Adaptable decentralized task allocation for hierarchically-defined domains

Vera A. Kazakova
Gita R. Sukthankar
Department of Computer Science, University of Central Florida, US

ABSTRACT

Many real-world domains can benefit from adaptable decentralized task allocation through emergent specialization, especially in large teams of agents. We begin with an existing bio-inspired *response threshold reinforcement* approach for decentralized task allocation and extend it to handle hierarchical task domains. We test the extension on self-deployment of a large team of non-communicating agents to patrolling a hierarchically-defined set of areas. Results show near-ideal performance across all areas, while minimizing wasteful task switching through the development of specializations and subsequent respecializations when area demands change.

KEYWORDS

emergent behavior; specialization; dynamic multi-agent adaptation

1 INTRODUCTION

We investigate decentralized task allocation for dynamic domains with non-communicating workers who are neither limited nor informed by the actions of others or by task availability. An existing dynamic task allocation approach is adapted for a hierarchically-defined patrolling domain, consisting of multiple areas patrolled by 1000 agents. We assess whether the expected number of agents are present in each area over time, the stability of agents' self-assignments when area demands remain stable, and the adaptability of these self-assignments when area demands change.

We focus on decentralized emergent cooperation among agents whose task choices are neither limited nor informed by the choices of the other agents. Many real world domains involve ubiquitously available tasks which can be taken up by any number of agents at any time, though the actual task demands may vary. We refer to these as *ongoing* tasks. Examples include patrolling, monitoring, equipment diagnostics and maintenance, gathering resources, etc. Existing patrolling studies often ignore a need for flexible and scalable task allocation, neglect considerations of communication and load balancing, and focus on deterministic and centralized approaches [32] which usually rely on some form of task availability limitations (e.g. only one agent can win a task auction). *Ongoing* tasks without task supply limits are seldom discussed in existing task allocation literature.

StimHab, commonly referred to as response threshold with reinforcement [36][1], is a probabilistic biologically-inspired decentralized approach which has been previously applied to *ongoing* task

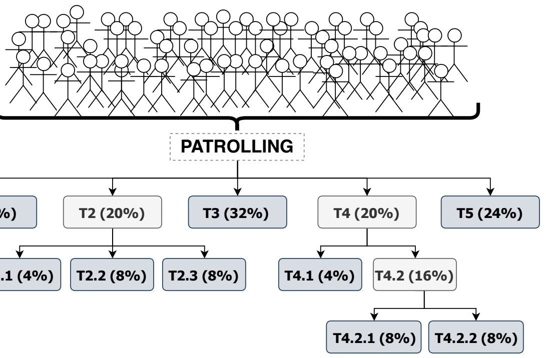


Fig. 1: Hierarchical Patrolling Domain
Nested areas T1-T5 and their sub-areas have patrolling demands (shown as percentages of total work-hours available per step) that change over time. Non-communicating agents self-allocate to the areas based on the areas' stimuli and agents' own evolving habituation thresholds for each area. If a chosen area has sub-areas (shown in light gray), selection repeats among its sub-areas.

allocation [18, 36, 40]. Biologically-inspired techniques for emergent task allocation show comparatively high scalability, adaptability, and resource utilization, as well as low complexity and communication requirements [41]. Decentralized multiagent approaches offer increased robustness and scalability, as they are not dependent on the availability of a central element [2, 37]. StimHab employs task *stimuli* and action-reinforced task *habituation thresholds* to calculate agents' action probabilities, which increase for tasks with higher stimuli and lower habituation thresholds [36]. StimHab agents do not communicate, nor are they aware of each other's capabilities, preferences, circumstances, or even existence, making the approach highly scalable. Agents sense current task stimuli, which change over time based on the system's performance on each task. Agents also maintain individual adaptable habituation thresholds, indicating their preferences for each task. An agent's probability to act on a task increases with given a higher stimulus and a lower habituation threshold. An agent increases its threshold for the chosen task and increases its thresholds for the other tasks, becoming more likely to repeatedly select the same action, leading to emergent specialization over time. Here, *specialization* refers to an agent's preference and the resulting increased propensity to act on some task over the alternatives. Emergent cooperation among decentralized agents is characterized by adaptable behavior, beneficial to dynamic applications, such as patrolling [2, 32]. Specialized agents lead to decreased interference and task switching, as well as to increased performance [1, 4, 16, 20, 26, 27, 31, 34].

Within StimHab, the basis of agents' habituation thresholds and current system needs are domain-specific. In addition to action-reinforced habituation, agents' thresholds can reflect experience,

[1]Other approaches exist where agents respond based on adapted thresholds, such as when agents act deterministically when a stimulus exceeds an agent's threshold [21]. To avoid ambiguity, we uniquely refer to the model defined in [36] as StimHab, referencing its use of stimuli and habituation thresholds to calculate agents' action probabilities.

current circumstances, or physical suitability. Task demands are unknown to the agents and system needs are inferred from performance levels on each task (see section 4 for details). Performance can be relayed to the agents by surveillance cameras, via a central informer transmitting global state of performance, or even through agents individually observing storage levels, lengths of request or production queues, counts of encounters of each task type, etc. As the cost of monitoring performance is less affected (if at all) by increasing numbers of agents, StimHab scales better than approaches relying on explicit agent coordination.

In this work, we further test the capabilities of StimHab by extending it to handle hierarchical sets of *ongoing* tasks by a team of 1000 agents, without relying on any additional methods of coordination. StimHab is applied to a high-level patrolling domain composed of nested areas, each with dynamic patrolling demands. We assess (1) the performance for each area over time, measured by how closely agents match the established area deployment requirements; (2) how well are the agents able to specialize, as measured by the amount of area switches (i.e. task switches) over time, and (3) how well agents reallocate when area-demands change. Our tests show that StimHab allows agents to self-allocate proportionately, while promoting specialization and maintaining adaptability. Additional tests show that task allocation stability can be improved by small changes to the relation between stimulus and habituation.

2 RELATED WORK

There is little research on *ongoing* tasks allocation with no communication. Below we review some existing approaches to decentralized task allocation and discuss their applicability to *ongoing* tasks.

Task allocation often relies on task supply limitations and information about the choices of others. Discrete tasks are commonly presented one at a time (e.g. bidding on truck-painting jobs [7]). When multiple tasks are available in unlimited quantities, agents must decide which task is needed more and for which is the agent better suited. Not having to recruit or confer with others can allow for faster responses to system changes, as well as for improved scalability, but having no communication nor limits on task availability does preclude the use of approaches employing auctions [24, 30, 42], token-passing [12, 22], or inter-agent recruitment [9–11, 39].

Ongoing task allocation without supply limitations has been addressed by probabilistic state transitions and by StimHab. For proportionate agent deployment to multiple locations (e.g. multi-area surveillance), locations/tasks can be assigned probabilities for agents to transition from one task to another [3, 14, 15]. Relying on global transition-probabilities does not promote specialization, while the need for explicitly defined transition values complicates adaptability in dynamic environments. StimHab can address generalized task allocation with no communication for a linear set of *ongoing* tasks, promoting specialization to reduce wasteful task switching [40]. StimHab may struggle during respecialization in dynamic domains, but performance can be improved by strategically resetting agent specializations [18]. While many other applications of StimHab exist, they commonly rely on limited task supply, having agents compete for each "job" (e.g. RoboCup Rescue [8], factory job assignments [5, 7, 13, 28, 29]; threshold dependent competition [25]; and token-passing for UAV surveillance [35]). To our knowledge, *ongoing* hierarchical tasks are not addressed in the literature.

3 DOMAIN: HIERARCHICAL DEPLOYMENT

We apply StimHab to a high-level patrolling domain, with multiple hierarchically-defined areas to be patrolled by a large group of non-communicating agents. Below we discuss why hierarchical task definitions are interesting and how patrolling fits multiagent task allocation for hierarchically-defined *ongoing* tasks.

A hierarchical domain can be seen as a layered version of multiple sets of linear tasks, to which StimHab has been successfully applied [18]. This layering groups tasks under parent tasks, where the amount of work needed for the parent equals the sum of the amounts of work needed by the siblings; i.e. if insufficient agents allocate themselves to a parent task, then there will be insufficient allocations for its children tasks. Thus, choosing among parent tasks obscures the individual needs of the children tasks, which could hinder the agents' ability to task allocate effectively. However, selecting among all the tasks in a flattened domain does not scale well to domains with many tasks. When selection is layered, large sets of tasks can be eliminated by deciding against a single parent task, resulting in better scaling. Additionally, a hierarchical task breakdown into sub-domains allows agents to self-allocate at higher levels, while lower levels can employ alternative task allocation methods, potentially better suited for these sub-domains. For example, while explicit scheduling may be prohibitive in very large systems as a whole, it may be the optimal choice for smaller sets of tasks that require more precise task allocation or more explicit cooperation. Hierarchically-defined StimHab would allow a subset of agents to individually self-allocate to the sub-domain of interest, and then switch to an alternative multiagent control method. Even if StimHab is used at all levels, other parameters can be varied per level, such as how often agents reconsider their current task choice, given that different sub-domains may be more or less dynamic, have higher or lower task switching costs, etc. A final benefit of hierarchical domain definition is that it can allow for a more intuitive domain breakdown, potentially simplifying system design.

Multi-area patrolling fits multiagent allocation for *ongoing* tasks: multiple locations in constant need of patrolling, a natural hierarchical breakdown of the overall space to be patrolled, patrolling demands per area that vary over time, and performance that benefits from each agent specializing on any one area. Decentralized approaches to patrolling usually involve some form of communication among the agents. Patrolling approaches that rely solely on marking patrolling frequency, such as using a pheromone diffusion model [6], do not promote specialization, which has been shown to be beneficial for patrolling [1]. Hierarchical patrolling represents an intuitive breakdown of a large area to be patrolled (e.g. a school campus subdivided into quadrants, each quadrant with multiple buildings, each building with multiple floors). Each sub-area represents a subtask of the larger area (or task) that encompasses it. Specializing on patrolling a single area diminishes the time agents spend traveling between areas, thus increasing patrolling efficiency.

To minimize the dependence of our results on domain-specific area adjacency and traveling costs, we model patrollable areas as an abstract set of *ongoing* tasks. Patrolling performance then depends on whether agents self-deploy proportionately to demands, using stimuli for indirect coordination. The patrolling performance of each area can be defined as the ratio of [current patrolling units in the area] to [the desired patrolling units in the area], where the desired number can depend on expected targets within the area, scheduled events, etc. The actual performance calculations is highly domain specific, but for physical areas it can be setup as counters of patrolling units vs. targets entering and exiting an area. Agents need no direct awareness of where the other agents are currently patrolling, only whether an area is currently sufficiently patrolled.

4 PROBABILISTIC ACTION USING STIMHAB

We propose an extension for an existing biologically-inspired approach for decentralized task allocation [36], which we term StimHab (for brevity and to distinguish it from other threshold-response methods). First, we review how stimuli and habituation thresholds are defined, updated, and used within StimHab to calculate the agents' action probabilities, as well as how the resulting actions lead to agents' specializations. We then propose the changes needed to extend StimHab's applicability to hierarchical task domains.

4.1 Global Task Stimuli

Agents perceive system needs through globally observable stimuli (e.g. dimensions of a fire to be put out [17] or task performance [18]) and use them as a fitness measure for decentralized coordination.

StimHab stimuli do not directly indicate task demands, instead being a sort of gas pedal to incite agents to act more or less on a given task. Knowing the total number or ratio of agents needed by a task per some time unit does not help an agent decide whether to take on that task. Instead, agents can decide whether their services are needed based on how well the task is being handled, i.e. task performance. Using inverse performance values directly as stimuli, however, results in an environment that is too unstable for specialization. No activity on a task corresponds to maximal stimulus (1.0), while the correct amount of activity corresponds minimal stimulus (0.0). Achieving the correct work distribution leads to no stimulus, a sharp drop in activity, and a subsequent drop in performance, leading to an increasing stimulus, an increase in activity back to previous levels, leading to zero stimulus again, etc. This oscillating stimulus precludes agents from stabilizing their task assignments.

Under StimHab, task performance is used to update stimuli. Ideal 100% performance corresponds to having the exact number of agents on a task to match the task's demand, leading to no change in stimulus and allowing agents to continue acting as they are. Excess work decreases stimulus, thus decreasing activity on the task; insufficient work increases stimulus, thus increasing activity. Task t's stimulus s_t is updated based on a per-step expenditure (i.e. show much stimuli increase in absence of agent action) and the team's per-step performance (i.e. ratio of achieved work to needed work):

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

up to a $\min(s_t) = 0.0$ and $\max(s_t) = 1.0$. If agents perform sufficient work to offset the per-step expenditure, s_t stops changing, indicating that agents are doing exactly enough work to match the task's demand. Expenditure Δs_t increases 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 each step, given no agent work on task t, s_t increases by 1.0/(number of steps in cycle).

4.2 Individual Task Habituation Thresholds

Each agent maintains its own habituation threshold for every task. These are often referred to as "response thresholds", which can be misleading when agents do not respond based solely on these thresholds. Thus, we place focus on the fact that these thresholds represent how habituated an agent is toward performing each task.

When agent a acts on a task t, its threshold for that task $\theta_{a,t}$ is reduced, while thresholds for the other tasks are increased. Lower thresholds increase action probability (see section 4.3), causing agents to become more likely to act on the same task in the future, leading to specialization over time. As agent a repeatedly acts on task t, $\theta_{a,t}$ tends to 0.0, while all other $\theta_{a,t'\neq t}$ tend to 1.0 according to the following threshold reinforcement rules:

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

with θ restricted to the range [0.0, 1.0]. Affinity and aversion rates dictate how fast agents habituate based on their actions.

4.3 Action Selection

StimHab achieves task allocation through probabilistic actions based on global task stimuli and agents' individual habituations [36]. An increase in a task's stimulus indicates that more work is needed on that task, while a lower agent's task threshold indicates that agent has developed an affinity for this task, inciting it to act at lower stimuli than agents with higher thresholds for that task.

Every time step, each agent a calculates the probability $P_{a,t}$ to act on every task t, by combining the task's stimulus s_t with its own affinity for that task $\theta_{a,t}$:

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

The redefinition at $s_t = \theta_{a,t} = 0$ avoids division by zero, while leading to a 50%/50% chance to select the task or not, which falls precisely between the adjacent values $P_{a,t} = 1.0$ for $(s_t > 0.0, \theta_{a,t} = 0.0)$ and $P_{a,t} = 0.0$ for $(s_t = 0.0, \theta_{a,t} > 0.0)$, while also matching the values along the rest of the diagonal where $s_t = \theta_{a,t}$, as every $[s_t = \theta_{a,t}] > 0.0$ results in $P_{a,t} = (s_t^2/(s_t^2 + s_t^2) = 0.5$.

Having multiple *ongoing* tasks available at every step, agents must choose whether to act on any one of them. In prior work, we extend StimHab to allow agents to choose among multiple tasks [40], as the original definition presents agents with one task at a time [36]. Agents consider tasks in descending order of $P_{a,t}$, which promotes proportionate task allocation and specialization. The resulting order may differ across agents, as agents calculate $P_{a,t}$ using their own thresholds. When considering each task, if a random value $\in [0.0, 1.0]$ is below $P_{a,t}$, the agent acts on that task.

Otherwise, the task with the next highest $P_{a,t}$ is considered. If no task triggers action, the agent idles until the next time step.

4.4 Hierarchical Task Assignment

We extend StimHab to hierarchical domains composed of tasks and subtasks. To this end, task selection and the reinforcement of task habituation thresholds both require newly recursive definitions.

Agents will choose from among a subset of tasks at each level, recursively diving deeper in the hierarchy until selecting a task with no subtasks, or defaulting to idle. Under StimHab with a linearly defined set of tasks, once an action is selected, the agent can act. For a hierarchical set of tasks, a chosen task must then be checked for subtasks: if the task has subtasks, the agent will repeat the task selection process, this time choosing among the subtasks of the originally selected task; if the task has no subtasks, the agent will act on the chosen task. At any level of the hierarchy, if the probabilities $P_{a,t}$ for the tasks under consideration do not trigger any one task to be selected, the agent will idle. Thus, action selection always ends at a "leaf" task (either an actual task or at idling, which is a type of "leaf" task implicitly available at every level).

Selecting a task must trigger habituation threshold updates for that task and its sibling tasks, i.e. we are choosing to specialize on that task and against the alternatives. Under StimHab with a linearly defined set of tasks, acting on a task causes its threshold to reduce, while the thresholds for all other tasks increase. For a hierarchical set of tasks, only the tasks at the same level as the chosen task (i.e. sibling tasks) will have their thresholds increased. Additionally, since choosing a task means its parent task (if any) was also chosen along the path down the hierarchy, updates must then propagate upward. This means the parent task's threshold will also be reduced, while the thresholds of the parent task's siblings will be increased; the update will then be moved further up to the grandparent task and so on until the top level is reached.

5 EXPERIMENTAL SETUP

To verify the benefits of StimHab for decentralized allocation with specialization, we compare agents behavior under StimHab to that resulting from a direct averaging of the two signals for our objectives, i.e. stimuli and habituation thresholds. While agents could be made to follow stimulus or thresholds alone, neither would achieve the dual aim of maximizing performance while minimizing task switching. Below we present the specifics of the tested approaches, the hierarchical set of tasks used, and the actual demand values used for the different tasks throughout the simulations.

We test two approaches that combine stimuli and habituations to maximize performance and minimize task switching: StimHab (adapted to hierarchical domains) and Average(s,θ). Average(s,θ) uses the same definitions of stimuli and thresholds, while its probability to act is the average of the two values, accounting for the inverse relationship of thresholds and action probability:

Average(s,θ) $P_{a,t} = \frac{s_t + (1-\theta_{a,t})}{2}$ vs. **StimHab** $P_{a,t} = \frac{s_t^2}{s_t^2 + \theta_{a,t}^2}$,

where a is the agent, t is the task, s_t is the task's stimulus, and $\theta_{a,t}$ is the agent's habituation threshold for task a. For both approaches, all s_t are initialized to 0.0 and all $\theta_{a,t}$ are initialized to uniformly random values $\in [0.0, 1.0]$.

We test two versions of each approach: one **with resets** of thresholds back to uniformly random values when demands change and one **without resets**, requiring agents to respecialize from the previously developed specializations $\theta_{a,t}$. Adaptability of decentralized solutions often depends on population diversity, which can be lost after the initial adaptation, complicating re-adaptation [18, 23, 33]. Partial or full resets of conditioning can be used to improve performance: forgetting older observations in adversarial decision making [38], forgetting older training environments in Case-Based Reasoning robotic navigation [19], and resetting habituations in a task allocation using StimHab [18]. Consequently, we test four approaches: StimHab with threshold resets, StimHab without resets, Average(s,θ) with resets, and Average(s,θ) without resets.

Our patrolling scenario has 12 hierarchically arranged areas, i.e. 12 tasks (see hierarchy in figure 1). Agents are not directly aware of task demands, instead sensing globally monitored task performance values. Performance on task t is the ratio between [work achieved] and [work needed], i.e. how many agents worked on this task during this step vs. how many agents we needed to work on this task. Ideal behavior corresponds to continuously having the correct number of agents working on each task. To establish an optimal baseline, we ensure that demands at each level in the hierarchy always add up to 100% of the available agents (i.e. demands for T1-T5 always add up to 100% of agents; if demand for T2=x%, demands for its subtasks T2.1, T2.2, T2.3 add up to x%). Thus, any misallocation will cause some tasks to fall below 100% performance. The team consists of 1000 non-communicating, homogeneous agents with uniformly random initial habituation thresholds $\in [0.0, 1.0]$. Each simulation run lasts for 500 cycles, i.e. 50000 steps. Patrolling demands change every 50 cycles (5000 steps or task-choosing decisions by each agent), for a total of 10 times per simulation.

CYCLE	T1	T2	T2.1	T2.2	T2.3	T3	T4	T4.1	T4.2	T4.2.1	T4.2.2	T5
0-49	4%	20%	4%	8%	8%	32%	20%	4%	16%	8%	8%	24%
50-99	4%	16%	8%	4%	4%	24%	32%	8%	24%	12%	12%	24%
100-149	16%	40%	4%	20%	16%	4%	12%	4%	8%	4%	4%	28%
150-199	24%	12%	4%	4%	4%	28%	16%	4%	12%	4%	8%	20%
200-249	4%	32%	12%	8%	12%	16%	36%	12%	24%	8%	16%	12%
250-299	36%	16%	4%	4%	8%	4%	20%	8%	12%	4%	8%	24%
300-349	16%	16%	4%	4%	8%	8%	12%	4%	8%	4%	4%	48%
350-399	56%	12%	4%	4%	4%	4%	20%	12%	8%	4%	4%	8%
400-449	12%	16%	8%	4%	4%	4%	52%	36%	16%	12%	4%	16%
450-499	32%	24%	8%	12%	4%	20%	12%	4%	8%	4%	4%	12%

Table 1: Patrolling demands per area and simulation period
Demands change every 50 cycles, i.e. 10 times over a 500-cycle simulation. For example, on cycle 100, the domain changes from needing 4% of all patrolling to be on area T1 to needing 16%. T2 changes from 16% to 40%: 4% on T2.1, 20% on T2.2, 15% on T3.3.

6 RESULTS AND DISCUSSION

Initially, we compare StimHab to a direct averaging of the stimulus and habituations. Results show that agents can approach the expected task allocation at the cycle level, but oscillations are observed at the step level after an initial adaptation period. These findings lead to an additional set of tests, considering an alternative version of StimHab, StimHabAlt, intended to help agents respecialize as conditions change. Below we provide the two sets of comparisons: (1) StimHab vs. Average(s,θ) and (2) StimHab vs. StimHabAlt.

6.1 StimHab vs Average(s, θ)

To assess the ability of a decentralized team to self-allocated to a set of *ongoing* hierarchically-defined tasks, we compare the abilities of StimHab and Average(s, θ) approaches to deploy agents to a set of areas with dynamic patrolling demands. We look at task allocation over sample runs to clearly observe the behavior, then at average runs to assess the stability of the observed behavior. We compare the approaches' ability to provide the expected number of agents to each area as demand change, as well as to allow for specialization by reducing the amount of task switching over time.

Figures 2-5 depict per-area performances for each cycle of a 500-cycle representative sample run. The x-axis shows cycles and the y-axis shows $performance_t = (work\ achieved_t / work\ needed_t)$. Each line represents performance for an area. Ideal task allocation corresponds to 100% performance in every area, as we do not want too many nor too few agents allocated to an area. Patrolling demands for each area change every 50 cycles, causing spiking from 100% performance. Under StimHab with resets (fig. 3) and Average(s, θ) with resets (fig. 5) performances approach 100% performances faster than their versions without resets, corroborating prior research that indicates that threshold reinforcement is more effective starting from random thresholds (i.e. specialization) than starting from previously reinforced thresholds (i.e. respecialization) [18].

Figures 6-9 show an averaged view of the same sample runs. The x-axis represents an average n-th cycle of every 50 cycles and the y-axis displays the average percentage deviation from the desired allocation. Deviations are used in place of performance values to ensure that values above and below 100% performance do not cancel out: $deviation_t = |100 - performance_t|$. Thus, ideal performance corresponds to 0% deviation. We see that StimHab with resets (fig. 7) and Average(s, θ) with resets (fig. 9) both reach near 0% deviation for all areas over time, while for versions without resets some areas remain at as high as 25% (fig. 6) and 20% (fig. 8) deviation.

Figure 10 plots the task switches for each of the four approaches, averaged over the ten changes in demands. The x-axis depicts the average n-th cycle of a 50-cycle period (i.e. between changes in patrolling demands) and the y-axis shows the average number of times agents switch tasks during that cycle. Given 1000 agents and 100-steps per cycle, a maximum number of switches per cycle is 100000. We see here that, although StimHab and Average(s, θ) (both with resetting) had similar ability to fulfill average patrolling demands, StimHab with resetting is able to reduce task switches to approximately 2000 per cycle, amounting to 2 switches per agent per 100 task choosing decisions (one decision per step, thus 100 per cycle). StimHab without resetting comes in second, only reducing task switching to approximately 18000 times per cycle, while Average(s, θ) with and without resetting result in twice as much task switching. Thus, we see that StimHab greatly reduces task switching, though never quite eliminates it.

To further investigate the causes and repercussions of these task switches, we graph task allocation performance at each step of a representative sample run for the approaches with lowest overall task switching: StimHab with resets in fig. 11 and Average(s, θ) with resets in fig. 12. To ensure visibility, only the first 5000 steps (of the total 50000) are shown, but the discussed behavior repeats itself every 5000 steps. We see that although task allocations averaged for each cycle were nearing the ideal 100%, at the step level there is considerable oscillation in both approaches. It is nevertheless clear that while Average(s, θ) performance oscillates continuously throughout the entire 5000-step period (notice the dense spiking in fig. 12), StimHab performance (fig. 11) initially approaches the desired task allocation and then begins to oscillate. The spikes here are sparser than under Average(s, θ), but have higher amplitude, indicating intermittent large shifts in agent assignment.

Looking closer at the simulation data, we see that StimHab's performance begins to oscillate once tasks with too many agents (performance line above 100%) reach $s_t = 0$. By this point, agents have strongly developed habituations ($\theta_{a,t} = 1.0$ one of the tasks), causing $P_{a,t}$ to drop from 1.0 to 0.5, and leading to a decrease of activity on that task and an increase on the others. A sharp decrease in activity brings s_t down, restarting the cycle. This oscillation hinders fine-tuning hierarchical specializations and is precisely why resetting is beneficial for threshold adaptation: fully habituated agents have trouble choosing what is needed over what they now prefer [18]. We hypothesize that respecialization can be improved by altering probability calculation in favor of tasks where $s_t = 1.0$.

6.2 StimHabAlt: altering StimHab behavior near the point ($s_t = 1.0, \theta_{a,t} = 1.0$)

To facilitate respecialization, the probability formula can be adjusted to favor ($s = 1.0, \theta_{a,t} = 1.0$), i.e. to favor the task that agents have habituated against, but which has now reached maximum stimulus due to continuous insufficient activity. Below we present our initial investigation of altering StimHab $P_{a,t}$ to improve agents' ability to find proportionate and stable task assignments. StimHab with resets is compared against StimHabAlt with and without resets.

As a preliminary test of this hypothesis, we update StimHab's $P_{a,t}$ formula. To favor under-performing tasks without disturbing the system's ability to specialize, we test a slight increase of $P_{a,t}$ at $[s_t = \theta_{a,t}] = 1.0$. We balance this increase by a simultaneous decrease in activity for intermediate stimuli, to maintain a similar overall distribution of possible $P_{a,t}$. In depth analysis is needed to establish a principled definition of the weights and powers to be employed. For this initial test, we select values that approximate the hypothesized distribution of $P_{a,t}$ needed to favor highly under-performing tasks, namely: a weight of 60% for s and 40% for $\theta_{a,t}$, as well as a power of 3 for s and of 2 for θ. We term the resulting probability formulation as **StimHabAlt**: $P_{a,t} = \frac{(0.6s)^3}{(0.6s)^3 + (0.4\theta)^2}$

Probability values for StimHab and StimHabAlt are shown in fig. 14a and fig. 14b, respectively. We hypothesize that this alteration will not only reduce oscillations at the step-level, but may also help agents respecialize when demands change even if their specializations are not reset to random values beforehand.

Figures 15 and 16 depict per-area performances for representative sample runs of StimHabAlt without and with threshold resetting, respectively. Although resetting still proves beneficial as seen from reduced spiking in fig. 16, both methods approach 100% after each time demands are reset (every 50-th cycle), indicating an ability to respecialize, which supports our hypothesis of the benefit of the altered $P_{a,t}$ mapping in StimHabAlt. Figures 17 and 18 show average adaptation behavior over an average 50-cycle period, further

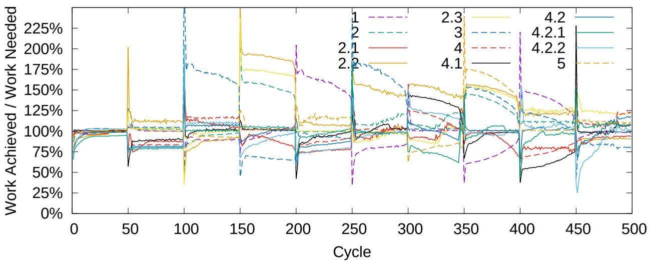


Fig. 2: Sample run 500 cycles: StimHab without resets

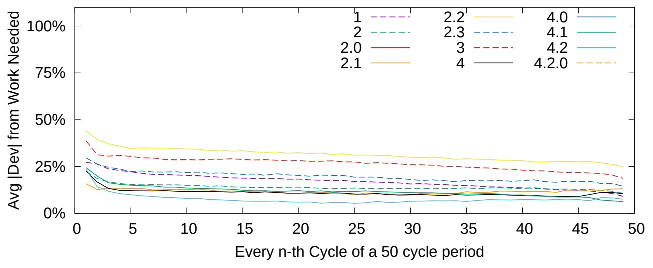


Fig. 6: Sample run 50-cycle avg: StimHab without resets

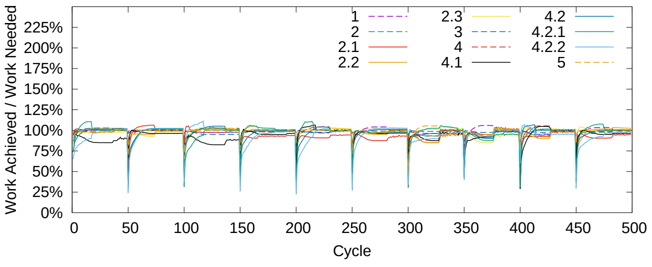


Fig. 3: Sample run 500 cycles: StimHab with resets

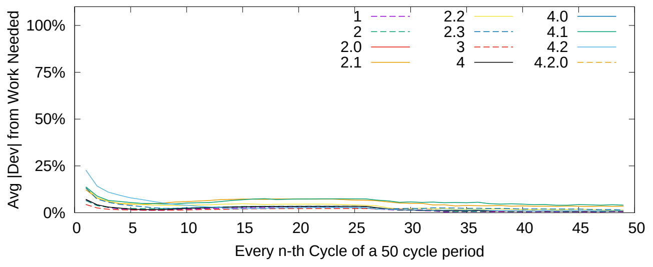


Fig. 7: Sample run 50-cycle avg: StimHab with resets

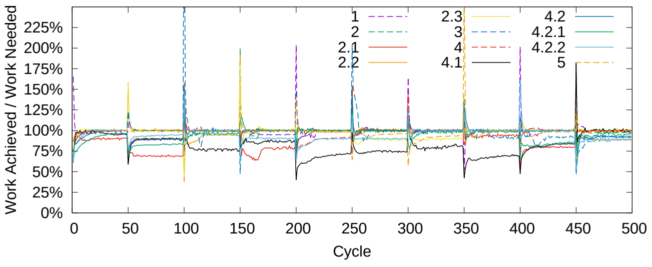


Fig. 4: Sample run 500 cycles: Average(s, θ) without resets

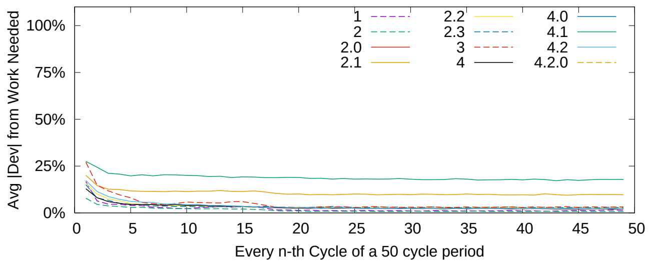


Fig. 8: Sample run 50-cycle avg: Average(s, θ) without resets

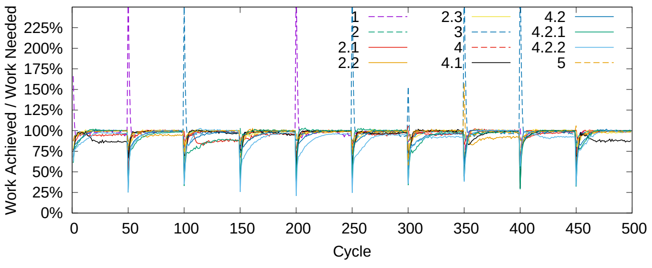


Fig. 5: Sample run 500 cycles: Average(s, θ) with resets

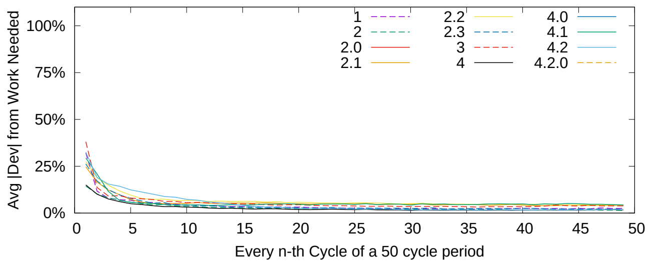


Fig. 9: Sample run 50-cycle avg: Average(s, θ) with resets

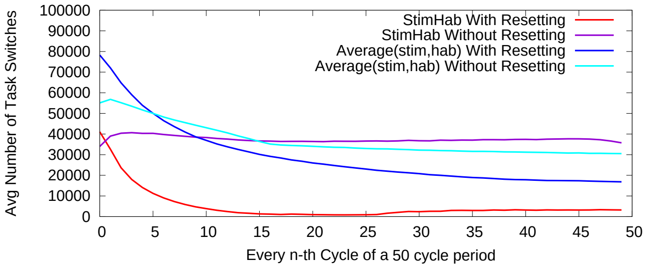


Fig. 10: Sample run: task switching counts
Methods with threshold resetting result in fewer task switches.

supporting that StimHabAlt does not require threshold resetting to allow for respecialization. Both versions of StimHabAlt produce task performances comparable to StimHab with resetting (fig. 3,7).

As StimHab with resetting and StimHabAlt with and without resetting result in similar task performances, in fig. 19 we compare their average task switching and find that all three are similarly able to reduce task switching over time, indicating agents' ability to specialize. To further analyze StimHabAlt, we assess its performance per step. Fig. 13 depicts performance for the first 5000 steps of a representative sample run, though the observed behavior repeats every 5000 steps, with small variations. We see that the number and amplitude of performance oscillations are reduced as compared to StimHab (fig. 11), though not fully eliminated.

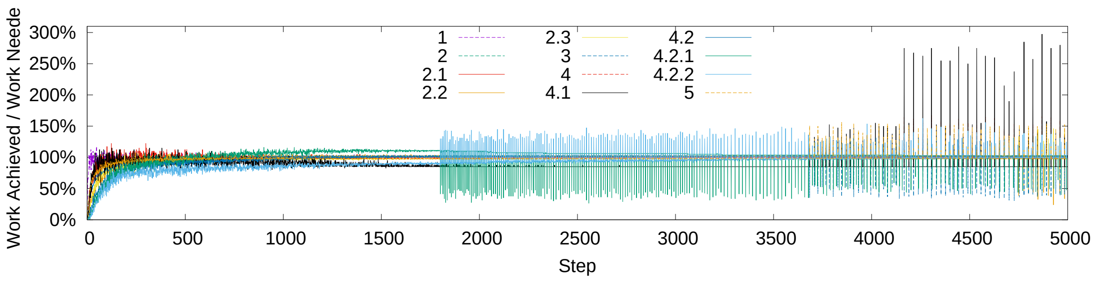


Fig. 11: Sample run first 5000 steps (first 50 cycles): StimHab with resets
Numbered lines correspond to performance on each patrollable area, with 100% being ideal.

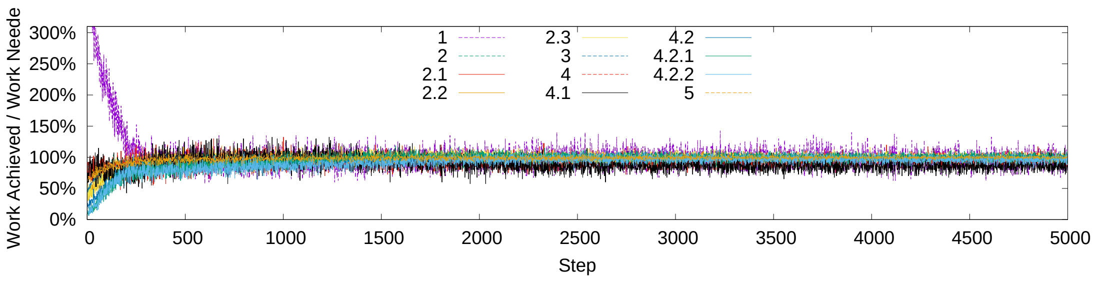


Fig. 12: Sample run first 5000 steps (first 50 cycles): Average(s, θ) with resets
Numbered lines correspond to performance on each patrollable area, with 100% being ideal.

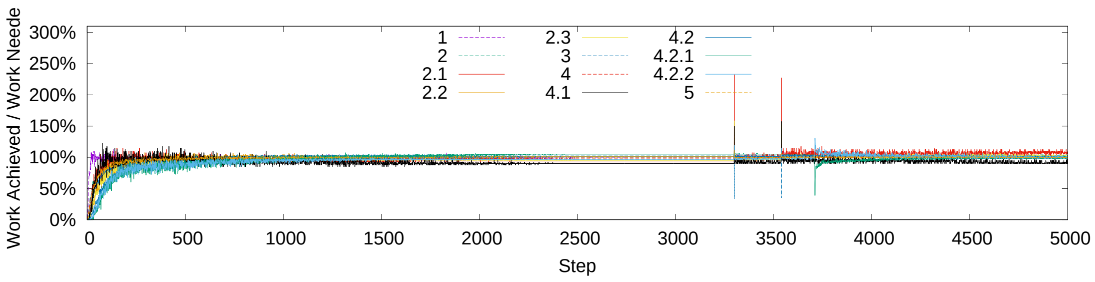
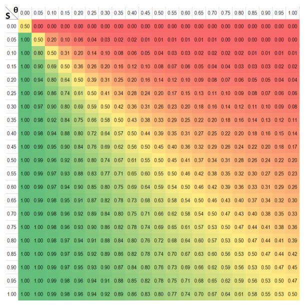
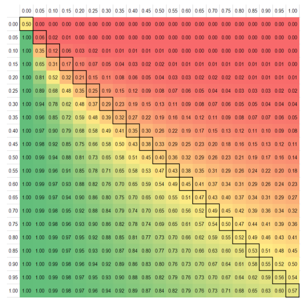


Fig. 13: Sample run first 5000 steps (first 50 cycles): StimHab-Alt with resets
Numbered lines correspond to performance on each patrollable area, with 100% being ideal.



s \ θ	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.05	1.00	0.50	0.20	0.10	0.06	0.04	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.10	1.00	0.80	0.50	0.31	0.20	0.14	0.10	0.08	0.06	0.05	0.04	0.03	0.03	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01
0.15	1.00	0.90	0.69	0.50	0.36	0.26	0.20	0.16	0.12	0.10	0.08	0.07	0.06	0.05	0.04	0.04	0.03	0.03	0.03	0.02	0.02
0.20	1.00	0.94	0.80	0.64	0.50	0.39	0.31	0.25	0.20	0.16	0.14	0.12	0.10	0.09	0.08	0.07	0.06	0.05	0.05	0.04	0.04
0.25	1.00	0.96	0.86	0.74	0.61	0.50	0.41	0.34	0.28	0.24	0.20	0.17	0.15	0.13	0.11	0.10	0.09	0.08	0.07	0.06	0.06
0.30	1.00	0.97	0.90	0.80	0.69	0.59	0.50	0.42	0.36	0.31	0.26	0.23	0.20	0.18	0.16	0.14	0.12	0.11	0.10	0.09	0.08
0.35	1.00	0.98	0.92	0.84	0.75	0.66	0.58	0.50	0.43	0.38	0.33	0.29	0.25	0.22	0.20	0.18	0.16	0.14	0.13	0.12	0.11
0.40	1.00	0.98	0.94	0.88	0.80	0.72	0.64	0.57	0.50	0.44	0.39	0.35	0.31	0.27	0.25	0.22	0.20	0.18	0.16	0.15	0.14
0.45	1.00	0.99	0.95	0.90	0.84	0.76	0.69	0.62	0.56	0.50	0.45	0.40	0.36	0.32	0.29	0.26	0.24	0.22	0.20	0.18	0.17
0.50	1.00	0.99	0.96	0.92	0.86	0.80	0.74	0.67	0.61	0.55	0.50	0.45	0.41	0.37	0.34	0.31	0.28	0.26	0.24	0.22	0.20
0.55	1.00	0.99	0.97	0.93	0.88	0.83	0.77	0.71	0.65	0.60	0.55	0.50	0.46	0.42	0.38	0.35	0.32	0.30	0.27	0.25	0.23
0.60	1.00	0.99	0.97	0.94	0.90	0.85	0.80	0.75	0.69	0.64	0.59	0.54	0.50	0.46	0.42	0.39	0.36	0.33	0.31	0.29	0.26
0.65	1.00	0.99	0.98	0.95	0.91	0.87	0.82	0.78	0.73	0.68	0.63	0.58	0.54	0.50	0.46	0.43	0.40	0.37	0.34	0.32	0.30
0.70	1.00	0.99	0.98	0.96	0.92	0.89	0.84	0.80	0.75	0.71	0.66	0.62	0.58	0.54	0.50	0.47	0.43	0.40	0.38	0.35	0.33
0.75	1.00	1.00	0.98	0.96	0.93	0.90	0.86	0.82	0.78	0.74	0.69	0.65	0.61	0.57	0.53	0.50	0.47	0.44	0.41	0.38	0.36
0.80	1.00	1.00	0.98	0.97	0.94	0.91	0.88	0.84	0.80	0.76	0.72	0.68	0.64	0.60	0.57	0.53	0.50	0.47	0.44	0.41	0.39
0.85	1.00	1.00	0.99	0.97	0.95	0.92	0.89	0.86	0.82	0.78	0.74	0.70	0.67	0.63	0.60	0.56	0.53	0.50	0.47	0.44	0.42
0.90	1.00	1.00	0.99	0.97	0.95	0.93	0.90	0.87	0.84	0.80	0.76	0.73	0.69	0.66	0.62	0.59	0.56	0.53	0.50	0.47	0.45
0.95	1.00	1.00	0.99	0.98	0.96	0.94	0.91	0.88	0.85	0.82	0.78	0.75	0.71	0.68	0.65	0.62	0.59	0.56	0.53	0.50	0.47
1.00	1.00	1.00	0.99	0.98	0.96	0.94	0.92	0.89	0.86	0.83	0.80	0.77	0.74	0.70	0.67	0.64	0.61	0.58	0.55	0.53	0.50

(a) StimHab: $P = \frac{s^2}{s^2+\theta^2}$



	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.05	1.00	0.06	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.10	1.00	0.35	0.12	0.06	0.03	0.02	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.15	1.00	0.65	0.31	0.17	0.10	0.07	0.05	0.04	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
0.20	1.00	0.81	0.52	0.32	0.21	0.15	0.11	0.08	0.06	0.05	0.04	0.03	0.03	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01
0.25	1.00	0.89	0.68	0.48	0.35	0.25	0.19	0.15	0.12	0.09	0.08	0.07	0.06	0.05	0.04	0.04	0.03	0.03	0.03	0.02	0.02
0.30	1.00	0.94	0.78	0.62	0.48	0.37	0.29	0.23	0.19	0.15	0.13	0.11	0.09	0.08	0.07	0.06	0.05	0.05	0.04	0.04	0.04
0.35	1.00	0.96	0.85	0.72	0.59	0.48	0.39	0.32	0.27	0.22	0.19	0.16	0.14	0.12	0.11	0.09	0.08	0.07	0.07	0.06	0.05
0.40	1.00	0.97	0.90	0.79	0.68	0.58	0.49	0.41	0.35	0.30	0.26	0.22	0.19	0.17	0.15	0.13	0.12	0.11	0.10	0.09	0.08
0.45	1.00	0.98	0.92	0.85	0.75	0.66	0.58	0.50	0.43	0.38	0.33	0.29	0.25	0.23	0.20	0.18	0.16	0.15	0.13	0.12	0.11
0.50	1.00	0.99	0.94	0.88	0.81	0.73	0.65	0.58	0.51	0.45	0.40	0.36	0.32	0.29	0.26	0.23	0.21	0.19	0.17	0.16	0.14
0.55	1.00	0.99	0.96	0.91	0.85	0.78	0.71	0.65	0.58	0.53	0.47	0.43	0.38	0.35	0.31	0.29	0.26	0.24	0.22	0.20	0.18
0.60	1.00	0.99	0.97	0.93	0.88	0.82	0.76	0.70	0.65	0.59	0.54	0.49	0.45	0.41	0.37	0.34	0.31	0.29	0.26	0.24	0.23
0.65	1.00	0.99	0.97	0.94	0.90	0.86	0.80	0.75	0.70	0.65	0.60	0.55	0.51	0.47	0.43	0.40	0.37	0.34	0.31	0.29	0.27
0.70	1.00	0.99	0.98	0.95	0.92	0.88	0.84	0.79	0.74	0.70	0.65	0.60	0.56	0.52	0.49	0.45	0.42	0.39	0.36	0.34	0.32
0.75	1.00	1.00	0.98	0.96	0.93	0.90	0.86	0.82	0.78	0.74	0.69	0.65	0.61	0.57	0.54	0.50	0.47	0.44	0.41	0.39	0.36
0.80	1.00	1.00	0.99	0.97	0.95	0.92	0.88	0.85	0.81	0.77	0.73	0.70	0.66	0.62	0.59	0.55	0.52	0.49	0.46	0.43	0.41
0.85	1.00	1.00	0.99	0.97	0.95	0.93	0.90	0.87	0.84	0.80	0.77	0.73	0.70	0.66	0.63	0.60	0.56	0.53	0.51	0.48	0.45
0.90	1.00	1.00	0.99	0.98	0.96	0.94	0.92	0.89	0.86	0.83	0.80	0.76	0.73	0.70	0.67	0.64	0.61	0.58	0.55	0.52	0.50
0.95	1.00	1.00	0.99	0.98	0.97	0.95	0.93	0.90	0.88	0.85	0.82	0.79	0.76	0.73	0.70	0.67	0.64	0.62	0.59	0.56	0.54
1.00	1.00	1.00	0.99	0.98	0.97	0.96	0.94	0.92	0.89	0.87	0.84	0.82	0.79	0.76	0.73	0.71	0.68	0.65	0.63	0.60	0.57

(b) StimHab-Alt: $P = \frac{(0.6s)^3}{(0.6s)^3+(0.4\theta)^2}$

Fig. 14: All possible probability values
In StimHab (a), minimally needed tasks with high habituation $(s=\theta=0)$ and maximally needed tasks with low habituation $(s=\theta=1)$ both result in $P_{a,t}=0.5$, hindering respecialization. We propose favoring $(s=\theta=1)$ by shifting $P_{a,t}$ values, as shown in (b).

Behavioral stability is verified through 50 runs of StimHab with resets and StimHabAlt with resets. Average performance deviations and 95% confidence intervals are shown in fig. 20; average task switching is shown in fig. 21. StimHabAlt has slightly lower deviations and task switch counts, with considerably higher stability, seen in narrower shaded confidence intervals. Results suggest that while StimHab can produce proportionate decentralized allocation for hierarchical tasks with changing demands, task assignment stability can be improved further by altering the probabilistic relation between task stimuli and agents' habituation thresholds.

6.3 Conclusions and Future Work

We apply biologically-inspired decentralized task allocation [36] (StimHab) to a dynamic hierarchically-defined patrolling domain. Agents self-allocate proportionately to the dynamic patrolling needs of a set of nested areas using stimuli as a fitness measure for coordination. After initial adaptation, task switching in the system reduces to about 2% of the actions at every step, indicating highly

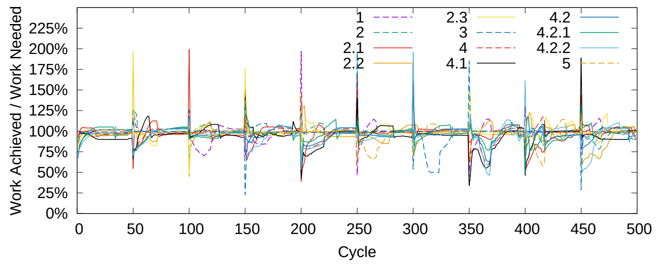


Fig. 15: Sample run (500 cycles): StimHab-Alt without resets

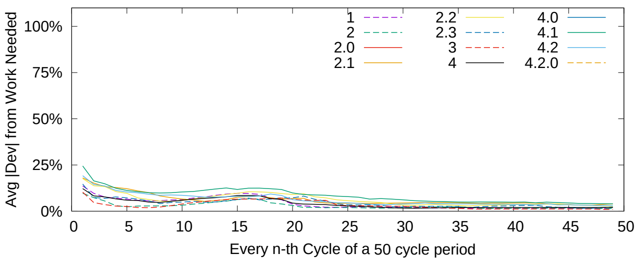


Fig. 17: Sample run 50-cycle avg: StimHab-Alt without resets

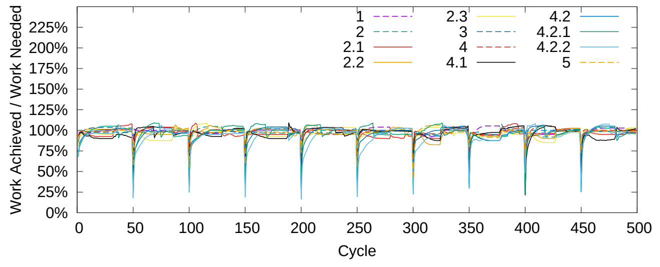


Fig. 16: Sample run (500 cycles): StimHab-Alt with resets

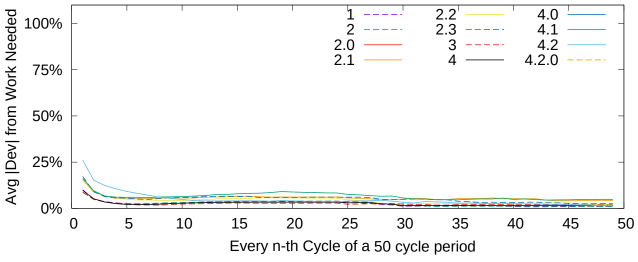


Fig. 18: Sample run 50-cycle avg: StimHab-Alt with resets

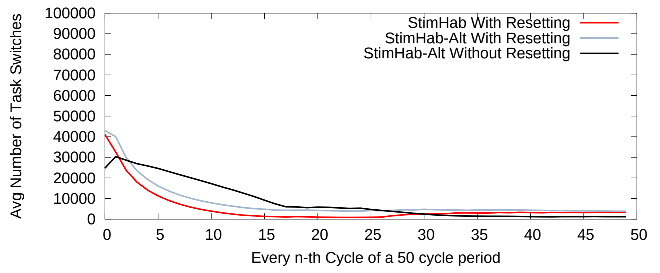


Fig. 19: Sample run 50-cycle task switching average: StimHab with resets vs StimHab-Alt with and without resets

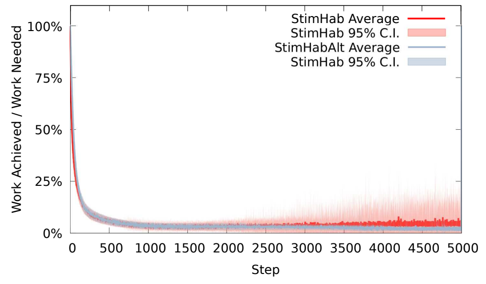


Fig. 20: 50-run performance deviation average: StimHab with resets vs. StimHabAlt with resets

specialized behavior, which has been proven beneficial in many multiagent domains. A team of 1000 agents self-allocate effectively without communicating, demonstrating StimHab's scalability. The applicability of the resulting task allocation and specialization extend to decentralized dynamic domains, hierarchical or not, that have continuous tasks available in unlimited quantities (e.g. resource gathering, maintenance, demand-based production...), many agents, and no inter-agent communication, potentially impossible or undesirable in some domains. We present some initial testing of potential parameter changes to further increase the stability of agents' task assignments and to improve overall performance.

Scheduling and communication-based approaches do not always scale well to many agents and dynamic environments. As agents are not directly adapting to changes in demand, but rather to changes in performance, the approach is suitable for a variety of dynamic environments, such as those with agent failure/replacement, or with environmental variations that can change how work translates into performance. Additionally, StimHab can be used to reduce overall micromanagement in dynamic multiagent systems. Agents' ability to self-allocate to hierarchical tasks allows StimHab to direct large groups of agents toward sub-domains, where the newly formed subgroups can subdivide further using StimHab, or switch to a more

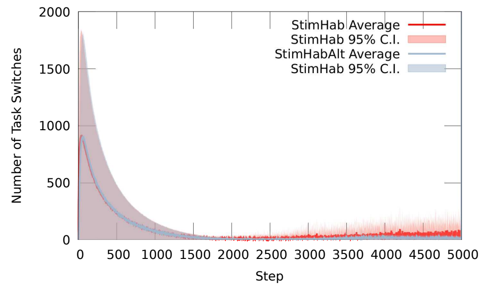


Fig. 21: 50-run task switching average: StimHab with resets vs. StimHabAlt with resets

controlled but less scalable approach, such as scheduling, auctions, or graph algorithms [2, 32], depending on subdomain specifics.

Future work includes a more in-depth assessment of the repercussions of the weight and power parameters in the StimHab probability formula. Other extensions include explicitly asynchronous agent actions resulting in asymmetrical information, imperfect stimulus information, subsets of agents with unvarying thresholds (i.e. preset capabilities), and mapping the hierarchical domain to a graph topology to evaluate the behavior on actual physical layouts.

REFERENCES

[1] Noa Agmon, Daniel Urieli, and Peter Stone. 2011. Multiagent Patrol Generalized to Complex Environmental Conditions. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI'11).*

[2] Alessandro Almeida, Geber Ramalho, Hugo Santana, Patrícia Tedesco, Talita Menezes, Vincent Corruble, and Yann Chevaleyre. 2004. Recent Advances on Multi-agent Patrolling. In *Advances in Artificial Intelligence – SBIA 2004*, Ana L. C. Bazzan and Sofiane Labidi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 474–483.

[3] S. Berman, A. Halasz, V. Kumar, and S. Pratt. 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.* 2318–2323.

[4] Adam Campbell and Annie S Wu. 2011. Multi-agent role allocation: issues, approaches, and multiple perspectives. *Autonomous agents and multi-agent systems* 22, 2 (2011), 317–355.

[5] Mike Campos, Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. 2000. Dynamic scheduling and division of labor in social insects. *Adaptive Behavior* 8, 2 (2000), 83–95.

[6] Hoang Nam Chu, Arnaud Glad, Olivier Simonin, Francois Sempe, Alexis Drogoul, and François Charpillet. 2007. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, Vol. 1. IEEE, 442–449.

[7] Vincent A. Cicirello and Stephen F. Smith. 2004. Wasp-like Agents for Distributed Factory Coordination. *Autonomous Agents and Multi-Agent Systems* 8, 3 (01 May 2004), 237–266.

[8] Daniela Scherer dos Santos and Ana L.C. Bazzan. 2012. Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. *Applied Soft Computing* 12, 8 (2012), 2123 – 2131.

[9] Fernando dos Santos and Ana LC Bazzan. 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.* ACM, 73–80.

[10] Fernando Dos Santos and Ana LC Bazzan. 2011. Towards efficient multiagent task allocation in the robocup rescue: a biologically-inspired approach. *Autonomous Agents and Multi-Agent Systems* 22, 3 (2011), 465–486.

[11] Frederick Ducatelle, Alexander Förster, Gianni A Di Caro, and Luca Maria Gambardella. 2009. New task allocation methods for robotic swarms. In *9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions.*

[12] Alessandro Farinelli, Luca Iocchi, Daniele Nardi, and Vittorio Amos Ziparo. 2006. Assignment of dynamically perceived tasks by token passing in multirobot systems. *Proc. IEEE* 94, 7 (2006), 1271–1288.

[13] Roberto Ghizzioli, Shervin Nouyan, Mauro Birattari, and Marco Dorigo. 2005. *An ant-based algorithm for the heterogeneous dynamic task allocation problem.* Technical Report TR/IRIDIA/2005-005. Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle (IRIDIA).

[14] Adám Halász, M Ani Hsieh, Spring Berman, and Vijay Kumar. 2007. Dynamic redistribution of a swarm of robots among multiple sites. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on.* IEEE, 2320–2325.

[15] M Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. 2008. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence* 2, 2-4 (2008), 121–141.

[16] M Ani Hsieh, Ádám Halász, Ekin Dogus Cubuk, Samuel Schoenholz, and Alcherio Martinoli. 2009. Specialization as an optimal strategy under varying external conditions. In *Robotics and Automation, ICRA'09. IEEE International Conference on.* 1941–1946.

[17] Anshul Kanakia, Behrouz Touri, and Nikolaus Correll. 2016. Modeling multi-robot task allocation with limited information as global game. *Swarm Intelligence* 10, 2 (2016), 147–160.

[18] Vera A. Kazakova and Annie S. Wu. 2018. Specialization vs. Re-Specialization: Effects of Hebbian Learning in a Dynamic Environment. In *Florida Artificial Intelligence Research Society Conference FLAIRS-31.*

[19] Zsolt Kira and Ronald C Arkin. 2004. Forgetting bad behavior: memory for case-based navigation. In *Intelligent Robots and Systems.(IROS). Proceedings, 2004 IEEE/RSJ International Conference on*, Vol. 4. 3145–3152.

[20] Ling Li, Alcherio Martinoli, and Yaser S Abu-Mostafa. 2002. Emergent specialization in swarm systems. In *International Conference on Intelligent Data Engineering and Automated Learning.* Springer, 261–266.

[21] Wenguo Liu, Alan FT Winfield, Jin Sa, Jie Chen, and Lihua Dou. 2007. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive behavior* 15, 3 (2007), 289–305.

[22] Hang Ma, Jiaoyang Li, TK Kumar, and Sven Koenig. 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, 837–845.

[23] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. 2017. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation* 33 (2017), 1–17.

[24] Mitchell McIntire, Ernesto Nunes, and Maria Gini. 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, 1078–1086.

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

[26] Antonio Murciano, José del R. Millán, and Javier Zamora. 1997. Specialization in multi-agent systems through learning. *Biological Cybernetics* 76, 5 (01 May 1997), 375–382.

[27] GS Nitschke, MC Schut, and AE Eiben. 2008. Emergent specialization in biologically inspired collective behavior systems. In *Intelligent complex adaptive systems.* IGI Global, 215–253.

[28] Shervin Nouyan. 2002. Agent-based approach to dynamic task allocation. In *International Workshop on Ant Algorithms.* Springer, 28–39.

[29] Shervin Nouyan, Roberto Ghizzioli, Mauro Birattari, and Marco Dorigo. 2005. An Insect-Based Algorithm for the Dynamic Task Allocation Problem. *KI* 19, 4 (2005), 25–31.

[30] Ernesto Nunes, Mitchell McIntire, and Maria Gini. 2016. Decentralized allocation of tasks with temporal and precedence constraints to a team of robots. In *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), IEEE International Conference on.* IEEE, 197–202.

[31] Norihiko Ono and Kenji Fukumoto. 1996. Multi-agent reinforcement learning: A modular approach. In *Second International Conference on Multiagent Systems.* 252–258.

[32] David Portugal and Rui Rocha. 2011. A survey on multi-robot patrolling algorithms. In *Doctoral Conference on Computing, Electrical and Industrial Systems.* Springer, 139–146.

[33] Richard Price and Peter Tiňo. 2004. Evaluation of adaptive nature inspired task allocation against alternate decentralised multiagent strategies. In *International Conference on Parallel Problem Solving from Nature.* Springer, 982–990.

[34] Jesús A Román, Sara Rodríguez, and Juan M Corchado. 2014. Improving intelligent systems: Specialization. In *International Conference on Practical Applications of Agents and Multi-Agent Systems.* Springer, 378–385.

[35] Janaína Schwarzrock, Iulisloi Zacarias, Ana LC Bazzan, Ricardo Queiroz de Araujo Fernandes, Leonardo Henrique Moreira, and Edison Pignaton de Freitas. 2018. Solving task allocation problem in multi Unmanned Aerial Vehicles systems using Swarm intelligence. *Engineering Applications of Artificial Intelligence* 72 (2018), 10–20.

[36] Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. 1998. Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B* 265 (1998), 327–332.

[37] Rinde RS van Lon and Tom Holvoet. 2017. When do agents outperform centralized algorithms? *Autonomous Agents and Multi-Agent Systems* 31, 6 (2017), 1578–1609.

[38] Pablo J Villacorta, David A Pelta, and Maria T Lamata. 2013. Forgetting as a way to avoid deception in a repeated imitation game. *Autonomous agents and multi-agent systems* 27, 3 (2013), 329–354.

[39] Jens Wawerla and Richard T Vaughan. 2010. A fast and frugal method for team-task allocation in a multi-robot transportation system.. In *ICRA.* 1432–1437.

[40] Annie S. Wu and Vera A. Kazakova. 2017. Effects of Task Consideration Order on Decentralized Task Allocation Using Time-Variant Response Thresholds. In *Florida Artificial Intelligence Research Society Conference FLAIRS-30.* 466–471.

[41] Zhongshan Zhang, Keping Long, Jianping Wang, and Falko Dressler. 2014. On swarm intelligence inspired self-organized networking: its bionic mechanisms, designing principles and optimization approaches. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 513–537.

[42] Xiaoming Zheng and Sven Koenig. 2011. Generalized reaction functions for solving complex-task allocation problems. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Vol. 22. 478.