Respecializing swarms by forgetting reinforced thresholds

Vera A. Kazakova, Annie S. Wu & Gita R. Sukthankar

Swarm Intelligence

ISSN 1935-3812

Swarm Intell
DOI 10.1007/s11721-020-00181-3

Springer

Swarm Intelligence
https://doi.org/10.1007/s11721-020-00181-3

Respecializing swarms by forgetting reinforced thresholds

Vera A. Kazakova[1] · **Annie S. Wu**[1] · **Gita R. Sukthankar**[1]

Received: 28 February 2019 / Accepted: 17 February 2020

Abstract
Response threshold reinforcement is a powerful model for decentralized task allocation and specialization in multiagent swarms. In dynamic environments, initial task assignments and specializations must be updated over time to meet changing system needs. The very nature of threshold reinforcement-based behavior can, however, hinder respecialization, limiting its usability in real-world applications. We propose a decentralized forgetting-based extension to response threshold reinforcement and show that it can improve the efficiency and stability of the resulting task assignments under changing system demands.

Keywords Threshold reinforcement · Emergent coordination · Decentralized division of labor · Specialization · Task allocation · Multiagent deployment

1 Introduction

Task allocation in real-world domains often requires decentralized, scalable, adaptable approaches. Decentralized systems are robust to failure of any one agent and scale better to scenarios of many agents than centralized alternatives (Almeida et al. 2004; van Lon and Holvoet 2017). Approaches that do not rely on inter-agent communication can improve scalability (Murciano et al. 1997) and robustness in environments where communication is unreliable or not feasible (Kanakia et al. 2016). Not requiring communication also allows for simpler agent hardware, allowing for scalability at a lower cost (Jones and Mataric 2003). Without communication, emergent cooperation often relies on specialization, which can improve decentralized task allocation by reducing interference and task switching, while increasing efficiency (Ono and Fukumoto 1996; Murciano et al. 1997; Li et al. 2002; Nitschke et al. 2008; Hsieh et al. 2009; Campbell

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11721-020-00181-3) contains supplementary material, which is available to authorized users.

✉ Vera A. Kazakova
kazakova@cs.ucf.edu

Annie S. Wu
aswu@cs.ucf.edu

Gita R. Sukthankar
gitars@eecs.ucf.edu

[1] University of Central Florida, Orlando, FL 32816-2362, USA

Published online: 05 March 2020

and Wu 2011; Agmon et al. 2011; Román et al. 2014; Ferrante et al. 2015). Self-organization coupled with adaptable specialization can increase the complexity of tasks a multiagent system (MAS) can handle, as task re-allocation may be needed to handle agent failures and changes in dynamic environments (Hsieh et al. 2008, 2009). In this work, we investigate how the adaptability of specialization affects task allocation within dynamic and decentralized MAS, using patrolling as an example. We employ a well-known communication-free task allocation through threshold adaptation method (Theraulaz et al. 1998) and propose a forgetting-based extension to improve its respecialization capabilities, as well as the efficiency and stability of the resulting task assignments. In our approach, agents assess how task stimuli are changing over time, in order to autonomously decide when to reset existing but outdated specializations and respecialize from scratch, resulting in improved task performance and task allocation efficiency and stability.

We focus on decentralized task allocation of simple robots in environments with multiple always-available tasks of changing demands. Consider that some tasks can be acted on by any agent, but only a subset of these actions would be beneficial, while extra work could be in fact become detrimental. Consider patrolling: 10 agents may be required to successfully patrol an area on each step, indefinitely, until the system's security needs change or until the agents become more urgently needed elsewhere within the system. Fewer patrolling agents would result in decreased security, while excess patrolling agents are wasteful and can lead to harmful interference. Additionally, patrolling efforts cannot be accumulated ahead of time: patrolling more now does not reduce patrolling needs later. Patrolling is also never completed, requiring ongoing agent action from a subset of the agents. Other examples include exploration, perishable resource gathering, diagnostics, maintenance, and production. We group such tasks under the term of *ongoing* tasks. Assuming homogeneous capabilities, any agent can independently choose to act on any one *ongoing* task at a time, or even no task, idling instead. Without a limit on how many agents can simultaneously take up an *ongoing* task and without communication, agents must make sensible choices based on the perceived system needs rather than based on task availability. Ongoing tasks without task supply limits are seldom discussed in the literature.

For our discussion, we employ the following working definitions:

Ongoing tasks: tasks available continuously, in unlimited supply but in limited demand, which can be acted on by any number of agents simultaneously (i.e., task availability is not limited by task demand nor by agents' actions).
Task demand: ongoing tasks are never finished, requiring continuous work instead; demand is the percentage of agents needed on a task per time step, but can be equivalently seen as explicit numbers of agents; percentages are used merely to quantify system needs with respect to resources. Task allocation is guided by actual amount of received versus needed activity on each task, so the goal is a specific number of agents self-allocating to each task in accordance with the demand, as opposed to merely a proportional allocation.
Task stimulus: a "concurrent benefit," contributed to and observable by the agents, such as putting out a fire, where heat intensity and area coverage stimulate agents to act (Kanakia et al. 2016); task stimuli can directly or indirectly represent current system needs, dependent on task demands.
Specialization: an agent's tendency to act on some task(s) more than on others; adaptive specializations are those that can adapt to changes in the environment, such as agent failures or variations in task demands. In this work, specializations do not correspond to

an increase in skill, thus only indicating a habit or preference formation, resulting from changes in agents' action probabilities over time.
Specialization resetting: forgetting learned behavior by resetting agents' learned or reinforced task thresholds back to random values.

Specialization is often considered fundamental to efficient functionality of complex adaptive systems (Nitschke et al. 2008), as it can improve a system's efficiency, adaptability, and scalability. A system's efficiency is increased when specialists collaborate on a common goal (Román et al. 2014). In distributed problem solving, allowing agents to specialize matches or outperforms homogeneous agent solutions (Li et al. 2002). In multi-task or multi-role environments, specialization helps synchronize agent behavior where tasks may be time-sensitive and switching from one to another can result in delays or even hinder overall functionality (e.g., herding Ono and Fukumoto 1996). While optimal controllers can be designed to suit known system needs, specialization allows for handling of uncertainty (Hsieh et al. 2009). Specialization tendencies can diversify a homogeneous workforce to suit environmental needs, leading to increased individual fitness and improved joint resource utilization (Murciano et al. 1997). Emergent cooperation through specialization scales better than more centralized approaches, by: (1) allowing for simpler software and cheaper hardware, as agents only need to handle a subset of all the behaviors needed within the system; (2) allowing for simplified agent coordination by limiting the agents' focus to a set of subtasks or subroles within the system (e.g., leader vs. follower), while (3) reducing interference resulting from agents attempting to take on the same tasks either physically or virtually (Ono and Fukumoto 1996; Campbell and Wu 2011). Specialization also improves problem scalability, as it allows for a natural mapping from larger and potentially intractable problems to tractable subproblems (e.g., agents tasked with patrolling a large area vs. patrolling a set of smaller subareas) (Agmon et al. 2011).

While specialization has been widely used for task allocation in environments where systems needs specified ahead of time, allowing the approaches to adapt to a variety of scenarios, the readaptation of specializations (i.e., respecialization) within dynamic environments is not well investigated. Efficient allocation of non-communicating swarm agents to *ongoing* tasks can be achieved through emergent specialization, resulting from agents reacting to a shared environment by following simple rules, such as response threshold reinforcement. Existing research suggests that specialization resulting from threshold reinforcement may improve system performance as compared to market-based solutions (Kittithreerapronchai and Anderson 2003). Additionally, threshold reinforcement can drive task allocation without explicitly modeling costs to non-specialization, with specialization simply emerging from a habit toward repeated activity (Theraulaz et al. 1998). Agents need to balance taking advantage of developed specializations with a flexibility to take on new specializations as system needs change. Maintaining this balance between specialization and responsiveness to change is closely related to the non-trivial balancing of exploitation and exploration in learning (Levinthal and March 1993). Adaptive solutions often depend on agents' diversity, which can be diminished after an initial adaptation to the environment, complicating or even precluding readaptation to new system conditions (Mavrovouniotis et al. 2017; Price and Tiňo 2004; Kazakova and Wu 2018). Forgetting-based approaches can be used to improve performance in decentralized domains, such as forgetting older observations in adversarial decision making (Villacorta et al. 2013), forgetting older training environments in case-based reasoning robotic navigation (Kira and Arkin 2004), or resetting agents' habit-based preferences for and against available tasks (Kazakova and Wu 2018).

If forgetting learned behavior can help agents adapt in dynamic environments, decentralized approaches require a decentralized way for agents to decide when forgetting is warranted. We hypothesize that observing changes in task stimuli over time may be sufficient for agents to individually decide when to reset their current specializations back to their prespecialized random values, without a need for central controllers, task limitations, or inter-agent communication. Although some domains may require global task performance/stimulus monitoring module or informant agents (surveillance cameras or drones), in many cases the state of the current task can be readily observable by the agents: growing volume of debris to cleanup, quickly depleting collected resources, long production queues, etc. Note also that the cost of monitoring environmental effects of agent action scales better than the cost of explicitly coordinating larger numbers of agents. Agents do not need to know how much work or how many agents a task requires, only focusing on whether the stimulus for that task is currently growing or decreasing. An increase in task's stimulus indicates that agents are currently under-working on that task, while a decrease in stimulus indicates over-working. Smaller stimulus changes are used to gradually redirect agent activity choices, but larger stimulus changes can serve as indications of environmental change that warrants a reset of specializations. Entomology research, which inspires much of the techniques for decentralized MAS, supports our hypothesis, as insect societies appear to use rate of stimulus change to achieve decentralized task allocation in dynamic natural environments (Westhus et al. 2013).

We propose a forgetting-based extension to a well-known insect-inspired task allocation model that is decentralized, communication free, and combines system needs and agents' threshold-reinforced specializations into task selection probabilities (Theraulaz et al. 1998). We introduce an automated specialization resetting heuristic, based on the number of system tasks and per-step stimulus increases. We first use a two-task setup to showcase how different amounts of change in task demands affect stimulus values. Then, we compare the performance achieved with the proposed automated resetting of specializations to performance without resetting, as well to a baseline performance when using a centralized signal to reset specializations when task demands change. All three approaches are tested on an abstract multi-area patrolling domain, with changing patrolling needs for each area. We then expand our testing to more agents, more tasks, and more demand changes to assess system scalability. Finally, we conduct a noise sensitivity test and propose an noise-based adjustment to the threshold resetting condition. Results demonstrate that agents are able to improve performance by automatically detecting when current task specializations have become outdated. Automated specialization resets decrease the number of unnecessary task switches while allowing faster readaptation to changes in the environment.

2 Related work

Without limits on task availability and without inter-agent communication, many existing task allocation approaches become unsuitable. Auctions and dominance contests for assigning a limited number of unit jobs require communication and limits on task availability: once an agent wins a task in an auction, others cannot win that same task, which reduces their choices (McIntire et al. 2016; Nunes and Gini 2015; Nunes et al. 2016; Zheng and Koenig 2011; Campos et al. 2000; Cicirello and Smith 2004; Ghizzioli et al. 2005; Nouyan 2002; Nouyan et al. 2005). Applying these approaches to *ongoing* tasks requires an artificial discretization of continuous system needs into individually assignable unit

jobs. Additionally, market-based approaches do not scale well with increasing numbers of agents and tasks (Golfarelli et al. 1997; Dias 2004). Token-passing approaches allow agents to take turns taking individual jobs off the list of options (Farinelli et al. 2006; Ma et al. 2017; Ferreira and Bazzan 2006; Ferreira et al. 2007, 2010; Schwarzrock et al. 2018; Tavares et al. 2017) and thus also require both communication and limitations on the available tasks. Task supply can also be limited indirectly, by restricting the number of spots available at the transition interface between sequentially dependent tasks, thus allowing agents to use their waiting times to estimate task needs (Frison et al. 2010; Pini et al. 2011; Brutschy et al. 2014). Another set of approaches employ inter-agent recruitment to a team or coalition responsible for some task (Dos Santos and Bazzan 2009, 2011, 2012; Ducatelle et al. 2009; Wawerla and Vaughan 2010), which also requires communication, role assignments, tracking the number of agents on a team, etc.

Task allocation for *ongoing* tasks can be viewed as proportionate agent deployment to multiple locations, applicable to multi-area surveillance, environmental monitoring, and aerial coverage for units on the ground (Hsieh et al. 2008). Under this view, locations/tasks can have explicit probabilities for agents to transition from one task to another (Berman et al. 2007; Halász et al. 2007). Performance of systems with task transition probability matrices is sensitive to initial conditions and to chosen transition probabilities. Additionally, transition probabilities are static, which poses a conflict between the speed of initial task allocation and the stability of the resulting task assignments: higher probabilities result in faster deployment, but also higher rates of task switching at equilibrium. Improvements to this sensitivity of transition probabilities are possible with the incorporation of a quorum technique, where agents become more likely to transfer out of a location/task when that location is close to the desired occupancy (i.e., close to the desired performance) (Hsieh et al. 2008). To respond to quorum conditions, agents do need an ability to assess occupancy or, equivalently, current task performance. Note, however, that approaches relying on transition probabilities do not distinguish among the transitioning agents and thus have no explicit specialization tendencies. The need for explicitly defined transition values also complicates adaptability in dynamic environments.

In lieu of direct communication, *stigmergic approaches* can be used, where agents coordinate through pheromone diffusion in their environment (Theraulaz and Bonabeau 1999). While pheromone techniques allow for decentralized task allocation (de Lope et al. 2012), they do require that agents are able to deposit and sense some form of environmental markers. Additionally, local markers do not inform agents about the demands of non-local tasks. Task allocation approaches that depend on temporary completion markers, such as patrolling of areas with the lowest concentration of pheromone (Chu et al. 2007), rely on breaking down the environment into discrete tasks with limited task availability, while also not promoting specialization. Furthermore, pheromone diffusion may not be sufficiently fast to allow for timely responses in dynamic environments (Garnier et al. 2007)

Decentralized task allocation without direct communication or explicit environmental markers can be achieved through *response threshold* techniques, i.e., approaches where agents respond to task stimuli surpassing some threshold value. Agents have preference values for and against each task, commonly referred to as *thresholds* (also known as sensitivity thresholds, affinity thresholds, habit thresholds, preference thresholds, specialization thresholds, etc.), which represent their current action-reinforced specialization, experience, suitability, or even circumstantial factors such as distance to task or battery levels. The lower the threshold, the more sensitive an agent is to that task's demand. As stimuli rise, agents with lowest thresholds begin acting first, slowing the increase in stimuli and, indirectly, preventing agents with higher thresholds from acting.

Each agent's thresholds can change over time in response to the agent's activity, which in turn is affected by these thresholds, creating a threshold reinforcement loop and leading to emergent specializations, beneficial when systems needs are unknown ahead of time (Murciano et al. 1997; Nitschke et al. 2008; Hsieh et al. 2009).

Threshold-based approaches have been used for both deterministic and probabilistic agent responses. Agents' task switches can be deterministically triggered when task stimuli surpass or fall below some threshold value; we refer to these as *trigger thresholds*. Examples include: identical (Agassounon et al. 2001) or different hard-coded trigger thresholds for all agents within a foraging domain (Krieger and Billeter 2000; Lee and Kim 2017); trigger thresholds calculated based on work availability, estimated individually, or through peer communication (Agassounon and Martinoli 2002); trigger thresholds adapted through environmental, internal, and social cues (Liu et al. 2007); and trigger thresholds communicated and updated following jamming avoidance response, approaching a uniform distribution over time to assign a proportionate number of workers to an ordered set of tasks (Lee and Kim 2016). Given the nature of these trigger thresholds, when a stimulus matches an agent's threshold, exceeds it by a small amount or exceeds it by a large amount, the agent responds with acting on that task with 100% probability. To achieve a more commensurate response, agents can instead use a probabilistic approach that combines their individual thresholds and the task stimuli (Theraulaz et al. 1998). Although this approach has been referred to by a variety of names, including "threshold reinforcement" and "response threshold," to avoid confusion with the various trigger threshold methods, we refer to this model as *StimHab*, to reflect its reliance on **stim**uli and action-reinforced task **hab**it thresholds. Given only the ability to sense tasks' current performance, agents decide how to act independently of each other, while also having the ability to specialize via threshold reinforcement, resulting in a highly scalable and efficient task allocation. Some decentralized probabilistic task allocation approaches rely on agents altering their action probabilities directly (instead of altering a threshold that affects these probabilities), in a variety of learning automata approaches with different probability-adapting formulas (Labella et al. 2006; Quiñonez et al. 2011; de Lope et al. 2012). The differences between direct and indirect probability adaptation approaches are not yet clear, although some comparison efforts have been made (de Lope et al. 2015).

Under StimHab, task stimuli alone can suffice to facilitate decentralized task allocation of simplistic agents, which is useful given that task stimuli are based on performance information, which is often already monitored in computational systems to assess operations. Additionally, while in the majority of existing task allocation approaches tasks are presented to agents one at a time, StimHab has been extended to allow agents to continuously consider multiple tasks simultaneously (Wu and Kazakova 2017). Note that although StimHab does not require communication, auctions, or token passing, it has been previously extended with these techniques for decentralized task allocation in domains with limitations on task supply: bee-inspired clustering for RoboCup Rescue (Dos Santos and Bazzan 2012); job queue and dominance contests based on delay prior to beginning a job for factory job assignment (Campos et al. 2000; Nouyan 2002; Cicirello and Smith 2004; Nouyan et al. 2005; Ghizzioli et al. 2005); identical, experimentally set, static thresholds for event-handling agents (Kalra and Martinoli 2006); pre-evolved stimuli and thresholds for a real-time strategy game (Tavares et al. 2014); and token passing for RoboCup Rescue (Ferreira et al. 2007, 2010) and for unmanned aerial vehicle surveillance (Schwarzrock et al. 2018). Communication-free StimHab has been applied to mail processing (Price and Tiňo 2004) and generalized task allocation with unlimited task supply (Kazakova and Wu 2018).

Threshold reinforcement does, however, struggle with respecialization, i.e., readaptation from an existing specialization to a new specialization when system conditions change (Price and Tiňo 2004; Kazakova and Wu 2018). Threshold reinforcement is guided by the principle of "keep preferring to do what you've been doing." Under StimHab, agents commonly begin with uniformly random thresholds, which are subsequently updated based on agents' individual action choices, developing a preference for one task and aversions toward the other tasks. The resulting specializations fit existing system needs, but when these needs change, agents may need to work on tasks to which they have grown averse. When beginning to respecialize, agents' thresholds correspond to pre-existing specializations. When these specializations are in opposition to new system needs, respecialization requires acting against existing conditioning, fighting against the very nature of threshold reinforcement, and resulting in slower task allocation, accompanied by an increase in task switching (Kazakova and Wu 2018). A forgetting-based solution can reset specialization capabilities, but has only been previously triggered for the agents in a centralized fashion (Kazakova and Wu 2018), which would be unsuitable for decentralized MAS.

3 Example domain: deployment for multi-area patrolling

To ground our discussion, we consider agent deployment for multi-area patrolling as an example of a decentralized task allocation for multiple *ongoing* tasks (i.e., deployment to the different patrollable areas), dynamic task demands (i.e., varying patrolling needs per area over time), no inter-agent communication, and an ability to benefit from specialization (staying in a subset of patrollable space can improve efficiency). Below we discuss why this view of patrolling is useful and how StimHab can be applied to this domain.

Decentralized selection among multiple patrollable areas with varying patrolling demands is an example of a dynamic domain with *ongoing* tasks. Existing patrolling approaches often rely on deterministic and centralized approaches, to the detriment of flexibility, scalability, and the costs of communication and load balancing (Portugal and Rocha 2011), while also relying on some form of task supply limits (e.g., Chu et al. 2007), which can restrict general applicability within real-world domains. Many-to-many communication does not scale well, but local communication may not be sufficient if the patrolled areas are disconnected, requiring some form of general information (task/area patrolling stimuli). Additionally, an agent choosing a particular area does not indicate that no more agents are needed in that area, so communicating patrolling choices may not be informative. Multi-area patrolling is a domain of multiple *ongoing* tasks, as all areas are always available to be patrolled. Thus, agents are not presented with one task at a time, but have to select from among all the choices every time they reconsider their current activity (potentially every time step). All agents can choose to patrol the same area or no area at all, but are ideally expected to distribute themselves across the areas in accordance with the patrolling demands. Each agent specializing on any one area can help minimize interference, while maximizing efficiency by reducing time-consuming area switching (Agmon et al. 2011). If patrolling demands change, agents need to redistributed themselves, i.e., a subset of the group needs to respecialize.

Emergent coordination is adaptable to dynamic patrolling needs (Almeida et al. 2004; Portugal and Rocha 2011). Under StimHab, current task performance can be inferred from each area's patrolling stimulus, allowing agents to self-deploy to the different areas without additional assumptions. Without exchanging or modeling each other's patrolling choices,

agents only consider whether each area is adequately patrolled. We measure patrolling performance as the ratio of [number of patrolling agents in area] to [desired number of patrolling agents in area]. The number of patrolling agents in each area can be tracked at the entry and exit points to each area. The desired number of patrolling agents can be set based on number or importance of targets within each area, importance of events within each area, time dependencies, etc. Variations in these area specifics would then lead to dynamically changing demands and thus require dynamically changing patrolling deployments.

To clearly assess the quality of decentralized deployment to different patrollable areas based on stimulus information alone, we use a high-level representation of the multi-area patrolling domain. Patrolling domains can have many specific elements and costs: number of patrollable nodes within each area, distance between nodes and between areas, agent speed, etc. To eliminate any domain-specific dependence on area adjacencies and travel-time costs, we consider patrollable areas as an abstract set of *ongoing* tasks. At every time step, each agent chooses whether to patrol any one of the areas or to idle, if it perceives that its services are not currently needed. In the presence of excess patrolling agents, idling can reduce interference, save expendable resources, and reduce wear and tear on the agents.

4 StimHab threshold reinforcement model

We employ a commonly used decentralized and communication-free threshold reinforcement model (Theraulaz et al. 1998), which we refer to as StimHab. Decentralized task allocation is achieved through probabilistic action, based on a combination of global task **stim**uli and agents' individual **hab**it thresholds for these tasks. StimHab increases the agents' probability to act on tasks with (1) higher task stimuli and (2) lower agent-specific task thresholds, thus addressing system needs while also promoting specialization (Theraulaz et al. 1998). Below we review how stimuli and habit thresholds are defined, updated, and used to calculate action probabilities, as well as how the resulting actions lead to specialization.

4.1 Global task stimuli

Agents sense system needs through task stimuli, which can be provided explicitly as globally available values or implicitly as observable environmental qualities (e.g., the dimensions of a fire Kanakia et al. 2016). An increase in a task's stimulus indicates an increase in the task's demand or a decrease in the task's fulfillment (e.g., number of targets in a given area has increased or some of the patrolling agents have moved away or have become inoperative). As task stimulus rises, more agents switch to that task, abandoning others; as the stimulus lowers, some agents switch to other tasks.

In StimHab, a task's stimulus does not directly indicate current task needs or the ratio of work that must be done on this task versus other tasks. Consider that knowing the number or ratio of agents currently needed for an *ongoing* task per some time period does not give an agent any indication of whether it should take on that task. Knowing how well the task is being handled, i.e., task performance, is a more informative heuristic for whether an agent's services are needed. Nevertheless, using performance values directly as stimuli results in an environment that is too unstable for specialization: ideal performance would correspond to zero stimulus, while no activity would correspond to maximum stimulus. Thus, reaching correct allocation would sharply drop activity and performance, increasing

stimulus, which would increase activity back to previous levels, as habits have not had time to change significantly, leading to zero stimulus again, and so on. This oscillating stimulus behavior causes agents to behave erratically, unable to fine-tune their specializations. This behavior was confirmed during initial testing, though the tests are omitted here for brevity.

In StimHab, task performance is the change in task stimulus, with ideal amount of work leading to no change, excess work leading to stimulus decrease, and insufficient work leading to stimulus increase. There are multiple ways of regarding stimulus changes: as the difference between system expenditure and work-based replenishment, as a change in the amount of a stored resource, or as a reflection of the ratio of performed versus ideal amount of work on a task per unit of time. Task t's stimulus s_t is updated as a change in the previous stimulus, with the assumption that the absence of work on a task with nonzero demand must lead to some stimulus increase. This per-step increase in stimulus represents either a drop in the level of some stored resource (e.g., materials being expended faster than collected) or a signal that the amount of work done was lower than desired for that task (e.g., fewer than ideal number of agents patrolling a given area).

$$\begin{aligned} s'_t &= s_t + (\Delta s_t \text{ given no work}) * \ (1 - (\text{step performance})) \\ &= s_t + \left(\frac{1}{\text{steps in cycle}}\right) \ * \ \left(1 - \frac{\text{step work done}}{\text{step work needed}}\right) \end{aligned}$$

up to a $\text{Min}(s_t) = 0.0$ and $\text{Max}(s_t) = 1.0$. If agents perform the exact amount of work that is needed on a task, s_t remains unchanged. We set Δs_t to increase the task's stimulus from minimum 0.0 to maximum 1.0 within a single *cycle*, defined as some number of consecutive decision *steps*. After any one step, if agents do no work on task t, s_t will increase by $1.0/(\text{steps in cycle})$, i.e., given a cycle of 100 steps, $s'_t = s_t + 0.01$.

4.2 Individual task habit thresholds

Each agent maintains a habit threshold for every task. These have sometimes been referred to as "response thresholds," which can be misleading in the case of StimHab, as agents do not respond based on these thresholds alone. When agent a acts on a task t, its threshold $\theta_{a,t}$ is reduced, while the thresholds for the other tasks are increased, over time leading to specialization. Lower thresholds increase action probability (see Sect. 4.3), causing agents to become more likely to act on the same task in the future.

When agent a chooses to act on task t, it specializes toward task t ($\theta_{a,t} \to 0.0$) and against all the other tasks ($\theta_{a,t' \neq t} \to 1.0$) according to the following threshold reinforcement rules:

$$\begin{aligned} \theta_{a,t} &= \theta_{a,t} - \xi \qquad \text{(where } \xi \text{ is the affinity-building rate)} \\ \theta_{a,t' \neq t} &= \theta_{a,t'} + \phi \quad \text{(where } \phi \text{ is the aversion-building rate),} \end{aligned}$$

with θ restricted to the range [0.0, 1.0]. Affinity and aversion rates indicate how fast agents specialize for and against tasks, based on their actions. We set affinity-building rate to $\xi = 1/(\text{steps in cycle})$, ensuring that thresholds can evolve from minimum (indicating highest affinity) to maximum (indicating highest aversion) in a single cycle. Aversion-building rate is set to $\phi = \xi/(\text{number of tasks} - 1)$. In the original setup, ϕ is set to be one-tenth of ξ for a domain with two tasks (Theraulaz et al. 1998). Instead, we assume that the total habit gained by one task should be lost elsewhere, consistent with potentially limited memory in simplistic swarm agents. Consequently, we split the habit loss (i.e., aversion-building) equally among the other tasks, excluding idling as doing nothing takes up no memory.

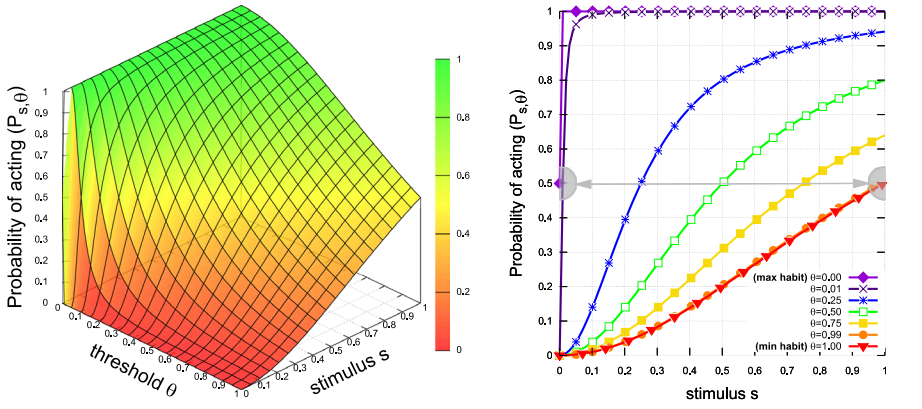
4.3 Action selection

StimHab achieves decentralized task allocation through probabilistic action based on a combination of global task stimuli and agents' individual affinities for these tasks (Theraulaz et al. 1998). Probability to switch to a task is directly proportional to the task's stimulus and inversely proportional to an agent's task threshold. A higher task stimulus indicates a higher need for that task within the system, while a lower agent's task threshold indicates that agent has developed an affinity for this task. When a task's stimulus rises, more agents abandon other tasks in favor of the one with rising stimulus, beginning with agents that have the lowest thresholds for that task and followed by those with higher thresholds. The result is a gradual response that is commensurate with system demands, similar to that achieved by fuzzy control systems. Below we review StimHab's probabilistic response in the presence of multiple tasks: (1) how probabilities are compared and (2) how action is chosen.

Every time step (or every domain-specific decision step), each agent a calculates the probability to act on every task t, by combining a globally known current task stimulus s_t with its own affinity for that task $\theta_{a,t}$ as follows:

$$P_{a,t} = s_t^2/(s_t^2 + \theta_{a,t}^2) \quad \text{where } s \in [0.0, 1.0], \theta \in [0.0, 1.0]$$
$$P_{a,t} = 0.5 \quad \text{where undefined } (s_t = \theta_{a,t} = 0.0)$$

For reference, the space of all combinations of s_t and $\theta_{a,t}$ is provided in Fig. 1a. Note the formula redefinition above to avoid division by zero at $s_t = \theta_{a,t} = 0$, corresponding to the vertical edge to the left of 3D probability map in Fig. 1. The resulting probability is a 50/50 chance to select the task or not, which is precisely between the adjacent values of $P_{a,t} = 1.0$ for $(s_t > 0.0, \theta_{a,t} = 0.0)$ and $P_{a,t} = 0.0$ for $(s_t = 0.0, \theta_{a,t} > 0.0)$ (upper and lower edges of the map, respectively), while also matching the other values along the rest of the "$s_t = \theta_{a,t}$" line (depicted in yellow and dividing the map horizontally in half): when $s_t = \theta_{a,t} > 0.0$,



(a) All values of $P_{a,t} = \frac{s_t^2}{s_t^2 + \theta_{a,t}^2}$ **(b)** $P_{a,t} = \frac{s_t^2}{s_t^2 + \theta_{a,t}^2}$ for different specializations $\theta_{a,t}$

Fig. 1 StimHab probabilities. Notice that $P_{s=1,\theta=1} = P_{s=0,\theta=0} = 0.5$ (left-most edge/point vs. right-most point in **a** or, equivalently, gray arrow-pointed points in **b**), causing fully specialized agents (those with a max habit for one task and min habit for all the others) to have difficulty readapting. Consequently, adaptability can be improved by resetting agents' habit thresholds θ_a, t to random values as demands change (Kazakova and Wu 2018)

$P_{a,t} = (s_t^2/(s_t^2 + s_t^2) = (s_t^2)/(2s_t^2) = 0.5$. In Fig. 1b, we provide an alternative view of the agents' action space, but focusing instead on the responses that different specialization thresholds produce given various stimulus values. In this view, we can also see why StimHab agents struggle to respecialize. Consider the gray arrow in the center of the graph, which points to a $P_a, t = 0.5$ for both the "max habit" specialization $\theta = 0$ at minimal stimulus $s_t = 0$ and the "min habit" specialization $\theta = 1$ at maximum stimulus $s_t = 1$. As these endpoints lead to identical probability to choose the task, there is no way for the agents to favor tasks with higher need for which they have developed maximally high thresholds through repeated action-based reinforcement (Kazakova and Wu 2018).

Under standard StimHab, only a single task is presented to the agents at a time (Theraulaz et al. 1998), but given an environment of multiple *ongoing* tasks, agents require a task selection mechanism. At every decision step, each agent a calculates the StimHab-based probability $P_{a,t}$ to act on every task t, sorts tasks in descending probability order, and begins considering acting on each task starting from the most probable. Existing research shows that this ordering is conducive to improved specialization characterized by reduced number of task switches (Wu and Kazakova 2017). Note that agents are likely to end up with different task orderings, as probabilities are calculated by each agent using agent-specific thresholds. When considering each task, if a randomly generated value is below the calculated task probability, the agent will act on that task. Otherwise, the task with the next highest $P_{a,t}$ is considered. Once a task is selected to be acted on, tasks further down the list are not considered. If no task is chosen after all tasks have been considered, the agent will idle until next decision step.

5 Triggering automated threshold resetting

In this section we discuss a level of minimal system change that may warrant agent respecialization. To benefit from StimHab's fast decentralized specialization, as well as from faster respecialization shown with centrally triggered threshold resets (Kazakova and Wu 2018), we allow agents to individually reset their habit thresholds back to random values based on a new assessment of stimulus change. Changes under the level of interest indicate normal probabilistic action and specialization development, while changes above this level indicate additional environmental changes that may warrant respecialization. First, we discuss what changes in stimuli mean for system stability and how they can be used to detect when existing specializations are no longer producing desired results. We then establish a minimum level of change, exceeding which will trigger agents to reset their $\theta_{a,t}$ back to uniformly random values in order to respecialize more effectively.

5.1 Monitoring stimulus changes to understand environmental changes

StimHab reliably leads to adequate specialization when starting with randomly assigned habit thresholds, but struggles to respecialize effectively if thresholds correspond to previous specializations when demands change (Kazakova and Wu 2018). As resetting habit thresholds effectively turns respecialization into specialization, agents can respecialize faster if they can recognize that a reset is needed. We hypothesize that observing changes in stimuli can let agents recognize that their existing specializations have become outdated, warranting a reset.

Below we discuss the meaning behind stable versus unstable task stimuli and consider how this information can be leveraged by decentralized agents for automated threshold resetting.

Although agents only have task stimuli available to them (not the actual task demands, number of agents, agent assignments, etc.), observing how these stimuli change over time can be sufficient to assess overall system stability. As agents' specializations develop over time, their assignments to tasks become consistent, allowing for *stabilization* in the changes of tasks' stimuli: if agents are able to fulfill a task's demands exactly, without under- or over-working, per-step expenditure will be offset, keeping the stimulus stable; if agents consistently under-work, stimulus will rise steadily; if agents over-work, stimulus will drop steadily. A sudden increase in the rates of stimuli changes, i.e., *destabilization*, can thus be used as an indication of a change within the environment. As existing specializations evolved to fit a previous environmental condition, respecialization may help agents adapt to the new conditions. Threshold resetting is only beneficial when conditions change after specializations have been fully developed (i.e., reached max and min threshold values due to repeated threshold reinforcement), as that is when StimHab struggles to respecialize. Thus auto-detection of changes is only needed when the system destabilizes; changes in an unstable environment do not need to be detected, as agents that have not yet fully specialized are capable of respecializing without resetting.

Perceiving only the tasks' stimuli, agents can recognize system destabilization by observing how these stimuli change over time. Stimuli changes are the velocity of the stimuli, and the changes of these changes represent stimuli accelerations. Decelerating stimuli represent stabilization: stimuli values approach a constant rate of change (or no change) as agents' task assignments become more stable with specialization. Accelerating stimuli correspond to destabilization: stimuli changes become less constant (or spike up or down) as agents' task assignments no longer lead to the same stimulus updates as before.

Agents can recognize destabilization by detecting acceleration in the tasks' stimuli, i.e., an increase in the change of the changes in stimuli. Within an environment of multiple *ongoing* tasks, we use the highest change of stimulus among all of the tasks as our velocity for a current step, calculated as follows.

$$\text{MAX}|\Delta s_t| = \text{MAX}\{|\forall t, \text{ current } s_t - \text{ prev. } s_t|\},$$

where prev.s_t is the stimulus for task t at the start of the step and new s_t is the value at the end of the step, after all actions take place. We use absolute value of the change to establish whether stimuli are changing more rapidly than before; whether stimuli are actually rising or falling does not represent system stability. We then compare this highest velocity to the highest velocity calculated identically over the previous step. The difference between the two velocities is the acceleration of the highest stimulus change:

$$\Delta\text{MAX}|\Delta s_t| = \text{current MAX}|\Delta s_t| - \text{previous MAX}|\Delta s_t|,$$

indicating whether stimuli just grew faster than before (positive value) or slower than before (negative value), corresponding to destabilization and stabilization, respectively. Thus, in order to assess whether the system destabilized, agents need to calculate and store $\text{MAX}|\Delta s_t|$ for the last two steps, i.e., the newest max change and the previous max change among the tasks' stimuli.

$$\text{If current MAX}|\Delta s_t| > \text{prev. MAX}|\Delta s_t| \rightarrow \Delta\,\text{MAX}|\Delta s_t| > 0, \text{ faster change: destabilization;}$$
$$\text{If current MAX}|\Delta s_t| < \text{prev. MAX}|\Delta s_t| \rightarrow \Delta\,\text{MAX}|\Delta s_t| < 0, \text{ slower change: stabilization.}$$

These comparisons involve only stimuli values s_t, already available to agents within StimHab.

5.2 Analyzing stimulus destabilization

Within an MAS based on probabilistic actions, some fluctuation is to be expected. Resetting specialization thresholds can be detrimental if specializations are not outdated, so agents need to distinguish between natural stimulus fluctuations versus destabilization requiring respecialization. To establish when the system has changed sufficiently to warrant resetting, agents need to establish what $\Delta\text{MAX}|\Delta s_t|$ are sufficiently high as to indicate an environmental change has occurred. We hypothesize that the smallest possible change in task demands is a reasonable heuristic value for the distinction between fluctuation versus destabilization. We investigate how the number of steps in cycle and number of tasks within the system affect the smallest possible change in stimuli. We then establish a formula for the minimum positive $\Delta\text{MAX}|\Delta s_t|$ that represents sufficient destabilization to trigger agents to reset their thresholds and respecialize.

The smallest possible change in stimuli over a single decision step is a function of total steps in a system's *cycle*. Recall that under StimHab, in the absence of agent action, stimuli increase over time based on some resource expenditure or rate of growing need for a task that is not being attended to. Assuming stimuli are confined to the [0.0, 1.0] range, stimuli of unattended tasks increase from minimum ($s_t = 0.0$) to maximum ($s_t = 1.0$) values over a single cycle defined as some n number of steps (e.g., a simulation day defined as some number of hours), increasing by $(1/\text{steps in cycle})$ every time step (e.g., every simulation hour). Note that even in setups without explicit "cycles," there must be some amount of consumption that decreases s_t on every step, maximizing the stimulus over some number of steps. Agents can only do one task per step. The smallest change in task demand is then $\pm$ a single work step out of the total available within a cycle, i.e., $(1/\text{steps in cycle})$.

The smallest possible change in stimuli is also a function of the total number of tasks within the system. If all agents are needed to fulfill tasks demands, the smallest $\text{MAX}|\Delta s_t|$ is observed when all tasks require the same number of steps and then one step shifts from one task to another. Task demands affect the expenditures incurred on each time step, which corresponds to total expenditure for the cycle divided by number of steps in a cycle. With a setup of two tasks, this minimal change corresponds to an initial demand set of (T1-50%, T2-50%), meaning that 50% of the cycle steps need to be spent on Task 1 and the other 50% on Task 2, shifting to a new demand set of either (T1-51%, T2-49%) or (T1-49%, T2-51%). This demand change is of 1% out of 50%, where 50% is the maximum demand amount when total steps are distributed equally between two tasks. With a setup of five tasks, minimum change would happen going from the demand set (T1-20%, T2-20%, T3-20%, T4-20%, T5-20%) to, for example, (T1-19%, T2-20%, T3-20%, T4-20%, T5-21%). This change is of 1 out of 20 steps, where 20 cycle steps are the maximum demand amount when the steps are distributed equally among 5 tasks. Thus, we see that the same demand change of a single step corresponds to a ratio of 1/((steps in cycle)/2) for two tasks and 1/((steps in cycle)/5) for five tasks. Consequently, to ensure that this change ratio can represent per-step changes consistently, we incorporate the number of tasks into the value delimiting the lowest amount of change to be regarded as destabilization:

$$\frac{1/\text{steps in cycle}}{\text{number of tasks}} = \frac{\text{number of tasks}}{\text{steps in cycle}}$$

Thus, we propose that, given some number of steps per cycle and the number of tasks, agents automatically reset their specialization thresholds $\theta_{a,t}$ to uniformly random values when:

$$\Delta\text{MAX}|\Delta s_t| \geq \frac{\text{number of tasks}}{\text{steps in cycle}}$$

6 Experiments

We conduct a series of experiments to investigate whether tracking changes in $\text{MAX}|\Delta s_t|$ can be sufficient to improve respecialization of decentralized agents employing threshold reinforcement. We first describe our testing setup and the metrics used. Then, we test how changes in task demands are manifested in $\text{MAX}|\Delta s_t|$ values under a series of progressively smaller task demand changes, as well as under a set of random-valued demand changes. Finally, we compare the behavior of the proposed Auto-Reset approach to that of standard StimHab, referred to here as No-Reset for clarity of distinction, as well as to the approach of Central-Reset of $\theta_{a,t}$ thresholds to uniformly random values each time demands are reset, which has been shown to outperform No-Reset (Kazakova and Wu 2018). We test how 100 agents dynamically self-allocate under the three approaches given maximal, minimal, and random changes in the demands of five tasks. Finally, we conduct a test with more random changes in demands, 1000 agents, and 10 tasks. The provided results show that StimHab with Auto-Reset is able to combine the benefits of No-Reset and Central-Reset approaches, ultimately leading to efficient and stable task allocation under all of the tested variations in demands and without a need for a central signal for agents to reset their specializations.

6.1 Baselines and metrics

To assess whether tracking $\Delta\text{MAX}|\Delta s_t|$ can improve decentralized threshold reinforcement-based respecialization, below we define the three compared approaches, describe our experimental settings, and define the performance metrics used.

We test three approaches on decentralized multiagent deployment to multiple patrollable areas seen as a set of *ongoing* tasks with dynamic demands. Agents will reinforce their habits, over time specializing on a single patrollable area or task. When demands change, agents will attempt to respecialize accordingly. The tested approaches are:

Auto-Reset: agents monitor changes in $\text{MAX}|\Delta s_t|$ and reset their specialization thresholds $\theta_{a,t}$ when the detected change is above the minimal change value equal to $\frac{\text{number of tasks}}{\text{steps in cycle}}$.

No-Reset: standard StimHab (Theraulaz et al. 1998), where all threshold adjustments are due solely to threshold reinforcement rules and thus there is No-Reset under this approach.

Central-Reset: when demands change, agents receive a central signal to reset their specialization thresholds $\theta_{a,t}$. This is our theoretical optimum, as the goal of automated resetting is to eliminate the need for a central signal, while still achieving comparable respecialization, give that Central-Reset outperforms standard No-Reset StimHab (Kazakova and Wu 2018).

The following basic system settings are used for all three approaches. The underlying system is always StimHab, with the only differences being whether and when system-wide habit threshold resets take place. During each simulation, 100 agents work on 5 *ongoing* tasks (T1-T5) over the course of 500 cycles or 50,000 steps. During each step, each agent can choose to work on a single task or to idle. Given 100 steps per cycle and 5 tasks, when choosing to act on a task, the agent's threshold for this task decreases by $\xi = 1/100$, while the thresholds for the other 4 tasks' increase by $\phi = \xi/4 = 1/400$ (as discussed in Sect. 4.2).

Task demands change every 50 cycles = 5000 steps, for a total of 10 times over a simulation, always adding up to 100% of the population, requiring the correct allocation of every agent to fully and continuously satisfy all *ongoing* tasks' to 100% performance. Task demands are defined as percentage of the total population. No excess work can be saved from one step to the next (excess current patrolling does not offset deficiencies in future patrolling). To assess respecialization behavior of each approach, we define the following terms:

Demand Period: a time period during which task demands remain stable; when demands change, a new demand period begins and will end when demands change again. Thus, observing average behavior across demand periods showcases the agents' average ability to adapt to changes in task demands.
Task Performance (per step or per cycle): ongoing tasks are never finished; thus, performance represents the proportion of work done as compared to work needed on a task per unit of time, such as per time step (a period of one action from each agent) or per cycle (a period of multiple consecutive agent actions).
Deviation from 100% Task Performance: the percentage of work misallocation with respect to a task, either above or below the ideal task requirement; as task performance can exceed or fall below 100%, averaging deviations prevents positive and negative misallocations from canceling each other (e.g., average task performance of two steps of 95% and 105% performance is 100%, but the average deviation is 5%, more meaningfully showcasing the task allocation behavior).
Task Switch: a change in agent activity, switching from acting on one task to acting on another (including switching to or from idling).
Task-Switching Rate: for any time period, the percentage of all actions that involved a task switch (e.g., given 100 steps per cycle and 100 agents, 10,000 actions take place in a cycle; a 20% task-switching rate indicates 2000 task switches for that cycle)
Ideal Task Allocation: a task allocation resulting in continuous task performances of 100% or, equivalently, average task performance deviations of 0%, with the number of average task switches approaching zero over time.
Adaptation Period: a period between demand changes, during which agents must respecialize in order to fulfill new system needs. Looking at average adaptation period behavior we can assess the expected task allocation behavior in dynamic environments.

Performance is assessed using: (1) task performance percentages over time, (2) average task deviations calculated for each cycle of a 50-cycle adaptation period, and (3) average task switches calculated for each cycle of a 50-cycle adaptation period. Observing task performance percentages over all 10 demand periods of a simulation showcases the actual behavior of each approach in a dynamic environment. In a dynamic environment, ideally satisfied tasks are those that quickly reach and maintain 100% *task performance* after task demands change, while task performances below and above 100% indicate insufficient

and excess agents on the tasks, respectively. Observing average task performance deviations and average task-switching rates showcases how each approach is able to respond to changes in task demands. Average deviations show the agents' ability to fulfill newly updated task demands, with deviations approaching 0% over time indicating near ideal task fulfillment resulting from an appropriate task allocation. Average task-switching rates show the stability of the task assignments, with values approaching 0% over time, indicating agents becoming highly specialized, continuously working on a single task.

6.2 How changes in demands affect stimuli

To ground our discussion of $\Delta\text{MAX}|\Delta s_t|$, we conduct a series of preliminary experiments to observe how $\text{MAX}|\Delta s_t|$ changes over time under different demand changes for two tasks: (1) maximal changes, (2) large changes, (3) small changes, (4) minimal changes, and (5) a set of auto-generated random changes. We first graph and discuss the $\text{MAX}|\Delta s_t|$ fluctuations observed during each of the demand changes. We then provide task performance graphs, showcasing how StimHab agents handle demands after each change. We discuss what can be learned about changes in stimuli with respect to changes in task demands from looking at $\text{MAX}|\Delta s_t|$ and the corresponding task performances, simultaneously. Finally, we discuss how these observations can improve respecialization through automated threshold resetting.

We observe fluctuations in $\text{MAX}|\Delta s_t|$ given the following changes in task demands:

(format: "task name—percentage of all agents that are continuously needed on this task")

- maximal: alternating (T1-1%, T2-99%) and (T1-99, T2-1%);
- large: alternating (T1-10%, T2-90%) and (T1-90%, T2-10%);
- small: alternating (T1-60%, T2-40%) and (T1-40%, T2-60%);
- minimal: alternating (T1-50%, T2-50%) and (T1-51%, T2-49%);
- random: (T1-2%, T2-98%), (T1-24%, T2-76%), (T1-66%, T2-34%), (T1-48%, T2-52%), (T1-18%, T2-82%), (T1-72%, T2-28%), (T1-40%, T2-60%), (T1-9%, T2-91%), (T1-49%, T2-51%), (T1-57%, T2-43%).

The resulting $\text{MAX}|\Delta s_t|$ for each simulation step are graphed in Fig. 2-left column, with the corresponding task performance values shown on the right.

Figure 2-left graphs maximum absolute changes in s_t per step, for a setup of two tasks. Each graph in the figure corresponds to a different demand change amount: (a) maximal, (b) large, (c) small, (d) minimal, and (e) random. For each $\text{MAX}|\Delta s_t|$ graph on the left, the horizontal axis tracks simulation steps, while the vertical axis tracks $\text{MAX}|\Delta s_t|$ (e.g., if the most changed task stimulus for the step drops by 0.2, a spike of height 0.2 will appear for that step). Demands are changed every 5000 steps, and in most cases producing in a visible spike: given a relatively large positive $\Delta\text{MAX}|\Delta s_t|$: current $\text{MAX}|\Delta s_t| \gg$ previous $\text{MAX}|\Delta s_t|$, the new y-value is considerably larger, resulting in a spike. For maximal demand changes in graph (a)-left, spikes of $\text{MAX}|\Delta s_t|$ are the tallest, as the largest change in a task's demand results in the largest change in the corresponding task stimulus, be it positive or negative. For minimal changes in demand represented in graph (d)-left, minimal fluctuations are observed after the initial specialization over the first 50 cycles, as agents continue working based on their existing specializations. Overall, as demand changes decrease moving from graph (a) to (b) to (c) to (d), so does the

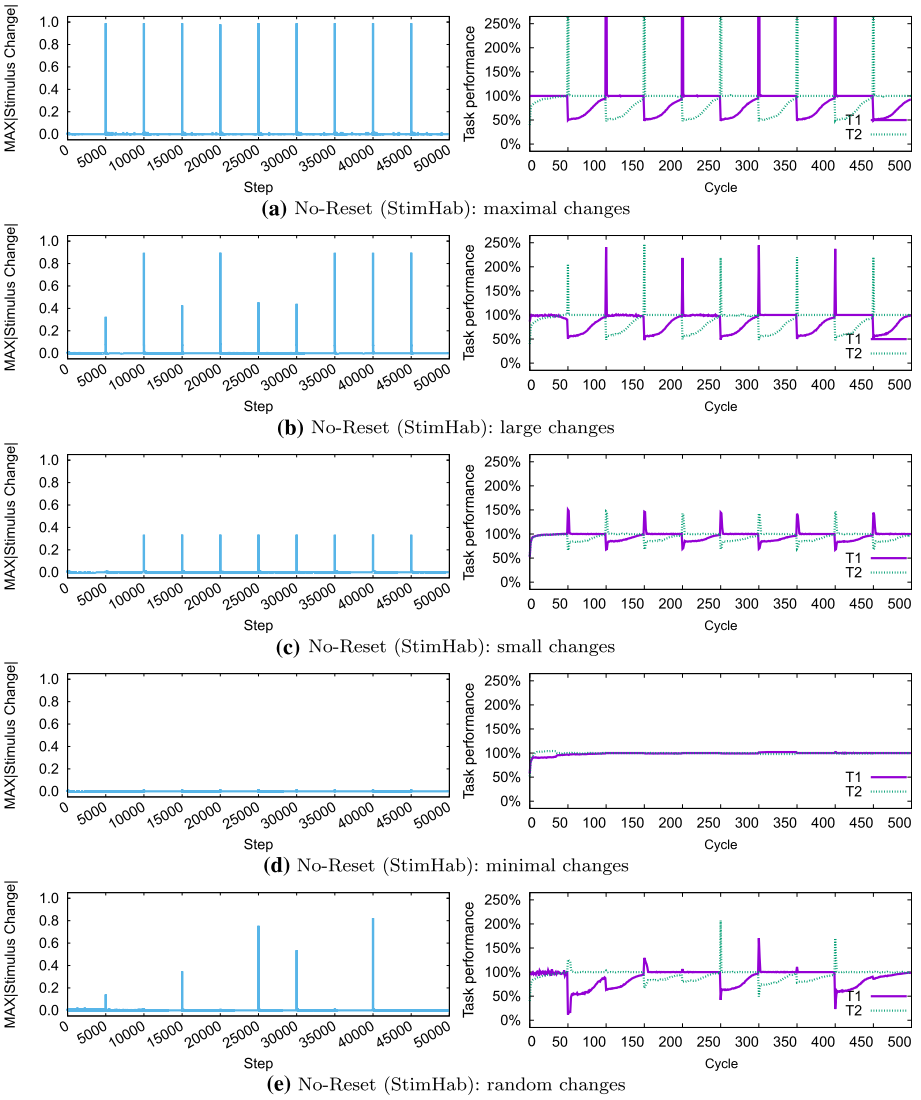


Fig. 2 Effect that changing task demands has on (left) MAX$|\Delta s_t|$ fluctuations per step under the approach with No-Reset (standard StimHab) and (right) the corresponding task performances for a setup of 2 tasks with demands changing 10 times, every 5000 steps = 50 cycles. We see that, generally, larger demand changes correspond to taller spikes in MAX$|\Delta s_t|$ (left) and larger spikes in performance (upward or downward) (right). By looking at task performances (right), we also see that after the initial specialization (cycles 0–50), returning to 100% performance takes significantly longer during each subsequent 50-cycle readaptation, demonstrating a struggle to respecialize

height of MAX$|\Delta s_t|$ spikes in the graphs, culminating in barely visible spikes for minimal demand changes in graph (d). Some exceptions to this tendency can be observed in graphs (b) and (e). Graph (b) depicts relatively large demand switches, from (T1-10%, T2-90%) to (T1-90%, T2-10%) leading to relatively tall spikes, but of inconsistent height, despite consistent demand shift amounts. Looking closely, we can see that prior to all shorter spikes,

the MAX$|\Delta s_t|$ line fluctuates. When task allocation has not yet achieved the zero change in stimuli (characteristic of the system reaching an equilibrium point after initial adaptation), changes in task demands will not result in as large of a spike in MAX$|\Delta s_t|$. Graph (e) depicts spikes of varying heights, but these do not directly relate to the amount of shifted demand. Notice, for example, that a shift from T1-2% to T1-24% at step 5000 results in a spike of about 15%, while a shift from T1-24% to T1-66% at step 10,000 does not result in a larger spike, as may be expected (since the first change was of 22% and the second of 42%) and, in fact, appears to not result in a spike at all. Looking closer, we see that all shorter and missing spikes are preceded by fluctuations in MAX$|\Delta s_t|$, while taller spikes follow periods of MAX$|\Delta s_t| = 0.0$ (i.e., flat lines).

Figure 2-right depicts the per-task performances given the stimulus changes depicted in the left column. The horizontal axis tracks simulation cycles (of 100 steps each), while the vertical axis tracks task performance. Ideal task performance is 100%; values below indicate insufficient work is done on the task, while values above indicate excess work. We see that, as demand changes decrease from graph (a) to (b) to (c) to (d), so do the fluctuations of task performances, leading to smaller initial spikes away from the ideal 100% task performance. Given minimal changes in demand represented in graph (d), minimal fluctuations are observed after the initial specialization over the first 50 cycles, as agents continue working based on their existing specializations. Although we limit the y-axis to 200% to ensure visibility, maximal demand changes on graph (a) lead to task performance peaks nearing 1000%, while the smaller demand changes in graph (b) result in smaller peaks, nearing 250%.

A deeper understanding of why larger changes in demand do not always lead to larger spikes in MAX$|\Delta s_t|$ can be gained by analyzing both columns in Fig. 2, simultaneously. Consider that when a task is not getting sufficient work steps, its stimulus increases over time, until reaching the maximum of 1.0; a task that is getting too many work steps will have its stimulus decrease over time, until reaching a minimum of 0.0. This is precisely the case in Fig. 2e-right, over the demand period between cycles 50 and 100: T1 struggles to reach enough workers (its performance line is continuously below 100%), resulting in its stimulus increasing to $s_1 = 1.0$; T2 starts out with too many workers (line above 100%), causing stimulus to reach $s_2 = 0.0$ before finding the correct number of workers (line drops to 100% performance). At step 10,000 (cycle 100), demand for T1 increases (from 24 to 66%), resulting in T1 being even further from the correct amount of work steps, but its 1.0 stimulus can rise no further. Demand for T2 decreases (from 76 to 34%), but its 0.0 stimulus can fall no further. As neither task's stimulus can change any further in the direction determined by the changes in demand, no spike is observed in Fig. 2e-left. The same situation is observed on step 20,000 (cycle 200), 35,000 (cycle 350), and 45,000 (cycle 450). Thus, if demand changes alter stimuli further in the direction they have been evolving thus far, the curbing of stimuli to the range [0.0, 1.0] obscures the spike in MAX$|\Delta s_t|$.

We hypothesize that positive spikes in MAX$|\Delta s_t|$ observed in Fig. 2-left can help improve respecialization in decentralized systems using threshold reinforcement. As spikes in MAX$|\Delta s_t|$ are present for most changes in demand, they can serve as a heuristic to indicate system destabilization that warrants a reset of specialization thresholds. The special case of stimuli going below minimum or above maximum measured values occurs under continuous drops/rises in stimulus over the course of slow StimHab respecialization. To address this issue, un-curbed changes in s_t can be monitored (i.e., unrestricted to the [0.0, 1.0] stimulus range), revealing any s_t increases above 1.0 and drops below 0.0. In this work, however, we assume that agents cannot accurately calculate stimulus values beyond the maximum and minimum values. Consequently, in our tests, we work with s_t

values restricted to [0.0, 1.0] in order to investigate whether such edge-case blindness to demand changes hinders $\Delta\text{MAX}|\Delta s_i|$-based Auto-Reset approach to respecialization. As resetting thresholds can allow agents to respecialize as effectively as they are able to specialize (Kazakova and Wu 2018), we hypothesize that with timely threshold resetting, continuous deficiencies in task allocation leading to the min/max stimulus edge case described above will not be common. Additionally, if no spike is observed and threshold reset is not triggered, agents simply fall back to standard StimHab functionality, modifying their task habits through threshold reinforcement alone.

6.3 Experiments with max, min, and random changes in demands of five tasks

We test how respecialization using automated threshold resetting performs given minimal, maximal, and random changes in the demands of 5 tasks. We compare Auto-Reset behavior to the baseline represented by StimHab's native method with No-Reset of thresholds, as well as to Central-Reset, previously shown to outperform No-Reset (Kazakova and Wu 2018). We plot task performances over the entire simulation run, to show performance over time in a system with dynamic demands. We also plot average task performance deviations and average task-switching rates, to assess the expected adaptation behavior each time demands change. The corresponding numerical data are made available in the Online Resource. Our results show that the proposed Auto-Reset trigger causes timely threshold resets, leading to improved readaptation, thus making it viable decentralized strategy for improving dynamic task allocation under response threshold reinforcement strategies such as StimHab.

6.3.1 Maximal changes in task demands

First, we test how the three approaches handle maximal changes in demands. For 5 tasks and 100 agents, such a change corresponds to initial demands of 1% of agents on all tasks except one, which has a demand of the remaining 96% of agents, switching to a new demand set that swaps any of the 1% tasks with the 96% task. On the next demand period (50 cycles later), this swap is repeated with any other 1% task and so on. Note that in order to calculate per-task performances we need a nonzero demand for each task and we cannot assign fewer than 1 agent to a task, which, given 100 agents, corresponds to a 1% demand.

Figure 3-left shows task performance percentages over all 10 demand periods for (a) Auto-Reset, (b) Central-Reset, and (c) No-Reset. The horizontal axis indicates simulation cycles; the vertical axis indicates task performance percentages. Lines near 100% are best, indicating appropriate amount of work on that task over time; spikes below or above 100% indicate insufficient or excess work, respectively. Auto-Reset and Central-Reset show nearly identical behavior, both quickly converging back to 100% after each change in demands. No-Reset does significantly worse than the other two: it takes approximately half of each demand period (25 cycles = 2500 agent decision steps) to find the right task allocation, at which point it quickly converges to 100% performance. Given how long it takes No-Reset to find appropriate task assignments, it can become unsuitable for systems where demands change more frequently. Although the actual cycles needed to adapt are specific to the tested threshold reinforcement affinity and aversion rates, we can clearly see that task reallocation benefits from threshold resetting. Figure 3-right corroborates these findings by plotting average performance deviations from 100% for each cycle of an adaptation period. Each n-th cycle represents the average of the n-th cycles of all ten demand switches

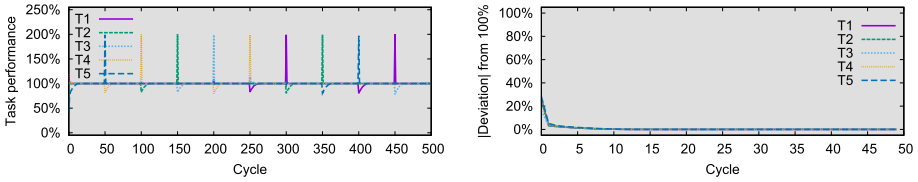
depicted in the graphs in the left column. The horizontal axis indicates simulation cycles; the vertical axis indicates task performance deviations from 100%. Here, lines near 0% are best, as we ideally want no deviation from 100% task performance. These average adaptation period graphs showcase what behavior can be expected given such large changes in system demands. We see that, when demands change on cycle 0 of an average adaptation period, Auto-Reset quickly converges to near 0% deviations after just one cycle, followed closely by Central-Reset, while No-Reset takes substantially longer to settle on the correct agent allocations.

Figure 4 shows the task-switching rates of the three approaches for each cycle of an average 50-cycle adaptation period. The horizontal axis shows simulation cycles; the vertical axis shows average task-switching rates. No-Reset incurs the most task switching, which remains around 60% for 25 cycles, at which point it quickly drops to 0%. This indicates that respecializing without first resetting the habit thresholds can be inefficient. Auto-Reset and Central-Reset both result in virtually identical task-switching rates that quickly decrease to 10% by cycle 5 and reaching 0% by cycle 15, demonstrating that resetting allows for faster adaptation to change and that Auto-Reset is able to match Central-Reset capabilities, while allowing agents to make independent reset decisions without requiring a central signal.

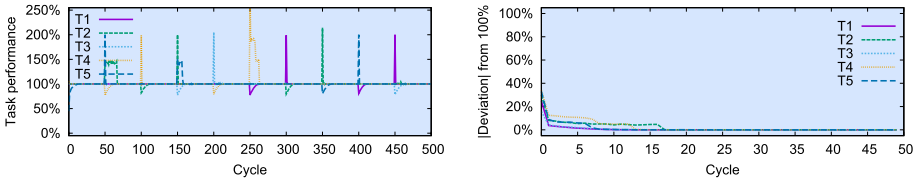
Table 1 in the Online Resource shows the numerical data for average deviations and task switching shown in Figs. 3-right and 4, respectively. Blue-to-white color gradient corresponds to average amount of task switching and corresponding confidence intervals, with darker blue indicating more task switching and thus less stable task assignments. Red-to-white color gradient corresponds to the average performance deviations and their confidence intervals for each task over an average 50-cycle adaptation period, with darker red indicating higher deviations and thus less ideal task allocation. Overall, lighter cells indicate better dynamic adaptation. No-Reset leads to much higher task switching and lower task performance deviations through the first 25 cycles. Average performance deviations for all tasks under Auto-Reset and Central-Reset begin ranging from 15 to 37% for cycle 0 and drop to 1% by cycle 8, but Central-Reset does end up settling on a considerably worse deviation of 10% for T2 by cycle 13, while Auto-Reset stabilizes at 0% deviation by cycle 10. No-Reset, however, begins with deviations ranging from 256 to 519% on cycle 0 and does not reach 0% deviation until cycle 27. All three approaches are able to settle on a stable task assignment, nearing 0% task switching per step, although Auto-Reset and Central-Reset reach it by step 16, while No-Reset takes until step 28. Additionally, No-Reset also starts out with more than double the average number of task switches on cycle 0 (60%, as compared to 27% under Auto-Reset and 27% under Central-Reset) and continuing with similarly high task switching until cycle 21, when if finally begins dropping. Note the nonzero deviations present for T2 throughout Central-Reset. For this set of tests, T2 requires only one agent over several of the demand periods. Misallocation of this one agent over any one demand period results in 0% performance and 100% deviation for that period, increasing the average deviation for T2 under Central-Reset, as seen in Table 1 of the Online Resource.

6.3.2 Minimal changes in task demands

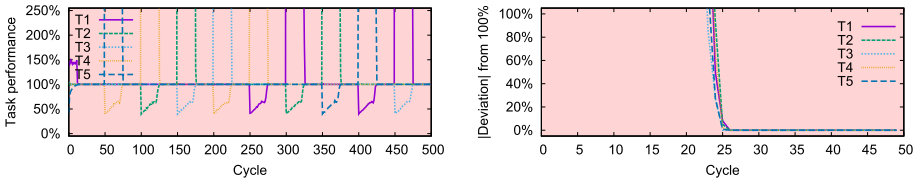
Next, we test how the three approaches handle minimal changes in demand. Recall that we need nonzero demands on all tasks for performance calculations to be possible. Thus, the minimal possible change in system demands for a setup with 5 tasks corresponds to one fewer agents needed on one task, while one more agent is needed on another. Given 100



(a) Auto-Reset: maximal changes



(b) Central-Reset: maximal changes



(c) No-Reset (StimHab): maximal changes

Fig. 3 Dynamic task allocation behavior of 100 agents, given maximal changes in the demands of 5 tasks (T1–T5), with demands changing every 50 cycles, shown as (left) per-task performances throughout a sample run of 10 demand periods and (right) per-task deviations from 100% performance, averaged over these 10 demand periods for **a** Auto-Reset, **b** Central-Reset, and **c** No-Reset (standard StimHab). The correct continuous amount of work for a task is shown as color lines near 100% performance on the left and lines near 0% deviation on the right (Color figure online)

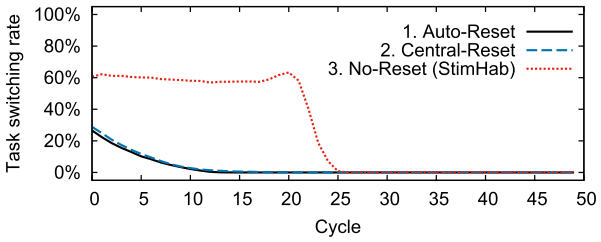


Fig. 4 Task-switching rates per cycle of an average demand period for Auto-Reset, Central-Reset, and No-Reset, given maximal changes in the demands of 5 tasks. Lines near 0% are best. These rates show how task assignments stabilize over the course of an average adaptation period. Approaches where thresholds are reset prior to re-adaption adapt more efficiently, reaching near 0% task switching by cycle 15

agents, such a change corresponds to initially equal 20% demands across all tasks, switching to a new demand set of 21% on any one task, 19% on another, and 20% on the remaining tasks; for the next demand period (50 cycles later) demands switch back to the equally distributed demand set, and so on, for the remainder of the 10 demand periods.

Figure 5-left shows task performance percentages over 10 demand periods for all three approaches. This time, No-Reset remains near 100% performance, as it allows the majority

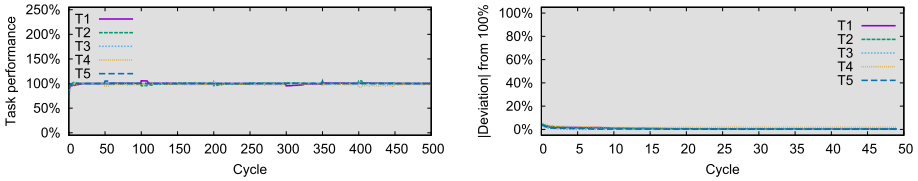
of the agents to continue working according to their initially developed specializations (cycles 0–49) throughout the run. The last demand switch causes a spike in performance Central-Reset task performances spike away from 100% every time demands change, as agents receive a signal to reset their thresholds back to random values. When such drastic readaptation is not needed, given a small environmental change, resetting can become a destructive force. This makes a case against resetting agents' existing specializations every time demands change. Auto-Reset does not trigger any threshold resets given the small changes, essentially falling back to basic StimHab functionality, closely matching No-Reset performance. This suggests that Auto-Reset can be useful in limiting the destructive force of specialization resets to the cases where existing specializations are not longer useful. Figure 5-right corroborates these findings by plotting average deviations away from 100% task performance for each cycle of an average 50-cycle adaptation period. Given the forced resets, Central-Reset displays the highest performance deviations for the first 5 steps of each demand period. Auto-Reset and No-Reset show identically low and nearly ideal deviation averages throughout the 50 cycles.

Figure 6 shows the average task-switching rates for each cycle of an average 50-cycle adaptation period for all three approaches. Central-Reset incurs the highest task-switching rate near the beginning of readaptation, given its system-wide resets and the subsequent periods of readaptation, but quickly drops from 20% task switching to near 0% over the first 5 cycles. Auto-Reset and No-Reset both remain at near 0% task switching for the entirety of the 50 cycles. Note, however, that none of the systems quite reach 0% average task switching, as the small agent misallocations across tasks continue changing stimuli and causing agents to keep adjusting their task choices throughout.

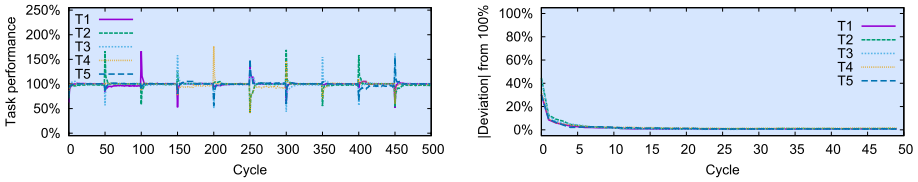
Table 2 in the Online Resource shows the numerical data for the average deviations and the task-switching rates depicted in Figs. 5-right and 6, respectively. Recall that lighter cells correspond to better task allocation: darker blue cells indicate higher amounts of task switching, while darker red cells indicate higher task performance deviations. We see that all three approaches result in low-performance deviations across all tasks. Nevertheless, Central-Reset does incur a large increase in task switching and performance deviation averages for the first two cycles after demands change: task-switching rates for the 0-th cycle of an average adaptation period are 22% for Central-Reset, compared to 5% for Auto-Reset and 7% for No-Reset; per-task performance deviations for the 0-th cycle reach as high as 44% for Central-Reset, compared to 6% for Auto-Reset and 5% for No-Reset. While this increase is short-lived, as Central-Reset quickly reaches task allocation similar to that of the other two approaches, depending on the domain, even a temporary increase in task switching and fluctuations in task performance may be costly, again suggesting that forced specialization resets every time demands change may be too destructive for some applications.

6.3.3 Random changes in task demands

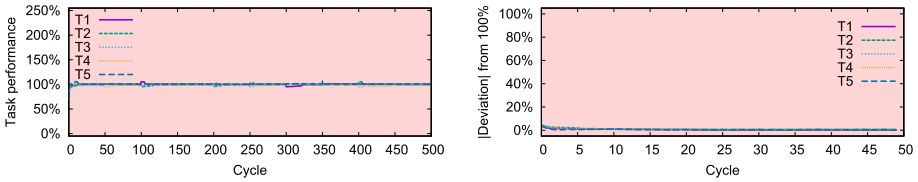
Finally, we test how the three approaches handle random-sized changes in task demands. Given that real-world domains are likely to have a variety of demand variations, this test is more representative of real-world performance than the maximal and minimal change experiments. For the setup of 5 tasks, we auto-generate a set of random task demands adding up to 100% of the available agents for every step of each cycle, ensuring that every agent is needed continuously. Per-task demands are shown in Table 1. Recall that percentages can instead be equivalently seen as the corresponding explicit numbers of agents.



(a) Auto-Reset: minimal changes



(b) Central-Reset: minimal changes



(c) No-Reset (StimHab): minimal changes

Fig. 5 Dynamic task allocation behavior of 100 agents, given minimal changes in the demands of 5 tasks (T1–T5), with demands changing every 50 cycles, shown as (left) per-task performances throughout a sample run of 10 demand periods and (right) per-task deviations from 100% performance, averaged over these 10 demand periods for **a** Auto-Reset, **b** Central-Reset, and **c** No-Reset (standard StimHab). The correct continuous amount of work for a task is shown as color lines near 100% performance on the left and lines near 0% deviation on the right (Color figure online)

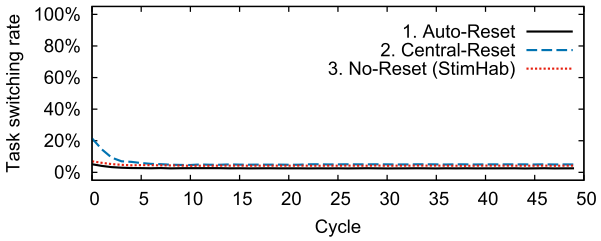


Fig. 6 Task-switching rates per cycle of an average demand period for Auto-Reset, Central-Reset, and No-Reset, given minimal changes in the demands of 5 tasks. Lines near 0% are better. These rates show how task assignments stabilize over the course of an average adaptation period. Given almost no change after initial demands, the approaches perform similarly, quickly reaching high specialization seen in the near 0% task-switching rates and remaining there for the duration of the run

Figure 7-left shows task performance percentages the 10 demand periods for all three systems. Auto-Reset task performances closely approximate Central-Reset, generally quickly converging to the ideal 100% performance across all tasks. No-Reset task performance lines are rarely near 100% after the initial specialization (cycle 0–50), showcasing the agents struggling to respecialize. Figure 7-right corroborates these findings by plotting

Table 1 Per-task demands for each of the 10 demand periods for "random changes" test

Demand period	T1 (%)	T2 (%)	T3 (%)	T4 (%)	T5 (%)
#1 (cycles 0– 49)	6	20	32	18	24
#2 (cycles 50–149)	11	10	24	30	25
#3 (cycles 100–149)	15	41	7	9	28
#4 (cycles 150–199)	26	13	27	6	28
#5 (cycles 200–249)	3	34	19	35	9
#6 (cycles 250–299)	35	18	15	6	26
#7 (cycles 300–349)	31	5	13	3	48
#8 (cycles 350–399)	54	14	9	18	5
#9 (cycles 400–449)	11	19	2	51	17
#10 (cycles 450–499)	30	22	25	14	9

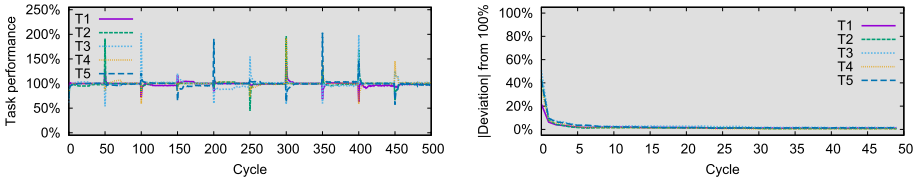
average performance deviations away from 100% for each cycle of an average 50-cycle adaptation period. Auto-Reset and Central-Reset averages match closely, both quickly dropping to 0% deviations for all tasks, while No-Reset deviations remain at 20–40% for the majority of the 50-cycle adaptation period. This supports our hypothesis that Auto-Reset can improve StimHab-based task allocation under dynamic conditions, while remaining fully decentralized (since it does not rely on a centralized signal to reset the way Central-Reset does).

Figure 8 shows the average task-switching rates for each cycle of an average 50-cycle adaptation period. As expected from the behavior observed in Fig. 7, average task-switching rates for Auto-Reset and Central-Reset are comparable, both starting at 20% and dropping to near 0% by cycle 5, while No-Reset incurs vastly larger amounts of task switching, starting near 40% and barely reaching 20% by the end of the 50 cycles. Given the many benefits of stable task assignments (diminished interference, skill-acquisition, time efficiency, etc.), we see that Auto-Reset and Central-Reset can offer an advantage.

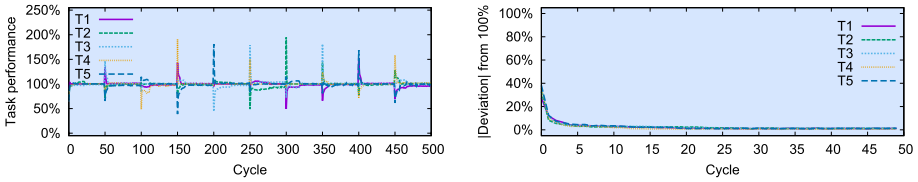
Table 3 in the Online Resource shows the numerical data for average deviations and task switching depicted in Figs. 7-right and 8, respectively. Auto-Reset closely approximates Central-Reset and both readapt well overall: average deviations for all tasks drop below 5% by cycle 5. No-Reset struggles, both in terms of task performance deviations and task-switching rate unable to match Auto-Rest and Central-Reset performance even by the end of the 50-cycle adaptation period. Additionally, No-Reset confidence intervals are significantly higher, indicating that tasks are experiencing large performance fluctuations throughout, which can be extremely detrimental to overall system behavior.

6.4 Experiments with more agents, more tasks, and more demand switches

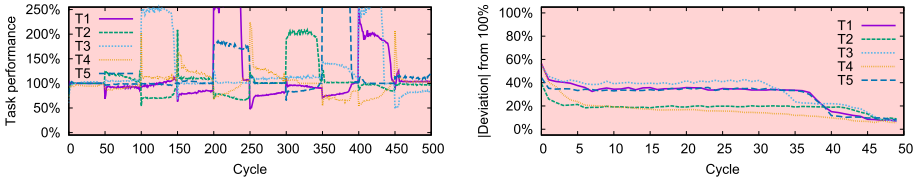
To further assess the reliability of the observed behaviors, we now repeat our last test with random changes in task demands, this time with more agents, more tasks, and over a longer period of time. We increase the number of agents from 100 to 1000, increase the number of tasks from 5 to 10, and increase the duration of each test from 10 to 100 changes in demands (i.e., from 500 to 5000 cycles or, equivalently, from 50,000 to 500,000 decision steps). We present deviation and task-switching graphs for the Auto-Reset, Central-Reset, and No-Reset approaches, showcasing their dynamic task allocation quality and stability for an average adaptation period.



(a) Auto-Reset: random changes



(b) Central-Reset: random changes



(c) No-Reset (StimHab): random changes

Fig. 7 Dynamic task allocation behavior of 100 agents, given random changes in the demands of 5 tasks (T1–T5), with demands changing every 50 cycles, shown as (left) per-task performances throughout a sample run of 10 demand periods and (right) per-task deviations from 100% performance, averaged over these 10 demand periods for **a** Auto-Reset, **b** Central-Reset, and **c** No-Reset (standard StimHab). the correct continuous amount of work for a task is shown as color lines near 100% performance on the left and lines near 0% deviation on the right (Color figure online)

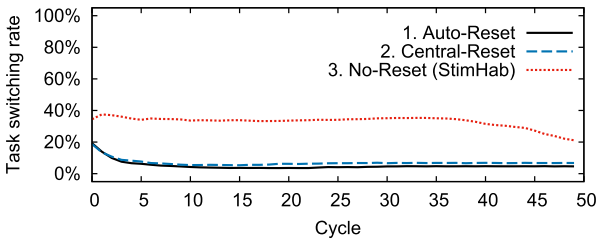
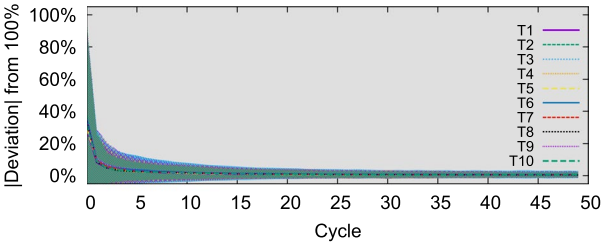
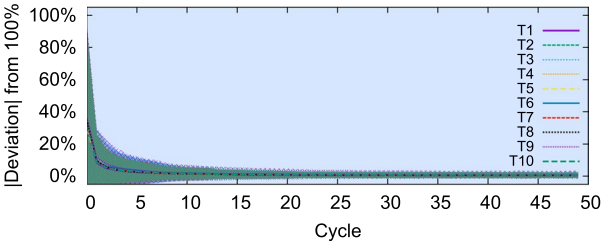


Fig. 8 Task-switching rates per cycle of an average demand period for Auto-Reset, Central-Reset, and No-Reset, given random changes in the demands of 5 tasks. Lines near 0% are better. The downward slope of all lines shows that task assignments stabilize during average adaptation period for all three approaches, but approaches where thresholds are reset prior to re-adaption reach significantly more stable task assignments (below 10% task switching) and do so significantly faster (approximately by cycle 5)

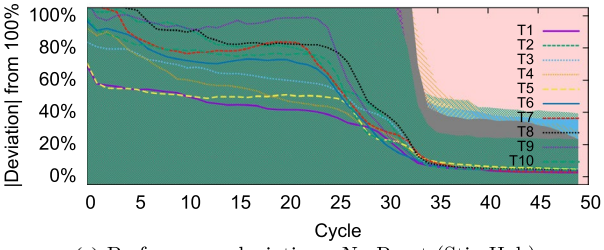
In Fig. 9, we provide average performance deviations and the corresponding 95% confidence intervals (C.I.) per task for each of the three approaches, showcasing both the quality and robustness of their dynamic task allocation. Note here that, as we have augmented the graph to depict C.I. for each task's performance deviation, where all C.I. overlap, a dark green shaded area can be seen. Note also that this time we omit graphing per-task performances across all cycles, as it is not feasible to print 100 changes



(a) Performance deviations: Auto-Reset

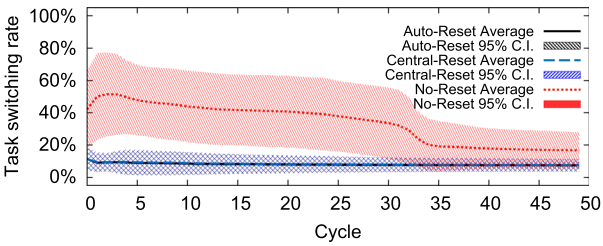


(b) Performance deviations: Central-Reset



(c) Performance deviations: No-Reset (StimHab)

Fig. 9 Task performance deviations for an average adaptation period. Color lines are average deviations and shaded regions are 95%C.I. for 10 tasks T1–T10, obtained with 1000 agents and demands changing every 50 cycles, 100 times per test. Lines near 0% and narrower shaded regions represent more accurate and reliable task allocation. Dark green areas where all C.I. overlap indicate similar performance across tasks (Color figure online)



Task Switching Rate: Auto- vs. Central- vs. No-Reset (StimHab)

Fig. 10 Task-switching rates per cycle of an average demand period for Auto-Reset, Central-Reset, and No-Reset. Averages (lines) and C.I. (shaded regions) are obtained with 1000 agents, 10 tasks, and demands changing every 50 cycles, 100 times per test. Lines near 0% and narrower shaded regions represent more stable and efficient task allocation (Color figure online)

in demands intelligibly (as compared to the 10 changes in the earlier tests). The average deviation graphs confirm earlier observations, with Auto-Reset closely matching Central-Reset behavior: in both cases, deviations quickly drop to 10% across all tasks, reaching near 0% by cycle 15 of an average adaptation period; C.I. intervals are identical for both systems, ultimately settling around $\pm 1\%$, indicating extremely dependable

behavior across tasks and demand changes. Both Auto-Reset and Central-Reset drastically outperform the No-Reset approach, under which task deviations remain above 50% for the first 25 cycles of adaptation, and then slowly fall to near 5%. Furthermore, we see that C.I. regions under No-Reset are much wider for all tasks, overlapping and covering most of the graph, showcasing significantly less dependable behavior for each task.

In Fig. 10, we provide the task-switching rates for all three approaches, along with the corresponding 95% C.I. We again see identical behavior for Auto-Reset and Central-Reset, with task switching starting at 11% at the beginning of an average adaptation period and dropping to 7% toward the end. While these rates are higher than the ideal 0%, they far outperform No-Reset, which incurs 40–50% task switching at the beginning of adaptation and slowly decreases to 16% by the end of the 50-cycle adaptation period. Furthermore, both Auto-Reset and Central-Reset demonstrate highly reliably behavior, seen in the narrow C.I. that reach $\pm 5\%$, while No-Reset C.I. remain around $\pm 20\%$ for over 30 cycles, later narrowing to around $\pm 10\%$. Overall, No-Reset is unable to match the stable and reliable task allocations achieved by the approaches employing threshold resetting before readaptation.

These results showcase the extensibility of the automated threshold resetting approach to larger decentralized teams of agents and to domains with more tasks. The earlier tests with fewer agents, fewer tasks, and fewer task switches, discussed in Sect. 6.3.3 produced comparable results. Note, however, that the presented tests are only assessing behavior in a general case of random demand changes across tasks. Special edge-case behaviors, such as when task demands change very slowly, adding up to larger changes over time, would need to be considered separately and may require additional safeguards or stability checks (livelocks may occur, with agents cycling through a subset of tasks to continuously compensate for outdated specializations).

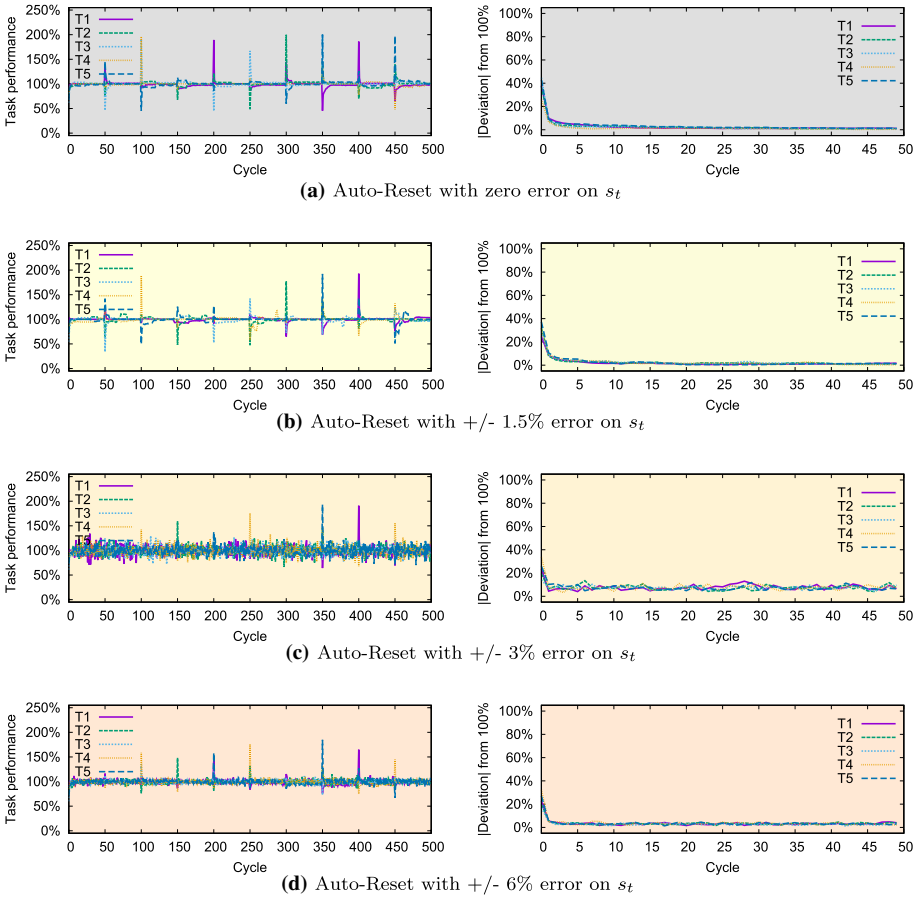
7 Noisy stimuli: a sensitivity analysis

As automated threshold resetting relies on detecting stimulus changes, noise in sensing these stimuli can affect the agents' ability to reset as needed. In this section we present Auto-Reset experiments with varying levels of Gaussian noise on the stimuli that are being independently sensed by the agents. We first compare per-task deviations and task-switching rates for Auto-Reset given no noise versus three-level noise. We then incorporate the expected noise range into the resetting threshold formula and repeat the experiment. Results show that Auto-Reset based on observing changes in stimuli can be robust given small levels of noise if the expected noise range is known and accounted for in the threshold reset-triggering value.

For our experiments, we apply Gaussian perturbations to the stimuli sensed by the agents. Each agent senses the stimulus for each task as the actual stimulus value plus an error equal to a percentage of that value. This error follows a Gaussian distribution, with a mean $\mu = 0.0$ and one of three standard deviations (st.dev.): $\sigma = 0.005$, $\sigma = 0.01$, and $\sigma = 0.02$, representing ranges of $\pm 1.5\%$, $\pm 3\%$, and $\pm 6\%$ error, respectively. Thus, 99.7% of the sensed stimuli fall within the range $[s_t - (s_t * 3 * \sigma), s_t + (s_t * 3 * \sigma)]$. For this test we return to the earlier setup of 100 agents, 5 tasks, and 10 demand changes of random size, identical to Sect. 6.3.3.

We first assess the Auto-Reset behavior given the resetting condition defined in Sect. 5.2 under the four noise levels. Resulting task performances and deviations are shown in Fig. 11, in the left and right columns, respectively. We see that $\pm 1.5\%$ noise performs comparably to the noise-free setup, but higher noise ($\pm 3\%$ and $\pm 6\%$) leads to an increase in deviations. Perhaps curiously, we see worse performance for $\pm 3\%$ than for $\pm 6\%$; the cause is more resetting being triggered for $\pm 6\%$ errors, causing agents to specialize less, but respond more to stimuli. This is corroborated by the task-switching rates in Fig. 12: $\pm 1.5\%$ leads to minimal task switching, similar to no noise, while higher noise leads to higher task switching. While low average performance deviations indicate that task allocation still works given these errors, recall that in our implementation there is no cost to switching tasks; in a system where task switching is not free, the higher task switching will negatively affect performance.

We hypothesize that if the range of the expected stimulus noise for a given system is known, it can be incorporated into the threshold resetting trigger we defined in Sect. 5.2 to



(a) Auto-Reset with zero error on s_t

(b) Auto-Reset with +/- 1.5% error on s_t

(c) Auto-Reset with +/- 3% error on s_t

(d) Auto-Reset with +/- 6% error on s_t

Fig. 11 Auto-Reset per-task performance given reset trigger that ignores the error. Task performance negatively affected by errors in stimulus estimation (see fluctuations away from 100% in the left column). Looking at average performance deviations (right column), we see that higher errors appear to lead to higher performance deviations

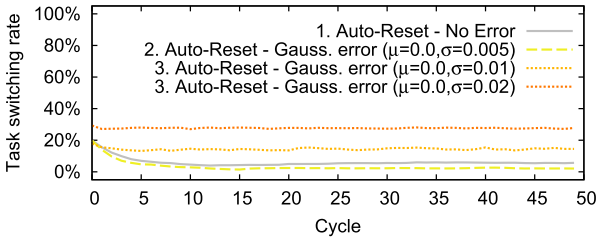


Fig. 12 Auto-Reset task-switching rates given reset trigger that ignores the error. Task-switching rates increase very quickly given small increases in the task stimulus estimation errors. Recall that there is no explicit cost to task switching in the current systems; otherwise, these increases in task-switching rates would have also resulted in significantly higher fluctuations in task performance

allow agents to ignore small error-based fluctuations, potentially leading to improved auto-resetting capabilities. Considering that two consecutively sensed task stimuli can now fall 6 error standard deviations apart (i.e., 3σ in both the positive and the negative directions), we propose that, given some number of steps per cycle and the number of tasks, agents automatically reset their specialization thresholds $\theta_{a,t}$ to uniformly random values when:

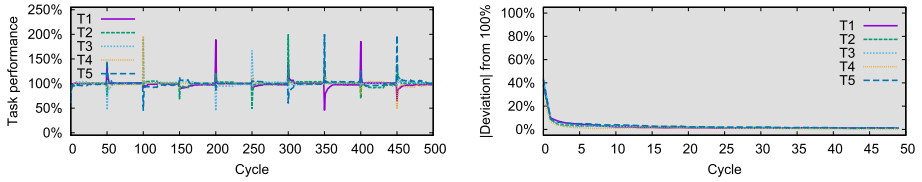
$$(\Delta\text{MAX}|\Delta s_t| + 6 * \sigma) \geq (\text{number of tasks})/(\text{steps in cycle})$$

While increasing the amount of change needed for agents to reset their thresholds can result in missed resetting given small changes, in the average case we expect improved performance.

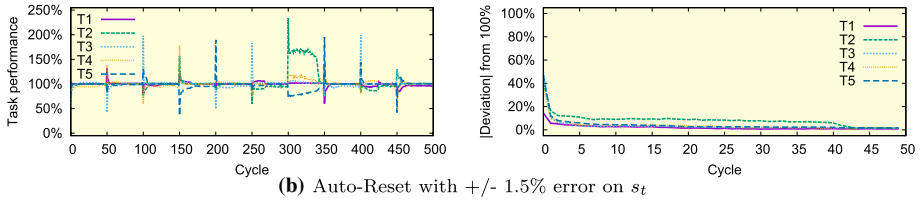
Figure 13 shows the effect of this updated resetting trigger on Auto-Reset given the four error rates. As expected, we see that twice a needed reset was missed at cycle 300, under $\pm 1.5\%$ and $\pm 6\%$ errors, increasing average deviations for T2. In all other cases, resets now happen when demands change, but the performance at $\pm 6\%$ error suffers anyway; the cause is that agents do not reset excessively as before, beginning to specialize instead, but the higher stimulus errors confuse the agents' threshold reinforcement path to specializations. Figure 14 showcases the lower resetting and higher specialization for all noise levels, with task switching quickly stabilizing to near 0%. While deviations are lower under the original resetting trigger, if task switching is not free, the lower task switching under the updated resetting trigger will ultimately benefit performance. These results support the viability of improving dynamic task allocation via appropriately timed automated threshold resetting.

8 Conclusions and future directions

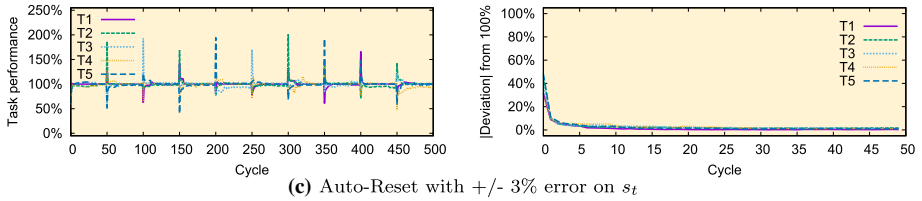
In this work, we investigated whether observing changes in stimuli can serve as an indication of changes in system needs that require agents to respecialize. We tested whether agents can independently detect when their specializations have become outdated and reset their per-task specialization thresholds back to uniformly random values, in order to respecialize without fighting pre-existing conditioning, thus improving adaptability in dynamic environments. We proposed a new automated trigger value for resetting agents' thresholds based on the minimal possible change in task demands and tested the resulting adaptability behavior on a set of *ongoing* tasks representing patrollable areas, with patrolling requirements changing over time. Results show that augmenting standard StimHab (defined in (Theraulaz et al. 1998) and presented here as No-Reset) with automated threshold resetting (presented here as Auto-Reset)



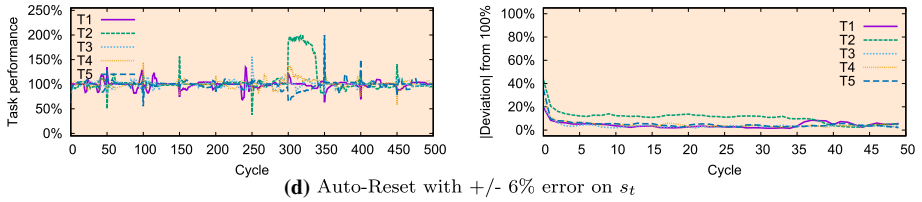
(a) Auto-Reset with zero error on s_t



(b) Auto-Reset with +/- 1.5% error on s_t



(c) Auto-Reset with +/- 3% error on s_t



(d) Auto-Reset with +/- 6% error on s_t

Fig. 13 Auto-Reset per-task performances given a reset trigger that accounts for error. While task assignment stability is negatively affected by errors in stimulus estimation (see fluctuations away from 100% in the left column), accounting for the error range in the Auto-Reset threshold still results in low average deviations across all tasks

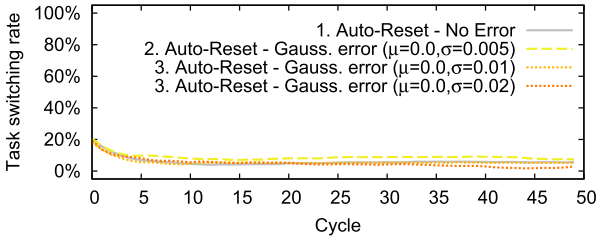


Fig. 14 Auto-Reset task-switching rates given a reset trigger that accounts for error. Accounting for the expected error range in the agents' task stimulus estimation when establishing the value of the Auto-Reset threshold $\Delta\text{MAX}|\Delta s_t$ leads to lower task-switching rates at all tested error levels, showing that this approach is capable of task allocation stability comparable to that of the error-free setup

outperforms standard StimHab, while matching and in some cases outperforming StimHab augmented with a central resetting signal sent to the agents when demands change (presented here as Central-Reset), which has been previously shown to improve respecialization under response threshold reinforcement (Kazakova and Wu 2018).

The proposed Auto-Reset method outperformed Central-Reset and No-Reset by leveraging the strengths of the latter two approaches under a variety of changes in task demands. No-Reset achieved accurate and stable task assignments only under small changes in demands, as initially developed remained generally well suited for the subsequent sets of similar demands. Given larger changes, No-Reset struggled to respecialize, as agents had to re-adapt while fighting pre-existing conditioning. Central-Reset drastically improved upon No-Reset under most conditions, as pre-existing conditioning was erased by the resetting of the habit thresholds. Central-Reset did, however, produce overly destructive behavior when respecialization resetting was not needed given only small changes in demands. Auto-Reset can forego resetting existing specializations if demand changes are minimal, thus approximating No-Reset behavior in such cases. Given larger changes, Auto-Reset successfully triggered decentralized threshold resetting, thus producing a behavior that is more sensitive to the actual system needs, as opposed to resetting never (as under the No-Reset approach) or always (as under the Central-Reset approach). Thus, Auto-Reset task allocation: (1) closely approximated No-Reset's less destructive task assignment given small changes, (2) matched Central-Reset's efficient respecialization in cases of larger changes, (3) maintained low task-switching rates in all cases, (4) all while eliminating the need for centralized resetting triggers.

The proposed approach seamlessly scaled to more tasks and agents, producing reliable adaptation. Noise sensitivity analysis indicated that some levels of noise, especially if known, can be handled without significant disruption to the behavior. Nevertheless, there are edge cases not being considered, such as when minimal demand changes slowly, leading to larger shifts over time: as Auto-Reset only considers immediate changes, if that change is undetected by the trigger threshold, repeated changes will also go undetected. A secondary safeguard may be needed to detect such outdated specializations (e.g., if ideal task allocation implies remaining on one task, average time on current task could be used in conjunction with time since the last reset: if it has been a while since the agent reset, but the time on current task is quite short, we can assume that there is some continuous task switching happening, characteristic of poor task allocation). Additionally, for a more in-depth analysis, further investigation should be conducted into environments with even higher numbers of tasks, with continuously changing demands, with changing agent number and capabilities, with too few agents to keep all tasks at 100% performance, or where agents must maintain specializations on more than one task simultaneously (e.g., consider alternating sets of tasks, such as when some factory tasks are needed during the day and others are needed during the night).

Acknowledgements This research was supported in part by ONR Grant N000140911043 and NSF Grant IIS1816777.

References

Agassounon, W., & Martinoli, A. (2002). Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: Part 3* (pp. 1090–1097). ACM.

Agassounon, W., Martinoli, A., & Goodman, R. (2001). A scalable, distributed algorithm for allocating workers in embedded systems. In *2001 IEEE international conference on systems, man, and cybernetics* (Vol. 5, pp. 3367–3373).

Agmon, N., Urieli, D., & Stone, P. (2011). Multiagent patrol generalized to complex environmental conditions. In *Proceedings of the twenty-fifth conference on artificial intelligence (AAAI'11)*.

Almeida, A., Ramalho, G., Santana, H., Tedesco, P., Menezes, T., Corruble, V., et al. (2004). Recent advances on multi-agent patrolling. In A. L. C. Bazzan & S. Labidi (Eds.), *Advances in artificial intelligence: SBIA 2004* (pp. 474–483). Berlin: Springer.

Berman, S., Halasz, A., Kumar, V., & Pratt, S. (2007). Bio-inspired group behaviors for the deployment of a swarm of robots to multiple destinations. In *Proceedings 2007 IEEE international conference on robotics and automation* (pp. 2318–2323).

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., & Dorigo, M. (2014). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-agent Systems*, *28*(1), 101–125.

Campbell, A., & Wu, A. S. (2011). Multi-agent role allocation: Issues, approaches, and multiple perspectives. *Autonomous Agents and Multi-agent Systems*, *22*(2), 317–355.

Campos, M., Bonabeau, E., Theraulaz, G., & Deneubourg, J. L. (2000). Dynamic scheduling and division of labor in social insects. *Adaptive Behavior*, *8*(2), 83–95.

Chu, H. N., Glad, A., Simonin, O., Sempe, F., Drogoul, A., & Charpillet, F. (2007). Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In *19th IEEE international conference on tools with artificial intelligence, 2007. ICTAI 2007* (Vol. 1, pp. 442–449). IEEE.

Cicirello, V. A., & Smith, S. F. (2004). Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 237–266.

de Lope, J., Maravall, D., & Quiñonez, Y. (2012). Decentralized multi-tasks distribution in heterogeneous robot teams by means of ant colony optimization and learning automata. In *International conference on hybrid artificial intelligence systems* (pp. 103–114). Springer.

de Lope, J., Maravall, D., & Quiñonez, Y. (2015). Self-organizing techniques to improve the decentralized multi-task distribution in multi-robot systems. *Neurocomputing*, *163*, 47–55.

Dias, M. B. (2004). *Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments* (p. 153). Robotics Institute: Pittsburgh.

Dos Santos, D. S., & Bazzan, A. L. (2012). Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. *Applied Soft Computing*, *12*(8), 2123–2131.

Dos Santos, F., & Bazzan, A. L. (2009). An ant based algorithm for task allocation in large-scale and dynamic multiagent scenarios. In *Proceedings of the 11th annual conference on genetic and evolutionary computation* (pp. 73–80). ACM.

Dos Santos, F., & Bazzan, A. L. (2011). Towards efficient multiagent task allocation in the robocup rescue: A biologically-inspired approach. *Autonomous Agents and Multi-agent Systems*, *22*(3), 465–486.

Ducatelle, F., Förster, A., Di Caro, G. A., & Gambardella, L. M. (2009). New task allocation methods for robotic swarms. In *9th IEEE/RAS conference on autonomous robot systems and competitions*.

Farinelli, A., Iocchi, L., Nardi, D., & Ziparo, V. A. (2006). Assignment of dynamically perceived tasks by token passing in multirobot systems. *Proceedings of the IEEE*, *94*(7), 1271–1288.

Ferrante, E., Turgut, A. E., Duéñez-Guzmán, E., Dorigo, M., & Wenseleers, T. (2015). Evolution of self-organized task specialization in robot swarms. *PLoS Computational Biology*, *11*(8), e1004273.

Ferreira, P., & Bazzan, A. L. (2006). Swarm-gap: A swarm based approximation algorithm for e-gap. In *First international workshop on agent technology for disaster management* (pp. 49–55).

Ferreira, P. R., Boffo, F. S., & Bazzan, A. L. (2007). Using swarm-gap for distributed task allocation in complex scenarios. In *International conference on autonomous agents and multiagent systems* (pp. 107–121). Springer.

Ferreira, P. R., Dos Santos, F., Bazzan, A. L., Epstein, D., & Waskow, S. J. (2010). Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. *Autonomous Agents and Multi-agent Systems*, *20*(3), 421–443.

Frison, M., Tran, N. L., Baiboun, N., Brutschy, A., Pini, G., Roli, A., Dorigo, M., & Birattari, M. (2010). Self-organized task partitioning in a swarm of robots. In *International conference on swarm intelligence* (pp. 287–298). Springer.

Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, *1*(1), 3–31.

Ghizzioli, R., Nouyan, S., Birattari, M., & Dorigo, M. (2005). An ant-based algorithm for the heterogeneous dynamic task allocation problem. Technical Report, TR/IRIDIA/2005-005.

Golfarelli, M., Maio, D., & Rizzi, S. (1997). Multi-agent path planning based on task-swap negotiation. In *Proceedings of the 16th UK planning and scheduling SIG workshop* (p. 69).

Halász, A., Hsieh, M. A., Berman, S., & Kumar, V. (2007). Dynamic redistribution of a swarm of robots among multiple sites. In *IEEE/RSJ international conference on intelligent robots and systems, 2007. IROS 2007* (pp. 2320–2325). IEEE.

Hsieh, M. A., Halász, Á., Berman, S., & Kumar, V. (2008). Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, *2*(2–4), 121–141.

Hsieh, M. A., Halász, Á., Cubuk, E. D., Schoenholz, S., & Martinoli, A. (2009). Specialization as an optimal strategy under varying external conditions. In *IEEE international conference on robotics and automation, ICRA'09* (pp. 1941–1946).

Jones, C., & Mataric, M. (2003). Adaptive division of labor in large-scale minimalist multi-robot systems. In *2003 IEEE/RSJ international conference on intelligent robots and systems, 2003 (IROS 2003). Proceedings* (Vol. 2, pp. 1969–1974). IEEE.

Kalra, N., & Martinoli, A. (2006). Comparative study of market-based and threshold-based task allocation. In *Distributed autonomous robotic systems 7* (pp. 91–101). Springer.

Kanakia, A., Touri, B., & Correll, N. (2016). Modeling multi-robot task allocation with limited information as global game. *Swarm Intelligence*, *10*(2), 147–160.

Kazakova, V. A., & Wu, A. S. (2018). Specialization vs. re-specialization: Effects of Hebbian learning in a dynamic environment. In *Florida artificial intelligence research society conference FLAIRS-31*.

Kira, Z., & Arkin, R. C. (2004). Forgetting bad behavior: Memory for case-based navigation. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS). Proceedings* (Vol. 4, pp. 3145–3152).

Kittithreerapronchai, O., & Anderson, C. (2003). Do ants paint trucks better than chickens? Markets versus response thresholds for distributed dynamic scheduling. In *The 2003 congress on evolutionary computation, 2003. CEC'03* (Vol. 2, pp. 1431–1439). IEEE.

Krieger, M. J., & Billeter, J. B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, *30*(1–2), 65–84.

Labella, T. H., Dorigo, M., & Deneubourg, J. L. (2006). Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *1*(1), 4–25.

Lee, W., & Kim, D. (2016). Local interaction of agents for division of labor in multi-agent systems. In *International conference on simulation of adaptive behavior* (pp. 46–54). Springer.

Lee, W., & Kim, D. (2017). History-based response threshold model for division of labor in multi-agent systems. *Sensors*, *17*(6), 1232.

Levinthal, D. A., & March, J. G. (1993). The myopia of learning. *Strategic Management Journal*, *14*(S2), 95–112.

Li, L., Martinoli, A., & Abu-Mostafa, Y. S. (2002). Emergent specialization in swarm systems. In *International conference on intelligent data engineering and automated learning* (pp. 261–266). Springer.

Liu, W., Winfield, A. F., Sa, J., Chen, J., & Dou, L. (2007). Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, *15*(3), 289–305.

Ma, H., Li, J., Kumar, T., & Koenig, S. (2017). Lifelong multi-agent path finding for online pickup and delivery tasks. In *Proceedings of the 16th conference on autonomous agents and multiagent systems, international foundation for autonomous agents and multiagent systems* (pp. 837–845).

Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, *33*, 1–17.

McIntire, M., Nunes, E., & Gini, M. (2016). Iterated multi-robot auctions for precedence-constrained task scheduling. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems, international foundation for autonomous agents and multiagent systems* (pp. 1078–1086).

Murciano, A., Millán, J. D. R., & Zamora, J. (1997). Specialization in multi-agent systems through learning. *Biological Cybernetics*, *76*(5), 375–382.

Nitschke, G., Schut, M., & Eiben, A. (2008). Emergent specialization in biologically inspired collective behavior systems. In *Intelligent complex adaptive systems* (pp. 215–253). IGI Global.

Nouyan, S. (2002). Agent-based approach to dynamic task allocation. In *International workshop on ant algorithms* (pp. 28–39). Springer.

Nouyan, S., Ghizzioli, R., Birattari, M., & Dorigo, M. (2005). An insect-based algorithm for the dynamic task allocation problem. *KI*, *19*(4), 25–31.

Nunes, E., & Gini, M. L. (2015). Multi-robot auctions for allocation of tasks with temporal constraints. In *AAAI* (pp. 2110–2116).

Nunes, E., McIntire, M., & Gini, M. (2016). Decentralized allocation of tasks with temporal and precedence constraints to a team of robots. In *IEEE international conference on simulation, modeling, and programming for autonomous robots (SIMPAR)* (pp. 197–202). IEEE.

Ono, N., & Fukumoto, K. (1996). Multi-agent reinforcement learning: A modular approach. In *Second international conference on multiagent systems* (pp. 252–258).

Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., & Birattari, M. (2011). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, *5*(3–4), 283–304.

Portugal, D., & Rocha, R. (2011). A survey on multi-robot patrolling algorithms. In *Doctoral conference on computing, electrical and industrial systems* (pp. 139–146). Springer.

Price, R., & Tiňo, P. (2004). Evaluation of adaptive nature inspired task allocation against alternate decentralised multiagent strategies. In *International conference on parallel problem solving from nature* (pp. 982–990). Springer.

Quiñonez, Y., Maravall, D., & de Lope, J. (2011). Stochastic learning automata for self-coordination in heterogeneous multi-tasks selection in multi-robot systems. In *Mexican international conference on artificial intelligence* (pp. 443–453). Springer.

Román, J. A, Rodríguez, S., & Corchado, J. M. (2014). Improving intelligent systems: Specialization. In *International conference on practical applications of agents and multi-agent systems* (pp. 378–385). Springer.

Schwarzrock, J., Zacarias, I., Bazzan, A. L., de Araujo Fernandes, R. Q., Moreira, L. H., & de Freitas, E. P. (2018). Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence. *Engineering Applications of Artificial Intelligence*, *72*, 10–20.

Tavares, A. R., Azpúrua, H., & Chaimowicz, L. (2014). Evolving swarm intelligence for task allocation in a real time strategy game. In *2014 Brazilian symposium on computer games and digital entertainment (SBGAMES)* (pp. 99–108). IEEE.

Tavares, A. R., Zuin, G. L., Azp, H., Chaimowicz, L., et al. (2017). Combining genetic algorithm and swarm intelligence for task allocation in a real time strategy game. *SBC Journal on Interactive Systems*, *8*(1), 4–19.

Theraulaz, G., & Bonabeau, E. (1999). A brief history of stigmergy. *Artificial Life*, *5*(2), 97–116. https://doi.org/10.1162/106454699568700.

Theraulaz, G., Bonabeau, E., & Deneubourg, J. L. (1998). Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B*, *265*, 327–332.

van Lon, R. R., & Holvoet, T. (2017). When do agents outperform centralized algorithms? *Autonomous Agents and Multi-agent Systems*, *31*(6), 1578–1609.

Villacorta, P. J., Pelta, D. A., & Lamata, M. T. (2013). Forgetting as a way to avoid deception in a repeated imitation game. *Autonomous Agents and Multi-agent Systems*, *27*(3), 329–354.

Wawerla, J., Vaughan, R. T. (2010). A fast and frugal method for team-task allocation in a multi-robot transportation system. In *ICRA* (pp. 1432–1437).

Westhus, C., Kleineidam, C., Roces, F., & Weidenmller, A. (2013). Behavioural plasticity in the fanning response of bumblebee workers: Impact of experience and rate of temperature change. *Animal Behaviour*, *85*(1), 27–34.

Wu, A. S., & Kazakova, V. A. (2017). Effects of task consideration order on decentralized task allocation using time-variant response thresholds. In *Florida artificial intelligence research society conference FLAIRS-30* (pp. 466–471).

Zheng, X., & Koenig, S. (2011). Generalized reaction functions for solving complex-task allocation problems. In *IJCAI proceedings-international joint conference on artificial intelligence* (Vol. 22, p. 478).