Continuous Foraging and Information Gathering in a Multi-Agent Team

Somchaya Liemhetcharat, Rui Yan, Keng Peng Tee
Institute for Infocomm Research
Agency for Science, Technology and Research
Singapore 138632, Singapore
{liemhet-s, ryan, kptee}@i2r.a-star.edu.sg

ABSTRACT

We are interested in continuous foraging with multi-agent teams, where resources are replenished over time, and the goal is to maximize the *rate* of foraging. Existing algorithms for continuous foraging and area sweeping typically consider homogeneous agents. We are interested in heterogeneous teams, where agents have radically different capabilities. In particular, we consider two types of agents: a foraging agent that moves in the environment and forages resources, and a reconnaissance agent that gathers information by visiting locations and determining the number of resources available. In this paper, we consider three models of resource replenishment: a Bernoulli model and a Poisson model where resources appear probabilistically, and a stochastic Logistic model where resources increase based on the existing number of resources. We contribute three foraging algorithms that are inspired by existing algorithms, and contribute a novel algorithm for the reconnaissance agent to gather information. We extensively evaluate our algorithms in simulation, showing that our foraging algorithms outperform the existing algorithms. We demonstrate the efficacy of our information gathering algorithm in improving the overall team performance, even without communication between the foraging agents, and with noisy observations.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

Multi-agent; foraging; information gathering; reconnaissance

1. INTRODUCTION

We are interested in continuous foraging, where agents visit locations in the environment to forage resources and deliver them to a home location. The resources replenish over time, and we consider three models of resource replenishment. The first two are the Bernoulli and Poisson models where resources replenish probabilistically, which is suitable for scenarios with independently-occurring resources, such as mail. The second model is the stochastic Logistic model where the rate of resource replenishment depends on the number of existing resources. The Logistic model is suitable when the resources are populations of living things, such as fish in the ocean. Thus, the continuous foraging problem is general and applicable to many real-life scenarios.

In this paper, we formally define the continuous foraging problem, where the goal is to maximize the *rate* of foraging resources from the environment. The multi-agent team comprises foraging agents that forage items back to a home location, and a reconnaissance agent that gathers and shares information with the team. A motivating example of this problem is in fishing, where various fishing boats can be sent to pre-determined fishing locations, and the goal is to maximize the rate of fish caught. The reconnaissance agent would then be an unmanned aerial vehicle that is able to detect the number of fish, but is unable to forage them.

Existing algorithms, which we detail in the related work section, typically consider homogeneous agents that each carry a single item. We are interested in agents that carry multiple items, and in teams with a reconnaissance agent that cannot forage items, but gathers information. The addition of the reconnaissance agent makes the continuous foraging problem more interesting: how will the reconnaissance agent increase the team's foraging rate?

We consider the scenario where the foraging agents have individual models of the resources available at locations, and can only communicate within a limited distance and with limited bandwidth, namely to communicate their destination and payload, and not to share their models. The reconnaissance agent broadcasts its observations to all foraging agents. We believe such a scenario is realistic since it is difficult for the foraging agents to remain in constant communications contact with one another.

We contribute three foraging algorithms that solve the continuous foraging problem: one for the Bernoulli and Poisson distributions, and two for the Logistic distribution. In particular, the algorithms are amenable to information gathered by the reconnaissance agent, which will improve the agents' models. We contribute an information gathering algorithm that uses the expectation of resource replenishment, in order to determine the best locations for reconnaissance. The information gathering algorithm is general and applicable to any foraging algorithm.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

We evaluate our algorithms in simulation, and benchmark the performance against existing algorithms in sustainable foraging [17] and continuous area sweeping [1].We show that our algorithms outperform the benchmarks in the Bernoulli, Poisson, and Logistic scenarios when no reconnaissance agent is present, and demonstrate that the addition of the reconnaissance agent further improves the team foraging rate. Further, we consider cases where the foraging agents are unable to communicate at all, and when the observations of the number of resources available are noisy, and demonstrate the resilience of our algorithms.

The layout of the paper is as follows: in Section 2, we discuss related work in foraging and the novel aspects of this paper. In Section 3, we formally define the problem, and give an overview of our approach. Section 4 presents the three models of resources we consider, and Sections 5 and 6 present our algorithms for the foraging and reconnaissance agents. Section 7 presents our experiments and results, and we conclude in Section 8.

2. RELATED WORK

Foraging has been considered as a multi-robot task allocation (MRTA) problem [4]. Some approaches consider decentralized algorithms for the robots so as to ensure scalability (e.g., [8, 16]). Other approaches use market-based techniques to determine how the foraging tasks are assigned to the robots [18]. Synergy graphs have been considered for modeling team performance in collaborative multi-agent teams [9, 13], in areas such as role assignment [10], and configuring modules for multi-robot teams [12, 11]. In this paper, we consider the case where a relatively small number of robots are used for foraging, where each robot has a large capacity, as we believe that such a scenario better reflects real-life scenarios. Hence, we contribute decentralized algorithms for such agents.

Foraging algorithms have also been inspired from biology, such as ant-colony optimization [3]. Ant-based algorithms typically use pheromones that are left in the world, in order to guide the other agents' paths, e.g., [15, 14], or local communication to emulate pheromones [6]. Non-pheromone based algorithms have also been considered, e.g., [7, 2], where techniques used by bees are translated into multi-agent behaviors. We are interested in the selection of which location each agent should forage, and not on the search for resources or the optimization of the path to the locations, which ant and bee algorithms excel in. In contrast, ant and bee algorithms use simple heuristics such as shortest path to determine which locations the agents should forage.

There is also interest in sustainable foraging, i.e., foraging items from locations so as not to completely deplete the resources but allow the resources to replenish [17]. We are interested in such scenarios as well, but our main focus is on continuous foraging, i.e., maximizing the rate of resource foraging. However, there are many similarities between sustainable and continuous foraging, and thus we compare our algorithms with that of [17]. In addition, continuous area sweeping (e.g., [1]) has similarities to continuous foraging, with the main difference being that the agents have a carrying capacity and must periodically return to the home location. We believe the carrying capacity is applicable to real-world scenarios (e.g., the inverse of energy on the robots), and also compare to [1].

3. PROBLEM AND APPROACH

In this section, we formally present the continuous foraging problem, and give an overview of our approach. We are interested in a multi-agent team comprising foraging and reconnaissance agents, that maximizes the rate of foraging, through the use of the reconnaissance agent by gathering and sharing information. To aid in the explanation of the problem, we use a motivating scenario.

3.1 Motivating Scenario

Suppose that there is a fleet of fishing boats that capture fish from the ocean and deliver them to a "home" location, e.g., the harbor. There are pre-determined locations where fish may be found (e.g., locations determined through satellite imagery), and the fishing boats model the number of fish are present and they change over time.

The fishing boats are unable to determine the number of fish at a location unless they are physically there, due to the limited range of their on-board sensors, which may also be noisy. After a fishing boat arrives at a location, it captures as many fish as possible. The fishing boats are only capable of communication with limited range and bandwidth, and thus can only coordinate their destinations and payloads to one another, and cannot share their models and observations.

An unmanned aerial vehicle (UAV) is launched periodically from the home location. Due to the UAV's limited power, it is only able to visit a subset of the locations. When the UAV arrives at a location, it uses its sensors to obtain a (possibly noisy) measurement of the number of fish, and broadcasts that information to all fishing boats.

3.2 Formal Problem Definition

Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be the foraging agents, e.g., the fishing boats. Each agent a_i has an associated speed s_i, maximum capacity c_i, and payload $y_i \leq c_i$, i.e., the number of resources a_i is currently carrying. Let R be the reconnaissance agent, e.g., the UAV, that performs information gathering.

Let $\mathcal{L} = \{l_0, \ldots, l_m\}$ be the set of locations, where l_0 is the agents' home location, e.g., $l_1, \ldots, l_m$ are the fishing locations. Let $v_{j,t}$ be the number of resources available at location l_j at timestep t, e.g., the number of fish at l_j.

The number of resources at each location changes over time, and we assume that the change is Markovian, i.e., the number of resources $v_{j,t}$ at a location l_j at timestep t depends only on $v_{j,t-1}$. Let $\hat{v}_{j,t}^{(i)}$ be a_i's estimate of $v_{j,t}$.

When a foraging agent a_i arrives at a location l_j ($j > 0$), $\max(v_{j,t}, c_i - y_i)$ resources at l_j (i.e., all resources at l_j subject to the remaining capacity of a_i) are transferred to a_i, and a_i makes an observation $\hat{v}_{j,t}^{(i)}$ of the number of resources remaining. When a_i arrives at l_0, all y_i resources carried by a_i are transferred to l_0.

Let $D : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}^+$ be the distance function of the locations. Thus, an agent a_i takes $\left\lceil \frac{D(l_j, l_k)}{s_i} \right\rceil$ timesteps to move from location l_j to location l_k. For notational simplicity, we denote $t(a_i, l_j, l_k) = \left\lceil \frac{D(l_j, l_k)}{s_i} \right\rceil$.

Every timestep, the reconnaissance agent R observes the number of resources at $M \leq m$ locations. In our motivating scenario, this corresponds to the UAV being launched from l_0, instantaneously visiting M locations, observing the number of resources, and sharing the information with all $a_i \in \mathcal{A}$. The instantaneous nature of the reconnaissance

agent's movement in our problem is motivated by the relative speeds of a UAV and a fishing boat, and the limited $M \leq m$ is motivated by the lower battery capacity of a UAV.

The goal is to maximize the rate of resources foraged to l_0 after T timesteps, i.e., maximize $\frac{v_{0,T}}{T}$.

3.3 Our Approach

Our approach to solving the continuous foraging problem is:

- We assume that $v_{j,t}$ follows a known model — in this paper, we assume that $v_{j,t}$ follows the Bernoulli model (Section 4.1), the Poisson model (Section 4.2), or the stochastic Logistic model (Section 4.3). The parameters of these models may or may not be known *a priori*;
- The estimates $\hat{v}_{j,t}^{(i)}$ are updated using the models, observations from foraging agents a_i visiting locations l_j, and observations by the reconnaissance agent R;
- The foraging agents do not share their models $\hat{v}_{j,t}^{(i)}$, but they share their destinations with one another, subject to a maximum communication distance C, i.e., agents communicate only if they are within distance C of each other.
- We contribute algorithms that control the foraging agents a_i, that use $\hat{v}_{j,t}^{(i)}$ and other agents' destinations and payloads, to plan their next destination (Section 5.2);
- We contribute observation algorithms that computes M locations that are observed by the reconnaissance agent (Section 6).

4. RESOURCE MODELS

In this section, we discuss three models of resource replenishment at the locations $l_j \in \mathcal{L}$. The first is a Bernoulli model, where resources have a static, independent probability of being generated every timestep. The second is a Poisson model, where the number of resources generated every timestep follows a mean value. The third is a stochastic Logistic model, where the rate of increase of resources depends on the number of existing resources.

4.1 Bernoulli Model of Resources

In the Bernoulli model, every location $l_j \in \mathcal{L}$ is associated with a probability $p_j \in (0, 1]$. Every timestep, there is a p_j probability that a new resource is generated at l_j:

$$v_{j,t} = \begin{cases} v_{j,t-1} + 1 & \text{with } p_j \text{ probability} \\ v_{j,t-1} & \text{otherwise} \end{cases} \quad (1)$$

The Bernoulli model was chosen for a number of reasons:

- It is intuitive and easily-understood, and corresponds to real-life scenarios such as mail entering a mailbox;
- New resources are generated independently;
- $v_{j,t}$ is non-deterministic even if p_j is known.

However, the Bernoulli model has a drawback — there is no upper limit to the number of items at a location. Thus, if a location l_j is not visited for a long time, there is a high probability that the number of items exceeds an agent's capacity.

One method to circumvent the drawback is to impose an maximum M_j for each location l_j. However, we believe that such a limit would be artificial, and instead considered the Logistic model that we describe later.

4.2 Poisson Model of Resources

The Poisson model is similar to Bernoulli, except that the resource generation follows a Poisson distribution:

$$v_{j,t} = v_{j,t-1} + \alpha_j, \text{ where } \alpha_j \sim \text{Pois}(\lambda_j) \quad (2)$$

Also, the Poisson model allows more than one resource to be generated per timestep, and we are interested to investigate the impact of this property on the multi-agent team performance. For example, suppose that an agent a_i last visited a location l_j at time t and foraged all its resources, and will next visit l_j at time t'. In the Bernoulli model, there is at most $t' - t$ resources at l_j, but in the Poisson model this may not be true.

4.3 Stochastic Logistic Model of Resources

The Logistic model has been widely used in multi-robot systems to study the optimization of communication [19] and sustainable foraging [17]. In order to consider the density-dependent population growth and environmental stochasticities, the resource is generated from the following stochastic Logistic model [5]:

$$\frac{dv_{j,t}}{dt} = rv_{j,t}(1 - \frac{v_{j,t}}{K}) + \sigma_e v_{j,t} \circ dW_e(t) \quad (3)$$

where r is the unconstrained population growth rate, K is the maximum population, σ_e is the intensity of the growth rate fluctuation, and $dW_e(t)$ is white noise with mean zero and randomly changing sign within any short time interval, i.e. $dW_e(t)$ is delta-autocorrelated. The process with increments $dW_e(t)$ representing the noise is the Brownian motion. Furthermore, "$\circ$" denotes "Stratonovich calculus".

Now rewrite (3) in the integral form:

$$v_{j,t} = v_{j,0} + \int_0^t (rv_{j,s}(1 - \frac{v_{j,s}}{K}))ds + \int_0^t \sigma_e v_{j,s} \circ dW_e(s) \quad (4)$$

and consider the following Riemann-Stieltjes integral

$$\int_0^t \sigma_e v_{j,s} dW_e(s) = \lim_{n\to\infty} \sum_{q=1}^{n} \sigma_e v_{j,\tau_q}(W_e(s_{q+1}) - W_e(s_q))$$

where $\tau_q \in [s_q, s_{q+1}]$.

Since $W_e(\cdot)$ is not smooth in the above model, the limit will depend on τ_q's value. $\tau_q = s_q$ leads to "Ito calculus" denoted by $\int_0^t \sigma_e v_{j,s} \cdot dW_e(s)$ and $\tau_q = (s_q + s_{q+1})/2$ leads to "Stratonovich calculus" denoted by $\int_0^t \sigma_e v_{j,s} \circ dW_e(s)$. Furthermore the relationship between "Stratonovich calculus" and "Ito calculus" is as follows:

$$\int_0^t \sigma_e v_{j,s} \circ dW_e(s) = \int_0^t \frac{1}{2}\sigma_e^2 v_{j,s} ds + \int_0^t \sigma_e v_{j,s} \cdot dW_e(s)$$

Therefore, Equation (4) can be rewritten as:

$$v_{j,t} = v_{j,0} + \int_0^t \left(rv_{j,s}(1 - \frac{v_{j,s}}{K}) + \frac{1}{2}\sigma_e^2 v_{j,s} \right) ds + \int_0^t \sigma_e v_{j,s} \cdot dW_e(s)$$

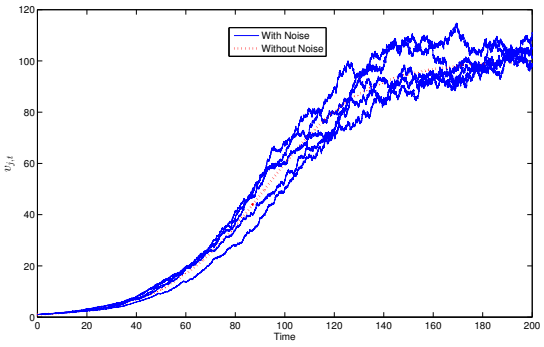

Figure 1: Resources following the stochastic Logistic model.

By Euler's method, we obtain a discretized approximation $\bar{v}_{j,t}$ of the solution $v_{j,t}$ between t and $t + \Delta t$ by

$$\bar{v}_{j,t+\Delta t} = \bar{v}_{j,t} + \left(r\bar{v}_{j,t}(1 - \frac{\bar{v}_{j,t}}{K}) + \frac{1}{2}\sigma_e^2 \bar{v}_{j,t} \right) \Delta t + \sigma_e \bar{v}_{j,t}(W_e(t + \Delta t) - W_e(t)) \quad (5)$$

The distribution $W_e(t+\Delta t) - W_e(t)$ can be simulated by generating a standard Gaussian distribution $\mathcal{N}(0, 1)$ multiplied by $\sqrt{\Delta t}$. For example, we chose the stochastic environmental noise $\sigma_e = 0.02$ and the other parameters $r = 0.04$, $K = 100$ and $v_{j,0} = 1$. Note that the increment $W_e(t+\Delta t) - W_e(t)$ can be generated as new independent draws from a Gaussian distribution at each iteration due to the independent increment of Brownian motion. Here we used Monte Carlo simulation, with $N = 100$ scenarios. Fig. 1 shows several iterations of the approximated resource evolution $v_{j,t}$. The red dotted curve shows the resource evolution without noise.

5. FORAGING ALGORITHMS

The multi-agent team $\mathcal{A}$ consists of agents that visit the locations $l_j \in \mathcal{L}, j > 0$, foraging the resources available at these locations, and deliver them to the home location $l_0 \in \mathcal{L}$.

In this section, we first describe how the foraging agents maintain individual models of the number of resources at each location. We next describe our three distributed foraging algorithms.

5.1 Maintaining a Model of Resources

Each agent maintains its own model of the resources available at every location. Let $\hat{v}_{j,t}^{(i)}$ be agent a_i's model of the number of resources at location j at timestep t.

We assume that the type of resource model is known to the multi-agent team, i.e., whether the resources follow the Bernoulli, Poisson, or stochastic Logistc model. However, the parameters of the models are not known *a priori*:

1. Bernoulli model: p_j is not known; the agents use a preset estimate $\hat{p}_j$;
2. Poisson model: λ_j is not known; the agents use a preset estimate $\hat{\lambda}_j$;
3. Stochastic Logistic model: The unconstrained population growth rate r and maximum population K are known, but the intensity of growth rate fluctuation σ_e is unknown; the agents use a preset estimate $\hat{\sigma}_e = 0$.

In each agent's resource model, the estimate $\hat{v}_{j,t}^{(i)}$ is updated every timestep using the equations from Section 4, but with their estimates of the model parameters. Whenever a foraging agent makes an observation o_j of the number of resources at a location j (i.e., when the agent visits the location), the agent resets its estimate so that $\hat{v}_{j,t}^{(i)} = o_j$. Similarly, when the reconnaissance agent visits a location and makes an observation o_j, it broadcasts o_j to all the foraging agents, and they update their estimates so that $\hat{v}_{j,t}^{(i)} = o_j$.

All the agents in the multi-agent team assume that $\hat{v}_{j,0}^{(i)} = 0$ for the Bernoulli and Poisson models (i.e., there are no resources at any location at time 0), and that $\hat{v}_{j,0}^{(i)} = \frac{K}{2}$ for the stochastic Logistic model (i.e., the population size is half the maximum at time 0).

5.2 Algorithms for Foraging

We now describe our three distributed foraging algorithms.

5.2.1 *Greedy Rate*

We contribute the Greedy Rate algorithm, that actively replans the agent a_i's destination based on the estimates $\hat{v}_{j,t}^{(i)}$. The Greedy Rate algorithm is inspired by algorithms proposed for continuous area sweeping [1]. The main difference is that instead of selecting the location with the highest expected number of resources, our Greedy Rate algorithm selects the location with the highest expected *rate* of resource foraging. Further, our Greedy Rate algorithm ignores resources that have been "earmarked" by another robot (described below). Algorithm 1 shows the pesudo-code of the Greedy Rate algorithm. The seemingly small differences between our Greedy Rate algorithm and the continuous area sweeping algorithm [1] has a large effect, as we show in the results section.

Algorithm 1 Compute the next destination of agent a_i that is currently at location l_α

GreedyRate(a_i, l_α)

```
1: if c_i = y_i then
2:    return l_0
3: end if
4: // Compute the rate if a_i heads home
5: (r_best, l_best) ← (y_i / t(a_i, l_α, l_0), l_0)
6: // Compute the rate if a_i visits l_j then heads home
7: for all l_j ∈ L s.t. j > 0 do
8:    e_j ← Σ_{a_k ∈ A heading to l_j} (c_k − y_k)
9:    y'_i ← max(c_i, y_i + max(0, v̂^(i)_{j,t+t(a_i,l_α,l_j)} − e_j))
10:   r' ← y'_i / (t(a_i, l_α, l_j) + t(a_i, l_j, l_0))
11:   if r' > r_best then
12:      (r_best, l_best) ← (r', l_j)
13:   end if
14: end for
15: return l_best
```

Line 4 of Algo. 1 computes the rate if a_i heads to l_0 with its current payload, and lines 7–9 compute the rate if a_i visits a new location l_j, then heads to l_0. Line 7 first computes the number of resources "earmarked" by other foraging agents

heading to l_j (if communication is available and the agents were in communication range). Line 8 uses the estimate $\hat{v}^{(i)}_{j,t+t(a_i,l_\alpha,l_j)}$, which is the expected number of items at l_j after $t(a_i, l_\alpha, l_j)$ timesteps from the current timestep t.

We use "earmarking" (instead of more complex coordination strategies) to minimize communication among the foraging agents. We believe that foraging agents tend to have limited computational power and communication bandwidth, and so we sought to minimize the computational and communication requirements. Further, we believe that if communication among foraging agents was impossible, it would still be feasible for a foraging agent to infer another agent's destination by observing its direction of travel.

Thus, the Greedy Rate algorithm seeks to improve the marginal rate of foraging by considering possible locations, and using the estimates $\hat{v}^{(i)}_{j,t}$. Since the estimates $\hat{v}^{(i)}_{j,t}$ are used, the performance of the Greedy Rate algorithm may be improved when further observations of $v_{j,t}$ are made by the reconnaissance agent. The Greedy Rate algorithm has a computational complexity of $\mathcal{O}(Tn|\mathcal{L}|)$ (T total timesteps, and $\mathcal{O}(n|\mathcal{L}|)$ operations per timestep).

5.2.2 Adaptive Sleep

Our Adaptive Sleep algorithm is adapted from an algorithm from sustainable foraging [17]. Algorithm 2 shows the psuedo-code of our Adaptive Sleep algorithm. Each agent a_i chooses a location l_α, and the agent forages from the location when it has $\frac{K_\alpha}{2}$ resources, where K_α is the maximum number of resources at l_α. Whenever the agent is at the home location l_0, it sleeps (stays idle) until $\hat{v}^{(i)}_{\alpha,t+t(a_i,l_0,l_\alpha)}$, a_i's estimated number of resources at l_α (considering the travel time $t(a_i, l_0, l_\alpha)$), is at least $\frac{K_\alpha}{2}$, since the rate of increase of resources (assuming the Logistic model) is maximum when there are $\frac{K_\alpha}{2}$ resources.

l_α is chosen randomly from $\mathcal{L}$, with the caveat that foraging agents that are initially within communication range coordinate so that there are no conflicts; agents outside communication range can pick the same location.

Algorithm 2 Compute if agent a_i that is assigned to location l_α should sleep further

AdaptiveSleep(a_i, l_α)

```
1: if a_i is not at l_0 then
2:     return false
3: end if
4: if v̂_{α,t+t(a_i,l_0,l_α)} < K_α/2 then
5:     return true
6: else
7:     return false
8: end if
```

One key difference in our Adaptive Sleep algorithm is that the agent sleeps until the estimated number of resources is at least $\frac{K_\alpha}{2}$. In [17], the agent computes an initial amount of time to sleep before it forages resources, and the sleep time is adjusted when it forages and observes the number of resources, and is not able to take advantage of observations by the reconnaissance agent.

Another difference is that the foraging agents coordinate on the locations, subject to communication range limits, while the sustainable foraging algorithm [17] uses a centralized static allocation based on the distance to the locations.

Also, the sustainable foraging algorithm [17] is deeply concerned with over-foraging and causing locations to collapse (not able to produce any more resources), we are mainly focused on maximizing the overall rate of foraging, as this makes our problem more general beyond sustainable foraging. However, to prevent over-foraging, we ensured that our Adaptive Sleep algorithm always leaves a minimum population at each location (e.g., if 5 resources are available and a minimum of 2 are required for resource generation, then only 3 resources are foraged).

5.2.3 Adaptive Sleep with Target Change

The Adaptive Sleep with Target Change algorithm is a slightly modified version of the Adaptive Sleep algorithm. While "sleeping" at the home location, an agent a_i may visit another location l_β that was not assigned to any foraging agent (to a_i's best knowledge). a_i selects l_β such that the travel time is within the expected amount of time to sleep, and maximizes the number of resources collected. However, due to the limited communication range, it is possible that another agent selected l_β as its primary location.

This modification of the Adaptive Sleep algorithm allows the foraging agents to potentially increase the overall rate of foraging, since locations that were previously ignored by Adaptive Sleep may now be foraged (if $m > n$). However, if the communication range is small (or the agents cannot communicate), then agents may visit one another's assigned locations and affect the overall foraging rate.

6. INFORMATION-GATHERING TO IMPROVE FORAGING PERFORMANCE

In this section, we describe our algorithm that determines the $M \leq m$ locations in $\mathcal{L}$ the reconnaissance agent should observe and broadcast to the foraging agents.

We contribute the Expected Observation algorithm that runs m simulations, one for each location $l_j \in \mathcal{L}$. The expected number of resources at each location is used for the simulations. The difference in the m simulations is that different information is given to the foraging agents, e.g., in the first iteration, it is assumed that l_1 is observed, so all foraging agents a_i receive an observation $\hat{v}^{(i)}_{1,t}$. Note that the observations received by the foraging agents may not be equal, e.g., $\hat{v}^{(i)}_{1,t} \neq \hat{v}^{(i')}_{1,t}$, and this reflects both the error in observation by the reconnaissance agent, and communication errors between reconnaissance and foraging agents.

The m simulations are run for a constant T_{forward} timesteps, and the number of resources at the home location l_0 is noted. The algorithm then returns the M locations whose simulations correspond to the highest resources at l_0.

It is interesting that the m simulations start off identically, but the difference in information provided to the foraging agents give different results. For example, if l_1 is observed by the reconnaissance agent, an agent that would have headed to l_1 might decide to go to l_2 instead, and such a change in plans has repercussions further down the road.

Our Expected Observation algorithm assumes that the foraging agents' algorithms are known to the reconnaissance agent. We believe this assumption is reasonable, since the foraging and reconnaissance agents are in the same collaborative team. While our Expected Observation algorithm may seem intuitive and straight-forward, our contribution lies not only in the Expected Observation algorithm, but

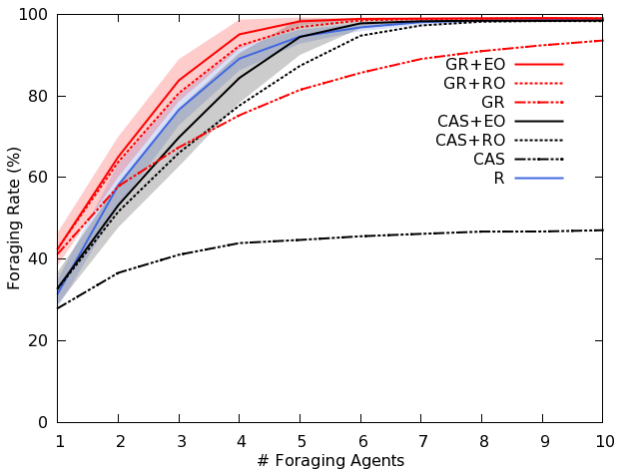

Figure 2: Comparison of our Greedy Rate (GR) algorithm against the benchmark of Continuous Area Sweeping (CAS) when resources follow the Bernoulli model of replenishment.

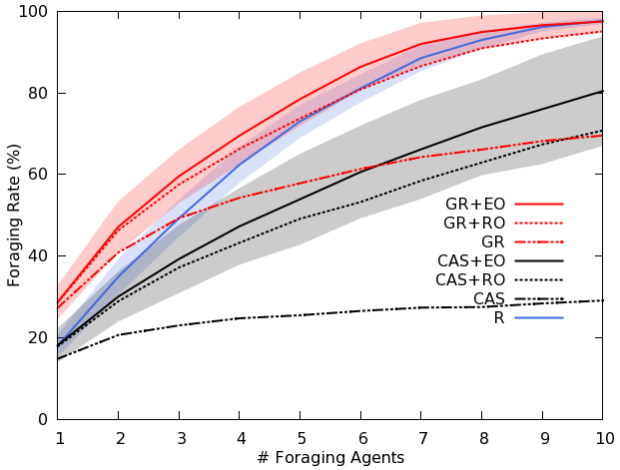

Figure 3: Comparison of our Greedy Rate (GR) algorithm against the benchmark of Continuous Area Sweeping (CAS) when resources follow the Poisson model of replenishment.

in the combination of both the foraging *and* information-gathering algorithms. Previous foraging algorithms generally do not consider incorporating new information from other sources (e.g., the reconnaissance agent), while our foraging algorithms exploit the fact the new information can arrive at any time, and thus improves the overall team foraging rate, as we describe in the next section.

7. EXPERIMENTS AND RESULTS

We describe the extensive experiments we conducted to analyze the performance of our algorithms we introduced in the previous sections. We compare against the baselines from sustainable foraging and continuous area sweeping, and evaluate our Expected Observation algorithm.

7.1 Experimental Setup

The foraging locations were randomly generated, following a uniform distribution over a square of size $N \times N$, and

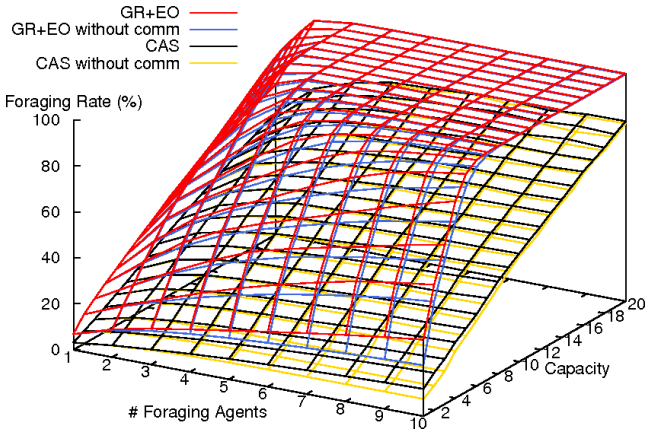

Figure 4: Effect of communication among foraging agents when resources follow the Bernoulli model of replenishment.

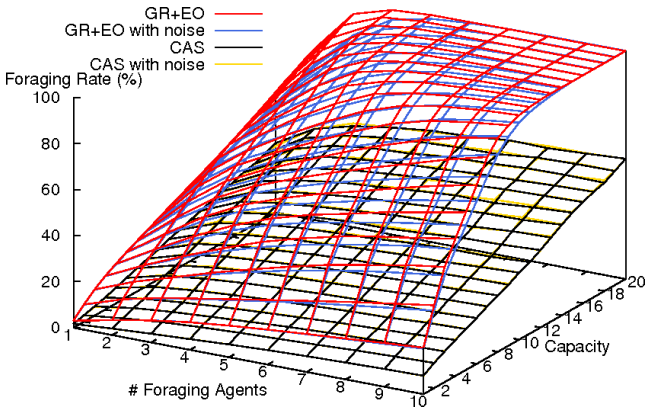

Figure 5: Effect of observational noise when resources follow the Poisson model of replenishment.

either followed the Bernoulli, Poisson, or the stochastic Logistic models. The agents' initial positions were randomly generated to be also be within the $N \times N$ square. In each experiment, we simulated $T = 1000$ timesteps, and recorded the number of resources $v_{0,T}$ foraged to the home l_0.

We varied the number of agents n from 1 to 10, and the capacities from 1 to 20. We chose 10 and 20 because it was sufficient for a Random foraging algorithm (i.e., agents that randomly select their destinations) to forage almost all resources in the Bernoulli and Poisson scenarios. We set the number of locations $|\mathcal{L}| = 20$ (since it is twice the number of agents) and set the number of location that the reconnaissance agent could visit at each timestep to be $M = \frac{|\mathcal{L}|}{4}$.

As a baseline, we assumed that foraging agents were capable of limited communication when they were within $\frac{N}{10}$ distance, and could communicate their destinations and payloads. We assumed that observations are not noisy as a baseline. We investigate the effects of no communication and noisy observations in the Bernoulli and Poisson models.

7.2 Experiments with Bernoulli and Poisson Models of Replenishment

We compared our Greedy Rate (GR) algorithm against

the Continuous Area Sweeping (CAS) algorithm [1]. As a baseline, we used a Random foraging (R) algorithm where agents randomly select their destination.

Both our GR algorithm and the CAS algorithm can use information gathered by the reconnaissance agent, and we compared our Expected Observation (EO) algorithm to a Random Observation (RO) algorithm, where M locations would be randomly chosen by the reconnaissance agent.

Figures 2 and 3 show the performance of our algorithms when the capacities of the agents are 10. Since the number of resources generated in simulation varied, we measured the percentage of resources that were successfully foraged at the end of the experiment. The solid red, black and blue lines show Greedy Rate with Expected Observation (GR+EO), Continuous Area Sweeping with Expected Observation (CAS+EO), and Random Foraging (R), and the shaded areas show the standard deviations of these algorithms. The dotted and dashed lines show other combinations of foraging and reconnaissance algorithms.

As the number of agents increase, R outperforms both GR and CAS ($p = 1 \times 10^{-24}$ and $p = 3 \times 10^{-27}$ with a 2-tailed T-test on R vs GR in Bernoulli and Poisson respectively), primarily because the agents do not share their models, so agents tend to head to similar locations. Even though agents coordinate when possible, the limited range of communication causes inefficiencies in foraging.

However, the introduction of a reconnaissance agent substantially improves both GR and CAS. Our EO algorithm outperforms the RO algorithm ($p = 1 \times 10^{-15}$ and $p = 2 \times 10^{-39}$ for GR+EO vs GR+RO on Bernoulli and Poisson respectively), and GR+EO outperforms CAS+EO and R ($p = 3 \times 10^{-26}$ and $p = 1 \times 10^{-29}$ for Bernoulli, and $p = 5 \times 10^{-36}$ and $p = 1 \times 10^{-31}$ for Poisson). It is interesting to note that CAS+EO performs substantially better than the baseline CAS. In general, adding a single reconnaissance agent with EO provides a much higher benefit than increasing the number of foraging agents.

We investigated having no communication (among the foraging agents) and noisy observations. Figures 4 and 5 shows the effects as the number of agents and their capacities vary. While a lack of communication and noisy observations affect our algorithms, the effect is minimal (a median of 0.3% and 2.2% respectively for communication and noise, thus illustrating that our algorithms are robust to a lack of communication and noisy observations.

In addition, Figures 4 and 5 clearly illustrate the efficacy of our GR+EO algorithms over the baseline CAS, across all numbers of foraging agents and agent capacities. We chose 10 to be the maximum number of foraging agents, and 20 to be the maximum capacity, because our algorithm have already hit the 100% foraging rate before that point. In contrast, CAS does not reach 100% even with 10 foraging agents with a capacity of 20 each.

7.3 Experiments with Stochastic Logistic Model

We compared our Adaptive Sleep (AS), Adaptive Sleep with Target Change (ASTC) algorithms against Sustainable Foraging (SF) [17], and a Random (R) foraging algorithm as baseline. Only our algorithms could use information gathered by the reconnaissance agent.

Fig. 6 shows the algorithms' performance when the agent capacities are 20, and the stochastic Logistic noise is $\sigma_e = 0.08$. The shaded regions show the standard deviations of

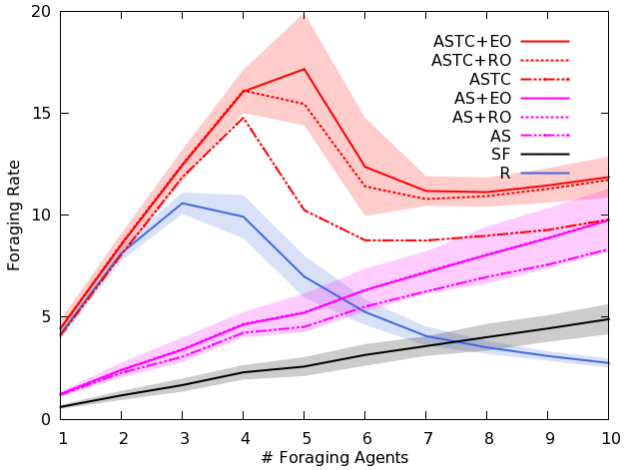

Figure 6: Comparison of our Adaptive Sleep (AS) and Adaptive Sleep with Target Change (ASTC) algorithms against the benchmark of Sustainable Foraging (SF) when resources follow the stochastic Logistic model of replenishment.

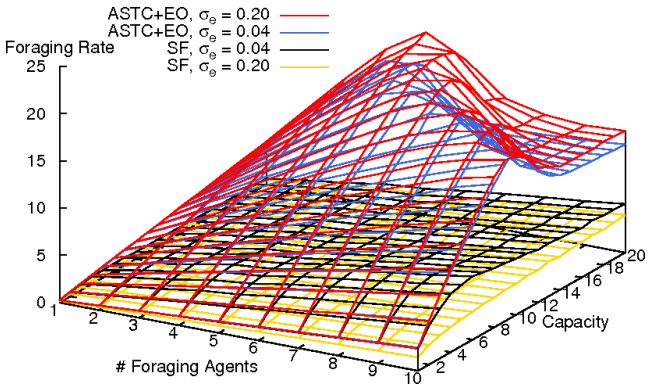

Figure 7: Effect of noise in the stochastic Logistic model.

ASTC+EO, AS+EO, SF and R. SF and AS both increase linearly, since the agents select a single destination; our AS algorithm outperforms SF ($p = 7 \times 10^{-61}$).

R outperforms AS and SF when the number of agents are small, primarily because changing destinations allows resources to replenish at a higher rate. However, as the number of agents increase, R's performance begins to plummet as locations become over-foraged and the replenishment rate lowers. ASTC combines the benefits of AS and R, allowing agents to choose a destination, and also visit other unassigned locations. Thus, the shape of the ASTC curve is similar to R, albeit at a much higher foraging rate.

The introduction of a reconnaissance agent improves the foraging rate. EO and RO perform similarly with the AS algorithm. For ASTC, EO and RO perform similarly when the number of agents $n < 5$ ($p = 0.14$), but EO outperforms RO when there are $n \geq 5$ ($p = 8 \times 10^{-6}$). The constant 5 corresponds to the number of locations the reconnaissance agent visits: EO determines the best locations to visit, compared to RO's random choice. When $n < 5$, there is a high probability that RO visits all agents' locations.

Fig. 7 shows the effect of stochastic Logistic noise ($\sigma_e = 0.04$ to 0.20). SF performs poorly as the noise increases, but ASTC+EO performs better with higher noise, showing that

our algorithm takes advantage of the noise (noise creates a probability of generating resources ahead of schedule).

7.4 Summary of Experimental Analysis

Across the Bernoulli and Poisson models of replenishment, our Greedy Rate (GR) algorithm outperforms the baseline Continuous Area Sweeping (CAS). Further, the addition of the reconnaissance agent improves the performance of both GR and CAS, since additional information is provided to both algorithms. Our Expected Observation (EO) algorithm outperforms the Random Observation (RO) algorithm, showing that although the reconnaissance agent can visit $\frac{|\mathcal{L}|}{4}$ of the locations each timestep, selecting which locations to visit still plays a very important role. Random selection (which will visit every location every 4 timesteps on average) improves the team foraging rate, but not as much.

In addition, it is important to note that our EO algorithm significantly improves the CAS algorithm's foraging rate, so our information-gathering algorithm is not specific to our foraging algorithms, but can be applied to any foraging algorithm that makes use of new information. Also, our algorithms are robust to noise in observations, and performs with minimal degradation when communication among the foraging agents are unavailable.

Similarly, for the stochastic Logistic model of replenishment, our Adaptive Sleep (AS) and Adaptive Sleep with Target Change (ASTC) algorithms outperform the baseline of Sustainable Foraging (SF), across all numbers of foraging agents and agent capacities. Our algorithm is robust to the noise in the stochastic Logistic model, and the EO algorithm improves our foraging algorithms significantly. The ASTC algorithm incorporates both the features of the AS algorithm (to maximize the foraging rate at the assigned location) and the Random algorithm (to exploit the resource replenishment at unassigned locations).

Thus, our experiments show that our algorithms are distributed and require little communication among the foraging agents, and are robust to noise in observations, a lack of communication, and noise in the models. We outperform the baselines significantly, and the addition of the reconnaissance agent improves the multi-agent team's foraging rate, even for foraging algorithms that we did not create.

8. CONCLUSION

We formally defined the continuous foraging problem, where agents visit known foraging locations to collect and deliver resources to a home location. The resources replenish over time, and we defined three models of resource replenishment: the Bernoulli and Poisson models where resources replenish probabilistically (e.g., mail entering a mailbox), and a stochastic Logistic model where the rate of resource replenishment depends on the number of existing resources (e.g., a population of fish).

We considered two types of agents: foraging agents that actively forage resources, and a reconnaissance agent that cannot forage items, but can visit a subset of the locations to determine the number of resources available.

We contributed algorithms for the foraging and reconnaissance agents, and to evaluate our algorithms, we performed experiments in simulation, benchmarking against existing algorithms in sustainable foraging and continuous area sweeping. We showed that our algorithms outperform the existing ones even without the use of the reconnaissance agent. Further, we demonstrated that the reconnaissance agent further improves the foraging rate of the multi-agent team, even in the presence of noisy observations and no communication among the foraging agents, thus illustrating the benefits of our algorithms.

Acknowledgments

This work was supported by the A*STAR Computational Resource Centre through the use of its high performance computing facilities. The authors also thank Wei Li Cheah for her help with visualization of the results.

REFERENCES

[1] M. Ahmadi and P. Stone. Continuous Area Sweeping: A Task Definition and Initial Approach. In *Proc. Int. Conf. Advanced Robotics*, pages 316–323, 2005.

[2] S. Alers, D. Bloembergen, D. Hennes, S. de Jong, M. Kaisers, N. Lemmens, K. Tuyls, and G. Weiss. Bee-inspired Foraging in an Embodied Swarm. In *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, pages 1311–1312, 2011.

[3] M. Dorigo and M. Birattari. Ant colony optimization. In *Encyclopedia Machine Learning*, pages 36–39. 2010.

[4] B. P. Gerkey and M. J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *J. Robotics Research*, 23(9):939–954, 2004.

[5] H. Hakoyama and Y. Iwasa. Extinction Risk of a Density-dependent Population Estimated from a Time Series of Population Size. *J. Theoretical Biology*, 204(3):337–359, 2000.

[6] N. Hoff, A. Sagoff, R. Wood, and R. Nagpal. Two Foraging Algorithms for Robot Swarms using only Local Communication. In *Proc. Int. Conf. Robotics and Biomimetics*, pages 123–130, 2010.

[7] N. Lemmens, S. Jong, K. Tuyls, and A. Nowe. Bee Behaviour in Multi-agent Systems. In *Adaptive Agents and Multi-Agent Systems III*, volume 4865, pages 145–156. 2008.

[8] K. Lerman, C. Jones, A. Galstyan, and M. J. Mataric. Analysis of Dynamic Task Allocation in Multi-Robot Systems. *J. Robotics Research*, 25(3):225–241, 2006.

[9] S. Liemhetcharat and M. Veloso. Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, pages 365–375, 2012.

[10] S. Liemhetcharat and M. Veloso. Weighted Synergy Graphs for Role Assignment in Ad Hoc Heterogeneous Robot Teams. In *Proc. Int. Conf. Intelligent Robots and Systems*, pages 5247–5254, 2012.

[11] S. Liemhetcharat and M. Veloso. Forming an Effective Multi-Robot Team Robust to Failures. In *Proc. Int. Conf. Intelligent Robots and Systems*, 2013.

[12] S. Liemhetcharat and M. Veloso. Synergy Graphs for Configuring Robot Team Members. In *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, pages 111–118, 2013.

[13] S. Liemhetcharat and M. Veloso. Weighted Synergy Graphs for Effective Team Formation with Heterogeneous Ad Hoc Agents. *Journal of Artificial Intelligence*, 208(2014):41–65, 2014.

[14] W. Liu, A. Winfield, J. Sa, J. Chen, and L. Dou. Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots. *Adaptive Behavior*, 15(3):289–305, 2007.

[15] L. Panait and S. Luke. A Pheromone-Based Utility Model for Collaborative Foraging. In *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, pages 36–43, 2004.

[16] D. A. Shell and M. J. Mataric. On Foraging Strategies for Large-Scale Multi-Robot Systems. In *Proc. Int. Conf. Intelligent Robots and Systems*, pages 2717–2723, 2006.

[17] Z. Song and R. Vaughan. Sustainable Robot Foraging: Adaptive Fine-Grained Multi-Robot Task Allocation for Maximum Sustainable Yield of Biological Resources. In *Proc. Int. Conf. Intelligent Robots and Systems*, pages 3309–3316, 2013.

[18] L. Vig and J. Adams. Market-based Multi-Robot Coalition Formation. In *Proc. Int. Symp. Distributed Autonomous Robotics Systems*, pages 227–236, 2006.

[19] E. Yoshida, T. Arai, J. Ota, and T. Miki. Effect of Grouping in Local Communication System of Multiple Mobile Robots. In *Proc. Int. Conf. Intelligent Robots and Systems*, pages 808–815, 1994.