Optimal Scalability-Aware Allocation of Swarm Robots: From Linear to Retrograde Performance via Marginal Gains

Simay Atasoy Bingöl,*† Tobias Töpfer,* Sven Kosub,* Heiko Hamann,*† Andreagiovanni Reina*†‡

***Abstract*—In collective systems, the available agents are a limited resource that must be allocated among tasks to maximize collective performance. Computing the optimal allocation of several agents to numerous tasks through a brute-force approach can be infeasible, especially when each task's performance scales differently with the increase of agents. For example, difficult tasks may require more agents to achieve similar performances compared to simpler tasks, but performance may saturate nonlinearly as the number of allocated agents increases. We propose a computationally efficient algorithm, based on marginal performance gains, for optimally allocating agents to tasks with concave scalability functions—including linear, saturating, and retrograde scaling—to achieve maximum collective performance. We test the algorithm by allocating a simulated robot swarm among collective decision-making tasks, where embodied agents sample their environment and exchange information to reach a consensus on spatially distributed environmental features. We vary task difficulties by different geometrical arrangements of environmental features in space (patchiness). In this scenario, decision performance in each task scales either as a saturating curve (following the Condorcet's Jury Theorem in an interference-free setup) or as a retrograde curve (when physical interference among robots restricts their movement). Using simple robot simulations, we show that our algorithm can be useful in allocating robots among tasks. Our approach aims to advance the deployment of future real-world multi-robot systems.**

***Index Terms*—multi-agent system, task allocation, collective decision-making, scalability, Condorcet's Jury Theorem, swarm robotics**

I. Introduction

Groups can outperform individuals, if appropriately sized. Task allocation plays a crucial role in both natural and artificial collective systems, enabling groups to work together efficiently. The cornerstone of social insects' success is their extraordinary ability to organize workers into groups specialized in performing one of the tasks needed for the colony's survival (e.g., foraging, nursing, or defense) [1]. Multi-robot systems can also exploit the parallelization of work by allocating teams of robots to different tasks (e.g., resource collection, transportation, and storage) [2], [3].

In such systems, ensuring efficient allocation of workers—animals or robots—to different tasks is crucial to optimize collective performance. In real-world applications, workers are provided in finite numbers and allocating them optimally across multiple tasks is non-trivial. Adding workers to a task could either improve performance, lead to diminishing returns, or even degrade overall performance if the resulting group is too large to operate efficiently. Given the finite set of workers, allocating more workers to one task requires reducing the number of workers assigned to other tasks. This tradeoff in resource distribution plays a crucial role in selective attention [4], [5], through which both individual and collective organisms allocate their resources (e.g., energy, time, or workers) to the most critical tasks [6].

In this work, we propose an efficient strategy for optimally allocating N agents among T tasks characterized by different scalability functions, defining how task performance changes for different numbers of agents allocated to it (see Fig. 1 for an overview of our method). Our analysis shows that as the number of agents and tasks increases, a combinatorial explosion prohibits to search exhaustively for the optimal allocation. We propose an algorithm that has reduced computational complexity (polynomial time) compared to the brute-force approach, yet generates the optimal robot allocation. We assume that tasks are spatially separated and task switching is costly. Hence, agents (e.g., robots) must be allocated offline, that is, before they are deployed and head to their operation areas to execute the tasks in parallel.

Similar allocation processes have been investigated in several disciplines, for example, in ecology, 'ideal free distribution' describes how animals distribute themselves across different habitats [7]–[9]; in swarm robotics, 'task allocation' algorithms enable robots to allocate themselves to tasks as a function of the task's requirements [3], [10]–[13]; and in game theory, 'hedonic games' model the splitting of players into coalitions [14], [15]. In the literature on multi-robot task allocation, it is generally assumed that tasks require a minimum number of agents $n_{\min}$ to achieve successful completion [13], [16], [17]. When this requirement is not met ($n < n_{\min}$), the task is considered unhandled. This leads to a binary modeling of task completion C, where the robots' task achievement is classified as completed ($C(n) = 1$ if $n \geq n_{\min}$) or not executed ($C(n) = 0$ if $n < n_{\min}$). Even though this abstraction simplifies analysis and algorithm design, it may overlook intermediate cases where already a single agent could implement some progress ($0 < C(1) \ll 1$). To address this previous limitation, we consider a more general task allocation scenario where task progress is determined by scalability functions $C(n)$, that is, task performance gradually varies with the number n of assigned agents.

Prior studies on resource and budget allocation have addressed diminishing returns using submodular optimization

* Department of Computer and Information Science, Universität Konstanz, Germany
† Centre for the Advanced Study of Collective Behaviour (CASCB), Universität Konstanz, Konstanz, Germany
‡ Department of Collective Behaviour, Max Planck Institute of Animal Behavior, Konstanz, Germany

techniques [18], [19], primarily to tackle scalability and strategic behavior in systems with limited resources. However, these approaches are restricted to settings with monotonic diminishing returns. In contrast, our work focuses on a broader class of concave scalability functions, including non-monotonic (retrograde) cases that fall outside the scope of standard submodular optimization frameworks.

Frequency-dependent selection (FDS) in evolutionary biology provides a complementary perspective on agent allocation, where individual payoffs depend on how many others adopt the same strategy, leading in the case of negative FDS to diminishing returns and equilibrium behavior. Examples include the classic evolutionary game known as producer-scrounger model, in which individuals can either search for resources themselves (producers) or exploit resources discovered by others (scroungers), scrounging payoffs decrease as more individuals choose it [20]. Ideal free distribution theory predicts that animals distribute across resource patches to equalize intake rates under crowding [21]. These mechanisms are similar to our task allocation framework, where we assume concave scalability functions that induce diminishing marginal gains as more agents are assigned to a task. Both evolutionary and game-theoretic models reach equilibria in which no unilateral reassignment improves performance, similar to equilibria in potential games and evolutionary stable strategies [22]. These equilibria are typically reached through decentralized adaptation over time. In contrast, our approach explicitly computes the globally optimal allocation in polynomial time under known scalability functions, which makes it particularly well suited for engineered multi-agent systems such as robot swarms.

Our algorithm generalizes to task allocation for any concave scalability functions (see Fig. 2), which are frequently found both in computational and swarm robotics systems, as both face similar scalability constraints [23]. In parallel computing, performance is limited by bottlenecks, such as memory bandwidth or CPU constraints, resource contention when multiple processes compete for shared resources, and process synchronization overhead, which slows execution due to coordination delays. Similarly, in swarm robotics, scalability is constrained by physical interference among robots and with the environment, competition for shared resources such as communication bandwidth or physical space, and coordination overhead to create an efficient collective behavior. Building on this similarity, we consider three types of scalability functions that are most found in the literature, which exhibit linear, saturating, and retrograde trends.

We validate our algorithm with multi-agent and robotics simulations. In our experiments, we model tasks as collective decision-making problems, which are a common application and benchmark in swarm robotics [24]. To make consensus decisions on the true environmental state, agents need to collect information, process it, and collectively select one option. As agents can only sample locally, they acquire inaccurate, incomplete, or non-representative environmental information. However, if we deploy several individuals who take and combine independent samples from the same area, we can reduce variance and improve decision accuracy.

Combining independent samples to increase the precision of an estimate is known as variance reduction [25]. Variance reduction is common in natural and artificial collective systems that exploit the plurality of the individuals to maximize the quality of sampled information on which the decision will be based [6], [26], [27].

Thus, our task allocation scenario can be seen as a form of *collective selective attention*, where the swarm improves the decentralized acquisition of environmental information by allocating individuals to a set of sampling tasks.

Collective performance can be improved by allocating more agents to more difficult tasks, that is, areas where agents make more frequent observation errors due to a higher variance in their individual observations. The intuition is that by investing more resources (i.e., more agents) into acquiring environmental information, observational variance is reduced. A similar variance-reduction technique has been observed in cognitive neuroscience studies of human perceptual decisions with two stimuli (equivalent to our spatial features). The participant's resource was time and the most efficient solution was to allocate more time to categorize the noisier (i.e., higher variance) stimulus [28]. Similarly, another study found that ants adaptively vary their nest-site selection strategy to allocate more attention (resources) on higher variance attributes, which are more difficult to be categorized [6]. In statistics, the idea of using sampling groups with sizes proportional to the stimulus variance is formalized as the importance sampling technique [29], [30].

In our collective decision-making scenario, agents combine social information through majority votes, hence we can measure the scaling of performance through the Condorcet's Jury Theorem (CJT). This theorem states that, in a binary decision problem, assuming each agent votes independently and has a greater tendency to vote for the correct rather than the incorrect decision, the probability of the majority decision being correct (i.e., the group accuracy) increases with the group size [31]. In addition, we test our algorithm in a series of simple robot simulations, indicating the relevance of our solution for real-world problems as seen in large-scale robot systems.

Our results show that task scalability functions are not the only relevant parameters in defining the optimal allocation of N agents to T tasks. Instead, the proportion of agents allocated to each task that maximizes collective accuracy is size-dependent. This means that the optimal allocation of resources is not a constant proportion of the swarm size, rather the relative team sizes are different when the swarm has fewer or more agents. This size-dependent allocation is only present in nonlinear scalability functions, where collective performance saturates to a maximum value when a large number of agents are assigned to a task.

Our main contributions are as follows: (i) we formally characterize the multi-task allocation problem and show the naive exhaustive approach has exponential algorithmic complexity; (ii) we propose a polynomial-time algorithm adapting marginal-gain optimization to compute the optimal allocation for any concave scalability function describing how performance depends on group size; (iii) we exploit this algorithm

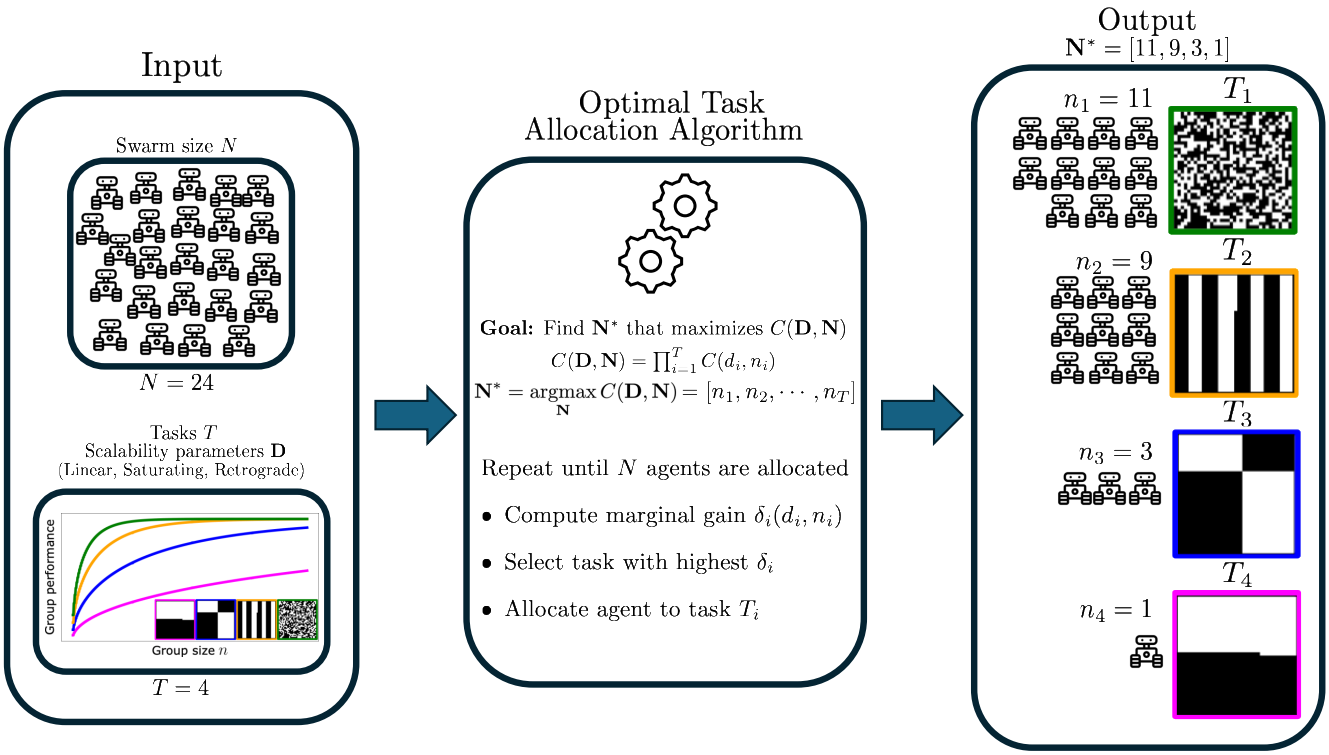

Fig. 1. Overview of the proposed algorithm, showing the steps from input to final agent–task assignment. In this representative example, a total of $N = 24$ agents are allocated across $T = 4$ tasks, each characterized by a distinct scalability curve $C(d_i, n_i)$. Based on marginal performance gains, the algorithm iteratively computes the optimal allocation as $N^* = [11, 9, 3, 1]$. In the illustrated collective decision-making scenario (collective sensing of environmental features shown as black and white floor tiles), the robot allocation is biased in favor of the easiest tasks, that is, tasks where robots can measure the correct state of the world with increased individual accuracy. In contrast, when swarms are large (resources to be allocated are abundant), the optimal solution is biased in favor of the more difficult tasks (Sec.V-B).

to study how optimal solutions change with swarm size N and number of tasks T, revealing counter-intuitive cases in which assigning more agents to the most difficult task is suboptimal for small swarms; and (iv) we validate the relevance of our approach by applying the algorithm to swarm robotics scenarios and demonstrating its effectiveness through extensive multi-agent simulations.

II. Multi-task Allocation Problem

We focus on the problem of allocating N identical agents to a set of $T < N$ tasks to achieve maximum collective performance. Each agent can only be allocated to a single task. Maximum performance is achieved by maximizing the performance in each task. Failure in one task means the entire mission fails.

Our algorithm relies on the following assumptions:

- The task allocation is computed by a central unit with full knowledge of individual task performance curves and agent availability.
- Agents are homogeneous; they have identical capabilities.
- Tasks are independent of each other; there are no interactions or interference effects between tasks.
- The scalability function of every task, defining how the task performance changes with respect to the number of allocated agents, is concave.
- Agents are allocated to tasks prior to execution; there is no task switching or reallocation during runtime.
- Each task is of equal importance in the allocation framework.

a) Collective performance: We formalize the multi-task allocation problem as follows: the vector $\mathbf{N} = (n_1, \ldots, n_T) \in \mathbb{N}_{>0}^T$, with $\sum_{i=1}^{T} n_i = N$ represents how the N agents are distributed among T tasks and $\mathbf{D} = (d_1, \ldots, d_T)$ represents the parameters for the scalability curves of all tasks. We measure the collective performance as the product of performances of all individual tasks:

$$C(\mathbf{D}, \mathbf{N}) = \prod_{i=1}^{T} C(d_i, n_i)\ . \tag{1}$$

Here, $C(\cdot)$ represents the scalability function, and $C(d_i, n_i)$ gives the individual task performance with scalability parameters d_i and n_i agents allocated to it.

Computing the collective performance as a multiplication of $C(d_i, n_i)$ assumes that all tasks are of equal importance. While this work does not explicitly address the scenario when some tasks are more critical than others, our approach can also be applied seamlessly to a collective performance function that includes different task importances through a weighted sum.

b) Goal: Our goal is to find the vector $\mathbf{N}$ that maximizes Eq. (1) for a given $\mathbf{D}$, which, in mathematical terms, is

formalized as follows:

$$\mathbf{N}^* = \underset{\mathbf{N}}{\operatorname{argmax}}\, C(\mathbf{D}, \mathbf{N}). \tag{2}$$

To model swarm performance for a variety of tasks, we follow how scalability is measured and modeled in distributed computing systems. Computational performance in parallel systems depends on how shared resources are managed and is influenced by bottlenecks and contention. Performance in swarm systems shows similar scalability constraints due to physical interference and communication overhead [23]. Building on this similarity, we model the different scalability functions (i.e., different tasks) to show linear, saturating, and retrograde trends (see Fig. 2). We summarize the structure of the proposed method in Fig. 1.

c) Linear scalability: Linear scalability refers to the ability of the system to increase its throughput (i.e., performance) proportionally to the number of resources (i.e., agents) added (blue curve in Fig. 2). We use Gustafson's Law [32] (GL) to model tasks which are highly parallelizable, displaying a (near-)linear relationship between the number of agents and the task performance. This relationship is caused by the assumption that most work of the task can be performed independently, in parallel, without any interference by each agent.

We formalize GL as

$$C_{\text{GL}}(\lambda, n) = n - \lambda(n-1)\ , \tag{3}$$

where n is the number of agents allocated to the task and $\lambda \in (0, 1)$ is the fraction of work that cannot be parallelized.

In swarm robotics, we can model tasks, such as foraging and surveillance, as GL because, typically, their performance (e.g., number of collected items or area coverage) scales linearly with the number of robots, assuming the environment and object availability/distribution grows proportionally [33].

d) Saturating scalability: Systems that exhibit saturating scalability increase their performance as resources are added, but eventually plateau at a maximum performance level, regardless of how many additional resources are introduced (orange curve in Fig. 2). In swarm robotics, collective decision-making usually shows saturating scalability [34], which we model using the Condorcet's Jury Theorem (CJT). The CJT gives the probability of the majority of a group of n decision-makers having the correct opinion in binary decisions. The theorem assumes that each agent makes its decision independently (i.e., without being influenced by other group members and acquiring uncorrelated external information). For example, robots deciding on the current state of the environment make independent observations from different locations. Every agent makes the correct personal decision with a probability p.

The CJT indicates that if the personal decisions are aggregated with a majority rule, that is, the collective decision is the one chosen by the majority of the agents, the average collective accuracy increases as

$$\begin{aligned} C_{\text{CJT}}(p, n) = \ & \textstyle\sum_{k=\lfloor \frac{n}{2}+1 \rfloor}^{n} \binom{n}{k} p^k (1-p)^{n-k} \\ & + \tfrac{1}{2}\binom{n}{\frac{n}{2}} p^{\frac{n}{2}} (1-p)^{\frac{n}{2}} (1 - n \bmod 2)\ , \end{aligned} \tag{4}$$

where the term $\binom{n}{k}$ represents the binomial coefficient and $\lfloor . \rfloor$ the floor operator (rounds down to the nearest integer). Eq. (4) is a sum with the last term being non-zero only when n is an even number. This term covers the probability that ties occur (i.e., an equal number of votes for both options), which are resolved randomly (50-50% decision).

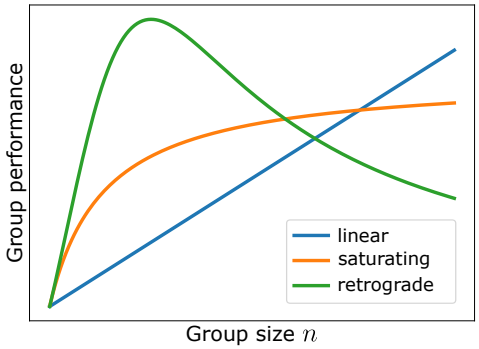

Fig. 2. Schematic plot of three types of scalability functions—with linear, saturating, and retrograde trends—describing the group performance as a function of group size. Each task, depending on the particular application and scenario, can scale differently.

e) Retrograde scalability: In systems exhibiting retrograde scalability, performance initially increases with the addition of resources, it eventually achieves a peak, and beyond that point, it degrades as more resources are added to the system (green curve in Fig. 2). We use the Universal Scalability Law (USL) to model tasks with retrograde scalability [35], using two parameters: the degree of contention α (i.e., competition for shared resources) and the lack of coherence β (i.e., overhead from maintaining consistency and coordination).

The USL for a swarm of size n is

$$C_{\text{USL}}(\alpha, \beta, n) = \frac{n}{1 + \alpha(n-1) + \beta n(n-1)}\ . \tag{5}$$

The USL can also represent linear, saturating, and superlinear scalability (e.g., for $\alpha < 0$) [36]. In this work, we only focus on the retrograde case by setting $\beta \geq 0$.

In swarm robotics, physical interference between robots affects performance as the swarm size increases, resulting in diminishing returns or performance degradation in larger swarms [23], [37].

f) Task scaling: The interpretation of the scalability parameter d_i varies depending on the scalability function. For task i with a linear scalability curve, d_i is equal to λ_i, which defines the slope of the line as $1-\lambda_i$. In the CJT-based models, d_i corresponds to the agent's probability p_i of making a correct decision in task i. If task i exhibits retrograde scalability (modeled using USL), d_i represents the pair of parameters α_i and β_i expressed as $d_i = (\alpha_i, \beta_i)$.

III. Modeling Majority Decision-Making

In this section, we show that collective decision-making tasks in swarm robotics can exhibit either saturating or retrograde scalability, depending on the scenario constraints and assumptions. When there is no physical interference between robots (or virtual agents), collective decision-making via majority rule is well captured by the saturating CJT function (Eq. (4)). In contrast, when robots experience physical

interference, swarm performance follows retrograde scalability and is well described by the USL (Eq. (5)).

We model these collective decision-making tasks using one of the standard benchmarking scenarios in swarm robotics [38]–[40]. Robots must reach a consensus on the prevalent color of the floor. Our robots operate in environments with a floor composed of black and white tiles. Despite having access only to local and incomplete information, they combine each other's estimates to reach a consensus on the predominant color.

a) Simulated environment: The simulated environment of each task has a size of 36×36 space-units (su). We also show that using environments with different sizes does not affect the results (see larger arena simulation results in Supplementary Fig. S1). The binary classification task is characterized by both the proportion of black and white tiles (fill ratio f) and the spatial distribution of colors (e.g., large clusters of tiles of one color). In particular, we consider five fill ratios $f \in \{0.51, 0.52, 0.53, 0.54, 0.55\}$ and four spatial distributions—checkerboard, striped, four rectangles, and halved environments—illustrated in Fig. 3. Our choice of environments is motivated by the benchmarking framework proposed by Bartashevich et al. [41] which introduced different spatial patterns to capture task difficulty in collective perception (see Supplementary Sec. 2 and Supplementary Figs. S2 - S3 for additional results and discussion on the specific choice of fill ratios). In the checkerboard environment, tiles of square size 1 su are placed uniformly at random. In the striped environment, bars of a uniform color with size 5×36 su are alternated (note that some bars are 1 su smaller to achieve the exact fill ratio f). In the four rectangles environment, the floor is divided into four large quadrants with a uniform alternated color in each of them. In the halved environment, each half has a single color (note that the partitioning line is not precisely at half of the environment but is set to match the desired fill ratio f).

b) Simulated robots: Robots are modeled as particles that move in a 2D environment. In a first set of experiments, we assume no collisions among robots—that is, they are treated as dimensionless points. This eliminates physical interference and allows for ideal scalability, where adding more robots never impedes their movement. In Sec. III-C, we also consider a more realistic scenario in which robots' physical embodiment affects one another's movement. In this second case, we observe qualitatively different scaling of task performance, exhibiting a retrograding trend when too many robots are deployed in the same confined environment.

Robots can move, exchange opinion votes with one another, and are equipped with proximity and ground sensors. The robots move at a speed of 0.1 su per timestep and rotate on the spot at an angular speed of 3° per timestep. Proximity sensors are three binary sensors to detect the presence of obstacles in front of the robot at a distance smaller than 4.5 su. One sensor faces the robot's heading direction (0°) and the other two at $\pm 45^\circ$. Proximity sensors are used to detect walls surrounding the environment and (in embodied simulations) other robots. Once an obstacle is detected, the robot initiates an avoidance maneuver, which consists of rotating in place until there is no obstacle in the detection range. Ground sensors allow robots to detect the color (black or white) of the tile beneath them. To avoid redundant sampling from the same floor tile, they track their covered distance and take two subsequent floor samples at a distance of 1 su apart. Given the robots' limited motion speed and sensing range, they often make inaccurate estimates of the state world. However, they can combine their opinions with others to improve group accuracy.

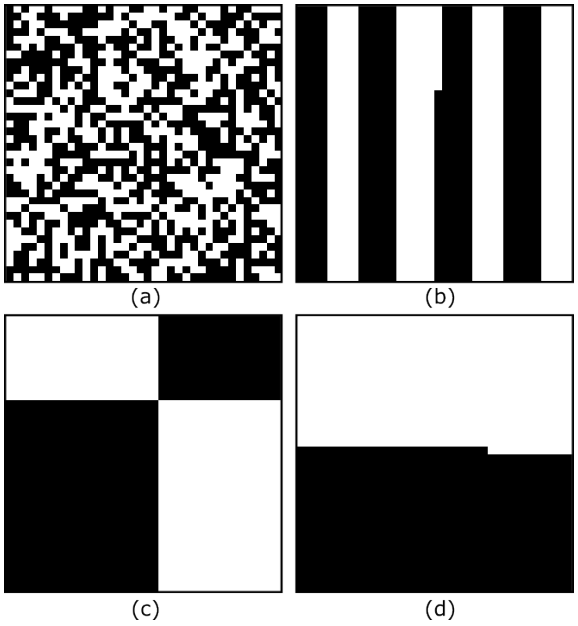

Fig. 3. Simulated binary classification tasks with fill ratio $f = 0.52$. The environment floor (sized 36×36 su) is composed of black and white tiles (sized 1×1 su). The robots are tasked with reaching a consensus on the most frequent color. We consider four spatial distributions: (a) checkerboard, (b) striped, (c) four rectangles and (d) halved environments.

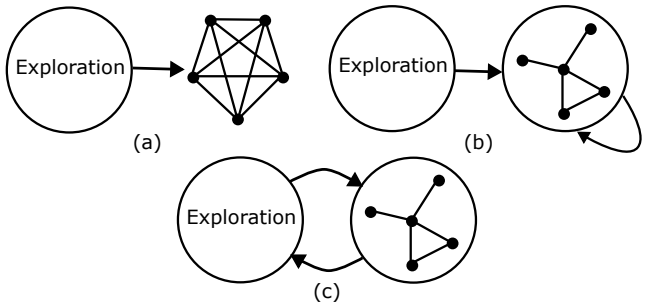

Fig. 4. (a) Centralized controller: All robots start in the exploration state, during which they gather environmental data and form their initial opinion. After exploring, they communicate through all-to-all interactions to reach a majority-based consensus. (b) Decentralized controller: Robots form their initial opinion similarly to the centralized controller. However, communication is restricted to local interactions and robots disseminate their opinions to neighbors until the entire swarm reaches a unanimous decision. (c) Iterative controller: This controller follows an iterative process of exploration and dissemination where they repeatedly combine personal and social information. Robots simultaneously disseminate their opinions, listen to neighbors, and continue exploring. The iterative process continues until the swarm reaches a unanimous decision.

A. Three Robot Controllers

We consider three alternative controller algorithms to enable the robots to perform the sampling tasks collectively: central-

ized, decentralized, and iterative (see Fig. 4). The centralized controller relies on all-to-all communication for a single-step majority decision, the decentralized controller spreads opinions locally until consensus emerges, and the iterative controller continuously refines opinions by integrating both individual and social information through repeated exploration and dissemination.

The **centralized** controller (Fig. 4a) is the closest application of the CJT. All robots first explore the environment through a random walk to form their initial opinion on the state of the world (prevalence of white or black tiles). The exploration time is different for each robot and is drawn from a uniform distribution with a mean of 1200 timesteps to model asynchronous robot operations. Then, each robot votes for their opinion to make a majority decision. Robots are in an all-to-all communication network. Hence, all robots have global information, aggregate the opinions of the entire swarm, and reach the same decision.

The **decentralized** controller (Fig. 4b) uses the same initial opinion formation process as the centralized controller; however, the communication range of each robot is restricted to 7.5 su. Robots' local interactions combined with their random walk in the environment lead to opinion spreading on a sparse time-varying communication network. Therefore, each robot only receives a part of others' opinions and consensus cannot always be reached in a single step. Hence, robots repeatedly make the majority decision only using the opinions of neighbors in their communication range. This process continues until the swarm achieves a unanimous consensus.

The **iterative** controller (Fig. 4c) extends the decentralized controller by iteratively combining personal and social information. At the start, there is no social information available yet, hence each robot—in the same fashion as the other controllers—forms its opinion through environmental exploration and shares it with its neighbors. Robots repeatedly explore and sample the environment by making (local) majority decisions where one of the processed votes is the result of their last individual observation of the environment. This majority decision forms the robot's new opinion which is then broadcast to its neighbors. This iterative process continues until the swarm reaches a unanimous consensus.

B. Majority Decision Results

We conduct a series of multi-agent simulations to compare the theoretical prediction based on the CJT with the results obtained from our simulations, where agents run one of the three robot controllers of Sec. III-A. These are simple robot simulations implemented using a Pygame-based simulator.[1] For the simulations studied in this section, agents have no volume and do not collide with each other (in contrast to Sec. III-C below).

As mentioned above, our simulated case study is the binary floor-color classification task which is a standard benchmark in swarm robotics [38], [40], [42]. We conduct 250 independent experiments for each environment and swarm size $N \in \{1, \ldots, 29\}$. We consider five fill ratios $f \in \{0.51, 0.52, 0.53, 0.54, 0.55\}$ for the checkerboard environment and four spatial distributions with the same fill ratio $f = 0.52$ (see Fig. 3). To compare the multi-agent simulations with the CJT predictions, we measure the individual accuracy probability p of any agent to make the correct environmental estimate during the allocated exploration time interval for each environment type. We measure p as the average of all individual agent estimates of our experiments (i.e., approximately 10^5 data points). Table I illustrates the relationship between the estimated individual accuracy p and the different types of environments.

[1]Swarmy: Robot swarm simulator https://github.com/tilly111/swarmy

TABLE I
INDIVIDUAL AGENT ACCURACY FOR THE CONSIDERED ENVIRONMENTS.

	estimated individual probability p			
fill ratio f	checkerb.	striped	four rect.	halved
0.51	0.5361	/	/	/
0.52	0.6017	0.5698	0.5402	0.5177
0.53	0.6603	/	/	/
0.54	0.7454	/	/	/
0.55	0.8069	/	/	/

As already documented in previous studies [38], [43], [44], the task becomes harder and the individual accuracy p decreases either by decreasing the fill ratio (pushing it closer to $f = 0.5$) or by increasing the spatial correlation of colors. In addition, the value p also depends on the behavioral and sensing capabilities of the robots, such as the type of random walk they perform or how frequently they sample the environment, as well as the coverage they achieve during exploration. Increased coverage of the arena results in more representative samples and thus increases the probability of making a correct decision. By decreasing the fill ratio, we reduce the difference in frequency between white and black tiles, therefore the probability p of making correct individual observations is reduced. It may seem less intuitive why we observe also a reduced probability p when we keep the fill ratio unchanged and only increase the spatial correlations of colors (clusters of uni-color tiles and hence more patchy environment). Agents taking local samples in highly correlated environments are less likely to sample representative environmental data. This spatial correlation analysis shows that embodied agents (e.g., robot swarms) are exposed to several factors that can cause inaccuracies in their individual observations, characterizing the task scalability curve.

Given that individual accuracies p_i for any considered task i are larger than the chance level ($\forall i, p_i > 0.5$), the CJT states that as the number n_i of robots assigned to task i increases, the collective accuracy of majority decisions (Eq. (4)) monotonically increases and asymptotically converges to one [45]. The results of Fig. 5(a) show that for all three considered controllers, there is a good agreement between the CJT predictions (solid lines) and the multi-agent simulations (dashed/dotted lines) for the five considered fill ratios $0.51 \leq f \leq 0.55$. Hence, we can use the CJT of Eq. (4) to predict how the group performance scales for larger group sizes $n \in [1, 500]$,

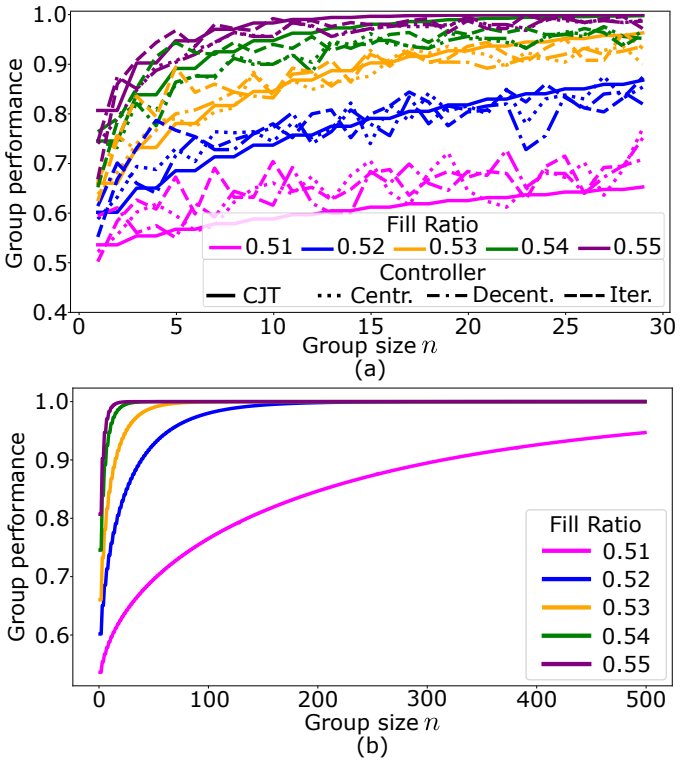

Fig. 5. Scaling of the group accuracy (y-axis) in making majority decisions for different group sizes (x-axis) in the standard checkerboard environment (see Fig. 3(a)). The different colors show results for different fill ratios. Solid line shows the CJT predictions and, in (a), the dotted, dash-dotted, and dashed lines show the multi-agent results for the centralized, decentralized, and iterative controllers, respectively. There is a good agreement between the CJT predictions and the multi-agent results. In (b), the CJT lines are extended to larger swarm sizes, using the same values of p as in (a).

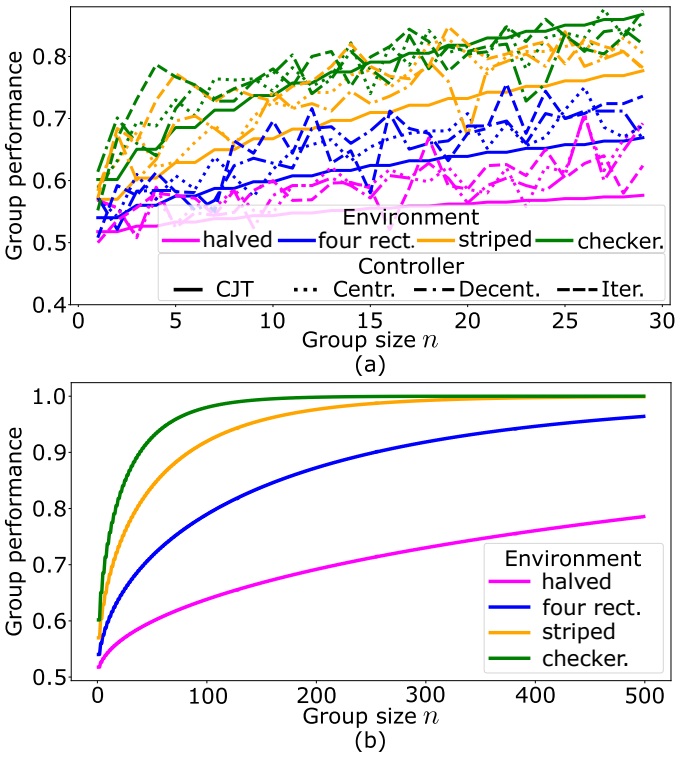

Fig. 6. Results for the three spatially correlated environments: striped, rectangles, and halved (see Fig. 3(b, c, d)). Scaling of the group accuracy (y-axis) in making majority decisions for different group sizes (x-axis). The different colors show results for different spatial distributions and the line types match the ones used in Fig. 5. As also shown in Table I, when the color correlation increases, the task gets harder and the accuracy decreases.

as shown in Fig. 5(b). Fig. 6 shows a similar analysis for the four different spatial distributions, showing again a good match between the theorem and the simulations. As there is an inverse relationship between the spatial correlation of the colors in the environment and the individual agent accuracy p (see Table I), this trend is also shown at the group level with lower collective performance in environments with high correlated spatial features (Fig. 6). Nevertheless, in any tested environment, group performance increased with group size n.

C. Simulations with Physical Interference

Above we assumed agents to be without volume that do not physically interfere with each other, meaning there were no collisions between agents restricting their movement (aside from interactions with walls). However, in real-world applications (e.g., applications of swarm robotics), such an idealized assumption cannot hold. Robots often operate in confined shared spaces where physical interference inevitably leads to collisions or even congestion that can significantly affect the robot system's efficiency. Therefore, it is not guaranteed that collective accuracy monotonically increases with swarm size. Beyond a critical swarm size, increasing the number of robots can actually reduce collective performance, as physical interference leads to traffic jams and bottlenecks [23]. Such a critical swarm size is normally expressed in terms of robot density, that is, the number of robots per space unit.

Given the importance of physical interference in practical applications, we include robot-to-robot physical collisions and preemptive avoidance maneuvers in a modified version of the multi-agent simulator described in Sec. III. Robots detect each other at a distance of 4.5 su and initiate avoidance maneuvers. As a side effect of these maneuvers—consisting of in-place rotations—robots temporarily stop collecting new environmental samples, as sampling only occurs when the robot has moved at least 1 su from its previous sample location.

We ran a series of experiments in the checkerboard environment to show the effect of physical embodiment. As an emergent effect of robot-to-robot interference, we observe retrograde scalability. Fig. 7(a) shows the multi-robot simulation results (dotted/dashed lines) with the fitted USL functions computed via nonlinear least squares fitting. Table II shows the fitted parameters for the various fill ratios $f \in \{0.51, 0.52, 0.53, 0.54, 0.55\}$. To normalize the curves to the interval $[0, 1]$, we introduce a proportionality constant k: $kC_{\text{USL}}(\alpha, \beta, n)$. We also provide the root mean square error (RMSE) as a metric to evaluate the goodness of fit. For small groups $n < 10$, where robot densities and physical interference are low, group accuracy increases with n. However, for groups of approximately $n > 10 \sim 15$ robots, group accuracy deteriorates as n increases due to detrimental physical inter-

TABLE II
ESTIMATED PARAMETERS FOR USL-BASED SCALABILITY CURVES FOR INTERFERENCE EXPERIMENTS

fill ratio f	α	β	k	RMSE
0.51	0.7971	0.0012	0.5194	0.0305
0.52	0.6376	0.0021	0.5270	0.0325
0.53	0.6750	0.0016	0.6093	0.0241
0.54	0.7089	0.0010	0.6814	0.0231
0.55	0.7526	0.0003	0.7201	0.0204

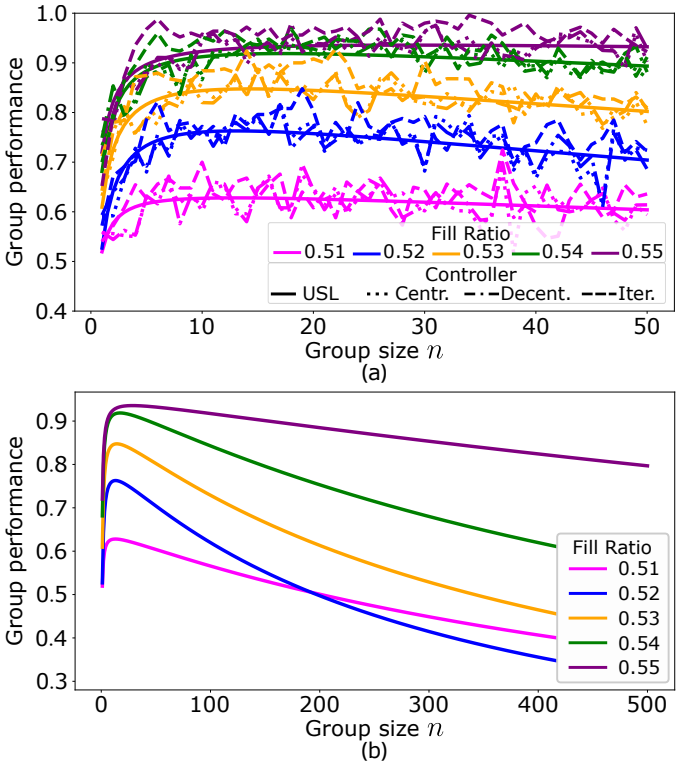

Fig. 7. Scalability curves for multi-agent simulations with robot-to-robot physical interference. In (a), the dashed/dotted lines show the group accuracy computed from 250 runs per condition in the checkerboard environment with fill ratio $f \in \{0.51, 0.52, 0.53, 0.54, 0.55\}$. The solid lines are the fitted USL curves with parameters obtained via nonlinear least squares fitting. In (b), the fitted USL curves are extended to larger swarm sizes, using the estimated parameters in Table II.

ference, leading to retrograde scalability. This decline in group performance is also captured by the fitted USL curves: as a positive parameter $\beta > 0$ (see Table II) indicates a retrograde scalability function. Note that the prediction of performance by the fitted USL shown in Fig. 7(b) tends to be pessimistic, because it is unlikely that the performance C would go below the fill ratio f ($C < f$). As observed in Fig. 7, although group accuracy decreases, the decline is slow and linear, indicating that robots tend to resolve interference and resume productive operation quickly [46].

IV. OPTIMAL TASK ALLOCATION ALGORITHM

Next, we present our approach (Alg. 1) to allocate N agents to T tasks so that overall swarm performance (Eq. (1)) is maximized. Alg. 1 is designed to find $\mathbf{N}^*$ from Eq. (2). Our underlying intuition is that of Cassey et al. [28], the optimal allocation of resources is a function of the variability of the accumulation of evidence. Here, each task is characterized by its respective performance curves.

Note that the total number of possible agent allocations grows exponentially with N and T. The number of possibilities of allocating N agents to T tasks is given by the ordered partition function, that is, the number of possibilities of writing an integer $N > 0$ as the sum of $T > 0$ positive integers in an ordered way. The ordered partition function for parameters N, T takes value $\binom{N-1}{T-1}$, which is asymptotically $\Theta\left(\frac{2^N}{\sqrt{N}}\right)$ for $T = \lfloor \frac{N+1}{2} \rfloor$. Therefore, a brute-force approach to determine the optimal allocation is generally infeasible.

We begin by introducing the general version of our task allocation algorithm and prove that it computes the optimal solution under the assumption that each task performance function is concave. The computational complexity of the algorithm is analyzed in Sec. IV-A, where we show that it runs in polynomial time. We then show that the algorithm can be further simplified—reducing its complexity—by tailoring specific steps (marginal gain function) to each scalability function type: linear, saturating, and retrograde.

A. The General Case

Depending on the performance function $C(d_i, n_i)$ for task i with scalability parameter d_i, we define the absolute gain function $\Delta(d_i, n_i)$ and the marginal gain function $\delta(d_i, n_i)$ as follows for $n_i > 0$:

$$\Delta(d_i, n_i) \coloneqq C(d_i, n_i + 1) - C(d_i, n_i) \,, \tag{6}$$

$$\delta(d_i, n_i) \coloneqq \frac{\Delta(d_i, n_i)}{C(d_i, n_i)} \,. \tag{7}$$

Given these definitions, we find how the addition of one agent to the i-th task changes the collective performance C:

$$C(\mathbf{D}, \mathbf{N} + e_i) = (1 + \delta(d_i, n_i))C(\mathbf{D}, \mathbf{N}) \,. \tag{8}$$

Alg. 1 follows a simple principle: at each step, the next available agent is assigned to the task that yields the largest relative improvement in collective performance; concavity ensures that these relative gains decrease as more agents are assigned, guaranteeing the global optimality of this greedy selection. Given the number of agents N, the number of tasks T, and the scalability parameters $\mathbf{D}$, the algorithm first assigns one agent to each task (line 1). Then it initializes an array of size T where each entry at respective position i represents the gain δ achieved by assigning an additional agent to task i. In each iteration (lines 5 to 11), the task offering the highest gain is selected for the assignment of one new agent. In line 7, we introduce a parameter $\varepsilon \geq 0$ to define a threshold beyond which assigning an agent to a task is no longer considered to be beneficial. This threshold can be interpreted as the maintenance cost of a robot, meaning that if the gain from assigning an additional agent to a task is lower than ε, the cost of deployment outweighs any benefits of the operating agent. If all marginal gains are lower than this threshold, all remaining agents are placed in a 'non-task' $T+1$

Algorithm 1 Optimal task allocation algorithm

Input:
number of agents N
number of tasks $T \leq N$
scalability parameters $\mathbf{D} = (d_1, \ldots, d_T)$
Output: best allocation $\mathbf{N}^* = (n_1, \ldots, n_T, n_{T+1})$

1: $(n_1, \ldots, n_T) \leftarrow (1, \ldots, 1)$
2: **for all** $i \in (1, \ldots, T)$ **do**
3: $marginal\_gain[i] \leftarrow \delta(d_i, 1)$
4: **end for**
5: **while** $\sum_{i=1}^{T} n_i < N$ **do**
6: $m \leftarrow \arg\max_i \ marginal\_gain[i]$
7: **if** $marginal\_gain[m] \leq \varepsilon$ **then**
8: $n_{T+1} \leftarrow n_{T+1} + (N - \sum_{i=1}^{T} n_i)$
9: **break**
10: **end if**
11: $n_m \leftarrow n_m + 1$
12: $marginal\_gain[m] \leftarrow \delta(d_m, n_m)$
13: **end while**

that represents an idle pool of agents and the loop terminates. The algorithm continues looping until all agents are assigned.

Theorem 1. *Alg. 1 is optimal for all scalability functions of task performance $C(d, n)$, such that the marginal gain $\delta(d, n)$ is decreasing in n.*

Proof. Let $\mathbf{N} = (n_1, \ldots, n_T)$ be the computed solution of Alg. 1 and let $\mathbf{M} = (m_1, \ldots, m_T) \neq \mathbf{N}$ be some solution with $C(\mathbf{D}, \mathbf{M}) \geq C(\mathbf{D}, \mathbf{N})$. Without loss of generality, assume $n_1 > m_1$ and $n_2 < m_2$. Since Alg. 1 selects in every step the task i maximizing the marginal gain and the gain function $\delta(\cdot, \cdot)$ is monotonically decreasing in the second parameter, we have

$$\delta(d_1, m_1) \geq \delta(d_1, n_1 - 1) \geq \delta(d_2, n_2) \geq \delta(d_2, m_2 - 1) \ .$$

Now consider the solution $\mathbf{M}' =_{\text{def}} (m_1 + 1, m_2 - 1, m_3, \ldots, m_T)$. Eq. 8 implies

$$C(\mathbf{D}, \mathbf{M}') = \frac{1 + \delta(d_1, m_1)}{1 + \delta(d_2, m_2 - 1)} C(\mathbf{D}, \mathbf{M}) \geq C(\mathbf{D}, \mathbf{M}) \ .$$

Hence, by repeating the above procedure, we can construct the solution $\mathbf{N}$ from $\mathbf{M}$ without decreasing the resulting collective performance. This implies $C(\mathbf{D}, \mathbf{M}) = C(\mathbf{D}, \mathbf{N})$ and the theorem follows. □

We analyze the running time of Alg. 1. In order to provide efficient access to the current maximal gain, the marginal gain values can be stored in a priority queue. In this case, the initialization in lines 2 to 4 needs $\mathcal{O}(T \log(T))$. The loop in lines 5 to 11 is executed $N - T$ times. In each iteration, the algorithm needs to select and update the current maximal gain. We define $D(n_i)$ as the time needed to compute the $\delta(d_i, n_i)$. Since this time is at most $D(N)$, we can use $D(N)$ as a general upper bound in our complexity analysis, that is, $D(n_i) \leq D(N)$. The extraction of the current maximum and insertion of the updated value is $\mathcal{O}(\log T)$. In total, Alg. 1 runs in time $\mathcal{O}(N D(N) \log(T))$. In particular, if the marginal gain can be updated in constant time, the running time simplifies to $\mathcal{O}(N \log(T))$.

B. Influence of Different Scalability Types

We analyze each of the three scalability functions—linear, saturating, and retrograde—in detail. For each, we examine how agents are optimally allocated and highlight the specific aspects or conditions required to guarantee the optimality of our algorithm.

a) Linear Scalability: Recall that the linear performance curve is defined as $C_{\text{GL}}(\lambda_i, n_i) = n_i - \lambda(n_i - 1)$ where $\lambda_i \in [0, 1]$ is the scalability parameter of task i. The corresponding marginal gain is $\delta(\lambda_i, n_i) = \frac{1-\lambda_i}{n_i(1-\lambda_i)+\lambda_i}$. Since this function is decreasing in n_i, we can apply Alg. 1 to compute the optimal task allocation. Maximizing $\delta(\lambda_i, n_i)$ is equivalent to selecting task i that minimizes the inverse $\frac{1}{\delta(\lambda_i, n_i)} = n_i + \frac{\lambda_i}{1-\lambda_i}$.

In the special case that $\lambda_i < \frac{1}{2}$ for all tasks i, we have $\frac{\lambda_i}{1-\lambda_i} < 1$, and the optimal solution corresponds to distributing agents evenly across tasks, with any remaining agents assigned first to those tasks with lower λ_i. In the general case, tasks with $\lambda_i \geq \frac{1}{2}$—tasks with a large fraction of non-parallelizable work—receive fewer agents. More precisely, consider two tasks with $\lambda_i > \lambda_j$. The algorithm only selects task i over task j if $\delta(\lambda_i, n_i) > \delta(\lambda_j, n_j)$, which is equivalent to $n_j - n_i > \frac{\lambda_i - \lambda_j}{(1-\lambda_i)(1-\lambda_j)}$. This means, if enough agents are available, the total difference in the number of assigned agents to both tasks is given by $\frac{\lambda_i - \lambda_j}{(1-\lambda_i)(1-\lambda_j)}$ (rounded up or down). Therefore, the total difference in the number of assigned agents between tasks i and j is bounded by a constant determined solely by their parameters λ_i and λ_j. As the total number of agents N increases, this fixed difference becomes relatively negligible, and the optimal allocation approaches a uniform distribution across tasks.

b) Saturating Scalability: We analyze the CJT curve $C(p_i, n_i)$ defined in Eq. (4), which models collective decision-making performance as a function of the number of agents n_i and the individual accuracy probability p_i. Notably, adding a second agent to make n_i even does not improve performance over adding a first agent to make n_i odd. One can formally verify that $C(p_i, n_i + 1) = C(p_i, n_i)$ for odd n_i. This implies that the marginal gain $\delta(p_i, n_i) = 0$ for odd n_i and hence, the marginal gain function is not monotonically decreasing when agents are added one at a time.

To resolve this issue and preserve optimality, we modify the allocation process to assign agents in pairs. This leads to a reformulated scalability function, defined as $\tilde{C}_{\text{CJT}}(p, k) =_{\text{def}} C_{\text{cond}}(p, 2k - 1)$, where $(k - 1)$ is the number of agent pairs assigned to a task. When using this function, if $n - T$ is odd, one agent remains unallocated and can either be assigned arbitrarily or left unallocated, as it does not improve the collective performance defined in Eq. (8).

Direct evaluation of the function $\tilde{C}_{\text{CJT}}$ is computationally costly and numerically unstable as n_i increases. To overcome this, we derive an incremental method, based on marginal gain, which requires constant computation times. We assume that all individual accuracy probabilities p_i satisfy $1/2 \leq p_i \leq 1$ in every task i.

By decomposing the binomial coefficients, the absolute gain function becomes:

$$\Delta(p,k) = \binom{2k-1}{k-1}(2p-1)(p(1-p))^k \ . \tag{9}$$

From this, we derive a recursive formula for the marginal gain $\delta(p,k)$, valid for $k > 1$:

$$\delta(p,k) = 2(2-\frac{1}{k})p(1-p)\frac{\delta(p,k-1)}{1+\delta(p,k-1)} \ . \tag{10}$$

This recurrence allows marginal gains to be computed efficiently, in constant time.

Finally, since $2(2-1/k) \leq 4$ and $p(1-p) \leq 1/4$, the marginal gain $\delta(p,k)$ is guaranteed to be monotonically decreasing in k. Therefore, Alg. 1 computes the optimal task allocation under saturating scalability.

c) Retrograde Scalability: Unlike the linear and saturating functions where collective performance increases monotonically with n_i, the USL function $C_{\text{USL}}(\alpha_i, \beta_i, x)$ of Eq. (5) reaches a maximum at $n_i = \sqrt{(1-\alpha_i)/\beta_i}$. This introduces a qualitative difference: adding more agents not only yields diminishing returns but eventually becomes detrimental to performance, as the marginal gain $\delta < 0$. In the discrete setting, the maximum is attained at either $n_i = \lfloor\sqrt{(1-\alpha_i)/\beta_i}\rfloor$ or $n_i = \lceil\sqrt{(1-\alpha_i)/\beta_i}\rceil$. To achieve maximum performance, no additional agents are assigned to a task once the marginal gain $\delta(\alpha_i, \beta_i, n_i)$ becomes negative.

As a result, some agents may remain unallocated if all tasks have reached their respective performance peaks (i.e., saturation threshold). In Alg. 1, line 7 checks whether all marginal gains are non-positive under the current swarm allocation. If so, the remaining agents are allocated to the idle pool (task $T+1$; see line 8), finalizing the allocation process.

The marginal gain $\delta(\alpha_i, \beta_i, n_i)$ is decreasing as long as the second derivative of C_{USL} is non-positive. If $\alpha_i \geq \beta_i$, the USL curve remains concave—without inflection points—within the interval $(0, \sqrt{(1-\alpha_i)/\beta_i})$, ensuring that Alg. 1 computes the optimal task allocation under retrograde scalability.

V. Empirical Results

We first verify the optimality of Alg. 1 by studying the allocation of robots across the collective decision-making tasks described in Sec. III. By considering only a limited number of tasks ($T = 2$ and $T = 3$) and agents ($N \leq 150$), we can exhaustively explore the solution space and compute the optimal solution. A small T allows a simpler visualization of the possible allocations including the optimal one. Then, in Sec. V-B, we demonstrate the possibility of scaling to a larger number of tasks (up to $T = 8$) and agents (up to $N = 2000$). Although our algorithm can efficiently handle significantly more tasks ($T \gg 8$), we limit the presentation to $T = 8$ for clarity. In all scenarios, we set $\varepsilon = 0$, meaning that there is no cost in deploying robots.

A. Task Allocation Results with a Limited Number of Tasks

We test Alg. 1 using results from our robot swarm simulations (Sec. III-B) to study the optimal allocation of $N = 30$ robots among $T = 2$ decision-making tasks exhibiting saturating scalability as shown in Fig. 5. Each task i is characterized by its decision difficulty, that is, the difficulty for the robot to make a correct individual estimate. As shown in Sec. III-B, agents' individual decision accuracy depends on environmental factors, such as fill ratio f_i and feature correlation (e.g., patches of tiles with the same color). We define task T_1 as either more difficult than task T_2 or equally difficult, with difficulty determined by the fill ratios ($f_1 \leq f_2$), that is, agents have lower or equal individual decision accuracy $p_1 \leq p_2$. We use the individual accuracy value p, obtained from robot simulations, to compute the group accuracy for task i using the saturating scalability curve $C_{\text{CJT}}(p_i, n_i)$ defined in Eq. (4), which closely fits the robot simulation data. The overall collective performance of the swarm of N agents—see Eq. (1)—is computed by multiplying the group accuracies across all decision tasks.

We present results for all possible allocations of the $N = 30$ agents across the $T = 2$ tasks (see Fig. 8). The collective accuracy shows an inverted U-shape that, with the peak position varying across task pairs—resulting in different maximum collective performances. Fig. 8 shows that the allocation yielding the highest collective performance (circle) always matches the result of Alg. 1 (cross). As expected, the optimal allocation consists shifts more agents towards the more difficult task, while in the symmetric case, the best solution is to divide the agents equally between the two tasks.

We extend the analysis to the allocation of $N = 30$ or $N = 150$ agents across $T = 3$ decision-making tasks. Task T_1 is always the most difficult, task T_2 has medium difficulty, and task T_3 is the simplest (fill ratios $f_1 < f_2 < f_3$). We show the results in Fig. 9(left) as color maps in ternary plots. Interestingly, the results for $N = 30$ (left column) deviate from the trend observed in the case of $T = 2$ tasks. Contrary to intuitive expectations, the collective performance is not maximized when the allocation is biased towards the most difficult tasks. In the plots of Fig. 9(left), this would correspond to a region of high collective performance (dark red) near the bottom right corner associated with task T_1 (fill ratio $f_1 = 0.51$), which is not observed.

In the results for swarm size $N = 150$ (Fig. 9, right), the color maps exhibit a clear shift toward T_1, aligning more closely with our initial intuition. As swarm size increases, the optimal allocation becomes increasingly biased toward the more difficult tasks. This shift results from the non-linearity of the CJT (Fig. 5(b)) with respect to group size n, which makes optimal allocation dependent on swarm size. These results highlight the importance of an efficient algorithm, as the optimal agent allocation must be recomputed for each swarm size, number of tasks, and the specific scalability functions associated with those tasks.

Fig. 9 also shows that the allocation of our algorithm (cross symbol) always matches one of the allocations leading to maximum collective accuracy (circle symbols). For the $T = 3$ tasks results, note that there are three optimal solutions and our algorithm always returns one of them. The presence of multiple equivalent optimal solutions is the consequence of the stepwise nature of CJT-based accuracy, where adding one

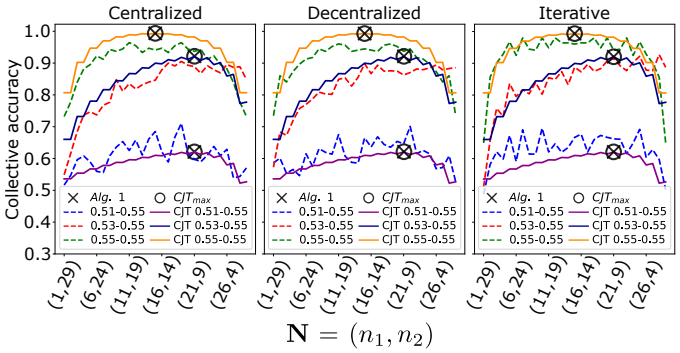

Fig. 8. Collective performance for two-task environments ($T = 2$) with fill ratios $(f_1, f_2) \in \{(0.51, 0.55), (0.53, 0.55), (0.55, 0.55)\}$. We set $f_1 \leq f_2$; hence, task T_1, with fill ratio f_1, is the most difficult (or equally difficult) task compared to task T_2 with fill ratio f_2. On the x-axis, we vary the agent allocation $\mathbf{N} = (n_1, n_2)$ to the two tasks. Solid lines are obtained from the CJT (Eq. (4)), and dashed lines show the results from robot simulations (250 repetitions per condition, data from Fig. 5). The left, center, and right plots show the results for the centralized, decentralized, and iterative controllers, respectively. The cross marker indicates the agent allocation computed by Alg. 1, while the circle marker represents the maximum performance predicted by the CJT. In all scenarios, the cross and the circle coincide.

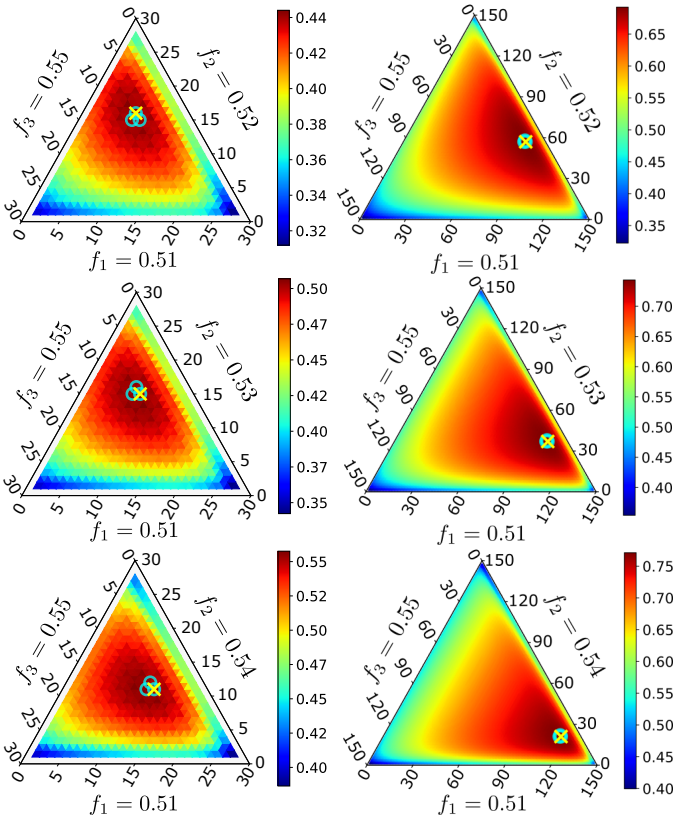

Fig. 9. Three scenarios for the allocation of agents to $T = 3$ collective decision-making tasks (T_1, T_2, T_3) with fill ratios $(f_1, f_2, f_3) \in \{(0.51, 0.52, 0.55), (0.51, 0.53, 0.55), (0.51, 0.54, 0.55)\}$. We set $f_1 < f_2 < f_3$, hence task T_1 is always the most difficult, T_2 has intermediate difficulty, and T_3 is the easiest. The color map shows the collective performance computed using Eq. 1 (see bar legends) for (left) small swarms with $N = 30$ agents and (right) large swarms with $N = 150$ agents. Blue circles mark the agent allocation yielding the highest performance, while the yellow cross indicates the allocation computed by our algorithm, which always coincides with one of the blue circles.

agent to an odd group does not change the collective accuracy (see Supplementary Fig. S4 for additional task allocation results).

Fig. 10 clearly illustrates the size-dependent shift in allocation, with swarm sizes varying as $N \in \{5, 10, 15, \ldots, 1000\}$. The crosses in Fig. 10 indicate the optimal agent allocations computed by our algorithm, shown as proportions of the swarm ($\frac{n_i}{N} \in [0, 1]$). As swarm size increases, agent allocation becomes increasingly biased in favor of the more difficult task (T_1 with $f_1 = 0.51$). For tasks exhibiting saturating scalability, the optimal allocation may not be a fixed proportion of the swarm size.

B. Task Allocation Results with Many Tasks

One of the key strengths of our algorithm is its scalability, which allows it to handle scenarios involving multiple tasks and large numbers of agents efficiently. We analyze three scenarios (corresponding to linear, saturating, and retrograde scalability) and present the optimal allocation results for swarm sizes up to $N = 2000$ agents, with $N \in \{25, 26, 27, \ldots, 2000\}$. Fig. 11 shows the results as stacked plots, where each layer indicates the proportion of agents allocated to each task (i.e., n_i/N) as a function of swarm size N. Colors correspond to the scalability functions reported in the respective insets, except for the gray layer, which represents agents allocated to the idle pool (i.e., not assigned to any task). Idle agents only appear in panel (c) for tasks with retrograde scalability.

In Fig. 11(a), we analyze the linear case, considering $T = 5$ tasks with scalability parameters $(d_1, d_2, d_3, d_4, d_5) \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Because d_1, d_2, and d_3 are smaller than 0.5, the optimal allocation consists of an equal distribution of agents among T_1, T_2, and T_3. In contrast, since d_4 and d_5 are larger than 0.5, fewer agents are allocated to T_4 and T_5 than the other three tasks (see also discussion in Sec. IV-B). This difference in optimal group sizes among tasks is relatively small in the number of allocated agents (about 6 to 8 agents). Because the stacked plot reports proportions of agents, this small difference is graphically accentuated for small swarm sizes. For example, for $N = 25$, the difference between n_1 and n_5 is 6 agents, corresponding to 24% of the swarm; instead, for $N = 2000$, the difference $n_1 - n_5$ is 8 agents, corresponding to 0.4% of N only. However, except for small swarms, the optimal group sizes n_i are approximately a constant proportion $n_i = 1/T$.

In Fig. 11(b), we analyze the saturating case (i.e., diminishing returns), considering $T = 8$ tasks with scalability parameters d_i measured from our robot simulations and reported in Tab. I as individual accuracies p_i. Similar to the results of Sec. V-A, we observe a size-dependent task allocation, meaning that the optimal allocation is not a constant proportion of the swarm size but changes as N increases. These results are a consequence of different values of p_i (see inset), leading to marginal gains decreasing differently across tasks. Fig. 11(b) illustrates these results as stacks changing height (y-axis) as a function of N (x-axis). When allocating small swarms, the largest groups are assigned to tasks with the highest p_i values (i.e., the easiest tasks) because they

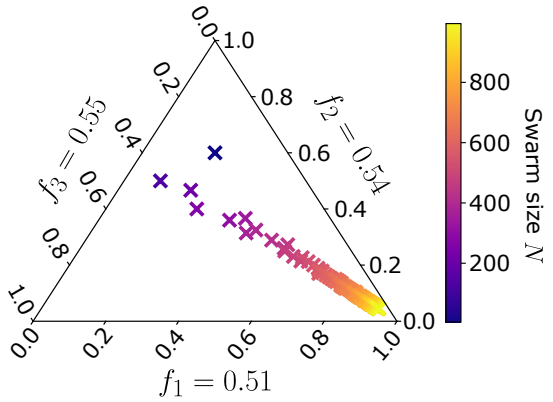

Fig. 10. Allocation of agents to $T = 3$ tasks (T_1, T_2, T_3) with fill ratios $(f_1, f_2, f_3) = (0.51, 0.54, 0.55)$. Crosses represent the optimal agent allocations computed by Alg. 1 for swarm sizes $N \in \{5, 10, 15 \ldots, 1000\}$, with each cross corresponding to a different swarm size. Axes of the ternary plot indicate the proportion of agents allocated to each respective task.

provide the highest marginal gain due to a steeper increase in performance compared to more difficult tasks with lower p_i. In contrast, when allocating large swarms, the more difficult tasks receive the largest proportion of agents because the easiest tasks have already approached the maximum performance (e.g., blue tasks in the inset) and provide minimal marginal gain compared to the harder tasks.

In Fig. 11(c), we analyze the retrograde case considering $T = 6$ tasks illustrated in the inset. A distinctive characteristic of retrograde tasks is the presence of a peak performance at a given group size n_i. Beyond this point, assigning more agents becomes detrimental and reduces the task performance, corresponding to a negative marginal gain. When all tasks have negative marginal gain, $\forall i, \delta(d_i, n_i) < 0$, or, more precisely, smaller than the robot deployment cost ε (assumed to be $\varepsilon = 0$ for simplicity), our algorithm stops allocating agents to tasks and all unallocated agents are assigned to the idle pool (gray stack in Fig. 11(c)). In the tested scenario, that point is reached at swarm size $N = 322$, beyond which the task allocations across the T tasks remain unchanged, and any additional agents are assigned to T_{T+1} only.

VI. Conclusion

Computing optimal task allocations for multi-agent systems is essential for maximizing overall performance. However, it can be challenging and computationally expensive, especially when dealing with large numbers of agents and tasks. In large-scale systems, evaluating all possible allocations and selecting the best one becomes computationally infeasible due to combinatorial explosion. We propose a polynomial-time algorithm for optimally allocating any number of agents to any number of tasks, applicable to any concave scalability function describing how performance depends on group size.

In our study, we model tasks by taking inspiration from the performance scaling principles of parallel computing systems. We run a series of analyses both to showcase the optimality of our algorithm and to study how optimal task allocations change for different task scalability functions and group sizes. We find that in linear and retrograde scalability, task allocation strategies are generally intuitive: linear tasks lead to an approximately even distribution of agents, while retrograde tasks require allocating agents up to their performance peak, leaving any excess agents unassigned. However, saturating performance functions—and retrograde tasks before reaching their peak—present unique challenges. The diminishing returns characteristic of these curves require a more refined allocation strategy.

In addition to multi-agent simulations, we also run a series of experiments with simulated robot swarms performing a benchmark case study in swarm robotics: collective perception of environmental features [38]–[40]. This is a type of collective decision-making scenario where agents select the predominant environmental feature through majority decisions. We model the different task difficulties by changing either the feature's frequency (i.e., fill ratio) or the feature's spatial correlation [44] (i.e., sizes of unicolor patches). These collective decision-making tasks scale with swarm size with a saturating curve—following the Condorcet's Jury Theorem (CJT)—when there is no interference among robots. We also show that when we include body collisions, physical interference among robots changes the scalability function to retrograde. The swarm robotics simulations showcase the relevance of the considered scalability functions for distributed systems and the importance of task allocation for improving the performance of large-scale systems.

We naively assumed that it would always be optimal to bias the allocation of agents in favor of the most difficult tasks. Our initial intuition was based on results from social science on human behavior [28], where the best allocation of resources (time) to each task is proportional to the task difficulty; however, our analysis contradicts these earlier findings. On the contrary, our theoretical and computational analyses indicate that this solution is only optimal when resources are abundant (i.e., in large swarms). In contrast, in small swarms, assigning more agents to the most difficult task can be suboptimal. These results highlight the importance of an efficient algorithm for task allocation because the optimal solution is not a linear rescaling of smaller-scale systems. Instead, the best allocation should be recomputed as system size, number of tasks, or task difficulties change.

Allocating agents to different tasks is relevant and important in both engineering and biology. For example, in swarm robotics, system efficiency can be improved by distributing the workload among the robots. Assuming different tasks are performed using different types of sensors, task allocation can also be exploited for cost-effective design of robot swarms, where sensors are allocated as a function of each task's requirements. Modeling this problem could also help in understanding biological systems, for example, the composition of mixed-species groups [47], in which animals of different species cooperate by contributing complementary sensing capabilities to the group (e.g., for predator detection).

The current study considers an offline allocation setting with homogeneous agents, centralized computation, independent tasks, and known scalability functions. While these assumptions enable optimality guarantees and polynomial-time com-

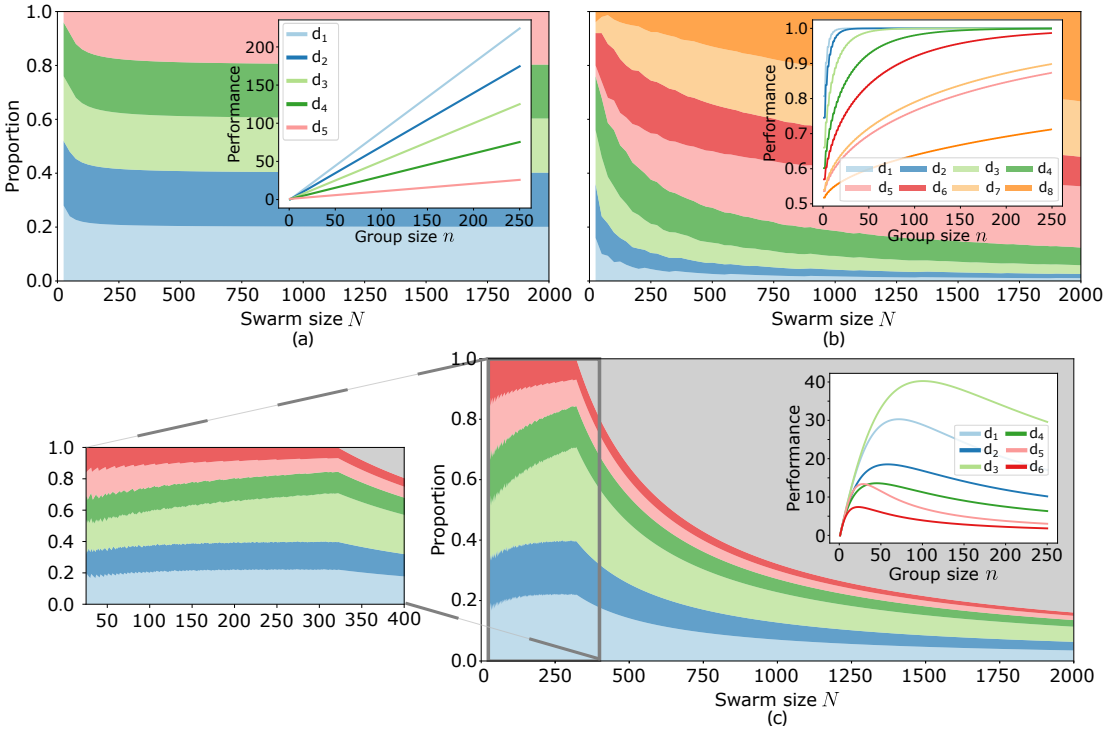

Fig. 11. Stacked plots with linear, saturating and retrograde scalability performance curves. The stacked plot represents the proportion of agents assigned to each task, with colors corresponding to the tasks shown in the line plot for consistency. The light-gray color in the stacked plot represents the portion of unallocated agents. Line plots show the individual performance of each task with respect to swarm size. (a) The allocation of agents to linear $T = 5$ tasks $(T_1, T_2, T_3, T_4, T_5)$ with $d_i = \lambda_i$, for $i \in \{1, 2, 3, 4, 5\}$ where $(d_1, d_2, d_3, d_4, d_5) \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. (b) The allocation of agents to saturating $T = 8$ tasks $(T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8)$ with $d_i = p_i$, for $i \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ where $(d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8) \in \{0.8069, 0.7454, 0.6603, 0.6017, 0.5361, 0.5698, 0.5402, 0.5177\}$ and line plots are presented in Fig. 5(b) and Fig. 6(b). (c)The allocation of agents to retrograde $T = 6$ tasks $(T_1, T_2, T_3, T_4, T_5, T_6)$ with $d_i = (\alpha_i, \beta_i)$, for $i \in \{1, 2, 3, 4, 5, 6\} \in \mathbb{Z}$ where $(d_1, d_2, d_3, d_4, d_5, d_6) \in \{(0.005, 0.0002), (0.02, 0.0003), (0.005, 0.0001), (0.03, 0.0005), (0.004, 0.0013), (0.05, 0.002)\}$.

plexity, they abstract away some challenges inherent to real-world swarm systems, including agent heterogeneity, stochastic task dynamics, partial observability, and communication constraints. A natural extension is to replace the predefined scalability functions with online performance models learned from locally observed task performance, enabling adaptive task allocation under uncertainty.

Future research can also extend the present study to include heterogeneity among agents and tasks. Our algorithm assumes homogeneous agents, where any agent contributes equally to a task, and it could be potentially generalized to optimally allocate swarms of heterogeneous agents—i.e., certain robots have better capabilities than others in performing the task (e.g., sampling the environment and making the correct decision [48]). Our algorithm can also be extended—by merely changing the marginal gain calculation step—to handle concurrently tasks with different types of scalability functions. These future extensions could broaden the applicability of our method to real-world applications with heterogeneous tasks and agents. Our framework applies to any concave scalability function. While we mainly focus in linear, saturating and retrograde functions, other types of concave functions (e.g., logistic growth) would lead to similar allocation patterns. The critical property is concavity, which ensures diminishing marginal gains and keeps the optimization computationally feasible. Non-concave functions could exhibit multiple local optima and would require different algorithmic tools, which we leave for future work.

We hope that our work not only provides a foundation for optimal task allocation in scalable multi-agent systems but also helps improve approaches for task allocation in swarm robotics and collective decision-making. There are many open questions about dynamically changing scalability functions, online task allocation, group sizes changing at runtime (e.g., due to broken robots), and a fully decentralized approach. Our presented insights may inspire cross-disciplinary research, bridging robotics, distributed computing, social science, and biological systems. We believe our approach is just the beginning of research on modern techniques for complex task allocation problems for the next generation of intelligent multi-robot systems, unlocking new possibilities for large-scale real-world deployment of robot systems.

ACKNOWLEDGMENTS

This work has been partially supported by the DFG under Germany's Excellence Strategy, EXC 2117-422037984 (H.H. and A.R.) and by the Hector Foundation II. We thank Samer Al-Magazachi for kindly providing us with the source code of his Python-based robot swarm simulator 'Swarmy.'

REFERENCES

[1] G. E. Robinson, "Chapter 77 - division of labor in insect societies," in *Encyclopedia of Insects (Second Edition)* (V. H. Resh and R. T. Cardé,

eds.), pp. 297–299, San Diego: Academic Press, second edition ed., 2009.

[2] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[3] S. Berman, A. Halász, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[4] A. C. Paulk, J. A. Stacey, T. W. Pearson, G. J. Taylor, R. J. Moore, M. V. Srinivasan, and B. Van Swinderen, "Selective attention in the honeybee optic lobes precedes behavioral choices," *Proceedings of the National Academy of Sciences*, vol. 111, no. 13, pp. 5006–5011, 2014.

[5] S. P. Tipper, G. M. MacQueen, and J. C. Brehaut, "Negative priming between response modalities: Evidence for the central locus of inhibition in selective attention," *Perception & Psychophysics*, vol. 43, no. 1, pp. 45–52, 1988.

[6] T. Sasaki and S. C. Pratt, "Ants learn to rely on more informative attributes during decision-making," *Biology Letters*, vol. 9, no. 6, p. 20130667, 2013.

[7] S. D. Fretwell and J. S. Calver, "On territorial behavior and other factors influencing habitat distribution in birds: Ii. sex ratio variation in the dickcissel (spiza americana gmel)," *Acta Biotheoretica*, vol. 19, no. 1, pp. 37–44, 1969.

[8] A. Kacelnik, J. R. Krebs, and C. Bernstein, "The ideal free distribution and predator-prey populations," *Trends in Ecology & Evolution*, vol. 7, no. 2, pp. 50–55, 1992.

[9] N. Quijano and K. M. Passino, "The ideal free distribution: Theory and engineering application," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 154–165, 2007.

[10] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[11] A. Kanakia, B. Touri, and N. Correll, "Modeling multi-robot task allocation with limited information as global game," *Swarm Intelligence*, vol. 10, pp. 147–160, 2016.

[12] C. Fleming and J. A. Adams, "Recruitment-based robotic colony allocation," in *Distributed Autonomous Robotic Systems: The 14th International Symposium*, (Berlin, Germany), pp. 79–94, Springer, 2019.

[13] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[14] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous hedonic game for task allocation in a large-scale multiple agent system," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1534–1548, 2018.

[15] A. Bogomolnaia and M. O. Jackson, "The stability of hedonic coalition structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[16] H. Aziz, H. Chan, Á. Cseh, B. Li, F. Ramezani, and C. Wang, "Multi-robot task allocation—complexity and approximation," in *Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 133–141, IFAAMAS, 2021.

[17] H. Aziz, A. Pal, A. Pourmiri, F. Ramezani, and B. Sims, "Task allocation using a team of robots," *Current Robotics Reports*, vol. 3, no. 4, pp. 227–238, 2022.

[18] Y. Bachrach and J. S. Rosenschein, "Distributed multiagent resource allocation in diminishing marginal return domains," in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1103–1110, 2008.

[19] T. Soma, N. Kakimura, K. Inaba, and K.-i. Kawarabayashi, "Optimal budget allocation: Theoretical guarantee and efficient algorithm," in *International Conference on Machine Learning*, pp. 351–359, PMLR, 2014.

[20] C. J. Barnard and R. M. Sibly, "Producers and scroungers: a general model and its application to captive flocks of house sparrows," *Animal Behaviour*, vol. 29, no. 2, pp. 543–550, 1981.

[21] R. S. Cantrell, C. Cosner, and Y. Lou, "Evolution of dispersal and ideal free distribution in populations with logistic growth rates," *Mathematical Biosciences and Engineering*, vol. 4, no. 2, pp. 317–330, 2007.

[22] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[23] H. Hamann and A. Reina, "Scalability in computing and robotics," *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1453–1465, 2021.

[24] H. Hamann, *Swarm robotics: A formal approach*, vol. 221. Berlin, Germany: Springer, 2018.

[25] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo methods*. Hoboken, NJ, USA: John Wiley & Sons, 2013.

[26] A. B. Kao and I. D. Couzin, "Decision accuracy in complex environments is often maximized by small group sizes," *Proceedings of the Royal Society B: Biological Sciences*, vol. 281, no. 1784, p. 20133305, 2014.

[27] Y. Khaluf and P. Simoens, "Collective sampling of environmental features under limited sampling budget," *Journal of Computational Science*, vol. 31, pp. 95–110, 2019.

[28] T. C. Cassey, D. R. Evens, R. Bogacz, J. A. R. Marshall, and C. J. H. Ludwig, "Adaptive sampling of information in perceptual decision-making," *PLOS ONE*, vol. 8, no. 11, p. e78993, 2013.

[29] P. W. Glynn and D. L. Iglehart, "Importance sampling for stochastic simulations," *Management Science*, vol. 35, no. 11, pp. 1367–1392, 1989.

[30] R. M. Neal, "Annealed importance sampling," *Statistics and computing*, vol. 11, pp. 125–139, 2001.

[31] M. d. Condorcet, *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Paris: Imprimerie Royale, 1785.

[32] J. L. Gustafson, "Reevaluating Amdahl's law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[33] T. Balch, "Reward and diversity in multirobot foraging," in *Proceedings of the AAAI 2000 Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information*, AAAI, 2000.

[34] R. Zakir, M. Dorigo, and A. Reina, "Miscommunication between robots can improve group accuracy in best-of-n decision-making," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9014–9021, IEEE, 2024.

[35] N. J. Gunther, "A simple capacity model of massively parallel transaction systems," in *Proceedings of the Computer Measurement Group (CMG) National Conference*, pp. 1035–1044, 1993.

[36] N. J. Gunther, P. Puglia, and K. Tomasette, "Hadoop super-linear scalability: The perpetual motion of parallel performance," *ACM Queue*, vol. 13, no. 5, pp. 46–55, 2015.

[37] J. Kuckling, R. Luckey, V. Avrutin, A. Vardy, A. Reina, and H. Hamann, "Do we run large-scale multi-robot systems on the edge? more evidence for two-phase performance in system size scaling," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4562–4568, IEEE, 2024.

[38] G. Valentini, D. Brambilla, H. Hamann, and M. Dorigo, "Collective perception of environmental features in a robot swarm," in *Proceedings of the International Conference on Swarm Intelligence (ANTS)*, pp. 65–76, Springer, 2016.

[39] J. T. Ebert, M. Gauci, F. Mallmann-Trenn, and R. Nagpal, "Bayes bots: Collective bayesian decision-making in decentralized robot swarms," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 7186–7192, IEEE, 2020.

[40] R. Zakir, M. Dorigo, and A. Reina, "Robot swarms break decision deadlocks in collective perception through cross-inhibition," in *Proceedings of the International Conference on Swarm Intelligence (ANTS)*, pp. 209–221, Springer, 2022.

[41] P. Bartashevich and S. Mostaghim, "Benchmarking collective perception: New task difficulty metrics for collective decision-making," in *EPIA Conference on Artificial Intelligence*, pp. 699–711, Springer, 2019.

[42] J. T. Ebert, M. Gauci, and R. Nagpal, "Multi-feature collective decision making in robot swarms," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 1711–1719, 2018.

[43] Q. Shan and S. Mostaghim, "Collective decision making in swarm robotics with distributed bayesian hypothesis testing," in *Proceedings of the International Conference on Swarm Intelligence (ANTS)*, pp. 55–67, Springer, 2020.

[44] P. Bartashevich and S. Mostaghim, "Multi-featured collective perception with evidence theory: tackling spatial correlations," *Swarm Intelligence*, vol. 15, no. 1, pp. 83–110, 2021.

[45] A. J. King and G. Cowlishaw, "When to use social information: the advantage of large group size in individual decision making," *Biology Letters*, vol. 3, no. 2, pp. 137–139, 2007.

[46] H. Hamann, T. Aust, and A. Reina, "Guerrilla performance analysis for robot swarms: Degrees of collaboration and chains of interference events," in *Proceedings of the International Conference on Swarm Intelligence (ANTS)*, pp. 134–147, Springer, 2020.

[47] E. Goodale, G. Beauchamp, and G. D. Ruxton, *Mixed-species groups of animals: behavior, community structure, and conservation*. San Diego, CA, USA: Academic Press, 2017.

[48] W. Hoeffding, "On the distribution of the number of successes in independent trials," *The Annals of Mathematical Statistics*, pp. 713–721, 1956.

Simay Atasoy Bingöl is a Ph.D. student at the University of Konstanz, Germany, and a member of the Centre for the Advanced Study of Collective Behaviour. She received her M.Sc. degree from Middle East Technical University in 2022. Her research interests include swarm robotics, collective decision-making, multi-agent systems, and bio-inspired algorithms.

Tobias Töpfer is a Ph.D. student in the Department of Computer Science at the University of Konstanz, at the Chair of Theoretical Computer Science and Quantum Informatics. He holds a master's degree in Mathematics from the University of Bonn. His research interests include combinatorial optimization, with a focus on graph theory.

Sven Kosub is a professor for Theoretical Computer Science and Quantum Informatics at the University of Konstanz. He received a Diploma degree in Mathematics from University of Jena, Germany, a Doctorate degree (Dr. rer. nat.) from University of Würzburg, Germany, and a Habilitation degree (venia legendi) from Technical University of Munich, Germany. His research interests include algorithms and complexity theory, with a focus on quantum computing methods for data science.

Heiko Hamann is a professor for Cyber-physical Systems at the University of Konstanz, Germany, and a member of the Centre for the Advanced Study of Collective Behaviour in Konstanz, Germany since 2022. His main research interests are swarm robotics, bio-hybrid systems, evolutionary robotics, and modeling of complex systems. He is author of the book "Swarm Robotics: A Formal Approach" and is editor-in-chief of the Swarm Intelligence Journal since 2023.

Andreagiovanni Reina is a Research Group Leader at the Centre for the Advanced Study of Collective Behaviour, a joint research centre of the University of Konstanz and the Max Planck Institute of Animal Behaviour, Germany. From 2020 to 2023, he was a Research Fellow in Collective Behaviour at the Interdisciplinary Institute for Artificial Intelligence (IRIDIA), Université Libre de Bruxelles, Belgium, funded by the F.R.S.-FNRS as a Chargé de Recherches. From 2015 to 2020, he was a Research Fellow at the University of Sheffield, UK. He received a Ph.D. in Applied Sciences from the Université Libre de Bruxelles and a M.Sc. degree in Computer Engineering from Politecnico di Milano, Italy. Since 2009, he has participated in several European research projects on distributed robotic systems. His research focuses on collective behavior and distributed decision-making in biological and artificial systems, combining approaches from dynamical systems theory, statistical physics, network science, multiagent modeling, and swarm robotics.