Collective control as a decentralized task allocation testbed

Annie S. Wu*
H. David Mathias†
Joseph P. Giordano‡
Arjun Pherwani‡

June 10, 2021

Technical Report CS-TR-21-01

Abstract

We present a collective tracking problem in which the agents in a swarm collectively control a tracker to follow a target moving in two dimensional space. Tasks are represented by the four cardinal directions and task demand refers to differences between the target and tracker locations in each of the directions. In any given timestep, each agent can choose to address task demand in a single direction. Tracker movement in a timestep is generated by aggregating the choices of all of the agents in the swarm in that timestep. Thus, the agents in a swarm collectively move a tracker in response to a target path.

This problem is representative of a general decentralized task allocation problem in which one or more tasks exhibit demand that is addressed by allocating an appropriate number of agents to each task. The tracking problem allows us to define dynamically changing task demands in a systematic way in the form of target paths. Because the variety of target paths that can be defined is essentially infinite, this problem allows us to define task allocation problems over a wide range of characteristics requiring different responses from the swarm. In addition to quantitative evaluation metrics, the actual two dimensional target and tracker paths provide an intuitive way to visualize system performance.

1 Introduction

In this paper, we present a testbed for studying decentralized task allocation problems in swarms and other large scale multi-agent systems (MAS). The decentralized task

*University of Central Florida, Orlando, 32816-2362, aswu@cs.ucf.edu
†University of Wisconsin – La Crosse, La Crosse, WI 54601, dmathias@uwlax.edu
‡University of Central Florida, Orlando, 32816-2362, {joseph.giordano,arjunp}@knights.ucf.edu

allocation problem (DTAP) is the problem of distributing a decentralized swarm of agents appropriately among one or more tasks in response to task demands. Task demands may be static, or may change dynamically over time. In the latter case, agents are expected to dynamically re-distribute in response to changing task demands. While the focus of our work and, consequently, the terminology in this paper refers to swarms and swarm robotics [2, 3, 4, 7, 48, 63], we expect many of the concepts discussed here to also apply to other types of MAS.

Swarms are multi-agent systems consisting of large numbers of decentralized agents that collectively work towards one or more mutual goals. Agents are independent, relatively simple, and typically have similar or overlapping capabilities such that the swarm as a whole is highly redundant. Agents may be physically embodied or virtual. The redundant and decentralized structure of swarms engender desirable characteristics of robustness, flexibility, and scalability [63]. Studies of computational swarms are closely intertwined with those of natural swarms, particularly, social insect societies, and many of the principles of how computational swarms work are based directly on principles or hypotheses of natural swarms. For example, Bonabeau et al's [5, 6] examination of a threshold approach to regulating division of labor in natural swarms has become foundational work for both biological and computational studies on swarm self-organization, and Dorigo et al.'s Ant Colony Optimization algorithm [19, 20] adapts concepts relating to stigmergy and path finding [26, 27] into a flexible and successful optimization algorithm. Our motivation for developing the testbed problem described in this paper is to build a system on which we can investigate the hypothesized effects of inter-agent variation that have been identified as relevant in biological swarm coordination [66] in the context of computational swarms[1]. A better understanding of the general principles of swarm coordination and self-organization will improve our ability to build more effective and robust computational swarms.

A key challenge in building decentralized swarms is how to design agents that can coordinate and self-organize intelligently. Such coordination refers to having enough agents acting on each task that needs attendance at any given time. Without centralized control, each agent acts independently. Although each agent can individually attempt to make the best decision based on its observation of the current state, the fact that other agents may also be acting means that the "current state" to which agents are responding is a moving target. In addition, even in problems where task stimuli are global, agents must have the ability to distribute themselves among different tasks to address all demands. Thus, agents in a swarm must be able to respond differently to global stimuli, and not act in lockstep, for a swarm to be practically useful.

There is a significant body of work on the application of swarms and MAS to DTAPs. Examples of application areas studied include foraging [1, 9, 13, 37, 38, 39, 40, 51, 54, 71], paintshop scheduling [11, 15, 35], mail processing [25, 58], job shop scheduling [50], logistic resupply [24], and tracking [69]. In addition, studies have also been conducted on abstract task allocation problems [16, 17, 21, 32, 49]. A potential limitation of DTAP studies that are focused on a particular domain is that the characteristics of the problem domain may limit the DTAP characteristics that can be studied,

[1] Specifically, we examine biological hypotheses discussed in [66] on the effects of variation in response threshold [43, 69], variation in response probability [45, 70], and variation in response duration [22] on swarm stability and robustness.

which may in turn limit extent and the generalizability of the knowledge gained. DTAPs exist in many forms. For example, DTAPs may vary in the number of component tasks, the number of task demands that can change, the number of task demands that change at a time, the relative ranges of possible change, the severity of the change over time, etc. A DTAP may consist of task demands that change gradually over time, tasks demands that change abruptly, or a combination of both. These and other features can influence the effectiveness of a given coordination strategy. In order to understand how to design swarms that can address different types of DTAPs, it would be helpful to have a testbed on which a range of DTAP characteristics may be investigated purposefully and systematically.

We propose a general, scalable collective control problem, which we cast as a tracking problem to allow easy visualization, that can be used to study many aspects of DTAPs. While there are already multiple platforms available for studying swarm behavior, most of the current platforms are agent-centric in that they are designed to provide flexibility in the design, creation, and study of the agents that make up a swarm. The testbed that we present here is problem-centric. Our goal is to provide a platform on which a variety of DTAP problems can easily be defined and studied. The collective tracking problem is representative of a general DTAP because it consists of multiple tasks with time-varying demands that can be satisfied by the collective action of one or more agents from a swarm. The demand of a task at any given instant of time defines the number of agents that need to act on that task at that time. As task demands change over time, agents are expected to adapt their actions accordingly. The paths that define the collective tracking problem allow for systematic variation of problem characteristics as well as an intuitive way to evaluate swarm performance. This problem is based on the process by which honeybees collectively perform thermoregulation of nest temperature [30, 66], extending the one dimensional temperature regulation problem into a two dimensional tracking problem.

We begin by reviewing existing testbed platforms for studying swarms. We then describe our problem testbed, illustrate how this problem can be used to define DTAPS and vary their characteristics in a systematic way, and discuss analysis resources that are available in the system.

2 Background

The complexity of swarms has resulted in the development of multiple platforms for studying swarms and swarm behavior. These platforms are important as they provide testbeds that allow for proof of concept, repeatability of experiments, and benchmarking of results. They may also allow researchers to isolate particular swarm characteristics for study. In addition to agent coordination and task allocation, other example characteristics include specialization and division of labor [10, 38], time dependent coordination (sequential tasks) [9, 53, 54, 55, 56], and stability and minimization of task switching [68]. Platforms include both hardware systems that allow for experimentation on swarms of physical agents and simulations that allow for the study of virtual swarms.

2.1 Hardware Systems

Physical robot systems used and/or developed for swarm research vary significantly in size, capabilities, and cost. At one end of the spectrum is the Kilobot [62]. The primary objectives for development of Kilobot are low cost and easy scalability of swarms. A Kilobot is 3.3 cm in diameter and 3.4 cm tall with three rigid legs. Locomotion is via two vibrating motors. Communication between Kilobots, within 7 cm of each other, is via a single LED (send) and a color light sensor (receive). Proximity can be approximated by evaluating signal strength. No other sensing is included. In addition to limited communication, Kilobots are capable of moving forward and turning, allowing for reasonably intricate paths. As the name suggests, researchers have created Kilobot swarms of as many as 1024 robots. The Kilobot is produced by K-Team.

The Khepera IV [31] is significantly larger, more capable, and more than 20 times the cost of the Kilobot. It measures 14.0 cm in diameter and 5.8 cm tall. It is equipped with eight infrared sensors for obstacle detection, four for fall avoidance and line following and a three-axis accelerometer. Five ultrasonic sensors with a range of up to two meters are used for long-range object detection. Communication is via WiFi, Bluetooth, a camera, a microphone, and a speaker. A number of extensions are available, including a sophisticated gripper.

At 7.0 cm in diameter and 5.0 cm tall, the e-Puck [14] is somewhat larger and significantly more capable than the Kilobot but smaller and less expensive than Khepera IV. Developed as an education tool for swarm robotics, and engineering education more generally, it incorporates infrared sensors for proximity, an array of microphones allowing triangulation of a sound source, a camera, and an accelerometer. Two processors are onboard: one general purpose and one for digital signal processing. Output is via a number of LEDs and a speaker. The e-Puck is wheeled allowing much higher movement speed than the Kilobot. Additional modules can be attached to an e-Puck, perhaps most importantly for swarm research, a radio communication module. The cost of the e-Puck falls between that of the Kilobot and Khepera IV.

Two other robot platforms, Swarm-Bot and Swarmanoid, are designed for inter-robot collaboration, not only in terms of communication but also physical capabilities. Swarm-Bot is an integrated hardware and simulator package [47]. A Swarm-Bot consists of a homogeneous colony of s-bots, a puck-style robot with a diameter of 11.6 cm. Locomotion is via *treels*, a tread and wheel hybrid. The goal of Swarm-Bot is to explore the ability of self-assembling robot colonies to overcome obstacles such as rough terrain and gaps that individual robots would be unable to traverse. s-bots achieve this by physically linking, forming a chain that allows the robots to push and pull each other as needed. The swarmbot3D simulator provides realistic simulation of several s-bot variants. We are unable to find commercially available s-bots.

Swarmanoid is a heterogenous swarm of robots with task-specific physical characteristics [18]. As originally presented in 2013, Swarmanoid consists of three robot types: footbot, handbot, and eyebot, all based on the ASEBA modular design architecture [42]. Footbot, a treel-based puck with integrated gripper, includes infrared sensors for ground and obstacle detection and a force sensor to measure pressure when pushing. Handbot includes grippers that are larger than that on footbot. In addition, it includes a magnetic "gripper" to aid in climbing, as well as a suction cup and rope launcher. It

includes a camera for sensing. Handbot has no locomotion capability. Instead, it depends on footbots to push it as needed. Eyebot is an autonomous quadcopter robot that provides "eyes in the sky" for the other robot types. To maintain a fixed position despite the positional drift that is common for quadcopters, eyebots rely on an additional eyebot mounted at a fixed location on the ceiling. All three robot types share infrared and radio capabilities for communication. The Swarmanoid system uses the ARGoS simulator [52].

2.2 Software Systems

Although experimentation with physical robots is desirable because it more closely models actual swarm applications, simulators offer a number of advantages. Perhaps the most significant of these is cost. A simulation with hundreds of robots incurs little, if any, additional cost over a simulation with just a few. Clearly, the same is not true for experiments performed with physical robots. In addition, the ease and speed with which experiments can be performed is greatly improved with simulators. For example, resetting tens or hundreds of physical robots to begin a new run may take substantial time whereas reseting a simulator is likely to be trivial.

In the following discussion of swarm simulation platforms, we partition simulators into three categories: general robot simulators, specialized robot simulators, and general agent simulators. We highlight only a selection of examples in each category as there are too many to allow an exhaustive discussion.

General robot simulators can model a variety of robots and problems. The Stage simulator is capable of representing hundreds of robots that are accessed via the Player device server [23]. Player can also control physical robots, providing a uniform interface whether working in a simulated or real environment. One drawback of Stage is that the physics simulation is limited. For example, acceleration is not modeled. Gazebo, based on Player/Stage, adds a high-fidelity 3D environment; however, the computational resources required for the physics severely limits the number of robots that can be simulated [36]. Webots provides accurate models of popular robots such as Khepera and e-Puck, as well as some RoboCup models [46]. Originally a commercial product, it is now open source. USARSim is a 3D simulator based on the Unreal Engine game development platform [12]. Originally developed to simulate urban search and rescue, it is capable of modeling large worlds. It includes models of many robots and allows users to create their own models. CopelliaSim (formerly V-REP) simulates multiple heterogeneous robots, however, its focus is on simulation and control rather than swarms [61].

A number of simulators are specialized in some way, typically for a particular robot or problem. Kilombo is an easy to use simulator capable of simulating swarms consisting of thousands of Kilobots [29]. Due to the very slow movement of physical Kilobots, Kilombo simulations can improve the speed of runs by a factor of up to 100. ARK, Augmented Reality for Kilobots, is a hybrid system that allows physical Kilobots to operate in a virtual environment [59]. It provides location and state information to Kilobots through virtual sensors that are beyond the capabilities of the physical robots. Through virtual actuators, robots can change the state of the environment and these changes can be sensed by other Kilobots in the system. Roborobo is a lightweight, fast,

open source simulator for large-scale experiments. It presents a 2D world for Khepera and e-Puck robots. It has been used extensively to explore environment-driven evolutionary adaptation in swarms [8]. In an example of a highly specialized simulator, Deepbots extends Webots for deep reinforcement learning through incorporation of the OpenAI Gym interface [33]. swarmbot3D and ARGoS are simulators that are part of large projects that incorporate development of physical robots with implementation of a simulator. swarmbot3D, part of the Swarm-Bot project, was developed to work with s-bot, an innovative puck robot created for cooperative behaviors at the hardware level. ARGoS provides real-time simulation of large heterogeneous swarms and is capable of simulating thousands of robots. It is part of the ambitious Swarmanoid project but works with e-Puck robots in addition to footbot, handbot, and eyebot. ARGoS allows use of multiple physics engines which improves speed.

Several simulators have been developed to explore complex adaptive systems, such as large swarms, independent of robotics. StarLogo [60] and NetLogo [65] provide interfaces that extend the Logo programming language, best known as a visual aid for teaching children to program. They allow the user to describe agent actions and replicate them to create a swarm. The agents act in a user-defined environment. The use of Logo somewhat limits the capabilities of these systems. MASON is a widely-used open source, very general, agent-centric simulator developed in Java [41]. Developing models requires programming in Java as well. Because it is agent-centric rather than robot-centric, it allows researchers to focus on general agent and swarm behaviors rather than solely on robot behaviors. Development of MASON continues and now includes a distributed version known as Distributed MASON.

2.3 Problem-centric platforms

The platforms described above are largely swarm-centric, focusing on agents and swarm composition. Our testbed is problem-centric, using a single problem to explore the effects that the different task demands, created by a variety of problem instances, have on decentralized task allocation. Several other problem-centric testbeds exist. In some cases they include some degree of simulation, while in others they simply define a problem.

Among the earliest systems in this category is Tileworld [57]. This single-agent environment consists of a 2-D grid in which there exist obstacles, tiles, and holes. The agent must attempt to fill holes by pushing tiles into them. Packet-World is a multi-agent 2-D grid problem in which agents must deliver colored packets, which are scattered throughout the environment, to destinations of the same color [67]. Agent actions include picking up packets and putting them down, carrying a packet, or doing nothing. Communication allows agents to request information or establish collaborations. The implementers have used Packet-World to investigate perception, synchronization, forms of collaboration, and adaptability.

Used as a testbed to demonstrate the capabilities of ARGoS, the area coverage problem is a deployment problem in which agents must self-deploy to cover an unknown, partitioned space [28]. The problem is framed as deployment of a mobile sensor network by a swarm with distributed sensing capability. The approach uses virtual potential fields to repel agents from each other and from obstacles.

One popular system combines a specific problem, a purpose-built simulator, and support for physical robots all overlaid with an annual competition. RoboCup is a robotic soccer competition with a name derived from the quadrennial World Cup [34]. While the humanoid robot version of the competition garners much of the attention, the RoboCup Soccer Simulator provides a low cost of entry platform for swarm researchers. The problem is sufficiently complex to allow exploration of a number of swarm characteristics, including task allocation, task decomposition, specialization, sequential tasks and dynamic adaptation. Built using the RoboCup simulator, RoboCup Rescue is a disaster rescue simulation developed with the goal of creating a real-world capable system [64].

3 Collective tracking problem

The collective tracking problem consists of two main elements: a target that moves in a two-dimensional plane along a user-specified path and a tracker that is collectively controlled by a swarm. The target moves at a fixed velocity for a specified number of timesteps. The agents in the swarm collectively control the tracker's movement, with a goal of minimizing the distance between the target and tracker at any given time. Though the system is currently implemented for two dimensions, extending to higher dimensions is straightforward. We note that dimensionality greater than three highlights that, while this testbed can model a physical system, it can also model more abstract DTAPs.

In each timestep, the target location is given by $x_g(t), y_g(t)$ and the tracker location is given by $x_k(t), y_k(t)$. The difference between the target and tracker locations along the x and y axes are given by $\Delta x(t) = x_g(t) - x_k(t)$ and $\Delta y(t) = y_g(t) - y_k(t)$, respectively. $\Delta x(t)$ represents task demand to push east in timestep t; $-\Delta x(t)$ represents task demand to push west; $\Delta y(t)$ represents task demand to push north; and $-\Delta y(t)$ represents task demand to push south. Negative demand is considered to be zero demand. Note that for this problem, there will be at most two tasks with non-zero demand at any given time. This restriction is due to the fact that movement in a 2D plane is defined by a vector composed of at most two directional components.

In each timestep, t, the target moves a distance of L along a specified path. Each agent selects one of five possible tasks: *push_north*, *push_east*, *push_south*, *push_west*, or remain idle. All of the agents that select to push in a given direction contribute to addressing the task demand in that direction. Let N be the total number of agents in the swarm and $n_i(t)$ be the number of agents pushing in direction $i : i \in \{N, E, S, W\}$ in timestep t. Let R be the *step ratio*, which specifies the maximum distance that the tracker can move in one timestep relative to the target's step length; that is, if $R = 2.0$, the tracker can move twice as far as the target in one timestep if all agents push in the same direction. The distance, $D_i(t)$, that the tracker will move in direction i in time t is

$$D_i(t) = \frac{n_i(t)}{N} \times R \times L$$

This movement is executed in each of the four directions to complete the tracker's movement in timestep t.

As with any other DTAP, performance may be measured by directly comparing the task demands and agent responses in each timestep. This problem provides an additional, more intuitive, evaluation metric in the two dimensional traces of the target and tracker paths. Two quantities define a swarm's success for the tracking problem:

Goal 1 *Minimize the average positional difference, per time step, between the target location and the tracker location.*

Goal 2 *Minimize the difference between the total distance traveled by the target and the total distance traveled by the tracker.*

To understand the need for both of these goals, we present the following scenarios. We first motivate the need for average positional difference. Consider a target that moves 500 distance units east while the swarm moves 500 distance units west. The path length difference is 0 though the tracking is not at all successful. To motivate the need for path length difference, suppose again that the target moves 500 units east but that the tracker is moved in a zigzag pattern crossing the target path in each timestep. The average positional difference is low but the path traveled by the tracker is much longer than that of the target.

4 Target paths define task demands

The key contribution of this testbed is the use of *target paths* to define task demands. Because the goal of the swarm is to minimize the distance between the target and tracker, the task demand $\tau_i(t)$ in each direction $i \in \{N, E, S, W\}$ in a given timestep t is defined as:

$$\tau_N(t) = \begin{cases} \Delta y(t) & \text{if } \Delta y(t) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \tau_E(t) = \begin{cases} \Delta x(t) & \text{if } \Delta x(t) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_S(t) = \begin{cases} -\Delta y(t) & \text{if } \Delta y(t) < 0 \\ 0 & \text{otherwise} \end{cases} \qquad \tau_W(t) = \begin{cases} -\Delta x(t) & \text{if } \Delta x(t) < 0 \\ 0 & \text{otherwise} \end{cases}$$

The shape or trajectory of a path determines the $\Delta x(t)$ and $\Delta y(t)$ at any timestep t. As a result, defining target paths as shapes allows for the definition of both static and dynamic task demands over a period of time. This approach allows us to systematically define DTAPs with gradually or abruptly changing task demands across a spectrum of characteristics.

We note that, in this section, the term "task demand" refers only to the raw task demands generated by the movement of the target in a single timestep, and assumes that the tracker is able to follow the target movement perfectly in each timestep. If tracker movement does not perfectly mimic target movement in each timestep, then the actual observed task demand will be the sum of the raw task demand plus any deviation between the tracker and target locations from the previous timestep.

For example, consider a *square* path in which the target starts at the bottom left corner of a square and moves north, east, south, and west for an equal distance in

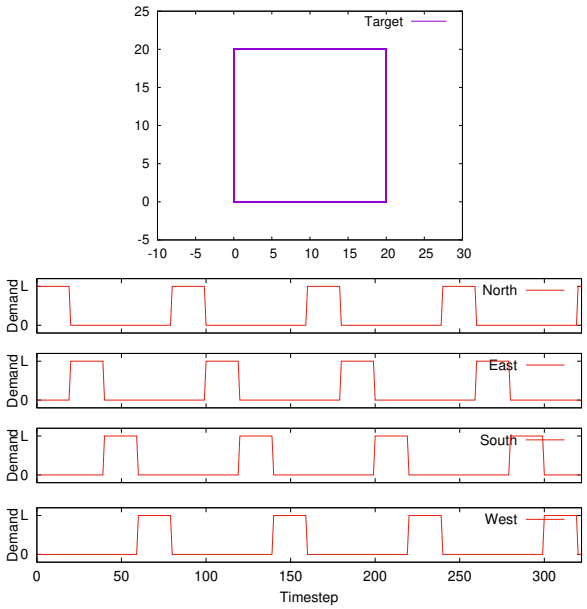

Figure 1: A square path and corresponding task demands. The target travels along the square path approximately four times around. L is the maximum possible demand.

each direction to form a square. Figure 1 shows an example of such a path and its corresponding task demands. For this path, there is only a single task demand active at any given time. The amount of demand for any task is either the maximum possible demand[2], L, or zero, and changes in demand switch abruptly between these two values.

Figure 2 shows an example of a *circle* path and its corresponding task demands. In this example, the target starts at 12 o'clock and travels clockwise around the circle just over four times. At the beginning of the path, task demand is focused to the east. As the target travels from 12 o'clock to 3 o'clock, demand to the east gradually decreases and demand to the south increases. Demand to the south peaks at L when the target is at the 3 o'clock position, then gradually decreases down to zero as the target travels from 3 o'clock to 6 o'clock. At the same time, demand to the west gradually increases from zero to the maximum amount, L. In the quadrant from 6 o'clock to 9 o'clock, demand to the west decreases from L to zero and demand to the north increases from 0 to L. In the quadrant from 9 o'clock to 12 o'clock, demand to the north decreases from L to zero and demand to the east increases from 0 to L. The task demands in the circle path change in a more gradual fashion than what is seen in the square path. In addition, up to two different tasks may be active at any given time so agents must distribute themselves appropriately among multiple active tasks.

Figures 3 and 4 show two example paths that are less regular and more random in

[2]The maximum demand, L, is defined by the maximum distance that the target can travel in a single direction in one timestep.

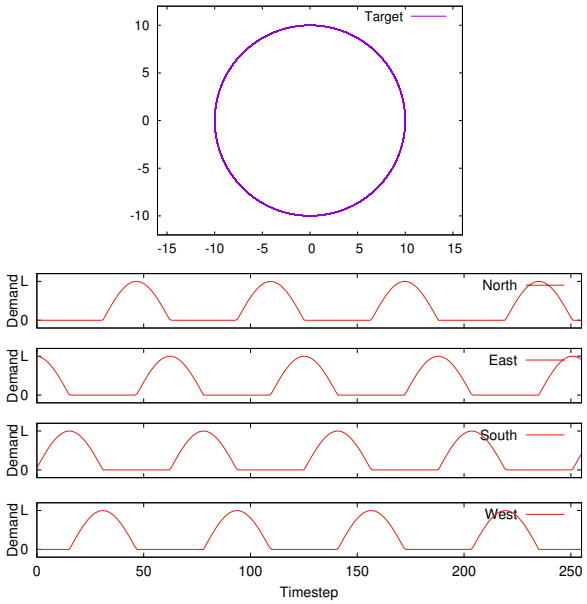

Figure 2: A circle path and corresponding task demands. The target travels along the circular path for approximately four rotations. L is the maximum possible demand.

their behavior. In the *random* path in Figure 3, task demands randomly change gradually over time. The magnitude of the maximum change in heading in each timestep is an adjustable parameter of the random path. In the *sharp* path in Figure 4, task demands are constant between random abrupt changes. Figure 5 shows a *zigzag* path which consists of constant task demand to the east, varying task demands to the north and south, and no task demand to the west. Finally, a straight path in any direction represents constant static task demands for one (if due N, E, S, or W) or two tasks.

These examples illustrate how this testbed paradigm can be used to define a variety of time-varying task demands in a systematic way. Task demands may be regular or irregular. They may change gradually or abruptly. The number of tasks that are active or that change at any given time can vary. In addition to examining swarm response to the task demands for a single path, this system allows the ability to investigate a spectrum of task demands. For example, Figure 6 shows how task demands change on a zigzag path with fixed amplitude and increasingly larger periods. Moving from smaller to larger periods increases the relative amount of task demand to the east and decreases the relative amount of task demand to the north and south. Figure 7 shows that circle paths with increasing radii produce task demands with decreasing rate of change.

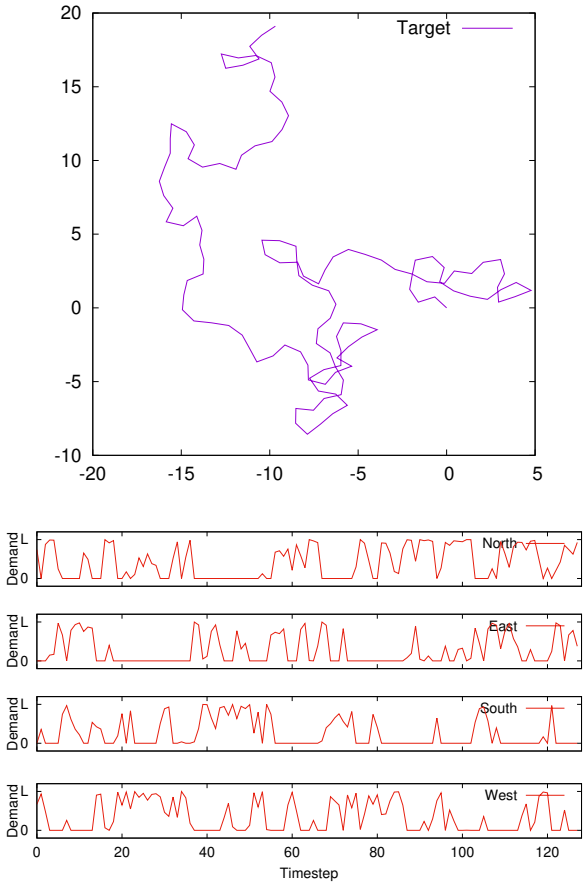

Figure 3: Random path and corresponding task demands. L is the maximum possible demand.

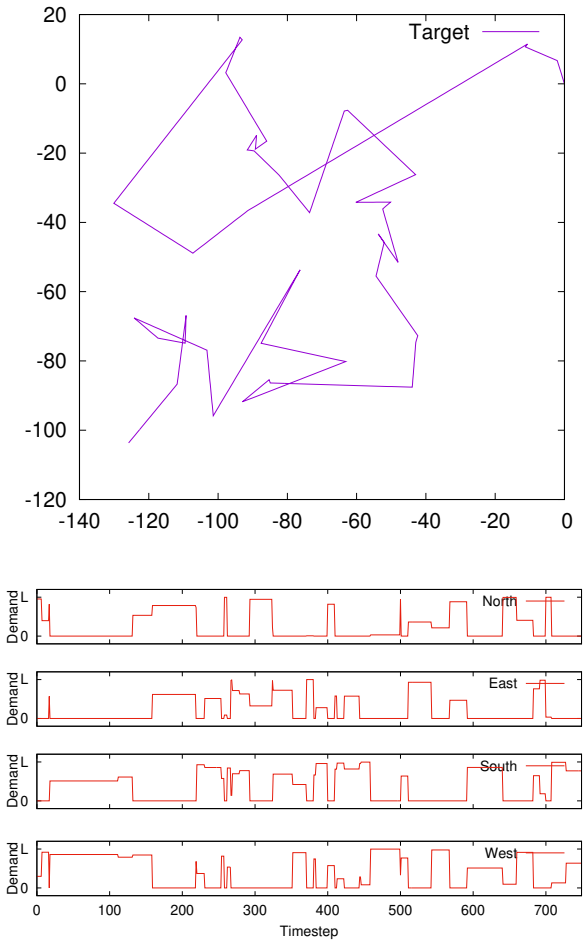

Figure 4: Sharp path and corresponding task demands. L is the maximum possible demand.

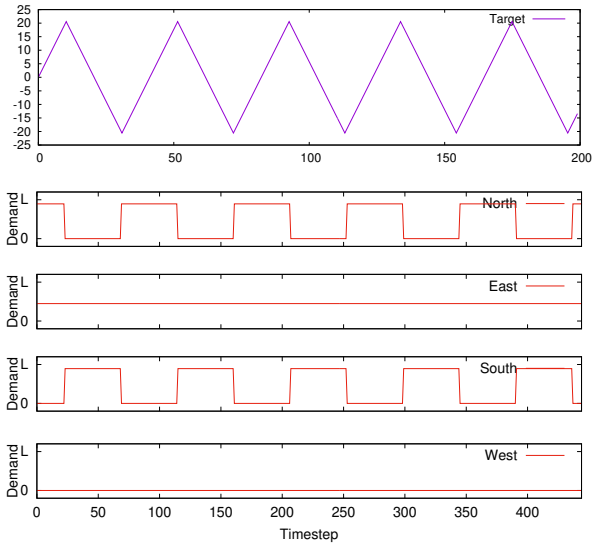

Figure 5: Zigzag path and corresponding task demands. L is the maximum possible demand.

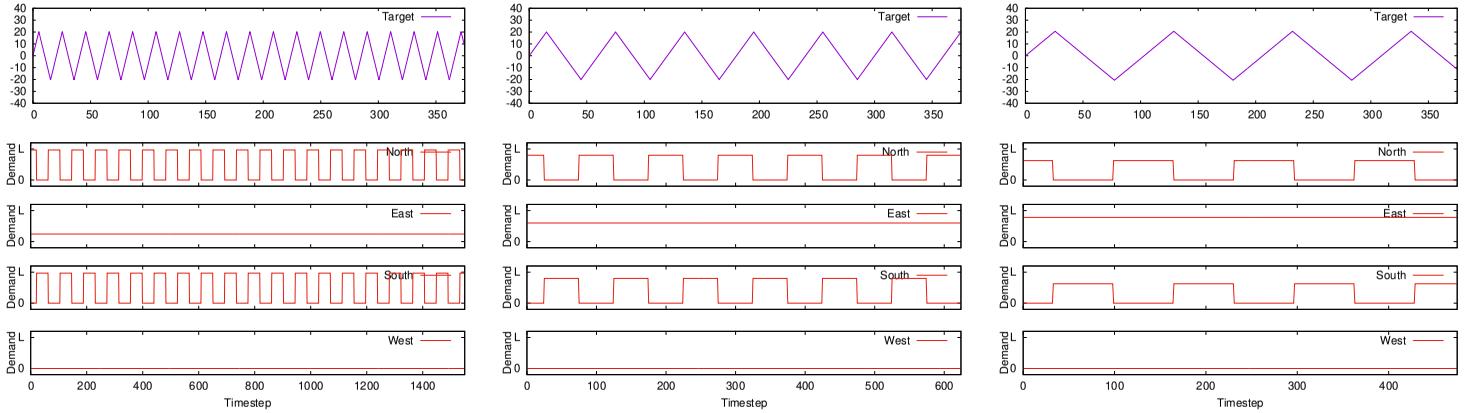

Figure 6: Zigzag path and corresponding task demands. Amplitude is fixed at 20; period set to 20, 60, and 100. L is the maximum possible demand.

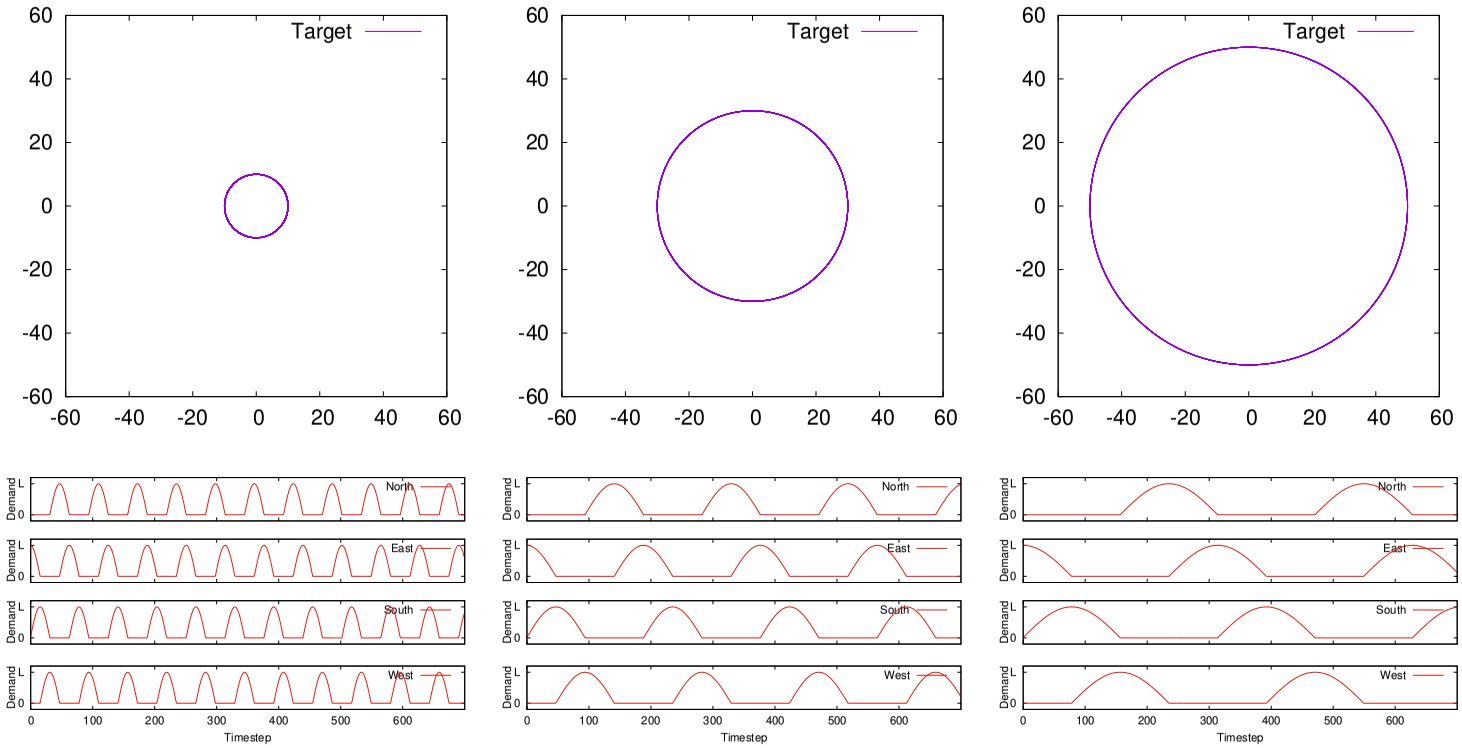

Figure 7: Circle path and corresponding task demands. Diameters of 20, 60, and 100. L is the maximum possible demand.

Path name	Abrupt or gradual change	Regular or irregular	Maximum non-zero task demands	Adjustable parameters
Move west	Static	N.A.	1	N.A.
Move northeast	Static	N.A.	2	N.A.
Circle	Gradual	Regular	2	Radius
Random	Gradual	Irregular	2	Change magnitude
S-curve	Gradual	Regular	2	Amplitude, period
Square	Abrupt	Regular	1	Edge length
Diamond	Abrupt	Regular	2	Edge length
Sharp	Abrupt	Irregular	2	Change probability
Zigzag	Abrupt	Regular	2	Amplitude, period

Table 1: Paths currently available in the collective control testbed.

Table 1 lists some of the paths that are currently available in the testbed and their corresponding characteristics. The adjustable parameters describe features that may be varied and studied across a range of values. For all paths, the target moves at constant velocity of L units per timestep. Three classes of paths are available in the system: static paths, gradually changing dynamic paths, and abruptly changing dynamic paths.

Constant target movement in a single direction generates static task demands. The task demand in each direction remains constant for all timesteps.

- The **move west** path moves due west, generating constant static task demand to the west.
- The **move northeast** path moves to the northeast generating constant task demand to the north and to the east.

Target movement along curved paths generates gradually changing task demands. Such changes require a small number of agents to change their tasks at any given time. Examples of target paths with gradually changing task demands include:

- The **circle** path starts at 12 o'clock and travels clockwise around the circle.
- The **random** path makes a small random change in heading in each timestep. The magnitude of the change in heading in each timestep is derived from a Gaussian distribution centered on zero and can be adjusted by varying standard deviation.
- The **s-curve** path travels in a serpentine path from west to east.

Target movement along paths that make sharp turns generates abruptly changing task demands. Such changes require a large number of agents to change their tasks at a given time. Example paths with abruptly changing task demands include:

- The **square** path starts at the bottom left corner of the path and moves north, east, south, then west for a specified distance in each direction.

- The **diamond** path starts at the leftmost point of the path and moves NE, SE, SW, then NW for a specified distance in each direction. This path differs from the square path in that it creates demand for two tasks in every timestep.
- The **sharp** path moves in straight lines with occasional random changes in direction. The frequency of the random direction changes is defined by a probability value that gives the probability that a direction change will occur in each timestep.
- The **zigzag** path travels in a sawtooth path from west to east. It consists of a mix of static and dynamic task demands.

In summary, the collective tracking problem provides a systematic way to define DTAPs over a wide range of charateristics. These problems may be used to test and stress a swarm in every conceivable way.

5 Resources for analysis

In addition to providing mechanisms for specifying a variety of task allocation problems, this testbed also allows for detailed monitoring of agent actions which can aid in analyses of how a given swarm addresses task demands over time. This monitoring includes collecting data over the course of a simulation run as well as aggregating summary statistics after a run has ended. Because we are using this testbed to study coordination in threshold-based swarms,[3] the discussion below is given in that context. Nevertheless, the agents' decision making mechanisms can be replaced by other swarm coordination methods to study other approaches. This testbed is an equally challenging and equally informative testbed for studying other approaches to decentralized task allocation.

5.1 Data collected over the course of a simulation run

Observations of a swarm's actions over the course of a run allow users to evaluate how effectively and how promptly a swarm responds to dynamic task demands. The testbed allows for the collection and visualization of a variety of data over each timestep of a simulation run. The following are examples of the types of data that can be examined.

The collective control problem generates both clearly defined task demands and clearly defined swarm responses. This allows a user to observe how the agents in a swarm respond to changes in task demand, including both the quality and timeliness of the response. In addition, the target and tracker paths provide an intuitive way to visualize the performance of a swarm on this problem. Figure 8 shows an example of target and tracker paths (top plot) and the corresponding task demands and swarm responses (bottom four plots). Target data is indicated by solid purple and red lines and tracker data is indicated by the dashed green lines. We can see that, in this example, there is a slight delay between the timing of the task demands and the swarm response.

[3] Threshold-based swarms are systems in which each agent has a threshold for each task that it can perform and these thresholds determine when agents will act and what task they choose.

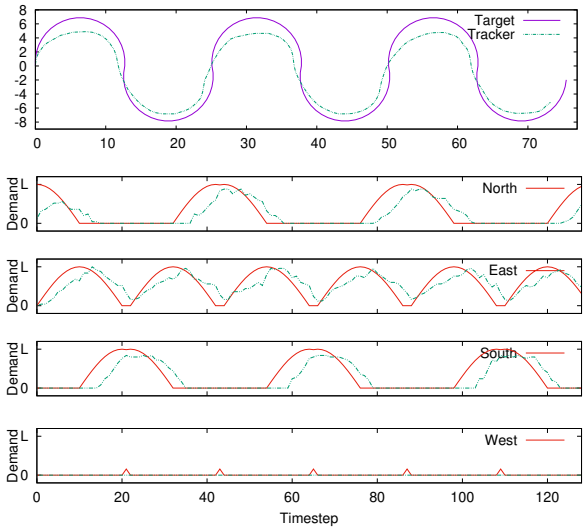

Figure 8: S-curve path and corresponding task demands. L is the maximum possible demand. The dashed lines indicate the swarm's performance in tracking the target's movement.

This delay manifests in the resulting tracker path as "cutting corners" relative to the target path.

Figure 9 shows a visualization of the actions of all agents in every timestep of a simulation run. Agent actions for three paths are shown: zigzag, s-curve, and circle. The x-axis of each plot indicates the agent; the y-axis of each plot counts timesteps. The color of each cell indicates the action of a given agent in a given timestep: blue = *push_north*, yellow = *push_east*, red = *push_south*, green = *push_west*, and white = remain *idle*. Each column represents the sequence of actions of a single agent over the course of a run. The agents in these example runs have static, uniformly random thresholds for each task. Such visualizations allow us not only to observe the distribution of agent actions over time, but also examine metrics such as how often agents switch tasks, whether agents specialize (only act of a few tasks) or generalize (act on all tasks), and the percent of a swarm that remains idle. The plots in Figure 9 show that a swarm consisting of agents with static uniformly random thresholds respond differently and appropriately for different paths (different types of dynamic task demands).

Data on the number of agents that switch tasks in each timestep can also examined. Figure 10 shows such data for example runs of the zigzag and s-curve target paths. These results indicate that, for a zigzag path, task switches are more frequent around the sharp turns of the zigzag path and occur with significantly less frequency along the straightaways. By comparison, an s-curve path results in a relatively constant and moderate amount of task switching throughout the course of a run.

For systems in which agent characteristics can adapt over time, visualization of how such data changes over time can provide feedback on the effectiveness and utility

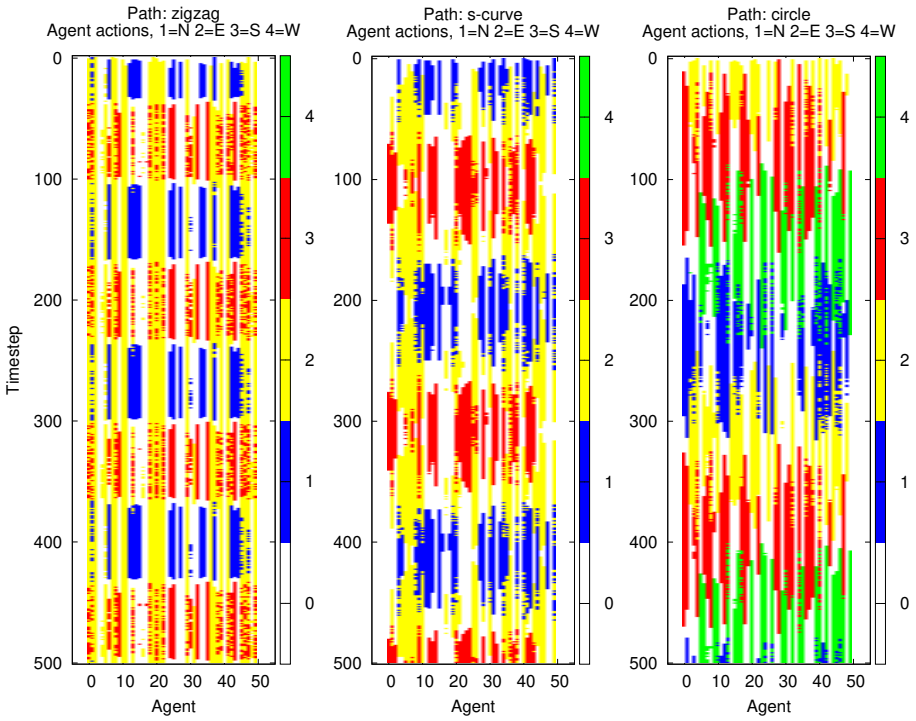

Figure 9: Agent actions in each timestep of a simulation run for the zigzag, s-curve, and circle target paths. Blue = *push_north*, yellow = *push_east*, red = *push_south*, green = *push_west*, and white = *idle*. The swarms in these runs are threshold based swarms in which agents are assigned task thresholds drawn from a uniform random distribution.

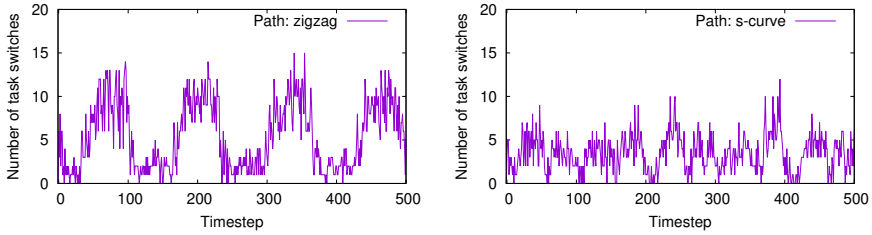

Figure 10: The number of agents that switch tasks in each timestep in an example run of the zigzag and s-curve target paths. The total number of agents in the swarm is 50. Agents are assigned task thresholds drawn from a uniform random distribution.

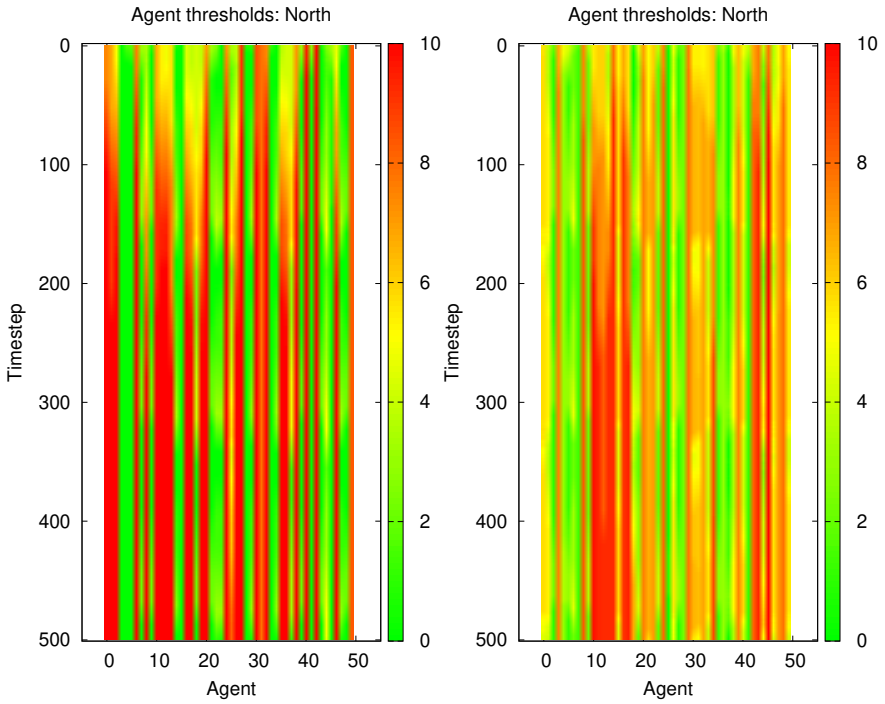

Figure 11: Agent threshold values for the *push_north* task on the s-curve target path.

of dynamic adaptations. For example, we are investigating swarms in which agents dynamically adapt task thresholds over time. Figure 11 shows how agent threshold values for the *push_north* task vary over an example simulation run on the s-curve target path. In each plot the x-axis indicates agents and the y-axis indicates timesteps. Each column of color shows the evolution of the *push_north* threshold for a single agent over time. Threshold values range from zero to ten, indicated by the colors ranging from green to yellow to red, respectively. The two plots shown in Figure 11 illustrate threshold adaptation in two different types of dynamic threshold methods. These plots show that the system on the left is less adaptive, converging over time to extreme threshold values. The system on the right maintains a more diverse collection of threshold values across the agent population that continue to evolve throughout the run.

5.2 Data summarized over a complete run

The clearly defined task demands and swarm response means that we are also able to collect and aggregate detailed statistics about swarm behavior and performance once a simulation run is complete. Beyond the overall swarm performance metrics, the system allows us to examine differences such as how work is distributed among swarm mem-

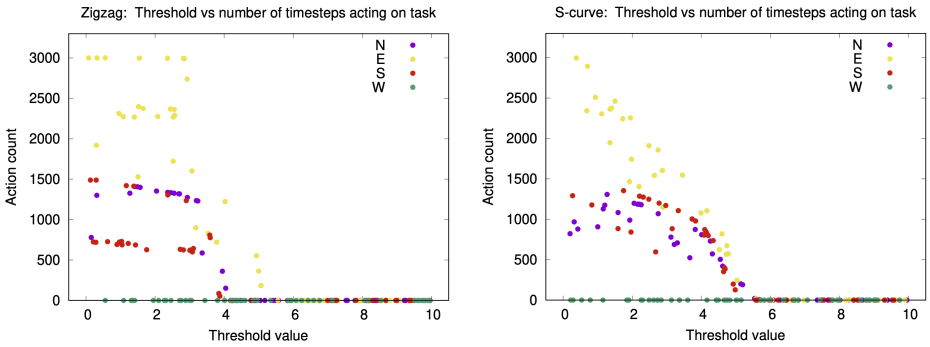

Figure 12: A comparison of agent threshold values and the number of times agents act on a task in example simulation runs lasting 3000 timesteps. The left plot shows results from a example run of the zigzag path which contains abruptly changing task demands. The right plot shows results from a example run of the s-curve path which contains gradually changing task demands.

bers and the stability of the division of labor in a swarm. The following data illustrate examples where aggregated data from this collective contol problem can reveal insights on how swarm dynamics differ for different types of problems.

Figure 12 shows the correlation between agent threshold values and the number of times agents act on a task in an example simulation run. The left plot shows results from an example run of the zigzag path; the right plot, the s-curve path. On both plots, the x-axis indicates threshold values and the y-axis indicates the number of timesteps in which the agent with a given threshold acted on that task. Note that each agent has a separate threshold for all four tasks (*push_north*, *push_east*, *push_south*, and *push_west*) so the number of data points on each graph is $4 \times N$ where N is the total number of agents in the swarm. In both runs, agent task thresholds are static and are drawn from a uniform random distribution.

These two examples show that swarm dynamics differ between problems with abruptly changing (zigzag path) and gradually changing (s-curve path) task demands. In the left plot (zigzag path with abruptly changing task demands), there are clearly agents that specialize on the *push_east* task, including six agents that select to *push_east* in all 3000 timesteps of the simulation. There are also agents that specialize on the *push_north* and *push_south* tasks. There appear to be groups of agents that act on a given task approximately the same number of times: two groups of agents select the *push_east* task approximately 3000 and 2300 times, two groups of agents select the *push_south* task approximately 1400 and 700 times, and two groups of agents select the *push_north* task approximately 1400 and 700 times. The two levels differentiate agents that specialize on a single task and agents that switch back and forth between two tasks. In the right plot (s-curve path with gradually changing task demands), the distribution of action counts is much more gradual. There is not a clear distinction of groups of agents that are acting similarly in concert. Instead there is a gradual decline in action count as threshold value increases. Significantly more high threshold agents

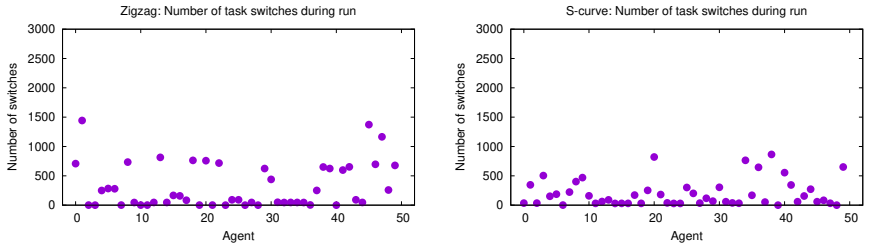

Figure 13: The number of task switches experienced by each agent in a swarm over a simulation run consisting of 3000 timesteps. The left plot shows results from a example run of the zigzag path which contains abruptly changing task demands. The right plot shows results from a example run of the s-curve path which contains gradually changing task demands.

act on the s-curve path than on the zigzag path (in the zigzag plot, only five thresholds above 4 have non-zero action counts; in the s-curve plot, significantly more have non-zero action counts). For both plots, the fact that the maximum count for the *push_north* and *push_south* tasks are approximately half of the maximum count for the *push_east* task correlates with the fact that the s-curve path exhibits movement to the east in every timestep while alternating between periods of demand to the north and south.

Figure 13 shows the number of task switches experienced by each agent in a swarm over two example simulation runs consisting of 3000 timesteps. The left plot shows results from an example run of the zigzag path; the right plot, the s-curve path. On both plots, the x-axis indicates agents and the y-axis indicates the number of task switches. In the left plot (zigzag path with abruptly changing task demands), agents appear to either switch tasks frequently or rarely. Task switching appears to be concentrated in a subset of agents with the remaining agents experiencing significantly less task switching. In the right plot (s-curve path with gradually changing task demands), there appears to be a greater variety of frequencies of task switching.

For agent characteristics that change over time, in addition to examining changes per timestep, we can also summarize the effective agent characteristics by visualizing, for example, the range and average values over the course of a run. Figure 14 shows example results from our investigations of agents with dynamic response intensities[4]. Specifically, it illustrates the aggregate *push_east* intensity data for all agents in a single simulation run of the s-curve (top plot) and zigzag (bottom plot) paths. The x-axis shows each of the 50 agents in the run. The left y-axis depicts intensity values and the right y-axis shows activation counts. Each agent's defined intensity range (range of possible intensity values) for pushing east is shown by the orange lines. Each agent's effective intensity range (range of actual executed intensity values) is shown as a blue line. Note that each effective intensity range is subsumed by the defined range. All

[4]Response intensity defines the magnitude of an agent's effort relative to others. It can model increased effectiveness due to experience on a task or characteristics such as strength or stamina [44, 45].

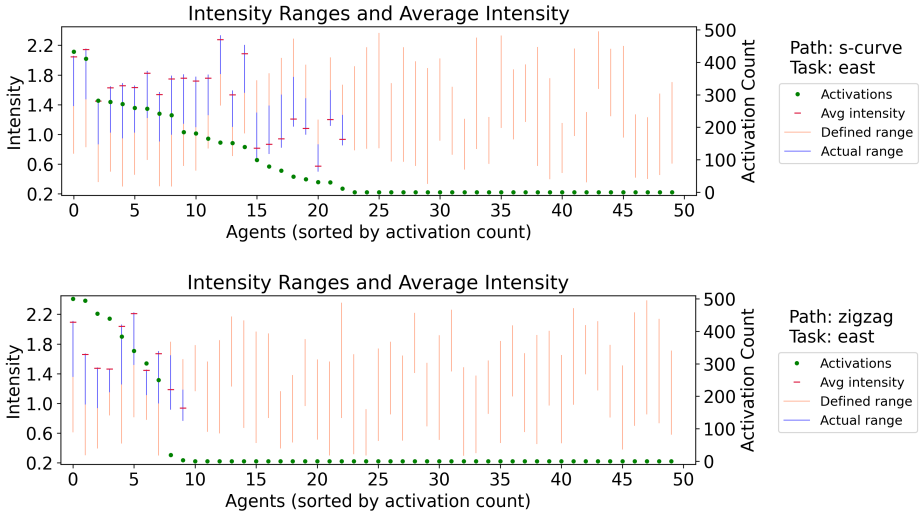

Figure 14: Average intensity values by agent for task *push_east* for paths s-curve (top) and zigzag (bottom). Orange lines show the defined intensity range. Blue lines show the range of intensities during the run for each agent. Red dashes mark the average intensity per agent. Green dots show the *push_east* activation count for each agent.

intensities are initialized to the midpoint of the defined range. An agent's average intensity value for a task, marked by a red dash, is calculated only for timesteps for which the agent activated for that task. The activation count for each agent appears as a green dot. Comparison of these two plots reveals swarm dynamics that concur with that shown in Figures 12 and 13. Activation for a given task (in this case *push_east*) is limited to a smaller group of agents for the zigzag task than for the s-curve task. That smaller group activates more frequently which pushes almost all of these agents to their maximum intensities. Activation is distributed among more agents in the s-curve path which results in greater diversity in the average intensity values relative to their corresponding ranges.

6 Discussion

If we are studying how swarms coordinate and adapt, being able to evaluate any proposed agent decision making strategies on a range of DTAPs with different characteristics will strengthen results and make them more likely to be generalizable. Applications do not always allow for a range of DTAP characteristics to be tested. The collective control problem described here, though abstract, has a number of features that make it a good testbed for studying DTAPS. First, the target paths provide a systematic way of defining a variety of DTAPS on which swarm dynamics may be studied. Second, the clearly defined task demands and agent responses allows for detailed examination and analyses of the dynamics of how different swarms respond to different types of DTAPs.

Finally, the tracking problem allows for a very intuitive way to visualize swarm performance: in the form of the actual paths taken by the target and tracker.

The key contribution of this testbed is the ability to systematically define a variety of DTAPs. Because task demand is represented by differences between target and tracker position in each of the four cardinal directions, target paths are a structured way to define how task demands change over time. Varying the parameters of a target path creates classes of DTAPs that have some features in common but vary along one or more specific axes. Thus, this test problem provides a non-arbitrary method for defining problems with task demands that change in a variety of ways, as well as the ability to define problems that vary across one or more spectrums of DTAP characteristics. This testbed can represent physical problems, as in the tracking manifestation we present here, or problems where task demands do not have a physical analog and the work of the swarm simply represents a response to the demand.

A secondary benefit of this testbed is its ability to monitor both problem demands and agent responses very precisely. The ability to accurately measure target and tracker movements and locations in each timestep allows us to precisely monitor both the magnitude and speed of a swarm's response during a simulation. In addition, the system also allows for aggregation of performance statistics over an entire simulation run. Although some of the data that may be desired will undoubtedly be system specific and depend on the agents' control algorithm or swarm composition, the system provides the capability to collect and visualize general data such as performance on the tracking problem, which agents act when, and frequency of task switching. Finally, the target and tracker paths provide a very natural way for users to visualize the progress and performance of a swarm on a dynamic task allocation problem.

We note that an unanticipated side effect of the fact that pairs of tasks in the tracking problem oppose each other, e.g. positive task demand to the north is essentially negative task demand to the south, means that target movement in a given direction can reduce unaddressed task demand in the opposite direction. Thus, task demands may be addressed by a combination of agent actions and the target's own movement. This effect is particularly evident in examples where the tracker cuts corners resulting in a tracker path whose length is shorter than the target path, but can be detected by the Goal 2 evaluation metric.

Future work on the development of this testbed is focused, at this time, on increasing the number of tasks in the problem and the number of tasks that may simultaneously have demand in a given timestep. This increase may be achieved in one of two ways. First, we can add additional active tasks by increasing the dimensionality of the tracking problem from two to three or more. Moving to a three dimensional space will result in six possible tasks and a maximum of three tasks that can have demand at the same time. Adding dimensions beyond three creates more abstract DTAPs that are not limited by modeling a physical space. Another way to increase the number of tasks is to define the task demands for each direction using a separate target path or function. Such an approach will allow all tasks to have demand in a given timestep even if the actual path travelled by the target aggregates the north and south demands and the east and west demands. In this case, the visualization will be less directly illustrative of the individual task demands but will still intuitively illustrate how well the tracker motion follows the aggregate target path.

7 Acknowledgments

This work was supported by the National Science Foundation under Grant No. IIS1816777.

References

[1] William Agassounon, Alcherio Martinoli, and Rodney Goodman. A scalable distributed algorithm for allocating workers in embedded systems. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 3367–3373, 2001.

[2] Gerardo Beni. From swarm intelligence to swarm robotics. In *Proceedings of the 2004 International Conference on Swarm Robotics*, volume LNCS 3342, pages 1–9, 2005.

[3] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Robots and biological systems: towards a new bionics?*, pages 703–712. Springer, 1993.

[4] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[5] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Quantitative study of the fixed response threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London B*, 263:1565–1569, 1996.

[6] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, 60:753–807, 1998.

[7] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7:1–41, 2013.

[8] Nicolas Bredeche, Jean-Marc Montanier, Berend Weel, and Evert Haasdijk. Roborobo! a fast robot simulator for swarm and collective robotics. *arXiv preprint arXiv:1304.2888*, 2013.

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

[10] Arne Brutschy, Nam-Luc Tran, Nadir Baiboun, Marco Frison, Giovanni Pini, Andrea Roli, Marco Dorigo, and Mauro Birattari. Costs and benefits of behavioural specialization. *Robotics and Autonomous Systems*, 60(11):1408–1420, 2012.

[11] Mike Campos, Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, 8(2):83–96, 2000.

[12] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1400–1405, 2007.

[13] Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Wenguo Liu, Alan F. T. Winfield, Yutaka Nakamura, and Hiroshi Ishiguro. Adaptive foraging for simulated and real robotic swarms: The dynamical response threshold approach. *Swarm Intelligence*, 10:1–31, 2018.

[14] Christopher M. Cianci, Xavier Raemy, Jim Pugh, and Alcherio Martinoli. Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In *Proceedings of the 2nd International Conference on Swarm Robotics*, pages 103–115, 2006.

[15] Vincent A. Cicirello and Stephen F. Smith. Distributed coordination of resources via wasp-like agents. In *Workshop on Radical Agent Concepts, LNAI 2564*, pages 71–80, 2002.

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

[17] Javier de Lope, Dario Maravall, and Yadira Quinonez. Response threshold models and stochastic learning automata for self-coordination of heterogeneous multi-task distribution in multi-robot systems. *Robotics and Autonomous Systems*, 61:714–720, 2013.

[18] M. Dorigo, D. Floreano, L.M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A.L. Christensen, A. Decugnière, G. Di Caro, F. Ducatelle, E. Ferrante, A. Förster, J. Guzzi, V. Longchamp, S. Magnenat, J.M. Gonzales, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Rétornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stützle, V. Trianni, E. Tuci, A.E. Turgut, and F. Vaussard. Swarmanoid: A novel concept for the study of heterogeneous robot swarms. *IEEE Robotics and Automation Magazine*, 20(4):60–71, December 2013.

[19] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[20] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[21] Fernando dos Santos and Ana L. C. Bazzan. An ant based algorithm for task allocation in large-scale and dynamic multiagent scenarios. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 73–80, 2009.

[22] Kaelen Engholdt, H. David Mathias, and Annie S. Wu. Variable response duration promotes self-organization in decentralized swarms. In *Proceedings of the 9th International Conference on Bioinspired Optimization Methods and their Applications*, pages 17–28, 2020.

[23] Brian P. Gerkey, Richard T. Vaughn, and Andrew Howard. The player/stage project: tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, 2003.

[24] Joseph P. Giordano, Annie S. Wu, Arjun Pherwani, and H. David Mathias. Comparison of desynchronization methods for a decentralized swarm on a logistic supply problem. In *Proceedings of the 30th International Conference on Autonomous Agents and Multiagent Systems*, 2021.

[25] Harry Goldingay and Jort van Mourik. The effect of load on agent-based algorithms for distributed task allocation. *Information Sciences*, 222:66–80, 2013.

[26] S. Goss, Serge Aron, Jean-Louis Deneubourg, and Jacques M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.

[27] S. Goss, R. Beckers, Jean-Louis Deneubourg, Serge Aron, and Jacques M. Pasteels. How trail laying and trail following can solve foraging problems for ant colonies. In *Behavioral Mechanisms of Food Selection, NATO ASI Series*, volume 20, pages 661–678, 1990.

[28] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, 2002.

[29] Fredrik Jansson, Matthew Hartley, Martin Hinsch, Ivica Slavkov, Noemí Carranza, Tjelvar Olsson, Roland Dries, Johanna Grönqvist, Athanasius Marée, James Sharpe, Jaap Kaandorp, and Veronica Grieneisen. Kilombo: a kilobot simulator to enable effective research in swarm robotics. 2016. arXiv:1511.04285v2.

[30] Julia C. Jones, Mary R. Myerscough, Sonia Graham, and Benjamin P. Oldroyd. Honey bee nest thermoregulation: Diversity promotes stability. *Science*, 305(5682):402–404, 2004.

[31] K-Team Corporation. `https://www.k-team.com/khepera-iv`, 2021. Last accessed 2021-04-06.

[32] Anshul Kanakia, Behrouz Touri, and Nikolaus Correll. Modeling multi-robot task allocation with limited information as global game. *Swarm Intelligence*, 10:147–160, 2016.

[33] M. Kirtas, K. Tsampazis, N. Passalis, and A. Tefas. Deepbots: A webots-based deep reinforcement learning framework for robotics. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 64–75, Cham, 2020. Springer International Publishing.

[34] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Elichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the Autonomous Agents Conference*, pages 340–347, 1997.

[35] Oran Kittithreerapronchai and Carl Anderson. Do ants paint trucks better than chickens? Market versus response threshold for distributed dynamic scheduling. In *Proceedings of the Congress on Evolutionary Computation*, pages 1431–1439, 2003.

[36] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3:2149–2154 vol.3, 2004.

[37] Michael J. B. Krieger and Jean-Bernard Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30:65–84, 2000.

[38] Thomas H. Labella, Marco Dorigo, and Jean-Louis Deneubourg. Division of labor in a group of robotics inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):4–25, 2006.

[39] Wonki Lee and DaeEun Kim. History-based response threshold model for division of labor in multi-agent systems. *Sensors*, 17(6):1232, 2017.

[40] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja Mataric. Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, 25:225–241, 2006.

[41] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[42] Stéphane Magnenat, Philippe Rétornaz, Michael Bonani, Valentin Longchamp, and Francesco Mondada. Aseba: A modular architecture for event-based control of complex robots. *IEEE/ASME Transactions on Mechatronics*, 16(2):321–329, 2011.

[43] H. David Mathias, Annie S. Wu, and Daniel Dang. Evolved response thresholds generalize across problem instances for a deterministic-response multiagent system. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[44] H. David Mathias, Annie S. Wu, Laik Ruetten, and Eric Coursin. Improving multi-agent system coordination via intensity variation. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, pages 368–373, 2020.

[45] H.David Mathias, Annie S. Wu, and Laik Ruetten. Heterogeneous response intensity ranges and response probability improve goal achievement in multi-agent systems. In *Proceedings of the 12th International Conference on Swarm Intelligence (ANTS)*, pages 148–160, 2020.

[46] Olivier Michel. Webots™: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1:39–42, 2004.

[47] Francesco Mondada, Giovanni C. Pettinaro, Andre Guignard, Ivo W. Kwee, Dario Floreano, Jean-Louis Deneubourg, Stefano Nolfi, Luca Maria Gambardella, and Marco Dorigo. Swarm-bot: A new distributed robotic concept. *Autonomous Robots*, 17:193–221, 2004.

[48] Nadia Nedjah and Luneque Silva Junior. Review of methodologies and tasks in swarm robitcs towards standardization. *Swarm and Evolutionary Computation*, 50:100565, 2019.

[49] Marta Niccolini, Mario Innocenti, and Lorenzo Pollini. Multiple UAV task assignment using descriptor functions. In *Proceedings of the 18th IFAC Symposium on Automatic Control in Aerospace*, pages 93–98, 2010.

[50] Shervin Nouyan. Agent-based approach to dynamic task allocation. In *Proceedings of the 3rd International Workshop on Ant Algorithms*, pages 28–39, 2002.

[51] Bao Pang, Chengjin Zhang, Yong Song, and Hongling Wang. Self-organized task allocation in swarm robotics foraging based on dynimcal response threshold approach. In *Proceedings of the 18th International Conference on Advanced Robotics*, pages 256–261, 2017.

[52] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, G Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6:271–295, 12 2012.

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

[54] Giovanni Pini, Arne Brutschy, Carlo Pinciroli, Marco Dorigo, and Mauro Birattari. Autonomous task partitioning in robot foraging: An approach based on cost estimation. *Adaptive Behavior*, 21(2):118–136, 2013.

[55] Giovanni Pini, Arne Brutschy, Alexander Scheidler, Marco Dorigo, and Mauro Birattari. Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer. *Artificial Life*, 20(3):291–317, 2014.

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

[57] Martha E. Pollack and Marc Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 183–189, 1990.

[58] Richard Price and Peter Tino. Evaluation of adaptive nature inspired task allocation against alternate decentralised multiagent strategies. In *Proceedings of the International Conference on Parallel Problem Solving from Nature, LNCS 3242*, pages 982–990, 2004.

[59] Andreagiovanni Reina, Alex J. Cope, Eleftherios Nikolaidis, James A.R. Marshall, and Chelsea Sabo. Ark: Augmented reality for kilobots. *IEEE Robotics and Automation Letters*, 2(3):1755–1761, 2017.

[60] Mitchel Resnick. Starlogo: An environment for decentralized modeling and decentralized thinking. In *Conference companion on Human factors in computing systems*, pages 11–12, 1996.

[61] E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013.

[62] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298, 2012.

[63] Erol Sahin. Swarm robotics: From source of inspiration to domains of application. In *Proceedings of the 2004 International Conference on Swarm Robotics*, volume LNCS 3342, pages 10–20, 2005.

[64] Tomoichi Takahashi, Satoshi Tadakoro, Masayuki Ohta, and Nobuhiro Ito. Agent based approach in disaster rescue simulation - from test-bed of multiagent system to practical application. In *Proceedings of RoboCup 2001*, Lecture Notes in Artificial Intelligence, pages 102–111, 2002.

[65] Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *Proceedings of the International Conference on Complex Systems*, volume 21, pages 16–21, 2004.

[66] Anja Weidenmüller. The control of nest climate in bumblebee ($Bombus\ terrestris$) colonies: Interindividual variability and self reinforcement in fanning response. *Behavioral Ecology*, 15:120–128, 2004.

[67] D. Weyns, A. Helleboogh, and T. Holvoet. The packet-world: A test bed for investigating situated multi-agent systems. In R. Unland, M. Calisti, and M. Klusch, editors, *Software Agent-Based Applications, Platforms and Development Kits*. Birkhäuser Basel, 2005.

[68] Annie S. Wu and H. David Mathias. Dynamic response thresholds: Heterogeneous ranges allow specialization while mitigating convergence to sink states. In *Proceedings of the 12th International Conference on Swarm Intelligence*, pages 107–120, 2020.

[69] Annie S. Wu, H. David Mathias, Joseph P. Giordano, and Anthony Hevia. Effects of response threshold distribution on dynamic division of labor in decentralized swarms. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, 2020.

[70] Annie S. Wu, R. Paul Wiegand, and Ramya Pradhan. Response probability enhances robustness in decentralized threshold-based robotic swarms. *Swarm Intelligence*, 14:233–258, 2020.

[71] Yongming Yang, Xihui Chen, and Qingjun Li. Swarm robots task allocation based on local communication. In *Proceedings of the International Conference on Computer, Mechatronics, Control, and Electronic Engineering*, pages 415–418, 2010.