Adaptive Collective Responses to Local Stimuli in Anonymous Dynamic Networks

Shunhao Oh
School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Dana Randall
School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Andréa W. Richa
School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, USA

Abstract

We develop a framework for self-induced phase changes in programmable matter in which a collection of agents with limited computational and communication capabilities can collectively perform appropriate global tasks in response to local stimuli that dynamically appear and disappear. Agents reside on graph vertices, where each stimulus is only recognized locally, and agents communicate via token passing along edges to alert other agents to transition to an AWARE state when stimuli are present and an UNAWARE state when the stimuli disappear. We present an Adaptive Stimuli Algorithm that is robust to competing waves of messages as multiple stimuli change, possibly adversarially. Moreover, in addition to handling arbitrary stimulus dynamics, the algorithm can handle agents reconfiguring the connections (edges) of the graph over time in a controlled way.

As an application, we show how this Adaptive Stimuli Algorithm on reconfigurable graphs can be used to solve the *foraging problem*, where food sources may be discovered, removed, or shifted at arbitrary times. We would like the agents to consistently self-organize, using only local interactions, such that if the food remains in a position long enough, the agents transition to a *gather phase* in which many collectively form a single large component with small perimeter around the food. Alternatively, if no food source has existed recently, the agents should undergo a self-induced phase change and switch to a *search phase* in which they distribute themselves randomly throughout the lattice region to search for food. Unlike previous approaches to foraging, this process is indefinitely repeatable, withstanding competing waves of messages that may interfere with each other. Like a physical phase change, microscopic changes such as the deletion or addition of a single food source trigger these macroscopic, system-wide transitions as agents share information about the environment and respond locally to get the desired collective response.

2012 ACM Subject Classification Theory of computation → Self-organization; Theory of computation → Random walks and Markov chains

Keywords and phrases Dynamic networks, adaptive stimuli, foraging, self-organizing particle systems, programmable matter

Digital Object Identifier 10.4230/LIPIcs.SAND.2023.6

Related Version *Full Version*: `https://arxiv.org/abs/2304.12771`

Funding *Shunhao Oh*: Funded in part by U.S. Army Research Office (ARO) award MURI W911NF-19-1-0233.
Dana Randall: Supported by the National Science Foundation (NSF) award CCF-2106687 and by U.S. Army Research Office (ARO) award MURI W911NF-19-1-0233.
Andréa W. Richa: Supported in part by the National Science Foundation (NSF) award CCF-2106917 and by U.S. Army Research Office (ARO) award MURI W911NF-19-1-0233.

Acknowledgements The authors thank the anonymous reviewers for useful feedback on the presentation.

1 Introduction

Self-organizing collective behavior of interacting agents is a fundamental, nearly ubiquitous phenomenon across fields, reliably producing rich and complex coordination. In nature, examples at the micro- and nano-scales include coordinating cells, including our own immune system or self-repairing tissue (e.g., [1]), and bacterial colonies (e.g., [26, 31]); at the macro-scale it can represent flocks of birds [7], shoals of fish aggregating to intimidate predators [27], fire ants forming rafts to survive floods [29], and human societal dynamics such as segregation [34]. Common characteristic of these disparate systems is that they are all self-actuated and respond to simple, local environmental stimuli to collectively change the ensemble's behavior.

In 1991, Toffoli and Margolus coined *programmable matter* to realize a physical computing medium composed of simple, homogeneous entities that can dynamically alter its physical properties in a programmable fashion, controlled either by user input or its own autonomous sensing of its environment [36]. There are formidable challenges to realizing such collective tasks and many researchers in distributed computing and swarm and modular robotics have investigated how such small, simply programmed entities can coordinate to solve complex tasks and exhibit useful emergent collective behavior (e.g., [32]). A more ambitious goal, suggested by self-organizing collective systems in nature, is to design programmable matter systems of *self-actuated* individuals that *autonomously respond to continuous, local changes in their environment.*

The Dynamic Stimuli Problem. As a distributed framework for agents collectively self-organizing in response to changing stimuli, we consider the *dynamic stimuli problem* in which we have a large number of agents that collectively respond to local signals or stimuli. These agents have limited computational capabilities and each only communicates with a small set of immediate neighbors. We represent these agents via a dynamic graph G on n vertices, where the agents reside at the vertices and edges represent pairs that can perceive and interact with each other. At arbitrary points of time, that may be adversarially chosen, *stimuli* dynamically appear and disappear at the vertices of G – these can be a threat, such as an unexpected predator, or an opportunity, such as new food or energy resources.

An agent present at the same vertex as a stimulus acts as a *witness* and alerts other agents. If any agent continues to witness some stimulus over an extended period of time, we want all agents to eventually be alerted, switching to the AWARE state; on the other hand, once witnesses stop sensing a stimulus for long enough, all agents should return to the UNAWARE state. Such collective state changes may repeat indefinitely as stimuli appear and disappear over time. Converging to these two global states enables agents to carry out *differing behaviors in the presence or absence of stimuli*, as observed by the respective witnesses. As a notable and challenging example, in *the foraging problem,* "food" may appear or disappear at arbitrary locations in the graph over time and we would like the collective to *gather around food* (also known as *dynamic free aggregation*) or *disperse in search of new sources*, depending on the whether or not an active food source has been identified. Cannon *et al.* [5] showed how computationally limited agents can be made to gather or disperse; however there the desired goal, aggregation or dispersion, is fixed in advance and the algorithm cannot easily be adapted to move between these according to changing needs.

In addition to the stimuli dynamics, we assume that the agents may *reconfigure the connections (edges) of the graph G over time*, but in a controlled way that still allows the agents to successfully manage the waves of state changes. In a nutshell, G is *reconfigurable* over time if it maintains recurring local connectivity of the Aware agents (i.e., if it makes sure that the 1-hop neighborhood sets of Aware agents stay connected over time), as its edge set changes. The edge dynamics may be fully in the control of an adversary, or may be controlled by the agents themselves, depending on the context (e.g., in the foraging problem presented in Section 5, the agents control the edge dynamics).

In this framework, we assume that at all times there are at most a constant number w of stimuli present at the vertices of G. Agents are anonymous and each acts as a finite automaton with constant-size memory, constant degree, and no access to global information other than w and a constant upper bound Δ on the maximum degree. Individual agents are activated according to their own Poisson clocks and perform instantaneous actions upon activation, a standard way to allow sites to update independently and asynchronously (since the probability two Poisson clocks tick at the exact same instant in time is negligible). For ease of discussion, we may assume the Poisson clocks have the same rate, so this model is equivalent to a *random sequential scheduler* that chooses an agent uniformly at random to be activated at discrete iterations $t \in \{1, 2, 3, \ldots\}$. We denote by G_t the configuration of the reconfigurable graph at iteration t. When *activated* at iteration t, an agent perceives its own state and the states of its current neighbors in G_t, performs a bounded amount of computation, and can change its own and its neighbors states, including any "tokens" (i.e., constant size messages received or sent).[1] At each iteration $t \in \{1, 2, 3, \ldots\}$, we denote by $\mathcal{W}_t \subseteq V$ the set of witnesses. The sets $\mathcal{W}_t$ can change arbitrarily (adversarially) over time, but $|\mathcal{W}_t|$ is always bounded by the constant w, for all t, since w is an upper on the number of concurrent stimuli.

Overview of Results. Our contributions are two-fold. First, we present an efficient, robust algorithm for the dynamic stimuli problem for a class of reconfigurable graphs. (The precise details of what constitutes a *valid reconfigurable graph* will be given in Section 4.) Whenever an agent encounters a new stimulus, the entire collective efficiently transforms to the Aware state, so the agents can implement an appropriate collective response. After a stimulus vanishes, they all return to the Unaware state, recovering their neutral collective behavior.

We show that the system will always converge to the appropriate state (i.e., Aware or Unaware) once the stimuli stabilize for a sufficient period of time. Specifically, if there are no witnesses in the system for a sufficient period of time, then all agents in the Adaptive Stimuli Algorithm will reach and remain in the Unaware state in $O(n^2)$ expected time (Theorem 14). Likewise, if the set of witnesses (and stimuli) remains unchanged for a sufficient period of time, all agents will reach and remain in the Aware state in expected time that is a polynomial in n and the "recurring rate" of G, which captures how frequently disconnected vertices come back in contact with each other (Theorem 15). In particular, if G is static or if G_t is connected, for all t, the expected convergence time until all agents transition to the Aware state is $O(n^5)$ (Theorem 4) and $O(n^6 \log n)$ (Corollary 16), respectively. Moreover, the system can recover if the witness set changes before the system converges to the Aware or Unaware states.

While the arguments are simpler for sequences of connected graphs generated by, say, an oblivious adversary, the extension to the broader class of reconfigurable graphs includes graphs possibly given by non-oblivious adversaries that may occasionally disconnect. This

[1] Since we assume a sequential scheduler, such an action is justified; in the presence of a stronger adversarial scheduler, e.g., the asynchronous scheduler, one would need a more detailed message passing mechanism to ensure the transfer of tokens between agents, and resulting changes in their states.

generalization provides more flexibility for agents in the Aware and Unaware states to implement more complex behaviors, as was used for applying the Adaptive Stimuli Algorithm to foraging, which we describe next.

Our second main contribution is the first efficient algorithm for the *foraging problem*, where food dynamically appears and disappears over time at arbitrary sites on a finite $\sqrt{N} \times \sqrt{N}$ region of the triangular lattice. Agents want to gather around any discovered food source (also known as *dynamic free aggregation*) or disperse in search of food. The algorithm of Cannon *et al.* [5, 25] uses insights from the high and low temperature phases of the ferromagnetic Ising model from statistical physics to provably achieve either desired collective response: there is a preset global parameter λ related to inverse temperature, and the algorithm provably achieves aggregation when λ is sufficiently high and dispersion when sufficiently low. We show here that by applying the Adaptive Stimuli Algorithm, the phase change (or bifurcation) for aggregation and dispersion can be self-modulated based on local environmental cues that are communicated through the collective to induce desirable system-wide behaviors in polynomial time in n and N, as stated in Theorems 23 and 24.

Collectively transitioning between Aware and Unaware states enables agents to *correctly self-regulate system-wide adjustments* in their bias parameters when one or more agents notice the presence or depletion of food to induce the appropriate global coordination to provably transition the collective between macro-modes when required. We believe other collective behaviors exhibiting emergent bifurcations, including separation/integration [4] and alignment/nonalignment [22], can be similarly self-modulated.

The distinction between polynomial and super-polynomial running times is significant here because our algorithms necessarily make use of competing broadcast waves to propagate commands to change states. A naive implementation of such a broadcast system may put us in situations where neither type of wave gets to complete its propagation cycle. This may continue for an unknown amount of time, so the agents may fail to reach an agreement on their state. The carefully engineered token passing mechanism ensures that when a stimulus has been removed, the *rate at which the agents "reset" to the Unaware state outpaces the rate at which the cluster of Aware agents may continue to grow*, ensuring that the newer broadcast wave always supersedes previous ones and completes in expected polynomial time. Moreover, while a stimulus is present, there is a continuous *probabilistic generation of tokens that move according to a d_{max}-random walk* among Aware agents until they find a new agent to become Aware, thus ensuring the successful convergence to the Aware state in expected polynomial time.

Related work. Dynamic networks have been of growing interest recently and have spawned several model variants (see, e.g., the surveys in [6] and [24]). There is also a vast literature on broadcasting, or information dissemination, in both static and dynamic networks (e.g., [23, 20, 8, 15, 14]), where one would like to disseminate k messages to all nodes of a graph G, usually with unique token ids and $k \leq n$, polylog memory at the nodes (which may also have unique ids), and often nodes' knowledge of k and possibly also of n. Note that any of these assumptions violates our agents' memory or computational capabilities. Moreover, since our collective state-changing process runs indefinitely, any naive adaptations of these algorithms would need that $k \to \infty$ to ensure that with any sequence of broadcast waves, the latest always wins.

Broadcast algorithms that do not explicitly keep any information on k (number of tokens or broadcast waves) or n would be more amenable to our agents. Amnesiac flooding is one such broadcast algorithm that works on a network of anonymous nodes without keeping any

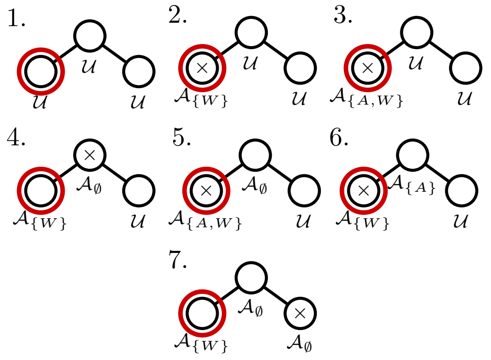


Figure 1 Illustration of how the presence of a witness (circled in red) gradually converts all agents to the Aware state through the distribution of alert tokens. The agent with the "×" is the agent activated in that step.

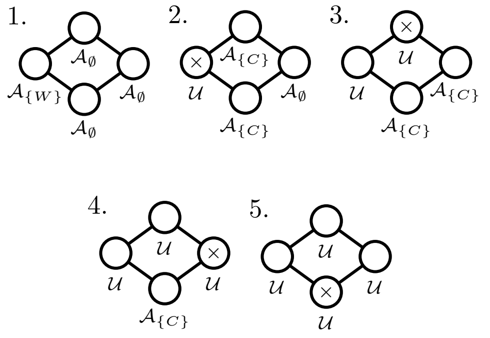


Figure 2 Illustration of how all-clear tokens are broadcast from an agent with the witness flag set that is no longer a witness (the leftmost agent). The agent with the "×" is the agent activated in that step.

state or other information as the broadcast progresses. In [21], Hussak and Trehan show that amnesiac flooding will always terminate in a static network under synchronous message passing, but may fail on a dynamic network or with non-synchronous executions.

Many studies in self-actuated systems take inspiration from emergent behavior in social insects, but either lack rigorous mathematical foundations explaining the generality and limitations as sizes scale (see, e.g., [19, 18, 9, 38]), often approaching the thermodynamic limits of computing [37] and power [10], or rely on long-range signaling, such as microphones or line-of-sight sensors [35, 16, 17, 30]. Some recent work on stochastic approaches modeled after systems from particle physics has been made rigorous, but only when a single, static goal is desired [5, 4, 25, 2, 33, 22].

2 The Adaptive Stimuli Algorithm

The *Adaptive Stimuli Algorithm* is designed to efficiently respond to dynamic local stimuli that indefinitely appear and disappear at the vertices of G. Recall the goal of this algorithm is to allow the collective to converge to the Aware state whenever a stimulus is witnessed for long enough and to the Unaware state if no stimulus has been detected recently. The algorithm converges in expected polynomial time in both scenarios under a reconfigurable dynamic setting, as we show in Sections 3-4, even as the process repeats indefinitely.

All agents know two parameters of the system: $\Delta \geq 1$, an upper bound on the maximum degree of the graph, and $w \geq 1$, an upper bound on the size of the witness sets $\mathcal{W}_t$ at all times t (which is needed to determine the probability $p < 1/w$ for some agents to change states or generate certain tokens). Our algorithm defines a carefully balanced token passing mechanism, where a *token* is a constant-size piece of information: Upon activation, an Aware witness u continuously generates *alert tokens* one at a time, with probability p, which will each move through a *random walk over Aware agents until they come in contact with a neighboring Unaware agent u*: The token is then consumed and u changes its state to Aware (Figure 1). On the other hand, if a witness notices that its co-located stimulus has disappeared, it will initiate an all-clear token broadcast wave which will proceed through agents in the Aware state, switching those to Unaware (Figure 2).

Algorithm 1 Adaptive Stimuli Algorithm.

```
procedure THE ADAPTIVE-STIMULI-ALGORITHM($u$)
    Let $p < 1/w \in (0,1)$
    $u.isWitness$ ←TRUE if $u$ is a witness, else $u.isWitness$ ←FALSE
    if $u.isWitness$ and $u.state \notin \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}\}$ then
        With probability $p$, $u.state \leftarrow \mathcal{A}_{\{W\}}$    ▷ $u$ becomes AWARE witness with prob. $p$
    else if $\neg u.isWitness$ and $u.state \in \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}\}$ then ▷ stimulus no longer at $u$
        for each $v \in N_{\mathcal{A}}(u)$ do    ▷ $N_{\mathcal{A}}(u)$ = current AWARE neighbors of $u$
            $v.state \leftarrow \mathcal{A}_{\{C\}}$    ▷ all-clear token broadcast to AWARE neighbors of $u$
        $u.state \leftarrow \mathcal{U}$
    else
        switch $u.state$ do
            case $\mathcal{U}$:
                if $\exists v \in N_{\mathcal{A}}(u), v.state = \mathcal{A}_S \in \{\mathcal{A}_{\{A\}}, \mathcal{A}_{\{A,W\}}\}$ then    ▷ $v$ has alert token
                    $v.state \leftarrow \mathcal{A}_{S\setminus\{A\}}$    ▷ $v$ consumes alert token
                    $u.state \leftarrow \mathcal{A}_{\emptyset}$    ▷ $u$ becomes aware
            case $\mathcal{A}_{\{A\}}$ or $\mathcal{A}_{\{A,W\}}$:
                $x \leftarrow$ random value in $[0,1]$
                if $x \le d_G(u)/\Delta$ then    ▷ $d_{max}$-random walk
                    $v \leftarrow$ random neighbor of $u$
                    if $v.state = \mathcal{A}_{S'} \in \{\mathcal{A}_{\emptyset}, \mathcal{A}_{\{W\}}\}$ then    ▷ AWARE state, no alert token
                        Let $u.state = \mathcal{A}_S$; $u.state \leftarrow \mathcal{A}_{S\setminus\{A\}}$    ▷ $u$ sends alert token to $v$
                        $v.state \leftarrow \mathcal{A}_{S'\cup\{A\}}$    ▷ $v$ receives alert token
            case $\mathcal{A}_{\{W\}}$:
                With probability $p$, $u.state \leftarrow \mathcal{A}_{\{A,W\}}$ ▷ generate alert token with prob. $p$
            case $\mathcal{A}_{\{C\}}$:
                for each $v \in N_{\mathcal{A}}(u)$ do
                    $v.state \leftarrow \mathcal{A}_{\{C\}}$    ▷ $u$ broadcasts all-clear token to all aware neighbors
                $u.state \leftarrow \mathcal{U}$
```

The differences between these two carefully crafted token passing mechanisms allow us to ensure that whenever there has been no stimulus in the network for long enough, the rate at which the agents deterministically learn this (through broadcasts of all-clear tokens) and become UNAWARE always outpaces the probabilistic rate at which the cluster of AWARE agents may still continue to grow. Thus, the UNAWARE broadcast wave will always outpace any residual AWARE waves in the system and will allow the collective to correctly converge to the desired UNAWARE state. On the other hand, if the witness set is non-empty and remains stable for a long enough period of time, all agents will eventually switch to the AWARE state since, after some time, there will be no all-clear tokens in G, as UNAWARE agents do not ever generate or broadcast tokens.

To define the Adaptive Stimuli Algorithm, we utilize the following flags and states:

- **Alert token flag** (A): used to indicate that an agent has an alert token.
- **All-clear token flag** (C): used to indicate that the agent has an all-clear token.
- **Witness flag** (W): used to indicate that the agent is a witness.
- **States**: The UNAWARE state is denoted by $\mathcal{U}$, while $\{\mathcal{A}_{\emptyset}, \mathcal{A}_{\{A\}}, \mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}, \mathcal{A}_{\{C\}}\}$ denote the AWARE states. In the AWARE states, the subscript denotes the subset of flags that are currently set. Note that the all-clear token flag is only set when the other two are not, giving us six distinct states in total.

Algorithm 1 formalizes the actions executed by each agent u when activated. Agent u's actions depend on the current state it is in and whether it senses a stimulus. We describe the behavior for each of the possible cases below:

- **Non-matching witness flag**: This is a special case that occurs if u is currently a witness to some stimuli but its witness flag has not been set yet, or if u has its witness flag set but is no longer a witness. This case takes priority over all the other possible cases, since u cannot take any action before its witness status and its state match. If u is a witness but does not have the witness flag set, then with probability p, switch u to state $\mathcal{A}_{\{W\}}$. On the other hand, if u is not a witness but has the witness flag set, switch u to UNAWARE (by setting $u.state = \mathcal{U}$) and broadcast an all-clear token to all of u's AWARE neighbors (this token overrides any other tokens the neighbors may have).
- **Unaware state ($\mathcal{U}$)**: If u has an AWARE neighbor v with an alert token (i.e., $v.state \in \{\mathcal{A}_{\{A\}}, \mathcal{A}_{\{A,W\}}\}$), then u consumes the alert token from v (by setting v's alert token flag to FALSE) and switches to the state $\mathcal{A}_{\emptyset}$.
- **Aware state with alert token ($\mathcal{A}_{\{A\}}$, $\mathcal{A}_{\{A,W\}}$)**: Pick a random neighbor of u such that each neighbor is picked with probability $\frac{1}{\Delta}$, with a probability $1 - \frac{d_G(u)}{\Delta}$ of staying at u (this executes a d_{max}-random walk [13]). If an AWARE neighbor v is picked and v does not have an alert nor an all-clear token (that is, $v.state \in \{\mathcal{A}_{\emptyset}, \mathcal{A}_{\{W\}}\}$), move the alert token to v by toggling the alert token flags on both u and v.
- **Aware state with witness flag but without an alert token ($\mathcal{A}_{\{W\}}$)**: With probability p, u generates a new alert token by switching to state $\mathcal{A}_{\{A,W\}}$.
- **Aware state with all-clear token ($\mathcal{A}_{\{C\}}$)**: Switch u to the UNAWARE state $\mathcal{U}$ and broadcast the all-clear token to all of its AWARE state neighbors.

The use of a d_{max}-random walk instead of regular random walk (Line 18 of Algorithm 1) normalizes the probabilities of transitioning along an edge by the maximum degree of the nodes (or a constant upper bound on that), so that these transition probabilities cannot change during the evolution of the graph. A d_{max}-random walk has polynomial hitting time on any connected dynamic network (while a regular random walk might not) [3].

3 Static graph topologies

For simplicity, we will first state and prove our results for *static connected graph topologies* (Theorems 3 and 4), where the edge set never changes and the dynamics are only due to the placement of stimuli. In Section 4, we show that the same proofs apply with little modification to the reconfigurable case as well.

In order to define our main theorems, we must first define the *state invariant*, that we know holds from an initial configuration where every agent is initialized in the UNAWARE state, as we show Lemma 2. In the remainder of this paper, a *component* will refer to a connected component of the subgraph induced in G by the set of AWARE agents.

▶ **Definition 1** (State Invariant). *We say a component satisfies the* state invariant *if it contains at least one agent in the states $\mathcal{A}_{\{W\}}$, $\mathcal{A}_{\{A,W\}}$ or $\mathcal{A}_{\{C\}}$. A configuration satisfies the state invariant if every component (if any) of the configuration satisfies the state invariant.*

▶ **Lemma 2.** *If the current configuration satisfies the state invariant, then all subsequent configurations reachable by Algorithm 1 also satisfy the state invariant.*

Proof. Starting from configuration where the state invariant currently holds, let u be the next agent to be activated. If u is a witness but $u.state \notin \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}\}$, switching u to state $\mathcal{A}_{\{W\}}$ does not affect the state invariant. Conversely, if u is not a witness, but

$u.state \in \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}\}$ switching u to state $\mathcal{U}$ can potentially split the component it is in into multiple components. However, as all neighbors of u will also be set to state $\mathcal{A}_{\{C\}}$, each of these new components will contain an agent in state $\mathcal{A}_{\{C\}}$.

Otherwise, if $u.state = \mathcal{U}$, it only switches to the AWARE state if it neighbors another AWARE agent. As the component u joins must contain an agent in states $\mathcal{A}_{\{W\}}$, $\mathcal{A}_{\{A,W\}}$ or $\mathcal{A}_{\{C\}}$, the new configuration will continue to satisfy the state invariant. If $u.state = \mathcal{A}_{\{C\}}$, similar to the earlier case where u is not a witness but $u.state \in \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}\}$, activating u may split the component it is in. As before, all neighbors of u will be set to state $\mathcal{A}_{\{C\}}$, so each of these newly created components satisfy the state invariant. The remaining possible cases only toggle the alert token flag, which does not affect the state invariant. ◀

This allows us to state our main results in the static graph setting. We let $T \in \mathbb{N}$ represent the time when the stimuli stabilize long enough to converge (where T is unknown to the agents) and show that we will have efficient convergence. Without loss of generality, for the sake of our proofs, we will assume that $\mathcal{W}_t = \mathcal{W}_T$, for all $t \geq T$, although this is really just representing a phase where the stimuli are stable.

▶ **Theorem 3.** *Starting from any configuration satisfying the state invariant over a static connected graph topology G, if $|\mathcal{W}_T| = 0$, then all agents will reach and remain in the* UNAWARE *state in $O(n^2)$ expected iterations, after time T.*

▶ **Theorem 4.** *Starting from any configuration satisfying the state invariant over a static connected graph topology G, if $|\mathcal{W}_T| > 0$, then all agents will reach and stay in the* AWARE *state in $O(n^5)$ expected iterations, after time T.*

The proof of these theorems relies on carefully eliminating residual aware agents from previous broadcasts. These agents have yet to receive an all-clear token and thus will take some time before they can return to the UNAWARE state.

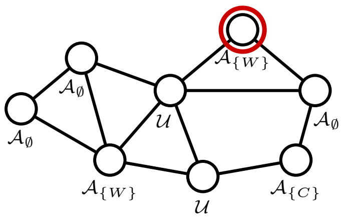


Figure 3 A configuration with two residual components. The component on the left is a residual component despite having a witness in it because it contains an agent with the all-clear flag; the component on the right is a residual component since it contains an agent in state $\mathcal{A}_{\{C\}}$.

▶ **Definition 5** (Residuals). *A* residual component *is a component that satisfies at least one of the following two criteria:*
1. *It contains an agent in state $\mathcal{A}_{\{C\}}$.*
2. *It contains an agent in state $\mathcal{A}_{\{W\}}$ or $\mathcal{A}_{\{A,W\}}$ that is not a witness.*

We call agents belonging to residual components residuals*.*

When starting from arbitrary configurations, there is likely to be a large number of residual components that need to be cleared out. Residuals are problematic as they do no stay in the aware state in the long term. This means they do not actually contribute to the main cluster of AWARE state agents, and the presence of residuals causes even more residuals to form. Furthermore, later on when we allow the graph to reconfigure itself, residuals may obstruct UNAWARE state agents from coming into contact with non-residual components, which may prevent alert tokens from reaching them. Our main tool to show that all residuals eventually vanish is a *potential function* that decreases more quickly than it increases.

▶ **Definition 6** (Potential). *For a configuration σ, we define its potential $\Phi(\sigma)$ as*

$$\Phi(\sigma) := \Phi_A(\sigma) + \Phi_{AT}(\sigma)$$

where $\Phi_A(\sigma)$ and $\Phi_{AT}(\sigma)$ represent total the number of AWARE agents and the number of AWARE agents with an alert token respectively.

We use the following lemma to guarantee, in Lemma 8, that the expected number of steps before all residuals are removed is polynomial.

▶ **Lemma 7.** *Assume $n \geq 2$, and let $0 < \eta < 1$. Consider two random sequences of probabilities $(p_t)_{t\in\mathbb{N}_{\geq 0}}$ and $(q_t)_{t\in\mathbb{N}_{\geq 0}}$, with the properties that $\frac{1}{n} \leq p_t \leq 1$ and $0 \leq q_t \leq \frac{\eta}{n}$, and $p_t + q_t \leq 1$. Now consider a sequence $(X_t)_{t\in\mathbb{N}_{\geq 0}}$, where*

$$X_{t+1} \begin{cases} \leq X_t - 1 & \text{with probability } p_t \\ = X_t + 1 & \text{with probability } q_t \\ = X_t & \text{with probability } 1 - p_t - q_t. \end{cases} .$$

Then $\mathbb{E}\left[\min\{t \geq 0 \mid X_t = 0\}\right] \leq \frac{nX_0}{1-\eta}$.

Proof. For each $k \in \mathbb{N}_{\geq 0}$, we define a random sequence $(Y_t^{(k)})_{t\in\mathbb{N}_{\geq 0}}$ such that $Y_0^{(k)} = k$ and

$$Y_{t+1}^{(k)} \begin{cases} = Y_t^{(k)} - 1 & \text{with probability } \frac{1}{n} \\ = Y_t^{(k)} + 1 & \text{with probability } \frac{\eta}{n} \\ = Y_t^{(k)} & \text{otherwise.} \end{cases}$$

For each such k, let $S_k := \mathbb{E}\left[\min\{t \geq 0 \mid Y_t^{(k)} = 0\}\right]$. Let $T_{k+1}^{(k)} = 0$ and for $i \in \{k, k-1, \ldots, 1\}$, let $T_i^{(k)} = \min\{t \geq 0 \mid X_t^{(k)} \leq i\} - T_{i+1}^{(k)}$ denote the number of time steps after $T_{i+1}^{(k)}$ before the first time step t where $Y_t^{(k)} \leq i$. We observe that each $T_i^{(k)}$ is identically distributed, with $\mathbb{E}T_i^{(k)} = \mathbb{E}T_1^{(k)} = S_1$. Also, we observe that $S_k = \mathbb{E}[\sum_{i=1}^k T_i^{(k)}]$, so $S_k = k \cdot S_1$ for all k. We can thus compute S_1 by conditioning on the first step:

$$S_1 = \frac{1}{n}(1) + \frac{\eta}{n}(S_2 + 1) + \left(1 - \frac{1+\eta}{n}\right)(S_1 + 1) = 1 + \frac{\eta}{n} \cdot 2S_1 + \left(1 - \frac{1+\eta}{n}\right) S_1$$

This implies $S_1 = \frac{n}{1-\eta}$ and thus $\mathbb{E}\left[\min\{t \geq 0 \mid Y_t^{(k)} = 0\}\right] = S_k = \frac{kn}{1-\eta}$.

We can then couple $(X_t)_{t\in\mathbb{N}_{\geq 0}}$ and $(Y_t^{(X_0)})_{t\in\mathbb{N}_{\geq 0}}$ in a way such that $Y_{t+1}^{(X_0)} = Y_t^{(X_0)} + 1$ whenever $X_{t+1} > X_t$, and $Y_{t+1}^{(X_0)} = Y_t^{(X_0)} - 1$ whenever $X_{t+1} < X_t$. We thus have $Y_t^{(X_0)} \geq X_t$ always, and so $\mathbb{E}\left[\min\{t \geq 0 \mid X_t = 0\}\right] \leq \mathbb{E}\left[\min\{t \geq 0 \mid Y_t^{(X_0)} = 0\}\right] \leq \frac{kX_0}{(1-\eta)}$. ◀

We show in Lemma 8 that after a polynomial number of steps in expectation, we will reach a configuration with no residual components.

▶ **Lemma 8.** *We start from a configuration satisfying the state invariant over a static connected graph G with no more than w witnesses at any point. Then the expected number of steps before we reach a configuration with no residual components is at most $2n^2/(1 - wp)$.*

Proof. We apply Lemma 7 to the sequence of potentials $(\Phi(\sigma_t))_{t\in\mathbb{N}_{\geq 0}}$ where σ_t is the configuration after iteration t. As long as there exists a residual component, there will be at least one agent will switch to the UNAWARE state on activation. This gives a probability of at least $1/n$ in any iteration of decreasing the current potential by at least 1.

There are only two ways for the potential to increase. The first is when a new alert token is generated by a witness, and the second is when an UNAWARE witness switches to an AWARE state. The activation of a witness thus increases the current potential by exactly 1, with probability p. As there are at most w witnesses, this happens with probability at most $wp/n < 1/n$. Note that the consumption of an alert token to add a new AWARE state agent to a residual component does not change the current potential. Neither does switching agents to the all-clear token state affect the potential.

By Lemma 7, as $\Phi(\sigma_0) \leq 2n$, within $2n^2/(1-wp)$ steps in expectation, we will either reach a configuration σ with $\Phi(\sigma) = 0$, or a configuration with no residual components, whichever comes first. Note that if $\Phi(\sigma) = 0$, then σ cannot have any residual components, completing the proof. ◀

We now show that as long as no agent is removed from the witness set, after all residual components are eliminated, no new ones will be generated:

▶ **Lemma 9.** *We start from a configuration satisfying the state invariant over a static connected graph G, and assume that no agent will be removed from the witness set from the current point on. If there are currently no residuals, then a residual cannot be generated.*

Proof. With no residual components in the current iteration, there will be no agents in state $\mathcal{A}_{\{C\}}$, and all agents in state $\mathcal{A}_{\{W\}}$ or $\mathcal{A}_{\{A,W\}}$ will be witnesses. This means that no agent on activation will switch another agent to the $\mathcal{A}_{\{C\}}$ state. All agents in states $\mathcal{A}_{\{W\}}$ or $\mathcal{A}_{\{A,W\}}$ will continue to be witnesses by assumption of the lemma, and agents will only switch to states $\mathcal{A}_{\{W\}}$ or $\mathcal{A}_{\{A,W\}}$ if they are witnesses. Thus no component will be residual in the next iteration. ◀

When $|\mathcal{W}_T| = 0$ and no witnesses exist in the long term, if the state invariant holds, then all agents will be in the UNAWARE state, giving us Theorem 3. On the other hand, in order to show Theorem 4,if a witness persists in the long term , we need to show that all agents eventually switch to the AWARE state. Lemma 10 establishes this as without residuals, no AWARE state agent can revert to the UNAWARE state.

▶ **Lemma 10.** *We start from a configuration satisfying the state invariant over a static connected graph G, and assume that there are no residual agents, the witness set is nonempty and no agent will be removed from the witness set from the current point on. Then the expected number of iterations before the next agent switches from the UNAWARE to the AWARE state is at most $O(n^4)$.*

Proof. In the case where there is no AWARE agent, there must be an UNAWARE witness, which on activation switches to the AWARE state with probability p. The expected time before this happens is no more than $O(\frac{n}{p})$. Thus for the rest of the proof, we may assume every witness is already in the AWARE state with the witness flag set.

If we have a bound on the expected time before we reach a configuration where every AWARE agent holds an alert token, all it takes following that is for any UNAWARE agent to be activated while holding an alert token. As the graph G is connected with least one UNAWARE and one AWARE agent, there will be some UNAWARE agents neighboring an AWARE agent, so the expected number of iterations before this occurs will is $O(n)$.

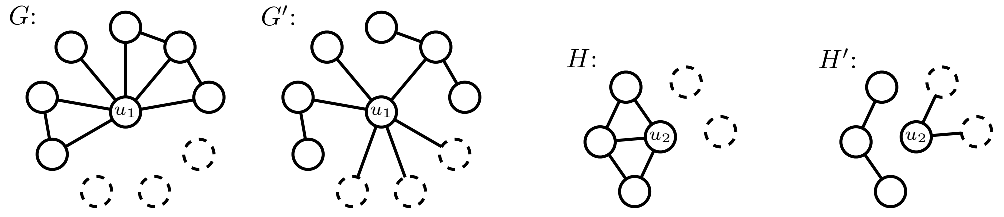


Figure 4 Two locally connected reconfigurations of the vertices u_1 (from G to G') and u_2 (from H to H') respectively, where only the Aware neighbors of the reconfigured vertex are shown. Vertices with dashed outlines are newly introduced neighbors.

We now bound the amount of time it takes for all Aware state agents to hold alert tokens. As an agent can only hold one alert token at a time, a new alert token can only be generated when a witness does not hold an alert token. Assume that there is at least one Aware state agent that does not have an alert token. Mark one such agent. Suppose that u is the marked agent and that the next agent v to be activated is a neighbor of u. If v has an alert token and v randomly chooses u as its outgoing neighbor (per the algorithm), then v transfers its alert token to u and receives the mark from u. Otherwise, for the sake of our analysis, we still have v pick an outgoing neighbor at random and receive the mark if the chosen neighbor is u.

The mark moving in this manner is equivalent to following a d_{max}-random walk over the subgraph induced by the Aware agents, which is static as long as no new agents are switching to the Aware state. As the configuration satisfies the state invariant and there are no residuals, the component of Aware agents the mark is in must contain at least one witness. The worst case hitting time of the d_{max}-random walk over this subgraph is $O(n^2)$, and thus the expected number of iterations before the mark lands on a witness is $O(n^3)$, allowing a new alert token to be generated (this new alert token is generated with a constant probability $p \in (0, 1)$). As there are at most n Aware agents, all Aware agents will be holding an alert token after $O(n^4)$ iterations in expectation, which translates to an expected time bound of $O(n^4)$ before a new Aware agent is added. Note that this is a loose bound - the bound has not been optimized for clarity of explanation. ◀

4 Reconfigurable topologies

We show that the same results hold if we allow some degree of reconfigurability of the edge set and relax the requirement that the graph must be connected. This gives us enough flexibility to implement a wider range of behaviors, like free aggregation or compression and dispersion in Section 5. When needed, we may refer to this as the *reconfigurable dynamic stimuli problem*, in order to clearly differentiate from the dynamic stimuli problem on static graphs that we considered in Section 3.

Instead of a static graph G, as we considered in Section 3, we now allow the edge set of the graph to be locally modified over time. These reconfigurations can be initiated by the agents themselves or controlled by an adversary, and they can be randomized or deterministic, but we require some restrictions on what reconfigurations are allowed, and what information an algorithm carrying out these reconfigurations may have access to. This will result in a restricted class of dynamic graphs, but will be general enough to be applied to the problem of foraging that we describe in Section 5.

The basic primitive for (local) reconfiguration of our graph by an agent u is replacing the edges incident to vertex u with new edges. We call this a reconfiguration of vertex u and define local connectivity to formalize which reconfigurations are allowed.

▶ **Definition 11** (Locally Connected Reconfigurations). *For any graph $G = (V, E)$, let G' be a the graph resulting from a reconfiguration of vertex $u \in V$ and let $N_{\mathcal{A}}(u)$ be the* Aware *neighbors of u in G. We say that this reconfiguration is locally connected if u has at least one* Aware *neighbor in G' and if for every pair of vertices v_1, v_2 in $N_{\mathcal{A}}(u)$ with a path from v_1 to v_2 in the induced subgraph $G[N_{\mathcal{A}}(u) \cup \{u\}]$, there is also a path from v_1 to v_2 in $G'[N_{\mathcal{A}}(u) \cup \{u\}]$.*

Examples of locally connected reconfigurations are given in Figure 4.

In the reconfigurable dynamic stimuli problem, we want to be able to define reconfiguration behaviors for agents in the Aware (without all-clear token) and Unaware states. To define what reconfigurations of an agent u are valid, we group our set of agent states into three subsets which we refer to as *behavior groups*. The three behavior groups are $\widehat{\mathcal{U}} := \{\mathcal{U}\}$, $\widehat{\mathcal{M}} := \{\mathcal{A}_{\emptyset}, \mathcal{A}_{\{A\}}\}$ and $\widehat{\mathcal{I}} := \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}, \mathcal{A}_{\{C\}}\}$, which we call Unaware, Mobile and Immobile respectively (the latter two referring to Aware state agents which are and are not allowed to change their neighboring edges respectively). We say that a locally connected reconfiguration of an agent u is *valid* if it is not allowed to reconfigure in the Immobile behavior group, and if u is in the Mobile behavior group, the reconfiguration of u must be locally connected (Definition 11). We show the following lemma:

▶ **Lemma 12.** *Let $G = (V, E)$ be a graph and let $G' = (V, E')$ be the graph resulting from a valid locally connnected reconfiguration of a vertex $u \in V$. If a configuration satisfies the state invariant on G, then the same state assignments satisfy the state invariant on G'.*

Proof. As locally connected reconfigurations of Unaware agents do not affect the invariant and Immobile agents cannot be reconfigured, it suffices to show that locally connected reconfigurations of Mobile agents maintain the state invariant.

To show that the state invariant holds on G', we show that any Mobile agent has a path to an Immobile agent in G'. Consider any such Mobile agent $v \neq u$. As the state invariant is satisfied on G, there exists a path in G over Aware vertices from v to an Immobile agent w. If this path does not contain the agent u, v has a path to w in G'. On the other hand, if this path contains the agent u, consider the vertices u_1 and u_2 before and after u respectively in this path. By local connectivity, there must still be a path from u_1 to u_2 in the induced subgraph $G'[N_{\mathcal{A}}(u) \cup \{u\}]$, and so a path exists from v to w over G'. It remains to check that u also has a path to an Immobile agent in G'. Once again by local connectivity, u must have an Aware neighbor in G', which must have a path to an Immobile agent over G'. ◀

In the reconfigurable version of the dynamic stimuli problem, we have a random sequence of graphs $(G_0, G_1, G_2, \ldots)$ where G_t for $t \geq 1$ denotes the graph used in iteration t. These graphs share a common vertex set V, the set of agents, but the edges may change from iteration to iteration. We do not consider fully arbitrary sequence of graphs, but instead one that is generated by what we call a *(valid) reconfiguration adversary $\mathcal{X}$*. Let the random sequence $(X_0, X_1, X_2, \ldots)$ denote the information available to the reconfiguration adversary on each iteration, and for each $t \geq 1$ we let the vector $\widehat{\sigma}_t : V \to \{\widehat{\mathcal{U}}, \widehat{\mathcal{M}}, \widehat{\mathcal{I}}\}$ denote the behavior groups of the agents at the end of iteration t (i.e., after the activated agent performed any computation/change of states of its neighborhood at iteration t). At iteration t, before an agent is activated in the stimuli algorithm, the new graph G_t and the next value X_t of the sequence are drawn as a pair from the distribution $\mathcal{X}(X_{t-1}, \widehat{\sigma}_{t-1})$, which assigns non-zero probabilities only to graphs that can be obtained through some sequence of valid locally connected reconfigurations of the vertices of G_{t-1}.

We note that the reconfiguration adversary can be deterministic or randomized (it can even be in control of the agents themselves), and is specifically defined to act based on the behavior group vectors $\widehat{\sigma}_t : V \to \{\widehat{\mathcal{U}}, \widehat{\mathcal{M}}, \widehat{\mathcal{I}}\}$ and *not* on the state vector of the agents. We explicitly do not give the reconfiguration adversary access to full state information, as convergence time bounds require that the reconfiguration adversary of the graph be oblivious to the movements of alert tokens. As a special case, our results hold for any sequence of graphs $(G_0, G_1, G_2, \ldots)$ pre-determined by an oblivious adversary. An example of a valid reconfiguration adversary that takes full advantage of the generality of our definition can be seen in the Adaptive α-Compression Algorithm, an algorithm that we will later introduce to solve the problem of foraging.

In the static version of the problem, the graph is required to be connected to ensure that agents will always be able to communicate with each other. Without this requirement, we can imagine simple examples of graphs or graph sequences where no algorithm will work. In particular, if a set of agents that contains a witness never forms an edge to an agent outside of the set, there would be no way to transmit information about the existence of the witnesses to the nodes outside the set. However, as the foraging problem will require disconnections to some extent, we relax this requirement that each graph G_t is connected, and instead quantify how frequently UNAWARE state agents come into contact with AWARE state agents.

We say an UNAWARE agent is *active* if it is adjacent to an AWARE agent. One way to quantify how frequently agents become active is to divide the iterations into "batches" of bounded expected duration, with at least some amount of active agents in each batch. The random variables $(D_1, D_2, D_3 \ldots)$ denote the durations (in iterations) of these batches, and the random variables $(C_1, C_2, C_3, \ldots)$ denote the number of active agents in the respective batches. Definition 13 formalizes this notion.

▶ **Definition 13** (Recurring Sequences). *Let $\mathcal{X}$ be a fixed valid reconfiguration adversary. We say that this $\mathcal{X}$ is (U_D, U_C)-recurring (for $U_D \geq 1$ and $U_C \in (0,1)$) if for each possible starting iteration t and fixed behavior group $\widehat{\sigma}$ with at least one UNAWARE and one IMMOBILE agent, we have the following property: There exists sequences of random variables $(D_1, D_2, D_3 \ldots)$ and $(C_1, C_2, C_3 \ldots)$ where for each $k \in \{1, 2, 3, \ldots\}$,*

1. *C_k denotes the number of active agents under the behavior group $\widehat{\sigma}$ between iterations $t + \sum_{i=1}^{k-1} D_i$ and $t + \left(\sum_{i=1}^{k} D_i\right) - 1$.*
2. *$\mathbb{E}\left[D_k \mid D_1, D_2, \ldots D_{k-1}, C_1, C_2 \ldots C_{k-1}\right] \leq U_D$.*
3. *$\mathbb{E}\left[\left(1 - \frac{1}{n}\right)^{C_k} \mid D_1, D_2, \ldots D_{k-1}, C_1, C_2 \ldots C_{k-1}\right] \leq U_C$.*

We can then define a *(valid) reconfigurable graph (or sequence)* as one that is generated by a valid reconfiguration adversary $\mathcal{X}$ and is (U_D, U_C)-recurring for some $U_D \geq 1, U_C \in (0,1)$. This allows us to state our main results for reconfigurable graphs as Theorems 14 and 15. The theorems we have shown for the static version of the problem (Theorems 3 and 4) are special cases of these two theorems.

▶ **Theorem 14.** *Starting from any configuration satisfying the state invariant over a reconfigurable graph, if $|\mathcal{W}_T| = 0$, then all agents will reach and remain in the UNAWARE state in $O(n^2)$ expected iterations, after time T.*

▶ **Theorem 15.** *Starting from any configuration satisfying the state invariant over a reconfigurable graph, if $|\mathcal{W}_T| > 0$ and the reconfiguration adversary is (U_D, U_C)-recurring, then all agents will reach and stay in the* Aware *state in $O\left(n^6 \log n + \frac{nU_D}{1-U_C}\right)$ expected iterations, after time T.*

In particular, if every graph G_t is connected, then the reconfiguration adversary is $\left(1, \left(1 - \frac{1}{n}\right)\right)$-recurring by setting $D_k = C_k = 1$ (as constant random variables) for all k, and we have the following corollary:

▶ **Corollary 16.** *Starting from any configuration satisfying the state invariant over a reconfigurable graph, if $|\mathcal{W}_T| > 0$ and every G_t is connected, then all agents will reach and stay in the* Aware *state in $O(n^6 \log n)$ expected iterations, after time T.*

Obviously, if G is a static connected graph, it falls as a special case of the corollary; a tighter analysis allowed us to prove the $O(n^5)$ expected convergence bound in Theorem 4.

The following two lemmas, which are analogous to Lemmas 8 and 9 but for reconfigurable graphs, are sufficient to show Theorem 14 (the case for $|\mathcal{W}_T| = 0$). The proof of Lemma 17 is identical to the proof of Lemma 8, so we only show Lemma 18.

▶ **Lemma 17.** *We start from a configuration satisfying the state invariant over a reconfigurable graph with no more than w witnesses at any point. Then the expected number of steps before we reach a configuration with no residual components is at most $2n^2/(1 - wp)$.*

▶ **Lemma 18.** *We start from a configuration satisfying the state invariant over a reconfigurable graph, and assume that no agent will be removed from the witness set from the current point on. If there are currently no residuals, then a residual cannot be generated.*

Proof. From the proof of Lemma 9, we know that state changes of agents do not generate a new residual. Valid reconfigurations of agents also cannot generate a new residual, as from Lemma 12, the state invariant always holds, so all components will always have an agent in state $\mathcal{A}_{\{C\}}$, $\mathcal{A}_{\{W\}}$ or $\mathcal{A}_{\{A,W\}}$. Reconfigurations cannot change the fact that there will be no agent of state $\mathcal{A}_{\{C\}}$, and that all agents in states $\mathcal{A}_{\{W\}}$ or $\mathcal{A}_{\{A,W\}}$ will be witnesses. ◀

The key result we will show is Lemma 19, a loose polynomial-time upper bound for the amount of time before the next agent switches to the Aware state. This gives a polynomial time bound for all agents switching to the Aware state when $|\mathcal{W}_T| > 0$, implying Theorem 15.

▶ **Lemma 19.** *We start from a configuration satisfying the state invariant over a (U_D, U_C)-recurring reconfigurable graph, and assume that there are no residuals, the witness set is nonempty, and no agent will be removed from the witness set from the current point on. Then the expected number of iterations before the next agent switches from the* Unaware *to the* Aware *state is at most $O\left(n^5 \log n + \frac{U_D}{1-U_C}\right)$.*

Proof Sketch. This proof largely follows the proof of Lemma 10, with some modifications to allow for reconfigurability. In the interest of space however, we will provide only a summary of the key ideas and calculations that go into the proof (which is available in the full version of the paper). Assuming that every witness is already in the Aware state with the witness flag set, we upper bound the expected time it takes for all Aware agents to obtain an alert token, followed by the time it takes for an agent to be activated while adjacent to an alert token and switch to the Aware state.

To bound the expected time for all AWARE agents to obtain an alert token, we apply the same strategy, marking an agent without an alert token and passing around the mark until it lands on a witness. We make use of the result of [3, 13], which states that the expected hitting time of the d_{max}-random walk on a connected evolving graph controlled by an *oblivious adversary* is $O(n^3 \log n)$ [13]. This corresponds to $O(n^4 \log n)$ iterations in expectation to generate a new alert token, which gives an upper bound of $O(n^5 \log n)$ iterations in expectation before all agents carry alert tokens.

Two complications arise however when applying this result - the requirement for the adversary controlling the dynamic graph to be oblivious and the requirement that the dynamic graph remains connected. The first issue is dealt with with an observation that with no change in the behavior group vector (as long as no new agent switches to the AWARE state), the sequence of graphs generated by agent movement is independent of the movement of alert tokens. The second issue is resolved with the observation that even though each graph $G_t[A]$ induced by the set of AWARE agents is not connected, the state invariant and the lack of residuals ensure that each of its connect components contains a witness. The witnesses are linked with imaginary edges to connect the graph, which we can do as we only care about the amount of time before the mark lands on any witness.

A new AWARE agent is added when an active UNAWARE agent is activated. The probability of adding a new AWARE agent on a given iteration with k active agents is thus $\frac{k}{n}$, as each of the n agents are activated with equal probability. Thus, if we denote by the random sequence $K_1, K_2, K_3, \ldots$ the number of active agents on each iteration following T_{full} (including T_{full}), we get the following expression for the expected value of X, which we use to denote the number of iterations following T_{full} before a new AWARE agent is added:

$$\begin{aligned}
&\mathbb{E}[X|K_1, K_2, K_3, \ldots] = \sum_{x=0}^{\infty} Pr\left(X > x | K_1, K_2, K_3, \ldots\right) \\
&= 1 + \sum_{x=1}^{\infty} \prod_{i=1}^{x} \left(1 - \frac{K_i}{n}\right) \le 1 + \sum_{x=1}^{\infty} y^{\sum_{i=1}^{x} K_i} \text{ where } y := \left(1 - \frac{1}{n}\right) \in (0,1) \\
&= 1 + \underbrace{y^{K_1} + y^{K_1+K_2} + \ldots + y^{\sum_{i=1}^{D_1} K_i}}_{D_1 \text{ terms}} + \underbrace{y^{C_1 + K_{D_1+1}} + \ldots + y^{C_1 + \sum_{i=D_1+1}^{D_1+D_2} K_i}}_{D_2 \text{ terms}} + \ldots \\
&\le D_1 + D_2 \cdot y^{C_1} + D_3 \cdot y^{C_1+C_2} + \ldots \text{ (as } \sum_{i=1}^{D_1+D_2+\ldots+D_x} K_i = C_{x+1} \text{ for all } x).
\end{aligned}$$

Thus, via the law of total expectation, we have

$$\begin{aligned}
\mathbb{E}[X] \le \sum_{i=1}^{\infty} \mathbb{E}[D_i y^{\sum_{j=1}^{i-1} C_j}] &\le \sum_{i=1}^{\infty} \mathbb{E}\left[y^{\sum_{j=1}^{i-1} C_j} \mathbb{E}[D_i | C_1, C_2, \ldots, C_{i-1}]\right] \\
&\le \sum_{i=1}^{\infty} U_D \mathbb{E}\left[y^{\sum_{j=1}^{i-2} C_j} \mathbb{E}[y^{C_{i-1}} | C_1, C_2, \ldots, C_{i-2}]\right] \\
&\le \ldots \le \sum_{i=1}^{\infty} U_D (U_C)^{i-1} = \frac{U_D}{1 - U_C}.
\end{aligned}$$

This gives us an expected time bound of $O\left(n^5 \log n + \frac{U_D}{1-U_C}\right)$ to add a new AWARE agent. ◀

5 Foraging via self-induced phase changes

Recall that in *the foraging problem*, we have “ants” (agents) that may initially be searching for “food” (stimuli, which can be any resource in the environment, like an energy source); once a food source is found, the ants that have learned about the food source start informing other ants, allowing them to switch their behaviors from the *search mode* to the *gather mode*, that leads them to start to gather around the food source to consume the food. Once the source is depleted, the ants closer to the depleted source start a broadcast wave, gradually informing other ants that they should restart the search phase again by individually switching their states. The foraging problem is very general and has several fundamental application domains, including search-and-rescue operations in swarms of nano- or micro-robots; health applications (e.g., a collective of nano-sensors that could search for, identify, and gather around a foreign body to isolate or consume it, then resume searching, etc.); and finding and consuming/deactivating hazards in a nuclear reactor or a minefield.

Our model for foraging is based on the *geometric Amoebot model* for programmable matter [11, 12]. We have n anonymous agents occupying distinct sites on a $\sqrt{N} \times \sqrt{N}$ piece of the triangular lattice with periodic boundary conditions. These agents have constant-size memory, and have no global orientation or any other global information beyond a common chirality. Agents are activated with individual Poisson clocks, upon which they may move to adjacent unoccupied sites or change states, operating under similar constraints to the dynamic stimuli problem. An agent may only communicate with agents occupying adjacent sites of the lattice. Food sources may be placed on any site of the lattice, removed, or shifted around at arbitrary times, possibly adversarially, and an agent can only observe the presence of the food source while occupying the lattice site containing it. This model can be viewed as a high level abstraction of the (canonical) *Amoebot model* [11, 12] under a *random sequential scheduler*, where at most one agent would be active at any point in time. One should be able to port the model and algorithms presented in this paper to the Amoebot model; however a formal description on how this should be done is beyond the scope of this paper.

At any point of time, there are two main states an agent can be in, which, at the macro-level, are to induce the collective to enter the *search* or *gather* modes respectively. When in *search mode*, agents move around in a process akin to a simple exclusion process, where they perform a random walk while avoiding two agents occupying the same site. Agents enter the *gather mode* when food is found and this information is propagated in the system, consequently resulting in the system compressing around the food (Figure 5).

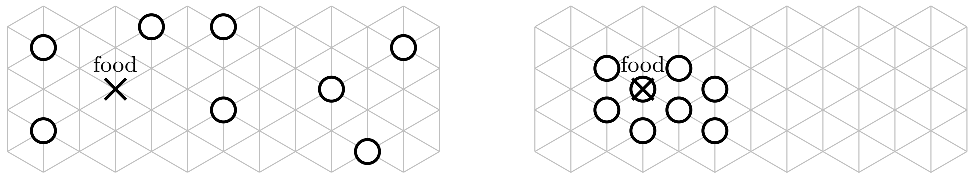


Figure 5 In the diagram on the left, a food source is placed on a lattice site. The diagram on the right illustrates a desired configuration, where all agents have gathered in a low perimeter configuration around the food source. If the food source is later removed, the agents should once again disperse, returning to a configuration like the figure on the left.

In the nonadaptive setting, Cannon *et al.* [5] designed a rigorous compression/expansion algorithm for agents that remain simply connected throughout execution, where a single parameter λ determines a system-wide phase: A small λ, namely $\lambda < 2.17$, provably corresponds to the search mode, which is desirable to search for food, while large λ, namely

$\lambda > 2 + \sqrt{2}$, corresponds to the gather mode, desirable when food has been discovered. Likewise, Li *et al.* [25] show a very similar bifurcation based on a bias parameter λ in the setting when the agents are allowed to disconnect and disperse throughout the lattice. Our goal here is to perform a system-wide adjustment in the bias parameters when one or more agents notice the presence or depletion of food to induce the appropriate global coordination to provably transition the collective between macro-modes when required. Informally, one can imagine individual agents adjusting their λ parameter to be high when they are fed, encouraging compression around the food, and making λ small when they are hungry, promoting the search for more food. A configuration is called *α-compressed* if the perimeter (measured by the length of the closed walk around its boundary edges) is at most $\alpha\, p_{\min}(n)$, for some constant $\alpha > 1$, where $p_{\min}(n)$ denotes the minimum possible perimeter of a connected system with n agents, which is the desired outcome of the gather mode.

Adaptive α-compression. We present the first rigorous local distributed algorithm for the foraging problem: The *Adaptive α-Compression* algorithm is based on the stochastic compression algorithm of [5], addressing a geometric application of the dynamic stimuli problem, where the AWARE state represents the “gather” mode, and the UNAWARE state represents the “search” mode. A witness is an agent that occupies the same lattice site as the food source. The underlying dynamic graph used by the Adaptive Stimuli Algorithm is given by the adjacency graph of the agents - two agents share an edge on the graph if they occupy adjacent sites of the lattice. As agents move around to implement behaviors like gathering and searching, their neighbor sets will change. The movement of agents thus reconfigures and oftentimes even disconnects our graph.

In this algorithm, agents in the UNAWARE (search) behavior group execute movements akin to a simple exclusion process (EXECUTE-SEARCH) while agents in the MOBILE (gather) behavior group execute moves of the compression algorithm (EXECUTE-GATHER) in [5]. We focus on the compression algorithm run by the AWARE agents. In a simple exclusion process, a selected agent picks a direction at random, and moves in that direction if and only if the immediate neighboring site in that direction is unoccupied. In the compression algorithm [5] on the other hand, a selected AWARE agent first picks a direction at random to move in, and if this move is a valid compression move (according to Definition 20), the agent moves to the chosen position with the probability given in Definition 21. With a (far-from-trivial) modification of the analysis in [5] to account for the stationary witness agent, we show that in the case of a single food source, the “gather” movements allow the agents to form a low perimeter cluster around the food. We present the following definitions, adapted from [5] to the set of AWARE agents running the compression algorithm:

▶ **Definition 20** (Valid Compression Moves [5]). *Denote by $N_{\mathcal{A}}(\ell)$ and $N_{\mathcal{A}}(\ell')$ the sets of AWARE neighbors of ℓ and ℓ' respectively and $N_{\mathcal{A}}(\ell \cup \ell') := N_{\mathcal{A}}(\ell) \cup N_{\mathcal{A}}(\ell') \setminus \{\ell, \ell'\}$. Consider the following two properties:*

Property 1: $|N_{\mathcal{A}}(\ell) \cap N_{\mathcal{A}}(\ell')| \geq 1$ *and every agent in $N_{\mathcal{A}}(\ell \cup \ell')$ is connected to an agent in $N_{\mathcal{A}}(\ell) \cap N_{\mathcal{A}}(\ell')$ through $N_{\mathcal{A}}(\ell \cup \ell')$.*

Property 2: $|N_{\mathcal{A}}(\ell) \cap N_{\mathcal{A}}(\ell')| = 0$*, ℓ and ℓ' each have at least one neighbor, all agents in $N_{\mathcal{A}}(\ell) \setminus \{\ell'\}$ are connected by paths within the set, and all agents in $N_{\mathcal{A}}(\ell') \setminus \{\ell\}$ are connected by paths within the set.*

We say the move from ℓ to ℓ' is a valid compression move *if it satisfies both properties, and $N_{\mathcal{A}}(\ell)$ contains fewer than five aware state agents.*

Algorithm 2 Adaptive α-Compression.

```
procedure ADAPTIVE-ALPHA-COMPRESSION($u$)
    $q \leftarrow$ Random number in $[0, 1]$
    $u.isWitness \leftarrow$TRUE if $u$ observes the food source, else $u.isWitness \leftarrow$FALSE
    if $q \leq \frac{1}{2}$ then                    ▷ With probability $\frac{1}{2}$, make a state update
        ADAPTIVE-STIMULI-ALGORITHM(u)
    else                                            ▷ With probability $\frac{1}{2}$, make a move
        if $u.isWitness$ or $u.state \in \{\mathcal{A}_{\{W\}}, \mathcal{A}_{\{A,W\}}, \mathcal{A}_{\{C\}}\}$ then    ▷ IMMOBILE agent
            Do nothing
        else if $u.state \in \{\mathcal{A}_{\emptyset}, \mathcal{A}_{\{A\}}\}$ then          ▷ MOBILE agent
            EXECUTE-GATHER(u)
        else if $u.state = \mathcal{U}$ then                                  ▷ UNAWARE agent
            EXECUTE-SEARCH(u)

procedure EXECUTE-GATHER($u$)
    $d \leftarrow$ Random direction in $\{0, 1, 2, 3, 4, 5\}$
    $\ell \leftarrow$ Current position of $u$
    $\ell' \leftarrow$ Neighboring lattice site of $u$ in direction $d$
    if Moving $u$ from $\ell$ to $\ell'$ is a valid compression move (Definition 20) then
        $p \leftarrow$ Random number in $[0, 1]$
        $d(u) \leftarrow$ number of neighboring agents of $u$ if $u$ were at position $\ell$
        $d'(u) \leftarrow$ number of neighboring agents of $u$ if $u$ were at position $\ell'$
        if $p \leq \lambda^{d'(u)-d(u)}$ then
            Move $u$ to position $\ell'$           ▷ Movements reconfigure the adjacency graph

procedure EXECUTE-SEARCH($u$)
    $d \leftarrow$ Random direction in $\{0, 1, 2, 3, 4, 5\}$
    $\ell' \leftarrow$ Neighboring lattice site of $u$ in direction $d$
    if Moving $\ell'$ is an unoccupied lattice site then
        Move $u$ to position $\ell'$               ▷ Movements reconfigure the adjacency graph
```

▶ **Definition 21** (Transition probabilities [5]). *Fix $\lambda > 2 + \sqrt{2}$, as sufficient for α-compression. An agent u transitions through a valid movement with Metropolis-Hastings [28] acceptance probability* $\min\{1, \lambda^{e(\sigma')-e(\sigma)}\}$*, where σ and σ' are the configurations before and after the movement, and $e(\cdot)$ represents the number of edges between* AWARE *state agents in the configuration.*

Note that even though $e(\cdot)$ is a global property, the difference $e(\sigma') - e(\sigma)$ can be computed locally (within two hops in the lattice, or through expansions in the Amoebot model [11, 12]), as it is just the change in the number of AWARE neighbors of u before and after its movement.

The condition for valid compression moves is notable as it keeps a component of aware agents containing the witness agent simply connected (connected and hole-free), which is crucial to the proof in [5] that a low perimeter configuration, in our case around the food source, will be obtained in the long term. We also show that these valid compression moves are locally connected (as per Definition 11), a sufficient condition for the reconfigurable dynamic stimuli problem to apply.

We first prove that the Markov chain representing the compression moves (EXECUTE-GATHER) is connected. The proof builds upon the ergodicity argument in [5]; however, the addition of a single stationary agent, the witness, in our context makes this proof significantly more complex, and we defer it to the full version of the paper.

▶ **Lemma 22.** *Consider connected configurations of agents on a triangular lattice with a single agent v that cannot move. There exists a sequence of valid compression moves that transforms any connected configuration of agents into any simply connected configuration of the agents while keeping v stationary.*

We may now state our main results, which verify the correctness of Adaptive α-Compression.

▶ **Theorem 23.** *If no food source has been identified for sufficiently long, then within an expected $O(n^2)$ steps, all agents will reach and remain in the* Unaware *state and will converge to the uniform distribution of nonoverlapping lattice positions.*

▶ **Theorem 24.** *If at least one food source exists and remains in place for long enough, then within $O(n^6 \log n + N^2 n)$ steps in expectation, all agents will reach and remain in the* Aware *state, and each component of* Aware *agents will contain a food source. In addition, if there is only one food source, the agents will converge to a configuration with a single α-compressed component around the food, for any constant $\alpha > 1$, with all but an exponentially small probability, for a large enough lattice region.*

The Adaptive α-Compression Algorithm fits the requirements of the reconfigurable dynamic stimuli model. In particular, the information X_t available to the reconfiguration adversary corresponds to the configuration of the lattice, and the graph G_t represents the adjacency of agents on the lattice at that time t. To show that a sequence of valid Aware agent movements in the Adaptive α-Compression Algorithm, which determine the configurations of $G_1, G_2, \ldots$, can be modeled via a valid reconfiguration adversary $\mathcal{X}$, we need to show that the reconfigurations resulting from the Execute-Gather procedure must be locally connected.

▶ **Lemma 25.** *The movement behavior of Adaptive α-Compression is locally connected.*

Proof. We only need to show that the Execute-Gather procedure maintains local connectivity (Definition 11). This is true as when reconfiguring an Aware agent u with Aware neighbor set $N_{\mathcal{A}}(u)$, we only allow valid compression moves (Definition 20) to be made. In the case of Property 1, all agents in $N_{\mathcal{A}}(u)$ will still have paths to u in G' through S. In the case of Property 2, all agents in $N_{\mathcal{A}}(u)$ will still have paths to each other within $G'[N_{\mathcal{A}}(u)]$, despite no longer having local paths to u. The agent u will have at least one Aware state neighbor after the move as this is a requirement of Property 2. ◀

As this is an instance of the reconfigurable dynamic stimuli problem, Theorem 23 follows immediately from Theorem 14. To show Theorem 24 however, we need to show polynomial recurring rates by arguing that the Unaware state agents following a simple exclusion process will regularly come into contact with the clusters of Aware state agents around the food sources.

▶ **Lemma 26.** *The movement behavior defined in the Adaptive α-Compression Algorithm is (U_D, U_C)-recurring with $U_D = 2N^2 + \frac{2}{n} + 1$ and $U_C = \frac{2}{3}$.*

Proof Sketch. In the interest of space, we give only a brief summary of the proof (which is available in the full version of the paper). We define the random sequences $(D_1, D_2, D_3, \ldots)$ and $(C_1, C_2, C_3, \ldots)$ by dividing the time steps after the starting iteration t into batches, where the k^{th} batch would take D_k iterations and would see C_k active agents over its duration.

A batch ends (and the next batch starts) when the agent movement places an Unaware agent u next to an Aware agent v, then attempts a movement of u or v after that. The duration D_k of the k^{th} batch can be computed with the hitting time of a simple exclusion process over the triangular lattice, plus a geometric random variable representing the number of iterations taken to select u or v after that. This gives us a uniform upper bound of $2N^2n + \frac{2}{n} + 1$ for $\mathbb{E}[D_k | D_1, D_2, \dots D_{k-1}, C_1, C_2 \dots C_{k-1}]$ for each batch $k \in \{1, 2, \dots\}$.

The number of active agents C_k within batch k is at least the number of iterations between the first time within the batch that an Unaware agent moves next to an Aware agent and the end of the batch. This is shown to stochastically dominate a geometric random variable Y with success probability p_Y, which we show in the full paper to be:

$$p_Y = \sum_{i=0}^{\infty} \frac{1}{2} \cdot \left(\frac{1}{2}\right)^i \left(1 - \left(1 - \frac{2}{n}\right)^i\right) = 1 - \frac{1}{2} \sum_{i=0}^{\infty} \left(\frac{1 - 2/n}{2}\right)^i = \frac{2}{n}\left(\frac{1}{1 + 2/n}\right)$$

As C_k stochastically dominates Y and $(1 - \frac{1}{n})^x$ is a decreasing function of x, we have:

$$\begin{aligned}\mathbb{E}\left[\left(1 - \frac{1}{n}\right)^{C_k} | D_1, D_2, \dots D_{k-1}, C_1, C_2 \dots C_{k-1}\right] &\le \mathbb{E}\left[\left(1 - \frac{p}{n}\right)^{Y}\right] \\ &= \sum_{y=0}^{\infty} \left(1 - \frac{1}{n}\right)^y p_Y \left(1 - p_Y\right)^y \\ &= p_Y \frac{1}{1 - (1 - 1/n)(1 - p_Y)} = \frac{2}{3}. \quad \blacktriangleleft\end{aligned}$$

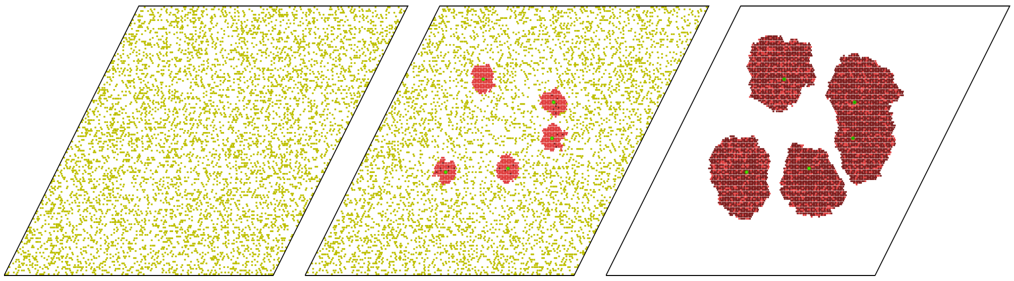
To show the first half of Theorem 24, we start from the first iteration beyond which no additional changes in the positions (or existence) of the food sources occur. We first show that if there is at least one food source, it will be found. As long as no food source has been found, there will be no witnesses, so every agent will return to the Unaware state by Theorem 14. Agents in the Unaware state move randomly following a simple exclusion process. Using the hitting time of a simple random walk on a regular graph (the triangular lattice) of N sites, we have a simple upper bound of $O(N^2 n)$ iterations in expectation before some agent finds a food source and becomes a witness.

From then on, there will be at least one witness, and the witness set can only be augmented, not reduced, as the other agents potentially find additional food sources, and since the agents already sitting on food sources are no longer allowed to move. As agent movement behaviors are recurring with polynomial bounds (Lemma 26), the reconfigurable Adaptive Stimuli Algorithm applies, yielding a polynomial bound on the expected number of iterations before all agents have switched to the Aware state with no residuals. Additionally, due to the maintenance of the state invariant and as there are no residuals, every component of Aware agents will contain at least one witness, meaning that every cluster of agents will be around some food source. This gives us the first part of Theorem 24.

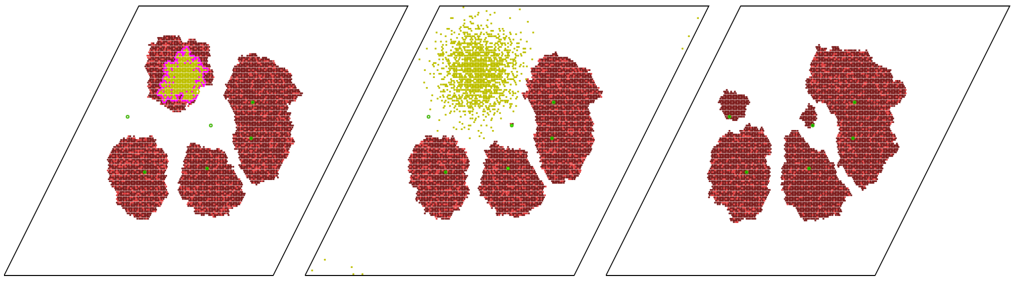
The second half of Theorem 24 states that a low perimeter (α-compressed) configuration is achievable in the case of a single food source. As the Markov chain representing the compression moves is irreducible (Lemma 22), the results of [5] guarantee that for any $\alpha > 1$, there exists a sufficiently large constant λ such that at stationarity, the perimeter of the cluster is at most α times its minimum possible perimeter with high probability.

6 Simulations of the Adaptive α-Compression Algorithm

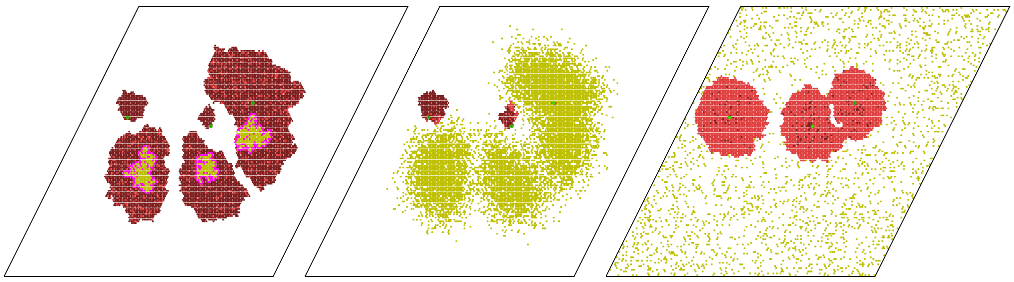
We demonstrate a simulation of the Adaptive α-Compression algorithm with 5625 agents in a 150×150 triangular lattice with periodic boundary conditions. Multiple food sources (stimuli) are placed and moved around to illustrate the gather and search phases. This simulation is shown as a sequence of 12 images in chronological order in Figure 6.



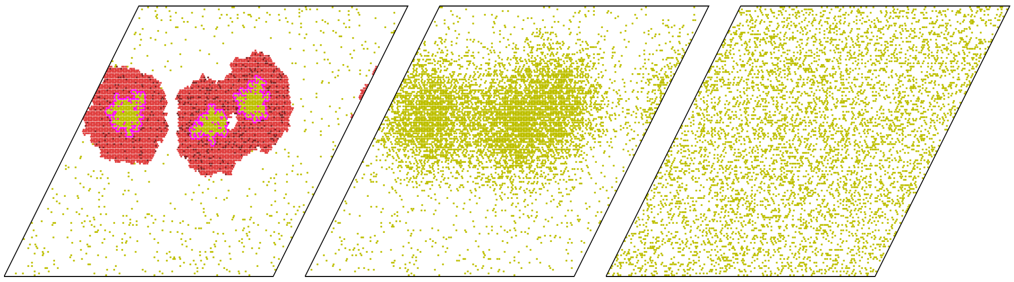
(a) All agents initially unaware. Five food sources added, all agents eventually switch to the AWARE state.



(b) One food source removed, two new added. A cluster disperses and agents rejoin other clusters.



(c) Three food sources removed, clusters disperse and gather around the remaining food sources.



(d) All food sources removed, agents disperse and converge to a uniform distribution over the lattice.

Figure 6 Simulation of Adaptive α-Compression with multiple food sources. The images are in chronological order. UNAWARE agents are yellow, AWARE agents are red (darker red if they have an alert token), agents with the all-clear token are purple, and food sources are green.

References

1 Simon Alberti. Organizing living matter: The role of phase transitions in cell biology and disease. *Biophysical journal*, 14, 2018.

2 Marta Andrés Arroyo, Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A stochastic approach to shortcut bridging in programmable matter. In *23rd International Con- ference on DNA Computing and Molecular Programming (DNA)*, pages 122–138, 2017.

3 Chen Avin, Michal Koucký, and Zvi Lotker. Cover time and mixing time of random walks on dynamic graphs. *Random Structures & Algorithms*, 52(4):576–596, 2018.

4 Sarah Cannon, Joshua J. Daymude, Cem Gökmen, Dana Randall, and Andréa W. Richa. A local stochastic algorithm for separation in heterogeneous self-organizing particle systems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, pages 54:1–54:22, 2019.

5 Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A Markov chain algorithm for compression in self-organizing particle systems. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 279–288, 2016.

6 Arnaud Casteigts. Finding structure in dynamic networks, 2018. arXiv:1807.07801.

7 Bernard Chazelle. The convergence of bird flocking. *J. ACM*, 61(4), 2014.

8 Andrea Clementi, Riccardo Silvestri, and Luca Trevisan. Information spreading in dynamic graphs. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, pages 37–46, 2012.

9 Nikolaus Correll and Alcherio Martinoli. Modeling and designing self-organized aggregation in a swarm of miniature robots. *The Int'l J. of Robotics Research*, 30(5):615–626, 2011.

10 Paolo Dario, Renzo Valleggi, Maria Chiara Carrozza, M. C. Montesi, and Michele Cocco. Microactuators for microrobots: a critical survey. *Journal of Micromechanics and Microengineering*, 2(3):141–157, 1992.

11 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The Canonical Amoebot Model: Algorithms and Concurrency Control. In *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:19, 2021.

12 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The canonical amoebot model: Algorithms and concurrency control. *Distributed Computing*, 2023. To appear.

13 Oksana Denysyuk and Luís Rodrigues. Random walks on evolving graphs with recurring topologies. In *Distributed Computing*, pages 333–345. Springer Berlin Heidelberg, 2014.

14 Michael Dinitz, Jeremy T. Fineman, Seth Gilbert, and Calvin Newport. Smoothed analysis of information spreading in dynamic networks. In Christian Scheideler, editor, *36th International Symposium on Distributed Computing (DISC)*, volume 246 of *LIPIcs*, pages 18:1–18:22, 2022.

15 Chinmoy Dutta, Gopal Pandurangan, Rajmohan Rajaraman, Zhifeng Sun, and Emanuele Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 717–736. SIAM, 2013.

16 Nazim Fatès. Solving the decentralised gathering problem with a reaction–diffusion–chemotaxis scheme. *Swarm Intelligence*, 4(2):91–115, 2010.

17 Nazim Fatès and Nikolaos Vlassopoulos. A robust aggregation method for quasi-blind robots in an active environment. In *ICSI 2011*, 2011.

18 Simon Garnier, Jacques Gautrais, Masoud Asadpour, Christian Jost, and Guy Theraulaz. Self-organized aggregation triggers collective decision making in a group of cockroach-like robots. *Adaptive Behavior*, 17(2):109–133, 2009.

19 Simon Garnier, Christian Jost, Raphaël Jeanson, Jacques Gautrais, Masoud Asadpour, Gilles Caprari, and Guy Theraulaz. Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In *Advances in Artificial Life*, ECAL '05, pages 169–178, 2005.

20 Bernhard Haeupler and David Karger. Faster information dissemination in dynamic networks via network coding. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 381–390, 2011.

21 Walter Hussak and Amitabh Trehan. On termination of a flooding process. In *Proc. of the 2019 ACM Symp. on Principles of Distributed Computing*, PODC '19, pages 153–155, 2019.

22 Hridesh Kedia, Shunhao Oh, and Dana Randall. A local stochastic algorithm for alignment in self-organizing particle systems. In *Approximation, Randomization, and Combinatorial*

Optimization. Algorithms and Techniques (APPROX/RANDOM 2022), volume 245, pages 14:1–14:20, 2022.

23 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, pages 513–522, 2010.

24 Fabian Kuhn and Rotem Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, 2011.

25 Shengkai Li, Bahnisikha Dutta, Sarah Cannon, Joshua J. Daymude, Ram Avinery, Enes Aydin, Andréa W. Richa, Daniel I. Goldman, and Dana Randall. Programming active granular matter with mechanically induced phase changes. *Science Advances*, 7, 2021.

26 Jintao Liu, Arthur Prindle, Jacqueline Humphries, Marçal Gabalda-Sagarra, Munehiro Asally, Dong-Yeon D. Lee, San Ly, Jordi Garcia-Ojalvo, and Gürol M. Süel. Metabolic co-dependence gives rise to collective oscillations within biofilms. *Nature*, 523(7562):550–554, 2015.

27 Anne E. Magurran. The adaptive significance of schooling as an anti-predator defence in fish. *Annales Zoologici Fennici*, 27(2):51–66, 1990.

28 Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

29 Nathan J. Mlot, Craig A. Tovey, and David L. Hu. Fire ants self-assemble into waterproof rafts to survive floods. *Proc. of the National Academy of Sciences*, 108(19):7669–7673, 2011.

30 Anil Özdemir, Melvin Gauci, Salomé Bonnet, and Roderich Groß. Finding consensus without computation. *IEEE Robotics and Automation Letters*, 3(3):1346–1353, 2018.

31 Arthur Prindle, Jintao Liu, Munehiro Asally, San Ly, Jordi Garcia-Ojalvo, and Gürol M. Süel. Ion channels enable electrical communication in bacterial communities. *Nature*, 527(7576):59–63, 2015.

32 Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, pages 10–20, 2005.

33 William Savoie, Sarah Cannon, Joshua J. Daymude, Ross Warkentin, Shengkai Li, Andréa W. Richa, Dana Randall, and Daniel I. Goldman. Phototactic supersmarticles. *Artificial Life and Robotics*, 23(4):459–468, 2018.

34 Thomas C. Schelling. Dynamic models of segregation. *The Journal of Mathematical Sociology*, 1(2):143–186, 1971.

35 Onur Soysal and Erol Şahin. Probabilistic aggregation strategies in swarm robotic systems. In *Proceedings 2005 IEEE Swarm Intelligence Symposium*, SIS 2005, pages 325–332, 2005.

36 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1):263–272, 1991.

37 David H. Wolpert. The stochastic thermodynamics of computation. *Journal of Physics A: Mathematical and Theoretical*, 52(19):193001, 2019.

38 Hui Xie, Mengmeng Sun, Xinjian Fan, Zhihua Lin, Weinan Chen, Lei Wang, Lixin Dong, and Qiang He. Reconfigurable magnetic microrobot swarm: Multimode transformation, locomotion, and manipulation. *Science Robotics*, 4(28):eaav8006, 2019.