Swarm Intelligence (2020) 14:233–258
https://doi.org/10.1007/s11721-020-00182-2

Response probability enhances robustness in decentralized threshold-based robotic swarms

Annie S. Wu[1] · **R. Paul Wiegand**[2] · **Ramya Pradhan**[1]

Received: 2 August 2019 / Accepted: 24 April 2020 / Published online: 6 June 2020

Abstract
In this paper, we investigate how response probability may be used to improve the robustness of reactive, threshold-based robotic swarms. In swarms where agents have differing thresholds, adding a response probability is expected to distribute task experiences among more agents, which can increase the robustness of the swarm. If the lowest threshold agents for a task become unavailable, distributing task experience among more agents increases the chance that there are other agents in the swarm with experience on the task, which reduces performance decline due to the loss of experienced agents. We begin with a mathematical analysis of such a system and show that, for a given swarm and task demand, we can estimate the response probability values that ensure team formation and meet robustness constraints. We then verify the expected behavior on an agent based model of a foraging problem. Results indicate that response probability may be used to tune the tradeoff between system performance and system robustness.

Keywords Response probability · Response threshold · Redundancy · Robustness · Threshold-based systems · Decentralized task allocation · Swarm robotics · Multi-agent systems

1 Introduction

In this paper, we investigate how response probability may be used to improve the robustness of reactive, threshold-based robotic swarms. One of the perceived advantages of swarm systems is robustness (Brambilla et al. 2013). If an agent or group of agents that are the primary actors for a task become unavailable, then the multi-agent composition

✉ Annie S. Wu
aswu@cs.ucf.edu

R. Paul Wiegand
wiegand@ist.ucf.edu

Ramya Pradhan
ramya.pradhan@knights.ucf.edu

1 University of Central Florida, Orlando, FL 32816-2362, USA

2 University of Central Florida, Orlando, FL 32826-3281, USA

of a swarm means that there are potentially other agents that can take the place of the primary actors. The availability of replacement agents provides uninterrupted task performance as long as the replacement agents have equal capability as the original actors. For tasks in which experience improves performance, however, maintaining task performance due to loss of agents is more challenging. If the replacement agents are less capable than the original agents, there can be a drop in task performance until the replacement agents gain enough experience to work as effectively as the original agents. In computational systems, experience often equates to information; thus, domains where this tradeoff is important include problems in which agents must work to gain information, for example, on the environment in which they are working or on how their team members behave, that can improve their contribution to the overall swarm performance.

From a system performance perspective, it is desirable for agents with the most experience to act on a task because that results in the most effective performance. From a robustness perspective, however, it may be beneficial for less experienced—and less effective—agents to act on a task because it would increase the pool of agents with experience, which can potentially temper the drop in system performance should the more experienced agents ever become unavailable. The question, then, is how to maintain system performance while also generating a large enough pool of agents with experience such that the loss of primary actors does not significantly reduce system performance on a task. More specifically, how can we achieve such functionality in a decentralized system.

Biological studies on social insect societies suggest that response probability is a potential mechanism for balancing the goals of maintaining system performance and building an experience pool (Weidenmüller 2004). When agents in a threshold-based system respond deterministically, only those agents with the lowest threshold for a task will act. As a result, only the primary actors for a task will gain experience on that task. If agents respond probabilistically instead of deterministically, then agents do not always act when a task stimulus exceeds their threshold. When any of the primary (lowest threshold) actors for a task do not act, agents with higher thresholds get an opportunity to gain experience on that task, thus increasing the pool of agents with experience on the task. Such non-deterministic responses in biological swarms may be internal to the agent, or may be due to reasons such as agents being occupied by other tasks, being unable to sense or accurately sense the task stimulus, or being physically blocked or otherwise unable to reach the task.

We hypothesize that such a mechanism can be applied in robotic swarms to improve system robustness. Robustness is particularly important for swarms applied to extreme problem domains (Scerri et al. 2005), such as UAV coordination, interplanetary exploration, and disaster rescue, where robust autonomy is crucial because human intervention is unavailable or difficult to access immediately. Many exploration and search-related applications have the property that experience improves performance because experience on such tasks accumulates information which can be used to generate more effective performance on the task in the future (Brutschy et al. 2012; Krieger and Billeter 2000; Yamauchi 1998). While such information is easily shared among all team members when a team is small, large-scale teams and difficult environments can make universal information sharing challenging. Such situations are likely to benefit from a response probability mechanism.

We focus on reactive, threshold-based swarms in which agents decide when to act on tasks based on their thresholds for each task. We assume that the swarm is decentralized, each agent acts independently, and there is no communication between agents. These characteristics maximize the potential for scalability in such systems. We study a single-task scenario in which a specified number of agents must act on a given task in order to satisfy the task demand. Each agent has a threshold for the task and agents consider acting on the task when the task stimulus exceeds their threshold value. A *response probability* parameter indicates the probability that an agent will actually act on the task when its threshold is met. For this work, we assume that the response probability value applies to all agents in the swarm.

Probabilistic response in threshold-based systems is not new; however, to our knowledge, our particular application of it is. We begin by explaining how probabilistic response has been used in the past and how it differs from the work in this paper. We then define an abstract model of a single-task scenario and show that this model can be used to estimate the range of response probability values that will allow a swarm of decentralized agents to satisfy task demand and to achieve a specified backup pool. Finally, we test the effects of response probability on an agent-based foraging problem. Results indicate that our theoretical model can predict how response probability affects team and backup pool formation, that there are ranges of response probability values that achieve both, and that response probability may be tuned to emphasize swarm performance or robustness.

2 Background

The response threshold approach is one of several commonly studied methods for achieving decentralized task allocation and division of labor in swarms and multi-agent systems. Threshold-based systems are reactive systems in which agents dynamically decide what task to take on based on the task stimuli sensed at any given time. Each agent possesses a threshold for each task that it can take on. At any given time, an agent's task choice is a function of the agent's task thresholds and the corresponding task stimuli. The simplest such decision function is a direct comparison between an agent's threshold, θ, to the corresponding task stimulus, τ, where $\tau > \theta$ results in agent action on the task. More complex decision functions may include additional factors such as a probability factor, success rate, or perceived actions of other agents. If there is more than one task under consideration and if $\tau > \theta$ may be true for more than one task at a time, system specifications will typically include a policy that describes how agents will select from among multiple tasks in need.

The response threshold approach has been used to generate division of labor in decentralized systems for problems such as paintshop scheduling (Campos et al. 2000; Cicirello and Smith 2002; Kittithreerapronchai and Anderson 2003), mail processing (Goldingay and van Mourik 2013; Price and Tino 2004), foraging (Castello et al. 2013, 2016; Krieger and Billeter 2000; Krieger et al. 2000; Pang et al. 2017; Yang et al. 2010), job shop scheduling (Nouyan 2002), and tracking (Wu and Riggs 2018). In addition, a number of studies have demonstrated division of labor in abstract problem domains (Correll 2008; de Lope et al. 2013; dos Santos and Bazzan 2009; Kanakia et al. 2016; Kazakova and Wu 2018; Niccolini et al. 2010; Wu and Kazakova 2017). In the majority of the systems that use a response threshold approach, agent decisions to act on a task are probabilistic (Castello et al. 2013, 2016; Cicirello and Smith 2002; Correll 2008; dos Santos and Bazzan 2009; Goldingay and van Mourik 2013; Kazakova and Wu 2018; Niccolini et al. 2010; Nouyan 2002; Nouyan et al. 2005; Pang et al. 2017; Price and Tino 2004; Yang et al. 2010), with the probability value typically calculated as a function of an agent's threshold for the task, the current task stimulus value, and additional possible factors. This probabilistic decision-making process occurs both in systems where all agents have the same threshold for a given task and in systems where different agents may have different thresholds for a given task.

The reasons for focusing on probabilistic instead of deterministic decisions in these systems appear to be both historical and practical. Historically, this approach is based on a model division of labor in social insect societies (Bonabeau et al. 1996, 1998), which models behaviors probabilistically because of the non-determinism of natural systems. Given a task stimulus, τ, and an agent threshold, θ, the probability that that agent will act on that task is calculated by

$$P = \frac{\tau^2}{\tau^2 + \theta^2} \tag{1}$$

This model is designed such that: when $\tau \ll \theta$, the probability that an agent will act on the task is close to zero; when $\tau \gg \theta$, the probability of action is close to one hundred percent; and when $\tau = \theta$, the probability of action is fifty percent. Many of the computational swarm approaches directly or in part use this biological model (Campos et al. 2000; Castello et al. 2013, 2016; Cicirello and Smith 2002; Correll 2008; de Lope et al. 2013; dos Santos and Bazzan 2009; Goldingay and van Mourik 2013; Kittithreerapronchai and Anderson 2003; Merkle and Middendorf 2004; Niccolini et al. 2010; Nouyan et al. 2005; Pang et al. 2017; Price and Tino 2004; Yang et al. 2010). Practically, a probabilistic decision-making process increases the diversity of agent behaviors in the swarm. If all agents in a decentralized system have the same threshold or thresholds, deterministic decision making means that all agents that detect the same stimuli will react in the same way. Probabilistic decision making causes some agents to not act when their task threshold indicates that they should. This added diversity in agent behavior increases the repertoire of responses that the swarm as a whole can offer (Ashby 1958), which makes it more likely that a swarm can appropriately cover all tasks that need attendance.

Systems that do not use probabilistic decision making can alternately achieve diversity in how agents respond to stimuli by assigning different threshold values to different agents for the same task (Krieger and Billeter 2000; Krieger et al. 2000; Wu and Riggs 2018). Agents respond to increasing task stimuli in order of increasing threshold value. Heterogeneous threshold values have an added benefit of allowing for division of labor which can improve system efficiency through reduced task switching and increased effectiveness. Because the lowest threshold individuals for a given task are the first to be triggered, they are the most likely to respond to that task. High threshold individuals may not ever be triggered if the lower threshold individuals can sufficiently address a task's demand. Thus, the same individuals are likely to respond to a given task which reduces task switching and, for tasks where experience improves performance, increases the effectiveness of those individuals on performing that task.

Biological studies on social insect societies suggest that, in systems where agents have different thresholds, probabilistic decision making can make a greater contribution beyond simply increasing the diversity in how agents respond to stimuli (Weidenmüller 2004). Instead of the lowest threshold agents always being the ones to respond, a probabilistic response means that some low threshold agents may not respond every time which gives some higher threshold agents an opportunity to act and gain experience. Thus, a non-deterministic response probability can enhance the robustness of the system as a whole by creating backup pools of agents with experience on each task. In decentralized systems where experience improves performance, response probability potentially provides a simple but effective mechanism for balancing the exploitation of experienced agents with providing learning opportunities for less experienced agents.

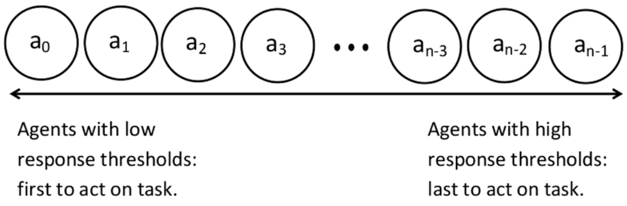
a_0
a_1
a_2
a_3
a_{n-3}
a_{n-2}
a_{n-1}
Agents with low response thresholds: first to act on task.
Agents with high response thresholds: last to act on task.

Fig. 1 The order in which agents in a swarm are expected to respond to the task stimulus is defined by their relative threshold values. Let θ_i denote the threshold of agent a_i where $i = 0, 1, \dots, n-1$. The ordering in this example means that $\theta_i \le \theta_{i+1}$ for all i

The work presented here extends previous work in the following ways. This work focuses on examining this potential new role for response probability rather than on illustrating the ability to achieve division of labor on a particular problem. We investigate how response probability, when added to a system in which agents have differing thresholds for a task, affects agent decisions to act and the number of agents that gain experience. To do this, we extend a system structured like that studied by Krieger and Billeter (2000) and Wu and Riggs (2018), in which agents have heterogeneous thresholds and use a simple direct comparison decision function, by adding a response probability that is applied after the result of the direct comparison. This approach allows us to model the system mathematically while, we expect, still generating subjectively similar behavior as a system based on Eq. 1.

3 Analysis of response probability in robotic swarms

We begin by defining a mathematical model of our system that consists of a decentralized swarm of agents in a single-task scenario. Using this model, we can analyze the system[1] from two perspectives: how response probability affects the swarm's ability to meet task demand and how response probability affects the formation of a backup pool.

3.1 Model

The basic elements of our model are a robotic swarm consisting of n decentralized agents and a task that requires x agents in attendance in order to fully satisfy the task demand, where $x \le n$. Each agent is assigned a threshold value within the range (0.0, 1.0]. The task has an associated stimulus value that grows with increasing need and decreases when agents act on the task. When the task stimulus exceeds an agent's threshold, that agent will respond to the task. As a result, agents with lower thresholds are the first to act on a task; agents with higher thresholds are the last to act on a task. If we implicitly sort the agents by threshold from lowest to highest threshold as shown in Fig. 1, the resulting permutation gives the order in which the agents' thresholds will be exceeded as task stimulus rises, which defines the order in which agents are expected to respond to the task. This implicit

[1] Parts of the analysis were presented earlier (Wu et al. 2012, 2016) and are included here for completeness with permission from AAAI.

ordering means that the likelihood that an agent will act on the task depends on its relative position in the ordering (Pradhan and Wu 2011). Note that the ordering we use below is just a construct for the mathematical analysis; the probabilistic decentralized task allocation method we describe *does not* explicitly order agents. We can use this construct without loss of generality since there exists an implicit ordering based on the response threshold.

The *response probability*, $s : 0.0 \leq s \leq 1.0$, is defined as the probability that an agent will act when the task stimulus exceeds the agent's response threshold. Our model assumes a single response probability value that applies to all agents in the swarm. If $s = 1.0$, the agents are deterministic and only the first x agents will ever act on the task. If $s < 1.0$, there is nonzero chance that agents may not act when the task stimulus exceeds their threshold. If any of the first x agents do not act, then one or more of the agents beyond the first x will have the opportunity to act and gain experience on the task. If s is too low, then there is the chance that not enough agents from the swarm will act to meet the task demand.

We define a *trial* to be one instance in which the swarm attempts to meet the task demand and a *team* to be the group of agents that act in one trial. A *complete team* is a team of size x; that is, a team that fully meets the task demand in a trial. During a trial, each agent may be in one of three states.

- *Candidate*: An agent that receives an opportunity to act because the task stimulus exceeds the agent's threshold. This state is a temporary state that immediately transitions to one of the other two states.
- *Actor*: A candidate that decides to act on the task. A candidate becomes an actor with probability s.
- *Inactive*: An agent that does not receive an opportunity to act or a candidate that chooses not to act.

During a trial, agents become candidates in order of increasing threshold values (in the order defined by the implicit ordering). Each candidate then decides whether to become an actor or remain inactive. A trial ends when either a complete team has been achieved or when all agents have become candidates.

Let z be the total number of agents in the swarm that gain experience (acts on the task at least once) over multiple trials. Figure 2 illustrates how the response probability, s, affects z. This example shows a swarm of size $n = 8$, a task that requires $x = 3$ agent to satisfy the demand, and a series of three trials. Although a maximum of x agents become actors in each trial, over multiple trials, when $s = 1.0$, $z = x$ is always true and when $0.0 < s < 1.0$, the value of z may vary but $z \geq x$ is true. Over multiple trials, as the value of s decreases, the value of z increases.

3.2 Meeting task demand

In order for a swarm to be useful, it must be able to meet the task demand by forming a complete team (a team of size x). While we can ensure that all agents in a swarm will gain experience on a task over multiple trials by setting s to a low value, having a swarm full of "experienced" agents is pointless if the swarm is unable to form a full response team. As a result, the first question that we ask is, given that we have n agents and a task that requires $x : x < n$ agents, can we determine what s values will allow the swarm to successfully form a complete team?

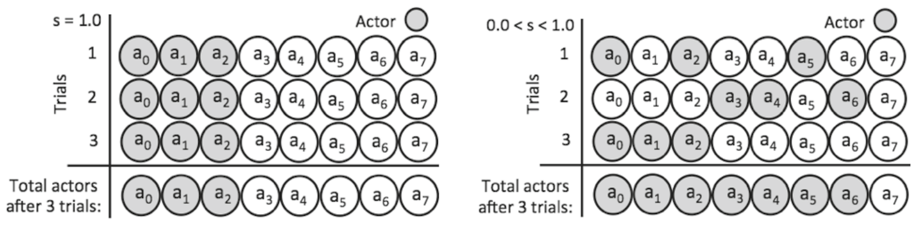

Fig. 2 Example of effects of response probability, s, on a swarm of size $n = 8$ and task demand of $x = 3$. Over multiple trials, when $s = 1.0$ only $z = x$ agents gain experience; when $0.0 < s < 1.0$, $z \geq x$ agents gain experience

We begin by traversing all n agents and marking each agent with probability s. If an agent is marked then it *will* choose to act if given the opportunity. Of course, there may be marked agents that never receive the opportunity, but this will not affect the analysis below. Let M be a random variable specifying the total number of marked agents, regardless of whether the agents participate in the team or not.

We use $\Omega(g(n))$ to denote the set of functions that asymptotically bound $g(n)$ from below. Formally:

$$\Omega(g(n)) = \left\{ \begin{array}{l} f(n): \text{ there exist positive constants } c \\ \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \\ \text{ for all } n \geq n_0 \end{array} \right\}$$

For those less familiar with this notation, it may be helpful for our discussion to point out that the class of functions $e^{-\Omega(g(n))}$ is the set of functions that drop toward zero as an exponential function of $g(n)$ or faster.

The following lemma provides the baseline bounds and bounding techniques that we will be using. The proof for the lemma originally appeared in (Wu et al. 2012). We provide the full proof here, as well, since it is needed for context and also corrects a couple of minor typographical errors in the original publication.

Lemma 1 *A single trial of the task allocation process will result in* $M \leq x - 1$ *with probability* $1 - e^{-\Omega(n)}$ *when* $s < \frac{x-1}{en}$. *It will result in* $M \geq x$ *with probability* $1 - e^{-\Omega(n)}$ *when* $s > \frac{x}{n}$.

Proof To complete this proof, we use Chernoff bounds (Motwani and Raghavan 1995), which provides ways of bounding the probability of deviating from the expected value of a Poisson process by a small factor. We will use both bounds. For the first bound, we use the fact that for some random variable X governed by a Poisson process:

$$\Pr\left[X > (1+\delta)\,\mathrm{E}[X]\right] < \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{E[X]} \tag{2}$$

For the second bound, we use that fact that for some random variable X governed by a Poisson process:

$$\Pr\left[X < (1-\delta)\,\mathrm{E}[X]\right] < e^{-\delta^2 E[M]/2} \tag{3}$$

Note that e is Euler's number ($e \approx 2.71828$), not a variable.

The expected number of marked agents is $n \cdot s$, $\mathrm{E}[M] = ns$, since there are n agents to be marked and each is marked with independent probability s. For the first bound, we choose a convenient factor δ to affect the expected number of marks. The objective of our choice is to simplify the bound algebraically. Since the first Chernoff bound presented above has a term of the form $(1+\delta)\,\mathrm{E}[M]$, we would like to construct a δ that will both simplify the bound and also give us a useful final bound. Specifically, we choose to let

$$\delta = \frac{x-1}{ns} - 1$$

and this allows us to simplify $(1+\delta)\,\mathrm{E}[M]$ as follows:

$$(1+\delta)ns = \left(1 + \frac{x-1}{ns} - 1\right)ns = x - 1$$

Substituting into Eq. 2, we get

$$\begin{aligned}\Pr[M > x-1] &< \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{E[M]}\\ &= \left[\frac{e^{\frac{x-1}{ns}-1}}{(\frac{x-1}{ns})^{\frac{x-1}{ns}}}\right]^{ns}\\ &= \frac{1}{e^{ns}}\left[\left(\frac{ens}{x-1}\right)^{\frac{x-1}{ns}}\right]^{ns}\end{aligned}$$

If $x-1 > ens$, this converges to 0 exponentially fast as n grows. Thus, $s < \frac{x-1}{en}$ implies $\Pr[M < x] = 1 - e^{-\Omega(n)}$, which bounds s from above.

For the second bound, consider the convenient factor $\delta = 1 - \frac{x}{ns}$ and note that

$$(1-\delta)ns = \left(1 - 1 + \frac{x}{ns}\right)ns = x$$

We use the second Chernoff inequality presented above to bound the probability that there are fewer than x marks. Substituting into equation 3, we get

$$\begin{aligned}\Pr[M < x] &= \Pr[M < (1-\delta)\,\mathrm{E}[M]]\\ &< e^{-\delta^2 E[M]/2}\\ &= e^{-\left(1-\frac{x}{ns}\right)^2 ns/2}\\ &= e^{-\frac{1}{2ns}(ns-x)^2}\end{aligned}$$

If $x < ns$, this converges to 0 exponentially fast as n grows. Thus, $s > \frac{x}{n}$ implies $\Pr[M \geq x] = 1 - e^{-\Omega(n)}$, which bounds s from below. □

Theorem 1 *With probability growing exponentially in n, $1 - e^{-\Omega(n)}$, a complete team will be formed when $s > \frac{x}{n}$ and will not be formed when $s < \frac{x-1}{en}$.*

Proof If there are fewer than x marks over all n agents, a complete team of x agents will not be formed, and a complete team can only be formed if there are x or more marks. Noting this, the conclusion follows from the marking-probability bounds proved in Lemma 1.

□

Asymptotic analysis gives us general bounds for the response probability as n grows; however, it is constructive to see how this practically relates to a specific scenario. Consider a system with $n = 100$ agents and a task that requires $x = 20$ agents. If s_1 equals the value of s for which a team is likely to be formed in a single trial, then

$$s_1 > \frac{x}{n} = \frac{20}{100} = 0.2$$

If s_2 equals the value of s for which a team is almost surely not be formed in a single trial, then

$$s_2 < \frac{x-1}{en} = \frac{19}{e \cdot 100} \approx 0.069897 \approx 0.07$$

We compare the predicted values of s_1 and s_2 with empirical results from a simulation of team formation by decentralized threshold-based agents. In the simulation, each agent has a threshold for a task; lower thresholds indicate quicker response. Recall that one trial is defined to be one instance of the task. In each trial, agents, in order of increasing thresholds, decide whether or not to act with a probability s. The trial ends when either a complete team is formed or when all agents have passed through the candidate state. Each *run* of the simulation consists of 100 trials, and we record the number of trials out of 100 in which a team is formed.

Figure 3 compares the predicted values for s_1 and s_2 for two scenarios with the corresponding team formation data from the simulation. The x-axis in both figures plots the s values where $0.01 \leq s \leq 1.0$. The y-axis shows the percentage of trials in a run in which a team is formed, averaged over 20 runs, with 95% confidence intervals. The s_1 and s_2 values calculated above are indicated by two vertical lines. The left plot shows the results from a simulation consisting of 100 agents ($n = 100$) and a task that requires 20 agents ($x = 20$). The right plot shows results from a simulation consisting of 500 agents ($n = 500$) and a task that requires 300 agents ($x = 300$). Simulations using other n and x values yield comparable results.

In both examples, the calculated value for s_1, which indicates the threshold above which a team is likely to be formed in a single trial, falls on s values where the team formation percentage is fifty percent or above. The calculated value for s_2, which indicates the threshold below which a team is unlikely to be formed in a single trial, falls on s values where team formation percentage is zero. It is likely that the $\frac{x}{n}$ bound is tight and that $\frac{x-1}{en}$ is overly cautious.

3.3 Forming a backup pool

For tasks where experience improves performance, system robustness can be improved by generating and maintaining a pool of agents beyond primary actors that have experience on the task. Because, in each trial, a maximum of x agents can act and gain experience, this redundancy can only be generated over multiple trials. The second question we ask is whether we can determine the appropriate value of s to use in a system when a specified level of redundancy is desired. To do so, we first look at what happens in a single trial and ignore the possibility of failing to make a team, then examine the combination of these results with the recommendations on team formation from the previous section.

We define $c > 1$ to be the desired level of redundancy where cx is the desired number of agents with experience (and the size of the backup pool is $cx - x$). Our goal is to determine the s values for which the cxth agent is highly likely to have gained experience and be part of the pool. The probability of the ith agent acting and therefore gaining experience (P_i) is no smaller for agents preceding the cxth in the ordering (i.e., $P_i \geq P_{cx}$ if $i \leq cx$) since those will be given the opportunity to act sooner and all agents have the same s. Here, we show formally that if $s \leq \frac{1}{ec}$, there is a constant probability that the cxth agent will gain experience and if $s > \frac{1}{c}$, then it will almost certainly fail to gain experience. After deriving these bounds, we present an empirical case to show that the real system is consistent with our formal advice.

The proof for this follows a similar structure as the above proof, and we begin by traversing all n agents and marking each agent with probability s. Let K be a random variable specifying the number of marks in the first cx agents. We first describe a bound on s that is *sufficient* to assure a reasonable P_i, then we describe a looser bound that is *required* if we do not want P_i to converge to 0 with team size. Once again, we replicate the proof from (Wu et al. 2012) for context.

Theorem 2 *In a single trial of the task allocation process, if* $s \leq \frac{1}{ec}$, *then* $P_{cx} = 1 - e^{-\Omega(1)}$.

Proof The expected number of marks in the first cx agents is $c \cdot x \cdot s$, $\mathrm{E}[K] = cxs$. We again start with a convenient factor,

$$\delta = \frac{1}{cs} - 1$$

and note that

$$(1+\delta)cxs = \left(1 + \frac{1}{cs} - 1\right)cxs = x$$

We use Chernoff inequality to bound the probability that there are at least x marks in the first cx agents:

$$\begin{aligned}\Pr[K \geq x] &= \Pr[K > (1+\delta)\,\mathrm{E}[K]] \\ &< \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{E[K]} \\ &= e^{-csx} \cdot (ecs)^x\end{aligned}$$

So when $s < \frac{1}{ec}$, this approaches 0 exponentially fast with x. Thus, the cxth agent almost surely is given the opportunity to act and $P_{cx} = s \cdot \left(1 - e^{-\Omega(x)}\right) \approx s = 1 - e^{-\Omega(1)}$. □

Theorem 2 gives us a bound for *sufficient* values of s for the cxth agent to gain experience. If the cxth agent has a constant probability of gaining experience in a single trial, then a constant number of repeated trials will ensure the agent eventually gains experience. As stated, agents that precede the cxth agent in the ordering will have no worse probability of gaining experience, so the same logic applies to all of them. The question of how many trials are necessary to ensure this experience is a different one which is addressed by (Wu et al. 2012). Now that we have seen what sufficient values of s are to ensure experience, let us examine what values of s are so large that they prevent the cxth agent from even being given the opportunity to gain experience.

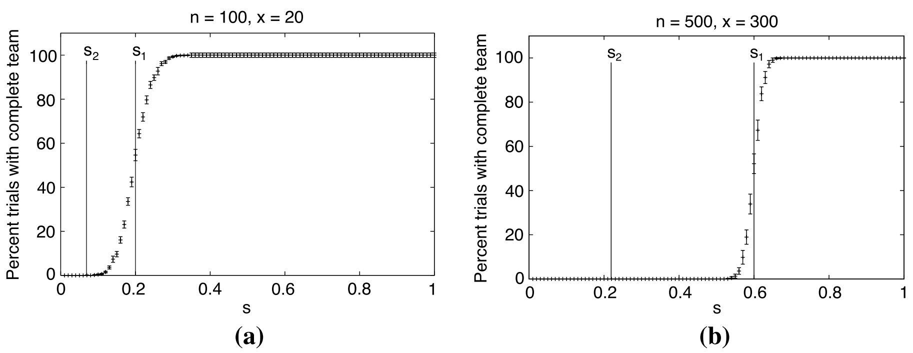

Fig. 3 Comparison of calculated values for s_1 and s_2 with empirical data on the percentage of trials in a run in which a complete team is formed, averaged over 20 runs, as s varies from 0.0 to 1.0: two example scenarios with $n = 100$ and $x = 20$ (**a**) and $n = 500$ and $x = 300$ (**b**). The x-axis in both figures indicates the response probability, s. The y-axis indicates the percentage of trials in a run in which a team is formed, averaged over 20 runs. When $n = 100$ and $x = 20$ (**a**), $s_1 = 0.2$ is the response probability value above which a team is likely to be formed in a single trial and $s_2 \approx 0.07$ is the response probability value below which a team is unlikely to be formed in a single trial. When $n = 500$ and $x = 300$ (**b**), $s_1 = 0.6$ is the response probability value above which a team is likely to be formed in a single trial and $s_2 \approx 0.22$ is the response probability value below which a team is unlikely to be formed in a single trial

Theorem 3 *In a single trial of the task allocation process, if* $s > \frac{1}{c}$, *then* $P_{cx} = e^{-\Omega(x)}$.

Proof Again $\mathrm{E}[K] = cxs$. Now let

$$\delta = 1 - \frac{1}{cs}$$

and note that

$$(1-\delta)cxs = \left(1 - 1 + \frac{1}{cs}\right)cxs = x$$

We use Chernoff inequality to bound the probability that there are fewer than x marks in the first cx agents:

$$\begin{aligned}\Pr[K < x] &= \Pr[K < (1-\delta)\,\mathrm{E}[K]] \\ &< e^{\delta^2 E[K]/2} \\ &= e^{-\frac{x}{2cs}\left(1-\frac{1}{cs}\right)^2}\end{aligned}$$

So when $s > \frac{1}{c}$, this approaches 0 exponentially fast with x. Thus, the probability that the cxth agent is even given the opportunity to act is exponentially small for constant s. □

Theorem 3 tells us that when $s > \frac{1}{c}$, later agents will very probably be starved of the opportunity to gain experience by the agents before them in the ordering. This expectation means that unless there are an exponential number of trials, it is unlikely that repeated trials will allow us to build up a redundancy of cx agents with experience. Of course, these are asymptotic results that become more correct as team size and redundancy factor increase. Therefore, it is instructive to consider empirical results for specific values.

We compare the predicted values to empirical data from the simulation from the previous section. Consider again the scenario in which there are 100 agents ($n = 100$) and we face a task that requires 20 agents ($x = 20$). We would like a redundancy factor of 2 ($c = 2$) which means that we would like the first 40 agents to gain experience on that task over multiple trials.

Figure 4 shows the influence of s on the formation of backup pool of agents. Again, the x-axis plots s values, but now the y-axis measures the average number of actors. The plot shows the average number of actors and 95% confidence intervals over 20 simulations. Figure 4 shows that, for the s value when the cxth agent is likely to act ($s \approx 0.18$), the maximum average number of actors obtained is around 85, which means the cxth or 40th agent has acted and is in the backup pool and the desired redundancy is achieved. At the s value above which the cxth agent is unlikely to act ($s = 0.5$), the maximum number of actors obtained is around 48. Again, the cxth or 40th agent makes it into the backup pool and the desired redundancy is achieved. These empirical results are consistent with our formal predictions even though the values for n, x, and c are relatively small. Moreover, for large c, the difference between $\frac{1}{c}$ and $\frac{1}{ec}$ becomes smaller, making the advice more specific as more redundancy is required.

Note that the average number of actors again drops when s is too small, even though $s < \frac{1}{ec}$. This occurs for a different reason: complete teams are not being formed when s is too small. Combining the theoretical advice from the previous section with this section and assuming the user wants to maximize the back up pool with *at least cx* agents with experience, we see that our theory can offer very specific and tangible advice.

Referring back to Fig. 3a where n=100 and x=20, complete teams will almost certainly form if $s \geq \frac{x}{n} = 0.2$ and, according to Fig. 4, c-factor redundancy is ensured if $s < \frac{1}{ec} \approx 0.18$. As we can see in Fig. 4, an s value in the range $0.18<s<0.2$ is indeed a good first choice to achieve c=2 redundancy. Note that if $s < \frac{x-1}{en} \approx 0.07$, we will almost certainly *fail* to form a team—which is roughly close to where we lose a redundancy factor of two on the left part of the graph. Also, if $s > \frac{1}{c} = 0.5$, we will almost certainly starve later agents and fail to get redundancy, which is roughly close to where we lose a redundancy factor of two on the right part of the graph.

4 Foraging problem

While the analysis above provides guidance on how response probability will affect if and when agents in a threshold-based swarm receive the opportunity to gain experience on a task, it is too simple to allow us to study how response probability may affect system robustness. To more thoroughly examine the latter, we study an agent-based simulation of a foraging problem.

The foraging problem is a common testbed for swarm robotics because it is a complex coordination and task allocation problem that is representative of many real-world applications (Winfield 2013). It is an effective testbed for examining the costs and benefits of single versus multiple agent systems (Brutschy et al. 2012; Krieger and Billeter 2000; Labella et al. 2006; Pini et al. 2013). In a foraging problem, all (Lerman et al. 2006; Pini et al. 2013, 2014) or a subset (Castello et al. 2016; Krieger and Billeter 2000; Labella et al. 2006; Pang et al. 2017; Yang et al. 2010) of the agents in a swarm search a working area to find and retrieve one (Krieger and Billeter 2000; Labella et al. 2006; Pini et al. 2013, 2014) or more (Lee and Kim 2017; Lerman et al. 2006) types of resources. Agents may or may

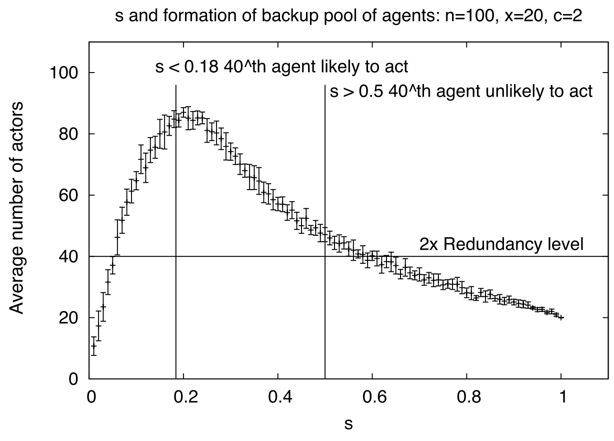

Fig. 4 Effect of s on the formation of a backup pool of $c = 2$ in a single task problem. The x-axis indicates the response probability, s, and the y-axis indicates the average number of actors. The vertical lines indicate our formal bounds, and the horizontal line shows when the redundancy factor is met

not adapt their behavior over time in response to environmental[2] (Brutschy et al. 2014; Lerman et al. 2006; Pini et al. 2011) or internal (Agassounon et al. 2001) factors.

In the problem that we use, a decentralized swarm of agents searches a two-dimensional working environment for a single resource that is available at one or more locations in the environment. Agents have multiple opportunities to search for the resource, gaining information about where in the environment to find the resource in each search attempt. Over multiple explorations, agents learn where to find the resource and how to get there efficiently. Thus, the task that the agents take on is to retrieve one unit of resource (from any location), and experience refers to an agent's information about the working environment and where in the environment the resource may be obtained.

It is important to note that this problem is a single-task problem where the task in question for each agent is whether or not to go forage for the resource. It is not a multi-task problem where each location is a separate task. Although the resource may be available at more than one location, all locations contain the same resource. Regardless of the number of resource locations, the goal of the swarm is to address the single task of retrieving sufficient units of resource. The task that each agent considers whether or not to take on is: shall I go retrieve one unit of the resource (from any location).

We first describe the simulation and how it relates to the analytical model. We then present the experimental setup, and discuss the results and their implications. We further connect the results with the bounds proved above for the simpler problem domain to better show how these two problems relate to one another.

4.1 System description

The simulation consists of a swarm of n agents searching for a resource in a two-dimensional environment of width w and length l. Agents enter and exit the environment at an entrance located at the center of the environment, at coordinates $(\frac{w}{2}, \frac{l}{2})$. Within the environment, there are ρ sources of the resource, where $\rho \geq 1$. These resource locations may be randomly generated or user-specified; in either case, they remain fixed through the duration of a simulation run. Figure 5 shows an example environment with $\rho = 4$. The four blue

[2] Includes perceived local or global swarm status.

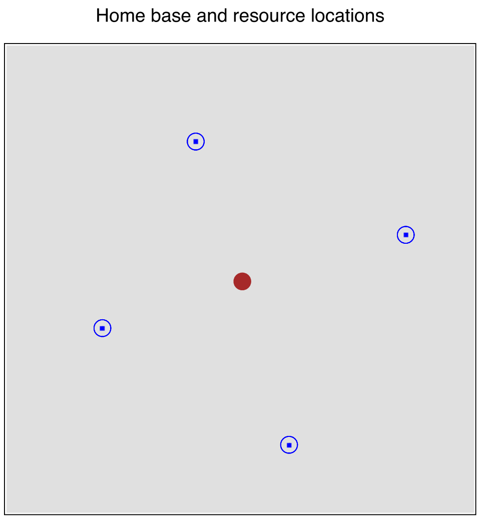

Fig. 5 An example environment where $\rho = 4$. The brown dot in the center indicates the swarm's entry point into the environment. The blue targets indicate locations where the resource is available

targets represent locations where the resource is available. The brown dot at the center of the environment marks the swarm's entry point into the environment.

Each *run* of the simulation consists of multiple *trials*. In each trial, x agents leave the home base to forage for the resource; each agent is able to travel a maximum of Ω *steps* before needing to return to the home base to recharge. Each agent's foraging attempt continues until either the agent has found a resource location or the agent has travelled Ω steps. A trial ends after x agents have foraged (successfully or not) or after all n agents have entered and transitioned out of the candidate state (as defined in Sect. 3.1). Each time an agent travels to a resource location, it retrieves one unit of the resource. The performance of the swarm is measured by the number of resource units retrieved by the swarm in each trial.

Each agent maintains its own *memory map* of the environment. An agent's memory map is a $w \times l$ grid. Each cell in the grid, $m_{i,j} : i = 0, \dots, w-1$ and $j = 0, \dots, l-1$, encodes a floating point value in the range [0.0, 2.0]. At the start of a run, all memory map cells are initialized to $m_{i,j} = 1.0$. Each time an agent forages, it keeps track of the path that it traverses in the environment. If the agent is successful in finding the resource, the agent records the path in its memory map as favorable. If the agent is unsuccessful in finding the resource, the agent records the path in its memory map as unfavorable. Specifically, let ϕ be the positive reinforcement factor and ξ be the negative reinforcement factor. Both ϕ and ξ are user-set parameters with range [0.0, 1.0]. If an agent is successful in finding the resource,

$$m_{i,j} = m_{i,j} + (2.0 - m_{i,j}) \times \phi \tag{4}$$

for all $m_{i,j}$ in the agent's path. If an agent is unsuccessful in finding the resource,

$$m_{i,j} = m_{i,j} - m_{i,j} \times \xi \times \frac{t(m_{i,j})}{\Omega} \tag{5}$$

for all $m_{i,j}$ in the agent's path. The function $t(m_{i,j})$ returns an integer value indicating what step number the cell $m_{i,j}$ is in the agent's path, allowing the later steps in the path to be reinforced more than earlier steps. Figure 6 shows snapshots of the memory map of one agent in an example run in the environment shown in Fig. 5. Over multiple trials, cells that are likely to lead to a resource location become more desirable (more green), and cells that are unlikely to lead to a resource location become less desirable (more red).

The path that an agent takes each time it forages is guided by the agent's current memory map. The maximum length of a path is Ω steps. The agent moves a distance of one cell in each step. In each step, the agent considers all immediate neighbors[3] from its current location and probabilistically selects the most favorable cell to which to move. The probability that a cell will be selected as an agent's next step is equal to the value of that cell divided by the sum of the values of all cells under consideration. At the start of a run, when all memory cells have been initialized to 1.0, agents basically move in a random walk. As an agent gains experience over multiple trials, positive and negative reinforcements to cell values based on successful and unsuccessful foraging attempts help the agent differentiate between promising and unpromising cells to which to move.

This foraging problem is appropriate for testing the conclusions of our analytical model because it is an example of a problem in which experience (information gained during foraging trials) improves performance (ability to travel efficiently to resource locations). The task that the swarm seeks to address is to forage sufficient amounts of a resource in each trial. The x agents called upon in each trial to address this task represent the expected number of agents needed to meet the foraging demand, given the fact that each agent can retrieve a single unit of resource per trial. The performance of the system is measured by the amount of resource collected per trial. As with the analytical model, the response threshold aspect of the system is simulated via an implicit ordering of the agents. This implicit ordering represents the order in which the agent thresholds would be triggered as task demand increases. Experience, in this problem, refers to an agent's knowledge of where in the environment to go to find the resource. Each trial is one opportunity for an agent to improve its knowledge of the environment which improves its ability to retrieve one unit of resource. The more trials that an agent receives to forage, the more likely it is able to find a resource location and, once a location has been found, the more strongly a path to that location will be reinforced. The response probability affects which of the n agents become the x actors in each trial: the first x agents from the implicit ordering that accept the task (given the response probability) are the agents that act.

Recall that, in a given trial, a single agent needs only to retrieve one unit of resource, from any location, to have successfully completed its task. As a result, a single agent will not necessarily find all resource locations and, because agents do not communicate or share memory maps, different agents may find different resource locations. The collective knowledge of the swarm includes all resource locations that have been found, as shown in Fig. 7.

While allowing agents to communicate and share information[4] has the potential to speed up the rate at which the swarm as a whole discovers and exploits a resource location (Pini et al. 2013), this speed comes at a cost. Maintaining a central shared repository of all agents' experience basically turns this problem into a model of stigmergic search which is known to focus swarm exploitation on the single closest resource location found (Goss et al. 1989, 1990). Focusing all agents on a single resource location leads to potential

[3] Von Neumann neighborhood with radius one.

[4] For example, by incorporating individual memory maps into a central map at the nest that is shared with all agents.

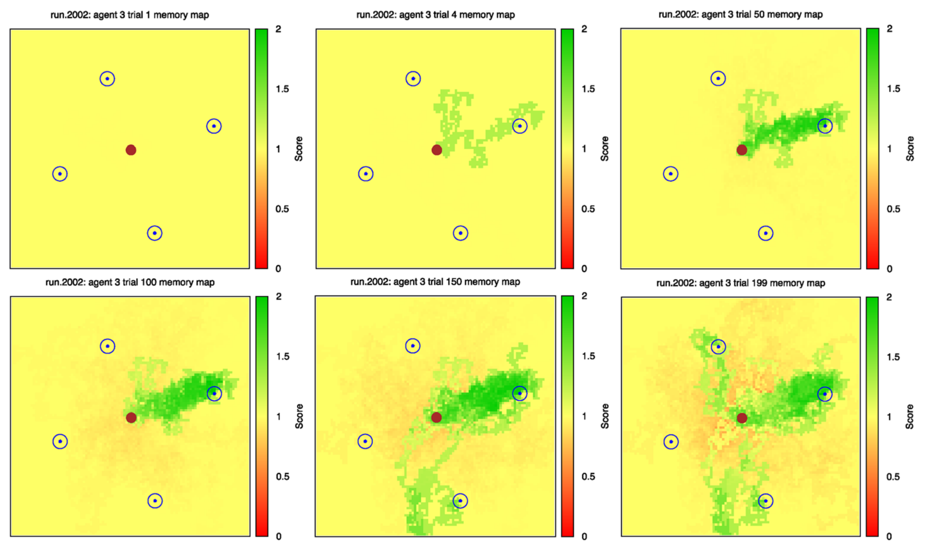

Fig. 6 Snapshots of a single agent's memory map after trials 1, 4, 50 (top row, left to right) and 100, 150, 199 (bottom row, left to right) in an example run in the environment shown in Figure 5. Green indicates that a cell is likely to lead to the resource. Red indicates that a cell is unlikely to lead to the resource. Yellow is neutral, indicating that a cell is equally likely or unlikely to lead to the resource. The brown dot in the center indicates the swarm's entry point into the environment. The blue targets indicate locations where the resource is available

interference and traffic jam issues and leaves the swarm vulnerable to gaps in resource availability: if the location exploited by the entire swarm runs out of resource, the swarm will experience a lack of resource until a new location can be found. Alternatively, our model, which disallows inter-agent communication, distributes the information gained by the swarm across multiple agents which distributes agent activity and lessens inter-agent interference. Because different agents may find and exploit different resource locations, if one resource location becomes non-productive, only a subset of agents will be affected. Agents that have focused on other locations will continue to perform at full capacity, while the affected agents revise their memory maps to reflect the new resource availability.

4.2 Experimental setup

We examine how a swarm's ability to find and retrieve a resource is affected by periodic loss of the swarm's primary foragers as response probability varies. Table 1 lists the parameters that we use in the example scenario presented here. Alternative scenarios and parameter settings result in comparable system behavior as response probability varies.

We use a swarm of size $n = 200$ working in the environment shown in Fig. 5. Each run consists of 1000 trials. In each trial, $x = 20$ agents leave the base to forage. At trials 125, 250, 375, and 500, the twenty most experienced agents are removed from the swarm and the remaining swarm members continue to carry out the foraging task. These twenty agents represent the twenty agents with the current lowest thresholds. This removal rate means that, if the agents are acting deterministically ($s = 1.0$), all experienced agents are removed from the swarm and the swarm continues after each removal with no knowledge of the

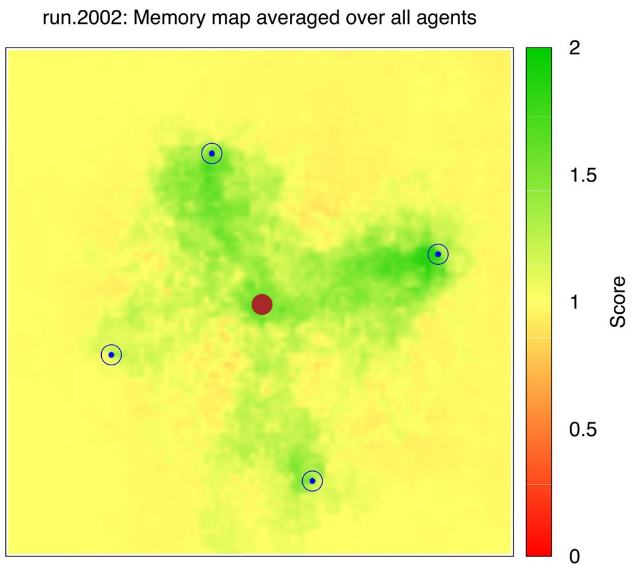

Fig. 7 The average of the memory maps of all agents in the swarm at the end of an example run in the environment shown in Fig. 5. Green indicates that a cell is likely to lead to the resource. Red indicates that a cell is unlikely to lead to the resource. Yellow is neutral, indicating that a cell is equally likely or unlikely to lead to the resource. The brown dot in the center indicates the swarm's entry point into the environment. The blue targets indicate locations where the resource is available

resource locations. For $s < 1.0$, there may be agents beyond the first twenty with some foraging experience.

Swarm performance is measured in terms of the number of resource units retrieved in each trial. In the trial immediately following an agent removal event, this amount drops then gradually increases again as the newly active agents gain experience. Robustness is evaluated as the distance of this drop and the resulting level of resource after the drop. Because agent behavior is probabilistic, both when $s < 1.0$ and when making decisions about where to move in each step, we perform 100 runs for each response probability value. For each run, we track the number of resource units retrieved by the swarm in each trial and average the results for each trial over all 100 runs.

Table 1 Parameter settings used in experiments

Parameter description	Variable	Value
Swarm size	n	200
Team size	x	20
Positive reinforcement factor	ϕ	0.3
Negative reinforcement factor	ξ	0.3
Maximum steps	Ω	1000
Environment width	w	100
Environment length	l	100
Number of resource locations	ρ	4
Number of trials per run		1000

These values were selected empirically

4.3 Results

We evaluate our system by examining both how a swarm performs from one trial to the next within a run and aggregating metrics over multiple runs to compare how performance varies for different response probability values.

Figure 8 shows the average expected behavior of a swarm during a run for different response probability values. Specifically, Fig. 8 shows the average resource units retrieved by a swarm in each trial of a run. Each plot shows the results, averaged over 100 runs, for one response probability value, from $s = 1.0$ to $s = 0.1$ in increments of 0.1. The x-axes indicate the trial number, and the y-axes indicate the average resource units retrieved. The blue solid line shows the average resource units retrieved, the green region around the blue line indicates the 95% confidence interval, and the red dashed line shows the average *drop level*, the average performance level to which the swarm drops each time the lowest threshold twenty agents are removed. Removals occur at trials 125, 250, 375, and 500.

Figure 9 shows aggregated measurements of the swarm's success rate at forming a complete team and the swarm's ability to withstand loss of agents. The x-axis indicates response probability values from $s = 0.05$ to $s = 1.0$. The y-axis on the left measures the percentage of trials that achieve complete teams. The y-axis on the right measures the number of resource units for evaluating drop amount and drop level. The individual elements of this plot will be described in the discussion below.

We can see from Fig. 8 that swarm performance (in terms of the number of resource units retrieved in a trial) improves as agents gain experience over multiple trials. Improvement is faster with higher s and slower with lower s. Higher s values focus the activity on fewer agents which gives those few agents more opportunities to gain experience and increases the rate at which those few agents improve in performance. Lower s values distribute activity among more agents which lowers the number of opportunities any individual agent gets to gain experience and improve performance.

Each time we remove the current lowest threshold agents, swarm performance drops abruptly due to the loss of experienced agents then improves again as the remaining agents gain experience. The amount by which the performance drops is highest for $s = 1.0$ and decreases as s decreases. The dashed pink line in Fig. 9 plots the *drop amount*, the average number of resource units by which performance drops after agent removal for response probability values from $s = 0.05$ to $s = 1.0$. The observed proportional relationship is consistent with the fact that higher s values mean that swarm effort and experience are focused on the agents with the lowest thresholds, resulting in the remaining agents having fewer opportunities to gain experience. As a result, if the primary actors are removed, the performance capabilities of the remaining agents are much lower than that of swarms with lower s which give more agents opportunities to gain experience, and the drop in performance will be much larger.

The impact of a performance drop cannot be fully evaluated without also considering the level to which the performance drops. Because higher response probabilities result in better overall swarm performance, swarms with higher s values can withstand larger drops before reaching unacceptable levels of performance. The red dashed lines in the plots from Fig. 8 indicate the average performance level immediately after a drop. The solid blue line in Fig. 9 consolidates this data into a single plot and shows how the average drop level varies as s varies. As the response probability decreases from $s = 1.0$ to $s \approx 0.25$, the average drop level increases, indicating that the removal of agents has lower impact on swarm

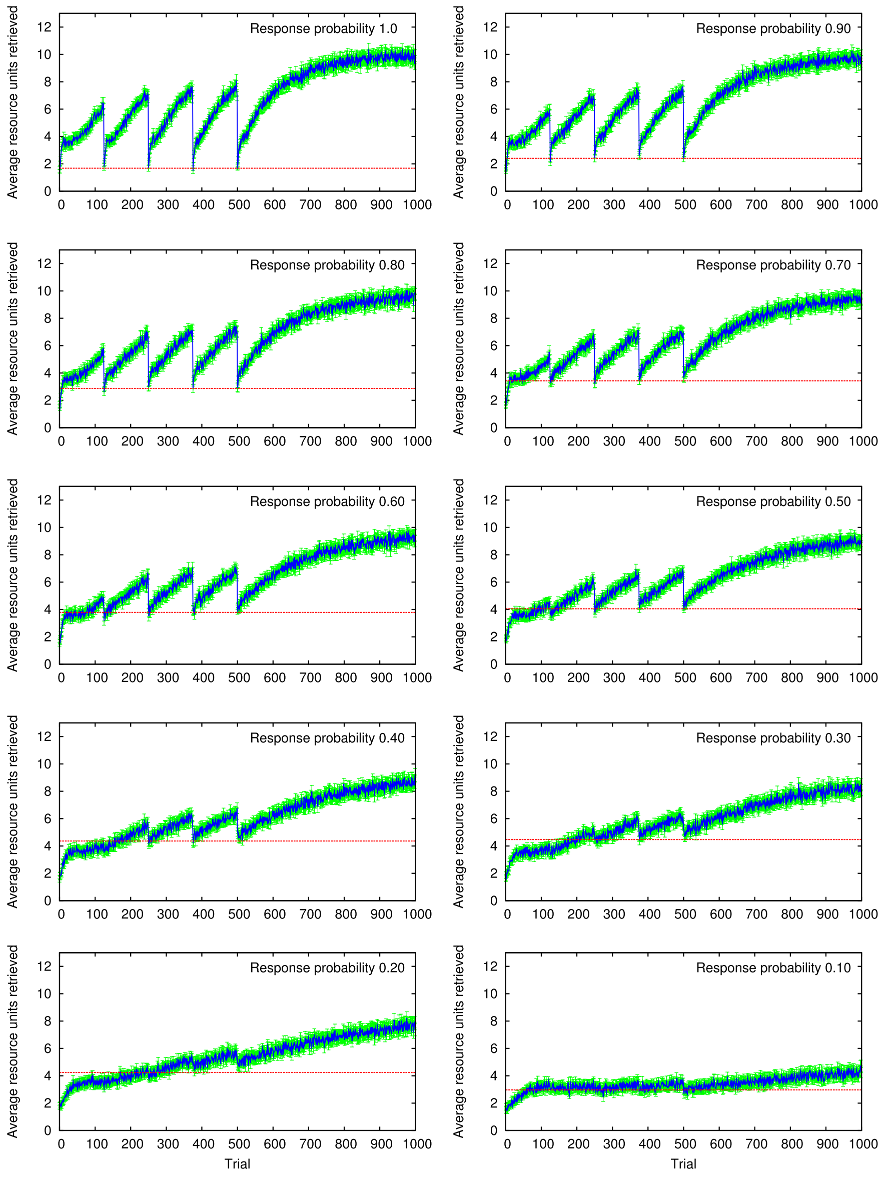

Fig. 8 Average number of resource units (blue solid line) obtained in each trial and 95% confidence interval (green region), averaged over 100 runs. The red dashed line shows the average performance level to which the swarm drops each time the lowest threshold twenty agents are removed

performance. In the scenario studied here, average drop level peaks at $s \approx 0.25$. Below $s = 0.25$, the average drop level decreases and approaches zero as s approaches zero.

The dotted red line in Fig. 9 shows the average percentage of trials in a run in which the swarm forms a complete team, averaged over 100 runs. This plot is consistent with Theorem 1 that predicts that a complete team is likely to be formed when $s_1 > \frac{x}{n} = \frac{20}{120} = 0.17$

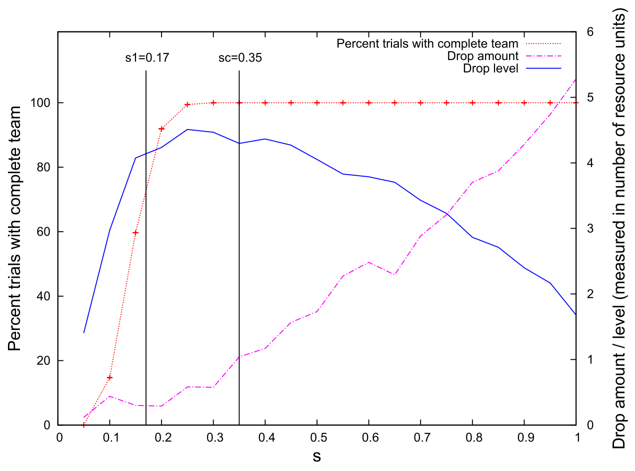

Fig. 9 Measurements of swarm team formation, drop amount, and drop level for response probability values from $s = 0.05$ to $s = 1.0$ on the environment shown in Fig. 5. The x-axis indicates the response probability, s

for this problem.[5] Empirical analysis finds that, for this scenario, the swarm achieves a complete team in every trial of every run for $s_c \geq 0.35$. These results help to explain the decline in the drop level for $s < 0.25$. This decline is likely a decline in overall swarm performance due to inability to form a complete team, which would result in substandard performance due simply to the inadequate number of actors.

The results presented above use the environment shown in Fig. 5. To ensure that there is no bias due to the particular resource location configuration of that environment, we re-run the same experiments with $\rho = 4$ resource locations but generate a random placement of the resources for each run. The resource locations are fixed and unchanged for all trials of a run, but each run has a different randomly generated configuration. Figure 10 shows the average drop amount and the average drop level for response probability values from $s = 0.05$ to $s = 1.0$, averaged over 100 runs. While the magnitudes are different and there is more variability, the relative changes in the plots as s varies are consistent with the corresponding results from Fig. 9. The swarm again achieves a complete team in every trial of every run for $s_c \geq 0.35$.

The cumulative swarm performance in each period between the incidents of agent loss further supports our hypothesis that probabilistic agent response can generate a backup pool that can mitigate performance drop when actors are lost. Table 2 shows, for the data from Fig. 8, the total amount of resource units collected by a swarm within each period of stable agent population size. Each row gives the data for one s value, and each column gives the data for one period of stable agent population size. The bottom row, for $s = 1.0$, shows the performance of the system when all learning opportunities are concentrated on the lowest threshold agents and the lowest threshold agents are the only ones that act. In other words, this row shows the swarm performance when the agents that act have all gained the maximum amount of experience possible. For all

[5] Although the runs begin with $n = 200$, four losses (of 20 agents each) drops the swarm size down to $n = 120$ for the entire second half of a run.

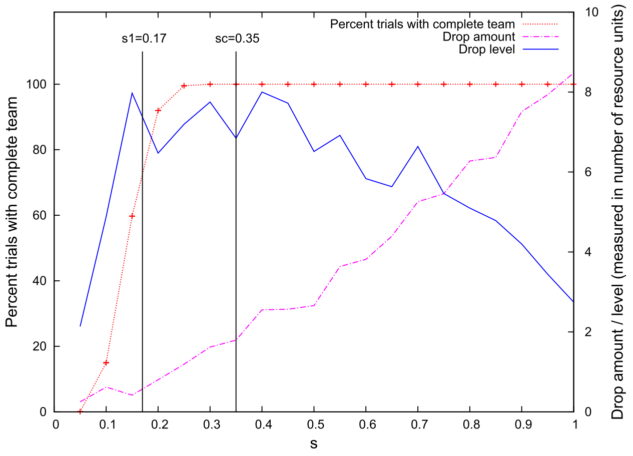

Fig. 10 Measurements of swarm team formation, drop amount, and drop level for response probability values from $s = 0.05$ to $s = 1.0$ on random environments with $\rho = 4$. The x-axis indicates the response probability, s

other rows, the primary actors sacrifice some opportunities to gain experience to backup actors.

Table 3 shows, for each s and each period, the amount of resource retrieved by a swarm as a percentage of the amount of resource retrieved when $s = 1.0$. That is, the data in Table 3 is calculated by dividing the values in each of the rows of Table 2 by the corresponding value in the last row of Table 2, which shows us how swarm performance when primary actors do not receive maximal experience ($s < 1.0$) compares with performance when primary actors do ($s = 1.0$). Values of 100% or above indicate that performance is as good or better than the maximal experience case.

Column 2 of Table 3 gives a relative measure of swarm performance (relative to a swarm with $s = 1.0$) for the period consisting of trials 0–124. These data illustrate that, when starting from a population of inexperienced agents, the more that task experience is focused on the most experienced agents (the higher the s), the better the performance achieved. As the swarm builds a backup pool of agents with experience (columns 3-5), performance loss due to agent removal decreases until performance is as good or better than a swarm with $s = 1.0$. As s decreases, it takes longer to build up a sufficient backup pool to mitigate agent loss: swarms with $s = 0.7$ achieve a sufficient backup pool size by the second period (trials 125–129) while swarms with $s = 0.4$ do not achieve a sufficient backup pool until the fourth period (trials 375–499). Note that columns 2–5 each represents a period consisting of 125 trials while column 6 represents a period consisting of 500 trials. When no agent loss occurs, focusing task experience on the most experienced agents maximizes swarm performance, and there is no benefit to be gained from using $s < 1.0$. When agent loss may occur, swarms with lower s are better able to cope with and adapt to agent loss.

Table 2 Total amount of resource units retrieved by the swarm within each period of stable agent population size as per Fig. 8

Response probability s	Period (first trial–last trial)				
	0–124	125–249	250–374	375–499	500–999
0.10	338.17	393.83	397.19	408.12	1841.84
0.20	398.69	488.88	580.61	655.06	3284.75
0.30	423.75	539.72	642.25	690.50	3523.45
0.40	437.04	579.07	669.13	698.92	3643.27
0.50	456.01	607.97	674.15	699.22	3768.16
0.60	469.03	624.63	681.02	701.07	3869.81
0.70	487.03	642.21	685.81	707.81	3960.30
0.80	503.98	640.73	685.22	708.58	4035.44
0.90	521.99	641.31	680.19	698.66	4094.79
1.00	541.67	641.52	676.78	695.54	4183.42

These values are the average results averaged over 100 runs. The twenty most experienced agents in the swarm are removed at the start of trials 125, 250, 375, and 500

Table 3 Amount of resource retrieved by the swarm within each period of stable agent population size as a percentage of the amount of resource retrieved by the swarm when agents decisions are deterministic and response probability is $s = 1.0$

Response probability s	Period (first trial–last trial)				
	0–124	125–249	250–374	375–499	500–999
0.10	62.43	61.39	58.69	58.68	44.03
0.20	73.60	76.21	85.79	94.18	78.52
0.30	78.23	84.13	94.90	99.28	84.22
0.40	80.68	90.27	98.87	**100.49**	87.09
0.50	84.19	94.77	99.61	**100.53**	90.07
0.60	86.59	97.37	**100.63**	**100.80**	92.50
0.70	89.91	**100.11**	**101.33**	**101.76**	94.67
0.80	93.04	99.88	**101.25**	**101.87**	96.46
0.90	96.37	99.97	**100.50**	**100.45**	97.88
1.00	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**

Each percentage is calculated by dividing the corresponding value in Table 2 by the last (bottom) value in the corresponding column of Table 2. Bold values indicate periods in which performance meets or exceeds the performance of a swarm with $s = 1.0$

These results along with the theoretical analyses from the previous section support our hypothesis that, for problems where experience improves performance, a non-deterministic response probability can temper a threshold-based swarm's overall performance decline when the primary actors for a task become unavailable. In addition, applying a response probability to a swarm of agents with variable thresholds can allow the swarm to build up a back up pool of agents with experience on a task while focusing the effort on the most experienced agents. Our results support the following conclusions:

- There is a tradeoff between improving the swarm's performance and improving the swarm's robustness. Focusing opportunities to gain experience on the same agents allows those agents to improve in performance very quickly, but leaves the swarm vulnerable to drastic declines in performance if those agents are lost. Distributing opportunities to gain experience over many agents allows a swarm to be more resilient to loss of agents, but slows down the rate at which the swarm's performance improves. The response probability value s may be used to adjust this tradeoff so long as s is high enough to form complete teams.

- The s_1 value given by Theorem 1 is a generous lower bound estimate of a response probability value that ensures formation of complete teams. For this problem, the value of s that guarantees complete team formation, s_c, is likely to be higher than s_1.
- The value of s_1 can be theoretically calculated for a problem with a known n and x, but the value of s_c and the relationships between drop amount and s and drop level and s are problem dependent and would have to be determined empirically.

These conclusions lead to the following recommendations:

- If the speed at which the swarm system's performance improves is important, use the highest s that meets robustness requirements.
- To maximize system robustness, use the lowest s that allows successful team formation.
- The general recommendation is to use the highest s that meets the performance constraints on the drop level (minimum acceptable performance) and drop amount (minimum acceptable perturbation) values. With respect to Figs. 9 and 10, this means selecting the highest s value to the right of s_1 or s_c, depending on the importance of a complete team always being formed, that generates acceptable values for either or both drop level and drop amount.

5 Conclusions

In this paper, we investigate the hypothesis that, in a swarm where agents have differing thresholds for a task, response probability can be used to regulate how much dependence the swarm places on the lowest threshold agents for a task as well as the size of the pool of agents that gain experience on the task over multiple instances. We examine such a model for a single-task scenario both mathematically and empirically.

Mathematically, we create a model of such a system and show that we can estimate the response probability values that ensure that a complete team will be formed. In addition, over multiple trials, we can estimate the response probability values needed to meet a specified backup pool size, where the backup pool consists of agents with experience on the task.

Empirically, we test this hypothesis on a decentralized foraging problem in which agents with differing thresholds gain experience over multiple trials as to where to find resources to forage. We periodically remove the most experienced (lowest threshold) agents from the swarm and measure the change in performance and time to recover. Our results indicate that, as expected, systems with a non-deterministic response probability are better able to withstand loss of agents than systems with a deterministic response (response probability of 1.0), due to more agents having gained experience on the task.

Higher response probability values concentrate learning experiences among just the lowest threshold agents. This concentration results in a faster increase in system performance but larger performance drops after removal of the lowest threshold agents. Lower response probability values distribute learning experiences among more agents. This distribution results in a slower increase in system performance but smaller performance drops after the removal of agents. As response probability decreases from 1.0 to 0.05, the level to which performance drops increases until the response probability is low enough that a complete team cannot be formed, at which point it decreases toward zero.

In summary, we show that, for swarms in which agents have differing threshold values, a response probability provides a way to balance maximizing system performance and distributing opportunities to gain experience. A non-deterministic response probability allows a decentralized swarm to focus most of the effort on the most experienced agents while at the same time generating a backup pool of additional agents with experience on a given task. This emergent backup pool can mitigate performance drop if the primary agents are lost or become unavailable. There is a tradeoff between improving the swarm's performance and improving the swarm's robustness. For a given swarm and task demand, we can estimate mathematically the response probability values that ensure team formation and meet a specified backup pool size. Empirical results indicate that, in general, one should consider using the highest response probability value that meets task performance constraints.

Acknowledgements This work was supported in part by NSF Grant IIS1816777 and ONR Grant N000140911043.

References

Agassounon, W., Martinoli, A., & Goodman, R. (2001). A scalable distributed algorithm for allocating workers in embedded systems. In *Proceedings of the IEEE international conference on systems, man, and cybernetics* (pp. 3367–3373).

Ashby, W. R. (1958). Requisite variety and its implications for the control of complex systems. *Cybernetica*, *1*(2), 83–99.

Bonabeau, E., Theraulaz, G., & Deneubourg, J. (1996). Quantitative study of the fixed response threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London B*, *263*, 1565–1569.

Bonabeau, E., Theraulaz, G., & Deneubourg, J. (1998). Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, *60*, 753–807.

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, *7*, 1–41.

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., & Dorigo, M. (2014). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, *28*(1), 101–125.

Brutschy, A., Tran, N.-L., Baiboun, N., Frison, M., Pini, G., Roli, A., et al. (2012). Costs and benefits of behavioural specialization. *Robotics and Autonomous Systems*, *60*(11), 1408–1420.

Campos, M., Bonabeau, E., Theraulaz, G., & Deneubourg, J. (2000). Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, *8*(2), 83–96.

Castello, E., Yamamoto, T., Dalla Libera, F. D., Liu, W., Winfield, A. F. T., Nakamura, Y., & H. Ishiguro (2016). Adaptive foraging for simulated and real robotic swarms: The dynamical response threshold approach. *Swarm Intelligence*, *10*(1), 1–31.

Castello, E., Yamamoto, T., Nakamua, Y., & Ishiguro, H. (2013). Task allocation for a robotic swarm based on an adaptive response threshold model. In *Proceedings of the 13th IEEE international conference on control, automation, and systems* (pp. 259–266). IEEE.

Cicirello, V. A., & Smith, S. F. (2002). Distributed coordination of resources via wasp-like agents. In *Workshop on Radical Agent Concepts, LNAI* (Vol. 2564, pp. 71–80). Springer.

Correll, N. (2008). Parameter estimation and optimal control of swarm-robotic systems: A case study in distributed task allocation. *Proceedings of the IEEE international conference on robotics and automation* (pp. 3302–3307). IEEE.

de Lope, J., Maravall, D., & Quinoñez, Y. (2013). Response threshold models and stochastic learning automata for self-coordination of heterogeneous multi-task distribution in multi-robot systems. *Robotics and Autonomous Systems*, *61*, 714–720.

dos Santos, F., & Bazzan, A. L. C. (2009). An ant based algorithm for task allocation in large-scale and dynamic multiagent scenarios. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (pp. 73–80).

Goldingay, H., & van Mourik, J. (2013). The effect of load on agent-based algorithms for distributed task allocation. *Information Sciences*, *222*, 66–80.

Goss, S., Aron, S., Deneubourg, J. L., & Pasteels, J. M. (1989). Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, *76*, 579–581.

Goss, S., Beckers, R., Deneubourg, J. L., Aron, S., & Pasteels, J. M. (1990). How trail laying and trail following can solve foraging problems for ant colonies. In R. N. Hughes (Ed.), *Behavioural mechanisms of food selection* (Vol. 20, pp. 661–678). Berlin: Springer.

Kanakia, A., Touri, B., & Correll, N. (2016). Modeling multi-robot task allocation with limited information as global game. *Swarm Intelligence*, *10*(2), 147–160.

Kazakova, V. A. & Wu, A. S. (2018). Specialization vs re-specialization: Effects of Hebbian learning in a dynamic environment. In *Proceedings of the 31st international Florida artificial intelligence research society* (pp. 354–359). AAAI.

Kittithreerapronchai, O., & Anderson, C. (2003). Do ants paint trucks better than chickens? Market versus response threshold for distributed dynamic scheduling. In *Proceedings of the 2003 Congress on Evolutionary Computation* (pp. 1431–1439). IEEE.

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

Krieger, M. J. B., Billeter, J., & Keller, L. (2000). Ant-like task allocation and recruitment in cooperative robots. *Nature*, *406*, 992–995.

Labella, T. H., Dorigo, M., & Deneubourg, J. (2006). Division of labor in a group of robotics inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems*, *1*(1), 4–25.

Lee, W., & Kim, D. (2017). History-based response threshold model for division of labor in multi-agent systems. *Sensors*, *17*(6), 1232.

Lerman, K., Jones, C., Galstyan, A., & Mataric, M. (2006). Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, *25*, 225–241.

Merkle, D., & Middendorf, M. (2004). Dynamic polytheism and competition for tasks in threshold reinforcement models of social insects. *Adaptive Behavior*, *12*(3–4), 251–262.

Motwani, R., & Raghavan, P. (1995). *Randomized algorithms*. Cambridge: Cambridge University Press.

Niccolini, M., Innocenti, M., & Pollini, L. (2010). Multiple UAV task assignment using descriptor functions. In *Proceedings of the 18th IFAC symposium on automatic control in aerospace* (pp. 93–98).

Nouyan, S. (2002). Agent-based approach to dynamic task allocation. In *ANTS 2002, Third International Workshop on Ant Algorithms* (pp. 28–39). Springer.

Nouyan, S., Ghizzioli, R., Birattari, M., & Dorigo, M. (2005). An insect-based algorithm for the dynamic task allocation problem. *Kunstliche Intelligenz*, *4*(5), 25–31.

Pang, B., Zhang, C., Song, Y., & Wang, H. (2017). Self-organized task allocation in swarm robotics foraging based on dynimcal response threshold approach. In *Proceedings of the 18th international conference on advanced robotics* (pp. 256–261). IEEE.

Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., & Birattari, M. (2011). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, *5*(3–4), 283–304.

Pini, G., Brutschy, A., Pinciroli, C., Dorigo, M., & Birattari, M. (2013). Autonomous task partitioning in robot foraging: An approach based on cost estimation. *Adaptive Behavior*, *21*(2), 118–136.

Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., & Birattari, M. (2014). Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer. *Artificial Life*, *20*(3), 291–317.

Pini, G., Gagliolo, M., Brutschy, A., Dorigo, M., & Birattari, M. (2013). Task partitioning in a robot swarm: A study on the effect of communication. *Swarm Intelligence*, *7*(2–3), 173–199.

Pradhan, R., & Wu, A. S. (2011). On the relationship between response probability and redundancy in teams of collaborating agents. In *Proceedings of the 5th international conference on collaborative computing: networking, applications, and worksharing*. IEEE.

Price, R., & Tino, P. (2004). Evaluation of adaptive nature inspired task allocation against alternate decentralised multiagent strategies. In *Proceedings of the international conference on parallel problem solving from nature, LNCS* (Vol. 3242, pp. 982–990). Springer.

Scerri, P., Farinelli, A., Okamoto, S., & Tambe, M. (2005). Allocating tasks in extreme teams. In *Proceedings of the international conference on autonomous agents and multi-agent systems* (pp. 727–734). ACM.

Weidenmüller, A. (2004). The control of nest climate in bumblebee (*bombus terrestris*) colonies: Interindividual variability and self reinforcement in fanning response. *Behavioral Ecology*, *15*(1), 120–128.

Winfield, A. F . T. (2013). Foraging robots. In R . A. Meyers (Ed.), *Encyclopedia of complexity and systems science* (pp. 3682–3700). Berlin: Springer.

Wu, A. S., & Kazakova, V. A. (2017). Effects of task consideration order on decentralized task allocation using time-variant response thresholds. In *Proceedings of the 30th international Florida artificial intelligence research society* (pp. 466–471). AAAI.

Wu, A. S., & Riggs, C. (2018). Inter-agent variation improves dynamic decentralized task allocation. In *Proceedings of the 31st international Florida artificial intelligence research society* (pp. 366–369). AAAI.

Wu, A. S., Wiegand, R. P., & Pradhan, R. (2016). Building redundancy in multi-agent systems using probabilistic action. In *Proceedings of the 29th international Florida artificial intelligence research society conference* (pp. 404–409). AAAI.

Wu, A. S., Wiegand, R. P., Pradhan, R., & Anil, G. (2012). The effects of inter-agent variation on developing stable and robust teams. In *Proceedings of the AAAI spring symposium*. AAAI.

Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the 2nd international conference on autonomous agents* (pp. 47–53). ACM.

Yang, Y., Chen, X., & Li, Q. (2010). Swarm robots task allocation based on local communication. In *Proceedings of the international conference on computer, mechatronics, control, and electronic engineering* (pp. 415–418). IEEE.