Demystifying Emergent Intelligence and Its Effect on Performance In Large Robot Swarms

John Harwell
Department of Computer Science and Engineering, University of Minnesota
harwe006@umn.edu

London Lowmanstone
Department of Computer Science, Harvard University
lowmanstone@college.harvard.edu

Maria Gini
Department of Computer Science and Engineering, University of Minnesota
gini@umn.edu

ABSTRACT

We investigate the emergence of swarm intelligence using task allocation in large robot swarms. First, we compare task decomposition graphs of different levels of richness and measure the emergent intelligence arising from self-organized task allocation by deriving STOCH-N1, a stochastic allocation algorithm which contextualizes per-robot task allocation decisions based on a previous task's neighborhood within the graph. The results are compared to other state of the art algorithms. Second, we derive MAT-OPT: a greedy algorithm that optimally solves the swarm task allocation problem by representing the swarm's task allocation space as a matroid under some restrictive assumptions. We compare the MAT-OPT allocation method, which disregards task dependencies, with STOCH-N1, which emphasizes collective learning of graph structure (including dependencies). Results from an object gathering task show that swarm emergent intelligence (1) is sensitive to the richness of task decomposition graphs (2) is positively correlated with performance, (3) arises out of learning and exploitation of graph connectivity and structure, rather than graph content.

KEYWORDS

Swarm Robotics; Foraging; Task Allocation; Task Decomposition; Matroids

ACM Reference Format:
John Harwell, London Lowmanstone, and Maria Gini. 2020. Demystifying Emergent Intelligence and Its Effect on Performance In Large Robot Swarms. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Swarm robotics (SR) is the study of large-scale robotic systems consisting of either homogeneous or heterogeneous robots called agents. Originally an offshoot of Swarm Intelligence, SR initially focused on approaches inspired from natural systems such as bees, ants, and termites [33] with which they share the core principles of scalability, emergence, flexibility, and robustness. *Scalability* is the ability of SR systems to scale to hundreds or thousands of agents, due to a lack of centralized control or single point of failure [20]. *Emergence* is the appearance of self-organizing behaviors (and therefore intelligence) arising through a collective search process within the swarm due to robot interactions [10, 38]. *Flexibility* results from reactive and adaptive swarm-level behaviors arising from localized decision making that mitigates adversity and exploits beneficial changes in dynamic environmental conditions and evolving problem definitions [15, 37]. *Robustness* is the swarm's ability to tolerate fluctuations in the number of agents due to introduction of new agents, robotic failures, or sensor/actuator noise. Unlike many multi-agent systems which cannot withstand unpredictable losses, a high failure rate within a swarm does not prevent the accomplishment of its objective [20].

The duality between SR and natural systems enables effective parallels to be drawn with many naturally occurring problems, such as foraging, collective transport of heavy objects, environmental monitoring/cleanup, self-assembly, exploration, and collective decision making [16, 32]. SR systems are well suited to tackle problems where robustness and flexibility is key to success, such as space [31].

We study foraging from a task allocation perspective. In a foraging task, robots gather objects from across a finite operating arena and bring them to a central location (often called the *nest*) for further processing. Foraging is one of the most extensively studied applications of SR due to its straightforward mapping to important real-world applications [16] such as tracking lake health, clearing a corridor on a mining operation, hazardous material cleanup, or search and rescue [16, 19, 32]. In this work, we seek to quantify the relationship between swarm emergent intelligence and the richness of the task decomposition graph used for allocation, and to understand the origin of the emergent intelligence that arises from various task allocation methods. Specifically, we seek answers to the following questions:

Q.1 Do *complex* task decomposition graphs result in greater swarm emergent intelligence than *compound* graphs, and is higher intelligence correlated with higher performance?

Q.2 Is the emergence of swarm intelligence within task decomposition graphs a function of the graph connectivity/structure, or the graph content (nodes and edge weights)?

Our definitions of *compound* and *complex* task decomposition graphs are drawn from the extended Multi-Robot Task Allocation (MRTA) taxonomy terminology proposed by [18]. They separate the concept of *atomic* tasks, which are not decomposable, from that of *decomposable* tasks, which are. They define *compound* tasks as tasks having exactly one possible decomposition (which is multi-agent allocatable), and *complex* tasks, as tasks which have multiple decompositions (each of which is multi-agent allocatable).

1.1 Background and Related Work

Matroids generalize notions of independence in linear algebra and graphs, and are attractive because matroidal problem formulations

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

are optimally solvable with simple greedy algorithms [23, 34, 35]. Briefly, a matroid $\mathcal{M}$ on $\mathcal{S}$ is an ordered pair $(\mathcal{S}, \mathcal{I})$, where $\mathcal{S}$ is the *ground set* of $\mathcal{M}$, and $\mathcal{I}$ is a collection of independent subsets of $\mathcal{S}$ satisfying the following conditions:

M.1 $\emptyset \in \mathcal{I}$
M.2 If $Y \in \mathcal{I}$ and $X \subset Y$, then $X \in \mathcal{I}$
M.3 If $X, Y \in \mathcal{I}$ and $|X| < |Y|$, then $\exists\, y \in \{Y \setminus X\}$ such that $X \cup \{y\} \in \mathcal{I}$

The third condition is the independence augmentation axiom, also sometimes referred to as the exchange property. If X is independent and there exists a larger independent set Y, then X can be grown to a larger independent set, implying that every maximal independent set is maximum; in other words, all maximal independent sets have the same cardinality.

Correll et al. have shown that SR systems are competitive with deterministic approaches when communication is tightly constrained and/or when only partial information about the environment is available [4]. Auction-based approaches modeling the task allocation problem as an intersection of matroids have been shown to be effective in achieving provably optimal bounds in decentralized auctions [36]. Auction approaches, while successful in spatially distributed task allocation in some applications, do not scale well with the number of robots or tasks [17], and further depend on regular communication of cost functions which may not be possible in unstable environments or with unreliable agents. Matthey et al. modeled the interactions of agents and their physical environment as chemical reactions between molecules [21] such that no *a priori* knowledge of the workload for task allocation was required.

Task partitioning involves dividing a *decomposable* task into simpler subtasks which are multi-agent allocatable [18, 29]. Many of the solutions to foraging tasks found by social insects employ this strategy, and task partitioning in both natural and SR systems has many well-known benefits, including (1) increased performance at group level, (2) stimulated specialization, (3) parallel task execution, (4) reduced interference between individuals due to self-organization around spatial locality of subtasks in comparison to the unpartitioned task, (5) improved exploitation of the environment, and (6) improved transport efficiency [12, 25, 27]. Prior work on task partitioning in SR has focused on allocating individuals to subtasks to maximize efficiency given the optimal task distribution *a priori* [2, 4, 17, 21] and did not consider sequential interdependencies between subtasks (with [3, 8, 9, 14, 27] being notable exceptions).

In many cases, *a priori* optimal task distributions are not feasible because (1) complete information on the environment is not available, (2) the environment itself is unstable[22, 30], or (3) the task decomposition is *complex*, and the space of possible agent-task allocations is exponential in the number of agents and tasks [18]. We investigate *self-organized* task partitioning as a possible solution in such cases, in which robots self-allocate tasks and the interactions between robots working on inter-dependent subtasks implicitly disseminate the costs of these tasks to the swarm as a whole, and give rise to measurable emergent intelligence [15].

Prior work on self-organized task partitioning in foraging has utilized a *compound* task decomposition graph representing task dependencies and hierarchy with a compound root task and two atomic subtasks which enabled utilization of a single existing cache [3, 8, 9, 14, 27, 28]. *Caches* are temporary storage sites where materials can be dropped and picked up asynchronously. Asynchronicity can be beneficial because it can reduce material losses due to imbalances between foraging and processing rates [13]. We extend this to a *complex* task decomposition graph with a multiplely decomposable root task which collectively enables more complex swarm behaviors such as cache creation, transfer, and depletion. This generalizes the task partitioning approach in [14] by eliminating the requirement for the cache in the arena to be maintained by an outside process.

To answer **Q.1**, we derive the algorithm STOCH-N1, which uses the neighborhood of a finished task within a task decomposition graph to stochastically allocate a new task. We evaluate its emergent intelligence and performance across *compound* and *complex* task decomposition graphs for a foraging task, and show that swarm emergent intelligence is strongly correlated with performance, and greater for complex than for compound task decomposition graphs.

To answer **Q.2**, we derive MAT-OPT, a matroid [23, 34, 35] theoretic method in which we are able to prove that if we disregard task dependencies from our task decomposition graph, an extension of our task decomposition graph is optimally solvable with a greedy algorithm for a single robot under some restrictions. It then follows that an optimal allocation policy for the swarm is the disjoint union of individual robot policies (intersection of matroids [36]). By comparing the performance of MAT-OPT under constraints with that of STOCH-N1, we can determine whether emergent intelligence is more tied to graph *content* (tasks within the graph, treated as independent by MAT-OPT), or to graph *structure*; MAT-OPT should be the highest performing allocation method if it is the former.

We compare the emergent intelligence and performance of MAT-OPT against STOCH-N1 (which is specifically designed to enable collective learning of graph connectivity, including task dependencies), and other state of the art approaches at real-world scales (swarms of > 1,000 robots). We show that swarm emergent intelligence is strongly tied to collective learning of graph connectivity and structure (as opposed to graph content) by injecting accurate knowledge about graph content (task costs), and comparing resulting performance. STOCH-N1 is the most highly performing method in all tested cases, providing strong quantitative evidence supporting the suitability of SR systems for dangerous/unstable environments in which only partial or incomplete information is available. MAT-OPT is shown to be suboptimal in many cases, due to its disregard for graph structure and dependencies; however, results suggest future synergies between theoretical methods leveraging emergent intelligence is possible.

2 TASK ALLOCATION SPACES

We first consider the derivation of the task allocation *space* swarms will move through as robots choose tasks, as separate from the task allocation *method*, which defines *how* a robot will move through the allocation space. Previous work [8, 9, 14, 27] on task partitioning divided the overall decomposable task $\mathcal{T}$ into a sequence of two interdependent atomic subtasks. We extend this decomposition by relaxing subtask atomicity, allowing robots to partition subtasks into yet simpler subtasks (Fig. 1). This *recursive* task partitioning process results in a *complex* task decomposition graph [18] with a *multiplely decomposable* root task $\mathcal{T}$. Recursive task partitioning

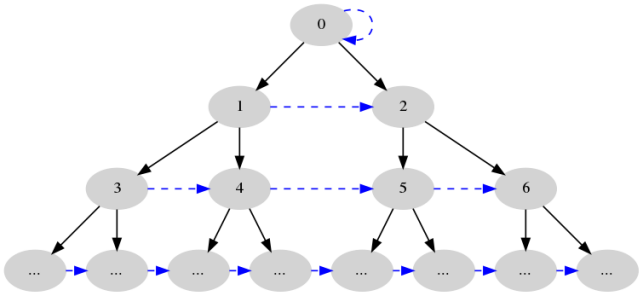

Figure 1: Complex decomposition of $\mathcal{T}$ via task partitioning for a single robot. $\mathcal{G}_d(\mathcal{V}_a, \mathcal{E}_d)$ is a complete binary tree formed by the black edges connecting the task nodes. Each ϕ_i is formed by connecting the nodes at height i within $\mathcal{G}_d(\mathcal{V}_a, \mathcal{E}_d)$ together with the blue edges.

results in a complete binary tree, which we denote by $\mathcal{G}_d(\mathcal{V}_a, \mathcal{E}_d)$, where $\mathcal{V}_a$ is the set of all tasks in the tree, and E_d is the set of connecting edges.

We consider a self-organizing swarm $\mathcal{S}$ given an objective O to minimize the cost of repetitively completing $\mathcal{T}$. Let $\phi_a = \{v \in \mathcal{V}_a | height(v) = 0, 1, 2, \dots\}$ be the set of sets of vertices with height i. Each $\phi_a \in \Phi_a$ is a topologically ordered task decomposition within $\mathcal{G}_d(\mathcal{V}_a, \mathcal{E}_d)$ (Fig. 1), and an equivalent means to accomplish O (different *functionalities* in the terminology of [36]). From a natural standpoint, these decomposition sequences and the tasks within them are analogous to the castes and observed task specialization common among insects such as ants, bees, etc. [7, 8].

Each $\mathbf{r}_s \in \mathcal{S}$ will execute a series of task allocations $a = 1, \dots, A$ (multiple task allocations may occur simultaneously, or involve a single robot at a single instant in time); all robot task allocation decisions are independent. Each time $\mathbf{r}_s$ finishes a task $v_f \in \mathcal{V}_{a-1}$, it allocates a new task $v_a \in \mathcal{V}_a$.

During allocation at T_a, $\mathbf{r}_s$ allocates a task v_a from $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ to execute next, minimizing a temporally varying cost function $C(v, t)$. $C(v, t)$ represents the cost to $\mathbf{r}_s$ of executing v_a starting at time t. As $\mathbf{r}_s$ allocates itself tasks over time, its allocation history clearly forms a directed, weighted path within $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$. $\mathbf{r}_s$ seeks a temporal path through $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ that is optimal (of minimal cost). Defining decision variables $x_{ij} \in \{0, 1\}$ indicating that task j has been allocated to robot i, and a utility u_{ij} for the allocation (i, j), solving this problem optimally for $\mathcal{S}$ is equivalent to solving the following linear program for each allocation $a \in A$:

$$\begin{aligned} \max \quad & f(x_{ij}, u_{ij}) \\ \text{s.t.} \quad & \sum_{a \in A} \sum_{j=1}^{|\mathcal{V}_a|} x_{ij} <= 1 \\ & \sum_{i=1}^{|\mathcal{S}|} x_{ij} <= T \end{aligned} \tag{1}$$

with $f(x_{ij}, u_{ij})$ a monotonically decreasing function. Each possible task is assigned at most once, and robots can change their allocated task at most once per timestep.

2.1 STOCH-N1 Task Allocation Space

To answer **Q.1**, we derive a task allocation space sensitive to task decomposition graph richness. We induce a subgraph $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ on $\mathcal{G}_d(\mathcal{V}_a, \mathcal{E}_d)$ containing all tasks available to $\mathbf{r}_s$ for allocation after it has finished task $v_f \in \mathcal{V}_A$ at time T_a by adding edges to $\mathcal{E}_d$ to form $\mathcal{E}_a$ (union of solid and dotted edges in Fig. 2). Formally:

$$\mathcal{E}_a = \{e_{ji} \; \forall \; e_{ij} \in \mathcal{E}_d\} \cup \{e_{ij} \; \forall \; v_i, v_j \in \mathcal{V}_a : i \neq j, sibling(v_i, v_j)\} \tag{2}$$

$\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ (Fig. 2) then represents the union of possible task allocations for each task $v_f \in \mathcal{V}_A$ $\mathbf{r}_s$ could have finished. We contextualize this task allocation graph around v_f by defining its *allocation neighborhood*, restricting the tasks that can be allocated:

$$\mathcal{E}_{nbhd} = \mathcal{E}_a \setminus \{e \in \mathcal{E}_a : dist(v_f, v) \; \leq \; r_{nbhd} \; \forall \; v \in \mathcal{V}_A\} \tag{3}$$

$\mathcal{E}_{nbhd}$ induces a subgraph $\mathcal{G}_{nbhd}(\mathcal{V}_{nbhd}, \mathcal{E}_{nbhd})$ on $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ with central vertex v_f and radius r_{nbhd}. In this work we restrict our study to $r_{nbhd} = 1$; future work may further explore larger neighborhoods. Examples of the three possible cases of this induced subgraph on $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ with $r_{nbhd} = 1$ are shown in Fig. 2.

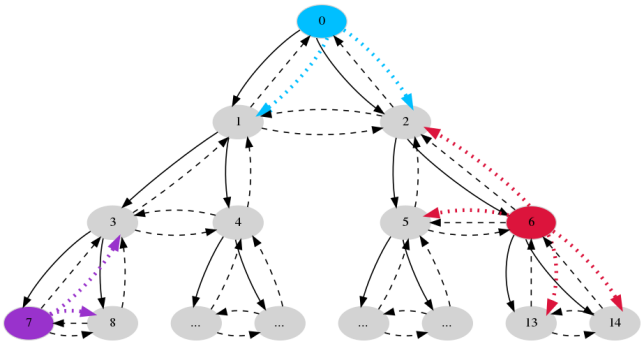

Figure 2: Task allocation neighborhoods $\mathcal{G}_{nbhd}(\mathcal{V}_{nbhd}, \mathcal{E}_{nbhd})$ within $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ induced by $\mathcal{E}_{nbhd}$ for each $v_f \in \mathcal{V}_a$. A task is either the root (node 0, blue), a leaf node (node 7, purple), or neither (node 6, red). The three possible cases of allocation neighborhoods reachable from v_f are shown via different colored edges.

2.2 MAT-OPT Task Allocation Space

We create a set of task allocation graphs $\mathcal{G}_{|A|}(\mathcal{V}_{|A|}, \mathcal{E}_{|A|})$ (one for each task allocation $\mathbf{r}_s$ performs) connected through time via sets of directed edges such that the connecting edges $\mathcal{E}_{a-1}$ from $\mathcal{G}_{a-1}(\mathcal{V}_{a-1}, \mathcal{E}_{a-1})$ to $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ each represent a possible task allocation decision leading from a specific $v_f \in \mathcal{V}_{a-1}$ to any $v_a \in \mathcal{V}_a$. Given such connectivity, it is clear that $\mathbf{r}_s$ can choose its next task v_a without restrictions resulting from the neighborhood of v_{a-1} (i.e., $r_{nbhd} = \infty$). Formally, $\mathcal{G}_{|A|}(\mathcal{V}_{|A|}, \mathcal{E}_{|A|})$ is defined by:

$$\begin{aligned} \mathcal{V}_{|A|} &= \bigcup_{a=1}^{A} \mathcal{V}_a \\ \mathcal{E}_{|A|} &= \Big(\bigcup_{a=2}^{A-1} \{v_{a-1} \in V_{a-1}, v_a \in V_a : e_{v_{a-1}, v_a}\} \Big) \end{aligned} \tag{4}$$

That is, for all $a = 2, \dots, A$, $deg^-(v) = deg^+(v) = |\mathcal{V}_a| \; \forall \; v \in \mathcal{V}_a$.

Using the notation of [36], we define an allocation $v \in \mathcal{V}_{|A|}$ more formally as a triplet (s, v_j, ϕ_k), read as "robot s performs task v_j to for ϕ_k". Then, O can be more precisely defined as $O \subseteq \{(v_j, \phi_k) | v_j \in \mathcal{V}_{|A|}, \phi_k \in \Phi_{|A|}\}$. That is, a robot allocates a task $v_j \in \mathcal{V}_a$ from $\phi_k \in \Phi_a$ in order to help complete O by performing a task from the task sequence at height k within $\mathcal{G}_{|A|}(\mathcal{V}_{|A|}, \mathcal{E}_{|A|})$. We assign an independent set $\mathcal{I}_{\mathbf{r}_s}$ to each $\mathbf{r}_s \in \mathcal{S}$, containing sets

of triplets $\{(s, v_j, \phi_k)\}$ which are the feasible sets of functionality-requirement [36] allocations. Then, $\mathcal{M}_{\mathbf{r}_s} \in \mathbb{M}, \mathcal{M}_{\mathbf{r}_s} = (\mathcal{V}_{|A|}, \mathcal{I}_{\mathbf{r}_s})$ are matroids containing the set of maximal independent sets for each $\mathcal{I}_{\mathbf{r}_s}$. From [36], an optimal task allocation policy can be obtained from the intersection of these matroids (a reframing of Eqn. (1)).

We now set our matroid optimality constraints and prove that $\mathcal{M}_{\mathbf{r}_s}$ satisfies the matroid properties **M.1**, **M.2**, **M.3**:

(1) $C(v, t)$ *monotonicity and accuracy.* We assume $C(v, t)$ is a monotonically increasing function which accurately estimates the cost of performing a task v starting at time t, and that this accuracy is maintained regardless of the task allocations chosen. That is, while a task allocation at T_a might affect future task costs at time $T_{a'}$, those cost changes do not violate monotonicity. By our definition, $-C(v, t)$ is a *sub-modular monotone* function [36].
Environments such as large-scale robotics warehouses are well modeled by cost functions of this type. For example, data about how long it takes to find an object of a specific type and bring it to a specific location are well known, and provide excellent estimates of the execution time for tasks. However, sensor/actuator error can still cause the theoretically collision-free paths to not be collision free in practice, thus increasing actual task execution times beyond estimates, but not to reduce them.
(2) *Task allocation independence.* While a task chosen by $\mathbf{r}_s$ might affect the future task cost estimates of another robot $\mathbf{r}_{s'}$, such effects will not change $\mathbf{r}_{s'}$'s allocation decisions, due to $C(v, t)$ monotonicity and submodularity.
(3) *Task allocation frequency.* We assume $\mathbf{r}_s$ can abort its current task and allocate a new one at any point in time.
(4) *Task Switching Cost.* We assume $\mathbf{r}_s$ can switch tasks without cost, or equivalently, with equal cost. Without loss of generality, task switching costs can be encoded in $C(v, t)$.
(5) *Temporal precedence constraint satisfaction*: All constraints for a task v_j are satisfied when $\mathbf{r}_s$ allocates itself v_j. That is, for a task v_j at position j within a task sequence ϕ_i, other robots have already completed all tasks $v_0 \dots v_{j-1}$ in the specific instantiation of the sequence. This is a restatement of the assumptions of [5, 9], in which task independence is unaffected by group dynamics, and effectively removes task dependencies from $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$. This is a necessary assumption for proving $\mathcal{M}_{\mathbf{r}_s}$ is a graphic matroid, because graphic matroids cannot contain cycles, and edges between tasks within the same $\mathcal{V}_a$ trivial form cycles between the allocations performed at T_a and T_{a+1}.

We wish to create an independence system $\mathcal{I}_{\mathbf{r}_s}$ over $\mathcal{V}_{|A|}$ and, furthermore, show that $\mathcal{M}_{\mathbf{r}_s} = (\mathcal{V}_{|A|}, \mathcal{I}_{\mathbf{r}_s})$ is a matroid. We define $\mathcal{I}_{\mathbf{r}_s}$ as:

$$\mathcal{I}_{\mathbf{r}_s} = \{I : I \subset \mathcal{V}_{|A|}, |I \cap \mathcal{V}_a| \leq 1 \; \forall \; a = 1, \dots, A\} \tag{5}$$

$\mathcal{I}_{\mathbf{r}_s}$ consists of every set of task allocation decisions that $\mathbf{r}_s$ can make ($\mathbf{r}_s$ cannot choose more than one task during a single allocation). Since tasks take varying amounts of time, task allocation decisions can occur at *any* $1 \leq t \leq T$. Depending on the previous decisions made (i.e., the members of a particular $X \in \mathcal{I}_{\mathbf{r}_s}$), at some time t^* another decision will be made and another vertex added to X (i.e., a robot aborts its current task and starts the new task, which had *smaller* cost than the remaining portion of its current task). For another $Y \in \mathcal{I}_{\mathbf{r}_s}, X \neq Y$, t^* will occur while $\mathbf{r}_s$ is executing a task, and 0 tasks from $\mathcal{G}_{t^*}(\mathcal{V}_{t^*}, \mathcal{E}_{t^*})$ will be added to Y (if the new task does not have a lower cost estimate than the remaining part of $\mathbf{r}_s$'s current task). Effectively, robots perform task allocation each timestep, choosing to continue their current task or to abort it and choose a new one, depending on which they think will have smaller cost and finish first.

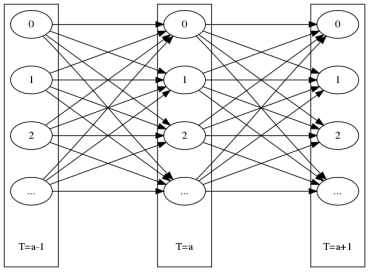

Figure 3: Unrolled task allocation graph $\mathcal{G}_{|A|}(\mathcal{V}_{|A|}, \mathcal{E}_{|A|})$ shown for arbitrary task allocation times $a-1, a, a+1$.

To prove $\mathcal{I}_{\mathbf{r}_s}$ is an independence system, we now prove **M.1**, **M.2**. **M.1** is true by construction, as $\mathcal{V}_{|A|} = \emptyset$ is admissable under Eqn. (5). **M.2** is also satisfied by construction, because we defined $\mathcal{I}_{\mathbf{r}_s}$ so that for every $X \in \mathcal{I}_{\mathbf{r}_s}$, no pair of elements in X are a part of the same $\mathcal{V}_a$, by Eqn. (5). Thus, for a subset of X, which has *fewer* elements, it would also be true that no pair of elements would be a part of the same $\mathcal{V}_a$. Since $\mathcal{I}_{\mathbf{r}_s}$ is an independence system, any $I \in \mathcal{I}_{\mathbf{r}_s}$ is an *independent set.*

Finally, we must prove **M.3** for $\mathcal{I}_{\mathbf{r}_s}$ for any $\mathbf{r}_s \in \mathcal{S}$, in a similar manner to [36]: for any two sets of task allocation decisions $\{X\}, \{Y\}$, if $|X| < |Y|$, we can take an allocation decision from Y and add it to X and still have a valid independent set. Since X and Y contain at most one element per allocation time, then we must have $|X| < |Y| \leq A$, and therefore there must be least one time T_{a^*} for which $|X \cap \mathcal{V}_{a^*}| = 0$ and $|Y \cap \mathcal{V}_{a^*}| = 1$. Then we know $Y \cap \mathcal{V}_{a^*} \subseteq Y \setminus X$ such that $X \cup \{Y \cap \mathcal{V}_{a^*}\} \subset \mathcal{I}_{\mathbf{r}_s}$. In other words, since $|Y| > |X|$, it must have done an allocation at a time when X did not, and this allocation can be added to X to form a new independent set.

This concludes the proof of **M.3**, and therefore $\mathcal{M}_{\mathbf{r}_s} = (\mathcal{V}_{|A|}, \mathcal{I}_{\mathbf{r}_s})$ for all $\mathcal{M}_{\mathbf{r}_s} \in \mathbb{M}$ are matroids.

3 TASK ALLOCATION METHODS

3.1 STOCH-N1 Task Allocation Method

We define a method capable of task allocation in each of the 3 cases for $\mathcal{G}_{nbhd}(\mathcal{V}_{nbhd}, \mathcal{E}_{nbhd})$ from Fig. 5 in Algorithm 1. *EmployPartitioning*() and *ChooseChild*() are the task partitioning and subtask selection methods from [14], and β_i is the *Task Allocation Block* (TAB), a binary tree consisting of a root partitionable task β_{00} and its two subtasks, β_{01}, β_{02}. We use numerical subscripts ij attached to β_{ij} to denote the jth task within a TAB rooted at vertex i (we use a left-to-right numbering scheme, as depicted in Fig. 1). In previous work, there was only a single TAB, $\beta_0 = \mathcal{G}_{nbhd}(\mathcal{V}_{nbhd}, \mathcal{E}_{nbhd})$ with $|\mathcal{V}_a| = 3$.

Algorithm 1 STOCH-N1()

INPUT:β,v_f **OUTPUT:**β', v_a

1: **if** $v_f = \beta_0$ **then**
2: $\beta' \leftarrow$ SwitchContext($\beta, \beta_{parent(v_f)}$)
3: **else**
4: $\beta' \leftarrow$ SwitchContext(β, β_{v_f})
5: **end if**
6: **if** EmployPartitioning(β') **then**
7: **return** β', *ChooseChild*(β')
8: **else**
9: **return** β',β'_0
10: **end if**

Lines 6-10 of Algorithm 1 are unambiguous if and only if $|\mathcal{V}_a| = 3$, because for such graphs, only a single partitionable task exists, and all partitioning decisions are rooted at that node, regardless of v_f. However, for $v_f \in \mathcal{V}_a, |\mathcal{V}_a| > 3$ that are neither the root or a leaf node, ambiguity exists in *which* node to consider as the pseudo-root for the purposes of partitioning (see Fig. 4). Should the pseudo-root be $parent(v_f)$, considering v_f to be a pseudo-leaf node within a 3 node graph rooted at $parent(v_f)$? Or should it be v_f, considering v_f as the root node of a 3 node tree encompassing v_f and its two children? We solve the ambiguity of *which* node to consider as the pseudo root by allowing multiple $\beta \subset \mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$; every vertex v that is not a root or leaf is both the root of a β rooted at v, and a leaf within a β rooted at $parent(v)$ (see Fig. 4). With this extended TAB definition, we now consider the question of *how* $\mathbf{r}_s$ should update the TAB associated with v_f (the definition of the *SwitchContext*() function). From the task decomposition graph

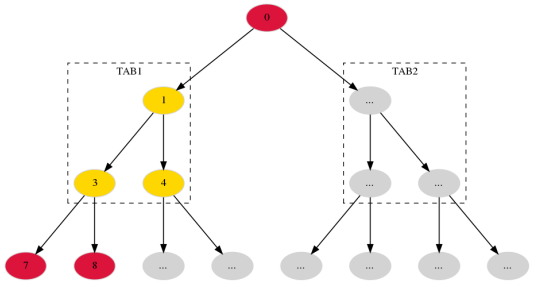

Figure 4: Task Allocation Contexts (TABs) within $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$. v_f with unambiguous partition points in red (root and leaf nodes), and ambiguous partition points in gold.

structure, each subtask (β_{i1}, β_{i2}) is half of the work of β_{i0}, it should therefore have half the cost under $C(v, t)$. The further this ratio becomes unbalanced, the more likely that stochastic variances in the swarm and/or environment will cause tasks to unexpectedly take longer, and the less reliable task estimates should be considered. So a robot attempting to stochastically minimize the cost of the tasks they choose should change the current β_i to a different β_j within $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$, depending on how "balanced" task cost estimates are between different $\beta \subset \mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$; that is, how close the cost estimate for $\hat{\beta_{i0}}$ is to the sum of the cost estimates of its subtasks $(\hat{\beta_{i1}} + \hat{\beta_{i2}})$. We formalize this intuition in the following equations, which comprise the essence of *SwitchContext*():

$$\theta_\beta(\beta_i, \beta_j) = \Omega_{\beta_r}\left(\frac{|\hat{\beta_{i0}} - (\hat{\beta_{i1}} + \hat{\beta_{i2}})|/\hat{\beta_{i0}}}{|\hat{\beta_{j0}} - (\hat{\beta_{j1}} + \hat{\beta_{j2}})|/\hat{\beta_{j0}}} - \Omega_{\beta_o}\right) \quad (6)$$

$$P_{\beta_{sw}}(\beta_i, \beta_j) = \frac{1}{1 + e^{-\theta_\beta(\beta_i, \beta_j)}} \quad (7)$$

Where Ω_{β_r} is the reactivity of the logistic function, and Ω_{β_o} is the offset. In the degenerate case in which the ratios are exactly equal, $P_{\beta_{sw}}$ is 0.5 (we choose $\Omega_{\beta_o} = 1.0$ in this work). Using Eqn. (7), and our extended TAB definition, a robot is constrained in its choices to update β_i by the structure of $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ to $\{\{\beta_i\} \cup \{\beta_j : v_f = \beta_{j1,j2}\}\}$ if $v_f = \beta_{i0}$, and $\{\{\beta_i\} \cup \{\beta_j : v_f = \beta_{j0}\}\}$ otherwise.

The STOCH-N1 algorithm (Algorithm 1) extends the task allocation algorithms from previous work [3, 8, 9, 14, 27] (lines 6-10) with lines 1-5, solving the problem of task partition node ambiguity. It realizes $\mathcal{G}_{nbhd}(\mathcal{V}_{nbhd}, \mathcal{E}_{nbhd})$ with $r_{nbhd} = 1$, because all $\beta \in \mathcal{G}_{nbhd}(\mathcal{V}_{nbhd}, \mathcal{E}_{nbhd})$ have radius 1, and therefore v_a will be a distance of at most 1 from v_f. It is therefore well suited to investigate the effect of $\mathcal{G}_a(\mathcal{V}_a, \mathcal{E}_a)$ richness on emergent intelligence. It approximately solves Eqn. (1) via stochastic greedy allocation for a single $a \in A$ at a time, not considering future allocations, and does not have any guarantees of optimality.

3.2 MAT-OPT Task Allocation Method

$\mathbf{r}_s$ seeks an optimal path (measured by $C(v, t)$) between v_{a1} and v_{aA}, which belong to $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_A(\mathcal{V}_A, \mathcal{E}_A)$, respectively. We define a global greedy algorithm *GG* capable of finding the lowest cost path between v_{a1} and v_{a2} by determining the best decision for a robot to take at any time. Let $\hat{v_a}$ be the robot's cost estimate of a task $v_a \in \mathcal{V}_a$ using our cost measure $C(v, t)$ at time T_a. The algorithm begins with $F_0 = \emptyset$, and iteratively selects at task allocation time T_1 the element $v_a \in \mathcal{V}_1$ independent from F_{a-1} maximizing $-C(v_a, t)$:

$$F_a = F_{a-1} \bigcup \left\{ \underset{v_a : F_{i-1} \cup \{v_a\} \in \mathcal{I}_{\mathbf{r}_s}}{\operatorname{argmax}} -C(v_a, t | F_{a-1}) \right\} \quad (8)$$

It follows that the *GG* algorithm gives the optimal decision path for $\mathbf{r}_s$ [23]. The path is not guaranteed to be unique; all optimal paths will have the same overall cost and cardinality due to **M.3**.

In real SR systems, a global greedy algorithm is not possible in general: environment conditions can change, robots can enter/leave an operating area, robots can fail, sensor/actuator errors can result in unpredictable congestion. All these events affect the optimal decision path, and cannot be known *a priori*.

We now define a local greedy algorithm (*LG*) for determining what decision a robot should make without any information about future costs, and show that this algorithm gives the same result as *GG*. The *LG* algorithm begins with an empty set F_{LG}, and at each task allocation $a = 1, \ldots, A$, it adds $\operatorname{argmax}_{v_a \in \mathcal{V}_a} \hat{v_a}$ to F_{LG} (accurate task selections ensured by accurate and monotonically increasing task cost estimates).

We prove that *LG* gives the same result as *GG* with a proof by contradiction. Assume that *LG* does worse than *GG*. That is, $\sum_{v \in F_{LG}} \hat{v} < \sum_{v \in F_{GG}} \hat{v}$. Note that since F_{GG} is maximally independent, and LG chooses exactly one task at each allocation, both F_{GG} and F_{LG} have exactly one task from each V_a. Thus, in order for

$\sum_{v \in F_{LG}} \hat{v} < \sum_{v \in F_{GG}} \hat{v}$ to be true, it must also be true that there exists some task allocation time T_{a^*} for which the task chosen by *GG* from V_{a^*} has a smaller estimated cost than the task chosen by *LG* from V_{a*}. However, this is impossible, because we have defined *LG* to take the task with the least cost from each V_a. Thus, since *LG* cannot do worse than *GG*, it must also be optimal. □

4 APPLICATION TO A FORAGING TASK

We instantiate the task decomposition graph shown in Fig. 5, extending previous work [8, 14, 27] which explicitly or implicitly defined the following tasks, implemented as vertices comprising $\mathcal{V}_a$ (gold vertices in Fig. 5):

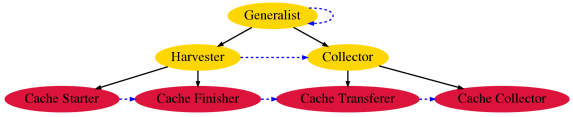

Figure 5: Task decomposition graph, $\mathcal{G}_d(\mathcal{V}_a, \mathcal{E}_d)$, for a foraging task. Solid edges indicate task decomposition, dashed blue edges indicate task sequences ($\phi_i, i = \{0, 1, 2\}$). Gold tasks were defined in previous work [14], red tasks are defined in this work.

- *Generalist*: Acquire a free block of highest utility and bring it to the nest. This is $\mathcal{T}$, and one way of accomplishing the swarm objective O via ϕ_0.
- *Harvester*: Acquire a free block of highest utility and bring it to the existing cache of highest utility, calculated as:

$$\mu_{\hat{C_j}}(t) = \frac{e^{-\tau_{\hat{C_j}}(t)} |\hat{C_j}|}{\|\mathbf{r}_s - \mathbf{C_j}\| \|\mathbf{C_j} - \mathbf{nest}\|} \tag{9}$$

  where $|\hat{C_j}|$ is the estimated size of the jth cache known to $\mathbf{r}_s$ (caches might not exist anymore, and therefore a robot only has estimates of their existence). This equation emphasizes selection of caches that are close to the position of $\mathbf{r}_s$, but also that are closer to the nest, while accounting for the relevancy of a robot's information about C_j [14]. This task accomplishes part of O via ϕ_1.
- *Collector* Acquire a block from the existing cache of highest utility (Eqn. (9)) and bring it to the nest. This task accomplishes part of O via ϕ_1.

We extend prior work with the following tasks, implemented as vertices comprising the task sequence ϕ_2 (red vertices in Fig. 5):

- *Cache Starter*: Acquire a free block of highest utility and bring it partway back to the nest, and then drop it at a feasible site to start a *De Novo* cache, according to Eqn. (10). A feasible site is one at least a distance η from all known caches $\hat{C_1} \ldots \hat{C_n}$. The best cache site $\mathbf{x}$ for $\mathbf{r}_s$ is computed as follows:

$$\begin{aligned} &\max_{\mathbf{x}} && \mu_{\mathbf{r}_s} \\ &\text{s.t.} && \left\| \mathbf{x} - \hat{\mathbf{C_m}} \right\| \geq \eta, \ m = 1, \ldots, n. \end{aligned} \tag{10}$$

  where

$$\mu_{\mathbf{r}_s} = \left(\|\mathbf{x} - \mathbf{x}_{\mathbf{r}_s}\| \left\| \mathbf{x} - \frac{\mathbf{x}_{\mathbf{r}_s} - \mathbf{nest}}{2} \right\| \right)^{-1} \tag{11}$$

  Intuitively, cache sites that are close to $\mathbf{x}_{\mathbf{r}_s}$ are better (less work to get to, less chance of being found unsuitable upon arrival), as are sites that are close to the halfway point between the robot's current position and the nest (bisecting the space maximizes available areas for *Collector* and *Harvester* tasks, reducing interference). This task accomplishes part of O via ϕ_2.
- *Cache Finisher*: Acquire a free block of highest utility and place it within a distance η to a *De Novo* cache (a single free block a distance η from all other blocks) of maximum utility (Eqn. (9) with $|\hat{C_j}| = 1$) in order to create a new cache. This task accomplishes part of O via ϕ_2.
- *Cache Transferer*: Acquire a block from the existing cache of highest utility and transport it to the existing cache with the second highest utility. This accomplishes part of O via ϕ_2.
- *Cache Collector*: Acquire a block from the existing cache of highest utility (Eqn. (9)) and bring it to the nest. This task accomplishes part of O via ϕ_2.

By defining ϕ_2, we have also implicitly defined β_1, β_2.

4.1 Experimental Setup

The experiments described in this paper have been carried out in the ARGoS [24] simulator. We employ a dynamical physics model of the robots in a three dimensional space for maximum fidelity (robots are still restricted to motion in the XY plane), using a model of the s-bot developed by [6]. For all experiments we average the results of 20 experimental runs of $T = 10,000$ seconds. We define our cost measure $C(v, t)$ as the execution time for a given task. We use a constant swarm density [11] (ratio of swarm size to arena area) of 10%, which helps to minimize the effects of increasing inter-robot interactions, allowing us to obtain more reliable measures of our quantities of interest. That is, by using a constant swarm density we maintain the same level of inter-robot interaction with increasing swarm sizes, removing artifacts from increasing robot interaction levels from the results, which would otherwise arise from variable density scenarios as swarm size is increased.

We evaluate our derived methods against the following controllers from other state-of-the-art literature [14, 26, 27], summarized briefly here:

- *Epsilon Greedy*, in which robots choose the task of least cost with probably ϵ, and a random task otherwise; has a linear regret bound [1, 26, 28].
- *UCB1*, in which robots treat task allocation as a multi-armed bandit problem; has a logarithmic regret bound [1, 26, 28].

4.2 Effect on Performance of Task Decomposition Graph Richness

To gain insight into the effect of task decomposition graph richness on performance (and answer **Q.1**), we compare all methods on *compound* and *complex* task decomposition graphs [18]. We define a *swarm configuration* as an (X,Y) point corresponding to a specific combination of swarm size and task allocation method. We compare and correlate the levels of emergent intelligence and performance on both types of graphs using the metric defined by [15], which is a function of swarm size and robot control algorithm. It computes the sub-linearity of increases in inter-robot interference across either (1) linearly increasing swarm sizes, or (2) task allocation methods. Linear or superlinear increases in interference levels indicates more

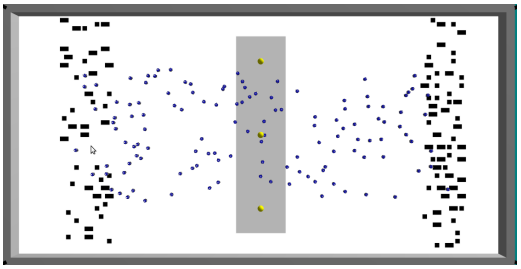

Figure 6: Screen shot of a dual source foraging scenario with multiple robots (blue blobs), objects to be collected (black squares), and the nest in the center (gray).

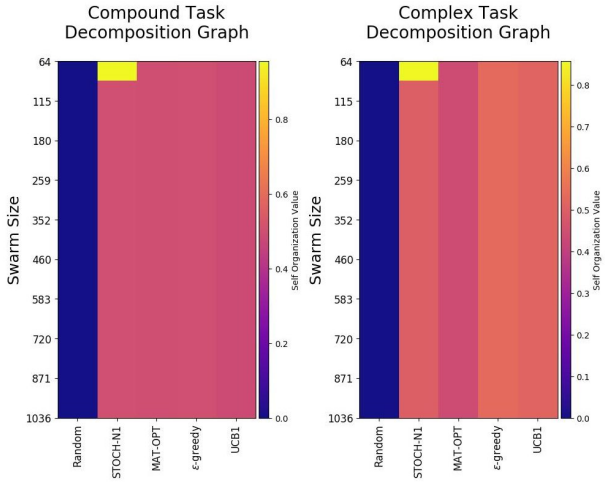

Figure 7: Observed emergent intelligence via self-organization for all methods on compound and complex task decomposition graph for scenario in Fig. 6 using a 10% swarm density under imperfect information.

random and/or disorganized motion and therefore less emergent intelligence. Previous work on task partitioning exclusively used single source foraging scenarios, in which all objects are distributed in a single cluster within a rectangular arena [8, 14, 15, 27]. In this work we utilize a *dual source* foraging scenario, shown in Fig. 6, in order to provide swarms a richer environment to learn and exploit, as opposed to the simpler single source environment.

From Fig. 7, we see that the observed levels of self-organization are uniformly higher for complex task decomposition graphs than for compound graphs across all tested methods, even those not specifically designed to be sensitive to task decomposition graph richness. The STOCH-N1 method, which was specifically designed to be sensitive to graph richness, has the highest emergent intelligence and performance across all scales. Comparing Fig. 7 and Fig. 8 we observe that, generally speaking, higher levels of swarm intelligence are positively correlated with higher performance.

4.3 Emergence of Swarm Intelligence from Task Decomposition Graph Richness

To investigate the origin of swarm emergent intelligence (**Q.2**), we ran two additional sets of experiments. First, we enforce our matroid optimality constraints by artificially injecting the swarm with perfect knowledge about evolving task costs ($C(v,t)$) at every

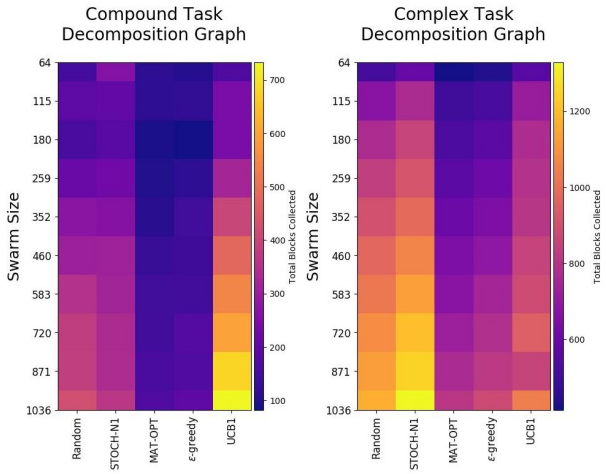

Figure 8: Performance for all methods on compound and complex task decomposition graphs scenario in Fig. 6 using a 10% swarm density under imperfect information.

timestep. From our measurements (not shown), in a ~1,000 robot swarm with a density of 10%, no more than 3% of robots are engaged in collision avoidance at any given time, on average (for the smaller swarms, it is much less), and $C(v,t)$ monotonicity is maintained. We cannot guarantee multiple robots will not interact within a finite operating arena (as strictly required to realize the MAT-OPT task allocation space without task dependencies), but our use of a low swarm density reduces the probability that interactions will violate cost function monotonicity (e.g., obviating the occurrence of excessive inter-robot interference at high densities).

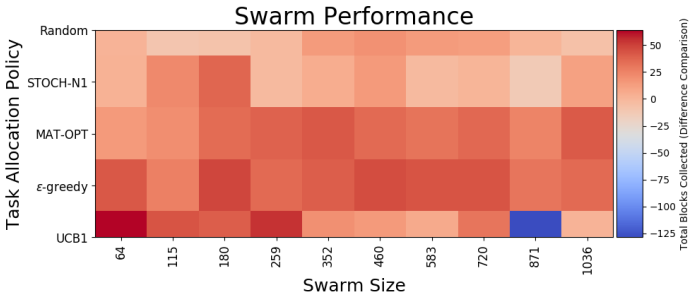

Figure 9: Performance difference comparison on a *compound* task decomposition graph between swarms operating under matroid optimality constraints vs. relaxation. Negative values indicate that a performance drop was observed when constraints were relaxed, when compared with Fig. 8.

We frame these results in terms of performance drops, comparing the drop in performance when matroid optimality constraints are relaxed between two otherwise identical swarms. In Fig. 9, *negative* values indicate a drop in performance, and *positive* values indicate that swarms operating under imperfect information performed better. We see performance drops consistently for both the MAT-OPT and the ϵ-greedy methods on the compound task decomposition graph, in the neighborhood of ~5% as expected, as they are the methods most sensitive task cost accuracy. The increased performance of UCB1 under relaxation is somewhat anomalous,

though we speculate that it could be due to many robots allocating themselves the same sequence of tasks, which at larger swarm sizes resulted in increased congestion at the shared caches. In Fig. 10, we see much smaller scales of performance drops in general, and more cases in which the swarm operating under perfect information does *better* than the one without it.

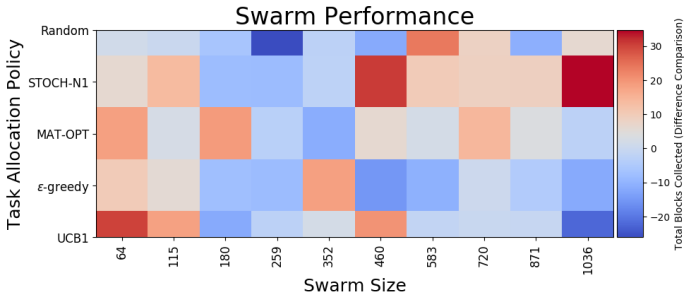

Figure 10: Performance difference comparison on a *complex* task decomposition graph between swarms operating under matroid optimality constraints vs. relaxation. Negative values indicate that a performance drop was observed when constraints were relaxed, when compared with Fig. 8.

5 DISCUSSION

We observe in Fig. 7 that the self-organization measure we employ (derived by [15]) is sensitive to the task allocation method, and that considerable self-organization emerges, in comparison to the baseline random allocation method. However, in this work we have only established a rough correlation between emergent intelligence and performance, as we see that even though swarms using random task allocation still outperform more "intelligent" methods such as UCB1 in many cases. One would expect that the MAT-OPT and ϵ-greedy methods, even without optimality constraints in place (as in this case), would do better than randomized allocations.

Comparing the heatmaps in Fig. 8 across all methods, it is clear that the richness of the task allocation graph positively affects self organization, which we believe is due to the swarm's increased ability to dynamically adapt to changing traffic flows as time progresses. From this we can answer **Q.1** and conclude that (1) task decomposition graph richness has an important and measurable effect on the emergent intelligence of a swarm, (2) more complex task decomposition graphs result in higher levels of emergent intelligence, (3) swarm emergent intelligence is positively correlated with performance.

Within Fig. 9 (compound task decomposition graph), we see empirical evidence that our matroid optimality constraints and method of enforcement were valid through the consistent performance drops observed across scales for MAT-OPT and ϵ-greedy. Examining Fig. 10, we see a general lack of trends between performance under relaxation vs. under constraints, for all tested methods, which we attribute to the higher levels of swarm intelligence present via the complex task decomposition graph. That is, the lack of trends is due to collective learning of the graph *structure*, rather than graph *content*, which gives rise to higher levels of swarm intelligence, and the richer structure of the complex decomposition graph overwhelms the effect of the optimality constraints.

For both types of decomposition graphs, the MAT-OPT method was *not* the optimal method, even under constraints, though we do observe the largest performance boost of all the tested methods under perfect information. MAT-OPT ignores task decomposition graph structure, and assumes independent tasks, which is shown to be an invalid assumption. In addition, from Fig. 8 and the performance drops in Fig. 9, Fig. 10, STOCH-N1 performs the best under both imperfect and perfect information, though it experiences performance drops, rather than increases, under perfect information. Random performed the second best in many cases, though it was outperformed by UCB1 in some cases. From these observations we can answer **Q.2**, concluding that from a task allocation perspective, swarm intelligence arises out collective learning of the structure and connectivity of the graph, rather than collective learning of the graph content (tasks and task costs), and that task dependencies cannot be ignored even in independent task allocation decisions made by individual robots.

From this discussion, it is clear that leveraging emergent intelligence is crucial in establishing (practically) optimal task allocation policies in robot swarms, and that maximally performing solutions (possibly without optimality guarantees) need to incorporate stochasticity in order to mitigate (or exploit) their inherent randomness. We have also shown that there are formal task allocation methods (UCB1) well suited to traditional typical environments when perfect information is not available. Finally, our MAT-OPT results suggest that emergent intelligence has the potential, under the right conditions, to work synergistically with theoretical models, if task independence is guaranteed.

6 CONCLUSIONS AND FUTURE WORK

We have investigated the effect that the richness of task decomposition graph has on emergent swarm intelligence. We derived compound task decomposition graphs enabling swarms to utilize existing caches, and complex task decomposition graphs enabling them to create/destroy caches. We have shown that task decomposition graph richness is positively correlated with swarm emergent intelligence and performance for many task allocation methods in an object gathering task. We have further studied the emergence of swarm intelligence as it relates to task decomposition graphs, showing that it arises out of learning and exploitation of graph structure, rather than graph context (task costs of nodes). Future work should investigate the conditions under which emergent swarm intelligence can be made to work synergistically with theoretical models. More work is also needed to refine the relationship between emergent intelligence and performance. In order to facilitate future research and collaboration, the code used for this research is open source, and can be found at https://github.com/swarm-robotics.

ACKNOWLEDGMENTS

We gratefully acknowledge Amazon Robotics, the MnDRIVE RSAM initiative at the University of Minnesota, and the Minnesota Supercomputing Institute for their support.

REFERENCES

[1] Peter Auer and Paul Fischer. 2002. *Finite-time Analysis of the Multiarmed Bandit Problem**. Technical Report. 235–256 pages.

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

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

[4] Nikolaus Correll. 2008. Parameter estimation and optimal control of swarm-robotic systems: A case study in distributed task allocation. In *Proc. IEEE Int'l Conf. on Robotics and Automation*. 3302–3307. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543714

[5] Torbjørn S. Dahl, Maja Matarić, and Gaurav S. Sukhatme. 2009. Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems* 57, 6-7 (2009), 674–687. https://doi.org/10.1016/j.robot.2008.12.001

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

[7] Ana Duarte, Franz J Weissing, Ido Pen, Laurent Keller, and Laurent Keller2. 2011. An Evolutionary Perspective on Self-Organized Division of Labor in Social Insects. *Source: Annual Review of Ecology, Evolution, and Systematics* 42 (2011), 91–110. https://doi.org/10.11

[8] Eliseo Ferrante, Ali Emre Turgut, Edgar Duéñez-Guzmán, Marco Dorigo, and Tom Wenseleers. 2015. Evolution of Self-Organized Task Specialization in Robot Swarms. *PLoS Computational Biology* 11, 8 (2015). https://doi.org/10.1371/journal.pcbi.1004273

[9] Marco Frison, Nam Luc Tran, Nadir Baiboun, Arne Brutschy, Giovanni Pini, Andrea Roli, Marco Dorigo, and Mauro Birattari. 2010. Self-organized Task Partitioning in a Swarm of Robots. In *Swarm Intelligence*. Vol. LNCS 6234. Springer, Berlin, Heidelberg, 287–298.

[10] Aram Galstyan, Tad Hogg, and Kristina Lerman. 2005. Modeling and mathematical analysis of swarms of microscopic robots. In *Proc. IEEE Swarm Intelligence Symposium, SIS 2005*. 209–216. https://doi.org/10.1109/SIS.2005.1501623 arXiv:cs/0604110

[11] Heiko Hamann. 2013. Towards swarm calculus: urn models of collective decisions and universal properties of swarm performance. *Swarm Intelligence* 7 (2013), 145–172. https://doi.org/10.1007/s11721-013-0080-0

[12] Adam G. Hart, Carl Anderson, and Francis L.W. Ratnieks. 2002. Task partitioning in leafcutting ants. (2002). https://doi.org/10.1007/s10211-002-0062-5

[13] Adam G Hart and Francis L W Ratnieks. 2000. Leaf caching in Atta leafcutting ants: discrete cache formation through positive feedback. *Animal Behaviour* 59 (2000), 587–591. https://doi.org/10.1006/anbe.1999.1332

[14] John Harwell and Maria Gini. 2018. Broadening applicability of swarm-robotic foraging through constraint relaxation. *IEEE Int'l Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (May 2018), 116–122. https://doi.org/10.1109/SIMPAR.2018.8376280

[15] John Harwell and Maria Gini. 2019. Swarm Engineering Through Quantitative Measurement of Swarm Robotic Principles in a 10,000 Robot Swarm. *International Joint Conference on Artificial Intelligence (IJCAI)* (August 2019).

[16] Joshua P. Hecker and Melanie E. Moses. 2015. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence* 9, 1 (2015), 43–70. https://doi.org/10.1007/s11721-015-0104-z

[17] M. Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. 2008. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence* 2, 2-4 (2008), 121–141. https://doi.org/10.1007/s11721-008-0019-z

[18] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. 2013. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research* 32, 12 (2013), 1495–1512. https://doi.org/10.1177/0278364913496484

[19] Thomas H. Labella and Marco Dorigo. 2006. Division of Labor in a Group of Robots Inspired by Ants' Foraging Behavior. *ACM Trans. on Autonomous and Adaptive Systems (TAAS)* 1, 1 (Sept. 2006), 4–25.

[20] Kristina Lerman, Alcherio Martinoli, and Aram Galstyan. 2004. A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems. In *Swarm Robotics. SR 2004*, E. Şahin and W.M. Spears (Eds.). Vol. LNCS 3342. Springer Berlin Heidelberg. http://www.isi.edu/~lerman/

[21] Löic Matthey, Spring Berman, and Vijay Kumar. 2009. Stochastic strategies for a swarm robotic assembly system. *Proc. IEEE Int'l Conf. on Robotics and Automation* (2009), 1953–1958. https://doi.org/10.1109/ROBOT.2009.5152457

[22] Nils Napp and Radhika Nagpal. 2014. Distributed amorphous ramp construction in unstructured environments. *Springer Tracts in Advanced Robotics* 104, 2014 (2014), 105–119. https://doi.org/10.1007/978-3-642-55146-8_8

[23] James G. Oxley. 2006. *Matroid Theory (Oxford Graduate Texts in Mathematics)*. Oxford University Press, Inc., New York, NY, USA.

[24] Carlo Pinciroli et al. 2012. ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intelligence* 6, 4 (2012), 271–295.

[25] Giovanni Pini, Arne Brutschy, Mauro Birattari, and Marco Dorigo. 2011. Task Partitioning in Swarms of Robots: Reducing Performance Losses Due to Interference at Shared Resources. In *Informatics in Control Automation and Robotics*. Springer, 217–228.

[26] Giovanni Pini, Arne Brutschy, Gianpiero Francesca, Marco Dorigo, and Mauro Birattari. 2012. Multi-armed Bandit Formulation of the Task Partitioning Problem in Swarm Robotics. In *Swarm Intelligence. ANTS 2012*, M. Dorigo et al. (Ed.). Vol. LNCS 7461. Springer, Berlin, Heidelberg. https://link-springer-com.ezp2.lib.umn.edu/content/pdf/10.1007

[27] Giovanni Pini, Arne Brutschy, Marco Frison, Andrea Roli, Marco Dorigo, and Mauro Birattari. 2011. Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence* 5, 3-4 (2011), 283–304. https://doi.org/10.1007/s11721-011-0060-1

[28] Giovanni Pini, Matteo Gagliolo, Arne Brutschy, Marco Dorigo, and Mauro Birattari. 2013. Task partitioning in a robot swarm: A study on the effect of communication. *Swarm Intelligence* 7, 2-3 (2013), 173–199. https://doi.org/10.1007/s11721-013-0078-7

[29] F.L.W. Ratnieks and C. Anderson. 1999. Task partitioning in social insects. *Insectes Sociaux* 46, 2 (1999), 95–108. https://doi.org/10.1086/303256

[30] Christopher Rouff. 2007. TR: FS-07-06: Papers from the 2007 AAAI Fall Symposium. (2007), 112–115. https://doi.org/10.1007/s11721-010-0040-x

[31] Christopher Rouff et al. 2004. Properties of a Formal Method for prediction of Emergent Behaviors in Swarm-based Systems. In *Proc. 2nd International Conference on Software Engineering and Formal Methods*.

[32] Erol Şahin. 2005. Swarm Robotics: From Sources of Inspiration to Domains of Application. In *Swarm Robotics (LNCS 3342)*. Springer, 10–20.

[33] Amanda J.C. Sharkey. 2007. Swarm robotics and minimalism. *Connection Science* 19, 3 (2007), 245–260. https://doi.org/10.1080/09540090701584970

[34] William Thomas Tutte. 1959. Matroids and graphs.

[35] Hassler Whitney. 1935. On the Abstract Properties of Linear Dependence. *American Journal of Mathematics* 57, 3 (1935), 509–533. https://www.jstor.org/stable/2371182?seq=1

[36] Ryan K. Williams, Andrea Gasparri, and Giovanni Ulivi. 2017. Decentralized matroid optimization for topology constraints in multi-robot allocation problems. *Proceedings - IEEE International Conference on Robotics and Automation* (2017), 293–300. https://doi.org/10.1109/ICRA.2017.7989038

[37] Alan F. T. Winfield, Wenguo Liu, Julien Nembrini, and Alcherio Martinoli. 2008. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence* 2, 2-4 (Dec. 2008), 241–266. https://doi.org/10.1007/s11721-008-0018-0

[38] Alan F. T. Winfield and Jin Sa. 2005. On Formal Specification of Emergent Behaviours in Swarm Robotic Systems. *Science* (2005), 363–371.