

# Sequential Task Execution in a Minimalist Distributed Robotic System

Chris Jones      Maja J. Matarić  
Computer Science Department  
University of Southern California  
941 West 37th Place, Mailcode 0781  
Los Angeles, CA 90089-0781  
{cvjones|maja}@robotics.usc.edu

## Abstract

The collective execution of a single task, such as foraging or clustering, has received considerable research attention in the minimalist distributed robotic systems (MDRS) community. In contrast, achievement of sequential tasks by MDRS has so far been considered in only a handful of studies. Sequential task execution requires a collective system to carry out a task, and then, in a coordinated fashion, move on to another task. This paper describes work in controlling a minimalist distributed robotic system in sequential task execution. We present two MDRS algorithms for sequential task execution in the foraging task domain, and validate them experimentally in simulation. One of the algorithms uses temporal behavior activation, the other makes use of probabilistic behavior activation. Both are effective in the partially-observable, non-stationary environments we tested them in, and their relative strengths are compared analytically.

## 1. Introduction

A Minimalist Distributed Robotic System (MDRS) is a society of simple robots, with each robot limited to only local sensing, control, and very simple capabilities in terms of intelligence and communication. Such robots maintain little or no state information, extract limited, local, and noisy information from their available sensors, and, in most MDRS implementations, cannot explicitly communicate with other robots in the system. In many cases, the robots are not even aware that other robots exist, or in any case cannot, with their simple sensors, distinguish them from other objects and obstacles in the environment. In spite of all these limitations, MDRS have been shown to be highly effective at certain collective tasks discussed below.

The aim of this work is to study ways of providing such MDRS with the capability of executing sequential

tasks. Sequential task execution in a distributed system is described by (Bonabeau et al., 1999) as “individuals tend[ing] to perform the same task before switching in relative synchrony to another task.” This capability is essential in a variety of task classes, especially those involving multiple, sequentially dependent goals.

This paper is organized as follows. In Section 2 we provide the motivation and relevant related work. In Section 3 we give a detailed description of the foraging task we use for algorithm evaluation in the rest of the paper. In Section 4 we describe our experimental task domain for empirical evaluation of sequential foraging in MDRS. In Section 5 we present two algorithms for sequential task execution, one using temporal behavior activation, the other using probabilistic behavior activation, and experimentally verify their performance on a set of sequential foraging tasks. In Section 6 we describe and analyze the experimental results, discuss them in Section 7, and draw conclusions about their effectiveness in Section 8.

## 2. Motivation and Related Work

MDRS have been shown to be a powerful platform for efficient, robust, and scalable task solutions to collective tasks in dynamic environments (Matarić, 1995b, Cao et al., 1997). Although consisting of extremely limited robots, such systems are capable of executing increasingly complex collective tasks. However, to date most MDRS have been designed for the achievement of a single task, such as object foraging, sorting, or clustering. In contrast, our work described here is focused on *sequential task execution in a MDRS, which requires a set of tasks to be executed in a specified order, with the initiation of a task occurring only after the termination of a required prior task.*

The addition of sequential task execution capabilities to a MDRS greatly increases its functionality. (Théraulaz et al., 1998) describe how the adaptability of complex social insect societies is increased by allowing

## Report Documentation Page

*Form Approved  
OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE <b>2005</b>	2. REPORT TYPE	3. DATES COVERED -			
4. TITLE AND SUBTITLE <b>Sequential Risk Execution in a Minimalist Distributed Robotic System</b>		5a. CONTRACT NUMBER			
		5b. GRANT NUMBER			
		5c. PROGRAM ELEMENT NUMBER			
6. AUTHOR(S)		5d. PROJECT NUMBER			
		5e. TASK NUMBER			
		5f. WORK UNIT NUMBER			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Defense Advanced Research Projects Agency, 3701 North Fairfax Drive, Arlington, VA, 22203-1714</b>		8. PERFORMING ORGANIZATION REPORT NUMBER			
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)			
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)			
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES <b>The original document contains color images.</b>					
14. ABSTRACT <b>see report</b>					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>		<b>9</b>	

members of the society to dynamically change tasks (behaviors) when necessary. Giving each robot the ability to dynamically change behaviors allows the MDRS to operate in a domain requiring the simultaneous regulation of many goals. This is analogous to the many interwoven tasks seen in social insect colonies, such as foraging, nest building, and brood sorting. These tasks in social insect colonies are interwoven through the inter-related proportions of individuals participating in various parts of the system. For example, the number of workers involved in foraging is related to the amount of nest building work available, which may depend on the state of the colony’s young, etc.

In general, the accomplishment of a set of sequential tasks requires sufficient information about the progress on the task in order to determine the appropriate action to take at any given time, and in particular at key steps of transitioning between tasks. However, in a MDRS, because of the robots’ very limited sensing, intelligence, and communication capabilities, there are many domains in which gathering information on the current state of task progress, part of the global state of the environment, may not be possible for the individuals in the system. Formally, to the individuals in a MDRS, the world is partially-observable and highly non-stationary, yet they must collectively achieve a global goal whose changing state they cannot perceive. This is the challenge our work is addressing.

In the research area of simulation and study of insect colonies and their behaviors, some of the most relevant work to ours is in sequential control of ant cemetery organization, ant brood sorting, and social insect nest building (Franks and Sendova-Franks, 1992, Franks et al., 1992). (Bonabeau et al., 1996) describe mechanisms of task regulation in insect societies through the use of response thresholds for task-related stimuli. In their model, members of a society participate in a task when the strength of the task-related stimuli is greater than some threshold. The motivation for our work comes from the task succession models presented in (Bonabeau et al., 1999) and (Bonabeau et al., 1994), which demonstrate the use of probabilistic local action selection in distributed construction and show it to result in increased coordination in the simulation of wasp nest construction.

We provide a brief summary of related work in physical MDRS, using robots similar to those our system is modeled on. (Beckers et al., 1994) demonstrate the collection and clustering of heterogeneous objects into homogeneous clusters. (Matarić, 1995a) provides early work on group coordination in MDRS using a collection of simple basis behaviors. (Werger and Matarić, 1996) demonstrate chain formation and its use for foraging in a MDRS. (Martinoli et al., 1999) demonstrate object clustering in a minimalist robotic system as

well as probabilistically modeling the robots’ physical behaviors. (Werger, 1999) shows MDRS coordination techniques applied to navigation in robot soccer. (Holland and Melhuish, 2000) use probabilistic behavior selection in minimalist robotic clustering and sorting. (Goldberg and Matarić, 2002) precisely define the foraging task for MDRS, provide a collection of general distributed behavior-based algorithms and their empirical evaluation.

### 3. Sequential Foraging Task

In the domain of MDRS, the foraging task - gathering a set of objects and transporting them to a home region - has been studied extensively. In its standard form, foraging is a single, non-sequential task, in that objects are transported in no particular order. We are using a sequential variation of foraging, in order to investigate the capabilities of a MDRS on sequential task execution.

#### 3.1 Task Description

Sequential foraging, in contrast to standard foraging, requires a collection of objects (pucks) to be collected in a specified order. Initially, the environment contains a collection of pucks whose number and distribution are not known to the MDRS. The collection of pucks consists of three distinct types: Puck<sub>Red</sub>, Puck<sub>Green</sub>, and Puck<sub>Blue</sub>, and the types are assumed to be distinguishable by the individual robots. The pucks are to be foraged in order of type; in our experiments the order was: Puck<sub>Red</sub> are to be collected before Puck<sub>Green</sub>, which are to be collected before Puck<sub>Blue</sub>.

As discussed above, due to the limited capabilities of the robots and the dynamics of the task and environment, it is not practical to assume the robots in our MDRS are capable of knowing the current global state of the environment or of task progress. This means that no robot has or can obtain global information such as the size and shape of the foraging arena, the initial number of pucks to be foraged (total or by type), the current number of pucks remaining to be foraged (total or by type), the number of pucks already foraged (total or by type), or the current number of active foraging robots. Also, it cannot be assumed that any robot or subset of robots will always be operational, that the number of foraging robots will remain constant, or that the pucks will remain in their initial positions until they are collected. Despite these constraints, as will be demonstrated below, MDRS are still capable of carrying out the sequential foraging task without the aid of extended sensing, keeping of history, or inter-agent communication.

### 3.2 Sequential Foraging Evaluation Metric

Toward proper evaluation of algorithm performance, we developed a cumulative metric that reflects the sequential requirements of the task. The metric, initialized to 0 at the start of every experiment, is updated at every simulation time-step (approximately every 0.1 seconds of simulated real-time). At each update, for all pucks,  $Puck_{New}$ , that are deposited in the home region at time  $t$ , the utility value,  $Util(t)$ , is updated according to the procedure:

```
Util(t) = Util(t-1)
for all puck in PuckNew
  if (puck == PuckGreen) then
    Util(t) = Util(t) + PropRed
  else if (puck == PuckBlue) then
    Util(t) = Util(t) + PropRed * PropGreen
```

Therefore, the maximum utility for a given experimental trial is equal to the number of  $Puck_{Green}$ s plus the number of  $Puck_{Blue}$ s. This maximum utility value is achieved only if all the  $Puck_{Red}$ s are collected before any of the  $Puck_{Green}$ s and  $Puck_{Blue}$ s, and if all the  $Puck_{Green}$ s are foraged before any of the  $Puck_{Blue}$ s. Although the utility function is not directly incremented by the successful collection of a  $Puck_{Red}$ , the foraging of  $Puck_{Red}$ s is implicitly incorporated into the utility function because the foraging and  $Puck_{Green}$ s and  $Puck_{Blue}$ s are only given full utility value if all  $Puck_{Red}$ s have already been collected.  $Puck_{Green}$ s and  $Puck_{Blue}$ s are given partial credit if foraged before all required prior pucks have been foraged based on the percentage of total required prior pucks already foraged.

At the end of an experimental trial, terminated at time  $t_{Final}$ , the sequential foraging algorithm is given a final utility value,  $Util_{Final}$ , based on the following formula:

$$Util_{Final} = 100.0 * (Util(t_{Final}) / (TPuck_{Green} + TPuck_{Blue})) \quad (1)$$

where  $TPuck_{Green}$  and  $TPuck_{Blue}$  are the total number of  $Puck_{Green}$  and total number of  $Puck_{Blue}$  in the environment, respectively.

The maximum possible  $Util_{Final}$  value is 100, representing perfect execution of the sequential foraging task.

## 4. Simulation Environment

All simulations were performed using Player and Stage. Player (Gerkey et al., 2001), is a server that connects robots, sensors, and control programs over the network. Stage (Vaughan, 2000) simulates a set of Player devices. Together, the two represent a high-fidelity simulation tool for individual robots and robot teams which has

been validated on a collection of real-world robot experiments using Player and Stage programs transferred directly to physical Pioneer 2DX mobile robots.

### 4.1 The Robots

The robots used in the experimental simulations are realistic simulations of the Pioneer 2DX mobile robot. Each robot, approximately 30 cm in diameter, is equipped with a differential drive, a forward-looking 180-degree field-of-view SICK laser rangefinder (used for obstacle avoidance in our work), and a forward-looking Sony color camera with a 45-degree field-of-view (used for puck detection and classification). The simulated robots also rely on a Global Positioning System (GPS), which is not available on physical indoor Pioneers, and is in our simulation work used only to determine the direction of travel when homing. Importantly, no history is kept based on the GPS information, including past puck location. Each robot is equipped with a 2-DOF gripper on the front capable of picking up and transporting a single puck at a time. The gripper has a break-beam sensor that can detect when something is between the gripper jaws.

### 4.2 Robot Behavior-Based Controller

All robots ran identical behavior-based controllers consisting of the following mutually exclusive behaviors: Random Walk, Collision Avoidance, Visual Servo, Grasp Puck, Drop Puck, and Homing. Descriptions of the behaviors used to implement the foraging algorithms are given below.

- The **Visual Servo** behavior causes the robot to visually servo toward the nearest puck detected by the vision system.
- The **Grasp Puck** behavior causes the robot to stop, close, and raise the gripper.
- The **Homing** behavior causes the robot to turn and move on a direct path toward the home region.
- The **Drop Puck** behavior causes the robot to stop, lower, and open the gripper.
- The **Collision Avoidance** behavior causes the robot to stop and turn away from a detected obstacle (arena wall, another robot) at a random turn-rate in the range [20,40] degrees/time-step for a period of 15 time-steps.
- The **Random Walk** behavior causes the robot to turn at a random turn-rate in the range [-20,20] degrees/time-step for a period of 20 time-steps.

Each behavior above has a set of activation conditions, based on the relevant sensor inputs. When met, the conditions cause the behavior to become active. A description of instances in which each activation condition is

true (1) is given below. In all other instances, the activation condition is false (0).

- The **Obstacle Detected** activation condition is true when an obstacle is detected by the laser scanner within a distance of 60 cm.
- The **Puck<sub>Det</sub> Detected** activation condition is true if a puck is detected by the color camera within a distance of approximately 120 cm. The detected puck is of type *Det* (e.g. red, blue, green).
- The **Grasping Puck** activation condition is true if the robot’s gripper is closed and raised.
- The **Gripper Break-Beam On** activation condition is true if the break-beam sensor between the detects something between the gripper jaws.
- The **Inside Home Region** activation condition is true if the robot is inside the home region. GPS is used to determine if the robot is inside the home region.

### 4.3 Experimental Environments

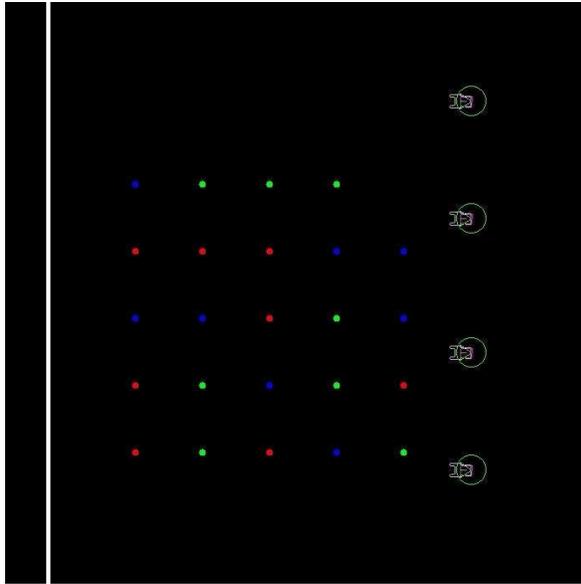


Figure 1: Sequential Foraging Arena. The four robots are lined up on the right, the pucks are the circles in the middle of the arena, and the home region is behind the white line on the left. Initially, the different puck types are distributed randomly.

As shown in Figure 1, the experimental environment consists of an arena with an initial collection of pucks located evenly in the center, their different types distributed randomly, and a home region on one side, to which the pucks are to be transported. Whenever a puck

is deposited in the home region, it is removed from the arena.

We used a group size of four robots in all experiments; and a fixed initial state with their locations on the right side of the arena, as shown in Figure 1.

Our experimental design involved the use of four different environment variations on the above arena, all with four robots simultaneously performing the sequential foraging task. The experimental environments varied in the relative proportion of puck types and the size of the foraging arena. Initial conditions of all four environments were held constant for experiments with all foraging algorithms. The characteristics of the four environments are shown in Table 1.

The four environments were designed to evaluate the adaptability of sequential foraging algorithms along two dimensions: 1) the relative puck type proportions and 2) the arena size. Environment 1 is the base case. Environments 2 and 3 vary the relative puck type proportions: Environment 2 has a high proportion of Puck<sub>Red</sub> and Environment 3 has a high proportion of Puck<sub>Blue</sub>. Environment 4 increases the arena size to four times the foraging area found in Environments 1-3.

## 5. Sequential Foraging Algorithms

We developed and tested two foraging algorithms: Timer-Based Foraging and Probabilistic Foraging. These were investigated and analyzed to assess their effectiveness in the sequential foraging task and their adaptability to different environmental characteristics. As a baseline for comparison, a traditional, non-sequential foraging algorithm, Standard Foraging, was also analyzed.

### 5.1 Standard Foraging

The Standard Foraging algorithm uses the behavior network shown in Table 2. In the behavior network, 1s mean the activation condition must be active, 0s mean it must not be active, and Xs mean the state of the activation condition is irrelevant. There is no notion of sequential foraging in the Standard Foraging algorithm as no distinction is made among puck types. The performance of this algorithm is used as a baseline for comparing the sequential foraging capabilities of the Timer-Based and Probabilistic Foraging algorithms.

### 5.2 Timer-Based Foraging

In the Timer-Based Foraging algorithm, each robot uses an internal timer to dictate which puck type should be foraged at a particular time. Each robot has its own independent timer and timers across robots are not explicitly synchronized.

Each robot’s timer,  $\text{Timer}_{\text{Robot}}$ , which is initialized to

Env #	Arena Size(m)	Total Pucks	Puck <sub>Red</sub>	Puck <sub>Green</sub>	Puck <sub>Blue</sub>
1	8.75 x 8.75	24	8	8	8
2	8.75 x 8.75	24	14	8	2
3	8.75 x 8.75	24	2	8	14
4	17.5 x 17.5	24	8	8	8

Table 1: Experimental Environments

Obstacle Detected	Puck Detected	Grasping Puck	Gripper Break-Beam On	Inside Home Region	Active Behavior
0	1	0	0	X	Visual Servo
0	X	0	1	X	Grasp Puck
0	X	1	1	0	Homing
0	X	1	1	1	Drop Puck
1	X	X	X	X	Collision Avoidance
0	0	0	0	X	Random Walk

Table 2: Behavior Network for Standard Foraging

0 at the beginning of an experiment and incremented by 1 at each simulation time-step of 1/10th of a second. A set of timer alarms are used to control which puck types can be foraged at a given  $\text{Timer}_{\text{Robot}}$  value. There is a timer alarm for each puck type:  $\text{Alarm}_{\text{Red}}$ ,  $\text{Alarm}_{\text{Green}}$ ,  $\text{Alarm}_{\text{Blue}}$ , respectively. When a puck is detected, a decision is made about whether to visually servo toward the detected puck; the decision is based on comparing the robot’s  $\text{Timer}_{\text{Robot}}$  value with the timer alarm value for the detected puck type. If the  $\text{Timer}_{\text{Robot}}$  value is greater than the timer alarm for the detected puck type, the robot’s  $\text{Timer}_{\text{Robot}}$  value will be reset back to the alarm value of the detected puck type and the robot will begin visual servoing toward the detected puck. Using  $\text{Timer}_{\text{Robot}}$  with appropriately set timer alarms, any robot can be made to sequentially forage by puck type.

For the following examples on how the  $\text{Timer}_{\text{Robot}}$  and timer alarms work, assume the  $\text{Timer}_{\text{Robot}}$  and Alarm settings as shown in Table 4.

$\text{Timer}_{\text{Robot}}$	$\text{Alarm}_{\text{Red}}$	$\text{Alarm}_{\text{Green}}$	$\text{Alarm}_{\text{Blue}}$
800	0	750	1500

Table 4: Example  $\text{Timer}_{\text{Robot}}$  and Alarm Settings for Timer-Based Foraging

Given these settings, if the robot detects a  $\text{Puck}_{\text{Red}}$ , the robot’s  $\text{Timer}_{\text{Robot}}$  will be reset to  $\text{Alarm}_{\text{Red}}$ , in this case 0, and the robot will visually servo toward the detected  $\text{Puck}_{\text{Red}}$ . If the robot detects a  $\text{Puck}_{\text{Green}}$ , the robot’s  $\text{Timer}_{\text{Robot}}$  will be reset to  $\text{Alarm}_{\text{Green}}$ , in this case 750, and the robot will visually servo toward the detected  $\text{Puck}_{\text{Green}}$ . With the above timer settings, a detected  $\text{Puck}_{\text{Blue}}$  will be ignored as the robot’s  $\text{Timer}_{\text{Robot}}$  value is less than the value of  $\text{Timer}_{\text{Blue}}$ ,

and the  $\text{Timer}_{\text{Robot}}$  value will remain unchanged. In this example, a  $\text{Puck}_{\text{Blue}}$  cannot be foraged until the robot’s  $\text{Timer}_{\text{Robot}}$  value is greater than 1500, the value of  $\text{Alarm}_{\text{Blue}}$ .

To implement the Timer-Based Foraging algorithm on the robot, we used the behavior network shown in Table 3, where  $\text{Puck}_{\text{Det}}$  is the detected puck type and  $\text{Alarm}_{\text{Det}}$  is the robot’s timer alarm value for the detected puck type. For example, if a  $\text{Puck}_{\text{Red}}$  is detected,  $\text{Alarm}_{\text{Det}} = \text{Alarm}_{\text{Red}}$ .

### 5.3 Probabilistic Foraging

The Probabilistic Foraging algorithm uses two probabilistic behavior activation conditions in each robot’s behavior network in order to encourage sequential foraging.

The first probabilistic activation condition introduced is whether a robot should visually servo toward a detected puck or ignore the detected puck and perform a random walk. Each robot has an assigned probability of ignoring a detected puck of each type. For the three puck types, these probabilities are:  $\text{PIgnore}_{\text{Red}}$ ,  $\text{PIgnore}_{\text{Green}}$ , and  $\text{PIgnore}_{\text{Blue}}$ , respectively.

Whenever the activation conditions for the Visual Servo behavior are true, the robot has some probability,  $\text{PIgnore}_{\text{Det}}$ , of ignoring the detected puck,  $\text{Puck}_{\text{Det}}$ , and executing a random walk. This probabilistic activation condition can be setup to pick up one puck type more frequently than another puck type, resulting in more effective sequential foraging. For example, if  $\text{PIgnore}_{\text{Red}}$  is less than  $\text{PIgnore}_{\text{Green}}$ , then assuming  $\text{Puck}_{\text{Red}}$  and  $\text{Puck}_{\text{Green}}$  are encountered uniformly during foraging,  $\text{Puck}_{\text{Red}}$  will be foraged proportionally faster than  $\text{Puck}_{\text{Green}}$ .

The second probabilistic activation condition is

Obstacle Detected	Puck <sub>Det</sub> Detected	Grasping Puck	Gripper Break-Beam On	Inside Home Region	Timer <sub>Robot</sub> Value	Active Behavior
0	1	0	0	X	$\geq$ Alarm <sub>Det</sub>	Visual Servo
0	1	0	0	X	$<$ Alarm <sub>Det</sub>	Random Walk
0	X	0	1	X	X	Grasp Puck
0	X	1	1	0	X	Homing
0	X	1	1	1	X	Drop Puck
1	X	X	X	X	X	Collision Avoidance
0	0	0	0	X	X	Random Walk

Table 3: Behavior Network for Timer-Based Foraging

Obstacle Detected	Puck <sub>Det</sub> Detected	Grasping Puck	Gripper Break-Beam On	Inside Home Region	Ignore	Drop	Active Behavior
0	1	0	0	X	$>$ PIgnore <sub>Det</sub>	X	Visual Servo
0	1	0	0	X	$\leq$ PIgnore <sub>Det</sub>	X	Random Walk
0	X	0	1	X	X	X	Grasp Puck
0	X	1	1	0	X	$>$ PDrop <sub>Det</sub>	Homing
0	X	1	1	0	X	$\leq$ PDrop <sub>Det</sub>	Drop Puck
0	X	1	1	1	X	X	Drop Puck
1	X	X	X	X	X	X	Collision Avoidance
0	0	0	0	X	X	X	Random Walk

Table 5: Behavior Network for Probabilistic Foraging

whether a grasped puck should be dropped before reaching the home region or whether the grasped puck should continue to be transported toward the home region. Each robot has an assigned probability of dropping a grasped puck of each type while not in the home region. For the three puck types, these probabilities are:  $PDrop_{Red}$ ,  $PDrop_{Green}$ , and  $PDrop_{Blue}$ , respectively.

Every time-step during which the activation conditions for the Homing behavior are true, the robot has some probability,  $PDrop_{Det}$ , of dropping the grasped puck, Puck<sub>Det</sub>, while not in the home region. This probabilistic activation condition can be setup to transport one puck type to the home region more reliably than another puck type, resulting in more effective sequential foraging. For example, if  $PDrop_{Red}$  is less than  $PDrop_{Green}$ , then assuming Puck<sub>Red</sub> and Puck<sub>Green</sub> are encountered uniformly during foraging, Puck<sub>Red</sub> will be foraged proportionally faster than Puck<sub>Green</sub>.

The second probabilistic activation condition, dropping a grasped puck before reaching the home region, is effective in breaking up clusters of pucks. For example, in cases where there is a Puck<sub>Red</sub> surrounded by a ring of Puck<sub>Green</sub> and Puck<sub>Blue</sub>, the Puck<sub>Red</sub> can be separated by picking up the surrounding pucks and dropping them elsewhere, essentially moving them out of the way. In the Timer-Based Foraging algorithm, this dispersing of clusters is not likely as eventually the robots' Timers will

cause them to move on to another puck type, thereby foraging the surrounding pucks before being able to detect and get at the important puck in the center of the cluster. The Standard Foraging algorithm will not disperse the pucks either. The pucks will be foraged in the order from the outside of the cluster to the inside.

The combination of these two probabilistic activation conditions used in the Probabilistic Foraging behavior network increases the effectiveness of sequential foraging. To implement the Probabilistic Foraging algorithm on the robot, we used the behavior network shown in Table 5. The activation conditions Ignore and Drop were random variables in the range [0,1], selected at every time-step.

## 6. Experimental Results

We ran the three foraging algorithms, Standard Foraging, Timer-Based Foraging, and Probabilistic Foraging, on the four experimental environments described in Section 4.3. Experimental results for each foraging algorithm are given below. The adaptability of Timer-Based Foraging and Probabilistic Foraging in varying environmental conditions is demonstrated by tuning the parameters of each algorithm to work well in Environment 1 and then applying the same algorithms, with the tuned parameters, to Environments 2-4. The parameter tun-

ing for both algorithms was time-consuming; therefore, making an algorithm that does not require constant re-tuning with varying environmental conditions desirable.

For each experimental environment and sequential foraging algorithm, the average  $Util_{Final}$ , as defined in Equation 1, is averaged over all trials. For each environment/algorithm pair, a total of five experimental trials were run.

In the Timer-Based Foraging algorithm, the  $Alarm_{Red}$ ,  $Alarm_{Green}$ , and  $Alarm_{Blue}$  values shown in Table 6 were used in all experiments.

$Alarm_{Red}$	$Alarm_{Green}$	$Alarm_{Blue}$
0	750	1500

Table 6: Timer-Based Foraging Parameters

In the Probabilistic Foraging algorithm, the  $PIgnore$  and  $PDrop$  values shown in Table 7 and Table 8, respectively, were used in all experiments.

$PIgnore_{Red}$	$PIgnore_{Green}$	$PIgnore_{Blue}$
0.0	0.065	0.12

Table 7: Probabilistic Foraging  $PIgnore$  Parameters

$PDrop_{Red}$	$PDrop_{Green}$	$PDrop_{Blue}$
0.0	0.065	0.12

Table 8: Probabilistic Foraging  $PDrop$  Parameters

For each experimental environment and sequential foraging algorithm, the average  $Util_{Final}$  over all trials is shown in Figure 2. The standard deviation of the experimental trails is shown in Figure 3.

In the trials using Environment 1, it is easily seen that the Timer-Based and Probabilistic Foraging algorithms achieved near perfect sequential foraging and greatly outperform the Standard Foraging algorithm, as should be expected.

Environments 2 and 3 investigate the adaptability along the varying puck proportion axis. In Environment 2, the relative puck proportions are changed to include a much higher proportion of  $Puck_{Red}$  and a much lower proportion of  $Puck_{Blue}$ . The parameter settings for the Timer-Based Foraging algorithm, shown in Table 6, and the Probabilistic Foraging algorithm, shown in Tables 7 and 8, are unchanged from the values used in Environment 1 experiments. As Figure 2 shows, both the Timer-Based and the Probabilistic algorithms maintain similar performance as that shown in the Environment 1 trials.

In Environment 3, the relative puck proportions are adjusted in the opposite direction: there are many fewer

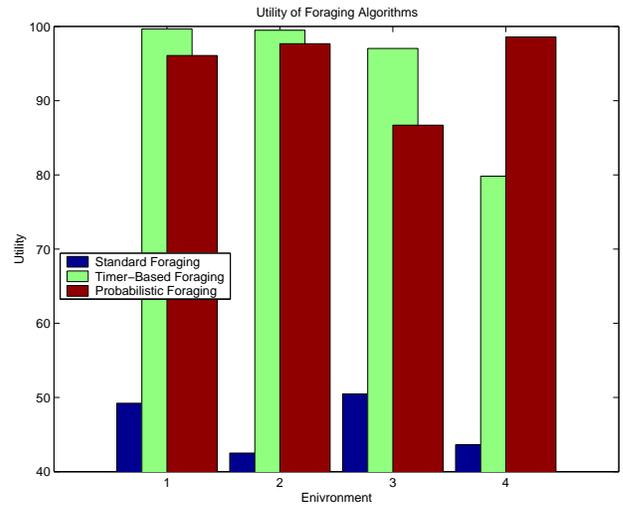


Figure 2:  $Util_{Final}$  Experimental Results

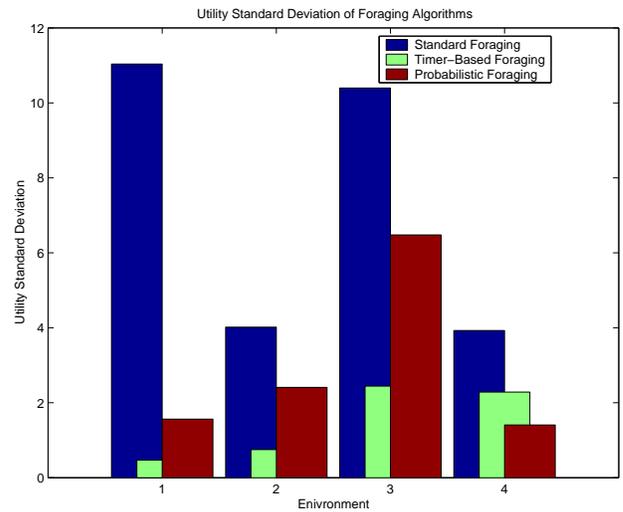


Figure 3:  $Util_{Final}$  Standard Deviation

$Puck_{Red}$ s than  $Puck_{Blue}$ s. Again, the Timer-Based Foraging algorithm maintains similar performance as that seen in Environments 1 and 2. However, the Probabilistic Foraging algorithm shows an interesting degradation in performance as compared with Environments 1 and 2. With  $Puck_{Red}$  being in such low proportion, and therefore infrequently encountered by the foraging robots, many  $Puck_{Green}$  and  $Puck_{Blue}$  were prematurely foraged. This represents an important characteristic of the Probabilistic Foraging algorithm: it does not adapt well if the proportions of pucks being collected shift heavily into the favor of pucks required to be foraged later in the task sequence over ones that should be collected sooner.

Environment 4 investigates the adaptability along the varying arena size axis. This environment has each puck type represented in even proportions as in Environment

1. In Environment 4, the performance of the Probabilistic Foraging algorithm achieves performance comparable to that seen in Environments 1 and 2. In this environment, however, the performance of the Timer-Based Foraging algorithm shows degraded performance as compared to Environments 1-3. This is intuitive since the larger the arena, the longer the foraging robots spend searching for pucks, which means there is an increase in probability that a robot’s timer alarm for the next puck type will be activated prematurely and thus that an out-of-order puck type will be collected. This environment demonstrates that the Timer-Based Foraging algorithm does not adapt well to increased arena size.

As the experimental results show, the Timer-Based Foraging algorithm adapts well along the dimension varying relative puck type proportions while the Probabilistic Foraging algorithms adapts well along the dimension of varying arena size. These properties could be used as guiding principles in selecting the appropriate sequential foraging algorithm for a given specific set of task properties.

## 7. Discussion

A means of improving foraging efficiency could involve each robot remembering where uncollected pucks were seen and returning to those locations. This is possible if the location of pucks is relatively stable over time, and if the robots are able to localize and store locations. Unfortunately, both of these conditions are typically not met in MDRS.

Remembering locations of objects, if it is possible, loses its effectiveness in highly dynamic environments, where the probability of objects being purposefully or accidentally pushed around by other robots is high. In general, in dynamic environments with large numbers of robots, remembering much about manipulable aspects of the world state, such as the location of pucks, is rarely useful. The second requirement, that of being able to localize, is a major challenge in mobile robotics, an in particular in MDRS. Although there are a number of localization techniques available (see (Borenstein et al., 1996) and (Fox et al., 1998) for reviews), most involve computation beyond what is usually embodied in MDRS.

Other methods for improving foraging efficiency involve knowledge of global world state or task state, such as the total number of objects or the number of objects remaining to be collected. Such knowledge is fundamentally global in nature, and thus not available in MDRS, where each robot only has a limited view of its immediate environment, and usually cannot communicate or if it can, it is only with local neighbors, not the whole distributed group. Thus, MDRS suffer from a rather extreme case of partial observability, and must get around it using clever means, such as using the environment to

not only sense but also store information to be used by other agents. This technique, commonly found in nature, is referred to as *stigmergy*, the process of using the environment as a means of indirect communication (Holland and Melhuish, 2000).

Stigmergy is defined as the environmental modifications resulting from one action stimulating the execution of a subsequent action (Holland and Melhuish, 2000). In the case of our Timer-Based and Probabilistic Foraging algorithms, the environment is modified by the removal of pucks through foraging. However, the removal of a puck does not directly stimulate the activation of a subsequent behavior, but does so indirectly by increasing the likelihood of a robot encountering other pucks. In the case of Timer-Based Foraging, the successful collection of all pucks of a certain type causes the  $Timer_{Robot}$  of all the foraging robots to move beyond the next puck type alarm value, resulting in the initiation of foraging the next puck type. In the case of Probabilistic Foraging, the continual removal of a certain puck type increases the likelihood of the foraging robots to encounter and eventually forage other puck types. Both the Timer-Based and the Probabilistic Foraging algorithms use a form of stigmergy, indirect communication through the environment through puck removal, to influence the future foraging activities of other robots.

## 8. Conclusions

A Minimalist Distributed Robotic System (MDRS) is a society of simple robots, each using only local sensing and control and limited capabilities in terms of intelligence, sensing, and communication. The robots in our MDRS maintain little or no state information, extract a limited amount of information from available sensors, and cannot explicitly communicate with other robots in the system.

The aim of this work is to provide a MDRS with the capability of sequential task execution. In this paper, we presented two sequential task execution algorithms, Timer-Based behavior activation and Probabilistic behavior activation, and experimentally verified them in a sequential foraging task. The two algorithms were tested on a number of experimental environments and their performance characteristics were compared.

In the sequential foraging task, the Timer-Based behavior activation method was shown to scale well with varying object type proportions but also to degrade in an increase of arena size. The Probabilistic behavior activation method was shown to scale well with an increase in arena size but had degraded performance with varying object type proportions.

Our future work includes investigating how the group size of an MDRS affects the performance of the two sequential task execution algorithms we described. Our preliminary experiments indicate that the performance

of the Timer-Based Foraging algorithm is sensitive to the number of active foraging robots. This sensitivity to the number of active foraging robots does not appear to be as prevalent in the Probabilistic Foraging algorithm.

## 9. Acknowledgments

This work is supported in part by the DARPA TASK Program, in part by the DARPA MARS Grant DABT63-99-1-0015, and in part by ONR Grant N000140110354.

## References

- Beckers, R., Holland, O., and Deneubourg, J. (1994). From local actions to global tasks: Stigmergy and collective robotics. In *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 181–189.
- Bonabeau, E., Dorigo, M., and Théraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.
- Bonabeau, E., Théraulaz, G., Arpin, E., and Sardet, E. (1994). The building behavior of lattice swarms. In Brooks, R. and Maes, P., (Eds.), *Artificial Life IV*, pages 307–312.
- Bonabeau, E., Théraulaz, G., and Deneubourg, J. (1996). Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings Royal Society of London B*, 263:1565–1569.
- Borenstein, J., Everett, B., and Feng, L. (1996). *Navigating Mobile Robots: Systems and Techniques*. A.K. Peters, Ltd.
- Cao, Y., Fukunaga, A., and Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27.
- Fox, D., Burgard, W., Thrun, S., and Cremers, A. (1998). Position estimation for mobile robots in dynamic environments. In *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*.
- Franks, N. and Sendova-Franks, A. (1992). Brood sorting by ants: Distributing the workload over the work-surface. *Behavioral Ecology and Sociobiology*, 30:109–123.
- Franks, N., Wilby, A., Silverman, B., and Tofts, C. (1992). Self-organizing nest construction in ants: Sophisticated building by blind bulldozing. *Animal Behavior*, 44:357–375.
- Gerkey, B., Vaughan, R., Stoey, K., Howard, A., Sukhatme, G., and Matarić, M. (2001). Most valuable player: A robot device server for distributed control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS-01)*, pages 1226–1231.
- Goldberg, D. and Matarić, M. J. (to appear 2002). Design and evaluation of robust behavior-based controllers. In *Robot Teams: From Diversity to Polymorphism*. AK Peters, (in press).
- Holland, O. and Melhuish, C. (2000). Stigmergy, self-organization, and sorting in collective robotics. *Artificial Life*, 5(2):173–202.
- Martinoli, A., Ijspeert, A., and Mondada, F. (1999). Understanding collective aggregation mechanisms: From probabilistic modeling to experiments with real robots. *Robotics and Autonomous Systems*, 29:51–63.
- Matarić, M. (1995a). Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:1:51–80.
- Matarić, M. J. (1995b). Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2–4):321–331.
- Théraulaz, G., E., B., and Deneubourg, J. (1998). Threshold reinforcement and the regulation of division of labour in insect societies. *Proceedings Royal Society of London B*, 265:327–335.
- Vaughan, R. (2000). Stage: A multiple robot simulator. *Institute for Robotics and Intelligent Systems Technical Report IRIS-00-393, Univ. of Southern California*.
- Werger, B. (1999). Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, 110:293–320.
- Werger, B. and Matarić, M. (1996). Robotic "food" chains: Externalization of state and program for minimal-agent foraging. *From Animals to Animats 4, Fourth International Conference on Simulation of Adaptive Behavior (SAB-96)*, pages 625–634.