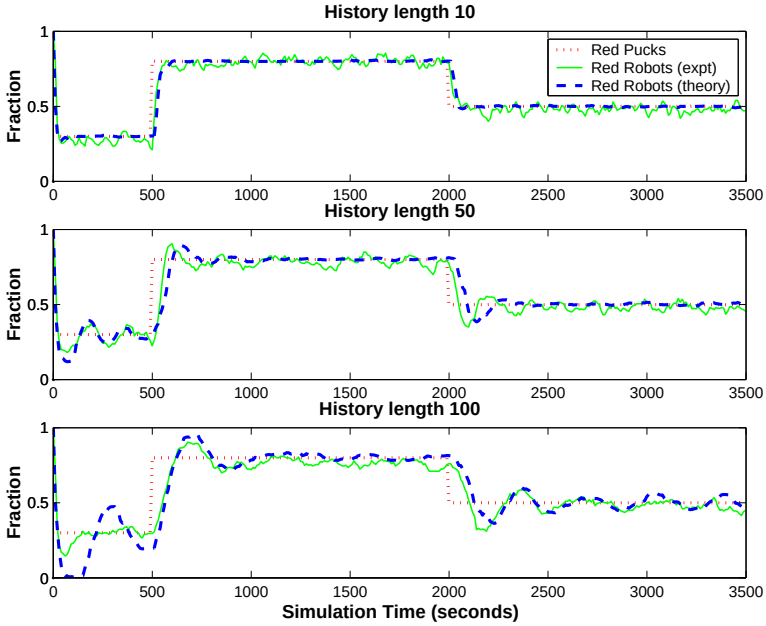
Solutions exhibit oscillations, although eventually oscillations decay and solutions relax to their steady state values. In all cases, the steady state value is the same as the fraction of red pucks in the arena. History-induced oscillations are far more pronounced for the linear transition function (Figure 15(a)) than for the power transition function (Figure 15(b)). For the power transition function, these oscillations are present but become evident only for longer history lengths. This behavior is probably caused by the differences between the values of transition functions near the steady state: while the value of the power transition function remains small near the steady state, the value of the linear transition function grows linearly with the distance from the steady state, thereby amplifying any deviations from the steady state solution. The amplitude and period of oscillations and the convergence rate of solutions to the steady state all depend on history length, and it generally takes longer to reach the steady state for longer histories. Another conclusion is that the linear transition function converges to the desired distribution faster than the power function, at least for moderate history lengths.

3.4 Target Localization with Microscopic Robots

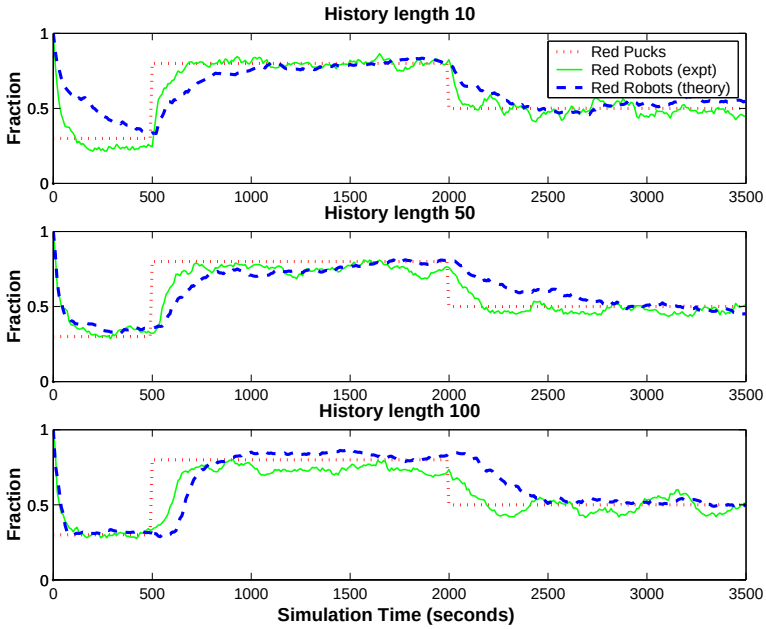
Let us consider a D–dimensional volume with multiple targets that release certain chemical into the environment. The task of the microscopic swarm is to aggregate at these targets in order to carry out some actions in the vicinity of the targets. This capability is fundamental to many medical applications envisioned for these microscopic robots. For example, the volume of fluid may be a blood vessel that has been damaged. Robots are required to aggregate at the injury site in order to assist in healing, forming clots, etc.

We consider a simple robot controller that on a high level can be thought to consist of 3 discrete states described below:

[6]The step function Θ is defined as $\Theta(z) = 1$ if $z \geq 0$; otherwise, it is 0. The step function guarantees that no transitions to *Red* state occur when $m_r < n_r$.



(a) Linear transition function



(b) Power transition function

Figure 15: Evolution of the fraction of *Red* robots for different history lengths and transition functions, compared to predictions of the model

State 1 (search): Do a biased random walk in the direction of the communicative signal concentration gradient.

State 2 (communicate): Move towards the chemical source following the concentration gradient of the target chemical and release communicative signal to other robots.

State 3 (disperse): Move away from the target in the direction opposite to the target chemical's concentration gradient for some specified time τ.

To fully specify a robot's behavior, we also have to describe the transitions between these states. The robots start out in State 1, the searching for targets using random diffusive motion and following the gradient of the communicative signal. Once the concentration of the target chemical at a certain point in space is sufficiently high the robot at that point will switch to the State 2: it will start moving towards regions of high concentration (using biased diffusion or gradient following) while releasing a new chemical which acts as a communication signal to attract other robots. With some probability (that can be fixed, or dependent on the concentration of the robots at the source), robots in the State 2 will switch to State 3, where they will disperse from the source, moving in the direction opposite to the gradient. Finally, robots in the State 3 will switch to the searching state with probability $1/\tau$. The last behavior ensures that robots will not be stuck at local maxima of the chemical potential.

3.4.1 Mathematical Model of Target Localization Using Chemical Fields

Let denote by $n_1(\mathbf{x})$, $n_2(\mathbf{x})$, $n_3(\mathbf{x})$ the fraction of robots in each state at point $\mathbf{x}$, with normalization condition

$$\int d\mathbf{x}\big(n_1(\mathbf{x}) + n_2(\mathbf{x}) + n_3(\mathbf{x})\big) = 1.$$

Let $\rho(\mathbf{x})$ and $c(\mathbf{x})$ be the concentrations of the chemical released from the targets and the communicative signal released by robots in State 2. We also denote by $\mathbf{V}_D^\rho$ and $\mathbf{V}_D^c$ the robots' drift velocity in the concentration gradients of chemical (released by the targets) and communicative signal (released by the robots), respectively. Then the set of equations describing the evolution of the system is as follows:

$$\begin{aligned}
\frac{\partial n_1}{\partial t} &= D_1\nabla^2 n_1 - \mathbf{v}\cdot\nabla n_1 - \nabla\cdot[\mathbf{V}_D^c n_1] \\
&- n_1 F(\rho) + \frac{n_3}{\tau} \qquad (34)\\
\frac{\partial n_2}{\partial t} &= D_2\nabla^2 n_2 - \mathbf{v}\cdot\nabla n_2 - \nabla\cdot[\mathbf{V}_D^\rho n_2] \\
&+ n_1 F(\rho) - G(n_2,\rho,c)n_2 \qquad (35)
\end{aligned}$$

$$\begin{aligned}\frac{\partial n_3}{\partial t} &= D_3\nabla^2 n_3 - \mathbf{v}\cdot\nabla n_3 + \nabla\cdot[\mathbf{V}_D^{\rho} n_3] \\ &+ G(n_2,\rho,c)n_2 - \frac{n_3}{\tau} \end{aligned} \tag{36}$$

where $F(\rho)$ is the concentration–dependent transition rate from State 1 to State 2, $G(n_2;\rho;c)$ is the transition rate from State 2 to State 3, and $1/\tau$ is the probability that a robot in State 3 will switch to State 1.

Equations 34–36 have a simple intuitive interpretation. The first two terms in Eq. 34 describe robots motion in State 1: diffusive searching and following communicative signal, if present. The third term describes the drift in the flow. The fourth term describes transitions to State 2 at the rate $F(\rho)$, which depends on the concentration of the target field. The last term describes transition of robots from State 3 to State 1 after the robots have moved in the direction opposite to the concentration gradient for a period of time τ. $G(n_2,\rho,c)$ is the rate at which robots transition from State 2 to State 3, and it could in principle depend on the local concentrations of the gradients, as well as the number of agents present at the target site, for example, when presence of a certain minimum number of robots is required for executing an action.

We have to complement these three equations with two more to account for the evolution of chemicals ρ and c as follows:

$$\frac{\partial \rho}{\partial t} = D_\rho\nabla^2\rho - \mathbf{v}\cdot\nabla\rho + \sum_{i=1}^{M} Q_i\delta(\mathbf{x}-\mathbf{x}_i) - \gamma_\rho\rho \tag{37}$$

$$\frac{\partial c}{\partial t} = D_c\nabla^2 c - \mathbf{v}\cdot\nabla c + q_c n_2 - \gamma_c c \tag{38}$$

In Eq. 37 $\mathbf{x}_i$-s, $i = 1, 2, ..M$ are the locations of the target sources, Q_i is the intensity of source i, and γ_ρ is the decay rate of the target chemical. Similarly, in Eq. 38 q_c is the intensity of communication signal released by a robot in State 2, while γ_c is the decay rate of the signal.

Simplification: 1-dimension In this section we present results for a $1D$ geometry and a single target scenario. We consider the case when the liquid flow is very slow compared to other time scales so we can set $\mathbf{v} = 0$. Also, since there is only one target, we neglect the third (dispersing) behavior so that two possible states are State 1 ("search") and State 2 ("communicate"). The target is located at $x = 1$ and serves as a point source for the chemical. We assume that the diffusion of the chemical happens much faster compared to robots' diffusion, and it quickly reaches its steady state profile. Hence, the equation for evolution of $\rho(x,t)$ can be solved separately, with a solution

$$\rho(x,\infty) \equiv \rho(x) = Q_0 e^{-\sqrt{\gamma_\rho/D_\rho}(1-x)}, 0 \le x \le 1. \tag{39}$$

For the results presented here we used $Q_0 = 0.1$, $D_\rho = 0.2$ and $\gamma_\rho = 0.5$.

All robots start at State 1 and are initially localized at $x = 0$. We assume that a transition from State 1 to State 2 happens whenever a robot in State

1 detects the target's chemical above a certain threshold level ρ_0, so that the transition rate is $F(\rho) = \theta(\rho - \rho_0)$, where $\theta(x)$ is the step function, $\theta(x) = 1$ if $x \geq 0$ and $\theta(x) = 0$, $x < 0$. While in State 2, robots move in the chemical gradient with a constant drift velocity V_D and release a communicative signal with intensity q_c.

To proceed further, we need to specify the dependence of the drift velocity in State 1 on the concentration of communicative signal c. Again, we assume that once a robot detects communicative signal above certain threshold c_0, it propels itself through the fluid in the direction of the gradient with a constant drift velocity V_D. Then the dynamics of the system is described by the following system of equations:

$$\frac{\partial n_1}{\partial t} = D_n \frac{\partial^2 n_1}{\partial x^2} - V_D \theta(c - c_0) \frac{\partial n_1}{\partial x} - F(\rho) n_1 \quad (40)$$

$$\frac{\partial n_2}{\partial t} = D_n \frac{\partial^2 n_2}{\partial x^2} - V_D \frac{\partial n_2}{\partial x} + F(\rho) n_1 \quad (41)$$

$$\frac{\partial c}{\partial t} = D_c \frac{\partial^2 c}{\partial x^2} + q_c n_2 - \gamma_c c \quad (42)$$

Analysis of Results To study the effect of different design parameters on aggregation behavior of the robots at the target, we solved the system Eq. 40–42 numerically. We used the following parameters (in dimensionless units): $D_n = 0.01$, $D_c = 0.05$, $V_D = 0.1$, $q_c = 0.1$, $\gamma_c = 0.01$. For the detection thresholds we used $c_0 = 0.001$ and $\rho_0 = 0.01$, the later assuring that that robots detect the chemical approximately midway in the interval $[0, 1]$. We used reflective boundary conditions for n_1 and n_2, $\partial n_1/\partial x|_{0,1} = \partial n_2/\partial x|_{0,1} = 0$, and absorbing boundary conditions for c, $c(0) = c(1) = 0$.

In Fig. 16 we plot the spatia–temporal evolution of robots' densities with and without communication. Clearly, the density peak at $x = 1$ is stronger for the system with communicative behavior. This suggests that communication indeed helps the robots to aggregate better. In addition, the aggregation process with communication happens faster than without communication. This is also shown in Fig. 17, where we plot the density of robots at $x = 1$ as a function of time for three different cases: free diffusion[7] ($V_D = 0$), gradient following without communication ($V_D \neq 0, q_c = 0$), and gradient following with communication ($V_D, q_c \neq 0$). As it can be seen from Fig. 17, the systems with gradient following and communicative behavior do demonstrate aggregative behavior, and it is more pronounced for the system with communication. For instance, at time $t = 10$ the robot density at $x = 1$ and with communication is more than 3 times higher than in the non–communicating case.

One of the design objectives is to have robots aggregate at the target fast enough, while at the same time not dissipating too much power due to the propelling. To examine this tradeoff, let us consider the dependence of the

[7]Note that the absence of aggregation for free diffusing robots is due to reflective boundary conditions at the source for n_1 and n_2. If one employs absorbing boundary conditions instead, robots will demonstrate aggregative behavior even with free diffusion.

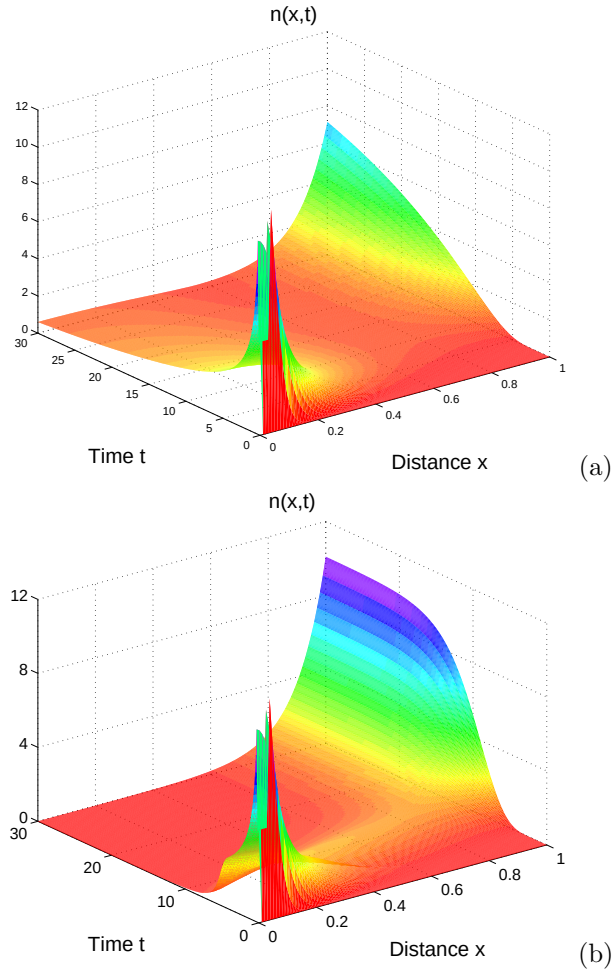


Figure 16: Time evolution of robot densities without communication (a) and with communication (b)

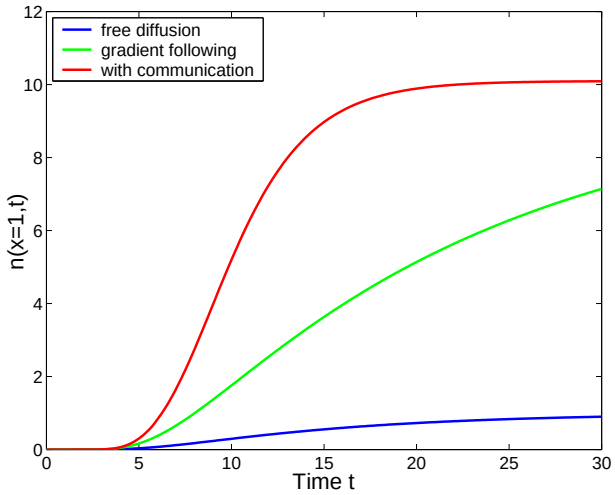


Figure 17: Time evolution of robot densities at $x = 1$ for three different strategies

aggregation time (defined as time needed for fraction n_0 of robots to reach the vicinity of the target which we define as the interval $[0.95, 1]$,) on the drift velocity V_D. In Fig. 18 we plot aggregation time vs V_D for three different values of n_0. One observes that if increasing the drift velocity from $V_D = 0$, the aggregation time decreases monotonically, with a steeper decline for larger n_0. However, it soon "saturates", so that increasing V_D further has very small effect on the aggregation time. This is because for large values of V_D/D_n, the aggregation time is mainly dominated by time required for robots to diffuse and detect chemical gradient, and increasing V_D clearly does not have any effect on this time. Hence, depending on the desired number of robots in the vicinity of the target, as well as the required aggregation time, the best strategy for robots might be to have a moderate drift velocity. Note that this type of analysis can be used to assess the energy–efficiency of various behaviors since power required to propel a robot through a fluid with velocity V_D scales with V_D.

4 Distributed Resource Allocation in the Grid

Grid computing is an emerging technology that enables users to share a large number of computing resources distributed over a network. The dynamic, *federating* nature of Grid policy environments is dominated by virtual organizations (VOs) which associate heterogeneous users and resource providers. Users have resource-consuming activities, or *jobs*, that must be mapped to specific resource providers through a *resource allocation* mechanism. The resource allocation mechanism may choose among alternate mappings in order to optimize some utility metric, within the bounds permitted by the VO policy environment.

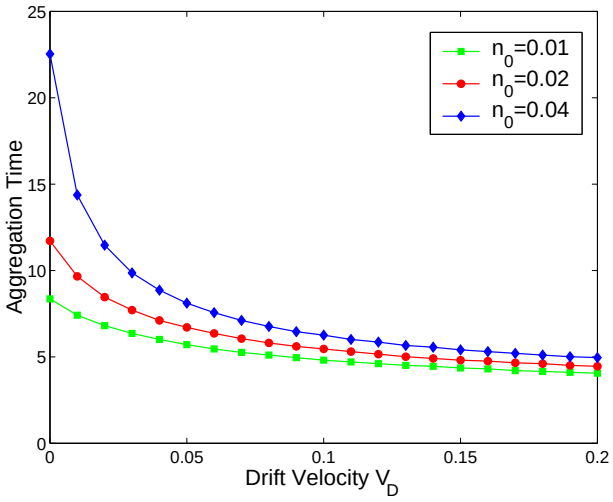


Figure 18: Aggregation time as a function of drift velocity, for three different values of n_0.

It is envisioned that deployment of Grid technology will grow from its current modest scale to eventually overlay the global Web. It is not known how large individual VOs will be, but it is reasonable to imagine resource sharing among populations with tens of thousands of users and thousands of resources. Hence, allocation mechanisms need to be highly scalable and robust to localized failures in resources and communication paths. From the perspective of a single VO, the dynamic policy environment can be viewed as the dynamic arrival and departure of users and resources (occurring at a higher rate than users and resources actually associate with and disassociate from the global Grid infrastructure). Some very large VOs may have an overlayed hierarchical structure, but this structure does not necessarily map to underlying physical or geographic hierarchy. Scalable Grid allocation mechanisms need to focus on the VO policy environment rather than physical locations.

Although there has been considerable attention given to the resource allocation problem in the Grid, very few researchers have addressed the problem from the perspective of learning and adaptation. Meanwhile, the multi-agent systems (MAS) and distributed AI communities have shown that groups of *autonomous learning agents* can successfully solve different load balancing and resource allocation problems [41, 12]. The goal of this paper is to apply multi-agent learning techniques to the problem of resource allocation in the Grid. The MAS approach is well suited for describing the Grid, because the distributed, autonomous nature of agents (Grid users and resources) reflects the federated nature of the Grid. Introducing learning allows the multi-agent system to adapt to changes, such as the changing resource capacities, resource failure, or introduction of new agents into the system. Furthermore, we believe that the MAS

approach will prove useful for policy design, because it can be used to study the performance of a VO implementing a given resource allocation strategy to verify that it does not lead to any unintended global consequences.

4.1 Grid Scheduling Issues

Due to decentralized Grid policies, portions of the Grid may use different allocation strategies, and a centralized allocation manager is not feasible. However, the Grid vision assumes that standard mechanisms will be deployed which can be configured with appropriate localized policies. To a large degree, traditional scheduling systems are distinguished by their strategy, as embodied in algorithms and deployment parameters. The wide deployment of the GRAM [48] job-submission interface has demonstrated that contemporary job scheduling systems are architecturally consistent. Previous Grid architecture work leads us to believe individual users, as well as brokering intermediaries, will apply allocation strategies to their own jobs in addition to the traditional resource providers making allocation decisions for sets of jobs onto large (high-performance or aggregate) resources. Hence, it is imperative to understand what the impact of these decision will be on the efficiency of overall resource utilization in the system. *Understanding of the effects of different resource allocation mechanisms on global system behavior will influence architectural decisions as well as the policies chosen within federated VOs.* In this paper we examine a specific case when the allocation decisions by individual users are based on reinforcement learning and study the global performance of a VO implementing this mechanism.

A further challenge to Grid resource allocation lies in the lack of accurate resource status information at the global scale. The allocation strategies employed by users and brokers have limited real-time environment knowledge at their disposal. This suggests that feasible allocation mechanisms should not depend strongly on the availability of current global knowledge. The multi-agent learning approach studied here relies on minimal monitoring capabilities to compare resources, only requiring that the agent obtain status signals for job requests issued by the same agent. However, a simplifying assumption in our simulations is that an existing discovery and policy-introspection system permits the agents to scope their internal model of available resources to an appropriate rough set.

4.2 Multi-Agent Reinforcement Learning

Reinforcement learning (RL) [44] is a powerful framework in which an agent, for example, a Grid user, learns optimal actions through a trial and error exploration of the environment and by receiving rewards for its actions. The reward (utility) function defines what the good and bad actions are in different situations. The agent's goal is to maximize the total reward it receives. For a single agent in a stationary environment, the problem reduces to finding the optimal policy. In the multi-agent setting, however, the environment is highly dynamic because of the presence of other learning agents, and the usual conditions for

convergence to an optimal policy do not necessarily hold. Nevertheless, various generalizations of single agent learning algorithms have been successfully applied to multi–agent settings.

We construct a multi-agent model of resource allocation for the Grid that is simplified, yet maintains the main features of the Grid environment: heterogeneity of dynamic, large-scale populations of users and resources. In our system, a large number of users submit jobs to one of the resources that are scheduled by a local scheduler according to local policies. The users are modelled as rational, selfish agents that try to maximize their utilities, (i.e., complete their jobs in the shortest possible time). The agents have no prior knowledge about the resources. Instead, they utilize a simple reinforcement learning scheme to estimate the efficiency of different resources based on their past experience. We analyze the global behavior of the system by numerical simulations, and compare it with a baseline algorithm that makes use of a global knowledge of current resource loads. Our results illustrate that reinforcement learning can be used to improve the quality of resource allocation in a large scale heterogenous system.

4.3 The Model

In real Grid applications the problem of mapping resources to specific jobs can be very complex, and may require co-allocation of different resources such as specific amount of CPU hours, system memory, network bandwidth for data transfer, etc. In this paper we neglect the need for co-allocation, and assume that jobs generated by a user require only certain CPU-time so that they are uniquely characterized by their duration.

4.3.1 Resources Providers

The local scheduling of computational tasks is a challenging problem in itself. Usually, resources are characterized by the number and speed of the processors available, system memory, as well as storage space. Multiple jobs can be run simultaneously in the system, with the allocation of the CPUs to the tasks determined by the local scheduling policies. There are many different scheduling frameworks, such as FCFS (First Come First Serve), LJF (Long Job First), etc. Some scheduling algorithms are adaptive: they chose the appropriate scheduling strategy depending on the type of the jobs in the flow.

The scheduling decision for a contemporary batch system is too computationally expensive for us to perform thousands of times per time-step in our agent simulations. In our model, we consider a simplified representation of the resources and local schedulers. Namely, we assume that each resource is characterized by its processing power P which is defined as a CPU time needed to complete a job of a unit length. Within this framework, there is only a single job running at the system at a given time (note that this approach is different from one adopted in Ref[41] where the capacity of the resource was assumed to be shared equally over all the jobs in the queue). For simplicity, we will assume that all the local schedulers prioritize the jobs by their arrival time (FCFS).

4.3.2 Users

In general, users can be thought of as either individual agents that generate jobs and try to map resources for their execution, or as external resource brokers that map jobs on behalf of many individual users. For the sake of concreteness, we consider the first scenario, although the modelling approach developed in this paper can apply to either case. For the case where user-to-broker relationships are relatively static, i.e., based on VO structural policies, we would not expect impact on the simulation scenarios other than that broker agents have higher densities of job request. If the user-to-broker relationship is dynamic, i.e., based on user observation of broker performance, the system behavior may be more dynamic and explicit multi-tier study is required for those scenarios.

We model users as heterogenous selfish agents that try to maximize their utilities. Clearly, one can define agent's utilities in various ways. Often, the agents are interested in minimizing the *waiting time* for the jobs they submit, hence, they will prefer the resource with the minimal (resource performance-normalized) queue length. Another user-centric measure is the *response time* which is the time elapsed between the job generation and its completion. Clearly, this metric depends not only on the length of the queue, but also on the actual processing capacity of a resource: on the resources with larger capacities the actual runtime of a job will be less. Other possible metrics might be based on the accuracy of the completion time prediction.

In this paper, we used weighted contributions of two metrics: $\rho_i = a_i T_w + (1 - a_i) T_{exc}$, where T_w is the queue wait time, and T_{exc} is the job execution time normalized to the duration of the job (i.e., inverse resource capacity). To account for the heterogeneity, the weights a_i were chosen randomly for each agent. Note that using only the second contribution ($a_i = 0$) would bias the selection towards the resource with the highest capacity with no concern about the queue length at that resource. For sufficiently high loads this would lead to infinitely growing queue. To prevent this from happening we used lower bound for a_i at $a_i = 0.2$

4.3.3 Resource Selection

To complete the definition of our model, we need to describe how the agents select resources. As the name "reinforcement learning" suggests, agents use their past experience to choose between the resources. There are many different ways to incorporate reinforcement learning. In this paper we use Q-learning. For each possible action (i.e., selecting a specific resource) the agent keeps a Q-value that indicates the efficiency of that resource in the past. For each new job, agents chooses a resource according to the ϵ-greedy rule: with probability $(1 - \epsilon)$ it choose the resource with the highest Q-value (ties are broken randomly), while with (small) probability ϵ the agent chooses randomly and uniformly chooses among the other resources. After each completed job, the agent gets a reinforcement signal (containing the start-time and the end time for that job), calculates the metric E_i, and translates it into a reward r for resource i that we have

chosen as follows: $r = sign(\langle\rho_i\rangle - \rho_i)$, where $\langle\rho_i\rangle$ is the utility averaged over all the submitted jobs. Finally, the agent updates the Q values according to $Q_{i,t+1} \leftarrow Q_{i,t} + \alpha(r - Q_{i,t})$, where α is the learning rate.

To compare the performance of the RL algorithm we also studied two other resource selection rules:

Random Selection: Agents are choose randomly with uniform probability between the resources. As we will see below, the performance of this algorithm is very limited in the case of widely heterogenous resource capacities that we are interested in.

Least loaded: In this model agents choose the least loaded resource to submit a job. Note that this selection rule assumes that agents have an up to date information about the current utilization level of the resources. This can be done by keeping a **global** registry with the load-level of each resource. To escape crowding effects, where many agents choose the least loaded resource simultaneously, the resource load has to be modified immediately after a job is submitted. This would lead to a near ideal schedule for our scenario. Note, however, that in real environments the information is usually not up to date. We have studied the impact of crowding effect by introducing a parameter p so that once a job is submitted to a resource, the load of that resource is updated only with probability p (for the results presented in this paper we have used $p = 1/4$). This parameter affects the apparent temporal coherence of the global knowledge shared by the agents.

4.4 Experimental Results

In this section we present the results of simulations of our model for $N = 1000$ agents and and $R = 250$ resources. We neglect the network topology and the communication costs associated with it. Instead, we assume that each of the users can submit jobs to any of the resources. At each time step, agents independently generate jobs at rate $P = [0.1, 0.2]$. The length of the jobs are taken randomly from the uniform distribution in the interval $[J_{min}, J_{max}]$. To take into account the wide dispersion in the job sizes in real Grid applications, we chose $J_{min} = 10$ and $J_{max} = 1000 = 100J_{min}$. Note that such a wide dispersion in the job sizes (as well as resource capacities) is typical for the Grid. The capacities of the resources were also chosen uniformly in the interval $[C_{min}, C_{max}]$.

Let us first consider a situation when the dispersion in the resource capacities $C_{max} - C_{min}$ is relatively small, $C_{max} = 350, C_{min} = 650$. To characterize the system performance, we define the *load* of a resource as the total queue length divided by the resources capacity. In Fig. 19 we plot the load in the system averaged over the resources as a function of time. Note that the non–zero average load for the Least Loaded algorithm is due to probabilistic failure to update the load levels after each submission. For small value of job arrival rate (the top figure) the random selection algorithm performs better than both Least Loaded and RL—if the job load is sufficiently low, choosing resources randomly guarantees load balancing. The situation changes drastically as one

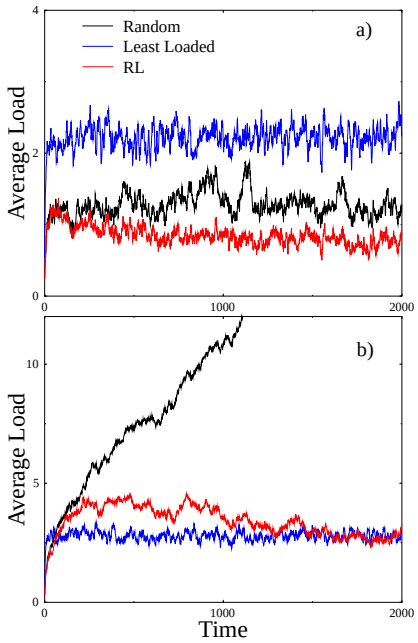


Figure 19: Average load vs time for job arrival rates a)P=0.15 and b)P=0.2

increase the job arrival rate, (or decreases C_{min}). In this case, the performance of the Random Selection algorithm is limited due to "bottlenecks". Because the randomizing agents choose the resources without considering their capacities, for sufficiently high loads the queues on the resources with small capacities will grow indefinitely. This is observed in 19b).

Clearly, the RL algorithm allows the agents to distribute jobs among the resources much more efficiently than the Random selection rule. More remarkably, we find out that for some parameter settings it performs quite well compared to the Least Loaded algorithm, as it is illustrated in Fig 20 where we plot the time evolution of the average job wait time. After a short transient (learning) time the average wait for the RL selection rule falls well below wait time for the Least Loaded. Thus, although the agents do not exchange information nor have any global knowledge on the current load levels in the system, the learning mechanism allows them to efficiently distribute jobs among the resources.

4.4.1 Effect of dynamic agent population

As we mentioned in the Introduction, it is envisioned that the users associated with a VO might join and leave the system dynamically. Hence, it is important to understand what is the effect of this dynamics on the resource allocation

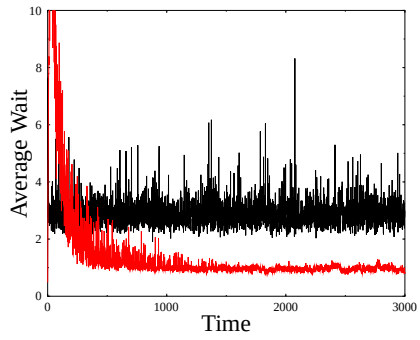


Figure 20: Average wait time for Least Loaded and RL selection rules ($P = 0.15$)

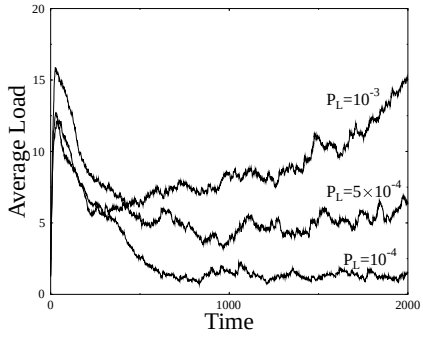


Figure 21: Average load vs time for different leaving probabilities P_L ($P = 0.15$)

mechanism. We addresses this in our simulations by assuming that at each time step each agent has a non-zero probability P_L of leaving the system. For each agent that leaves, we add a new agent, that has to start the learning procedure "from scratch." As one should expect, for small values of the leaving probability P_L the impact of the dynamics is negligible. In other words, introducing small number of *new* agents into the system does not affect the behavior of the others significantly. If, on the other hand, one increases the leaving probability P_L, the situation becomes different: The intrusion of large number of unlearned, and hence exploring, agents, deteriorates the system performance as illustrated in Figure 21, where the average load vs time is plotted for different values of leaving probability P_L.

4.4.2 Significance of Results

The benefit we have observed for the RL algorithm over random selection already suggests an improvement over existing Grid metascheduling strategies, many of which, while performing substantial planning of job sequences etc., make random or otherwise uniform distribution decisions to spread work among several (or many) large-scale resources [3]. Even when metaschedulers attempt to use environmental information, such as load levels, our results suggest that the RL algorithm can provide better adaptive behavior because each metascheduler would learn from the environments responses to its own queries. Divergence of the agent's experience from the resource characteristics published through the monitoring system will change the agent's job distribution. Such divergence can happen due to monitoring system errors, or more likely due to differences in access privilege or priority between the reporting entities in the monitoring system and the metascheduling agent.

A factor in real Grid resource allocation is the latency and limited quality of job status information. Our simulations suggest that the RL algorithm can cope with stochastic reward information, and it might not matter significantly whether the noise in the reward information is due to variance in actual resource behavior or in reporting. However, statistically biased reporting information from resource providers could lead to poor agent behavior. The delay in reward information, e.g., from learning after job completion instead of at job submission time, will length the training period. With bursty job arrival, an agent may perform worse during initial training or during adaptation than if it is able to learn from reward information immediately while processing a cluster of jobs in a short period.

5 Collective Mind Initiative

The overall goal of the *Collective Mind* Initiative is to show that

- Improved Equipment Performance,
- Weapons Effectiveness, and
- Mission Critical Readiness

can come from amassing and sharing collective knowledge derived from the community of equipment via on-board information sharing that embodies the functions and utility of agents. The knowledge found from a fleet of equipment is to be used to improve the overall performance of the fleet, each single piece of equipment in the fleet and all equipment with similar components.

Example 1 Commercial: examination of the logistics records from the collection of all Honda Accord automobiles shows that it is wise to change the engine-timing belt before 70,000 miles or risk failure soon thereafter.

Example 2 Military: examination of many failures of a military piece of equipment showed that a specific type of high tensile bolt was made from non-

MilSpec poor quality metal. All uses of the bolt in all equipment types were inspected for compliance and replacement if required.

Finding solutions to these examples is at present, human intensive. Our goal is to make it automatic by the development of new Artificial Intelligence Information Intensive techniques that fit the new DOD concepts of Net-Centric Warfare and apply over all of the Services.

We proposed to address the opportunity by the development of an innovative technology: the Collective Mind with Collective Learning and Collective Reasoning forming Collective Intelligence.

The approaches we propose to address these challenges of Collective Mind capitalize upon the existing data in the form of designed engineering models and existing field data; exploit the structure of platforms (equipment) as a network of interacting subsystems; and exploit the heterogeneous experience of the various platforms (equipment). The knowledge produced as a result of these efforts will then be used to improve maintenance procedures in general, provide focused help to the individual maintainer and will be integrated into sophisticated reasoning systems for planning and scheduling - and for determining mission modifications that could be proposed to improve mission operations.

The technical challenges are as follows:

1. Diagnosis and prognosis of individual components, subsystems and entire platforms;

2. Planning and scheduling of logistics, including maintenance, and mission activities to ensure mission success, and

3. Proposing changes in mission operations to ensure mission success. The key technical areas proposed for the work here are:

    - Reasoning: using substantial, appropriately represented knowledge,
    - Learning: from experience so that the system performance improves
    - Explanation of actions and recommendations, and
    - Robust Response to surprise and contingencies.

A collective of units can be made from data from many units in the field. Structural and statistically based algorithms would match successful and failed performance to maintenance procedures, equipment status and environmental conditions to identify or learn better ways to maintain equipment.

The scientific issues underlying this work are individualization and emergent behavior, model compositionality especially the composing of multimodal learning systems, with different strengths in different conditions.

We ran several workshops on the Collective Minds as well as holding many smaller meetings. The first workshop was attended by potential users, military and industrial, and the second and third workshops were attended mainly by researchers from academia and Industry.

The Research Agenda reflect both our opinions as well as those of the many workshop attendees.

5.1 Background

Our definition of the Collective Mind has three parts, Cognition, Action and Learning. Collective Cognition is the "sum" of all knowledge obtained from the collection of all assets in a group and/or similar groups. Collective-based Action comes from deductions from the Collective Mind resulting in a set of feasible actions that encompass group knowledge. Collective Learning is the ability to learn from the collective, mind and actions. This is especially valuable as the individual assets operate in real time in their individual missions.

The application focus of our work on Collective Intelligence is on Mission Critical Readiness of Military Equipment. The details of that domain motivate the working definitions of Collective Mind and serve to give metrics on its efficacy.

We postulate a scenario in which a mission is proposed for the asset; assume that the asset has not been asked to undertake the mission in the past, or if it has, the proposed mission may be in a new environment.

The technical challenge is to improve the mission readiness by ensuring that each and every asset (component, equipment, platform, etc.) employed in the mission is capable of performing the operations specified to achieve the objectives of the mission. This involves not only the selection of assets, but also their preparation for the mission and even their maintenance during operations. The first step is to search and see if there are any assets configured like that needed for the mission that have performed similar mission in a similar environment - and have a similar history. Here the collective is the set of all assets that are the same as the set needed for the mission. We therefore need for every asset a "model" that predicts its performance based upon the condition of its equipment, its past performance and the conditions expected during the ensuing mission. More specifically, we need the vector of performance variables that are some function of the attributes that represent the condition of the asset - perhaps in terms of its components. The condition of the equipment is in turn a function of its operation in the past (the missions it has performed including the environment it has performed in) and its maintenance history.

The Collective Mind for this case is the knowledge from the collective set of all assets as well as the ability to learn from the collective as these assets continue to operate in the missions.

If we consider each asset (component, equipment, platform) as an agent with its own reasoning and learning capability, the Collective Mind is a system of interrelated actions by these agents. The actions are purposeful and the agents are attentive to the actions of other agents. The agents construct their actions, understand that the system consists of connected action by themselves and others, and interrelate their actions within the system. In order to accomplish the foregoing, actions by the agents must be contributions to the goals of the system; there must be a common representation for each agent to understand the actions of others and the results of those actions; and the system must recognize the need for an agent to subordinate its actions to those of the system. In this conceptualization, the actions are really the mental processes of the collective

mind. The Collective Mind is in how the agents contribute and represent all actions, and produce improved group behavior

These actions can be categorized as collective learning and collective reasoning (recognizing that in some cases the distinction may b somewhat arbitrary).

A question is: ... does the Collective achieve the desired behavior?

Formally,

a) Let $f_i(x_i)$ be the function (or task) to be optimized by an individual agent A_i, where $i = 0, \ldots, N$ and N is the number of individuals in the collective, and x_i is the parameter of A_i.

b) Let $G(f_1(x_1), f_2(x_2), \ldots, f_i(x_i), \ldots, f_N(x_N))$ be the function that must be optimized by the entire collective.

We are interested in the characteristics of G. Furthermore, when G is given, can the system automatically determine $f_i(x_i)$ for the individuals? In the simplest case, when G is an additive (sum) function, then every individual should simply maximize $f_i(x_i)$ so that G will be maximized. However, in real world applications, G can be much more complex, and some individuals must "sacrifice" themselves (i.e., minimize their own $f_i(x_i)$) in order to maximize the value of G in the collective situations. Ideally, given a new application G, the "collective" should be able to automatically generate $f_i(x_i)$ for each individual. An example is the voting game, where individuals vote "yes" and "no", but their reward depends on the percentage of yes votes of the people of whole population. This global reward function is collected by a "Referee" and is not known by the individuals [45]. Similarly, if we make certain assumptions on the relationship between individuals (such as they act as constraints), then there are some studies that can perform distributed optimization when G is given and fixed [33].

Collective Learning In general, the collective learns by:

- Collective experience - relates knowledge gained from new experience to prior learning;
- Collective example - relate to events or objects via problem solving simulations; and
- Collective discovery, i.e., improvisation Also, learning takes place in (at least) two dimensions - time and space.

Machine learning is traditionally for single agents. Recently, there is the new trend of multi-agent learning. However, the topic of "collective mind" has several unique features that demand a new paradigm for machine learning that we call "Collective Learning". The new learning problems include: " How do individuals learn the structure (some call it topology) of the organization dynamically? Existing approaches such as hidden Markov models or Bayesian Networks are mostly about learning parameters and they avoid this structural learning problem because it is too hard. " How do individual agents learn a model of the environment and the same time a model of other individuals? Traditionally, these two modeling activities are fixed together, and most people

claim one would subsume the other. In the Collective Mind, these two may be related, but they definitely have different characteristics and require different learning techniques.

The Collective Mind envisions integrating domain knowledge from many sources with real-time data feeds from deployed platforms to support rapid problems identification and response. Sources of domain knowledge include the following:

- The "anatomy" of each platform (What are the components and subsystems? How are they physically located? How are they connected?),
- The "physiology" of each platform (How does each subcomponent of the platform contribute to the overall functioning of the platform? Typically, this is divided into separate models for each subsystem.)
- The maintenance history of each component (When manufactured. History of maintenance actions. Results of previous tests.)
- The deployment history of each platform (What missions has it participated in? Where? Under what environmental condition? How long mothballed? Where?)

The real-time data feeds include the following:

- On-board sensors on each vehicle
- Maintenance events
- Debrief from crew after each mission (or each shift?)

Collective learning involves a distributed set of learning platforms that must learn continually but that only occasionally have opportunities to communicate with each other. Under such conditions, it is not feasible to pool all of the sensor readings from all of the platforms in real-time. Instead, each platform must form its own hypotheses and then, when the opportunity arises, communicate its hypotheses to the other platforms (along with the key supporting data and observations).

Existing learning methods do not have good ways of making use of domain knowledge. Existing learning methods are designed for off-line batch training (e.g., constructing an optical character recognizer by training on a database of 1 million labeled hand-written characters). Existing learning methods are designed for learning from a single combined database, rather than by combining hypotheses from many other learning agents.

Making Learning Knowledge-Guided: Domain knowledge can guide learning in two ways. First, it can suggest the space of hypotheses to consider. For example, a learning system that only had sensor readings must learn to relate overall platform failure directly to the history of sensor readings. It might explain a platform failure in terms of the accumulation of several episodes of operation in

high ambient temperatures. But a knowledge-based learning system could explain the platform failure in terms of failure of the engine caused by the added load placed on the engine by the air conditioning system resulting from the crew needing more air conditioning to operate successfully in high ambient temperatures. A knowledge-based learner can relate sensor readings to individual subsystems and then explain overall platform failure in terms of the failure of certain subsystems.

Second, domain knowledge can constrain the space of possible explanations. If we consider the space of all mappings from raw sensor readings to platform failures, this is an immense space. Rapid learning from small amounts of data requires that the space of mappings be highly constrained. Domain knowledge can constrain the space by recasting it in terms of components and subsystems rather than just raw sensor readings. It can also suggest the direction of possible effects. For example, operating an engine at higher RPMs tends to reduce engine life; operating an engine at extreme temperatures tends to reduce engine life, etc. Without this kind of background knowledge, a learning system would need to consider (and reject) the hypothesis that lower RPMs and normal temperatures reduce engine life!

Existing research in Inductive Logic Programming and Probabilistic Relational Models shows how to use domain knowledge to define the space of possible hypotheses. However, these methods have not been scaled up to large problems or to problems involving very noisy, sensor-based data.

There is only a small amount of research showing how domain knowledge can constrain the space of possible hypotheses considered by the learning system. This research is largely ad hoc. Substantial work is needed to develop good modeling languages for describing the domain knowledge and good ways of converting the domain knowledge into constraints on the hypothesis space.

Making Learning Real-Time: There are two challenges to creating real-time learning systems. The first challenge is to design online versions of existing learning algorithms. There is a lot of existing work on online (real-time) algorithms for training neural networks, linear threshold units, and decision trees. Most batch search and optimization algorithms can be converted into online algorithms in principle. The challenge is to find practical, efficient online versions of these methods.

The second challenge is to make those online algorithms adaptive, by which we mean that they can deal with changing worlds in which new kinds of failures occur, new kinds of sensors become available, old sensors cease to be available, and the probabilities of different faults change because of changes in missions and the ways that platforms are being used in missions.

Existing research in expert-weighting and portfolio algorithms have been shown (theoretically) to adapt rapidly to changes in the phenomena being predicted [47]. A DARPA program could transition this work into real-world systems and show how it can be applied in noisy real-time settings.

Making Learning Collective: The challenge for collective learning is for multiple learning agents to pool their learned knowledge without pooling all of their sensor data. Existing research suggests the following directions to pursue:

- First, research on ensemble learning methods learns to take a weighted vote of the hypotheses learned by each individual learning agent [22]. This has been applied, for example, to solve very large learning problems by randomly dividing the available data into subsets, assigning a separate agent to learn on each subset, and then voting the resulting learned hypotheses. This has been shown to give results comparable to those obtained by training a single system on the entire data set [38].
- Second, research in support vector machines (and related algorithms) shows how to identify the key data points that support an hypotheses [6]. These points are known as the "support vectors", and they are sufficient to reconstruct the hypothesis perfectly. An interesting direction would be for the multiple agents to exchange their support vectors and then use all of these support vectors in learning.

5.2 Collective Reasoning (includes planning and scheduling)

In collective mind, since knowledge and information are distributed among many individuals, it makes reasoning and planning/scheduling much harder. One unique advantage this will offer is that damage to any individuals will not paralyze the entire organization. The individuals should know where to ask for and deliver information, know how to recover information when some nodes die, know what to communicate among themselves in order to make a good plan, and know how to evaluate a new plan/schedule collectively.

Another aspect of Collective Reasoning is how to divide a global task into a "workflow" of smaller tasks so that each small task can be performed by some individuals, and when those small tasks are finished, the results should be assembled in such a way that a global solution can be readily obtained. This is the divide and conquer problem and typically the "Task Allocation" problem.

An important opportunity for research is to integrate learned knowledge into sophisticated reasoning systems. The Collective Mind requires this, because the results of individual component and vehicle prognoses must be used by the mission + maintenance scheduler to decide when and how to schedule platforms for missions and for maintenance. Uncertainty in prognoses must be translated into uncertainty in mission success and/or the need for maintenance.

- There are at least two approaches to incorporating uncertain predictions into complex reasoning:
- propagation of uncertainty and ensembles.

Propagation of uncertainty computes a posterior distribution over random variables of interest (e.g., mission success, expected equipment losses) based on the distributions of other random variables (i.e., the prognostic predictions).

Ensemble methods construct a set of non-stochastic "alternative scenarios" or alternative models and compute schedules based on each scenario. The resulting schedules are then analyzed to identify consensus scheduling decisions

and/or ways of modifying the schedule so that it will succeed under all scenarios. Ensemble methods have been very popular in classification learning, but there has been little research on ensembles for reasoning.

Another research opportunity is to determine and propose modifications to missions both in planning and scheduling of operations. The space of potential mission modifications is huge, so some way is needed to constrain the Collective Mind from proposing ridiculous modifications. This can be viewed as the problem of reasoning about the "utility function" of the commander. We can imagine that several tradeoffs are operating in a battle theater: (a) mission goals, (b) safety of troops, (c) reliability of supply, (d) loss of equipment, (d) speed. There has been recent work on inferring multi-attribute utility functions by observing the choices made by humans, (e.g., in auctions or in video games). We would also like to develop systems that are instructable, so that commanding officer can say "Timing is critical; don't propose any modifications that delay the mission." This learning should begin during exercises and continue into the battlefield. Once the utility function has been acquired, the scheduler can generate and evaluate alternative mission modifications and choose the modifications and choose the modification that has the highest utility.

5.3 Collective Behavior

A collective should yield an emergent whole that is qualitatively more than the sum of the parts. Its functionality ought not to be something that one of the parts could do by itself if only it were bigger. Any such system faces a dual challenge.

- How does the behavior of the individual elements yield the emergent behavior of the whole? (Example from our case: how does local awareness of a platform's own state roll up into a global assessment if the state of readiness of the fleet?
- How can the global functionality be applied to the problem, given that the individual elements are the only sensors and effectors that the system has? (Example: if the system learns that alternators with exposure to high temperature and high humidity have unusually high failure rates, how does that knowledge affect local decisions at the company platoon level?

This issue is at the heart of the need for compositionality, which may be the critical issue at the heart of anything. What makes compositionality difficult is that both individual behaviors of the piece and their interactions are typically nonlinear. If one adopts a centralized approach, these can be addressed relatively simply, but such a solution does not scale well and is not robust against attack or acts of God.

Potential ways for addressing these issues draw heavily on simulation and concepts from statistical mechanics as a body of knowledge about how global properties emerge from locally interacting entities.

These approaches are also relevant when we assume that the computational and communications environment may be constrained - just as in the field.

The major challenge is to determine how the required learning and reasoning functions can be achieved under such constraints, providing graceful degradation (rather than catastrophic failure) as communications and computational power are incrementally degraded. Conventional learning and reasoning algorithms do not decompose neatly onto such an architecture, or at least they have not been shown to be decomposable this way. Swarming approaches, by contrast are ideally suited to such architectures, because of three of their characteristics:

- The individual processes are small compared with the overall system, so they can easily run on cycles scavenged from other applications on embedded processors;
- Each process interacts only with others that are co-located with it in some topology. The best fit comes when this topology is isomorphic with the physical distribution of the platforms, but even if it is not, it does provide a way to limit the interactions among processes and thus function in an environment with bounded communications.
- Their emergent dynamics are robust to incremental changes. They tend to degrade gracefully over wide parameter ranges. (There are, naturally, limits beyond which they cannot function, and characterizing these is critical to a program in this area, but they offer a much broader range of operability than do conventional mechanisms.)

Integration of Monitoring/Diagnosis with Scheduling and Planning: The modular vision of "first we assess the state of our platforms, then we plan the mission" is unrealistic in a highly dynamic environment in which platform state and mission constraints change constantly. We need new mechanisms that can incrementally learn and plan in tandem. Such mechanisms must have the "anytime" characteristic: they quickly produce an approximate answer, and can give more detail if more time and resources are available to them. In the context of closely coupled learning and planning, often an early approximation to one half of the problem (say, learning) can help constrain the space that the planning function must search, and the planner's early results can in turn help focus the learner, leading to more rapid convergence than would be possible in a sequential system.

Swarming algorithms are typically any-time, and have been demonstrated in both classification and planning tasks.

Environmental Integration: The issue here is discriminating between two possible interpretations of an aberrant sensor reading: system malfunction (the system is out of specs) vs. environmental or historical stress (the environment is out of spec). No matter how complete our models of our systems may be, complex electromechanical systems will always have emergent properties that surprise us. We need ways to use other platforms that share the same situation as an implicit engineering model to distinguish (local) equipment failure from

(shared) environmental stress. More generally, we need to compare behaviors across platforms that may not be co-situated right now, but that are near to one another in the space of shared histories.

This challenge relies directly on the notion of proximity among platforms in some topology (physical space-time; history space), and so lends itself naturally to swarming methods, one of whose hallmarks is locality of interaction.

5.4 Improvisation

"Making do" and "taking the initiative" are desirable actions of military personnel, tasked with supporting battlefield operations.

Improvisation involves reworking knowledge in time to meet the requirements of a given situation. Reworking refers to revising or abandoning planned-for procedures. Time is central to improvisation since the improviser's decision cannot be undone once it is done. Finally, meeting requirements means accounting for constraints in the decision setting while acting to meet the goals of the response. The question of when to improvise involves recognizing when planned-for procedures cannot or should not be applied. In problem-solving terms, it may therefore be conceptualized as a categorization problem, in which the ability of likelihood of a decision maker to categorize correctly is influenced by a number of factors, such as penalties associated with making an incorrect choice. The question of how to improvise involves developing and deploying new procedures in real-time. It may be conceptualized as a search and assembly problem, influenced by factors such as time available for planning, risk in the environment and the results of prior decisions.

Collective improvisation is an approach to supporting battlefield personnel when the need to develop and deploy new procedures arises. In collective improvisation, past knowledge - which may be contained in databases such as ontologies and may be operated upon by decision logics - is re-examined and reorganized in order to meet new requirements. Results of these improvisations are then fed back into the system, thereby completing the learning loop.

Related prior work lays the foundation for collective improvisation. Hayes-Roth and colleagues have developed a series of blackboard-style architectures to support and in some cases model improvisation. Models built upon these architectures are enabled with dynamic control, thereby allowing execution of real-time control plans which specify a sequence of tasks, parameter values and constraints. In order to support improvisation and capture the learning involved, the blackboard or any architecture must have access to and understanding of models of the physical systems of the platforms involved in battlefield operations.

5.5 Collective Mind Experiment and Prototype

Collective Mind may be defined as using collective learning and collective reasoning to produce a desired outcome, and thus to use Collective Knowledge of

the fleet to improve any and all individual platforms.The Collective Mind concept was studied, as described above, by several Workshop Groups comprised of experts from both the Academic and Military communities. The technical opportunities excited the Academicians and the practical opportunities excited the Military. It became clear that a proof of concept experiment, if it could be performed within the Study, would serve to crystallize the thinking of both Communities and show immediate tangible results to inspire both. A good experiment necessarily requires a good corpus of experimental data. While military data exists, the best and most accessible corpus of data was found in General Electric Company in their Locomotive Division's contract Maintenance Operation. GE Locomotives are sophisticated electromechanical machines that contain big diesel engines, have to operate in all climates, all weathers and terrains and move heavy equipment. They can be considered therefore as reasonable surrogates to military vehicles such as the Stryker and also, in a sense, to big armored tanks. GE locomotives are on-line via their on-board satellite dishes. The captured data is processed in real time to satisfy the needs of GE's railroad customers who require financial assurances that freight will move from A to B reliably and on-time or else a fee has to be paid in compensation. This problem is similar to the Military field commander who needs to move equipment from A to B reliably and on time for military reasons and with a different, even more severe pay-off function.

We selected Improving Mission Reliability, a component of Mission Operations, as the objective for the "Proof of Concept" experiment. Mission reliability was broadly defined as - given a mission of duration X-days, what percentage of units assigned to that mission are able to complete the mission without a critical failure. The motivation for high mission reliability in both commercial and DOD environments is two-fold. First, it gets the mission performed; second, it makes mission planning and execution more predictable and effective; third, it reduces the logistics footprint required to support a certain level of readiness. In the military domain, this may mean picking the best 5 vehicles to conduct a reconnaissance mission in swampy terrain; in the commercial sector, it may imply selecting the best 5 locomotives to deliver time-critical shipments from coast to cost. This problem is accentuated in the case of new mission types or new equipment platforms when insufficient data exists on how the equipment will behave in that environment.

The paradigm for new platforms focuses on the continuum and tension between engineering test cell projections made before deployment, and retrospective statistical measurement of performance found or measured after a substantial number of missions. During this 'gap' - the first wave of missions on new platforms, both the commander and the maintainer are selecting and repairing units 'blindly' with respect to their equipment's expected behavior. The collective mind approach tries to compensate for the scarcity of operational experience on any single unit by learning from ground performance of 'peer' units with current or past similar deployment experience.

5.5.1 The Experiments

The Study funded an experiment (in reality, a series of experiments) conducted by GE Global Research that applied peer-based learning to predict time-to-failure performance in locomotives. GE as part of their normal business keeps an extensive and perhaps unique data set of field failures and repair actions from customers locomotives. The data was obtained from GE locomotives owned and operated by GR Rail and Union Pacific. The data is obtained from normal computer control systems used in the Locomotive and delivered back to GE by a variety of means including from a satellite dish on each train. That is no special sensors were installed for the normal course of business or for our experiments. The data for our experiment, combined design, utilization and repair information on 1100 locomotives over a 2 year time period. Any individual locomotive's time-between-failures appears chaotic and unpredictable. Caveat: the following is a description of the best industrial practice we could find. GE used extensive files and records in their database.

This project utilized existing systems on the locomotives. No new sensors were added to collect these data. This Collective Mind approach capitalized upon existing data collection methods and did not design a priori new sensors or other data collection systems.

The data for the study was collected from four different sources:

1. Locomotive Design & Engineering Data from GE Rail: GE Rail manufactured the locomotives in this study. As the OEM, GE Rail possessed engineering data on locomotive models, configurations, date of manufacture, date of service, the date EOA service was installed, upgrades, and software modifications.

2. Locomotive Recommendation Data from GE Rail EOATM remote monitoring and diagnostics service: For each locomotive, there was a time-stamped record of when the Expert on Alert (EOATM) system detected abnormal patterns in the fault data leading to a recommendation being issued by GE Rail Locomotive Services. A red or yellow recommendation indicated a problem that was serious and required a fix in the next 7-10 days at most.

3. Locomotive Maintenance Data from Repair Shops: Each red or yellow recommendation used in the experiments was associated with maintenance feedback from railroads or GE repair shops which indicated the exact repair action that successfully fix the problem. Therefore the data included only maintenance intervals where a genuine problem existed on the locomotive that was verified by the maintenance personnel.

4. Locomotive Utilization data from a selected railroad: Each locomotive maintains an on-board record of a number of utilization-related parameters that are collected when a locomotive reaches a railroad yard. These parameters include odometer miles, total megawatt-hours, hours spent

motoring, hours spent in dynamic braking, cumulative engine hours, cumulative engine hours moving, percentage of time spent in each of the eight notch settings (analogous to gear settings) and others.

5. Diagnostics are done in GE's Diagnostic center staffed by a group of extremely experienced engineers who have final decision power over the machine computed recommendation produced via case-based reasoning. The diagnosis and repair recommendations are then sent to the local depot or to the of the train, wherever it is(known through on-board GPS).

The Collective Mind Computational Approach Peer-based learning methodologies were investigated since they provide a transparent, adaptable model mechanism. The particular approach taken was to focus on the representation and reasoning mechanisms of instance-based reasoning. Instance-based reasoning (IBR) relies on a collection of previously experienced data that can be kept in their raw representation. Unlike Case-based Reasoning (CBR), they do not need to be refined, abstracted and organized as cases. Like CBR, IBR is an analogical approach to reasoning, since it relies on finding previous instances of similar problems and uses them to create an ensemble of local models. Hence the definition of similarity plays a critical role in the performance of IBR's. Typically, similarity will be a dynamic concept and will change over the use of the IBR. Therefore, it is important to apply learning methodologies to define and adapt it. Furthermore, the concept of similarity is not crisply defined, creating the need to allow for some degree of vagueness in its evaluation. This issue was addressed by evolving the design of a similarity function in conjunction with the design of the attribute space in which the similarity was evaluated. After developing several exploratory peer-based models, a fuzzy instance-based classifier (FIBC) was used that was designed by an evolutionary search (instead of by a manual process). Specifically the following steps were used:

1. *Retrieval* of similar instances from the Data Base
2. *Evaluation* of similarity measure between the probe and the retrieval of instances
3. *Creation* of local models using the most similar instances (weighted by their similarity measures)
4. *Aggregation* of outputs of local mode to probe

It should be noted that no additional sensors were used for this experiment. All data came from on-board sensors used by the control systems that regulate the various subsystems in the locomotive. This constraint implies that it will not be necessary to over-instrument existing or new platforms to re-apply a similar process and obtain comparable results.

The experimental results reveal that consulting a unit's peers, the Collective, provides a significant increase in the ability to characterize the behavior of that unit in terms of completing the next mission. The peer-based approach is robust

and degrades gracefully with the information loss that is likely to be present in the battlefield. In addition, 'rules of thumb' such as using the newest units on a mission were actually shown to be damaging rather than beneficial.

With this limited data, the use of Evolved Peers provided the best overall accuracy (60.35% = over 3 times better than random selection) for past performance. When the selection was limited to a small fixed number of units, Evolved Peers provided an accuracy of 63.5% (over 10 times better than random selection) for the past performance. Finally, Evolved Peers provided the best overall accuracy (55% = 2.7 times better than random selection and 1.5× better than best heuristics) for future performance.

The Collective (peer-based) approaches have shown great robustness to information loss. This will enable mission reliability for minimally instrumented platforms operating with limited bandwidth.

The experiment showed the applicability of peer-based learning methodologies with evolutionary algorithms to select the best attributes for representing peers and to define similarity measures for identifying the most similar peers for a given unit. By evolving the models over different time slices, it has been shown the ability to dynamically adapt the neighborhoods of peers using incremental operational and maintenance data. In future work, structural design of the attribute space (for the definition of peers) could be extended by using genetic programming in lieu of evolutionary algorithms, and attribute selection and weighting to attribute construction. The fitness function to tradeoff classifier accuracy and confidence could be improved by adding measure of representation parsimony and find Pareto fronts for different tradeoffs.

Generating more sophisticated local models for predictions could also extend the approach. The present assumption was that each peer had a rather "feeble" track-history, which motivated the peer approach to begin with. In situations where the peers have a richer track-history, more complex models, whose parameters could be obtained using a local search method, could be developed. In addition to the aforementioned technical extensions, one or more experiments should be conducted using data describing equipment usage more typical of military operations. For example, the data should include instances of irregular use of equipment, equipment use in diverse environments in a variety of missions, equipment operating conditions that range from none to little usage, normal operations and stressed and overload.

5.5.2 Conclusion of the Experiment

The GE experiment showed significant improvement in fleet performance, so much so that GE has already tested some of the ideas in commercial practice. The military representatives at the meeting where the results were reported gave the work a very good report stating that the technology far exceeded any technology used in the military today. The follow on to this comment has been for the team, Sondheimer, Wallace and Will to continue to brief the personnel in extensively in the Pentagon and at various service locations on the potential benefits of research on the Collective Mind.

5.6 Technology Transfer to the Military

A key imperative from DARPA was to elicit and enlist support from a real military customer.

We made visits/presentations to the following organizations, in each case explaining our concept, listening to their feedback and evolving the concept to make it more suitable for technology transfer

- Department of Defense Condition Based Maintenance + (OSD CBM+)
- Department of Defense Office of Force Transformation (OSD OFT)
- US Air Force Expeditionary Logistics for the 21st Century (eLog21)
- US Air Force Knowledge Services (AFKS)
- US Air Force Research Laboratory (AFRL)
- US Army AMRDEC US Army Future Combat Systems (FCS)
- US Army Logistics Transformation Agency (LTA)
- US Army Materiel System Analysis Activity (AMSAA)
- US Army Objective Force
- US Army Research Laboratory
- US Army RDECOM SMS IPT
- US Army TARDEC Joint Strike Fighter Program Office
- US Marine Corp Systems Command (MARCORSYSCOM)
- US Navy Office of Naval Research

A public presentation was made to all services at the 2004 DoD Maintenance Symposium in Houston, Texas.

Workshops on

We also conducted four Workshops Self-Aware Platforms and the Collective Mind in January 23-24, 2003 at Strategic Analysis Inc. Arlington, VA; February 13-14, 2003, USC-ISI, Marina del Rey, CA; January 13, 2004, Erie, NY; April 1-2, 2004, Marina del Rey, CA.

5.7 Potential Impact

The Collective Mind Workshops showed that the Collective Mind topic was challenging and had military relevance and interest.

The Collective Mind Workshops showed that the Collective Mind topic was challenging and had academic and scientific interest.

The conclusion is that the objective of the DARPA study has been met; both military and research and development communities endorse the concept.

Realization the full benefits of the concept is DARPA-hard. A DARPA program could stimulate important research in collective learning to pursue the above and other emergent applicable ideas.

Robustness: now is the time to challenge learning researchers to develop robust engineering methodologies for deploying learning systems. Machine learning has, so far, taken place under "laboratory conditions" where PhD researchers hand-craft the systems to make them work. As a result, while machine learning provides a revolutionary new method for constructing intelligent systems (such as handwriting recognition, speech recognition, and artificially-intelligent simulated characters in games), the resulting systems do not learn after they are deployed.

Scaling: machine learning currently does not scale. Indeed, even human learning takes place only within the head of each individual person, and society spends billions of dollars to combine and communicate this learned knowledge to other people. The Collective Mind project envisions a learning technology that is able to rapidly combine knowledge learned separately by many distributed agents so that each agent can become a "super agent" that benefits from everything learned by the other agents. This might allow computers to learn very rapidly and identify patterns that people, with our limited ability to combine learned knowledge, cannot detect.

Military system must learn after deployment. This is the key to making all kinds of computer systems adaptive to the needs and environments of their users. Without real-time learning, our hand-crafted intelligent systems will remain brittle and hard-to-use.

Machine learning is currently too slow. Most systems must be trained on thousands or millions of training trials. Learning can be much faster if domain knowledge is available to guide and constrain the process. Success in the Collective Mind project will produce a widely-applicable knowledge-guided learning technology.

The core of Transformation in the Military is rapid learning. Introducing new equipment at a rapid pace demands fast learning. DARPA needs to step up to the challenge. It is DARPA-hard

6 Discussion of Results and Approach

The mathematical methods used to analyze collective swarm behavior are based on viewing individual agents as stochastic Markov processes. In order to construct a description of the behavior of a swarm, we do not need to know the exact trajectories of every agent; instead, we derive a model that governs the dynamics of the aggregate, or average, swarm behavior.

Mathematical models are straightforward to construct and analyze — in fact, they can be easily constructed from details of the individual robot controller [29]. The ease of use comes at a price, namely, the number of simplifying assumptions that were made in order to produce a mathematically tractable model. First, we assume that the robots are functioning in a dilute limit, where they are

sufficiently separated that their actions are largely independent of one another. Second, we assume that the transition rates can be represented by aggregate quantities that are spatially uniform (unless we are explicitly modeling interactions with external fields) and independent of the details of the individual robot's actions or history. We also assume the system is homogeneous, with modeled robots characterized by a set of parameters, each of them representing the mean value of some real robot feature: mean speed, mean duration for performing a certain maneuver, and so on. Real robot systems are heterogeneous: even if the robots are executing the same controller, there will always be variations due to inherent differences in hardware. We do not consider parameter distributions in our models as would be necessary to describe such heterogeneous systems. Finally, mathematical models more reliably describe systems where fluctuations (deviations from the mean behavior) can be neglected, as happens in large systems or when many experimental runs are aggregated. However, the success we achieved in quantitatively predicting and explaining results of experiments and simulations with real robots give us confidence for the validity of our approach.

7 Future Directions

We have successfully demonstrated that mathematical analysis can be used to describe collective behavior of large multi-agent systems, including real robot systems. Analysis is fundamental to building predictable and verifiable systems, two aspects critical to swarm deployment.

Our work also has significant consequences to the design of multi-agent systems. In fact, our vision is to give system designers tools to programmatically synthesize and optimize adaptive controllers for intelligent agents that will make up intelligent swarms.

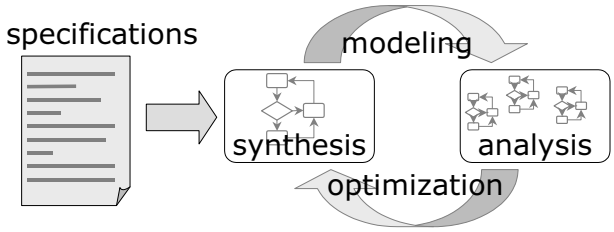


Figure 22: Design cycle of Agent-based computing algorithms

Our vision of the design lifecycle is based on the synthesis→analysis→optimization loop shown in Figure 22. The designer specifies behavior of an individual agent or component: its task requirements, its capabilities, its interactions with other agents, as well as its response to environmental stimuli and contingencies, such as possible actions of an adversary. We then apply Machine Learning techniques to learn the automaton that describes the agent controller. Once we

have the controller, we use the mathematical framework we developed in the course of the TASK program to model the behavior of an ensemble of agents executing this controller. The evolution of the collective behavior can be studied quantitatively, and results of analysis used to guide performance-enhancing modifications in the controller.

The realization of this vision requires further development of our mathematical framework to model more sophisticated agent behaviors, such as learning from experience, learning from environment, or responding to other agents' modifications of the environment. It will also require the development of tools to synthesize agent controllers from their specifications. Our initial study of automatic synthesis specifically [17] and design problem in general [5] shows this to be a promising approach.

References

[1] William Agassounon and Alcherio Martinoli. A macroscopic model of an aggregation experiment using embodied agents in groups of time-varying sizes. In *Proc. of the IEEE Conf. on System, man and Cybernetics SMC-02, Hammamet, Tunisia.*, pages 250–255. Oct 2002.

[2] Brian W. Arthur. Inductive reasoning and bounded rationality. *Am. Econ. Assoc. Papers Proc.*, 84:406, 1994.

[3] B. Barish and R.Weiss. Ligo and the detection of grav- itational waves. *Physics Today*, 52:44, 1999.

[4] D. Challet and Y.-C. Zhang. On the minority game: Analytical and numerical studies. *Physica*, A256:514, 1998.

[5] Valentino Crespi, Aram Galstyan, and Kristina Lerman. Comparative analysis of top-down and bottom-up methodologies for multi-agent systems. In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2005) (poster), Utrecht, Netherlands*, jun 2005.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, MA, 2000.

[7] Radek Erban and Hans G. Othmer. From individual to collective behavior in bacterial chemotaxis. *SIAM J. Applied Math*, 65(2):361–391, 2004.

[8] Miguel Schneider Fontan and Maja J Matarić. A study of territoriality: The role of critical mass in adaptive task division. In P. Maes, M. J. Matarić, J. A. Meyer, J. Pollack, and S. Wilson, editors, *From Animals to Animats 4: Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, pages 553–561, Cambridge, MA, 1996. MIT Press.

[9] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. Resource Allocation in the Grid Using Reinforcement Learning. In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004) (poster), New York, NY*, pages 1314–1315, Jul 2004.

[10] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. Resource allocation in the grid using reinforcement learning. *to appear in J. of Grid Computing*, 2005.

[11] Aram Galstyan, Tad Hogg, and Kristina Lerman. Modeling and mathematical analysis of swarms of microscopic robots. In *Proceedings of IEEE Swarm Intelligence Symposium (SIS-2005), Pasadena, CA*, jun 2005.

[12] Aram Galstyan, Shashikiran Kolar, and Kristina Lerman. Resource Allocation Games with Changing Resource Capacities. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2003), Melbourne, Australia*, pages 145–152, 2003.

[13] Aram Galstyan and Kristina Lerman. Adaptive boolean networks and minority games with time-dependent cutoffs. *Physical Review*, E66:015103, 2002.

[14] B. Gerkey, R. Vaughan, K. Stoey, A. Howard, G. Sukhatme, and M. Matarić. Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1226–1231, Maui, Hawaii, Oct 2001.

[15] Dani Goldberg and Maja J Matarić. Robust behavior-based control for distributed multi-robot collection tasks. Technical Report IRIS-00-387, USC Institute for Robotics and Intelligent Systems, 2000.

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

[17] Chris V. Jones. *A Principled Design Methodology for Minimalist Multi-Robot System Controllers*. PhD thesis, University of Southern California, 2005.

[18] Chris V. Jones and Maja J Matarić. Adaptive task allocation in large-scale multi-robot systems. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'03), Las Vegas, NV*, pages 1969–1974, Oct 2003.

[19] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier Science, Amsterdam, revised and enlarged edition, 1992.

[20] Stuart A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.

[21] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[22] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley & Sons, New York, NY,, 2004.

[23] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.

[24] Kristina Lerman and Aram Galstyan. Agent Memory and Adaptation in Multi-Agent Systems. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2003), Melbourne, Australia*, pages 797–803, Jul 2003.

[25] Kristina Lerman and Aram Galstyan. Macroscopic Analysis of Adaptive Task Allocation in Robots. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS-2003), Las Vegas, NV*, pages 1951–1956, oct 2003.

[26] Kristina Lerman and Aram Galstyan. Two paradigms for the design of artificial collectives. In Kagan Tumer and David Wolpert, editors, *Collectives and Design of Complex Systems*, pages 231–256. Springer Verlag, New York, 2004.

[27] Kristina Lerman, Aram Galstyan, Alcherio Martinoli, and Auke Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life Journal*, 7(4):375–393, 2001.

[28] Kristina Lerman, Chris V. Jones, Aram Galstyan, and Maja J. Matarić. Analysis of dynamic task allocation in multi-robot systems. *submitted to International Journal of Robotics Research*, 2005.

[29] Kristina Lerman, Alcherio Martinoli, and Aram Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In Sahin E. and Spears W., editors, *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342 in LNCS, pages 143–152. Springer-Verlag, Berlin Heidelberg, 2005.

[30] A. Martinoli, K. Easton, and W. Agassounon. Modeling of swarm robotic systems: A case study in collaborative distributed manipulation. *Int. Journal of Robotics Research*, to appear, 2003.

[31] A. Martinoli, A. J. Ijspeert, and L. M. Gambardella. A probabilistic model for understanding and comparing collective aggregation mechanisms. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, volume 1674 of *LNAI*, pages 575–584, Berlin, September 13–17 1999. Springer.

[32] O. Michel. Webots: Symbiosis between virtual and real mobile robots. In J.-C. Heudin, editor, *Proc. of the First Int. Conf. on Virtual Worlds, Paris, France,*, pages 254–263. Springer Verlag, 1998. See also http://www.cyberbotics.com/webots/.

[33] P. J. Modi, P. Scerri, W-M Shen, and M. Tambe. *Distributed Resource Allocation: A Distributed Constraint Reasoning Approach in Distributed Sensor Networks: A Multiagent Perspective*, volume New York, NY. Kluwer Academic Publishers, 2003.

[34] E. Nitz, R. C. Arkin, and T. Balch. Communication of behavioral state in multi-agent retrieval tasks. In Lisa Werner, Robert; O'Conner, editor, *Proceedings of the 1993 IEEE International Conference on Robotics and Automation: Volume 3*, pages 588–594, Atlanta, GE, May 1993. IEEE Computer Society Press.

[35] Esben H. Østergaard, Gaurav S. Sukhatme, and Maja J. Matarić. Emergent bucket brigading - a simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS-01)*, 2001.

[36] Hans G. Othmer and Thomas Hillen. The diffusion limit of transport equations ii: Chemotaxis equations. *SIAM J. Applied Math*, 62(4):1222–1250, 2002.

[37] Lynne Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[38] Chan P.K. and Stolfo S.J. Sharing learned models among remote database partitions by local meta-learning. In *Proceedings of KDD-96*, 1996.

[39] E. M. Purcell. Life at low Reynolds number. *American Journal of Physics*, 45:3–11, 1977.

[40] R. Savit, R. Manuca, and R. Riolo. Adaptive competition, market efficiency, phase transition. *Phys. Rev. Lett*, 82(10):2203, 1999.

[41] Andrea Schaerf, Yoav Shoham, and Moshe Tennenholtz. Adaptive load balancing: a study in co-learning. In Sandip Sen, editor, *IJCAI-95 Workshop on Adaptation and Learning in Multi-agent Systems*, pages 78–83, 1995.

[42] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *(American) National Conference on Artificial Intelligence*, pages 426–431, Menlo Park, CA, 1994. AAAI Press.

[43] Ken Sugawara and Masaki Sano. Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system. *Physica*, D100:343–354, 1997.

[44] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book. MIT Press, Cambridge, MA, 1998.

[45] B. Tung and Kleinrock L. Distributed control methods. In *2nd International Symposium on High Performance Distributed Computing, Spokane, Washington*, pages 206–215, 1993.

[46] Richard T. Vaughan, Kasper Støy, Gaurav S. Sukhatme, and Maja J. Matarić. Blazing a trail: Insect-inspired resource transportation by a robot team. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS), Knoxville, TN*, 2000.

[47] V.G. Vovk and Watkins C. Universal portfolio selection. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*, pages 12–23, 1998.

[48] R. Wolski. Forecasting network performance to sup- port dynamic scheduling using the network weather ser- vice. In *Proceedings of 6th IEEE Symp. on High Performance Distributed Computing, Portland, Oregon*, 1997.