A Game-theoretic Formulation of the Homogeneous Self-Reconfiguration Problem

Daniel Pickem[1], Magnus Egerstedt[2], and Jeff S. Shamma[3]

Abstract— In this paper we formulate the homogeneous two- and three-dimensional self-reconfiguration problem over discrete grids as a constrained potential game. We develop a game-theoretic learning algorithm based on the Metropolis-Hastings algorithm that solves the self-reconfiguration problem in a globally optimal fashion. Both a centralized and a fully decentralized algorithm are presented and we show that the only stochastically stable state is the potential function maximizer, i.e. the desired target configuration. These algorithms compute transition probabilities in such a way that even though each agent acts in a self-interested way, the overall collective goal of self-reconfiguration is achieved. Simulation results confirm the feasibility of our approach and show convergence to desired target configurations.

I. INTRODUCTION

Self-reconfigurable systems are comprised of individual agents which are able to connect to and disconnect from one another to form larger functional structures. These individual agents or modules can have distinct capabilities, shapes, or sizes, in which case we call it a heterogeneous system (for example [6]). Alternatively, modules can be identical and interchangeable, which describes a homogeneous system (see [9]). In this paper, we will present algorithms that reconfigure homogeneous systems and treat self-reconfiguration as a two- and three-dimensional coverage problem.

Self-reconfiguration is furthermore understood to solve the following problem. Given an initial geometric arrangement of cubes (called a configuration) C_I and a desired target configuration C_T, the solution to the self-reconfiguration problem is a sequence of primitive cube motions that re-shapes/reconfigures the initial into the target configuration (see Fig. 1). The problem setup is then the following.

- The environment $\mathcal{E}$ is a finite two- or three-dimensional discrete grid, i.e. $\mathcal{E} \subseteq \mathbb{Z}^2$ or $\mathcal{E} \subseteq \mathbb{Z}^3$.
- N agents $P = \{1, 2, \ldots, N\}$ move in discrete steps through that grid.
- Each agent has a restricted action set $\mathcal{R}_i$ which contains only a subset of all its possible actions A_i.
- An agent's utility $U_i(a \in A)$ is inversely proportional to the distance to the target configuration.

[1]D. Pickem is a Ph.D. student in Robotics, Georgia Institute of Technology, Atlanta, USA `daniel.pickem@gatech.edu`

[2]M. Egerstedt is with the Faculty of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA `magnus@gatech.edu`

[3]J. S. Shamma is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA `shamma@gatech.edu`, and with King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia `jeff.shamma@kaust.edu.sa`.

Many approaches to self-reconfiguration have been presented in the literature, each with certain short-comings. There have been centralized solutions [18], distributed solutions that required a large amount of communication [6] or precomputation [5], approaches that were either focused on locomotion [3] or functional target shape assemblies [10]. Distributed approaches have often relied on precomputation of rulesets [7], policies [5], or entire sets of paths/folding schemata of agents [4].

In this paper, we present a fully decentralized approach to homogeneous self-reconfiguration for which no central decision maker is required. Our method guarantees convergence to the target configuration even though each agent acts as a purely self-interested individual decision maker with local information only. Decision making requires no (in the two-dimensional case) or limited communication (in the three-dimensional case).

The rest of this paper is organized as follows. Section II discusses relevant related work. Section III presents the system setup and theoretical formulation of the problem. In Section IV we discuss the completeness of deterministic reconfiguration, which is used in Section V to prove the existence of a unique potential function maximizer. In Section VI we decentralize the stochastic algorithm and present simulation results in Section VII

II. RELATED WORK

In this section we want to highlight decentralized approaches in the literature that bear resemblance to methods presented in this paper. Especially relevant to the presented results in this paper are homogeneous self-reconfiguration approaches such as [3], and [9], which employ cellular automata and manually designed local rules to model the system. Similar work in [5] shows an approach for locomotion through self-reconfiguration represented as a Markov decision process with state-action pairs computed in a distributed fashion. The presented algorithms are decentralized but only applicable to locomotion and not the assembly of arbitrary configurations.

Precomputed rules are also used in [7], where a decentralized approach is presented based on graph grammatical rules. Graph grammars for the assembly of arbitrary acyclic target configurations are automatically generated. Whereas these approaches are able to assemble arbitrary target configurations, they rely on precomputing rulesets for every target configuration.

The algorithms presented in this paper are inspired by the coverage control literature, specifically game-theoretic
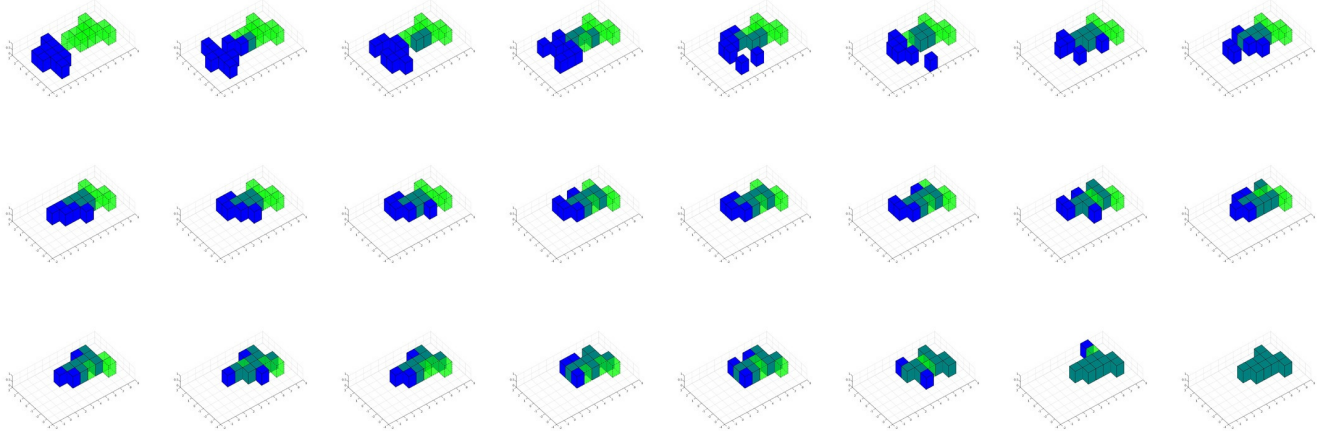
Fig. 1: Example of self-reconfiguration sequence from a random 2D configuration (left) to another random 2D configuration (right). Approximately every 20th time step is shown.

formulations such as [1], [11], and [20]. Both static sensor coverage as well as dynamic coverage control with mobile agents are discussed. Note, however, that agents in these papers are limited to movement in two dimensions and operate with a different motion model and most importantly different constraints.

We address these problems with a decentralized algorithm that does not rely on precomputed rulesets, can be used for locomotion as well as the assembly of arbitrary two- and three-dimensional shapes, can handle changing environment conditions as well as changing target configurations, and is scalable to large number of modules due to its decentralized nature.

III. PROBLEM FORMULATION

In this work, we represent agents as cubic modules that move through a discrete lattice or environment $\mathcal{E} = \mathbb{Z}^d$ in discrete steps.[1] Without loss of generality these cubes have unit dimension. Therefore, an agent's current state or action a_i (in a game-theoretic sense) is an element $a_i \in \mathbb{Z}^d$. Note that an agent's action is equivalent to its position in the lattice. Cubes can be thought of as geometric embeddings of the agents in our system. Therefore, a configuration $\mathcal{C}$ composed of N agents is a subset of the representable space $\mathbb{Z}^{dN}$ (see [16]). Moreover, we will deal with homogeneous configurations, in which all agents have the same properties and are completely interchangeable.

A. Motion Model

In the sliding cube model (see [3], [15], [16], [18]), a cube is able to perform two primitive motions - a sliding and a corner motion. In general, a motion specifies a translation along coordinate axes and is represented by an element $m \in \mathbb{Z}^d$. A sliding motion is characterized by $\|m_s\|_{L_1} = 1$, i.e.

<hr>

[1]Since we present two- and three-dimensional self-reconfiguration, throughout this paper, the dimensionality d will be $d \in \{2, 3\}$.

$m_{s,i} = 1$ for one and only one coordinate $i \in 1, \ldots d$, which translates a cube along one coordinate axis. A corner motion on the other hand is defined by $\|m_c\|_{L_1} = 2$ such that $m_{c,i} = 1$ for exactly two coordinates $i \in 1, \ldots d$, which translates a cube along two dimensions.

B. Game-theoretic Formulation

In this section we formulate the homogeneous self-reconfiguration as a potential game (see [14]), which is a game structure amenable to globally optimal solutions. Generally, a game is specified by a set of players $i \in P = \{1, 2, \ldots, N\}$, a set of actions $\mathcal{A}_i$ for each player, and a utility function $U_i(a) = U_i(a_i, a_{-i})$ for every player i. In this notation, a denotes a joint action profile $a = (a_1, a_2, \ldots, a_N)$ of all N players, and a_{-i} is used to denote the actions of all players other than agent i.

In a constrained game, the actions of agents are constrained through their own and other agents' actions. In other words, given an action set $\mathcal{A}_i$ for agent i, only a subset $\mathcal{R}_i(a)) \subset \mathcal{A}_i$ is available to agent i. A constrained potential game is furthermore defined as follows.

Definition 1: A *constrained exact potential game* (see [20]) is a tuple $G = (\mathcal{P}, \mathcal{A}, \{U_i(.)\}_{i \in \mathcal{P}}, \{R_i(.)\}_{i \in \mathcal{P}}, \Phi(A))$, where

- $\mathcal{P} = \{1, \ldots, N\}$ is the set of N players
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ is the product set of all agents' action sets $\mathcal{A}_i$
- $U_i : A \to \mathbb{R}$ are the agents' individual utility functions
- $R_i : A \to 2^{\mathcal{A}_i}$ is a function that maps a joint action to a restricted action set for agent i

Additionally, the agents' utility functions are aligned with a global objective function or potential $\Phi : \mathcal{A} \to \mathbb{R}$ if for all agents $i \in \mathcal{P}$, all actions $a_i, a_i' \in R_i(a)$, and actions of other agents $a_{-i} \in \prod_{j \neq i} \mathcal{A}_j$ the following is true

$$U_i(a_i', a_{-i}) - U_i(a_i, a_{-i}) = \Phi(a_i', a_{-i}) - \Phi(a_i, a_{-i})$$

The last condition of Def. 1 implies an alignment of agents' individual incentives and the global goal. Therefore, under unilateral change (only agent i changes its action from a_i to a_i') the change in utility for agent i is equivalent to the change in the global potential Φ. This is a highly desirable property since the maximization of all agents' individual utilities yields a maximum global potential. We can now formulate the self-reconfiguration problem in game-theoretic terms and show that it is indeed a constrained potential game.

Definition 2: Game-theoretic self-reconfiguration can be formulated as a constrained potential game, where the individual components are defined as follows:

- The set of players $\mathcal{P} = \{1, 2, \ldots, N\}$ is the set of all N agents in the configuration.
- The action set of each agent $\mathcal{A}_i \subset \mathbb{Z}^d$ is a finite set of discrete lattice positions
- The restricted action sets $R_i(a \in A)$ are computed according to Section III-C.
- The utility function of each agent is $U_i(a) = \frac{1}{\text{dist}(a_i, \mathcal{C}_T) + 1}$. Here, $\mathcal{C}_T$ is the target configuration and $\text{dist}(a_i, \mathcal{C}_T) = \min_{a_j \in \mathcal{C}_T} \|a_i - a_j\|$.
- The global potential $\Phi(a \in A) = \sum_{i \in \mathcal{P}} U_i(a)$.

Note that the utility of an agent is independent of all other agents' actions and depends exclusively on its distance to the target configuration. An agent's action set, however, is constrained by its own as well as other agents' actions. The goal of the game-theoretic self-reconfiguration problem is to maximize the potential function, i.e.

$$\max_{a \in \mathcal{A}} \Phi(a) = \max_{a \in \mathcal{A}} \sum_{i \in \mathcal{P}} U_i(a)$$

This can be interpreted as a coverage problem where the goal of all agents is to cover all positions in the target configuration. Therefore maximizing the potential is equivalent to maximizing the number of agents that cover target positions $a_i \in \mathcal{C}_T$. The following propositions shows that this formulation indeed yields a potential game (a proof is given in [17]).

Proposition 1: The self-reconfiguration problem in Def. 2 constitutes a constrained potential game with $\Phi(a) = \sum_{i \in \mathcal{P}} U_i(a)$ and $U_i(a) = \frac{1}{\text{dist}(a_i, \mathcal{C}_T) + 1}$.

As we will see in Section VI, this potential game structure allows us to derive a decentralized version of the presented learning algorithm.

C. Action Set Computation

A core component of constrained potential games is the computation of restricted action sets. Unlike previous work (see for example [12] and [20]), agents in our setup are constrained not just by their own actions, but also those of others. In this section we present methods for computing restricted action sets that obey motion constraints as well as constraints imposed by other agents.

a) 2D reconfiguration: In the two-dimensional case agents are restricted to motions on the xy-plane. Unlike in previous work (see [15] and [16]) where we required a

configuration to remain connected at all times, in this work, agents are allowed to disconnect from all (or a subset of) other agents. This approach enables agents to separate from and merge with other agents at a later time. To formalize this idea, we first review some graph theoretic concepts.

Definition 3: Let $G = (V, E)$ be the graph composed of N nodes with $V = \{v_1, v_2, \ldots, v_N\}$, where node v_i represents agent or location i. Then G is called the *connectivity graph* of configuration $\mathcal{C}$ if $E = V \times V$ with $e_{ij} \in E$ if $\|a_i - a_j\| = 1$.

This definition implies that two nodes v_i, v_j in the connectivity graph are adjacent, if agent or location i and j are located in neighboring grid cells. Note that a connectivity graph can be computed for any set of grid positions, whether these positions are occupied by agents or not. We furthermore use the notions of paths on graphs and graph connectivity in the usual graph theoretic sense. Note that G is not necessarily connected as (groups of) agents can split off. Based on the connectivity graph G and the current joint action, we now define the function $R_i : \mathcal{A} \to 2^{\mathcal{A}_i}$, which maps from the full joint action set to a restricted action set for agent i and is based on the following two definitions of primitive actions sets.

Definition 4: The set of all currently possible *sliding motions* is $\mathcal{M}_s = \{a_i' \in \mathbb{Z}^d \setminus a_{-i} : \|m_s\|_{L_1} = 1\}$, where $m_s = a_i' - a_i$.

Definition 5: The set of all currently possible *corner motions* is $\mathcal{M}_c = \{a_i' \in \mathbb{Z}^d \setminus a_{-i} : \|m_c\|_{L_1} = 2 \wedge m_{c,j} \in \{0, 1\}\}$, where $j \in [1, \ldots, d]$ and $m_c = a_i' - a_i$.

Note that $\mathcal{M}_s$ and $\mathcal{M}_c$ in Def. 4 and Def. 5 are equally applicable to 2D and 3D. These definitions encode the motion model outlined in Section III-A and allow us to define the restricted action set in two dimensions as follows.

Definition 6: The two-dimensional restricted action set is given by $R_i^{2D}(a) = \mathcal{M}_s \cup \mathcal{M}_c$.

This definition ensures that agent i can only move to unoccupied neighboring grid positions a_i' through sliding or corner motions (or stay at its current position a_i). All other agents replay their current actions a_{-i}.

b) 3D reconfiguration: Whereas in the 2D case agents were allowed to move to all unoccupied neighboring grid cell regardless of connectivity constraints, in the three-dimensional case we introduce the requirement of *groundedness*. An agent is immobile, if executing an action would remove groundedness from any of its neighbors. Groundedness requires a notion of ground plane, which is defined as follows.

Definition 7 (Ground Plane): The *ground plane* is the set $S_{GP} = \{s \in \mathcal{E} : s_z = 0\}$ where $\mathcal{E} \subseteq \mathbb{Z}^3$ and the corresponding connectivity graph is $G_{GP} = (V_{GP}, E_{GP})$ with $e_{ij} \in E_{GP}$ if $\|s_i - s_j\| = 1$.

Note that the ground plane is defined as the xy-plane and its connectivity graph G_{GP} is, by definition, connected. Positions $s \in S_{GP}$ are not allowed to be occupied by agents, therefore $a_i \in A_i \setminus S_{GP} \ \forall i \in \mathcal{P}$. Using the graph G_{GP}, we define $G' = (V', E')$ as $V' = V \cup V_{GP}$ and

$E' = V' \times V'$ such that $e_{ij} \in E'$ if for $v_i, v_j \in V'$ we have $\|a_i - a_j\|_{L_1} = 1$. Note that G' represents the current configuration including the ground plane, and a_i represents an action of an agent or an unoccupied position in the ground plane.

Definition 8 (Groundedness): An agent i is *grounded* if there exists a path on G' from $v_i \in V \subset V'$ to some $v_k \in V_{GP} \subset V'$, where v_i represents agent i in the connectivity graph G (see Def. 3). A configuration $\mathcal{C}$ is *grounded* if every agent $i \in \mathcal{P}$ is grounded.

The idea behind groundedness hints at an embedding of a self-reconfigurable system in the physical world, where agents cannot choose arbitrary positions in the environment. Additionally, like connectivity, groundedness enforces some level of cohesion among the agents, but without incurring costly global connectivity checks. Most importantly, we use the notion of groundedness to prove completeness of deterministic reconfiguration in Section IV.

An agent can verify groundedness in a computationally cheap way through a depth-first search, which is complete and guaranteed to terminate in time proportional to $O(N)$ in a finite space. The notion of groundedness also informs the restricted action set computation. If all neighbors $\mathcal{N}_i = \{v_j \in V : e_{ij} \in E\}$ (see Def. 3) of agent i can compute an alternate path to ground (other than through agent i) then agent i is allowed to move. To formalize this idea, let $G_{-i} = (V_{-i}, E_{-i})$ with $V_{-i} = V \cup V_{GP} \setminus \{v_i\}$ and $E_{-i} = V_{-i} \times V_{-i}$ such that $e_{ij} \in E_{-i}$ if for $v_i, v_j \in V_{-i}$ we have $\|a_i - a_j\|_{L_1} = 1$. G_{-i} is therefore the connectivity graph of the current configuration including the ground plane without agent i. $R_i^{3D}(a)$ is then defined as follows.

Definition 9: The three-dimensional restricted action set $R_i^{3D}(a) = \mathcal{M}_s \cup \mathcal{M}_c$ if all agents $v_j \in \mathcal{N}_i$ are grounded on G_{-i}. Otherwise, $R_i^{3D}(a) = \{a_i\}$.

This definition encodes the same criteria as the two-dimensional action set with the additional constraint of maintaining groundedness (see Fig. 2). If agent i executing an action would leave any of its neighbors ungrounded, agent i is not allowed to move.

IV. DETERMINISTIC COMPLETENESS

In this section we establish completeness of deterministic reconfiguration in two and three dimensions. We will show that for any two configurations $\mathcal{C}_I$ and $\mathcal{C}_T$ there exists a deterministically determined sequence of individual agent actions such that configuration $\mathcal{C}_I$ will be reconfigured into $\mathcal{C}_T$. These results are required to show irreducibility of the Markov chain induced by the learning algorithm outlined in Section V. Irreducibility guarantees the existence of a unique stationary distribution and furthermore a unique potential function maximizer. We first show completeness of 2D reconfiguration.

Theorem 1 (Completeness of 2D reconfiguration):
Any given two-dimensional configuration $\mathcal{C}_I$ can be reconfigured into any other two-dimensional configuration $\mathcal{C}_T$, i.e. there exists a finite sequence of configurations $\{\mathcal{C}_I = \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_M = \mathcal{C}_T\}$ such that two consecutive
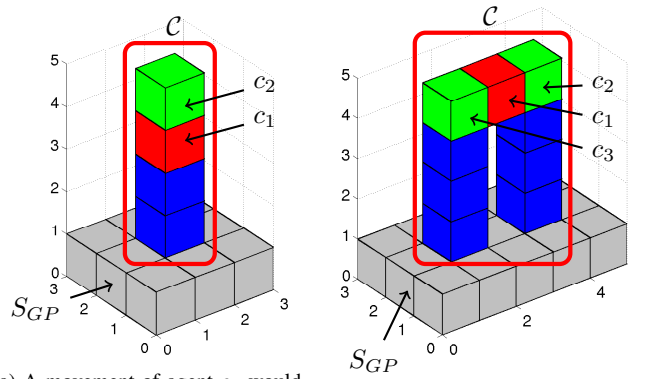
(a) A movement of agent c_1 would remove groundedness of agent c_2.

(b) Agent c_1 can move without breaking the groundedness constraint for agents c_2 and c_3.

Fig. 2: Examples of grounded configurations and feasible motions of cubes.

configurations differ only in one individual agent motion (a proof is shown in [17]).

The result in Theorem 1 holds for any configuration, even configurations that consist of multiple connected components. Before we can show a similar result for the 3D case we need to introduce a graph theoretic result.

Lemma 1: According to Lemma 6 in [18], any finite graph with at least two vertices contains at least two vertices which are not articulation points.[2]

Theorem 2 (Completeness of 3D to 2D reconfiguration): Any finite grounded 3D configuration $\mathcal{C}^{G,3D}$ can be reconfigured into a 2D configuration $\mathcal{C}_{Int}^{2D}$, i.e. there exists a finite sequence of configurations $\{\mathcal{C}^{G,3D} = \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_M = \mathcal{C}_{Int}^{2D}\}$ such that two consecutive configurations differ only in one individual agent motion (a proof is shown in [17]).

Corollary 1: Any finite grounded 3D configuration $\mathcal{C}_I^{G,3D}$ can be reconfigured into any other finite grounded 3D configuration $\mathcal{C}_T^{G,3D}$ (a proof is shown in [17]).

V. STOCHASTIC RECONFIGURATION

In this and the following section we present a stochastic reconfiguration algorithm that is fully distributed, does not require any precomputation of paths or actions, and can adapt to changing environment conditions. Unlike log-linear learning ([2]), which cannot handle restricted action sets, and variants such as binary log-linear learning ([1], [11], [12]), which can only handle action sets constrained by an agent's own previous action, the presented algorithm guarantees convergence to the potential function maximizer even if action sets are constrained by all agents' actions.

Our algorithm is based on the Metropolis-Hastings algorithm ([13],[8]), which allows the design of transition probabilities such that the stationary distribution of the underlying Markov chain is a desired target distribution, which we choose to be the Gibbs distribution. This choice enables a

---

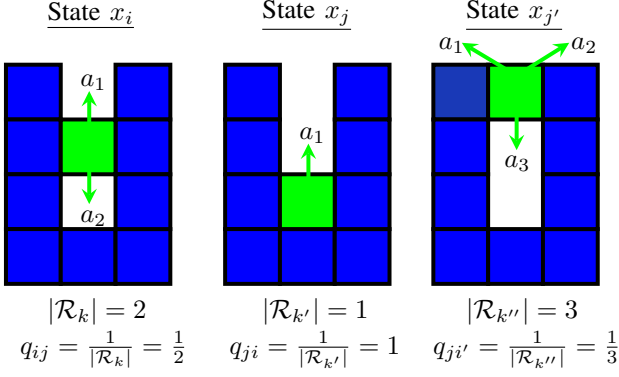[2]An articulation point is a vertex in a graph whose removal would disconnect the graph.

State x_i	State x_j	State $x_{j'}$

Fig. 3: Example of forward and reverse actions with their associated proposal probabilities q_{ij}, q_{ji}, and q'_{ji}. Note that x_i, x_j, x'_j are states of the entire configuration, and agent k is the currently active agent.

distributed implementation of the learning rule in Theorem 3 through the potential game formalism (see Corollary 2). The Metropolis-Hastings algorithm guarantees two results: existence and uniqueness of a stationary distribution. We will use these properties to show that the only stochastically stable state is x*, the potential function maximizer.

Theorem 3: Given any two states x_i and x_j representing global configurations, the transition probabilities

$$p_{ij} = \begin{cases} q_{ji} e^{\frac{1}{\tau}(\Phi(x_j) - \Phi(x_i))} & \text{if } e^{\frac{1}{\tau}(\Phi(x_j) - \Phi(x_i))} \frac{q_{ji}}{q_{ij}} \leq 1 \\ q_{ij} & \text{o.w.} \end{cases}$$

guarantee that the unique stationary distribution of the underlying Markov chain is a Gibbs distribution of the form $Pr[X = x] = \frac{e^{\frac{1}{\tau}\Phi(x)}}{\sum_{x' \in \mathcal{X}} e^{\frac{1}{\tau}\Phi(x')}}$ (a proof is shown in [17]).

Note that a transition from configuration x_i to x_j is accomplished by agent k executing action $a' \in \mathcal{R}_k$ starting at its current location a. Therefore, we can interpret q_{ij} as a transition probability for a forward action and q_{ji} as a reverse action (see Fig. 3). Theorem 3 applies equally for 2D and 3D configuration. However, for 3D reconfiguration, the proof relies implicitly on the notion of groundedness through the computation of R_i^{3D}. The following theorem requires the definition of stochastic stability.

Definition 10 (Stochastic Stability [19]): A state $x_i \in \mathcal{X}$ is *stochastically stable* relative to a Markov process P^ϵ if the following holds for the stationary distribution π $\lim_{\epsilon \to 0} \pi_{x_i}^\epsilon > 0$.
Note that the Markov process is defined through the transition probabilities in Theorem 3 and the stationary distribution is a Gibbs distribution. Furthermore ϵ is equivalent to the learning rate τ.

Theorem 4: Consider the self-reconfiguration problem in Def. 2. If all players adhere to the learning rule in Theorem 3 then the unique stochastically stable state x^* is the state that maximizes the global potential function (a proof is shown in [17]).

Note that the maximum global potential is achieved when all agents are at a target position $a_i \in \mathcal{C}_T$. Algorithm 1 shows an implementation of Theorem 3.

Algorithm 1 Centralized game-theoretic learning algorithm. Note that state x_j is the result of agent k applying action $a_{i \to j, k}$ and x_i and x_j refer to states of the entire configuration. Corollary 2 allows a decentralized version of this algorithm.

Require: Current and target configuration $\mathcal{C}$ and $\mathcal{C}_T$
  while True **do**
    Randomly pick an agent k in state x_i
    Compute restricted action set $\mathcal{R}_k$
    Select $a_{i \to j, k} \in \mathcal{R}_k$ with probability $q_{ij} = \frac{1}{|\mathcal{R}_k|}$
    Compute $\alpha_{ij} = \min\left\{1, \frac{q_{ji}}{q_{ij}} e^{\frac{1}{\tau}(\Phi(x_j) - \Phi(x_i))}\right\}$
    if $\alpha_{ij} = 1$ **then**
      $x_{t+1} = x_j$
    else
      $x_{t+1} = \begin{cases} x_j & \text{with probability } \alpha_{ij} \\ x_i & \text{with probability } 1 - \alpha_{ij} \end{cases}$
    end if
  end while

VI. A DECENTRALIZED ALGORITHM

One shortcoming of Algorithm 1 is its centralized nature that requires the computation of a global potential function $\Phi(x_i)$ and depends on the entire current configuration x_i. The formulation of the self-reconfiguration problem as a potential game allows us to rewrite the transition probabilities in a decentralized fashion as follows.

Corollary 2: The global learning rule of Theorem 3 can be decentralized through locally computable transition probabilities p_{ij}, where

$$p_{ij} = \begin{cases} q_{ji} e^{\frac{1}{\tau}(U_k(a'_k) - U_k(a_k))} & \text{if } e^{\frac{1}{\tau}(U_k(a'_k) - U(a_k))} \frac{q_{ji}}{q_{ij}} \leq 1 \\ q_{ij} & \text{o.w.} \end{cases}$$

These transition probabilities can be computed and executed with local information only (a proof is shown in [17]). Note that local in this context can mean multiple hops, because the computation of restricted action sets requires to maintain groundedness of all neighboring agents.

VII. IMPLEMENTATION

Algorithm 1 was implemented and evaluated in Matlab with a learning rate $\tau = 0.001$ that struck a balance between greedy maximization of agent utilities and exploration of the state space through suboptimal actions. Agents' restricted action sets depended on the agents' joint action and the environment - in our simulations only the ground plane (agents were initialized on or above the ground plane, i.e. their z-coordinate $z \geq 1$). In a straightforward extension to this algorithm, obstacles can be added to restrict the environment even further.

Fig. 4 shows convergence results of Algorithm 1 of configurations containing 10, 20, and 30 agents. Four types
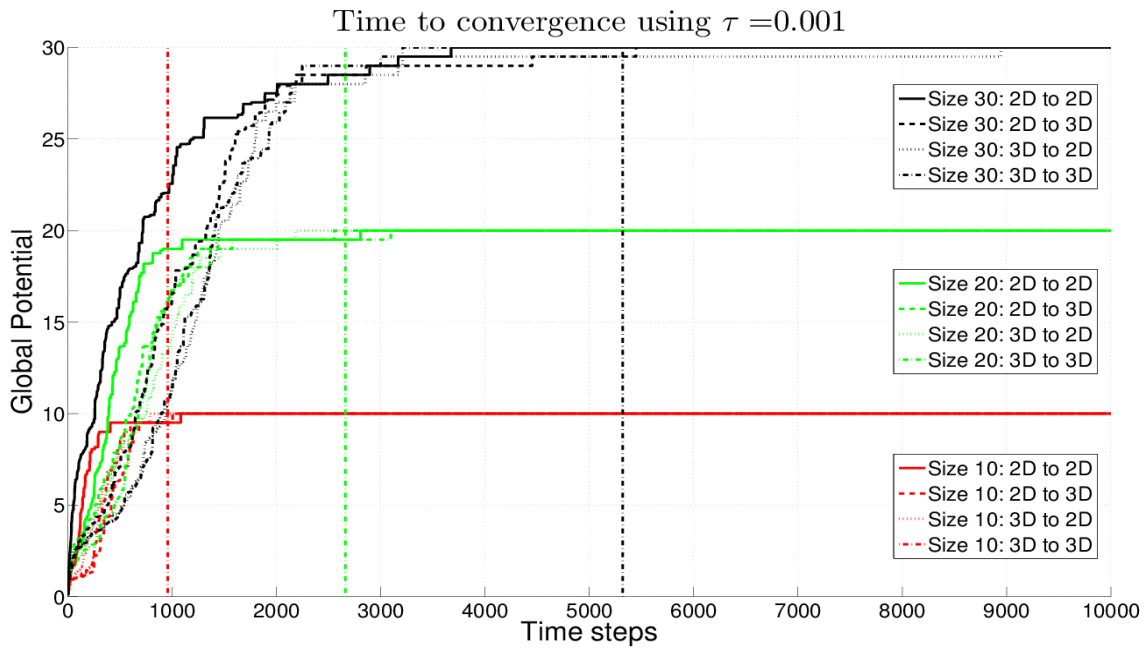
Fig. 4: Convergence times for different types of configurations and sizes ranging from 10 to 30 agents.

of reconfigurations have been performed: 2D to 2D, 2D to 3D, 3D to 2D, and 3D to 3D. Ten trials were conducted for each scenario and convergence was achieved once the configuration reached a global potential of $\Phi = N$, i.e. every agent has a utility of $U_i = 1$. The vertical lines in Fig. 4 represent the average time to convergence of all four types of reconfigurations of a certain size (e.g. the leftmost line represents average convergence of a configuration of 10 agents). Note that for the scenarios of Fig. 4, the target configuration was offset from the initial configuration by a translation of 10 units along the x-axis. One can observe that at the beginning of each reconfiguration the global potential ramps up very fast (within a few hundred time steps), but the asymptotic convergence to the global optimum can be slow (see the case 3D to 2D for 30 agents).

REFERENCES

[1] G. Arslan, J.R. Marden, and J.S. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, September 2007.

[2] Lawrence E. Blume. The statistical mechanics of strategic interaction. *Games and economic behavior*, 5(3):387–424, 1993.

[3] Zack Butler, Keith Kotay, Daniela Rus, and Kohji Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *The International Journal of Robotics Research*, 23(9):919–937, 2004.

[4] Kenneth C. Cheung, Erik D. Demaine, Jonathan Bachrach, and Saul Griffith. Programmable assembly with universally foldable strings (moteins). *IEEE Transactions on Robotics*, 27(4):718–729, 2011.

[5] Robert Fitch and Zack Butler. Million module march: Scalable locomotion for large self-reconfiguring robots. *The International Journal of Robotics Research*, 27(3-4):331–343, 2008.

[6] Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2460 – 2467, October 2003.

[7] Michael J. Fox. *Distributed Learning in Large Populations*. PhD thesis, Georgia Institute of Technology, August 2012.

[8] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[9] Keith Kotay and Daniela Rus. Efficient locomotion for a self-reconfiguring robot. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2963–2969, 2005.

[10] Haruhisa Kurokawa, Kohji Tomita, Akiya Kamimura, Shigeru Kokaji, Takashi Hasuo, and Satoshi Murata. Distributed self-reconfiguration of M-TRAN III modular robotic system. *The International Journal of Robotics Research*, 27(3-4):373–386, 2008.

[11] Yusun Lee Lim. Potential game based cooperative control in dynamic environments. Master's thesis, Georgia Institute of Technology, 2011.

[12] Jason R. Marden and Jeff S. Shamma. Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation. *Games and Economic Behavior*, 75(2):788–808, 2012.

[13] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[14] Dov Monderer and Lloyd S Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[15] Daniel Pickem and Magnus Egerstedt. Self-reconfiguration using graph grammars for modular robotics. In *Analysis and Design of Hybrid Systems (ADHS), 4th IFAC Conference on*, Eindhoven, Netherlands, June 2012.

[16] Daniel Pickem, Magnus Egerstedt, and Jeff S Shamma. Complete heterogeneous self-reconfiguration: Deadlock avoidance using hole-free assemblies. In *Distributed Estimation and Control in Networked Systems (NecSys'13), 4th IFAC Workshop on*, volume 4, pages 404–410, September 2013.

[17] Daniel Pickem, Magnus Egerstedt, and Jeff S. Shamma. A game-theoretic formulation of the homogeneous self-reconfiguration problem. *ArXiv e-prints*, 2015. Available: http://arxiv.org/abs/1509.00737.

[18] Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, Jan. 2001.

[19] Peyton H. Young. The evolution of conventions. *Econometrica*, January 1993.

[20] M. Zhu and S. Martínez. Distributed coverage games for energy-aware mobile sensor networks. *SIAM Journal on Control and Optimization*, 51(1):1–27, 2013.