Broadening Applicability of Swarm-Robotic Foraging Through Constraint Relaxation

John Harwell[1] and Maria Gini[2]

Abstract— **Swarm robotics (SR) offers promising solutions to real-world problems that can be modeled as foraging tasks, e.g. disaster/trash cleanup or object gathering for construction. Yet current SR foraging approaches make limiting assumptions that restrict their applicability to selected real-world environments. We propose an improved self-organized task allocation method based on task partitioning that removes restrictions such as: (1) *a priori* knowledge of object/cache location, and (2) strict limitations on intermediate cache behavior. We show with experiments in simulation that improved stochastic decision making at the level of each individual robot in the swarm leads to overall performance that for some combinations of robot capabilities/relaxed environmental conditions meets or exceeds that of previous works in this area.**

I. Introduction

Swarm robotics (SR) is the study of large-scale robotic systems consisting of either homogeneous or heterogeneous robots. SR derives many of its core principles from natural systems such as bees, ants, and termites [23]:

- *Decentralized control*: There is no centralized controller/authority to which all agents are subservient. This greatly reduces, and in many cases eliminates, the possibility of catastrophic system failure.
- *Autonomy*: Agents are autonomous, acting without the intervention of higher powers (overseers, queens, etc).
- *Localized sensing and communication*: There is no mechanism for broadcasting information to the entire system; instead all decisions are made by individual agents based on locally available information from its neighbors as well as its own limited sensor data.
- *Emergent behavior*: The previous characteristics cumulatively give rise to collective behaviors of the entire swarm that cannot be easily predicted from that of individual agents, i.e. the whole is greater than the sum of its parts. Studying emergent collective behavioral patterns through post-mortem/predictive modeling and analysis, instead of the characteristics of individual agents, distinguishes SR research distinct from distributed and/or autonomous robotic systems [16].

Due to the duality between SR and natural systems, SR researchers are able to draw effective parallels between naturally occurring problems such as foraging, collective transport of heavy objects, self-assembly, exploration, and collective decision making [2], [1], and similar real-world applications, such as pursuit-evasion and autonomous construction. Many of the leveraged solutions obtained by social insects employ *task partitioning*, i.e. dividing a large task into simpler subtasks that can be tackled by different agents/workers [21]. It has been shown that task partitioning in both natural and SR systems has many well-known benefits, such as (1) increased performance at group level; (2) stimulated specialization; (3) parallel task execution; (4) reduced interference between individuals; (5) improved exploitation of the environment; (6) improved transport efficiency [10], [19]. These benefits, coupled with the SR properties described above, make SR systems ideal for tackling many problems in dangerous and/or unstable environments, such as search-and-rescue, disaster/environmental cleanup, and space exploration/surveying [22].

A. Motivation

In this paper we focus on a foraging task, where a group of robots has to collect and transport material from one or more sources, and bring the collected material to a known nest location. Intermediate locations called *caches* can be utilized or ignored by robots, depending on what foraging strategy they are employing, and serve as temporary storage sites where materials can be dropped and picked up asynchronously. Any robust robotic foraging approach should be capable of creating and/or utilizing caches when they provide efficiency gains, and of not doing so when their use proves costly. Furthermore, such methods should be capable of strategic cache utilization even under environmental conditions that are not specifically contrived to incentivize cache usage.

The proposed method is based on local perception, and requires no explicit communication between robots. A given robot stochastically decides what type of foraging strategy to employ, as well as what subtask to execute (if applicable), based on its local estimates of the global execution time of tasks associated with each strategy, which are updated upon task completion/abortion. Through these individual decisions the robot swarm can robustly adapt its collective foraging strategy, employing task partitioning to divide the foraging task into interdependent subtasks, or to employ a non-partitioning strategy and tackle the foraging task holistically. Task partitioning (and cache utilization) only occurs when it is advantageous. This approach enables the elimination of the following constraints/assumptions of previous work regarding task partitioning and cache usage/behavior:

[1]Department of Computer Science and Engineering, University of Minnesota. harwe006@umn.edu

[2]Department of Computer Science and Engineering, University of Minnesota. gini@umn.edu

Preprint submitted to 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots . Received January 29, 2018.

- All robots have *a priori* knowledge of object/cache locations [4], [19], [8], [9]. This does not model partially observable/unstable environments, which are some of the most prominent applications for SR systems.
- All robots that cross over a cache area must pay a usage penalty[4], even if they do not utilize it. This restriction models an area of rough terrain that requires extra time to navigate safely, for which task partitioning is clearly advantageous.
- All robots executing the unpartitioned task must use a very long corridor to travel between the source and the nest [19], [9]. This restriction models environments in which debris/obstacles block the direct path between the source and the nest, and it is therefore slower to execute the unpartitioned task in general.

We show in simulation experiments that even with the relaxation of the above constraints that there are still many combinations of swarm sizes, robot capabilities, and environmental conditions for which employing task partitioning (and therefore utilizing an intermediate cache) still provides performance increases when compared to an unpartitioned strategy. This work moves SR foraging approaches closer to a more broadly applicable real-world model, showing them to be effective even under ideal conditions while continuing to perform robustly in conditions simulating more volatile environments.

II. Related Work

In most prior work, *a priori* task partitioning focuses on the problem of allocating individuals to subtasks in a way that maximizes efficiency once the optimal distribution of tasks to robots is known [5], [3], [14], [13]. Auction-based approaches, in which robots place competing bids for tasks, have been successfully applied to the assignment of agents. However, such approaches rely on: regular communication of cost functions, *a priori* information about execution times, and centralized control. In unstable environments or those with large numbers of unreliable agents, such as the ones studied in this work, such requirements are infeasible, as they do not scale well with the number of robots or tasks [13]. Correll et al. have shown that SR systems are competitive with deterministic approaches to task allocation when communication is tightly constrained or in partially observable environments; in addition, they scale well and are robust to multiple failures [5].

Matthey et al. modeled the interactions of agents and their environment as chemical reactions between molecules [14], and based their controller synthesis methodology on Markov Decision Processes. By modeling swarm behavior using the Chemical Network Reaction (CRN) framework, the reaction (i.e. task completion) rates can be tuned so that some tasks can be prioritized over others. This paradigm, while solving issues of scalability and robustness, still requires *a priori* knowledge of the workload for task allocation.

In many cases, task partitioning cannot be done *a priori* either because complete information on the environment is not available, or the environment itself is unstable and hence optimal *a priori* allocations cannot be achieved. In such situations, *self-organized task partitioning* is a suitable approach. Most prior work involving self-organized task allocation or partitioning rarely considered problems that have interdependencies, much less sequential interdependencies (with notable exceptions [19], [4], [8], [9]), and assumed independent tasks not affected by group dynamics [9], [6]. Division of labor via task partitioning may have associated costs, such as task switching delays or work transfer between subtasks. Nevertheless, it is advantageous in many situations, such as those in which environmental factors result in a lower cost of performing subtasks vs. the unpartitioned task, or when reduction in inter-robot interference due to spatial locality of subtasks compared to the unpartitioned task reduces average task execution time.

In the context of a foraging task, the choice of employing task partitioning is equivalent to deciding the object transfer method. If partitioning is not employed, then no transfer occurs, and a single robot carries the object from the source to the nest. If partitioning is employed, then object transfer can be direct (robotic object handoff) [4] or indirect [9], [19], and use caches as asynchronous pickup/drop points. This asynchronous transfer can be beneficial because it can reduce material losses due to imbalances between foraging and processing rates [11]. It can also serve as a means of traffic control, regulating congestion by enabling spatially disjoint task execution.

The rest of the paper is organized as follows. In Section III we describe the problem and detail the proposed method. In Section IV we describe the experimental framework used. In Section VI we report and discuss the results. Finally, in Section VII we summarize our contribution and present directions for future research.

III. Problem Statement and Proposed Method

A. Problem Statement

A *task* T is a unit of work that can be completed by a *task sequence* composed of one or more atomic and/or partitionable tasks. *Atomic tasks* are non-divisible tasks that can be executed by a robot k. A *partitionable task* can either be executed holistically by a robot k as an *unpartitioned task*, or subdivided into disjoint atomic tasks. A *Binary Partitionable Task* T can be partitioned into exactly two disjoint halves, τ_1 and τ_2. Subtasks τ_1 and τ_2 of parent task T are sequentially interdependent if one subtask must be completed before the other starts, represented by $\tau_1 \succ \tau_2$. T is completed iff both τ_1 and τ_2 have been completed in order.

We refer to such subtasks as being adjacent, and to share a *task interface* Π, where robots working on different subtasks interact. These interactions can include the exchange of task-related information (via direct communication or stygmergically) and the direct transfer of objects between agents [4]. A robot k executing a given subtask has two opportunities to change its partitioning strategy: at a task interface Π_i (possibly giving up the task due to waiting for a dependent task to finish), and at task completion. Not all tasks have task interfaces; robots executing tasks that do not can only change their partitioning strategy upon task completion (such tasks therefore cannot be aborted). The *task set* Φ for a given partitioning strategy is the total set of tasks available for a robot k to execute. This may be equivalent to the task sequence ϕ corresponding to the robot's chosen partitioning. In this work, the statement $\phi \subseteq \Phi$ always holds true. Finally, the *active task* is the task a robot k is currently executing.

All robots that complete a task τ_i that ends at a task interface Π_i must pay an associated *interface cost* π_i [4], [19], which is usually modeled as a time cost. Common examples of interface costs in real systems are costs due to tool changes, spatially dispersed subtasks, inherent costs due to specialization (or lack thereof) in a subtask [4], or object transfer/manipulation costs. For Binary Partitionable Tasks, each subtask will have exactly one task interface, which is the point in the parent task at which it was partitioned. For example, if this point involved picking up from/dropping a block into a cache, then the task interface cost would be the cost of doing so.

B. The Proposed Method

1) Depth 0 Partitioning Strategy:

$$\Phi_0 = \phi_0 = \{\tau_{generalist}\} \tag{1}$$

The simplest possible task set Φ_0 (Eqn. *1*) for a foraging task contains a single atomic task in which executing robots will look for a block (by some strategy), and then bring the located block back to the nest. This strategy has no task interface. We implement two variants of this model: one in which robots remember blocks that they have seen as they move around, which we term the *Stateful Generalist* strategy, and one in which they do not, which we term the *Stateless Generalist*. We use these simple strategies as a baseline to compare our proposed method against.

In the Stateful Generalist strategy, robots employ virtual pheromones [15], [12] to track the relevance of their environmental information. The pheromones of different robots do not influence each other, commensurate to the strictly stymergic interactions, such as collisions and competition over object selection, that are common in SR. The pheromone level of different objects seen in the environment is stored as a probabilistic occupancy grid. Every timestep when a robot sees an object at location (i, j) within the arena, it deposits a unit quantity of pheromone on the corresponding location within its occupancy grid; every timestep that a robot does not see an object, the pheromone (and object relevance) decays (Eqn. *2*). ρ $(0 < \rho < 1)$ is the pheromone decay parameter that controls the rate of decay.

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

After depositing a block in the nest, a robot k will decide what block to acquire next by looking at its occupancy

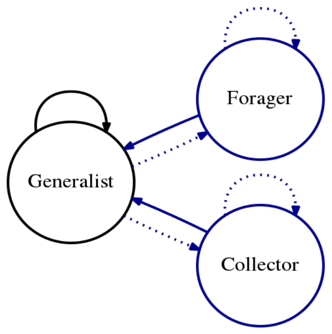

Fig. 1: A depth 1 task set definition. Dotted lines indicate transitions in which partitioning was employed, and solid lines indicate transitions in which it was not. Transitions from children to parent tasks due to task abortion are omitted for clarity. Depth 0 definitions are in black, depth 1 are in blue. Each robot can choose to partition a task T, with its chosen depth inducing the task set ϕ from which it can select its active task.

grid, and choosing the block with the highest utility $\mu_b^k(t)$ (Eqn. *3*). This block utility definition encourages gathering blocks that are further from the nest before those that are closer, while also accounting for the nearness of a block to the current location of robot k, as well as its perceived relevance. In the degenerate case in which a robot does not know of any blocks, a Stateful Generalist devolves into a Stateless Generalist.

$$\mu_b^k(t) = \frac{\left\|k_{ij}(t) + nest_{ij}\right\|}{\left\|k_{ij}(t) + b_{ij}\right\|} e^{\tau_{b_{ij}}^{k}(t)} \tag{3}$$

2) Depth 1 Partitioning Strategy:

Here, we formalize the more robust foraging model first presented by [8], [4], [19], which is depicted in Fig. *1*. This strategy's task set Φ_1 is:

$$\Phi_1 = \{\tau_{harvest}, \tau_{collect}\}$$
$$\phi_1 = \tau_{harvest} \succ \tau_{collect}$$

- *Harvester*: Acquire a free block (one not in a cache) and bring it to a cache (a static cache is assumed, similar to previous work [19], [4], [18], [9]). The task interface $\Pi_{harvest}$ is the time spent acquiring a cache for block drop after a block has been picked up.
- *Collector*: Acquire a block from a cache and bring it to the nest. The task interface $\Pi_{harvest}$ is defined as the portion of the task before a block is acquired in which the robot looks for a cache.

$$\pi_i = \pi_c + \pi_d \tag{4}$$

The execution of ϕ_1 (possibly by different robots) corresponds to a single execution of the overall task T. Robots engaged in either of the tasks in Φ_1 will incur a flat cache usage cost (π_c) in Eqn. *(4)* due to block drop/pickup, in addition to delay due to group dynamics such as congestion/collision avoidance at the cache, or out-of-date information about arena contents (π_d). π_i contributes to the robot's overall estimation of $\hat{\tau_i}$ (Eqn. *5*), as well as its abort probability (Eqn. *11*).

The execution time of a task τ_i after j times is estimated using an exponential moving average:

$$t_{\hat{\tau_{j+1}}} = (1-\alpha) t_{\hat{\tau_{j-1}}} + \alpha t_j \tag{5}$$

where α is the exponential weight factor and t_j is how long it took to execute task on the j-th iteration.

When a robot finishes a non-partitionable task (Harvester, Collector), then the parent of its current task (in this case, a Stateful Generalist) is used for the partitioning decision. The probability P_p of employing partitioning after task τ_i has been completed (or aborted) is calculated as [19]:

$$\theta_p = \begin{cases} \Omega_{pr}(\frac{t_{\hat{\phi_{NP}}}}{t_{\hat{\phi_1}} + t_{\hat{\phi_2}}} - \Omega_{po}) & \text{if } t_{\hat{\phi_{NP}}} > (t_{\hat{\phi_1}} + t_{\hat{\phi_2}}) \\ \Omega_{pr}(1 - \frac{(t_{\hat{\phi_1}} + t_{\hat{\phi_2}})}{t_{\hat{\phi_{NP}}}}) & \text{else} \end{cases} \tag{6}$$

$$P_p = (1 + e^{-\theta_p})^{-1} \tag{7}$$

where $t_{\hat{\tau_{NP}}}$ is an estimate (Eqn. *5*) of the average time $t_{\tau_{NP}}$ required to complete the unpartitioned task (i.e. the cost of the "no partition" strategy) and $t_{\hat{\tau_1}}$ and $t_{\hat{\tau_2}}$ are estimates of times t_{τ_1} and t_{τ_2} required to perform each of the two sub-tasks (i.e. the cost of the "partition" strategy). Ω_{pr} is a steepness factor that influences the probability associated with partitioning; larger values cause small differences between the two estimates to increase/decrease the probability more quickly (i.e. robots are more reactive to environmental changes). Ω_{po} is an offset factor used to determine how much the difference between the cost of the partition/no partition strategies will be allowed to grow before the probability of employing partitioning begins to grow exponentially. A piecewise definition ensures that as the difference between the cost of the partition/no partition strategy estimates grows (in either direction), $|P_p| < 1$.

If a robot chooses to employ task partitioning, it must choose what subtask to execute. If the last executed task was non-partitionable, then employing partitioning just means repeating the same task (see Fig. *1*). If the last executed task τ_i was partitionable then the next task is chosen from subtasks $\{\tau_{i1}, \tau_{i2}\}$ as follows, where τ_{i1} was the most recently executed subtask of τ_i (if neither subtask has been executed before, random selection is employed):

$$P_{ij} = \frac{1}{1 + e^{-\Theta_{ss}}} \gamma \tag{8}$$

$$\Theta_{ss} = \frac{1}{K}\Big(\frac{\hat{\tau_{i2}}}{r(\hat{\tau_{i1}}, \hat{\tau_{i2}})} - M\Big) \tag{9}$$

$$r(\hat{\tau_{i1}}, \hat{\tau_{i2}}) = \begin{cases} \frac{\hat{\tau_{i2}}^2}{\hat{\tau_{i1}}} & \text{if } \hat{\tau_{i1}} > \hat{\tau_{i2}} \\ \\ \hat{\tau_{i2}} & \text{else} \end{cases} \tag{10}$$

This method is similar to [4], with the important difference that the time estimates are for overall *execution* time of a task, rather than *interface* time. This means that robots should always choose the subtask with a *lower* time estimate, rather than the *higher* time estimate implied by the method developed by Brutschy et al.

Finally, the probability of aborting a task τ_i at Π_i is:

$$P_a(\tau_i) = (1 + e^{\theta_a})^{-1} \tag{11}$$

$$\theta_a = \begin{cases} \Omega_{ar}(\Omega_{oa} - \frac{\delta_i}{t_{\hat{\Pi_i}}}) & \text{if } \frac{\delta_i}{t_{\hat{\Pi_i}}} \leq \Omega_{ao} \\ \Omega_{ar}(\frac{\delta_i}{t_{\hat{\tau_i}}} - \Omega_{ao}) & \text{else} \end{cases} \tag{12}$$

where δ_i is the incremental interface wait time for a task τ_i, and Ω_{ao} is an offset factor that determines how far beyond the estimate of the interface time $\hat{\Pi_i}$ (calculated using Eqn. *5*) a robot will wait before the abort probability begins to grow rapidly, and Ω_{ar} defines the rate of growth after this threshold is exceeded. This differs from the definition in [19], as it is driven by deviation from the estimated task *execution* time, rather than deviation from estimated task *interface* time.

In the model described here, robots stochastically choose a partitioning strategy for their next task allocation by calculating P_p (Eqn. *7*) after finishing/aborting a task, and then possibly use Eqn. *(8)* to allocate themselves a subtask. If applicable, they use Eqn. *(11)* at the task interface to determine whether their active task should be aborted.

IV. Experimental Framework

The experiments described in this paper were carried out in the ARGoS [17] simulator, which allows real-time simulation of large swarms of robots. For this work, we chose a dynamic model of robots in a three-dimensional space, using a model of an s-bot, a real robotic platform developed in the Swarm-bots project [7].

All experiments carried out in this work assume:

- Homogeneous robots.
- Robots cannot communicate.
- Robots have infinite battery life.
- Robots can self localize in relation to a light source which they know is located directly above the nest.
- Robots initially know arena size, but not its contents.
- Robots cannot directly transfer objects to other robots.
- The environment is flat, open, and without obstacles (in contrast to [19], [20], [9], [8]).
- Robots start randomly distributed in the environment.
- The nest and cache have unlimited storage capacity.
- All blocks to be transported to the nest are clustered in a single source (See Fig. *2*).
- Robots move more slowly when carrying a block.
- Robots are only penalized if they utilize a cache (relaxing the condition present in [4]).

As depicted in Fig. *2*, abstracted blocks in simulation are represented as black squares detectable by the s-bot's ground sensors. Caches are abstracted similarly, with each cache

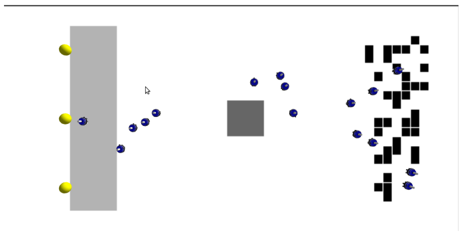

Fig. 2: Foraging scenario where all objects are clustered at one end of the arena (single source foraging), with a cache in the middle.

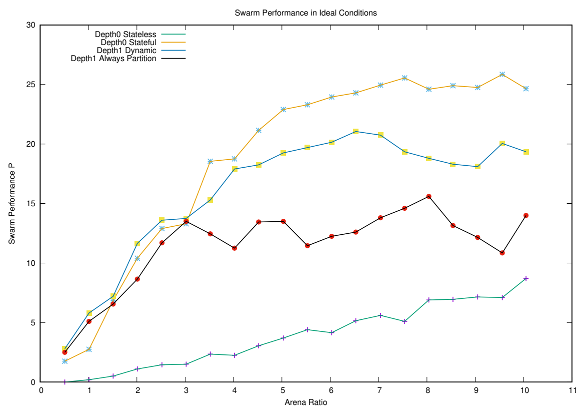

Fig. 3: Swarm performance across all controllers in ideal environmental conditions (π_c = 100, throttle 10%). Peak performance is achieved near the same arena ratio as [4] (5%).

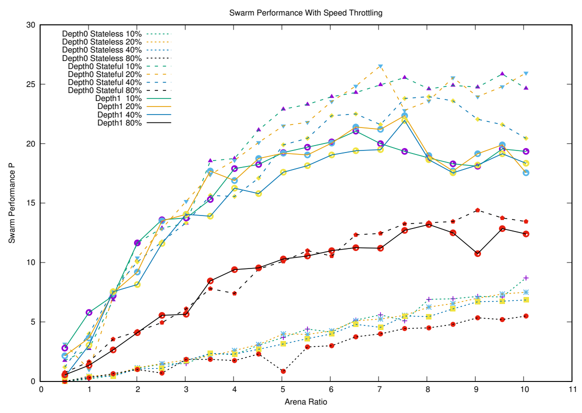

Fig. 4: Swarm performance by all controllers with π_c = 10, and varying the throttle applied on block carry. The adaptive task partitioning model tracks the simpler Stateful Generalist model very well for lower swarm densities, but breaks down due to the large numbers of robots overwhelming the cache's ability to regulate traffic.

maintaining a list of which blocks it currently contains, and appearing only as a medium gray square in the arena. The nest appears as a light gray. Whether or not a robot carries a block is also stored internally in the simulator, simplifying the complex physics involved in picking up/dropping blocks.

V. Experiments

All controllers (Depth 0 Stateless/Stateful Generalist, Depth 1) were tested on the scenario in Fig. 2, with swarm size $S = [4 \ldots 80]$, step size of 4. An "always partition" variant of the Depth 1 controller was also tested, in order to provide one of the boundary performance cases for the proposed method; the other "never partition" approach is the same as a Stateful Generalist. An arena of size 20m x 5m was used, and robot speed was set to a maximum of 5 cm/s. This low speed is due to the limited sensing range of the s-bot, and gives robots more time to gracefully maneuver around other robots without triggering obstacle avoidance. To avoid bias in the robot's behavior, $\tau_{\hat{harvest}}$, $\tau_{\hat{collect}}$, and $\tau_{\hat{NP}}$ are randomly initialized: $\tau_{\hat{harvest}}$ and $\tau_{\hat{collect}}$ are uniformly sampled in [1000, 2000], $\tau_{\hat{NP}}$ in [2000, 4000].

We tested combinations of environmental conditions and robot capabilities in three different sets of experiments. In the first, environmental conditions were ideal, with π_c=100, throttle=10%. π_c cannot realistically be 0 even under ideal conditions; to pickup/drop an object always requires a non-zero time interval. A similar argument can be made for non-zero throttle due to block carry. In the second set, π_c=10 was held constant, and the speed throttling on block carry was varied: {10%, 20%, 40%, 80%}, which explores variation in robot strength vs. object weight during the application of the proposed method. In the third set, the speed throttling on block carry was held constant at 10% and the cache usage cost was varied π_c = {100, 200, 400, 800}, which explores the performance of the proposed method as the cache becomes more costly to use (terrain variations, irregular objects, etc.).

VI. Results

For each experiment set described above, we calculate the total number of robots engaged in each strategy and in each task every Δt = 200 seconds. We use swarm performance P, defined as the number of blocks collected every Δt seconds [4], to evaluate the quality of the self-organized task allocation strategy deployed by the swarm overall. We use the *arena ratio* (ratio of robots to arena size [4], [19]) to compare swarm performance across different scenarios.

Results in Fig. 3 indicate that the while it is advantageous for agents to have pheromone-based memory (i.e. Stateful Generalists far outperform Stateless Generalists), it is not always advantageous for them to employ task partitioning under ideal conditions. For lower swarm densities (< 3), the cache's ability to regulate traffic patterns within the arena increases performance by restricting Collector/Harvester robots to spatially disjoint portions of the arena, thereby reducing overall congestion. For higher swarm densities, it is not advantageous, likely due to the number of robots overwhelming the cache, which is relatively small. We observe that while the Stateless Generalist is the lowest performing foraging approach, it is also the most scalable, and does not appear to approach an asymptotic limit as the arena ratio increases. Intuitively, as swarm size increases, so does congestion, leading to randomized motion as the most efficient way to navigate.

In the second set of experiments (Fig. 4), for swarm

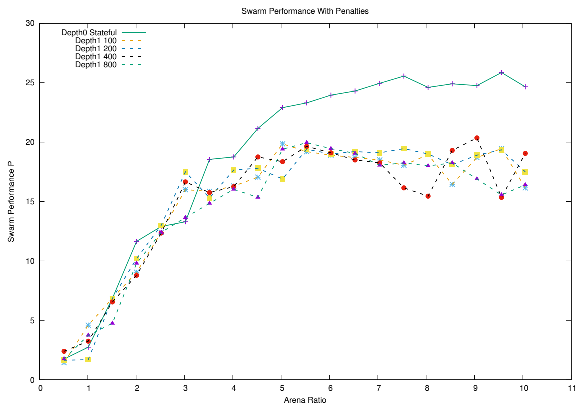

Fig. 5: Swarm performance of the Depth 1 controller, compared against the Stateful Generalist. Throttle = 10%, and the cache usage cost π_c is varied. Increasing usage penalties helps to further regulate traffic at lower densities, but also reduces performance by increasing the duration of a block's trip from source to nest, and degrades performance at higher densities.

densities < 5%, there are some points in the parameter space for which adaptive task partitioning is advantageous, and some for which it is not. Beyond this threshold, it appears to be advantageous to employ a strictly unpartitioned strategy. Finally, in the third set of experiments (Fig. 5), we see that for arena ratios of ~3%, the usage of a task partitioning strategy is much more advantageous than an unpartitioned strategy, even under constraint relaxation.

VII. Conclusions and Future Work

We have presented an adaptive, self-organized task partitioning approach to robotic foraging, and shown that even in ideal conditions it can still be advantageous to employ a partitioning strategy instead of an unpartitioned one. Under non-ideal conditions, we have relaxed some of environmental constraints present in previous work, and shown that there are numerous combinations of robot capabilities, swarm size, and cache behavior for which employing partitioning is still advantageous.

One direction for future work is to extend the task partitioning method "recursively", in which the Collector and Harvester sub-tasks are further partitioned. This extension would enable dynamic cache creation in the arena, thereby increasing swarm performance in scenarios where objects are randomly scattered around the arena, rather than clustered together. Another avenue to is to consider per-robot estimation of current swarm density, and to incorporate that into the strategy selection, as different strategies are more effective at different densities.

In order to facilitate future research and collaboration, the code for this work is open source, and can be found at `https://github.com/swarm-robotics/fordyca.git`.

References

[1] Levent BAYINDIR and Erol SAHIN. A Review of Studies in Swarm Robotics . *Turk J Elec Engin*, 15(2):115–147, 2007.

[2] Gerardo Beni. From swarm intelligence to swarm robotics. In Erol Sahin and William Spears, editors, *Swarm Robotics*. 2004.

[3] Spring Berman, Ádám Halász, M. Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, aug 2009.

[4] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 28(1):101–125, 2014.

[5] Nikolaus Correll. Parameter estimation and optimal control of swarm-robotic systems: A case study in distributed task allocation. In *IEEE International Conference on Robotics and Automation*, pages 3302–3307. IEEE, may 2008.

[6] Torbjørn S. Dahl, Maja Matarić, and Gaurav S. Sukhatme. Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(6-7):674–687, 2009.

[7] Marco Dorigo. Swarm-bot: An experiment in swarm robotics. *Proc. IEEE Swarm Intelligence Symposium*, 2005:199–207, 2005.

[8] Eliseo Ferrante, Ali Emre Turgut, Edgar Duéñez-Guzmán, Marco Dorigo, and Tom Wenseleers. Evolution of Self-Organized Task Specialization in Robot Swarms. *PLoS Computational Biology*, 11(8):1–21, 2015.

[9] Marco Frison, Nam Luc Tran, Nadir Baiboun, Arne Brutschy, Giovanni Pini, Andrea Roli, Marco Dorigo, and Mauro Birattari. Self-organized Task Partitioning in a Swarm of Robots. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6234 LNCS, pages 287–298, 2010.

[10] Adam G. Hart, Carl Anderson, and Francis L.W. Ratnieks. Task partitioning in leafcutting ants. *Acta Ethologica*, 2002.

[11] Adam G Hart and Francis L W Ratnieks. Leaf caching in Atta leafcutting ants: discrete cache formation through positive feedback. *Animal Bahavior*, 59:587–591, 2000.

[12] Joshua P. Hecker and Melanie E. Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, 2015.

[13] M. Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, 2(2-4):121–141, 2008.

[14] Löic Matthey, Spring Berman, and Vijay Kumar. Stochastic strategies for a swarm robotic assembly system. *Proc. IEEE International Conference on Robotics and Automation*, pages 1953–1958, 2009.

[15] Yan Meng and Jing Gan. A distributed swarm intelligence based algorithm for a cooperative multi-robot construction task. *IEEE Swarm Intelligence Symposium*, 2008.

[16] Kazuhiro Ohkura, Toshiyuki Yasuda, and Yoshiyuki Matsumura. Coordinating adaptive behavior for swarm robotics based on topology and weight evolving artificial neural networks. *Trans. of the Japan Society of Mechanical Engineers Series C*, 77(775):966–979, 2011.

[17] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Timothy S. Stirling, Álvaro Gutiérrez, Luca Maria Gambardella, and Marco Dorigo. ARGoS: A pluggable, multi-physics engine simulator for heterogeneous swarm robotics. *IRIDIA – Technical Report Series*, (December 2010), 2011.

[18] G. Pini, a. Brutschy, C. Pinciroli, M. Dorigo, and M. Birattari. Autonomous task partitioning in robot foraging: an approach based on cost estimation. *Adaptive Behavior*, 21(2):118–136, 2013.

[19] Giovanni Pini, Arne Brutschy, Marco Frison, Andrea Roli, Marco Dorigo, and Mauro Birattari. Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, 5(3-4):283–304, 2011.

[20] Giovanni Pini, Matteo Gagliolo, Arne Brutschy, Marco Dorigo, and Mauro Birattari. Task partitioning in a robot swarm: A study on the effect of communication. *Swarm Intelligence*, 7(2-3):173–199, 2013.

[21] F.L.W. Ratnieks and C. Anderson. Task partitioning in social insects. *Insectes Sociaux*, 46(2):95–108, 1999.

[22] Christopher Rouff. TR: FS-07-06: Papers from the 2007 AAAI Fall Symposium. pages 112–115, 2007.

[23] Amanda J.C. Sharkey. Swarm robotics and minimalism. *Connection Science*, 19(3):245–260, 2007.

Preprint submitted to 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots . Received January 29, 2018.