Analysis of Evolved Response Thresholds for Decentralized Dynamic Task Allocation

H. DAVID MATHIAS, University of Wisconsin—La Crosse, USA
ANNIE S. WU, University of Central Florida, USA
DANIEL DANG, Whitman College, USA

In this work, we investigate the application of a multi-objective genetic algorithm to the problem of task allocation in a self-organizing, decentralized, threshold-based swarm. We use a multi-objective genetic algorithm to evolve response thresholds for a simulated swarm engaged in dynamic task allocation problems: two-dimensional and three-dimensional collective tracking. We show that evolved thresholds not only outperform uniformly distributed thresholds and dynamic thresholds but achieve nearly optimal performance on a variety of tracking problem instances (target paths). More importantly, we demonstrate that thresholds evolved for some problem instances generalize to all other problem instances, eliminating the need to evolve new thresholds for each problem instance to be solved. We analyze the properties that allow these paths to serve as *universal training instances* and show that they are quite natural.

After *a priori* evolution, the response thresholds in our system are static. The problem instances solved by the swarms are highly dynamic, with schedules of task demands that change over time with significant differences in rate and magnitude of change. That the swarm is able to achieve nearly optimal results refutes the common assumption that a swarm must be dynamic to perform well in a dynamic environment.

CCS Concepts: • **Computing methodologies** → **Multi-agent systems**; **Genetic algorithms;**

Additional Key Words and Phrases: Inter-agent variation, response thresholds, generalization

ACM Reference format:
H. David Mathias, Annie S. Wu, and Daniel Dang. 2022. Analysis of Evolved Response Thresholds for Decentralized Dynamic Task Allocation. *ACM Trans. Evol. Learn. Optim* 2, 2, Article 5 (August 2022), 30 pages.
https://doi.org/10.1145/3530821

1 INTRODUCTION

In this work, we use a multi-objective **genetic algorithm (GA)** to evolve static agent response thresholds for a decentralized, threshold-based swarm. The decentralized and redundant qualities of swarms make them robust and scalable but also make the coordination of the agents that comprise a swarm a challenging problem. The response threshold approach is modeled after the division of labor in biological swarms and is a commonly used approach for coordinating artificial swarms. Optimal performance of such systems must address multiple goals and depends on

This work was supported by the National Science Foundation under Grant No. IIS1816777.
Authors' addresses: H. D. Mathias, University of Wisconsin—La Crosse, 1725 State Street, La Crosse, WI, 54669; email: dmathias@uwlax.edu; A. S. Wu, University of Central Florida, P.O. Box 162362, Orlando, FL, 32816-2362; email: aswu@cs.ucf.edu; D. Dang, Whitman College, 345 Boyer Avenue, Walla Walla, WA, 99362; email: gnadleinad@gmail.com.

2688-3007/2022/08-ART5 $15.00
https://doi.org/10.1145/3530821

effective assignment of threshold values among the agents of the swarm. Because agents work collectively, threshold assignments cannot be optimized locally, and, therefore, the problem to be solved entails finding a collection of threshold values that together result in optimal swarm performance for a given schedule of task demands. We show that a multi-objective GA is able to find nearly optimal solutions to this multi-objective, large-scale combinatoric problem and, more significantly, we demonstrate that response thresholds evolved for some problem instances generalize to provide near optimal performance for all other problem instances tested, despite significant differences in task demand schedules between problem instances.

Eusocial insects, such as ants, bees, and wasps, are species with complex social structures within which individuals divide labor among various tasks such as foraging for food, foraging for nest building materials, and brood care. Division of labor in the absence of centralized control of individual behaviors is a difficult problem that the eusocial insects have solved quite effectively [Jeanne 1986]. Response thresholds are one determinant of when an individual undertakes a task. In the response threshold model, individuals sense environmental stimuli such as the amount of stored food or the temperature in the hive and act if the stimulus exceeds a threshold value. Inter-individual variation, differences in when and how individuals respond to task demands, is an important mechanism for effective division of labor [Jeanson and Weidenmüller 2014; Weidenmüller 2004; Weidenmüller et al. 2019]. In particular, variability in response thresholds serves to desynchronize task activations by individuals, preventing the swarm from responding in lockstep.

Such natural swarm behaviors have inspired work in robotics for at least 30 years [Beni 1992; Theraulaz et al. 1991]. An artificial swarm consists of a number of agents working to achieve a common goal, usually through repetition of some number of tasks. Within this broad definition, there are many models defined by parameters such as inter-agent communication and the mechanism used to achieve division of labor. In a decentralized swarm, division of labor is achieved through individual agent decisions made without a shared controller. In this work, we focus on swarms consisting of threshold-based agents with no inter-agent communication.

As in natural swarms, homogeneity of agent behaviors in artificial swarms will result in little division of labor and poor goal attainment. Two main approaches have been used in artificial threshold-based swarms to promote heterogeniety in agent behaviors. First, threshold based systems can be probabilistic [Bonabeau et al. 1996; Kalra and Martinoli 2006; Price and Tino 2004]; agents act with some probability determined by the stimulus and the agent's threshold. The probabilistic action generates diversity in agent behaviors even if all agents have the same threshold. Second, and the approach we study here, threshold based systems can be deterministic but variable [Krieger and Billeter 2000a; Wu et al. 2020]; behavioral variability is due to inter-agent variation in threshold values. In this approach, it stands to reason that the method used to determine and assign response thresholds to agents could significantly impact the success of the swarm. Surprisingly little work, however, has examined how best to assign threshold values to achieve effective swarm performance [Campbell et al. 2011; Wu et al. 2020]. The work of Wu, et al. examines the effect of a number of probability distributions for randomly assigned response thresholds. They find that, from among the distributions tested, a uniform distribution provides the best goal achievement for a collective control two-dimensional (2D) tracking problem [Wu et al. 2021] in which the swarm controls a tracker object with the goal of following a target object in real time [Wu et al. 2020].

Because the agents in a swarm work collectively and because threshold assignments both across[1] and within[2] agents affect collective behavior, determining the optimal distribution and assignment

[1]Distribution of threshold values for a single task across all agents.
[2]Distribution of threshold values across tasks within a single agent.

of thresholds in a swarm is a non-trivial task. Making the problem even more challenging, task demands may change over time in many problems. For example, in the 2D tracking problem referenced above, the target may make frequent, random changes in direction. One way for the swarm to adapt to these changes is through dynamism in response thresholds [Castello et al. 2013; Theraulaz et al. 1998]. In practice, however, systems that use dynamic thresholds can have difficulty adapting to new task demands [Kazakova and Wu 2018; Meyer et al. 2015; Theraulaz et al. 1998]. Thus, we explore *a priori* evolution of thresholds rather than real-time adaptation. Genetic algorithms are known for their ability to find effective solutions to computationally expensive, high-dimensional problems [Forrest and Mitchell 1993].

In this article, we demonstrate that static response thresholds evolved by a GA, not only outperform random response thresholds, but in many cases achieve nearly optimal performance despite the highly dynamic task demands generated by the domain problems we use. Further, static evolved thresholds outperform dynamic thresholds despite the apparent advantage of dynamism: the ability to adapt in real time to changing task demands. Second, we show that thresholds evolved for some problem instances generalize to other problem instances with very different task demands. Two of the problem instances that we examine generalize to all other problem instances tested, making them *universal training instances*. Finally, we analyze generalization between problem instances to understand not only the relevant features in universal training instances for this problem but also to aid in identifying universal training instances for other domain problems.

Generalization of evolved response thresholds is a significant finding. Dynamic thresholds hold promise due to their expected ability to adapt to changing task demands. In practice, however, they often fail to fulfill that promise as they frequently evolve to extreme values. Once there, they tend to remain unchanged, due to the effects of the positive feedback loop, and are no longer meaningfully dynamic. A GA evolves thresholds that perform almost optimally. *A priori* evolution has two potential significant shortcomings: (1) evolving thresholds for every possible problem instance is prohibitively expensive and (2) the problem instance being solved is typically unknown. We demonstrate that generalization addresses both issues as it allows users to evolve thresholds for one of a small number of problem instances, which we denote *universal training instances*, and use those thresholds for a large number of other problem instances. This discovery, and the analysis of the properties of universal training instances, is the primary contribution of this work.

2 BACKGROUND AND RELATED WORK

Self-organizing swarms are both robust and adaptable. Robustness stems from the absence of centralized control. Adaptability is a result of the large behavior space created by individual agents' independent decisions. This behavioral diversity is critical for effective division of labor [Ashby 1958]. In response threshold models, individual decisions are determined by comparison of sensed stimuli with internal threshold values. The thresholds may be global or individual. Let τ be the stimulus and θ the threshold. In the most direct form of response threshold model, an agent activates for the task if $\tau \geq \theta$. In more complex models, activation decisions may depend on additional information such as past performance, the work of other individuals, and randomness.

Inspired by observations of insect societies, the probabilistic response threshold model uses the stimulus and threshold values to define a probability of response [Bonabeau et al. 1996, 1998]. The probability that agent i activates for task j is given by

$$P_{i,j} = \frac{\tau_j^2}{\tau_j^2 + \theta_{i,j}^2}. \tag{1}$$

This results in a response probability of 0.5 when $\tau = \theta$. The probability approaches 0 if $\tau \ll \theta$ and 1 if $\tau \gg \theta$. Diversity of behavior is inherent in this model due to its stochastic nature. Even two agents with the same threshold may not respond in the same way. This model has gained wide acceptance in the multi-agent systems community. Early studies [Campos et al. 2000; Cicirello and Smith 2002; Kittithreerapronchai and Anderson 2003; Nouyan et al. 2005] focus on Morley's paintshop problem [Morley 1996]. This model has also been applied to problem domains such as mail processing [Goldingay and van Mourik 2013; Price and Tino 2004], foraging [Castello et al. 2016, 2013; Lee and Kim 2019; Lee et al. 2020; Pang et al. 2017], RoboCup rescue [Ferreira et al. 2007], UAV monitoring [Amorim et al. 2020; Schwarzrock et al. 2018], as well as abstract task allocation problems [Correll 2008; de Lope et al. 2013; dos Santos and Bazzan 2009; Ferreira et al. 2007; Merkle and Middendorf 2004; Niccolini et al. 2010]. Innovations such as token passing have been incorporated with the basic response threshold approach to facilitate swarm coordination on problems consisting of more than one task [Amorim et al. 2020; Ferreira et al. 2007; Schwarzrock et al. 2018].

Eliminating stochasticity creates a determinisitic model in which agent activation is defined by

$$P_{i,j} = \begin{cases} 1 & \text{if } \tau_j \geq \theta_{i,j} \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

This determinism allows for an understanding of agent activations that is not possible in probabilisitic models. With homogeneous response thresholds in a deterministic response model with global stimuli, agents activate in lockstep. Let $\alpha_{i,j}$ be the activation count for agent i for task j. Then $\alpha_{i,j} = \alpha_{k,j} \; \forall \; i, k$. In deterministic response models, diversity in agent activations is achieved through heterogeneity in response thresholds [Campbell et al. 2011; dos Santos and Bazzan 2009; Kanakia et al. 2016; Krieger and Billeter 2000a; Krieger et al. 2000b; Riggs and Wu 2012; Wu and Riggs 2018]. Let $\theta_{i,j} < \theta_{k,j}$ for agents i, k. Then $\alpha_{i,j} \geq \alpha_{k,j}$. That is, an agent with a lower threshold for a task will activate for that task at least as many times as an agent with a higher threshold for that task. We note that if demand exceeds an agent's thresholds for more than one task, then a secondary selection mechanism must be applied.

Task specialization is common in natural swarms [Jeanne 1986] and desired in artificial swarms due to the resultant decrease in task switching. The degree of specialization may be proportional to colony size [Holbrook et al. 2013, 2011]. While specialization has obvious benefits, it may have costs as well. For example, a specialist robot may remain idle for a long period of time as it searches for an opportunity to undertake its preferred task [Brutschy et al. 2012].

As in natural swarms, artificial swarms can benefit from specialization due to costs of task switching such as energy consumption and time and due to the positive effects of learning for some tasks. In self-organized swarms, specialization must be designed into the system, developed through adaptation, or evolved by the system. Evolution may take the form of a simple parameter adaptation to develop division of labor [Labella et al. 2006]. In some cases, agent behaviors evolve specialization to the extreme case in which agents never switch tasks. A more desirable scenario is one in which agents evolve to specialize but maintain a sufficient degree of behavioral plasticity to allow activation for different tasks [Tuci 2014]. CONE, combining neuro-evolution with co-evolution techniques, evolves controllers that improve collective performance through agent specialization for a pursuit-evasion problem and a collective construction task [Nitschke et al. 2012a, b]. As the costs associated with task switching increase, groups may be more likely to evolve specialization [Goldsby et al. 2012, 2010].

Variability in response thresholds promotes specialization, since agents with low threshold values for a task will activate more frequently, reducing task switching. The probability distribution

used to generate threshold values affects the degree of specialization [Wu et al. 2020]. In threshold-based systems, evolving threshold values provides another mechanism for self-organized division of labor [de Oliveira and Campos 2019; Duarte et al. 2012].

There is evidence that for some insect species, experience on a task causes individuals to specialize for that task. For example, some species are known to increase activation probability for a task with experience performing that task [Gautrais et al. 2002; Langridge et al. 2004; Ravary et al. 2007; Theraulaz et al. 1998]. For example, *Leptothorax albipennis* and *Cerapachys birori* ants are more likely to activate for tasks on which they have had past success. This suggests that individuals adapt response thresholds with experience. This form of adaptability has been implemented in numerous multi-agent systems [Campos et al. 2000; Castello et al. 2016, 2013; Cicirello and Smith 2002; de Lope et al. 2013, 2015; Gautrais et al. 2002; Goldingay and van Mourik 2013; Kazakova and Wu 2018; Kazakova et al. 2020; Price and Tino 2004; Theraulaz et al. 1998]. By adapting agent behavior to changing task demands, dynamic response thresholds appear to promise improved specialization and swarm performance. In practice, however, agent thresholds often migrate to sink states in implementations that act as positive feedback loops. This achieves increased specialization but fails to deliver on adaptability [Kazakova and Wu 2018; Kazakova et al. 2020; Theraulaz et al. 1998]. Sink states may be determined by the first tasks for which agents activate [Meyer et al. 2015]. A recent alternative that is less susceptible to sink states is to adjust agent thresholds based on perceived need [Lee and Kim 2019; Lee et al. 2020].

Evolutionary computation has been used to evolve several aspects of artificial swarms. One of the most common applications is evolving agent and swarm behaviors. These include the use of grammatical evolution to evolve collective behaviors [Neupane and Goodrich 2019; Neupane et al. 2018], evolution of behavioral rules for a swarm [Samarasinghe et al. 2018], evolving diversity using a decentralized Multi-dimensional Archive of Phenotypic Elites algorithm (MAP-Elites) [Hart et al. 2018], evolving collaboration using cooperative coevolution [Panait et al. 2006], evolving cooperative behaviors [Wang et al. 2019], and evolving selfish behaviors [Yamada and Sakama 2013]. Swarm size has also been examined with respect to evolving swarm behaviors [Fischer et al. 2018].

Evolution of swarm organizations and coalitions is also the subject of significant research [Aubert-Kato et al. 2017; Guo et al. 2020; Yu et al. 2011]. Population management, in response to changing threat levels, has been evolved as well [Beckman and McKinley 2008].

In the domain of swarm robotics, evolution of control mechanisms is an active research area. Work in this area typically involves one of several forms of neuro-evolution. Examples include evolving controllers for: self-organization for a swarm of *s-bots* [Dorigo et al. 2004], aggregation behaviors [Baldassarre et al. 2003; Gomes and Christensen 2013; Gomes et al. 2013; Soysal et al. 2007; Trianni et al. 2003], coordinated motion [Baldassarre et al. 2007; Sperati et al. 2008], collective behaviors of autonomous vehicles [Hunag and Nitschke 2017], specialization for a robotic team undertaking a construction task [Nitschke et al. 2012b], communication network formation [Hauert et al. 2009], exploration and navigation [Sperati et al. 2011], rough terrain navigation [Trianni et al. 2006], transport problems [Groß and Dorigo 2008], agent communication [Ampatzis et al. 2008], learning behaviors [Pini and Tuci 2008], primitive behaviors triggered by a pre-programmed arbitrator [Duarte et al. 2014], several behaviors for aquatic surface robots [Duarte et al. 2016a], and intruder detection [Duarte et al. 2016b]. In addition, researchers have explored the relationship between evolution and the environment [Steyven et al. 2017].

Evolutionary computation is used to evolve agent-level parameters to elicit desirable swarm behavior [Narzisi et al. 2006] and agent-level ranking schemes for task selection [Moritz and Middendorf 2015].

As noted above, division of labor is an important factor in swarm success. In this, too, evolution may play a role. In evolutionary robotics, collective neuro-evolution is used to

evolve agent specialization for gathering and collective constuction problems [Nitschke 2009; Nitschke et al. 2012a, b]. Division of labor has been evolved in groups of clonal organisms [Goldsby et al. 2010]. Using grammatical evolution, specialization is evolved via task partitioning [Ferrante et al. 2015].

Duarte et al. evolved response thresholds for a probabilistic response-based swarm with two tasks [Duarte et al. 2012]. Two different single-objective outcomes are examined: work distribution and specialization. Task stimuli increase by a constant in every timestep and decrease by the work performed. The balance between task demands is fixed at either 1:1 or 3:1 throughout a run. They find that threshold values evolve to 0 when work distribution is the evolutionary objective, leaving assignment of workers to tasks entirely dependent on stimulus parameters. Specialization develops when costs associated with task switching are high. Our work differs from this in several significant ways. First, we use a deterministic threshold-based system. More importantly, our testbed allows exploration of task demands that are dynamic and highly variable requiring a much greater degree of adaptability from the swarm (see Section 3 for details). Finally, we explore the degree to which thresholds evolved for one set of task demands generalize to problem instances with very different task demands.

3 TASK ALLOCATION IN A THRESHOLD-BASED SYSTEM

The testbed we use in this work is a highly dynamic task allocation problem modelled as two-dimensional and three-dimensional collective control problems. Weidenmüller describes the problem of honeybee nest thermoregulation, a one-dimensional collective control problem from nature [Weidenmüller 2004]. We increase the number of dimensions and cast the problem as 2D and 3D tracking problems [Wu et al. 2021] to allow more natural visualization of results. We note, however, that the focus of this work is on achieving effective decentralized task allocation via inter-agent variation. Tracking provides a framework for presenting highly varied schedules of dynamic task demands, each of which is defined by a path dictating movement of the target object. The experiments in this work would be largely unchanged if we cast the problem into a different domain, including those without a physical manifestation. Thus, we abstract physical considerations out of the problem.

A simulation consists of a pre-determined number of timesteps. A target and a tracker object begin at the same location in a 2D or 3D space. Over the course of a simulation run, a target moves along a specified path in that space. A swarm collectively pushes a tracker object, attempting to follow the target as closely as possible. In two dimensions, four tasks are defined: `push_NORTH`, `push_EAST`, `push_SOUTH`, or `push_WEST`. Increasing to 3D adds `push_UP` and `push_DOWN`. Each agent i has a threshold $\theta_{i,\mathrm{D}}$ for each $D \in$ {`NORTH, EAST, SOUTH, WEST [, UP, DOWN]`}. Thresholds are in $[0, 1]$ and are mapped to the domain space via the `range` parameter. Task demands are determined by the relative positions of the target and tracker. In each timestep of a simulation run, let $\Delta x = \text{target}.x - \text{tracker}.x$ and $\Delta y = \text{target}.y - \text{tracker}.y$. Task stimuli are defined as $\tau_N = -\Delta y$, $\tau_E = -\Delta x$, $\tau_S = \Delta y$, $\tau_W = \Delta x$, $\tau_U = -\Delta z$, and $\tau_D = \Delta z$. The goal of the swarm is for agents to self-allocate themselves among the tasks in each timestep such that they satisfy all task demands in each timestep.

3.1 Target Motion

In each timestep, the target moves a fixed distance defined by parameter `Target_step_len`. The direction in which a target moves in each step of a simulation is determined by a target path. The defining characteristic of our testbed is that, by changing the target path, we can dramatically and dynamically alter the demand placed on the swarm with respect to task demands. Demand can

change gradually or abruptly, at a constant rate or a changing rate. The target can move and turn in all directions or in only some. A single problem instance can include many of these characteristics at once.

For example, in a `zigzag` path, demand for `push_EAST` is constant while, for `push_NORTH` and `push_SOUTH`, it is punctuated by abrupt changes (see Figure 1). Other than these occasional abrupt changes in demand for `push_NORTH` and `push_SOUTH`, task demands remain constant and unchanged from one timestep to the next. Over the course of an entire run, the total demand for `push_EAST` is highest, the total demand for `push_NORTH` and `push_SOUTH` are approximately equal, and there is no demand for `push_WEST`. In contrast, a `circle` path exhibits constant gradual change in demand for all tasks from one timestep to the next (see Figure 2). Over the course of an entire run, the total demand for each task is approximately the same. A `random` path generates random changes in demand for all tasks from one timestep to the next (see Figure 3).

The variety and dynamism of demands that can be placed on the swarm make this an ideal testbed for task allocation. In 2D, we examine six target paths in our experiments: `circle`, `diamond`, `random`, `scurve`, `square`, and `zigzag`. Detailed descriptions of each task appear below.

- `circle`: Target continuously revolves about a central point at a fixed distance, resulting in a circular path with radius r. This creates continuously changing task demands and requires the swarm to perform all tasks equally over the course of a run.
- `diamond`: Target moves continuously along the perimeter of a square rotated 45° from the axes. Thus, all motion creates simultaneous demand for two tasks. Size is determined by parameter `Edge_length`.
- `random`: In each time step, target direction is calculated from the current heading by adding an angle, in radians, drawn from the Gaussian distribution $\mathcal{N}(0.0, 1.0)$. With high probability, random requires all tasks. See Figure 3.
- `scurve`: A periodic, rounded path that oscillates north and south, moving from west to east. The motion is defined by parameters `Path_amplitude` and `Path_period`. Requires all tasks, however, there is very little demand for `push_WEST`. See Figure 4.
- `square`: Target continuously moves along the perimeter of an axis-aligned square with edge length defined by parameter `Edge_length`. Requires all tasks during a run but motion at any given time is in only one direction.
- `zigzag`: A periodic, oscillating path moving from west to east with identical, angular straight edges moving alternately northeast and southeast. As for `scurve`, amplitude and period are parameterized. No demand for `push_WEST` but all motion creates simultaneous demand in two dimensions. See Figure 1.

We examine six additional paths in 3D: `baseball_stitching`, `circular_sine`, `random`, `slinky`, `spiral`, and `spiral_updown`. Detailed descriptions appear below.

- `baseball_stitching`: Target moves along a smooth approximation of the stitching on a baseball. This generates small changes in demand in all three dimensions in every timestep. See Figure 5.
- `circular_sine`: Target path is a sine curve projected on a cylinder with a parameterized diameter. Demand in the z dimension is circular while x and y are defined by sine. Thus, changes in demand for all tasks are continuous and small.
- `random`: The 3D random path is analogous to the 2D version.
- `slinky`: Target moves along a path defined by a coil wrapped on a torus. While creating continues, small changes in demand for all tasks like other 3D paths, `slinky` provides significant variation in the balance of demands. See Figure 5.

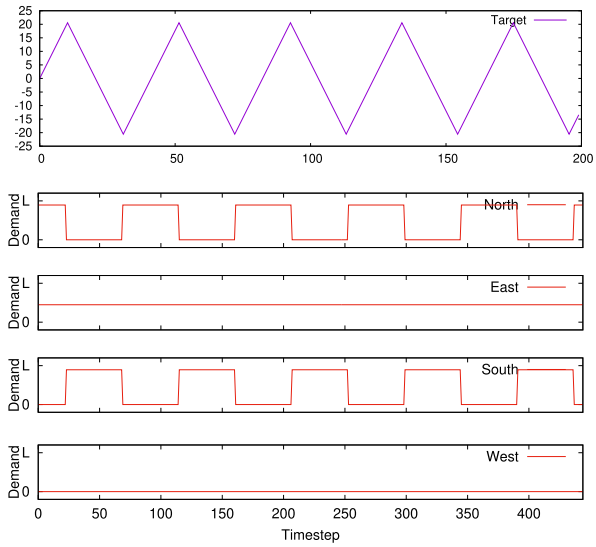

Fig. 1. Zigzag path and corresponding task demands. L represents the maximum possible demand.

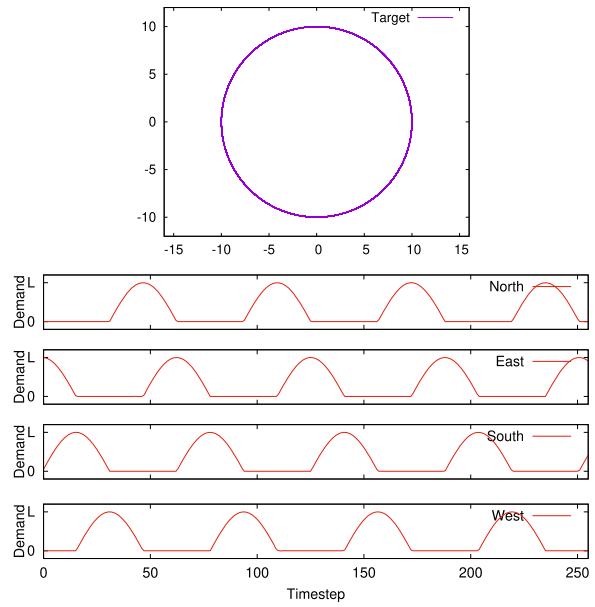

Fig. 2. A circle path and corresponding task demands. The target travels along the circular path for approximately four rotations. L represents the maximum possible demand.

- `spiral`: Target moves along a path defined by a coil wrapped on a cylinder. Demand for `push_UP` is constant. There is no demand for `push_DOWN`.
- `spiral_updown`: Target moves along a path defined by a coil wrapped on a cylinder. Movement is up in the first half of a simulation and down in the second half.

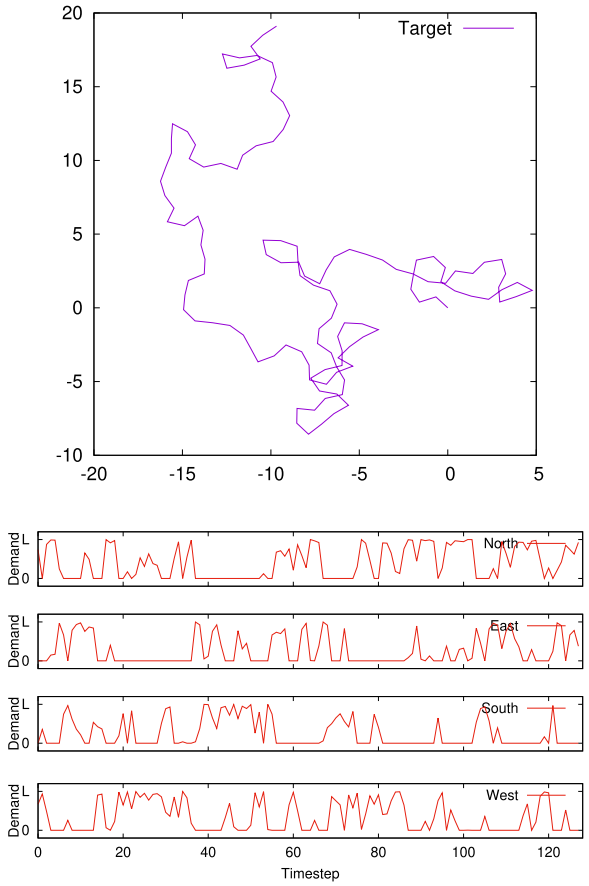

Fig. 3. Random path and corresponding task demands. L represents the maximum possible demand.

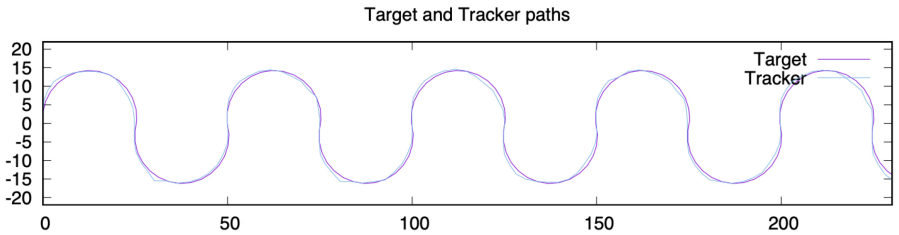

Fig. 4. Example of an scurve path.

3.2 Tracker Response

Tracker motion is determined by the aggregate agent task choices. In each timestep, each agent activates for one task or remains idle. Candidate tasks for agent i are those for which global stimulus $\tau_{\mathrm{D}} \geq \theta_{i,\mathrm{D}}$. If there is more than one candidate task in a timestep, then selection is random.

In each timestep, the tracker movement is determined as follows for each direction $D \in \{\texttt{NORTH, EAST, SOUTH, WEST [, UP, DOWN]}\}$ Let N be the number of agents in a swarm. Let n_D be the number of agents pushing in direction D. Parameter `step_ratio` is the ratio of

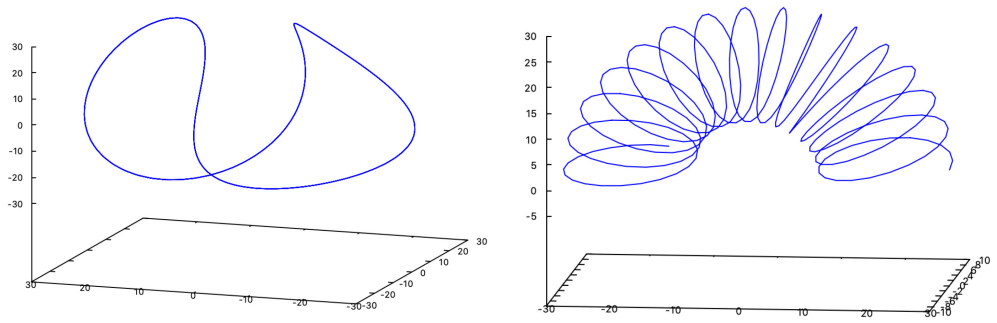

Fig. 5. Examples of `baseball_stitching` and `slinky` paths.

the maximum distance that a tracker can move in one timestep relative to the `Target_step_len` parameter. The distance d_D that the tracker moves in each direction D in a given timestep is

$$d_D = \frac{n_D}{N} \times \texttt{Step\_ratio} \times \texttt{Target\_step\_len}.$$

Our interest in this work is to examine the efficacy of evolved, static response thresholds. We compare the results for evolved thresholds with previous methods for assigning agent response thresholds that have performed well, including uniform static thresholds and dynamic response thresholds. In uniform static thresholds, agent thresholds for all tasks are generated uniformly at random in $[0, R]$, where R is a parameter that specifies the maximum threshold range. In the model of dynamism used for comparison in this work, threshold values can vary in $[0, R]$. $\theta_{i,\mathrm{D}}$ decreases in each timestep during which agent i activates for task D and increases in each timestep during which the agent activates for another task [Wu and Mathias 2020]. Threshold values are initialized uniformly at random in $[0, 1]$.

3.3 System Performance

Performance is evaluated according to the three following goals:

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

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

Goal 3. *Minimize the average number of task switches, per agent, over the course of a simulation run.*

The first two goals measure system accuracy. It is important to note that both accuracy criteria are necessary to gauge the swarm's success. If using only Goal 1, then the tracker could remain close to the target while zigzagging repeatedly across the target path. This would yield strong performance with respect average positional difference but a path length that is significantly greater than that of the target. Alternately, using only Goal 2, the tracker might travel a path that is the same length as that of the target but unrelated with respect to shape and direction.

The third goal measures system stability. In problems for which changing tasks is costly (in terms of time or energy), it is helpful for agents to minimize task switches. This measure is used as an indication of agent specialization.

It is not our goal to find the most efficient method to solve the tracking problem. Instead, this problem serves as our testbed because it allows us to present the swarm with schedules of dynamic

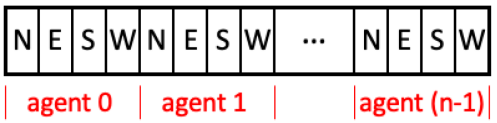

Fig. 6. Structure of the genome. It consists of a threshold for each of the tasks for each agent. The thresholds are real-valued.

task demands and to dramatically change the schedules by substituting one problem instance (path) for another. Tracking is a decentralized task allocation problem in which task demands are clearly defined and measured, dynamic variation in task demand over time can be systematically described, and overall performance can be accurately measured as well as visually assessed. We reiterate that our focus is on decentralized, dynamic task allocation via inter-agent variation in general and variation in response thresholds in particular. Modeling physics, or physical robots, is not the purpose of this work.

4 THE GENETIC ALGORITHM

Our GA is the well-known Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al. 2002] implemented in Python using the Distributed Evolutionary Algorithms in Python library [Fortin et al. 2012]. Though there are newer multi-objective GAs, NSGA-II is well understood, widely used, and very effective for small numbers of objectives. We use three optimization objectives:

- minimize average positional difference: the average, over all timesteps in a simulation, of the distance between the target and tracker
- minimize path length difference: the difference in total distance traveled by the target and tracker
- minimize average number of task switches: the average, over all agents, of the number of switches from one task to another

Each objective aligns with one of the system goals defined in Section 3.3. Each of the m individuals in the GA population represents a complete swarm consisting of n agents.

We note that average distance and total path length are not conflicting measures: It is possible to minimize both simultaneously; however, minimizing one does not necessarily minimize the other. Average task switches are orthogonal to the accuracy measures. GA runs performed without task switches as an objective demonstrate that accuracy results are strong but task switches increase significantly.

It might be possible to scalarize our objectives and use a single objective algorithm. There are, however, several potential obstacles to this,

- Effective scalarization is difficult to achieve, particularly when the scales of the objectives vary widely.
- In the end, we want to chose the individual that provides the smallest average positional distance found. With a single objective algorithm, it will be more difficult to make this selection.
- It is not clear that average switches is non-conflicting with respect to the other two objectives.

Each individual in the GA encodes the response thresholds for an entire swarm. Thus, the genome consists of kn real-valued numbers, k for each of the n agents represented by an individual, where k is the number of tasks (see Figure 6). The values for an agent represent the four task thresholds for that agent, each a real value in $[0, 1]$. Thus, in aggregate the genome is:

$[\theta_{i,\mathrm{D}}]$, $\forall$ D $\in$ {NORTH, EAST, SOUTH, WEST} and $\forall\ i \in [0, 49]$. Individuals are initialized with threshold values generated uniformly at random in $[0, 1]$.

NSGA-II is based on the concept of *domination*. Let $o_i, i \in \{1, \ldots, t\}$ be the optimization objectives and a, b be individuals in the GA population. We denote the value of o_i for individual a as a_i. Assuming that all objectives are to be minimized, individual a dominates individual b if $a_i \leq b_i\ \forall i \in \{1, \ldots, t\}$ and $\exists\ i\ \in \{1, \ldots, t\} : a_i < b_i$. That is, a is no worse than b for all objectives and is better than b for at least one objective.

NSGA-II imposes a partial order on the population using domination. This takes the form of placing individuals in a sequence of fronts. Front 0 consists of non-dominated individuals. Front j consists of individuals that are dominated by at least one individual in front $j - 1$.

In each generation, we create an offspring population that is initially a copy of the parent population. Offspring are randomly paired for crossover, which occurs with probability 0.9. Individuals that undergo crossover are mutated with probability 0.2 while those that do not are mutated with probability 1.0. After crossover and mutation are complete, the offspring are combined with the parent population. This combined population is winnowed to the defined population size via the replacement operator. More detailed descriptions of these genetic operators follow.

- crossover: The crossover operator is uniform with threshold values as the unit of exchange. Given parents p_1 and p_2, child c_1 inherits each threshold θ_j from p_1 with probability p and from p_2 with probability $1 - p$. Construction of child c_2 is symmetric.
- mutation: Each agent within a GA individual is mutated with a probability dependent on the swarm size. An agent mutation consists of choosing one threshold value at random, with equal probability, and adding a small Δ generated uniformly at random in a parameterized range centered at 0.
- replacement: We use the standard NSGA-II replacement operator. A non-dominated sort places the individuals into a series of fronts. Beginning with front 0, all individuals in each front are retained in the population until a front is reached such that retaining all individuals in that front would cause the population to exceed the defined population size. Call this front q. Since only some individuals in front q can be retained, they are distinguished using a secondary measure called the *crowding distance*. The crowding distance is, in effect, a measure of novelty. In selecting individuals from front q, more novel individuals are preferred.

Fitness for an individual in the GA population is determined by running the simulator using the threshold values for that individual and the objectives defined above. For deterministic paths, evaluation consists of a single simulation while for random paths we run three simulations and average the results.

5 EXPERIMENTAL METHODS

We demonstrate the scalability of our approach in two ways. First, we evolve thresholds for swarms with 50, 100, 500, and 1,000 agents. Second, we use 2D and 3D simulations, with four tasks and six tasks, respectively. In the latter case, we limit the experiments to 50 agents.

For each target path, we perform 32 runs of the GA. At the conclusion of each run, the individuals in the non-dominated front are sorted by their values for the average positional difference. The individual with the smallest value for this objective is selected. The thresholds from that individual are used in subsequent testing. For each test, we perform 30 runs of the simulation, averaging positional difference, path length difference and number of task switches across the runs. Table 1 lists the GA and simulator parameters used in our experiments.

Individuals in the GA population are evaluated by running swarm simulations. This is performed for each unevaluated offspring in each generation, up to m individuals per generation, where m

Table 1. Simulation Parameters (top) and GA Parameters (Bottom)

Simulation Parameter	Value
Number of agents, n	50, 100, 500, 1000
Simulation timesteps, training	200
Simulation timesteps, testing	500
Number of tasks, k	4, 6
Task selection	random
Target_step_len	3
Step_ratio	2.0

GA Parameter	Value
Population size, m	100 for $n = 50, 100$ 200 for $n = 500, 1000$
Initialization	$\mathcal{U}(0.0, 1.0)$
Mutation Δ	$\mathcal{U}(-0.1, 0.1)$
Parent selection	random
Crossover	uniform
Crossover probability	0.9
Uniform probability	0.7
Replacement	NSGA-II
Generations	2500

Values for Target_step_len and Step_ratio are used to allow comparison with previous results.

is the GA population size. For a deterministic path, this results in up to 500,000 simulations per GA run. To reduce the time required for these runs, we use only 200 timesteps per simulation for evaluation. During testing of the evolved thresholds, we use 500 timesteps per simulation to allow comparison with previous results.

To understand the effects of response thresholds evolved using a GA, we perform experiments using thresholds created in several ways.

- uniform: Generated uniformly at random, for each task for each agent. These thresholds are static, remaining unchanged throughout a simulation.
- dynamic: Thresholds that vary within a range. For each task for each agent, the ranges are generated by determining the minimum value uniformly at random in $(0.0, 0.5]$ and the maximum value uniformly at random in $(0.5, 1.0]$. When an agent activates for a task, its threshold for that task decreases by a learning factor. Conversely, the threshold for a task increases by a forgetting factor when the agent does not activate for that task. An agent's initial threshold value for a task is the midpoint of the agent's range for that task.
- evolved: Thresholds evolved *a priori* using a GA, as described in Section 4. Once evolved, the values are static.
- rand-evo: Thresholds evolved for the random path. They are tested on all paths to demonstrate generalizability.
- shuffled: Evolved thresholds that are permuted across the agents for each task. This maintains the *set* of thresholds for a task but changes the sequence of those thresholds, assigning them to different agents. Doing this disrupts the evolved balance of threshold values within each agent. This is done to explore the effect of the distribution of thresholds within agents.

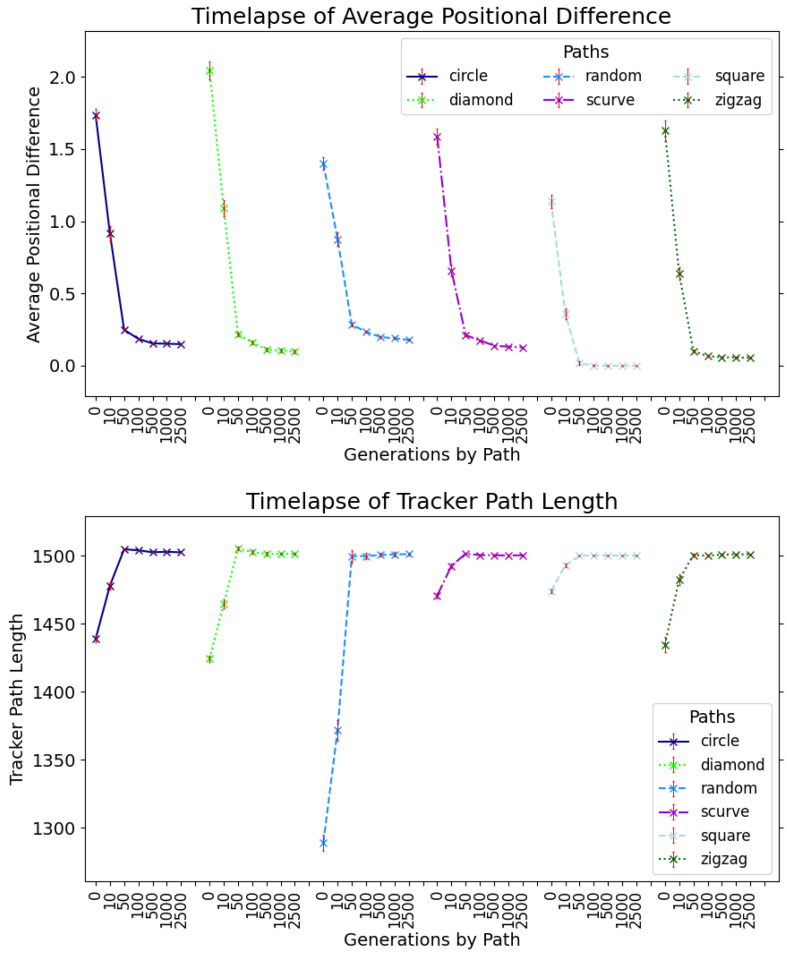

Fig. 7. Convergence of threshold evolution by target path. Each data point represents 30 simulation runs for each of the 32 GA runs. The 95% confidence intervals are shown in red. For each series, generation 0 represents thresholds generated uniformly at random, the starting point for evolution.

6 RESULTS

In this section, we provide experimental results demonstrating that evolved response thresholds outperform uniform thresholds, dynamic thresholds, and shuffled thresholds with respect to tracking metrics. In addition, thresholds evolved for one problem instance generalize, to varying degrees, to other problem instances. We identify a class of problem instances for which our algorithm evolves *universal response thresholds* that provide nearly optimal performance for all other instances tested. Finally, we analyze these universal training instances to identify features that make universality possible.

6.1 Effect of Evolved Thresholds

Figure 7 illustrates the effect of evolved thresholds on swarm performance for the 2D tracking problem. Each plot shows data for each of the six problem instances tested. The x-axes show

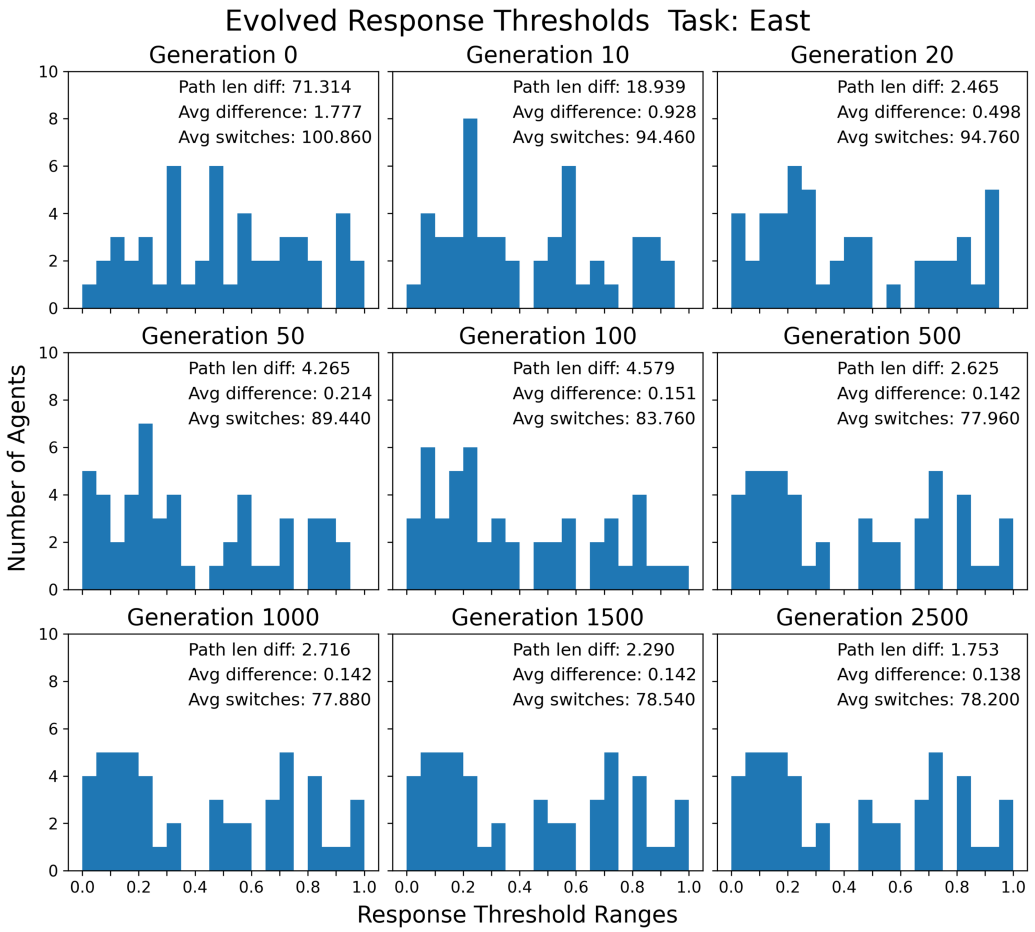

Fig. 8. Evolution of thresholds for a swarm of 50 agents. Initial thresholds are uniformly distributed at random. The histogram shows 20 buckets of uniform width. The GA is run for 2,500 generations, though for the run shown the histogram does not change further after generation 500.

generations of evolution for each of the instances. In the top plot, the y-axis represents the average positional difference between target and tracker. The y-axis in the bottom plot represents the tracker path length. Target path length for the runs shown is 1,500. Each data point represents 32 simulations, one for each run of the GA for that path. Each run was performed on the individual from the non-dominated front that achieved the minimum average positional difference; 95% confidence intervals are shown in red.

Recall that the thresholds of the initial population in the GA are uniformly distributed at random. Therefore, the generation 0 data points represent thresholds generated uniformly at random. Evolved thresholds significantly improve performance with respect to both domain goals for all problem instances. For tracker path length, performance is nearly optimal for all paths.

Figure 8 shows a series of histograms depicting the threshold values for task `push_EAST` at nine times during a run of the GA for target path `circle` with radius 10. On the x-axis, the threshold

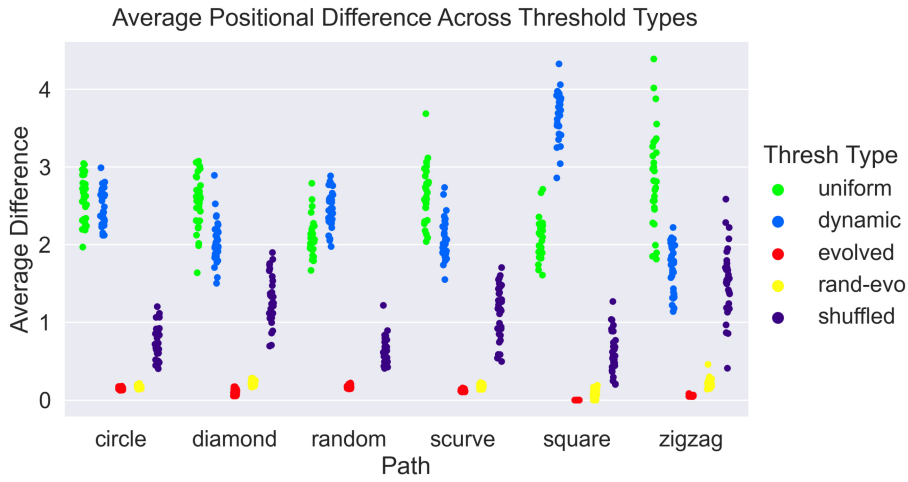

Fig. 9. Comparison of swarm performance for six problem instances (paths) using response thresholds generated in five ways: distributed uniformly at random, dynamically adjusted during a run, evolved by a GA for the test path, evolved by a GA for the `random` path, and shuffled. Shuffled thresholds are evolved for the test path but are then randomly permuted across agents for each task. This breaks the evolved balance of thresholds for each agent. Shuffled thresholds are discussed further in Section 6.1.

range of [0.0, 1.0] is divided into 20 buckets of width 0.05. The y-axis is the count of agents in each bucket. The swarm consists of 50 agents. In generation 0, the thresholds are the initial values, generated uniformly at random. In each subplot, the swarm's performance using the current threshold values is shown at the upper right. The individuals depicted are selected from the non-dominated front.

This sequence shows that the number of low thresholds very quickly increases, resulting in more agents activating for this task. For the GA run shown, the histogram for `push_EAST` is fixed by generation 500, though values within the buckets may continue to change.

Examining activation counts for agents with the final thresholds reveals that only 25 of the agents activate for `push_EAST`. In the plot for generation 500 in Figure 8, 24 agents are represented in the block with thresholds between 0.0 and 0.3. All of these agents, plus one of the agents in range 0.3 to 0.35 (with only 10 activations), activate at least once. None of the remaining agents activate for this task, `push_EAST`. This means that the thresholds for `push_EAST` for these agents constitute, in effect, a non-coding region of the genome as they do not affect fitness.

Introducing dynamism in response thresholds may allow a swarm to adapt to changing task demands and different problem instances. Because we evolve response thresholds for particular problem instances, it is natural to compare evolved thresholds to dynamic thresholds to determine if the expected benefit of dynamism is observed. Figure 9 illustrates this comparison. The x-axis consists of six groups, one for each problem instance. Within each group, we show data points for uniform thresholds, dynamic thresholds, thresholds evolved for that target path, and thresholds evolved for `random` paths but tested on the target path (rand-evo). In the group for `random`, we omit the fourth data, since they duplicate the third. The y-axis represents average positional difference. Each column represents 30 runs of the simulation for each of the 32 runs of the GA. Results for tracker path length are similar.

These results show that evolved thresholds significantly outperform both uniform thresholds and dynamic thresholds. Dynamism is beneficial for some paths but not for others. This may be due to the effects of the positive feedback loop on dynamic thresholds, causing them to migrate to

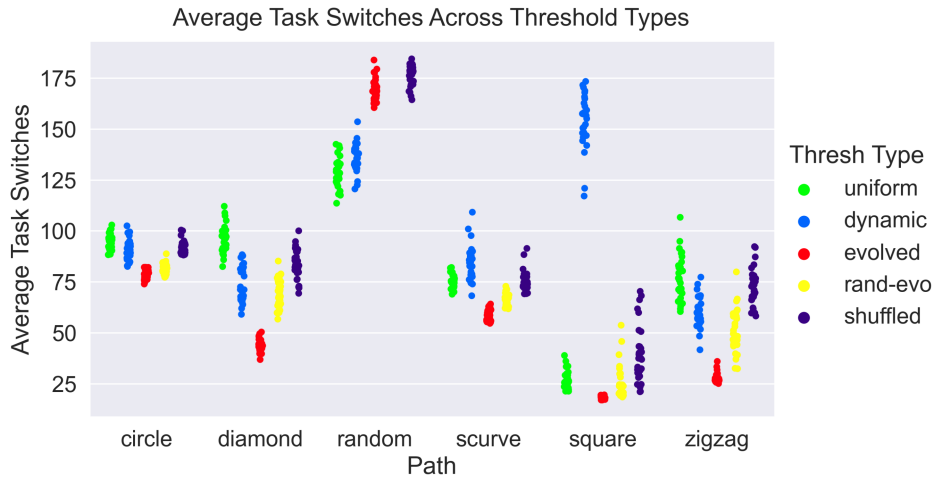

Fig. 10. Comparison of task switches for six problem instances (paths) using response thresholds generated in five ways.

sink states. One potential advantage of dynamic over evolved thresholds is that they adapt in real time to any path, perhaps making them more general than evolved thresholds. The rand-evo data demonstrate that this is not the case. Using thresholds evolved for `random` we find excellent performance for all paths suggesting that evolved thresholds can generalize even when used statically by the swarm.

Evolved thresholds also have a beneficial impact on specialization for most paths. For each of the 32 runs of the GA, we perform 30 swarm simulations using the evolved response thresholds. Figure 10 shows the average task switches for these simulations for all six problem instances. Improvement ranges from modest for `square`, for which specialization is unnecessary due to demand for only one task in any timestep, to substantial for `diamond` and `zigzag`, which share the properties of long periods of unchanging task demand and demand for multiple tasks in all timesteps. `random` is an outlier with a high number of task switches for evolved thresholds. We hypothesize that this is due to each evaluation of individuals during evolution using different `random` paths. Thus, while the balance of thresholds for each task is well developed, as evidenced in Figure 9, the balance across tasks for each agent is not.

To gauge the effect of the distribution of thresholds within an agent, we randomly shuffle the evolved thresholds by permuting the thresholds for each task. This maintains the distribution of threshold values for a task but disrupts the relative values across tasks for each agent. We shuffle the evolved values for all 32 GA runs for each of the six problem instances. We then perform 30 simulations with each set of shuffled thresholds. The results are represented by the "shuffled" data series in Figure 10. With shuffled thresholds, specialization is dramatically reduced compared to unshuffled evolved thresholds for all paths except `random`. For this path, shuffled and unshuffled thresholds exhibit similar specialization.

For the domain goals, evolved thresholds perform well for `random`, as shown in Figure 9. The figure shows that shuffled thresholds perform significantly better than uniform thresholds due to the improved distribution of thresholds for a task. They are not, however, as effective as the unshuffled evolved thresholds due to the lack of coordination of thresholds for each agent. Significantly, this demonstrates that static thresholds (those evolved by the GA) can perform almost optimally even for problems in which the task demands are highly dynamic.

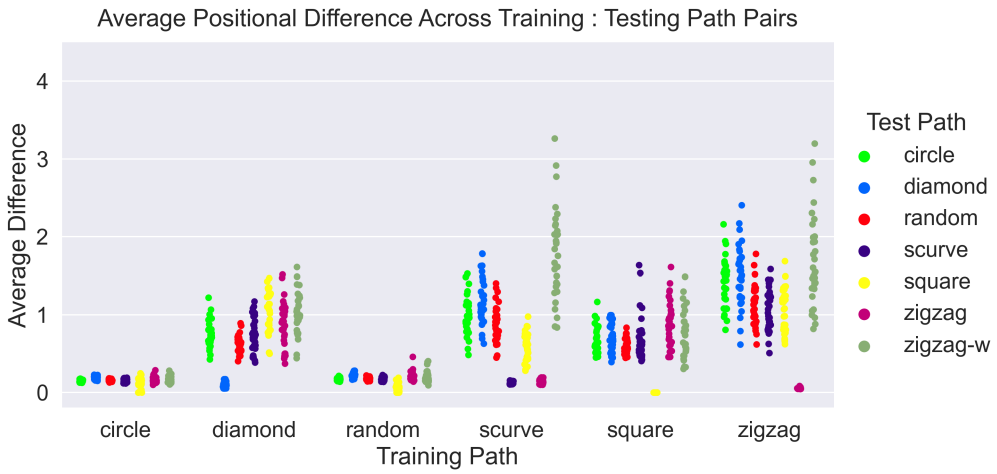

Fig. 11. Generalization of thresholds evolved for six paths. `circle` and `random` provide the best generalization.

6.2 Generalizing across Paths

A reasonable concern for *a priori* evolution of response thresholds is that the result will be specific to the problem instance used during evolution. If this is the case, then evolution of thresholds will be required for each problem instance to be solved, a time-consuming undertaking. We investigate this by testing the generalization of evolved thresholds across problem instances with greatly varying schedules of task demands. We show that response thresholds evolved for some problem instances generalize well to all other instances. By performing additional experiments, we attempt to identify the instance properties that facilitate generalization.

We have demonstrated that evolved thresholds result in better swarm performance, for the path for which they were evolved, than uniformly distributed thresholds. Uniformly distributed thresholds generalize well. That is, they perform well for all paths. This is not surprising, since uniformly distributed thresholds are general by definition. It is not obvious that thresholds evolved for the task demands of one problem instance should perform well for another instance with different demands. In this section, we show that evolved thresholds can generalize and that some problem instances result in thresholds that provide excellent performance for all others.

Figure 11 depicts the degree of generalization for evolved thresholds. The x-axis labels represent the path used for evolution. Each color represents a path used for testing. Each column represents 32 data points, one for each run of the GA. Each data point is the average of 30 simulations using one set of evolved thresholds. The y-axis shows average positional difference between the target and tracker.

The figure shows that `circle` and `random` generalize to all other paths. `circle` does slightly better than `random` for all paths except `square`. Neither `square` nor `zigzag` generalize well, though we note that the performance is approximately the same as uniform thresholds for each testing path. `scurve` generalizes well to `zigzag` but not to other paths. The trend is the same for the tracker path length objective, though the plot is omitted for space considerations.

The first step in understanding these results is to revisit the path descriptions. For `zigzag`, movement alternates periodically between line segments to the northeast and southeast. Inspired by a sine wave, `scurve` creates continuously changing task demands. This explains the asymmetry in generalization between these two paths. `zigzag` thresholds are not able to address the changes in

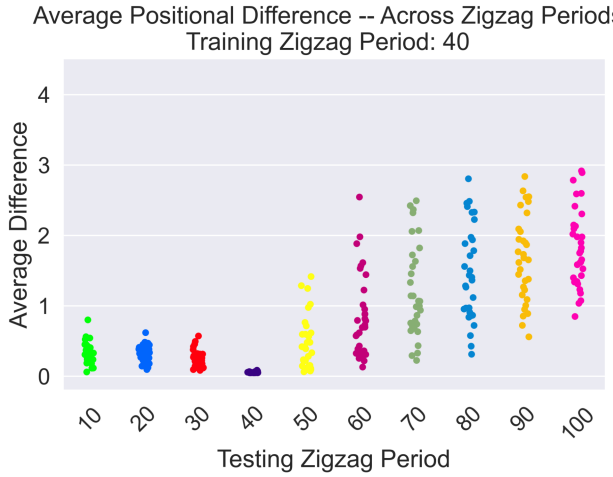

Fig. 12. Generalization results for thresholds evolved for zigzag with period 40, tested on zigzag with periods of 10 to 100.

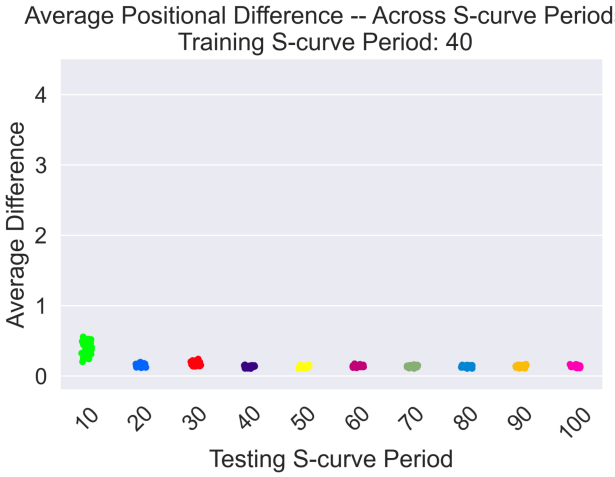

Fig. 13. Generalization results for thresholds evolved for scurve with period 40, tested on scurve with periods of 10 to 100.

task demands created by `scurve`. We hypothesize that thresholds evolved for `zigzag` specialize to the balance of task demands presented by a path that follows only two slopes. To test this hypothesis, we use thresholds evolved for `zigzag` with period 40 on runs for other `zigzag` instances with different periods. Changing the period while keeping the amplitude fixed, changes the slopes of the `zigzag` path segments. As seen in Figure 12, this minor change significantly degrades performance. In contrast, `scurve` thresholds work well for `zigzag` as they are not specialized to a particular balance of task demands. Figure 13 shows testing of `scurve` thresholds evolved for period 40 on `scurve` instances with periods from 10 to 100. There is very little change in performance across period values.

Neither `scurve` nor `zigzag` generalize well to `circle`, `diamond`, `random`, or `square`. This is because `zigzag` creates no demand for task `push_WEST` and `scurve` creates very little. Therefore,

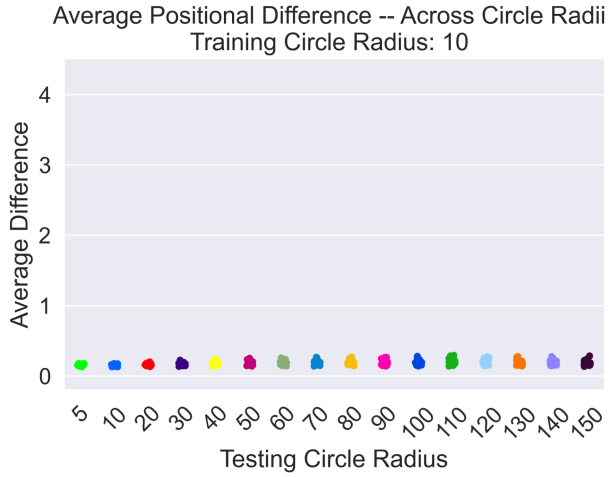

Fig. 14. Generalization results for thresholds evolved with a circle with radius 10, tested on circles with radii of 5 to 150.

west thresholds do not significantly affect individual fitness during evolution creating very little evolutionary pressure on these values. As a result, `scurve` and `zigzag` thresholds' ability to satisfy `push_WEST` demand is no better than chance. The last column in each group in Figure 11 illustrates this point. These data represent tests performed on `zigzag-w`, a version of `zigzag` in which the major direction of travel has been reversed to west but is identical to `zigzag` in all other respects. `circle` and `random` generalize to `zigzag-w` with almost identical results as those for `zigzag`. Neither `scurve` nor `zigzag` generalize to `zigzag-w`, supporting our claim.

`square` requires further analysis. `push_WEST` comprises one-quarter of the task demand for `square`, yet it does not generalize as well as `circle` and `random`, which have similar west demand as a fraction of total task demand. We hypothesize that this results from task demand for `square` being limited to one direction in any timestep. Thus, for square, there is no need for specialization as there is only ever one task to undertake at any time. Thresholds evolved in this environment, therefore, do not perform well when demand for multiple tasks exists.

We present results for `diamond` to add additional support for our hypotheses regarding the experimental results. Recall that `diamond` is simply `square` rotated 45° so that the corners are aligned with the four cardinal directions. Therefore, like `zigzag`, `diamond` consists of straight segments each of which creates demand for two tasks but like `square` it creates demand for all four tasks during the course of a simulation. Results for `diamond` are similar to those for `square`, though slightly worse for most target paths. As with `zigzag`, `diamond` evolves thresholds with a fixed balance between tasks for which there are simultaneous demands. `square` does not suffer from this effect. This conclusion is reinforced by the similar performance for `diamond` and `zigzag` for test path `square` despite the apparent advantage for `diamond` due to demand for `push_WEST`. That advantage is apparent in the performance for test path `zigzag-w` (Figure 11).

`circle` serves as a particularly robust training instance. As shown in Figure 11, `circle` thresholds generalize to all other paths. In addition, they are effective for `circle` instances independent of radius. A radius of 10 is used for evolving thresholds. As Figure 14 shows, tests on `circle` with radii from 5 to 150 show no significant change in performance. Further, thresholds evolved via GA runs in which fitness evaluation uses simulations making as little as one revolution of the `circle` results in values with the same performance as simulations making multiple revolutions. Figure 15 provides these results. Using only 0.75 revolutions results in only slight degradation.

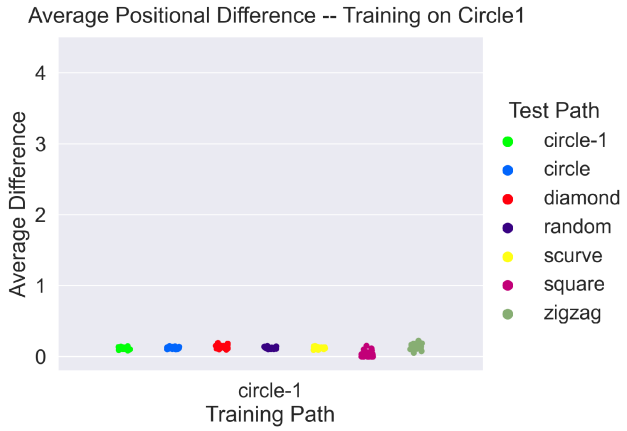

Fig. 15. Generalization results for thresholds evolved with fitness determined by only one revolution of a `circle`. The y-axis matches the previous plots to facilitate comparison.

These results suggest two necessary and sufficient features for a universal training instance: sufficient demand for all tasks, and a wide range of simultaneous demand levels to allow thresholds to evolve to reasonably address any balance of demand between multiple tasks. It is important to note that this does not require that the training instance includes every possible turn or curve that might be encountered in testing. A simple circle in which demand changes in the same way, from one timestep to the next throughout a run, is universal. There are no left turns, no sharp turns and no straight lines. The number of timesteps during which there is demand for only one task is extremely small as these occur only when the target and tracker are at the same x or y location but are not co-located. Though a very different path, `random` provides similar task demands with respect to variety in the balance of simultaneous demand for multiple tasks and a high probability of all tasks being represented. These results provide hope that simple universal training instances may exist for other problem domains.

6.3 Scalability

In the previous subsections, we demonstrate the utility and generalizability of evolved response thresholds for swarms of 50 agents solving a number of two-dimensional tracking problems that require agents to undertake four tasks. In this section, we report results that show the scalability of our approach with respect to the number of agents and the number of tasks.

The obvious impediment to evolving thresholds for larger swarms or an increased number of tasks is the increase in the size of the genome, which makes evolution more complex. To address this, we first experiment with swarms of sizes 100, 500, and 1,000. For 100 agents, we duplicate all runs performed for 50 agents.

Figure 16 demonstrates that the results for our approach with 100 agents are similar to those for 50 agents. As for Figure 9, the x-axis consists of six groups, one for each problem instance. Within each group, we show data points for uniform thresholds, dynamic thresholds, thresholds evolved for that target path, and thresholds evolved for `random` paths but tested on the target path, and shuffled evolved thresholds. In the group for `random`, we omit the results for rand-evo, since they duplicate random. The y-axis represents average positional difference. Each column represents 30 runs of the simulation for each of the 32 runs of the GA.

The plot shows that evolved thresholds outperform other types. Further, the distribution of thresholds within agents is as important for 100 agents as for 50, demonstrated by shuffled

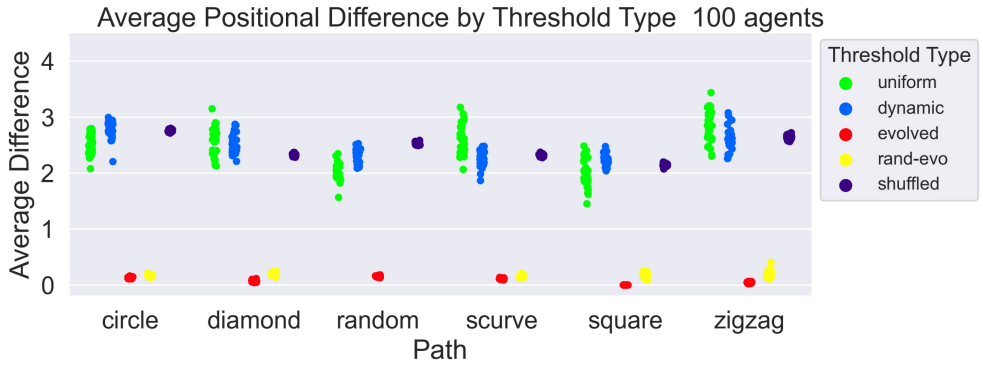

Fig. 16. Average positional difference for five threshold types for swarms of 100 agents.

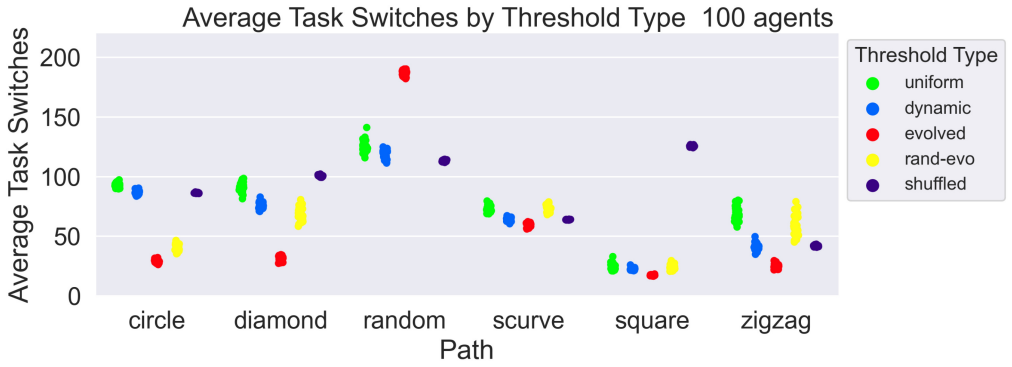

Fig. 17. Average task switches for five threshold types for swarms of 100 agents.

thresholds. The similarity in results for 50 and 100 agents is repeated for task switches, as seen in Figure 17.

The generalizability observed for 50 agents extends to larger swarms as well. Figure 18 demonstrates that thresholds evolved for `circle` and `random` paths generalize to all other paths, as they did for 50 agent swarms. The x-axis labels represent the path used for evolution. Each color represents a path used for testing. Each column represents 32 data points, one for each run of the genetic algorithm. Each data point is the average of 30 simulations using one set of evolved thresholds. The y-axis shows average positional difference.

For swarms of 500 and 1,000 agents, we perform GA runs only for the `circle` path as this allows demonstration of both effective evolution of response thresholds for larger swarms and of generalization. Figure 19 illustrates the results. Note that, in fact, generalization improves slightly as swarm size increases. We note two possible causes for this. First, in larger swarms, the effect of each agent is reduced, creating finer granularity in swarm behavior. Second, for swarm sizes 500 and 1,000, the GA population size is increased from 100 to 200 to facilitate evolution with a significantly larger genome.

Scalability manifests not only as the effectiveness of the thresholds evolved but also as the time required to generate those thresholds. Table 2 shows the average time required, as well as the effectiveness of the thresholds, averaged over all runs of our GA for `circle`, for each of the swarm sizes considered. The data show that the swarm size has little direct impact on runtime,

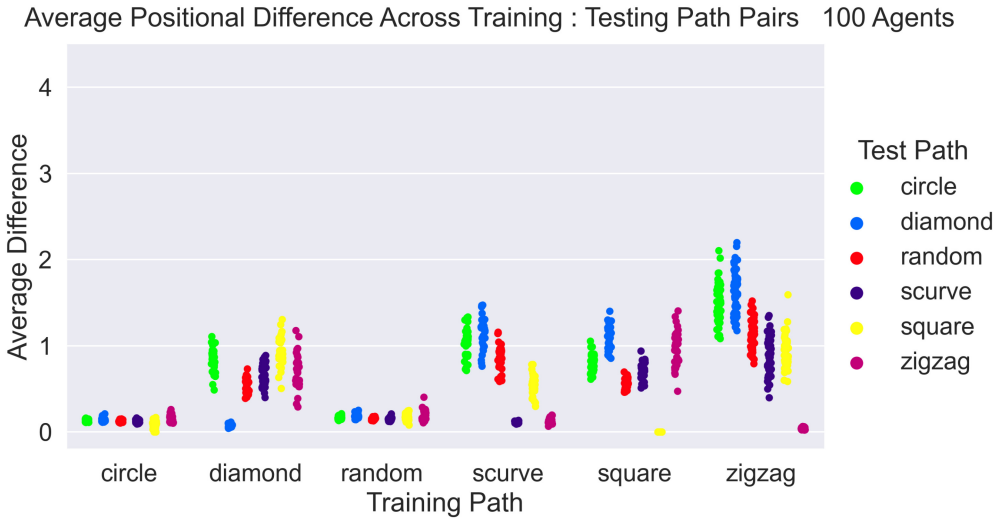

Fig. 18. Generalization of thresholds evolved for six paths for swarms with 100 agents. `circle` and `random` provide the best generalization.

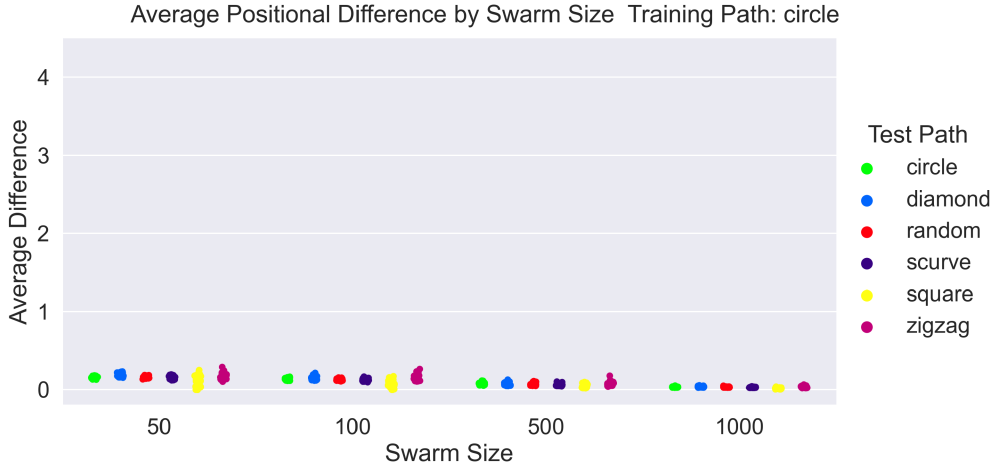

Fig. 19. Generalization of thresholds evolved for path `circle` for swarms of 50, 100, 500, and 1,000 agents.

however, the GA population size significantly increases runtime due to the quadratic dependence on population size for NSGA-II. This is relevant because for larger swarms, a larger GA population is necessary for effective evolution.

Another aspect of scalability we explore is the number of tasks. By adding a third dimension to our tracking simulator, we increase the number of tasks agents may undertake from four to six. To date, 3D experiments are performed for swarms of 50 agents. All other aspects of the simulator remain unchanged: All agents can perform all tasks, each agent has an independent threshold for each task, and we can substantially vary the task demands presented to the swarm by changing problem instances. The problem instances used in 3D are, of course, very different than those used in 2D and are described in detail in Section 3.

Table 2. Runtimes and Effectiveness of Evolved Thresholds for Swarm Sizes of 50, 100, 500, and 1,000 for the `Circle` Problem Instance

Number of agents	50	100	500	1000
GA population	100	100	200	200
GA runtime (minutes)	70.42	87.53	331.67	444.56
Avg. difference	0.1492	0.1185	0.0625	0.0310
Path length	1502.34	1501.79	1501.06	1500.05

Note that the GA population is larger for runs with 500 and 1,000 agents. This is the most significant factor in the increase in runtime.

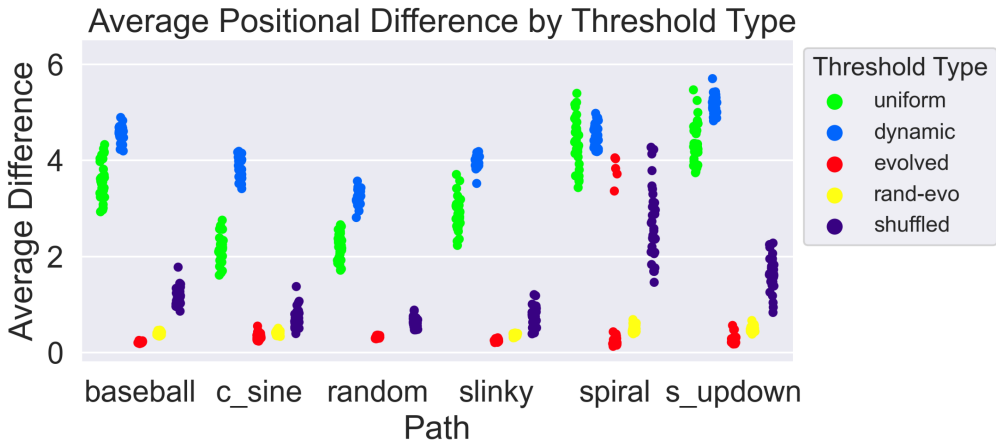

Fig. 20. Swarm tracking performance for five threshold types for swarms of 50 agents in three dimensions.

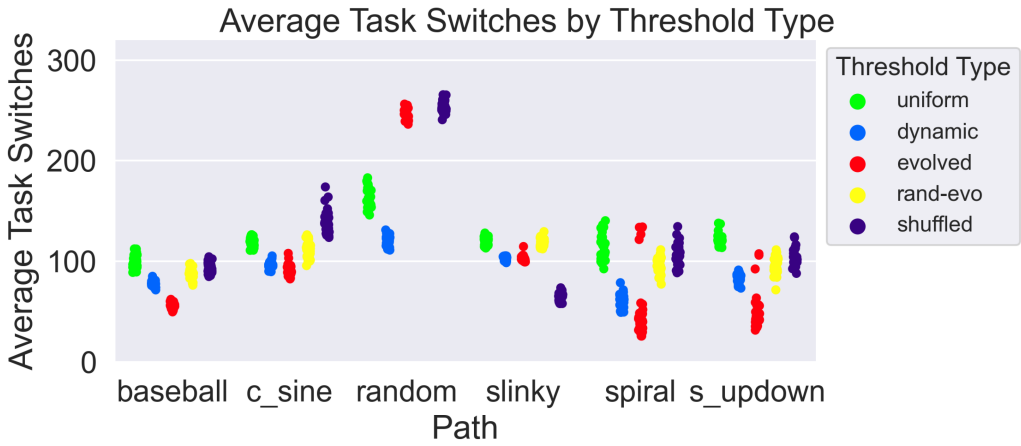

Fig. 21. Swarm specialization for five threshold types for swarms of 50 agents in three dimensions.

Figure 20 shows the results for different threshold types for 3D paths with respect to tracking accuracy. The trend is similar to that observed for 2D tracking. Figure 21 shows the same trend for task switches. These data demonstrate that generalization holds for at least modest increases in the number of tasks.

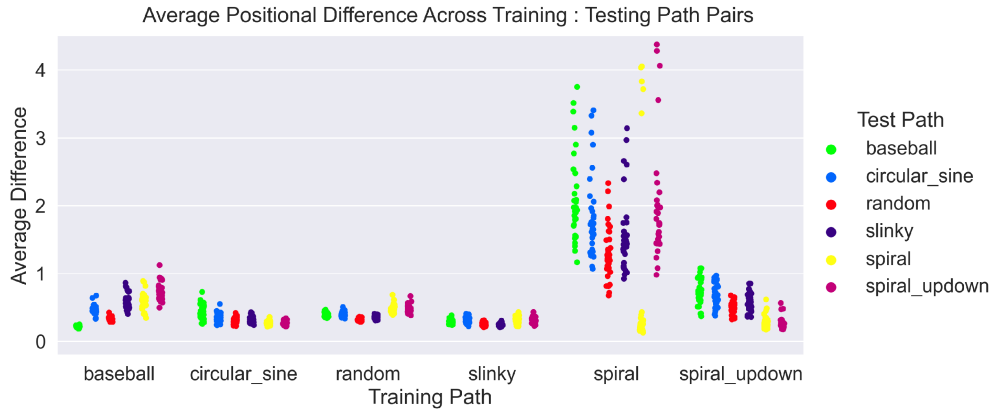

Fig. 22. Generalization of thresholds evolved for 3D paths for swarms of 50 agents.

Figure 22 demonstrates that generalizability extends to three dimensions. Paths `random` and `slinky` show the greatest propensity for generalizing with `baseball_stitching` also performing well. `spiral` does not generalize at all. These observations reinforce the discussion in Section 6.2. `spiral` generates no task demand for `push_DOWN`. Because all other paths create demand for that task, thresholds evolved for `spiral` cannot adequately address the demand for those paths. `spiral_updown` moves up for the first half of the simulation and down for the second half. While this creates demand for `push_DOWN`, somewhat improving generalization, the balance of demands created across the tasks is not highly varied. To test the claim that generalization of response thresholds is dependent on variation in the balance of task demands during evolution, we develop the `slinky` problem instance (see Figure 5). This path generates varying simultaneous demand for each possible combination of tasks. That `slinky` results in the highest degree of generalizability lends support to the claim that this property is relevant to universal training instances.

6.4 Threshold Distribution

Two factors explain the success of evolved thresholds: the distribution of threshold values for a task and the distribution of thresholds for each agent. The former is important for ensuring that an appropriate number of agents are capable of responding to various levels of demand for each task while the latter regulates whether an agent must select between multiple tasks for which their thresholds are satisfied.

The experiments performed with shuffled thresholds demonstrate the relevance of the distribution of thresholds across tasks for an agent. The shuffled data elements in Figures 9, 10, 16, 17, and 20 demonstrate how changing the relative values of task thresholds for an agent affect swarm performance. As previously described, these elements represent runs using evolved response thresholds that have been randomly permuted, maintaining the set of thresholds for each task but redistributing them among agents. This disrupts the relative values of thresholds for an agent across the tasks. Both domain problem performance and specialization are significantly decreased for shuffled thresholds.

The importance of threshold distribution for a task, across agents, is evident in Figure 8. The first and final histograms show the evolved change in distribution. The initial values, uniformly distributed at random, have 18 thresholds less than 0.35 and 8 less than 0.20. By generation 500, the evolved thresholds have 26 thresholds less than 0.35 and 19 less than 0.20. With many more

agents responding to lower demand, the tracker is able to maintain a much smaller distance to the target, by more than a factor of 10, as seen in the legend of each histogram.

7 CONCLUSION

In this article, we examine evolved response thresholds for a dynamic task allocation problem in a decentralized, deterministic threshold-based swarm. The testbed problem we use allows creation of highly dynamic task demands with significant variation in the balance of demands between tasks, frequency of change in demand, and magnitude of change in demand. Our primary finding is that response thresholds evolved for some problem instances generalize to all other problem instances despite significant differences in the schedules of task demands generated. `circle` in two-dimensions and `slinky` in three-dimensions result in evolved thresholds that generalize particularly well. Our experiments show that generalization derives from variation in the balance of simultaneous task demands during evolution. This result provides hope for an *evolve once, apply many* approach, negating the need to evolve thresholds for every problem instance.

Another significant finding is that it is possible to achieve nearly optimal performance in highly dynamic environments using static response thresholds. In this work, thresholds are evolved *a priori* and are static throughout use in simulations. Task demands generated in simulations are dynamic. Despite this, swarm performance is almost ideal, even when generalizing from `circle` or `slinky` in two or three dimensions, respectively. This refutes the common assumption that a swarm must be dynamic to perform in a dynamic environment.

Finally, this work demonstrates the importance of the distribution of response thresholds within each agent. Experiments show that disrupting those distributions degrades swarm performance by significantly reducing agent specialization, measured as the average number of times agents switch between tasks. A potentially interesting direction for future exploration is to repeat these experiments with different measures of specialization.

REFERENCES

Junier Caminha Amorim, Vander Alves, and Edison Pignaton de Freitas. 2020. Assessing a swarm-GAP based solution for the task allocation problem in dynamic scenarios. *Expert Syst. Appl.* 152 (2020), 113437.

Christos Ampatzis, Elio Tuci, Vito Trianni, and Marco Dorigo. 2008. Evolution of signaling in a multi-robot system: Categorization and communication. *Adaptive Behavior* 16, 1 (2008), 5–26.

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

Nathanael Aubert-Kato, Charles Fosseprez, Guillaume Gines, Ibuki Kawamata, Huy Dinh, Leo Cazenille, Andre Estevez-Tores, Masami Hagiya, Yannick Rondelez, and Nicolas Bredeche. 2017. Evolutionary optimization of self-assembly in a swarm of bio-micro-robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 59–66.

Gianluca Baldassarre, Stefano Nolfi, and Domenico Parisi. 2003. Evolving mobile robots able to display collective behaviors. *Artif. Life* 9, 3 (2003), 255–267.

Gianluca Baldassarre, Vito Trianni, Michael Bonani, Francesco Mondada, Marco Dorigo, and Stefano Nolfi. 2007. Self-organized coordinated motion in groups of physically connected robots. *IEEE Trans. Syst. Man Cybernet. B: Cybernet.* 37, 1 (2007), 224–239.

Benjamin E. Beckman and Philip K. McKinley. 2008. Evolution of adaptive population control in multi-agent systems. In *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*.

Gerardo Beni. 1992. Distributed robotic systems and swarm intelligence. *J. Robot. Soc. Jpn.* 10, 4 (1992), 31–37.

Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. 1996. Quantitiate study of the fixed threshold model for the regulation of division of labor in insect societies. *Proc. Roy. Soc. Lond.: Biol. Sci.* 263, 1376 (1996), 1565–1569.

Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. 1998. Fixed response thresholds and the regulation of division of labor in insect societies. *Bull. Math. Biol.* 60 (1998), 753–807.

Arne Brutschy, Nam-Luc Tran, Nadir Baiboun, Marco Frison, Giovanni Pini, Andrea Roli, Marco Dorigo, and Mauro Birattari. 2012. Costs and benefits of behavioral specialization. *Robot. Auton. Syst.* 60, 11 (2012), 1408–1420.

Adam Campbell, Cortney Riggs, and Annie S. Wu. 2011. On the impact of variation on self-organizing systems. In *Proceedings of the 5th IEEE International Conference Self-Adaptive and Self-Organizing Systems*.

Mike Campos, Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. 2000. Dynamic scheduling and division of labor in social insects. *Adapt. Behav.* 8, 2 (2000), 83–96.

Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Wenguo Liu, Alan F. T. Winfield, Yutaka Nakamura, and Hiroshi Ishiguro. 2016. Adaptive foraging for simulated and real robotic swarms: The dynamical response threshold approach. *Swarm Intell.* 10, 1 (2016), 1–31.

Eduardo Castello, Tomoyuki Yamamoto, Yutaka Nakamura, and Hiroshi Ishiguro. 2013. Task allocation for a robotic swarm based on an adaptive response threshold model. In *Proceedings of the 13th International Conference on Control, Automation, and Systems*. 259–266.

Vincent A. Cicirello and Stephen F. Smith. 2002. Distributed coordination of resources via wasp-like agents. In *Lecture Notes in Artificial Intelligence*, Vol. 2564. 71–80.

Nikolaus Correll. 2008. 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*. 3302–3307.

Javier de Lope, Dario Maravall, and Yadira Quinonez. 2013. Response threshold models and stochastic learning automata for self-coordination of heterogeneous multi-task distribution in multi-robot systems. *Robot. Auton. Syst.* 61, 7 (2013), 714–720.

Javier de Lope, Dario Maravall, and Yadira Quinonez. 2015. Self-organizing techniques to improve the decentralized multi-task distribution in multi-robot systems. *Neurocomputing* 163 (2015), 47–55.

Viviane M. de Oliveira and Paulo R. A. Campos. 2019. The emergence of division of labor in a structured response threshold model. *Physica A* 517, C (2019), 153–162.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 2 (2002), 182–197.

Marco Dorigo, Vito Trianni, Erol Sahin, Roderich Groß, Thomas H. Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, and Luca M. Gambardella. 2004. Evolving self-organizing behaviors for a swarm-bot. *Auton. Robots* 17 (2004), 223–245.

Miguel Duarte, Sancho Oliveira, and Anders Christensen. 2009. An ant based algorithm for task allocation in large-scale and dynamic multiagent scenarios. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 73–80.

Ana Duarte, Ido Pen, Laurent Keller, and Franz J. Weissing. 2012. Evolution of self-organized division of labor in a response threshold model. *Behav. Ecol. Sociobiol.* 66 (2012), 947–957.

Miguel Duarte, Vasco Costa, Jorge Gomes, Tiago Rodrigues, Fernando Silva, Sancho Moura Olivieira, and Anders Lyhne Christensen. 2016a. Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLoS One* 11, 3 (2016).

Miguel Duarte, Jorge Gomes, Vasco Costa, Sancho Moura Oliveira, and Anders Lyhne Christensen. 2016b. Hybrid control for a real swarm robotics system in an intruder detection task. In *Proceedings of the European Conference on the Applications of Evolutionary Computation*. 213–230.

Miguel Duarte, Sancho Oliveira, and Anders Christensen. 2014. Hybrid control for large swarms of aquatic drones. In *Proceedings of the 14th International Conference on the Synthesis and Simulation of Living Systems*. 785–792.

Eliseo Ferrante, Ali Emre Turgut, Edgar Dué nez Guzmán, Marco Dorigo, and Tom Wenseleers. 2015. Evolution of self-organized task specialization in robot swarms. *PLOS Comput. Biol.* 11, 8 (2015).

Paulo R. Ferreira, Felip S. Boffo, and Ana L. C. Bazzan. 2007. A swarm based approximated algorithm to the extended generalized assignment problem (E-GAP). In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. 1–3.

Dominik Fischer, Sanaz Mastaghim, and Larissa Albantakis. 2018. How swarm size during evolution imacts the behavior, generalizability, and brain connectivity of animats performing a spatial navigation task. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 77–84.

Stephanie Forrest and Melanie Mitchell. 1993. What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. *Mach. Learn.* 13 (1993), 285–319.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *J. Mach. Learn. Res.* 13 (July 2012), 2171–2175.

Jacques Gautrais, Guy Theraulaz, Jean-Louis Deneubourg, and Carl Anderson. 2002. Emergent polyethism as a consequence of increase colony size in insect societies. *J. Theor. Biol.* 215, 3 (2002), 363–373.

Harry Goldingay and Jort van Mourik. 2013. The effect of load on agent-based algorithms for distributed task allocation. *Inf. Sci.* 222 (2013), 66–80.

Heather J. Goldsby, Anna Dornhaus, Benjamin Kerr, and Charles Ofria. 2012. Task-switching costs promote the evolution of division of labor and shifts in individuality. *Proc. Natl. Acad. Sci. U.S.A.* 109, 34 (2012), 13686–13691.

Heather J. Goldsby, David B. Knoester, and Charles Ofria. 2010. Evolution of division of labor in genetically homogeneous groups. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'10)*.

Jorge Gomes and Anders L. Christensen. 2013. Generic behaviour similarity measures for evolutionary swarm robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*. 199–206.

Jorge Gomes, Paulo Urbano, and Anders Lyhne Christensen. 2013. Evolution of swarm robotics systems with novelty search. *Swarm Intell.* 7 (2013), 115–144.

Roderich Groß and Marco Dorigo. 2008. Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling. *Adapt. Behav.* 16, 5 (2008), 285–305.

Miao Guo, Bin Xie, Jie Chen, and Yipeng Wang. 2020. Multi-agent coalition formation by an efficient genetic algorithm with heuristic initialization and repair strategy. *Swarm Evol. Comput.* 55 (2020).

Emma Hart, Andreas S. W. Steyven, and Ben Paechter. 2018. Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 101–108.

Sabine Hauert, Jean-Christophe Zuffery, and Dario Floreano. 2009. Evolved swarming without positioning information: An application in aerial communication relay. *Auton. Robots* 26 (2009), 21–32.

C. T. Holbrook, T. H. Eriksson, R. P. Overson, J. Gadau, and J. H. Fewell. 2013. Colony-size effects on task organization in the harvester ant Pogonomymex californicus. *Insect. Soc.* 60, 2 (2013), 191–201.

C. Tate Holbrook, Phillip M. Barder, and Jennifer H. Fewell. 2011. Division of labor increases with colony size in the harvester ant pogonomyrmex californicus. *Behav. Ecol.* 22, 5 (2011), 960–966.

Chien-Lun Hunag and Geoff Nitschke. 2017. Evolving collective driving behaviors. In *Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems*. 1573–1574.

Robert L. Jeanne. 1986. The evolution of the organization of work in social insects. *Monit. Zool. Ital.* 20, 2 (1986), 119–133.

Raphaël Jeanson and Anja Weidenmüller. 2014. Interindividual variability in social insects—Proximate causes and ultimate consequences. *Biol. Rev.* 89, 3 (2014), 671–687.

Nidhi Kalra and Alcherio Martinoli. 2006. A comparative study of market-based and threshold-based task allocation. In *Distributed Autonomous Robotics Systems 7*. 91–101.

Anshul Kanakia, Behrouz Touri, and Nikolaus Correll. 2016. Modeling multi-robot task allocation with limited information as global game. *Swarm Intell.* 10 (2016), 147–160.

Vera A. Kazakova and Annie S. Wu. 2018. Specialization vs. re-specialization: Effects of hebbian learning in a dynamic environment. In *Proceedings of the 31st Florida Artificial Intelligence Research Society (FLAIRS'18)*. 354–359.

Vera A. Kazakova, Annie S. Wu, and Gita R. Sukthankar. 2020. Respecializing swarms by forgetting reinforced thresholds. *Swarm Intelligence* 14 (2020), 171–204.

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

Michael J. B. Krieger and Jean-Bernard Billeter. 2000a. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robot. Auton. Syst.* 30, 1–2 (2000), 65–84.

Michael J. B. Krieger, Jean-Bernard Billeter, and Laurent Keller. 2000b. Ant-like task allocation and recruitment in cooperative robots. *Nature* 406 (2000), 992–995.

Thomas H. Labella, Marco Dorigo, and Jean-Louis Deneubourg. 2006. Division of labor in a group of robotics inspired by ants' foraging behavior. *ACM Trans. Auton. Adapt. Syst.* 1, 1 (2006), 4–25.

Elizabeth A. Langridge, Nigel R. Franks, and Ana B. Sendova-Franks. 2004. Improvement in collective performance with experience in ants. *Behav. Ecol. Sociobiol.* 56 (2004), 523–529.

Wonki Lee and DaeEun Kim. 2019. Adaptive approach to regulate task distribution in swarm robotic systems. *Swarm Evol. Comput.* 44 (2019), 1108–1118.

Wonki Lee, Neil Vaughan, and DaeEun Kim. 2020. Task allocation into a foraging task with a series of subtasks in swarm robotic system. *IEEE Access* 8 (2020), 107549–107561.

Daniel Merkle and Martin Middendorf. 2004. Dynamic polyethism and competition for tasks in threshold reinforcement models of social insects. *Adapt. Behav.* 12, 3-4 (2004), 251–262.

Bernd Meyer, Anja Weidenmuller, Rui Chen, and Julian Garcia. 2015. Collective homeostasis and time-resolved models of self-organised task allocation. In *Proceedings of the 9th EAI Int'l Conf Bio-inspired Info & Comm Tech.* 469–478.

Ruby L. Moritz and Martin Middendorf. 2015. Evolutionary inheritance mechanisms for multi-criteria decision making in multi-agent systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 65–72.

R. Morley. 1996. Painting trucks at general motors: The effectiveness of a complexity-based approach. In *Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business*. 53–58.

Giuseppe Narzisi, Venkatesh Mysore, and Bud Mishra. 2006. Multi-objective Evolutionary Optimization of agent-based models: An application to emergency response planning. In *Proceedings of the 2nd International Conference on Computational Intelligence*. 224–230.

Aadesh Neupane and Michael A. Goodrich. 2019. Designing emergent swarm behaviors usign behavior trees and grammatical evolution. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2138–2140.

Aadesh Neupane, Michael A. Goodrich, and Eric G. Mercer. 2018. GEESE: Gramatical evolution algorithm for evolution of swarm behaviors. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 999–1006.

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

Geoff Nitschke. 2009. Neuro-evolution methods for gathering and collective construction. In *Proceedings of the 10th European Conference on Artificial Life*. 111–119.

G. S. Nitschke, A. E. Eiben, and M. C. Schut. 2012a. Evolving team behaviors with specialization. *Genet. Program. Evolv. Mach.* 13 (2012), 493–536.

G. S. Nitschke, M. C. Schut, and A. E. Eiben. 2012b. Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm Evol. Comput.* 2 (2012), 25–38.

Shervin Nouyan, Roberto Ghizzioli, Mauro Birattari, and Marco Dorigo. 2005. *An Insect-based Algorithm for the Dynamic Task Allocation Problem*. Technical Report. IRIDIA.

Liviu Panait, Sean Luke, and R. Paul Wiegand. 2006. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Trans. Evol. Comput.* 10, 6 (2006), 629–645.

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

Giovanni Pini and Elio Tuci. 2008. On the design of neuro-controllers for individual and social learning behaviour in autonomous robots: An evolutionary approach. *Connect. Sci.* 20, 2–3 (2008), 211–230.

Richard Price and Peter Tino. 2004. Evaluation of adaptive nature inspired task allocation against alternative decentralised multiagent strategies. In *Proceedings of the Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, Vol. 3242. 982–990.

Fabien Ravary, Emmanuel Lecoutey, Gwenael Kaminski, Nicolas Chaline, and Pierre Jaisson. 2007. Individual experience alone can generate lasting division of labor in ants. *Curr. Biol.* 17, 15 (2007), 1308–1312.

Cortney Riggs and Annie S. Wu. 2012. Variation as an element in multi-agent control for target tracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 834–841.

Dilini Samarasinghe, Erandi Lakshika, Michael Barlow, and Kathryn Kasmarik. 2018. Automatic synthesis of swarm behavioural rules from their atomic components. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 133–140.

Janaína Schwarzrock, Iulisloi Zacarias, Ana L. C. Bazzan, Ricardo Queiroz de Araujo Fernandez, Leonardo Henrique Moreira, and Edison Pignaton de Freitas. 2018. Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence. *Eng. Appl. Artif. Intell.* 72 (2018), 10–20.

Onur Soysal, Erkin Bahçeci, and Erol Şahin. 2007. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turk. J. Electr. Eng. Comput. Sci.* 15 (2007), 199–225.

Valerio Sperati, Vito Trianni, and Stefano Nolfi. 2008. Evolving coordinated group behaviours through maximisation of mean mutual information. *Swarm Intell.* 2 (2008), 73–95.

Valerio Sperati, Vito Trianni, and Stefano Nolfi. 2011. Self-organised path formation in a swarm of robots. *Swarm Intell.* 5 (2011), 97–119.

Andreas Steyven, Emma Hart, and Ben Paechter. 2017. An investigation of environmental influence on the benefits of adaptation mechanisms in evolutionary swarm robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 155–162.

Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. 1998. Response threshold reinforcement and division of labour in insect societies. *Proc. Roy. Soc. B* 265, 1393 (1998), 327–332.

Guy Theraulaz, Simon Goss, Jacques Gervet, and Jean-Louis Deneubourg. 1991. Task differentiation in polistes wasp colonies: A model for self-organizing groups of robots. In *Proceedings of the 1st International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. 346–355.

Vito Trianni, Roderich Groß, Thomas H. Labella, Erol Şahin, and Marco Dorigo. 2003. Evolving aggregation behaviors in a swarm of robots. In *Proceedings of the European Conference on Artificial Life*. 865–874.

Vito Trianni, Stefano Nolfi, and Marco Dorigo. 2006. Cooperative hole avoidance in a *swarm-bot*. *Robot. Auton. Syst.* 54, 2 (2006), 97–103.

Elio Tuci. 2014. Evolutionary swarm robotics: Genetic diversity, task allocation and task switching. In *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)*. 148–160.

Jane X. Wang, Edward Hughes, Christantha Fernando, Wojciech M. Czaarnecki, Edgar A. Duenez-Guzman, and Joel Z. Leibo. 2019. Evolving intrinsic motivations for altruistic behavior. In *Proceedings of the 18th International Conference Autonomous Agents and MultiAgent Systems*. 683–692.

Anja Weidenmüller. 2004. The control of nest climate in bumblebee (*Bombus terrestris*) colonies: Interindividual variability and self reinforcement in fanning response. *Behav. Ecol.* 15, 1 (2004), 120–128.

Anja Weidenmüller, Rui Chen, and Bernd Meyer. 2019. Reconsidering response threshold models – short-term response patterns in thermoregulating bumblebees. *Behav. Ecol. Sociobiol.* 73, Article 112 (2019).

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

Annie S. Wu, H. David Mathias, Joseph P. Giordano, and Anthony Hevia. 2020. 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*.

Annie S. Wu, H. David Mathias, Joseph P. Giordano, and Arjun Pherwani. 2021. *Collective Control as a Decentralized Task Allocation Testbed*. Technical Report CS-TR-21-01. University of Central Florida.

Annie S. Wu and Cortney Riggs. 2018. Inter-agent variation improves dynamic decentralized task allocation. In *Proceedings of the 31st International Florida Artificial Intelligence Research Society Conference*. 366–369.

Naoki Yamada and Chiaka Sakama. 2013. Evolution of self-interested agents: An experimental study. In *Proceedings of the 7th International Workshop on Multi-disciplinary Trends in AI*. 329–340.

Ling Yu, Zhiqi Shen, Chunyan Miao, and Victor Lesser. 2011. Genetic algorithm aided optimization of hierarchical multiagent system organization. In *Proceedings of the 10th International Conference Autonomous Agents and MultiAgent Systems*. 1169–1170.

Received June 2021; revised January 2022; accepted April 2022