A Self-Adaptive Framework for Modular Robots in Dynamic Environment: Theory and Applications

Chih-Han Yu and Radhika Nagpal

The International Journal of Robotics Research published online 13 October 2010
DOI: 10.1177/0278364910384753

The International Journal of
Robotics Research
00(000) 1–22

DOI: 10.1177/0278364910384753
ijr.sagepub.com

A Self-adaptive Framework for Modular Robots in a Dynamic Environment: Theory and Applications

Chih-Han Yu and Radhika Nagpal

Abstract

Biological systems achieve amazing adaptive behavior with local agents that perform simple sensing and actions. This has recently inspired the control strategies and design principles of modular robots. In this paper, we introduce a distributed control framework through which modular robots can achieve various self-adaptive tasks. By self-adaptive tasks, we imply tasks where the modular robot uses its distributed sensors to solve tasks and cope with environment changes. We show that modular robot self-adaptive tasks can be formulated as distributed constraint maintenance on a networked multi-agent system such that performing collective self-adaption can be simplified as satisfying local constraints. This formulation allows us to propose a control framework based on a class of multi-agent algorithms called distributed consensus. We further generalize this framework to capture a wide range of sensor–actuator networks in different distributed robotic systems. We prove various theoretical properties of the framework, including its scalability to network diameter and the number of modules. Based on our theoretical understanding, we demonstrate this framework with various tasks, including (1) self-adaptive structures that maintain their shapes in changing environments, (2) an adaptive column that can adapt to external force, and (3) a modular gripper that can manipulate fragile objects. This work provides a deep understanding of the theoretical properties of distributed consensus-type control and its applications to modular robots.

Keywords

Distributed control, modular robot, distributed robot system, networked robot, bio-inspired multi-agent systems

1. Introduction

In nature, biological systems achieve sophisticated and scalable group behavior with vast numbers of independent agents using simple control strategies, e.g. bird flocks (Reynolds 1987), fish schools, and multicellular organisms. Such biological phenomena have inspired the design of several distributed multi-agent systems, for example, swarm robotic systems (Groß et al. 2006), sensor networks (Akyildiz et al. 2002), and modular robots (Rus et al. 2002). Among these systems, the design of modular robots has been widely influenced by multicellular behaviors in biological systems (Pamecha et al. 1996; Rus et al. 2002; Zykov et al. 2005; Shimizu et al. 2005; Goldstein et al. 2005; Bishop et al. 2005; Lyder et al. 2008; Yu et al. 2008). Each module, like a cell, is an independent and autonomous component, and all modules need to coordinate to achieve the assigned task. From a hardware perspective, the incorporation of modularity into robot design allows a single modular robot to become a variety of robotic systems by changing the modules' connectivity. From a control perspective, treating each module as an autonomous individual gives robots the potential to be *robust* to individual module failure and *adaptive* to sudden environment change, similar to how multicellular systems achieve these two characteristics.

Most control strategies for modular robots have only focused on how modules can collectively achieve static goals, such as forming a predetermined shape or performing a precalculated locomotion pattern (Støy et al. 2002; Shen et al. 2006). If a modular robot can utilize the distributed sensing capabilities in its modules, it can potentially adapt its strategy to cope with environmental change more effectively. A small number of groups have addressed how to design adaptive control strategies for modular robot tasks, for example performing conforming locomotion on a snake robot (Kamimura et al. 2004; Yim et al. 2004). However, in these systems, the control laws are individually

School of Engineering and Applied Sciences, Wyss Institute for Biologically-Inspired Engineering, Harvard University, Cambridge, MA, USA

Corresponding author:
Chih-Han Yu
School of Engineering and Applied Sciences, Wyss Institute for Biologically-Inspired Engineering, Harvard University, 33 Oxford Street, Cambridge, MA 02138, USA
Email: chihanyu@gmail.com

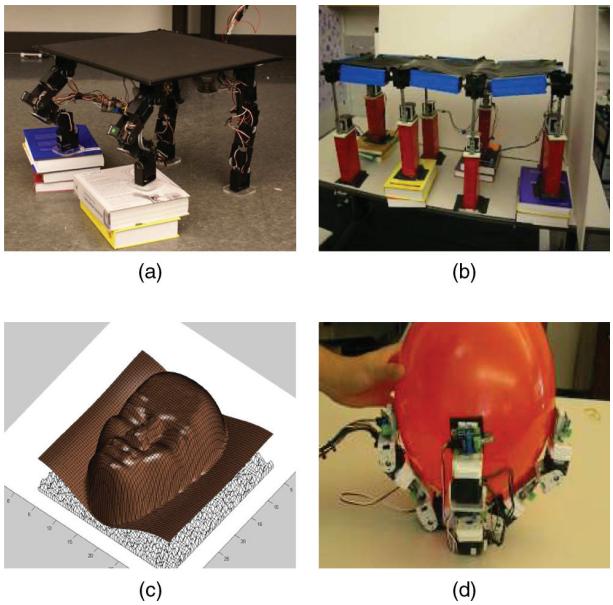

Fig. 1. Different self-adaptation tasks that are achievable with this control framework. (a) Self-balancing table. The module-formed legs are capable of maintaining a level table surface irrespective of tilt changes. (b) Terrain-adaptive bridge. The bridge is able to achieve a flat bridge surface when it is placed on a rough terrain. (c) A 3D Relief display that is capable of rendering complicated shapes. (d) A modular gripper. The modules can cooperatively form a configuration that grasps a fragile object, e.g. a balloon.

designed for a specific task and specific robot. These control strategies do not easily generalize to other systems. Furthermore, the theoretical properties of these control strategies are also rarely addressed, even their correctness. To design control algorithms for reconfigurable multi-agent systems such as modular robots, it is more effective to have a generalizable control framework that can be implemented in different systems and configurations. Furthermore, it is important to theoretically prove whether the designed strategies will correctly lead the robot to achieve the desired tasks.

Biological groups can provide common principles for individual agents to adaptively achieve collective goals. One common strategy by which biological groups achieve a collective goal is by each agent's self-adaptively controlling its state based on feedback from local neighbors (Potts 1984; Okubo 1986; Reynolds 1987). For example, each bird in a migrating flock decides its traveling direction based on its local neighbors' traveling directions, and the whole group can travel cohesively for thousands of miles. Although local interaction between agents is simple, the global behavior it can achieve is amazingly scalable and robust. Recently, there has been some success on analyzing flocking rules and applying them to allow robot swarms to perform flocking (Jadbabaie et al. 2003; Olfati-Saber 2006; Turgut et al. 2008). However, how to generalize such principles to robotic applications beyond flocking-type tasks is rarely addressed in the literatures.

In this paper, we propose a *self-organizing* framework and address the following open questions to translate such biological concepts into a control approach that can be widely applicable in modular and distributed robotics:

1. Can we design a control framework that can capture self-adaptive tasks and goals? Further, can we derive unified principles for applying it to different robots and tasks?
2. Can we provide a theoretical guarantee that the modular robot will achieve the desired global goal? Also, what are the factors affecting the speed of completion?
3. How can we express a wide range of modular robot tasks and goals with this framework?

We first show that a diverse set of modular robot self-adaptive tasks can be expressed as *distributed constraint-maintenance* on a multi-agent system. The relationship between neighboring modules can be expressed as sensory constraints in self-adaptive tasks. When each module satisfies its local constraints, the whole system achieves the desired task. Exploiting locality in this task formulation, we propose a *distributed homeostasis* algorithm that allows agents to achieve the desired tasks. We show that this algorithm relates strongly to a class of algorithms in control theory called distributed consensus (DC), which has been successfully used to model biological group behaviors such as flocking (Reynolds 1987) and firefly synchronization (Lucarelli and Wang 2004; Degesys et al. 2007). We further extend its scope by proposing a *generalized distributed consensus* (GDC) framework and deriving unified principles for designing controllers in distributed robot applications. This significantly enlarges the application area of traditional DC and allows various systems that can be viewed as sensor–actuator networks to be captured by this framework. We prove the correctness of the algorithm by showing that the desired global behavior will result from the agents' local interactions. Building upon this analysis, we characterize how various factors, for example network diameter, affect the system's convergence speed. Our theoretical study also increases our understanding of how to identify the type of tasks for which this kind of algorithm is well suited. The theoretical and algorithmic contributions of this work lead to a unified framework that provably allows modular robots to achieve various self-adaptive tasks with a common control strategy.

In addition to the theoretical work, we also present empirical results from implementing this framework on a diverse set of modular robot applications. In a chain-style modular robot system, we show that the modules can collectively form adaptive structures such as a self-balancing table and a terrain-adaptive bridge (Figure 1(a) and (b)). Our results show that the speed of the algorithm is fast enough to allow fast adaptation to constant perturbations. We further demonstrate that this framework can be extended to modular robots with indirect mapping between sensors and

actuators: a modular gripper that can dynamically adjust its grasping posture and maintain uniform pressure on the target object (Figure 1(d)); and a modular column that can adjust to external pressure to provide firm support. Finally, we provide a discussion about how various modular robot tasks can be viewed as self-adaptation, e.g. for a modular prosthetic structure that supports different locomotion scenarios and a 3D relief display (Figure 1(c)), and how this framework can be applied to many distributed autonomous systems beyond modular robots.

The rest of the paper is organized as follows: we start with a literature review in Section 2. We define our robot model in Section 3. We describe the distributed homeostasis algorithm and its generalization in Section 4. We present various theoretical properties of this framework in Sections 5 and 6. We then present four different self-adaptation tasks that are achievable within this framework and their experimental evaluations in Section 7. Furthermore, we provide discussions on potential extensions and limitations of this framework in Section 8. Finally, we draw conclusions in Section 9.

2. Related Work

A modular robot is a class of robots that are composed of many independent modules. Each module can communicate locally with other modules that are physically connected to it. With appropriate control, modular robots are capable of changing their configurations to become different structures or shapes (Pamecha et al. 1996; Vona and Rus 2001; Rus et al. 2002; Murata et al. 2002; Yim et al. 2004; Shen et al. 2006; Yu et al. 2008). The design of modular robots has been greatly influenced by multicellular systems in biology. Each module is like a cell in a multicellular organism that can potentially adapt to different environments through changing either its internal structure or its external connectivity with other "cells". The ultimate goal for modular robots is to allow such a system to achieve the task adaptively according to its external environment by either controlling actuator parameters or changing the overall connectivity of the modules.

Most groups have focused on static predefined goals that do not depend on sensors for shape transformation and locomotion (Rus et al. 2002; Murata et al. 2002; Zykov et al. 2005). This usually makes the robots difficult to operate in a dynamic environment. Only a few groups have addressed self-adaptive tasks using centralized or decentralized control (Rus et al. 2002; Murata et al. 2002; Yim et al. 2004; Shimizu et al. 2005; Shen et al. 2006). In chain-based robots, Yim et al. (2004) have demonstrated robot locomotion that conforms to the environment via a hand-designed gait table and distributed force feedback. They showed that such an approach allows the robot to negotiate through different obstacle environments. Another type of adaptive locomotion strategy for chain-based robots is based on Central Pattern Generator (CPG). Kamimura et al. and Moeckel et al. have demonstrated such an approach in the M-TRAN (Kamimura et al. 2004) and YaMoR (Moeckel et al. 2005) modular robots, respectively. However, most of the strategies in this category require a tedious design process. In addition, there is no theoretical guarantee whether the control laws will correctly lead all modules to achieve the desired task. The hormone-inspired control framework of Shen et al. (2004) and the resilient robot of Bongard et al. (2007) also address how one can design a controller that adapts to different modular robot topologies. However, their work does not address strategies that allow modular robots to adapt to different environments.

Rus et al. have demonstrated distributed algorithms for locomotion over obstacles (Rus et al. 2002). Bojinov et al. have presented control algorithms for several interesting examples that were tested in simulations: for instance, a hand that grasps an object and a table that adaptively supports a weight (Bojinov et al. 2000). Although Bojinov et al. provides a framework for describing adaptive tasks, it is difficult to generalize to other systems and theoretically prove its correctness. Furthermore, lattice-based systems have one major disadvantage in speed: shape change can only be achieved through module movement, which is slow in the hardware implementation. In this work, we mainly consider a chain-based system that can achieve fast adaptation.

Our proposed control strategy is closely related to DC algorithms in multi-agent systems (Bertsekas and Tsitsiklis 1989; Olfati-Saber et al. 2007). DC is considered to be an underlying principle in many biological group phenomena, from birds flocking to firefly synchronization. It has been widely applied in distributed (robotic) systems, including autonomous vehicle formation control (Jadbabaie et al. 2003; Fax and Murray 2004; Olfati-Saber 2006), sensor network time synchronization (Lucarelli and Wang 2004; Degesys et al. 2007), and the sensor coverage problem (Schwager et al. 2008).

Our work differs from traditional DC algorithms in the following three ways. (1) In the algorithmic aspect, we propose a generalized distributed consensus framework to break the constraint that agents' sensor space and control space must be the same, as in most of the current DC applications[1]. We further supply three sufficient conditions that allow one to design control laws for different sensor–actuator networked agent systems. This allows our framework to be applied to a variety of distributed robotic systems and tasks (Section 4.2). (2) In the application aspect, we extend such an approach to the area of modular robotics and show that it is powerful in solving self-adaptive tasks that require constant adaptation to environmental uncertainties. (3) In the theoretical aspect, we formulate a framework that allows one to view many sensor–actuator tasks as GDC tasks. In addition to proving GDC algorithms, this work also contributes to analyzing how various factors, e.g. agent topology and task complexities, influence GDC's convergence speed, such that it can predict the performance of

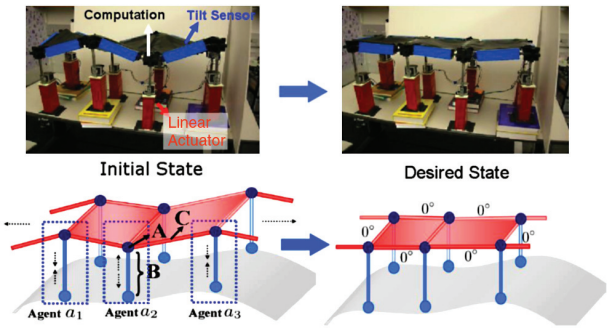

Fig. 2. Top: Terrain-adaptive bridge. Each pillar is an agent that is equipped with computational unit, linear actuator, and communication channel. The robot is initially placed on rough terrain (left) and is programmed to achieve a flat surface (right). Bottom: A skeleton view of the terrain-adaptive bridge. Each agent has computation (A), actuation (B), and communication (C) units.

the algorithm for a given modular robot connectivity and programmed task.

3. Modular Robot Model

In this section, we describe the robot model and the capabilities assumed in our framework. Our primary focus is on modular robotic systems in which the whole robot is composed of many independent and autonomous modules. Our system contains the following components.

3.1. Agent

We define an agent as an independent unit with the following capabilities:

- *Sensor.* Each agent is equipped with one or more sensors suited to different robotics applications. Sensors are used to measure the current state of the agent or relationship between a neighboring pair of agents. In some applications, sensors can also be shared between two neighboring agents to allow them to form inter-agent state relationships. For example, in the terrain-adaptive bridge in Figure 2, there is a tilt sensor mounted in between two pillar agents which allows them to compute inter-agent altitude differences.
- *Actuator.* Each agent is equipped with an actuator. We consider both linear and rotary actuators in our framework.
- *Computation/Communication.* Each agent is capable of performing simple computations such as addition and multiplication. It is also able to communicate with its immediate neighbors that are physically connected to it.

For example, in the terrain-adaptive modular bridge example (Figure 2), each module (pillar) has the above-described capabilities (tilt sensor, linear actuator, computational unit, and communication channel) and is therefore an agent. We therefore refer to a module as an *agent* in the rest of this article. Most of the current modular robots have these stated capabilities, e.g. M-TRAN (Murata et al. 2002), Odin (Lyder et al. 2008), and Superbot (Shen et al. 2006).

3.2. Networked Multi-agent System

In our model, agents have only a one-hop local view. They achieve global tasks by communicating and cooperating with their immediate neighbors. We can therefore view the model as a *networked* multi-agent system. This can be more succinctly represented as a coordination graph G in which vertices represent agents and edges represent communication between an agent and its neighbors. For example, in Figure 3 (left), edges correspond to the communication links between two neighboring agents in the bridge of Figure 2. The neighbor relationship between agents is symmetric, so the edges in graph G are undirected (Figure 3).

3.3. Self-adaptive Tasks as Distributed Constraint Maintenance

In this framework, the multi-agent tasks are described as *a set of inter-agent state or sensor constraints (differences)*, e.g. as the desired sensor difference between neighboring agents, and each agent iteratively tries to satisfy its local constraints. A desired task is achieved if all agents satisfy constraints with all neighbors. One merit of this approach is that once agents have deviated from satisfying the goal state, e.g. due to environment perturbations, each agent autonomously restarts and drives the whole system back to the desired state. This formulation simplifies programming strategies required for the modular robot to autonomously adapt to different conditions by viewing adaptive tasks as maintaining constraints.

Many modular robot tasks can be formulated under this task specification. For example, in the terrain-adaptive bridge example, one can simply specify that each agent needs to maintain zero tilt angle along the edges with all of its neighbors, as shown in the bottom diagram of Figure 2. Other robotic systems with no direct sensor–actuator mapping can also utilize such task specifications; for example, in the case of a modular gripper that maintains equal pressure on distributed sensors, the task can be described by simply programming each agent to maintain a pressure equal to that of its neighbors (Figure 1(d)); further details are given in Section 7.

Such a task description scheme does not require each agent to maintain the same state but can be extended to more complex specifications. For example, one can specify that the modular gripper grasp the object with a different desired pressure distribution (such as applying more pressure on the finger tips). We can also specify a modular surface robot to form more a complex shape by specifying the desired inter-agent constraints. Even in cases where agents are not equipped with actuation capabilities, we

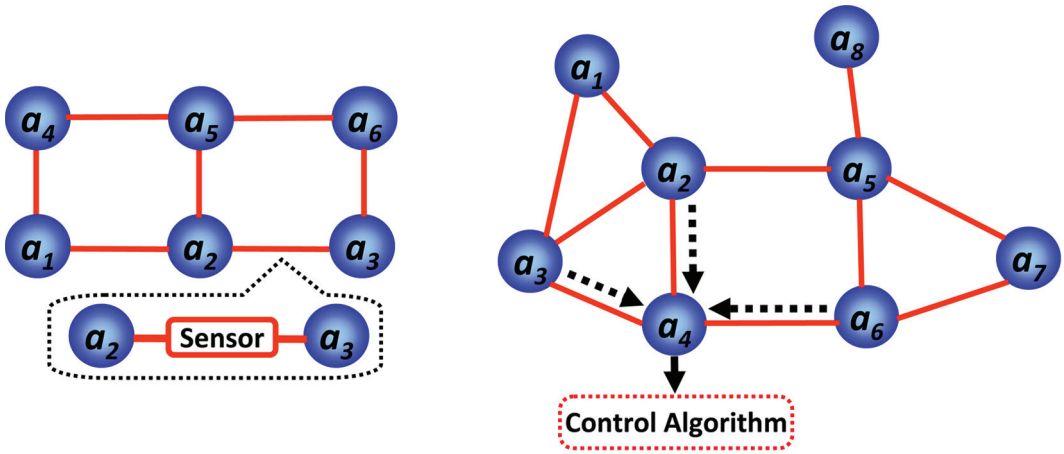

Fig. 3. Left: The modular robot can be viewed as a graph where nodes represent agents and edges represent inter-agent links. There is a sensor on each edge that provides inter-agent tilt angle. Right: An agent cooperation graph can be arbitrary connectivity. Each agent's control algorithm takes all neighboring sensor readings as inputs to compute the agent's new state.

can specify the task in terms of desired state relationships between agents. For example, in a sensor network time-synchronization task, we can simply program each agent to maintain the same timer as its neighbors (agents that are within its communication radius).

4. Control Algorithm

In this section, we begin by presenting a distributed homeostasis algorithm that is based on distributed consensus. We then present a more general form of the algorithm and sufficient conditions for agents to reach consensus. This generalized framework allows us to extend the control law to a wide range of applications which we cover in Section 7.

4.1. Distributed Homeostasis Algorithm

The algorithm is formulated as a process by which a group of networked agents come to a state of satisfying constraints by communicating only with *neighbors*. At each time step, each agent updates its new state according to the difference between its own state and its neighbors' states. This process can be formally written as follows.

Algorithm 1 (Distributed Homeostasis). We have

$$x_i(t+1) = x_i(t) + \alpha \sum_{a_j \in N_i} \left(x_j(t) - x_i(t) - \Delta^*_{ij} \right), \qquad (1)$$

where a_i indicates agent i, and $x_i(t)$ and $x_i(t+1)$ are actuation states[2] of agent i at time step t and $t+1$, respectively. Here N_i indicates the set of all one-hop neighbors of a_i. The small constant α is sometimes called the damping factor. Its value needs to fall within this range: $0 < \alpha < 1/|N_i|$. The inter-agent constraint variable Δ^*_{ij} is the desired state difference between agents a_i and a_j. As we described previously in Section 3, the inter-agent constraints, i.e. Δ^*_{ij} and for all $i,j \in N_i$, are specified when we assign the task to the robot.

If $\Delta^*_{ij} = 0$, for all $i,j \in N_i$, agents are programmed to achieve the same state and this process is DC. On the other hand, with $\Delta^*_{ij} \neq 0$, this is sometimes also called *biased* consensus.

There are three main assumptions buried in Equation (1). First, each agent is capable of directly observing its state and its neighbors' states. Second, each agent is capable of freely driving itself to a new state $x_i(t+1)$. Third, we assume state variable $x_i(t)$ to be scalar and is continuous.

4.2. GDC Algorithm

In many cases, the relationship between sensor space and agent's actuation state is not precisely known. For example, in the modular gripper example in Figure 1 (d), each module is equipped with a rotary motor and a pressure sensor. We program the gripper to grasp around the object while maintaining each module's pressure on the object to be equal. The mapping between the actuator's actuating angle and agent pressure sensor value cannot be directly computed. Therefore, we propose a more general form of the agent control algorithm by allowing agent constraints to be arbitrary sensory constraints between neighboring agents.

Algorithm 2 (Generalized Distributed Consensus). We have

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} \left(g(\theta_i, \theta_j) - \theta^*_{ij} \right) \qquad (2)$$

where θ_i is agent a_i's sensor reading and θ_j indicates the sensor reading of a_i's neighbor a_j, $g(\theta_i, \theta_j)$ is a sensory feedback function that agent a_i receives from its neighbor a_j, and θ^*_{ij}, for all $i,j \in N_i$ are task-specific inter-agent constraints, similar to Δ^*_{ij} in Equation (1).

We note that Equation (2) assumes that agents perform updates with round synchronization, i.e. each agent updates according sensory feedback at every time step. As we prove

in the following section, there are three important requirements for designing $g(\theta_i, \theta_j)$ to ensure that all agents will correctly complete the desired task

$$g(\theta_i, \theta_j) - \theta_{ij}^* = 0 \Leftrightarrow \theta_j - \theta_i = \theta_{ij}^*, \quad (3)$$

$$\text{sign}(x_j - x_i - \Delta_{ij}^*) = \text{sign}(g(\theta_i, \theta_j) - \theta_{ij}^*), \quad (4)$$

$$g(-\theta_i, -\theta_j) = -g(\theta_i, \theta_j). \quad (5)$$

Intuitively, condition 1 (Equation (3)) means that g only "thinks" the system is solved when it actually is; condition 2 (Equation (4)) means that when not solved, each sensory feedback g at least points the agent *in the correct direction* to satisfy the local constraint θ_{ij}^* with a neighboring agent a_j; and condition 3 (Equation (5)) means that g is *anti-symmetric*.

Equation (2) allows us to formulate many different tasks in sensor–actuator networks as a consensus-reaching process between agents. In the previous modular gripper example, $x_i(t)$ is used to represent the rotational angle of the agent's motor and θ_i is used to represent the agent's pressure sensor reading. We can immediately see that there is no direct mapping between an agent's rotational angle and its pressure reading. However, the desired task that agents are programmed to achieve equal pressure, i.e. $\theta_{ij}^* = 0$, for all $i, j \in N_i$, can be viewed as reaching a consensus state: $\theta_i = \bar{\theta}$, for all i.

To utilize the GDC algorithm to solve different tasks requires us to address the following challenge: we need to appropriately design the function g such that the overall dynamics of the agents is equivalent to the distributed consensus dynamics. In fact, Equations (3)–(5) are three simple principles that can guide us to efficiently design g. We discuss the theoretical properties of the GDC algorithm in Sections 5.2 and 6 and describe how to apply the principles of Equations (3)–(5) in designing control laws for different applications in Section 7.

5. Correctness of the Algorithm

In the control law we have presented, an agent sums the feedback from its local neighbors and acts in some fashion based on that feedback. Since the system is decentralized with many agents acting on local information in parallel, a key question is whether these actions always will produce the correct desired global goal (convergence from all initial states). Here we show the following: (1) that the dynamics of all agents can be modeled as a single collective dynamical system; (2) based on this formulation, that the correctness of the algorithms can be characterized for arbitrary connected graphs G and desired goals in both standard DC and generalized DC algorithms.

5.1. Proofs for Distributed Homeostasis Algorithm

We first demonstrate how to aggregate all agent update equations to become *collective dynamics*. This allows us to study the emergent global behavior of all agents by analyzing a single dynamical system. By leveraging results from spectral graph theory and stochastic matrix theory, we prove the convergence property for this dynamical system.

5.1.1. Collective Dynamics We first consider the case for standard distributed consensus algorithm (Equation (1)). Let $X(t)$ represent the ensemble of all agents' states at time t:

$$X(t) = (x_1(t), x_2(t), \ldots, x_n(t))'.$$

Based on Equation (1), we can write the collective dynamics of all agents as

$$X(t+1) = A \cdot X(t) + \tilde{b}, \quad (6)$$

where $A = [\mathbf{a}_{ij}]$, an $n \times n$ matrix with element $\mathbf{a}_{ij}$ defined by

$$\mathbf{a}_{ij} = \begin{cases} \alpha & \text{if } a_j \in N_i \text{ and } i \neq j \\ 1 - \alpha \cdot |N_i| & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

and where $\tilde{b}_i = \alpha \cdot \sum_{a_j \in N_i} \Delta_{ij}^*$ is a *bias vector*. We note that A is a stochastic matrix since each row sums to 1 (Seneta 1981). In addition, $\sum_i \tilde{b}_i = 0$, since $\Delta_{ij}^* = -\Delta_{ji}^*$ for all i, j.

Equation (6) can be rewritten as

$$X(t+1) - X(t) = -\alpha \cdot LX(t) + \tilde{b},$$

where L is the so-called *graph Laplacian* matrix $L = [l_{ij}]$, with

$$l_{ij} = \begin{cases} -1 & \text{if } a_j \in N_i \text{ and } i \neq j, \\ |N_i| & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

In the case that $\tilde{b} = 0$, this equation has been analyzed thoroughly. Olfati-Saber et al. (2007) have shown how it arises in different types of consensus problem, including the alignment problem in flocking (agreeing on a single heading) and the synchronization problem (agreeing on a single phase).

Their key results relate the eigenvalues of A and L to convergence of the local rules defined by these matrices[3]. The key point for us is that having a non-zero bias vector $\tilde{b}$ does not affect the eigenvalue analysis, so that the results of Olfati-Saber et al. (2007) apply here as well.

Specifically, let λ_i denote the ith *smallest* eigenvalue of the graph Laplacian L, and let μ_i be the ith *largest* eigenvalue of A. Then

$$\mu_i = 1 - \alpha\lambda_i.$$

Now, L has a simple eigenvalue $\lambda_1 = 0$ with associated eigenvector $\mathbf{1}$ (an all-ones vector), and A has as its largest eigenvalue $\mu_1 = 1$. As shown in (Mohar 1991a), when the coordination graph G is *connected*, the *second* smallest eigenvalue of L, λ_2, is strictly larger than zero. Thus, the second largest eigenvalue μ_2 of A is strictly smaller than one.

We define X^* as our desired state, it is then easy to see (as shown in Appendix A) that under the iteration of Equation (6),

$$\|X(t) - X^*\|^2 \leq \mu_2^{2t} \|X(0) - X^*\|^2.$$

Thus, since $\mu_2 < 1$, we have the following result.

Theorem 1 (Convergence, Distributed Homeostasis Algorithm). *Let X^* be the desired shape state that $x_j^* - x_i^* = \Delta_{ij}^*$ for all a_i and $a_j \in N_i$ and the cooperation graph G is connected. The collective dynamics will converge to X^* for all initial conditions with exponential rate μ_2.*

Proof. See Appendix A. □

Because μ_2 does not depend on the bias vector $\tilde{b}$ at all, this convergence analysis is independent of the desired task.

From the convergence proof, we can also see that

$$\|Y(t+1)\| \leq \mu_2 \|Y(t)\|, \tag{7}$$

where $Y(t) = \|X(t) - X^*\|$ represents the current distance from the desired goal.

5.2. Proofs for Generalized Distributed Consensus Algorithm

Now we consider the case of the generalized distributed consensus algorithm in Equation (2). We first let

$$\phi_{ij}(t) = \alpha \frac{g(\theta_i, \theta_j) - \theta_{ij}^*}{x_j(t) - x_i(t) - \Delta_{ij}^*}.$$

Then we can reexpress Equation (2) as

$$x_i(t+1) = x_i(t) + \sum_{a_j \in N_i} \phi_{ij}(t)\,(x_j(t) - x_i(t) - \Delta_{ij}^*). \tag{8}$$

Intuitively, comparing Equation (8) with Equation (1), the agent a_i's step length based on feedback is now a *state-dependent variable* instead of a constant factor α. In other words, this variable varies with the changing state of the agents. Following the same procedure in Section 5.1, we can rewrite the system dynamics as

$$X(t+1) = A(t) \cdot X(t) - \tilde{b}(t), \tag{9}$$

where $A(t) = [\mathbf{a}_{ij}(t)]$ is an $n \times n$ matrix with element $\mathbf{a}_{ij}(t)$ defined by

$$\mathbf{a}_{ij}(t) = \begin{cases} \phi_{ij}(t) & \text{if } a_j \in N_i \text{ and } i \neq j \\ 1 - \sum_{a_j \in N_i} \phi_{ij}(t) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

and $\tilde{b}_i(t) = \sum_{a_j \in N_i} \phi_{ij}(t)\,\Delta_{ij}^*$. We note that since $\phi_{ij}(t)$ is a state-dependent (and, thus, time-varying) variable, so $A(t)$ and $\tilde{b}(t)$ are also state dependent.

The style of convergence proof we used in the original case required the generating matrix A to be row stochastic and symmetric. That is, (i) $A_{ij} \geq 0$ for $i \neq j$; (ii) $A_{ii} \leq 0$ for all i; (iii) $A_{ij} = A_{ji}$ for all i,j; (iv) $\sum_j A_{ij} = 1$ for all i. If g satisfies the three requirements, $A(t)$ is actually row stochastic and symmetric for all t. To see why, note that for *off-diagonal* elements, $A_{i \neq j}(t) = \phi_{i \neq j}(t)$ for all a_i and $a_j \in N_i$. However, because of the condition on g in Equation (4), the numerator in the definition of ϕ_{ij} always has the same sign as the denominator, and so their ratio is non-negative. For the *diagonal* elements, note that $A_{ii}(t) = 1 - \sum_j \phi_{ij}(t) \leq 0$ for all i. This can always be ensured as long as α is chosen small enough. Finally, since $g(-x,-y) = -g(x,y)$ (Equation (5)), we have $\phi_{ij}(t) = \phi_{ji}(t)$ so the symmetry condition is ensured[4].

The guarantee of stochasticity and symmetry allows us to prove the following result.

Theorem 2 (Convergence, GDC Algorithm). *Let X^* be the desired shape state that $x_j^* - x_i^* = \Delta_{ij}^*$ and cooperation graph G be connected. The collective dynamics defined by Equation (9) will ensure that $X(t)$ converges to X^* for all initial conditions with convergence rate of at least*

$$\mu_2^* = \max_t \mu_2(A(t)).$$

Proof. See Appendix B. □

Intuitively, what this result says is that the analog of the original result holds, except that the guarantee becomes somewhat looser, as we now have to maximize $\mu_2(A(t))$ over all t. However, as long as we have a lower bound on λ_2, the second eigenvalue of the cooperation graph, the essential point still holds.

6. Factors Affecting Convergence Speed

In the previous section, we proved the correctness of the algorithms and derived convergence rates. We now address several important questions related to how convergence rates vary with assorted network graph parameters. More specifically, we examine the following factors:

1. **Scalability and impact of multi-agent topology.** How does the time to complete the desired task increase as we increase the number of agents and the diameter of the networked multi-agent system?
2. **Task complexity.** How does the complexity of the task affect the time to achieve the task?
3. **Agent failures.** How does the multi-agent system perform when some agents encounter actuation and/or communication failures?
4. **Reactivity.** How do the agents react to perturbations from the desired state?

We provide precise answers to these questions. In doing so, it is useful first to derive an inequality for the number of iterations required to achieve our desired task within a certain error tolerance. By error tolerance, we mean the following.

Definition 1. The task achieved by the agents at state $X(t)$ is an ϵ-*approximation* of the desired task if $X(t)$ satisfies $Y(t) = \|X(t) - X^*\| \leq \epsilon$.

The error tolerance ϵ represents the fact that agents have finite resolution in controlling their actuation, so some level of inaccuracy must be tolerated. From Theorems 1 and 2, we know that the agents approach the desired state X^* at an exponential rate. We can further express the number of time steps required to achieve the goal as a function of convergence rate μ_2, ϵ, the initial condition $X(0)$, and the goal condition X^*:

$$\begin{aligned} \|X(t^*) - X^*\| &\leq \mu^{t_{\max}} \|X(0) - X^*\| \leq \epsilon \\ \Rightarrow t_{\max} &\leq \log_\mu \left(\frac{\epsilon}{\|X(0) - X^*\|} \right). \end{aligned} \tag{11}$$

We can see from inequality (11) that the number of iterations required, $t_{\max}$, depends on two main factors: the connectivity of the cooperation graph G (as reflected by its corresponding matrix A's second eigenvalue μ_2) and the *distance* $\|X(0) - X^*\|$ from the initial state $X(0)$ to the desired goal X^*. The first factor is dependent only on G and is independent of the desired goal X^*, and vice versa for the second factor.

6.1. Scalability and Topology

Assuming that the initial distance from the desired goal is fixed, the number of iterations depends on μ_2. We showed in Section 5.2 that $\mu_2 = 1 - \alpha\lambda_2$, where λ_2 is the second eigenvalue of the graph Laplacian. This second eigenvalue has a special significance in graph theory and is called the *algebraic connectivity* because it encodes how well the graph is connected. While algebraic connectivity has been studied extensively in graph theory, its use in understanding decentralized algorithms is relatively new. Here, we show that there are several important bounds on λ_2 that will provide us with a concrete understanding of how the algorithm scales as we scale up the size of the multi-agent system.

Theorem 3 (Mohar (1991a)). *Let L be the graph Laplacian of G. Then the second eigenvalue λ_2 of L satisfies*

$$\lambda_2 \geq \frac{4}{|G| \cdot \mathrm{diam}(G)},$$

where $n = |G|$ and $D = \mathrm{diam}(G)$ are the size and diameter of G, respectively. Note that this theorem implicitly provides an upper bound on μ_2, which gives us the *worst case* convergence rate.

$$\mu_2 \leq 1 - \frac{4\alpha}{n \cdot D}. \tag{12}$$

For example, assume that the multi-agent system consists of n agents connected in a line. If we double the length of the line (thus doubling both n and D), λ_2 is reduced by a factor of four. Hence, the runtime of the algorithm is worst-case bounded by $O(n^2)$ for a line. We can see from here that the performance drop is greater than linear with the number of nodes or diameter. However, as we show in Section 6.4, the distance from the final state plays an important role, and thus in many cases the absolute number of iterations can be quite low.

Note that there are many forms of upper and lower bounds on λ_2 that can provide tighter estimates of the convergence time. Here, we use one of the simpler bounds that is easy to reason from, but in Appendix G we provide some more examples of known bounds based on different graph topological properties.

6.2. Task Complexity

Another important question is how the desired task impacts the speed of convergence. As we can see from inequality (11), a bound on the effect of task complexity on convergence is given by the *Euclidean distance* of the final state from the initial state. Given a random condition, one can compute the expected distance to a complex task and thus find an expected running time for achieving that task. This allows us to directly estimate convergence time given assumptions about the initial state and knowledge of the final desired goal. In fact, it is possible to derive a finite bound on the total number of iterations required for *any task* given a specific initial condition.

Theorem 4. *Let a collective dynamical system's convergence rate be μ. Assume $x_i(t) \in R^+$ for all i, t and that initial states of the agents are known. Let $\mathcal{C} = \sum_i x_i(0)$. Then the number of iterations required to achieve an ϵ-approximation of the desired goal is at most*

$$t_{\max} = \left\lceil \log_{\mu_2} \left(\frac{\epsilon}{\sqrt{2}\mathcal{C}} \right) \right\rceil.$$

Proof. See Appendix C. □

We can further replace μ_2 with the worst-case convergence rate obtained from Theorem 3: $\mu_2 = 1 - \alpha/(n \cdot D)$ or $\mu_2 = 1 - \phi_{\min}/(n \cdot D)$. This indicates that if the agents' *connection topology* and the agents' *initial states* are known, we can calculate the number of iterations *guaranteed* to achieve an ϵ-approximation of the desired goal.

6.3. Robustness to Agent Failures

In this section, we examine the correctness of the algorithm while having one or more agents fail in the process. The first type of failure we consider here is actuation failure. In this case, some agents are unable to drive themselves to the desired states by controlling their equipped actuators. They are then stuck in the state they have when the failures initially occur. We show that if there is one agent with an actuation failure, the whole system will still converge and achieve the desired task X^*. We then show that if there are

multiple such cases, the system will still converge, but may converge to a *non-satisfying* state.

Theorem 5 (Single-agent Actuation Failure). *Let agent a_i be the failed agent: a_i fails to cooperate and maintains its state x^*. Let all of the other agents execute either the distributed homeostasis or the GDC algorithm. The agents will eventually achieve the desired state X^*.*

Proof. See Appendix D. □

Theorem 5 intuitively tells us that all other agents will try to accommodate the single failed agent, such that the desired task is still eventually achieved. We now look at the case when there are multiple agent failures.

Theorem 6 (Multiple-agents Actuation Failures). *Let* **F** *be the set of all failed agents and $|\mathbf{F}| > 1$. Let all of the other agents executes either the distributed homeostasis or GDC algorithm. All agents will eventually converge to fixed states $X_{\mathbf{F}}$, but it is not guaranteed that $X_{\mathbf{F}} = X^*$.*

Proof. See Appendix E. □

The above two theorems tell us that when the networked multi-agent system executes our proposed algorithm, we can allow *one* agent to have *complete* actuation failure and still complete the desired task. In practice, there are many ways to prevent an agent's complete actuation failure. One simple way is to equip each agent with redundant actuators, similar to how multicellular systems increase their robustness with redundant cells. For example, we can equip three redundant actuators in each agent (leg) in the self-balancing table example. An agent will only have complete actuation failure when all three actuators fail. For detailed implementation of this approach, please refer to Yu et al. (2007).

Another type of agent failure is *communication failure*. In this case, some agents are not able to communicate with their neighbors, either temporarily or permanently. The agent communication graph is thus time-varying, denoted as $G(t)$. We define the time-varying agent communication graph as *periodically connected* if the *union* of graph $G(t)$ over a finite time period is connected. This leads to the following theorem.

Theorem 7 (Communication Failures). *Let $G(t)$ be the agent communication topology at time t. If some agents' communication links temporarily fail, and if $G(t)$ is periodically connected over time, the agents will eventually achieve the desired task.*

Proof. See Appendix F. □

Theorem 7 intuitively tells us that agents can still collectively achieve the desired task even if the communication links between agents fail temporarily, provided that the graph $G(t)$ satisfies the periodically connected definition.

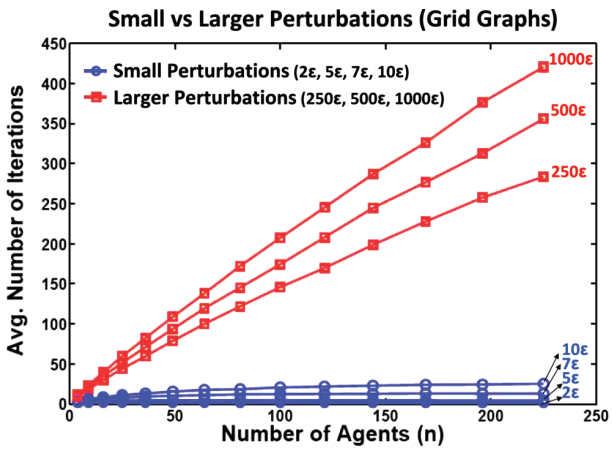

Fig. 4. Number of iterations required to complete the task versus number of agents, large and small perturbations with self-organizing (decentralized) approach. In our experiment, agents are connected to form a grid graph. We increase the number of agents of the grid graph from $1 \times 1, 2 \times 2, \ldots, 15 \times 15$ (x-axis). We apply small perturbations $(2\epsilon, 5\epsilon, 7\epsilon, 10\epsilon)$ and large perturbations $(250\epsilon, 500\epsilon, 1{,}000\epsilon)$ in each graph configuration. We can see that, in small perturbation cases, the number of iteration required stays low even when we increase the number of agents in the graph to several hundreds.

6.4. Reactivity

Another important criterion is how the networked multi-agent system reacts to perturbations. From inequality (11), we can see that if $Y(0)$ is small, then a few iterations will be sufficient to achieve an ϵ-approximation. Furthermore, even as the number of agents increases, the number of iterations remains low.

Figure 4 shows the system's reactivity for square grid topologies $(1 \times 1, 2 \times 2, \ldots, 15 \times 15)$. Red lines indicate larger perturbations $(Y(0) = 250\epsilon, 500\epsilon, 1{,}000\epsilon)$, and blue lines indicate smaller perturbations $(Y(0) = 2\epsilon, 5\epsilon, 7\epsilon,$ and $10\epsilon)$. Each point on Figure 4 represents the mean number of simulation rounds required of 1,000 random initial $X(0)$ that satisfy the given $Y(0)$. We can see from the figure that the networked multi-agent system's reactivities toward small perturbations (blue lines) scale well with the number of agents. If the environment changes smoothly, then even large changes will appear as small perturbations over time. This shows why the algorithm performs particularly well in adaptation tasks such as the self-adaptive structures shown in the next section.

7. Self-adaptive Tasks

Based on our theoretical analysis in the previous section, we can see that our proposed algorithm achieves superior performance in tasks that require the robot to adapt to external perturbations constantly. In this section, we illustrate several example modular robot self-adaptive tasks of this type. We demonstrate: (A) a module-formed table and bridge that

Fig. 5. (a) Concept diagram of self-adaptive structure. Red modules form a substrate of the surface while blue modules form the "legs" of the structure. (b) Self-balancing table hardware prototype. Each leg is an independent agent, and the agent topology is shown on the right. (c) Terrain-adaptive bridge prototype. Each pillar is an independent agent.

can quickly adapt to external perturbation while maintaining the table surface and bridge surface level; (B) A modular pressure-adaptive column that is capable of maintaining "pressure consensus" by changing shape itself to absorb uniform pressure; and (C) a modular gripper that is capable of grasping a fragile object while each module applies identical force on the balloon.

We first present how to apply our proposed algorithm in solving each of the above tasks with these modular robots. With each hardware prototype, we evaluate our proposed algorithm's performance in several different aspects, including: (1) its capability of adaptive to external perturbations; (2) how different initial conditions would affect the time required to complete the desired tasks; (3) its scalability to the number of modules.

7.1. Self-balancing Table and Terrain-adaptive Bridge

In this section, we describe two self-adaptive structures formed by modular robots: a self-balancing table (shown in Figure 5(b)) and a terrain-adaptive bridge (Figure 5(c)). In both applications, modules form "legs" of the structures and cooperatively maintain the top surfaces level. Figure 5(a) shows the concept diagram of the self-balancing table. The red modules in the figure form a substrate of the table surface while blue modules form the table's legs. We program each leg of the table, composed of three modules, as an agent. There is a tilt sensor in between each pair of legs to measure neighboring agents' orientation relationships. Agents are programmed to maintain zero tilt angles with respect to all of their neighbors. More specifically, our algorithmic procedure can be divided into the following three steps.

Step 1 (Initialization) Modules start sending messages to their neighbors. Based on these messages, modules can identify whether they belong to the surface or part of the leg (agent). The top module (dark red modules in Figure 5(a)) in each leg coordinates the control of all modules in the leg. We call these modules *pivot modules*.

Step 2 (Sensing) We call modules forming the surface in between two neighboring agents (legs), a_i and a_j, a *surface group*, denoted as S_{ij}. There is a tilt sensor available on each surface group. We denote the sensor reading on S_{ij} as θ_{ij}. At each time step modules in a surface group start propagating their sensor readings to neighboring modules. These messages contain sensory information and are aggregated by pivot modules. After collecting all of the messages, the pivot modules then transmit the aggregated information to the rest of the modules in the leg (agent).

Step 3 (Actuation) After receiving this sensory information, each agent uses the data as input from which to compute appropriate actuation parameters for each module. Each agent's state at t, $x_i(t)$, is used to denote the length of the leg. The Agents' control law is derived from Equation (2) and can be formally written as

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} \theta_{ij}. \tag{13}$$

Now the generalized feedback function $g(\cdot)$ is simply θ_{ij}, the tilt angle of the surface between agents a_i and a_j. We can in fact easily show that $g(\cdot)$ satisfies Equations (3)–(5): (1) $\theta_{ij} = 0$ only if $x_i = x_j$ (two legs have the same height); (2) The tilt angle on the surface is proportional to the height difference between two legs and has the same sign; (3) θ_{ij} is anti-symmetric.

The terrain-adaptive bridge follows a similar formulation as the self-balancing table. Each leg of the bridge is viewed as an agent, and the set of surface modules in between each pair of legs is equipped with a tilt sensor. All agents execute the control law in Equation (13).

Experimental Results In the first experiment, we examine how quickly and accurately the self-balancing table responds to consistent, rapid environmental changes. In this

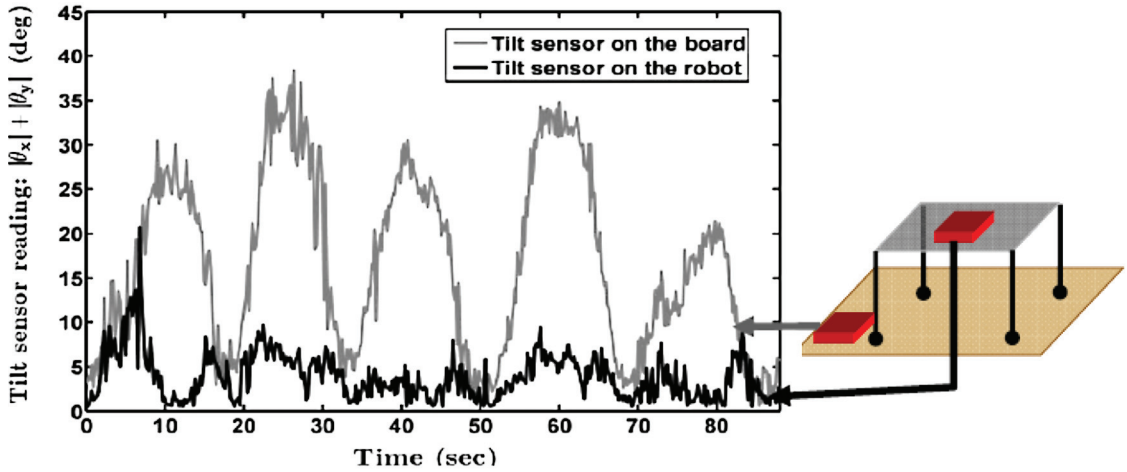

Fig. 6. The self-balancing table response time to repeated environment changes. The robot was able to maintain its surface's tilt angle within 5–8° even when the board is tilted 30–40° over a few seconds.

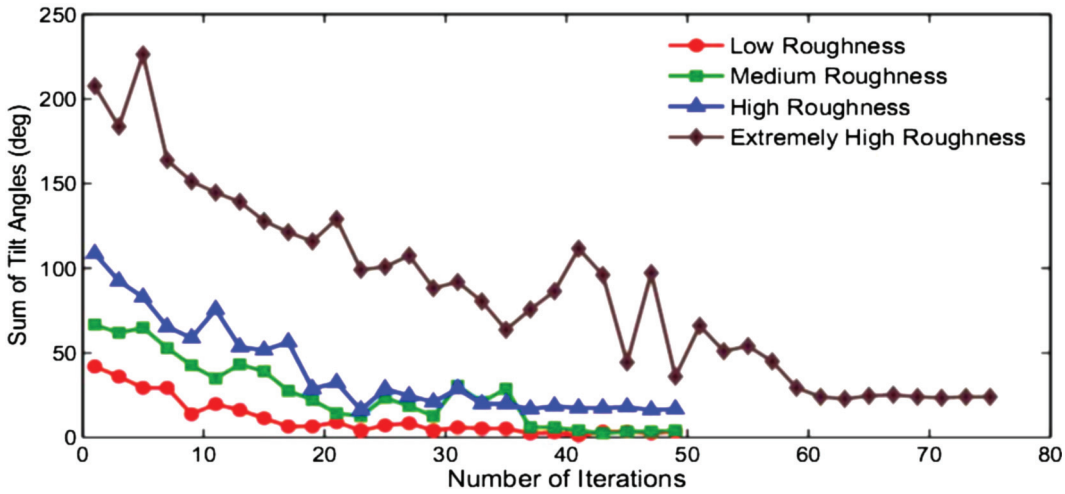

Fig. 7. The bridge structure's response time to achieve its surface's levelness. In low, medium, and high roughness cases, it is capable of achieving the average of passive links' tilt angles $\leq 3^\circ$ after 40 iterations. The average initial tilt sensor values for low, medium, high, and extremely high roughness are 5°, 8°, 12°, and 17°, respectively. In extremely high roughness case, it achieves the same level of levelness after 60 iterations.

experiment, we fix the robot's four supporting groups to a rigid board. We repeatedly change the orientation of the board to examine the robot's response. One additional tilt sensor is mounted on the board to record environmental changes. This sensor does not supply input to the robot. Empirically, the sensors we use are somewhat noisy, especially under high-speed motion (e.g. the first 5 seconds of Figure 6).

Agents are programmed to maintain a level surface, i.e. tilt angles in the x-axis and y-axis, θ_x and θ_y, equal to zero at all times. Therefore, $|\theta_x| + |\theta_y|$ is an error measure of how far the table surface is from a level state. Figure 7 shows the results of the experiment. We can see that even when the tilt angle of the floor is changed by 30–40° over a few seconds, the table is able to quickly respond and keep the surface level. The table never tilts more than 5–8°.

In the second experiment, we examine the terrain-adaptive bridge's adaptiveness to terrains of different roughness levels. The modules were assembled in the same bridge-like configuration as Figure 5(c). In this experiment, the assembly is placed on uneven terrain with the goal of achieving a flat top surface: this requires all tilt sensors to be zero. To test how fast the bridge can adapt to terrains of different roughness, we sequentially increase the roughness of the terrain by adding more underlying bricks. Since the surface achieves levelness when all tilt sensor readings are equal to zero, we use the sum of all tilt sensor readings' absolute values as our levelness measure, as shown in the following equation:

$$\epsilon = \sum_{\forall i,j \in N_i} |\theta_{ij}|.$$

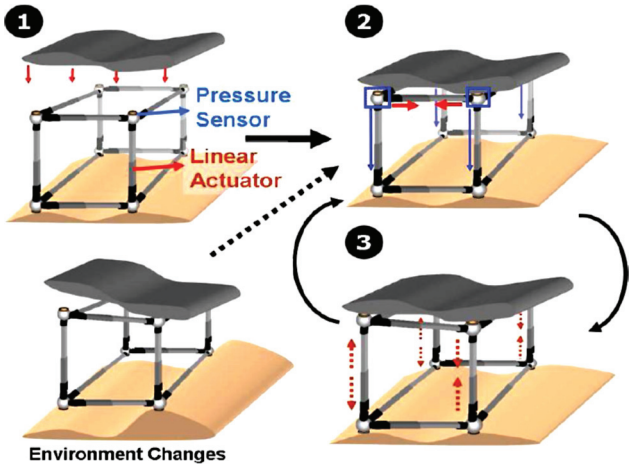

Fig. 8. The algorithmic overview of the pressure-adaptive column. Step 1: an unknown object is placed on the robot. Step 2: Each agent sends its pressure reading to neighbors. Step 3: Each agent continuously adapts based on its neighbor's states.

Figure 7 shows the number of iterations required to achieve levelness versus different levels of roughness of the underlying terrain. Since the tilt sensors we use are somewhat noisy, we define the bridge as having achieved levelness when each passive link's tilt angle is on average smaller than 3°. We can see from Figure 7 that the bridge surface is capable of achieving levelness after running the control algorithm for 40 iterations in low, medium, and high roughness cases (x-axis represents the number of iterations and y axis is). In extremely high roughness case, it achieves levelness after approximately 60 iterations.

In the above applications, if agents' actuation range does not allow them to fully achieve the desired task (e.g. due to the height range of each agent is limited), agents will collectively achieve a state within its actuation range that is closest to the goal state.

7.2. Pressure-Adaptive Column

One potential application for modular robotics is a reconfigurable structure: a structure that can reconfigure itself to achieve functional requirements irrespective of external environment changes. Examples include forming supporting structure for a building that absorbs uniform force, and a modular seat back that adapts to apply uniform pressure on the user. Motivated by this application area, we construct a pressure-adaptive column hardware with a modular robot.

As shown in Figure 8(1), each agent is equipped with a linear actuator whose length can be precisely controlled and a pressure sensor that can sense the force applied on each agent. We program agents to achieve a state where each agent absorbs equal force when an unknown object or structure is placed on it.

The algorithmic overview of the self-adaptive process is shown in Figure 8.

Step 1 (Initialization) An unknown object is placed on the robot.

Step 2 (Sensing) Each agent starts exchanging current pressure sensor feedback with its neighbors.

Step 3 (Actuation) Each agent computes its actuator's new parameters based on the sensor feedback that it receives from all of its neighbors. Each agent iterates between Step 2 and Step 3. When the environment starts changing again, the robot automatically goes back to Step 2.

In Step 3, each agent runs a control law to change the length of its linear actuator x_i based on sensory feedback from its neighbors. This control law can be written as

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} (\theta_j - \theta_i). \qquad (14)$$

Here, the feedback function g is simply $g(\theta_j, \theta_i) = \theta_j - \theta_i$, and g satisfies the conditions in Equations (3)–(5), since: (1) when $\theta_i = \theta_j$, $g(\theta_j, \theta_i) = \theta_j - \theta_i = 0$; (2) when sensory θ_i is smaller than θ_j, $g(\theta_j, \theta_i) > 0$ such that agent a_i increases its length to increase its pressure state θ_i. Therefore, $g(\theta_i, \theta_j)$ leads actuator to move in the same direction as minimizing $\theta_j - \theta_i$; (3) the g function is anti-symmetric. Therefore, the control law (Equation (14)) will allow the robot to converge to the desired state.

Experimental Results In the pressure-adaptive column experiment, we examine the control law's convergence property with different initial conditions. Each agent is equipped with a pressure sensor (force sensing resistor) with sensory readings ranging from 0 to 900. Agents are programmed to achieve equal pressure with their neighbors. The weight of the unknown object is roughly 1.5 lb. The robot starts in three different configurations, such that the number of initial contacting agents is different, ranging from one to three[5]. We define

$$\epsilon = \max_i \theta_i - \min_i \theta_i,$$

the difference between maximal and minimal sensory reading among agents, as a measure of distance from reaching consensus. We can see from Figure 9 that ϵ decreases from 800 to around 100 after 1,000 iterations ($\sim$ 10 seconds in real time) in all three cases. We note that the sensor we use is very noisy and sensitive to slight perturbations of the linear actuators. Therefore, we set the α in Equation (14) to be a very small constant to avoid the column from being oversensitive to perturbations. This naturally leads to a longer convergence time. We also note that the larger fluctuations in the blue curve are primarily due to the object significantly shifted its center of mass when more agents contact it.

7.3. Modular Gripper

In this section, we illustrate another application: a modular gripper. The gripper is capable of reconfiguring itself

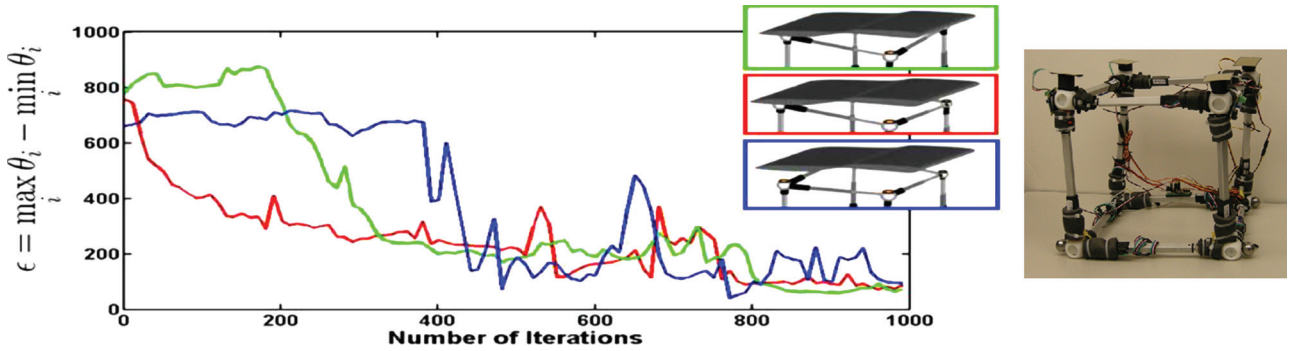

Fig. 9. Pressure-adaptive column with different initial conditions. We let the number of initial contacting agents be one (green), two (red), and three (blue) respectively and examine how the column respond with different initializations. After 1,000 iterations of running the control law, the distance measure, ϵ, decreases to less than 100. Right: Physical prototype of the pressure-adaptive column.

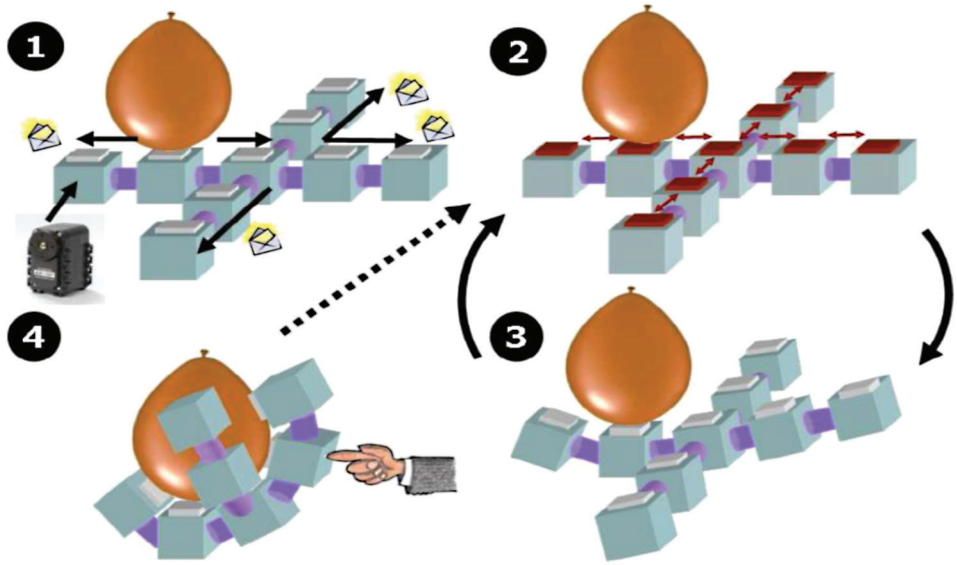

Fig. 10. The algorithmic overview of the grasping task. Step 1: The first module starts sensing the presence of the object. It starts sending messages to it neighbors. Steps 2 and 3: Agents perform iterative sensing and actuation until they converge to the desired state. Step 4: When the robot is perturbed by exogenous force, it goes back to Step 2.

to grasp an object using distributed sensing and actuation. The control law design follows a similar procedure as in the previous example. However, the analysis of the convergence property is somewhat different due to the fact that each agent's actuation affects more than its own sensor state.

As shown in Figure 10(1), a modular gripper is composed of a chain of modular agents, where each agent is equipped with a rotary servo and a pressure sensor. The goal of the agents is to grasp a convex object, e.g. a balloon, such that all of the agents apply equal pressure θ_p($\theta_{\min} \leq \theta_p \leq \theta_{\max}$).

The illustration of the algorithmic procedure is shown as Figure 10. It can be divided into the following steps.

Step 1 (Initialization) One of the agents starts sensing the object. When the sensor reading is in between $\theta_{\min}$ and $\theta_{\max}$, it starts sending messages to neighboring agents. Upon receiving a message, each agent propagates the message and its ID to neighboring agents (shown in Figure 10(1)). We denote R_i as the agent ID from which agent a_i receives the message and S_i as the ID of the agent to which it sends the message.

Step 2 (Sensing) Each agent starts sending its pressure sensor reading to its neighbors (as shown in Figure 10(2)). We note that this sensory reading message is passed only between an agent and its immediate neighbors.

Step 3 (Actuation) Each agent computes its new actuation state based on the sensor readings that it receives from its neighbors. The control law run by each agent is

$$x_i(t+1) = x_i(t) + \alpha \cdot (\theta_{R_i} - \theta_i). \tag{15}$$

Agents iterate between Step 2 and Step 3 until all agents have reached the desired state. When the robot is perturbed by exogenous force, it goes back to Step 2.

The control law we showed in Equation (15) satisfies condition 1, since sensory feedback $g(\cdot) = \theta_{R_i} - \theta_i = 0$ only when agent a_i's sensor reading is equal to its neighbor a_{R_i}. In addition, g is also anti-symmetric. However, it is non-trivial to evaluate whether the control law satisfies condition 2. This is primarily due to the fact that all agents are connected together in a chain and changing an agent's actuation parameter can potentially change more than its own sensor state. We refer readers to the appendix of Yu and Nagpal (2009) for details of the convergence proof.

Most of the controllers designed for grasping tasks have used a centralized architecture. The decentralized and modular robot approach that we propose here allows the whole system to adapt to local perturbations more efficiently. In addition, given any initial contacting module, the gripper is able to form a grasping configuration that conforms to the shape of the object. This control scheme is also applicable to different kinds of gripper configurations. We provide demonstrations of these capabilities in the following experimental results.

Experimental Results Here, we present an empirical evaluation of this control framework when applied to a modular gripper. Agents are programmed to apply equal pressure on a balloon. We first assess its adaptability towards repetitive perturbations. We then test the convergence properties of Equation (15) under different initial conditions and different numbers of agents.

Adaptation Towards Perturbations. After all agents achieve the desired state, we start applying an external force on the gripper. Figure 11 shows ϵ versus time as the gripper encounters four different perturbations. We can see that ϵ decreases to less than 3% after 50–70 iterations in each case. This shows that our decentralized control law can efficiently lead agents to recover from exogenous perturbations. We specifically note that the gripper achieves faster adaptation than the pressure-adaptive column is due to: (1) each agent's actuation has a long-range effect, so an agent is likely to assist more than its neighbors in the process; (2) the rotary servos we use here have better precision than the linear actuators.

Different Initial Conditions. We connect the agents to form a "cross" configuration as shown in Figure 12(a)–(d). We let different agents start to touch the balloon to examine the system's behavior under different initial conditions. Figure 12(a)–(h) shows a sequence of robot configurations while grasping the object. We use k to denote the first activated (contacted) agent's index. Let $\theta_i(t)$ be the pressure sensor reading of agent i at time t. After the first contact between the object and the robot, the object is held in place. This will lead all other agents to approach agent a_k's sensor reading $\theta_k(t)$ while reaching the consensus state. Therefore, we define the *percentage from achieving the task*, ϵ, as a ratio of the current distance for all agents to reach the first contacted agent's sensor reading $\theta_k(t)$ to the initial distance. This can be formally written as

$$\epsilon = \frac{\sum_i \|\theta_i(t) - \theta_k(t)\|}{\sum_i \|\theta_i(0) - \theta_k(0)\|}.$$

Figure 13 shows ϵ's value changing over time. We can see that the agents are capable of converging to $\sim 3\%$ from completing the task after 180 iterations, regardless of initial conditions. From this figure, we can also see that there is a correlation between the position of the first activated agent and the convergence time. The red curve shows the case when the middle agent is first activated. The maximum number of communication hops between it and all other agents is two. In this case, agents achieve faster convergence as compared with the case where the maximal hop is three and four, respectively (blue and green curve).

Scalability. We further evaluate the algorithm's scalability with the number of agents. In Figure 14, we increase the number of agents from five to nine. We can see from the figure that there is no significant increase in convergence time when we increase the number of agents. Here ϵ converges to less than 3% after 150 iterations in all three cases. However, we can see that the convergence time is slightly shorter in the five-agent case in which the diameter of the agent network is only one (in contrast to two in the other cases). This coincides with theoretical results in Section 6 that decreasing the diameter of the agent network can increase the convergence speed.

7.4. Other Distributed Constraint-maintenance Tasks

The framework we propose here can be used to solve more tasks than consensus-type tasks as presented in the previous sections. In Equations (13), (14), and (15), we set either $\theta^*_{ij} = 0$ or $\Delta^*_{ij} = 0$, for all $i,j \in N_i$ such that agents eventually achieve the same (sensory) states. In fact, if $\theta^*_{ij} \neq 0$ or $\Delta^*_{ij} \neq 0$. This allows us to use such a framework to solve a wider range of tasks. Furthermore, the connectivity graph of agents can be extended from 2D planar graphs, e.g. self-balancing table or terrain-adaptive bridge, to three-dimensional connectivity. Here we illustrate two of these tasks.

3D Relief Display This is an application where a modular robot forms arbitrary shapes as a novel form of 3D media and visualization. When modules are connected in a way such that some modules form a flexible surface while the rest form the supporting structure of the surface, they can act as a "relief" display. One key advantage of our approach is that once the desired shape is formed, the system will autonomously maintain the desired shape even if the underlying terrain is dynamically changing. As shown in Figure

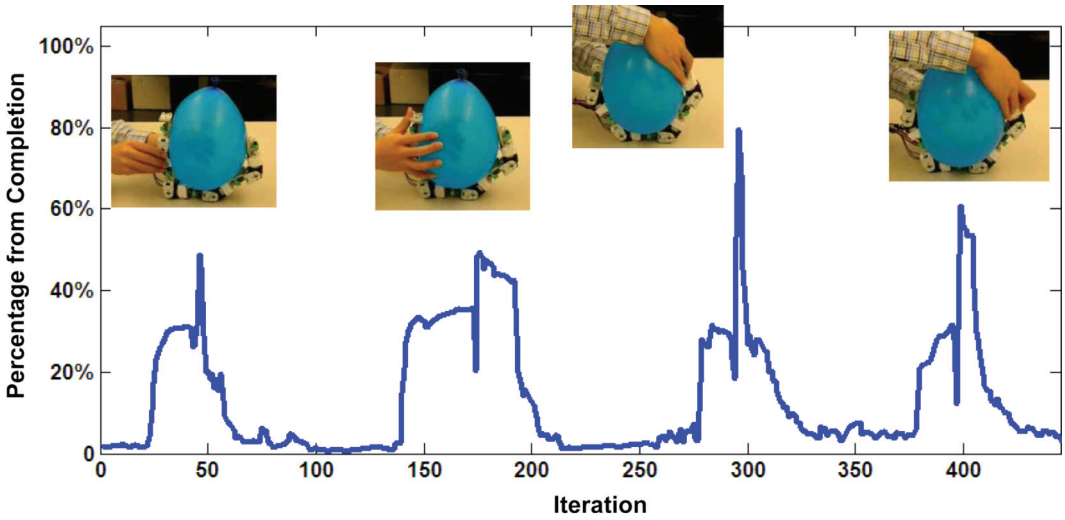

Fig. 11. After the robot has reached the desired state, we constantly perturb the gripper by applying force. The robot is able to re-adapt after being perturbed.

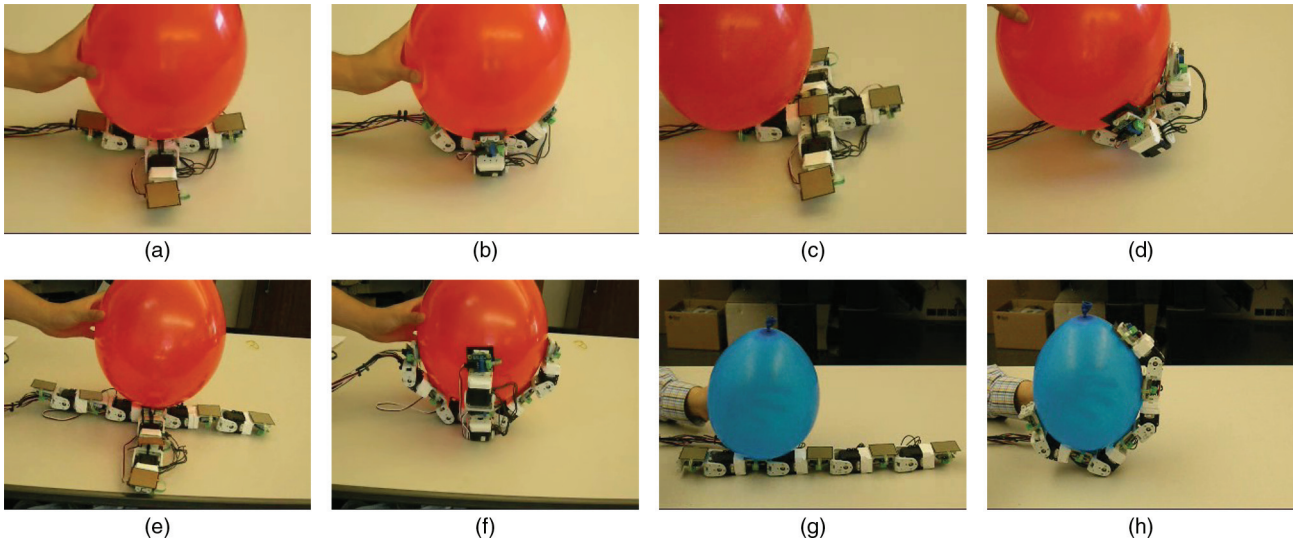

Fig. 12. (a)–(d) Different initial conditions for the grasping task. The robot is capable of completing the task irrespective of initial conditions. (e), (f) Scalability experiment. More modules are added to the robot. Empirically, the robot scales successfully to the number of module agents. (g), (h) The robot performs the grasping task with a different gripper configuration.

15(a), to form the desired shape, we need to specify each surface agent's local inter-agent constraints according to the corresponding location of the desired shape. Figure 15(a)–(c) shows several different complicated shapes that are rendered in simulation when we have 16,000 modules in our system.

Adaptive Building Structure In the previous applications, we connect modules to form a surface. One interesting futuristic application is to extend modules' connectivity to three dimensions to form an adaptive building structure. In this simulation (Figure 15(d)), modules form several surfaces that correspond to different levels of the building while some linear modules form the supporting pillars of the building. From the figure we can see that the building floors autonomously adapt to remain level while the underlying terrain changes.

8. Discussion

In this section, we provide a discussion of several promising extensions and potential limitations of our approach. We then discuss, in a larger scope, tradeoffs between this decentralized self-organizing and centralized hierarchical control approaches. This discussion provides deeper insights into the application scope of this approach, as well as its pros and cons.

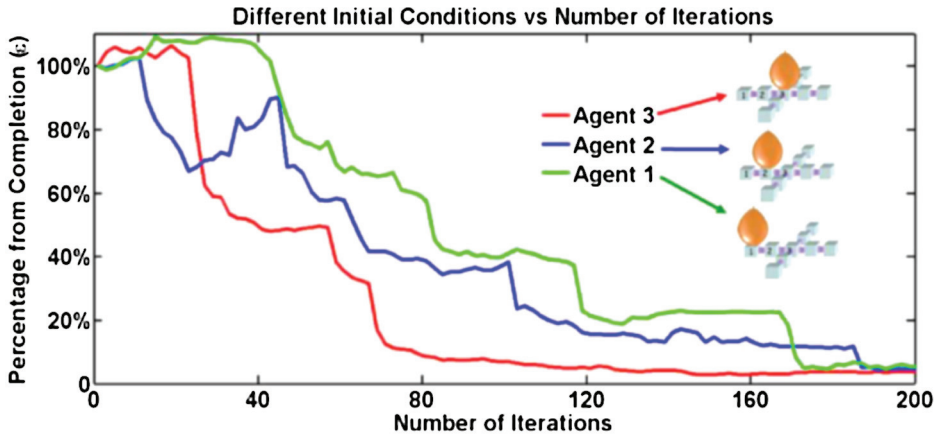

Fig. 13. Experiments with different initial conditions. After ∼ 180 iterations, agents are capable of achieving less 3% distance from the consensus state in all three cases.

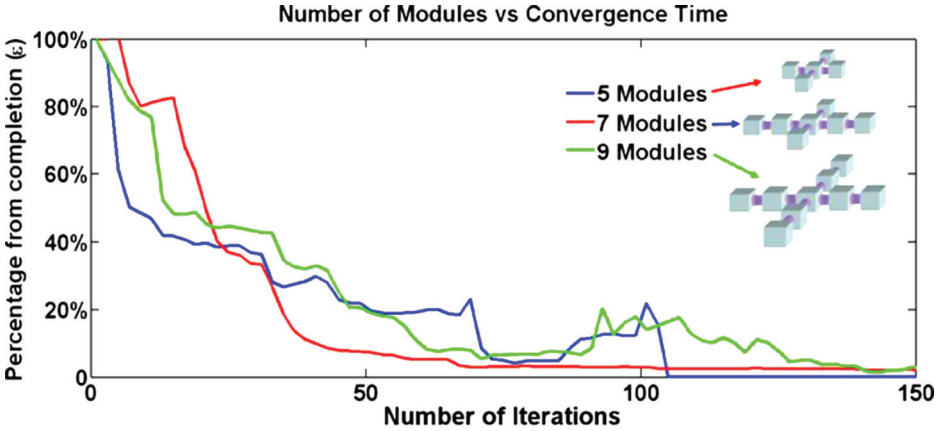

Fig. 14. Scalability experiment. The decentralized algorithm is scalable with the number of agents. On the other hand, the network structure might affect the convergence speed. In the seven- and nine-agents cases, the diameter of the network is two and it leads to longer times for the robot to complete the task.

8.1. Potential Extensions

When modules are equipped with different sensors and actuators, there are many other applications that can be generalized from this framework. Here, we illustrate some of them. (1) Light-adaptive modular panel: we can change the pressure sensors we mount on the robot to light sensors. Each agent is programmed to achieve the same light absorption as its neighbors. A similar concept can be applied to many environmental sensory adaptation tasks. (2) Adaptive prosthetic structure: existing prosthetic devices for children require manual reconfiguration to adapt to limb growth. If force (pressure) sensors are mounted on the device, it is possible to construct a self-reconfigurable prosthetic device. (3) A similar concept can be applied to a support structure for plants. The structure is capable of self-adaptation based on the growth of the plant and lighting conditions. (4) The described framework can also be potentially applied to solve dynamic tasks, such as locomotion. One straightforward generalization is to view dynamic tasks as a sequence of self-adaptations. In Yu and Nagpal (2009), we described how one can use this framework to program a strut-based modular robot to achieve adaptive locomotion.

8.2. Potential Limitations and Challenges

The type of tasks we illustrate with this framework share one similarity: they can be expressed as a single consensus state, e.g. modular gripper grasping tasks. This is the main limitation of the current framework. To further extend such a framework to solve more sophisticated tasks, it is necessary to have a mechanism that can decide among *different* consensus states (also called *biased* consensus states) based on different external states. For example, we might need different pressure distributions for a modular gripper to optimally grasp different types of objects, instead of using a uniform pressure distribution to grasp all objects. Understanding the scope of self-adaptation tasks and of this limitation is an important future direction of this research.

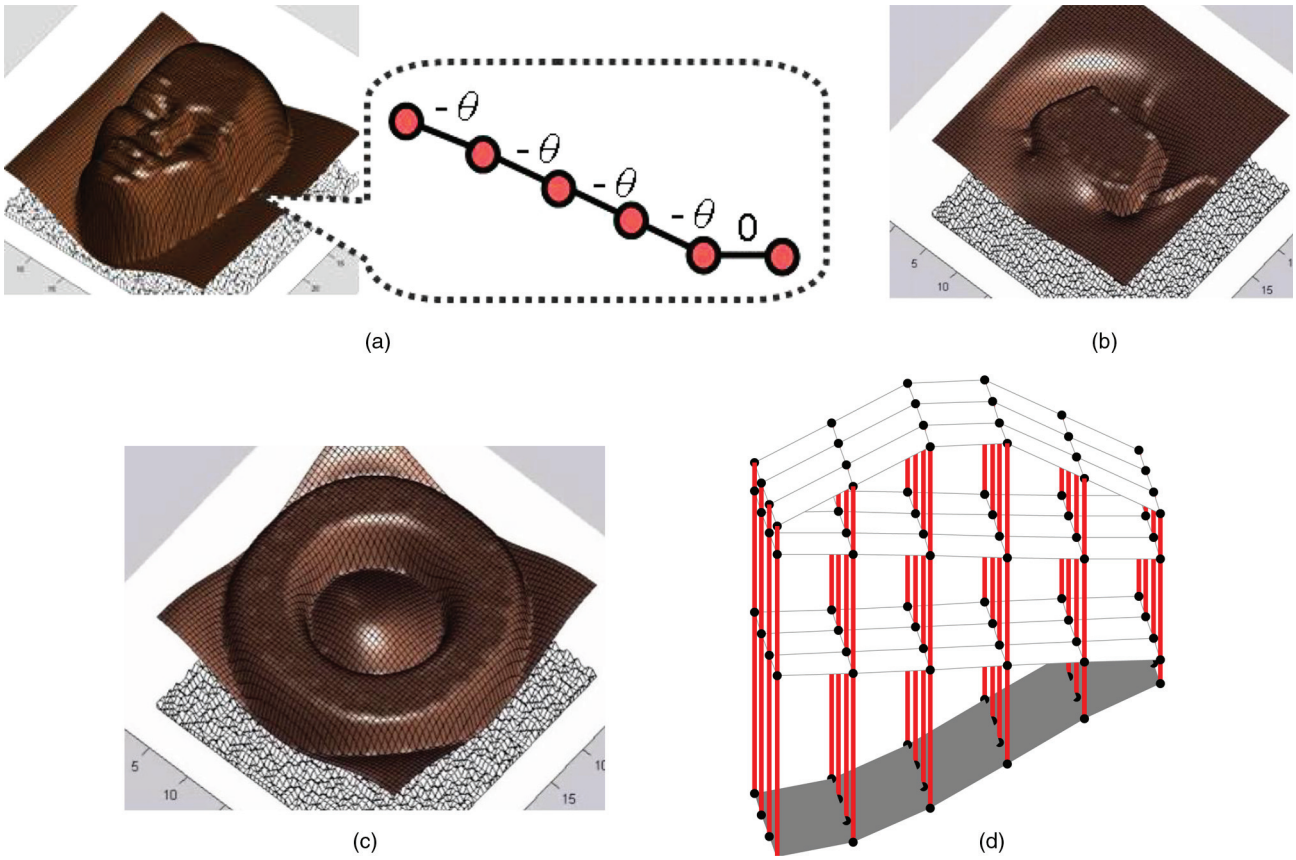

Fig. 15. Our framework can be viewed more generally as a distributed constraint-maintenance framework. (a) A human face shape can be specified with inter-agent constraints. The shape is formed by 16,000 modules. (b), (c) Several different shapes generated from the same task specification scheme as (a). (d) Agents can be connected to form 3D structures. An adaptive-building structure formed by the modules, and the gray region indicates uneven terrain.

One limitation of this approach is that the agent task must be specified in advance. In the tasks where global goal require all agents achieve the same states, it is easy to specify such tasks in terms of local constraints. This is primarily due to the fact that such global goals are complete when all agents achieve the same state as its neighbors. For global goals that need agents to achieve different states, translating a global tasks into local constraints amongst agents can also be challenging. One interesting future direction is to design a task compiling framework to automatically generate local constraints from global objective functions. While using generalized distributed consensus algorithm, one needs to appropriately design feedback function g based on three principles we present here. Another interesting direction is to automate this feedback function design process to derive g based on our desired task.

In some cases, the agent's structure might change over time and the original task specification might no longer be valid. In the case when all agents need to reach consensus states, agents' task specifications are not required to changed (since $\Delta_{ij} = 0$ for all i,j). However, it is still hard to achieve role replacement in biased consensus tasks ($\Delta_{ij} \neq 0$ for all i,j). One interesting future direction would be designing a mechanism that allows agents to change their modular structures arbitrary and dynamically assign task accordingly.

8.3. Self-organizing versus a Centralized Approach

An important question in networked multi-agent systems is whether to use a decentralized self-organizing approach, such as that described here, where agents iteratively communicate and react to arrive at a solution, or to use a centralized tree-based approach where a root agent collects all of the information from other agents. This question is not only relevant to modular robots, but also to robot swarms and sensor networks. It also applies to many problems, from shape formation to time synchronization. Using our results, we can describe the tradeoffs between these two approaches.

For the centralized algorithm, we assume that a root agent collects all of the information from all other agents using a spanning tree, computes a final state for every

agent, and then disseminates the results back to them. This results in two costs: (a) a communication cost of collecting/disseminating information; and (b) a computation cost for the root node. In most homogenous multi-agent systems, each agent has fixed communication and computation power. For the kinds of tasks we consider here, communication is often a more severe bottleneck: if an agent can only collect a constant amount of information per unit time, then the time to collect all the agents' states is $O(n)$ (n: number of agents) and not $O(d)$ (d: diam(G)). This cost is paid for every shape change, regardless of the distance between the initial and desired states. This results in poor performance in the case of small perturbations, where information must travel all the way to the root agent before it can be resolved. In contrast, the communication cost of the decentralized algorithm described here is $O(t)$ (t: iteration number), which depends on both topology and distance from goal. The relationship between topology and performance in some cases is worse than the centralized case. However, if the distance from goal is small (e.g. a small perturbation or slowly changing environment) then the system reacts rapidly in only a few iterations even when n is large. Figure 4) shows the self-organizing approach's capabilities of adaptation to large and small perturbations. We can see from the figure that the number of iterations required to achieve the task remains low in small perturbation cases, even when we significantly enlarged the size of the network.

This suggests that for consensus-like problems, while decentralized algorithms may pay a significant start-up cost to achieve a steady state, they are extremely reactive to perturbations. Thus, they are more appropriate when the goal is to maintain constraints over long periods of time under uncertain and changing conditions, rather than produce a solution once (as shown in applications of Section 7). Finally, they are more robust and less complex to implement in situations where agent errors and topology changes are common.

9. Conclusions

We have presented a self-organizing framework inspired by biological collective behaviors for self-adaptation tasks in modular robotics. We show that modular robots' adaptive tasks can be captured by distributed constraint maintenance when the robot can be abstractly viewed as a sensor–actuator network. Such a formulation allows the robot to exploit its distributed sensors to efficiently adapt to various environmental conditions, similar to the way biological systems achieve scalable self-adaptation. In addition, it can be implemented in a wide range of modular robot systems, including those with indirect relationships between their sensors and actuators.

We also have presented unified controller design principles for our framework and have further analyzed the various theoretical properties of this class of algorithms, including correctness, scalability, and robustness. In comparison with a centralized approach, this approach has a strong advantage in robustness and reactivity. Based on our theoretical understanding, we implemented our framework in a diverse set of modular robot applications, including: (1) self-adaptive structures; (2) a pressure-adaptive column; (3) an adaptive modular gripper; (4) other sensor–actuator network applications, e.g. an adaptive prothetic device. Our results show that such a control scheme is robust toward real-world sensor and actuator noise. These applications represent a small subset of what is achievable within this framework.

We plan to extend this work in several directions. First, we have illustrated several potential applications to which we can further apply this framework, including a self-adaptive support structure. Second, we are interested in applying this framework to other distributed robotics applications beyond modular robots, such as a team of mobile robots. Finally, we are interested in exploring more deeply the advantages and disadvantages of decentralized algorithms. One potential control solution based on this study is a mixed strategy that is composed of centralized and decentralized controllers. This allows us to exploit the strengths of both approaches. For example, a humanoid robot utilizes a centralized controller to reach an object, and decentralized controllers run on the gripper, allowing it to grasp the object.

Acknowledgements

This research is supported by an NSF EMT Grant (No. 0829745) and Harvard's Wyss Institute for Biologically-Inspired Engineering.

Notes

* The preliminary version of this work was described in Yu and Nagpal (2008, 2009).

1 For example, in sensor networks time synchronization, an agent can observe its neighbors' firing time and thus control its own firing time.

2 For example, if the agent's actuator is a linear actuator, $x_i(t)$ would represent the length of the actuator. If the actuator is a rotary one, it would represent the angle of the actuator.

3 We can show that A and L have the same eigenvectors. Let v_i be the ith eigenvector of L and $\mu_i = 1 - \alpha\lambda_i$. Here $Lv_i = \lambda_i v_i \Rightarrow Av_i = (I - \alpha L)v_i = (1 - \alpha\lambda_i)v_i = \mu_i v_i$. In addition, $\mu_1 = 1$ is a simple eigenvalue of A with associated eigenvector: $\mathbf{1}$.

4 We note that the assumption that $A(t)$ is symmetric (condition 3) can be relaxed, but the upper bound on convergence rate is less tight. The proof of this case is based on the theory of non-homogenous stochastic matrix products (Seneta 1981); the product $A(t)\cdots A(2)A(1)$ will converge to a rank-one matrix with exponential rate. The recent result in Cao et al. (2008) explicitly determines an upper bound on convergence rate.

5 In the case of one or two initial contacting agents, we provide slight external support to the object to prevent the rest of the agents from contacting the object.

6 The ith element of X^*: $x_i^* + \sum_{a_j \in N_i} \phi_{ij}(t)(x_j^* - x_i^*) - \sum_{a_j \in N_i} \phi_{ij}(t)\,\Delta_{ij}^* = x_i^*$.

7 Since all agents are identical and execute the same control law, it is safe to switch their indices.

References

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38: 393–422.

Bertsekas, D. P. and Tsitsiklis, J. (1989). *Parallell and Distributed Computation*. Upper Saddle River, NJ: Prentice-Hall.

Bishop, J., Burden, S., Klavins, E., Kreisberg, R., Malone, W., Napp, N. and Nguyen, T. (2005). Programmable parts: a demonstration of the grammatical approach to self-organization. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Alberta, Canada, pp. 3684–3691.

Bojinov, H., Casal, A. and Hogg, T. (2000). Emergent structures in modular self-reconfigurable robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, San Francisco, CA, pp. 1734–1741.

Bongard, J., Zykov, V. and Lipson, H. (2007). Resilient machines through continuous self-modeling. *Science*, 314(5802): 1118–1121.

Cao, M., Morse, A. S. and Anderson, B. D. O. (2008). Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization*, 47(2): 575–600.

Chung, F. R. K. (1994). *Spectral Graph Theory*. Providence, RI, American Mathematical Society.

Degesys, J., Rose, I., Patel, A. and Nagpal, R. (2007). Desync: self-organizing desynchronization and tdma on wireless sensor networks. In *Proceedings of the Sixth International Conference on Information Processing in Sensor Networks (IPSN)*, Cambridge, MA, pp. 11–20.

Fax, J. A. and Murray, R. M. (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9): 1465–1476.

Goldstein, S. C., Campbell, J. and Mowry, T. C. (2005). Programmable matter. *IEEE Transactions on Computer*, 38(6): 99–101.

Groß, R., Bonani, M., Mondada, F. and Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6): 1115–1130.

Jadbabaie, A., Lin, J. and Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6): 988–1001. http://dx.doi.org/10.1109/TAC.2003.812781.

Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K. and Kokaji, S. (2004). Distributed adaptive locomotion by a modular robotic system, M-Tran II. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, Sendai, Japan, pp. 2370–2377.

Lucarelli, D. and Wang, I.-J. (2004). Decentralized synchronization protocols with nearest neighbor communication. In *Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems (SenSys)*, New York, NY, 62–68. http://dx.doi.org/10.1145/1031495.1031503.

Lyder, A., Garcia, R. and Støy, K. (2008). Mechanical design of Odin, an extendable heterogeneous deformable modular robot. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, pp. 883–888.

Moeckel, R., Jaquier, C., Drapel, K., Upegui, A. and Ijspeert, A. (2005). Yamor and bluemove—an autonomous modular robot with bluetooth interface for exploring adaptive locomotion. *Proceedings of International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, Istanbul, Turkey, pp. 685–692.

Mohar, B. (1991a). Eigenvalues, diameter, and mean distance in graphs. *Graphs and Combinatorics*, 7: 53–64.

Mohar, B. (1991b). The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*. New York, John Wiley & Sons, Inc., pp. 871–898.

Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K. and Kokaji, S. (2002). M-TRAN: Self-reconfigurable modular robotic system. *IEEE/ASME Transaction on Mechatronics*, 7(4): 431–441.

Okubo, A. (1986). Dynamical aspects of animal grouping: Swarms, schools, flocks, and herds. *Advances in Biophysics*, 22: 1–94. http://dx.doi.org/10.1016/0065-227X(86)90003-1.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3): 401–420.

Olfati-Saber, R., Fax, J. A. and Murray, R. M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1): 215–233. http://dx.doi.org/10.1109/JPROC.2006.887293.

Pamecha, A., Stein, D. and Chirikjian, G. (1996). Design and implementation of metamorphic robots. *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, Irvine, CA.

Potts, W. K. (1984). The chorus-line hypothesis of manoeuvre coordination in avian flocks. *Nature*, 309(5966): 344–345. http://dx.doi. org/10.1038/309344a0.

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21(4): 25–34.

Rus, D., Butler, Z., Kotay, K. and Vona, M. (2002). Self-reconfiguring robots. *Communications of the ACM*, 45(3): 39–45.

Schwager, M., Slotine, J.-J. E. and Rus, D. (2008). Consensus learning for distributed coverage control. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, pp. 1042–1048.

Seneta, E. (1981). *Non-negative Matrices and Markov Chains*. Berlin, Springer-Verlag.

Shen, W.-M., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M. and Venkatesh, J. (2006). Multimode locomotion for reconfigurable robots. *Autonomous Robots*, 20(2): 165–177.

Shen, W.-M., Will, P., Galstyan, A. and Chuong, C.-M. (2004). Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, 17(1): 93–105.

Shimizu, M., Ishiguro, A. and Kawakatsu, T. (2005). Slimebot: A modular robot that exploits emergent phenomena. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, pp. 2982–2987.

Støy, K., Shen, W.-M. and Will, P. (2002). How to make a self-reconfigurable robot run. *Proceedings of the First*

International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Bologna, Italy, pp. 813–820.

Turgut, A., Celikkanat, H., Gokce, F. and Sahin, E. (2008). Self-organized flocking with a mobile robot swarm. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Estoril, Portugal, pp. 39–46.

Vona, M. and Rus, D. (2001). Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1): 107–124.

Yim, M., Eldershaw, C., Zhang, Y. and Duff, D. G. (2004). Limbless conforming gaits with modular robots. *International Symposium on Experimental Robotics (ISER)*, Singapore.

Yu, C., Haller, K., Ingber, D. and Nagpal, R. (2008). Morpho: A self-deformable modular robot inspired by cellular structure. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, pp. 3571–3578.

Yu, C.-H. and Nagpal, R. (2008). Sensing-based shape formation tasks on modular multi-robot systems: A theoretical study. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Estoril, Portugal, pp. 71–78.

Yu, C.-H. and Nagpal, R. (2009). Self-adapting modular robotics: A generalized distributed consensus framework. *Proceedings of ICRA*.

Yu, C.-H., Willems, F.-X., Ingber, D. and Nagpal, R. (2007). Self-organization of environmentally-adaptive shapes on a modular robot. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, pp. 2353–2360.

Zykov, V., Mytilinaios, E., Adams, B. and Lipson, H. (2005). Self-reproducing machine. *Nature*, 435(7038): 163–164.

Appendix A. Proof of Theorem 1

We first show that analyzing Equation (6) is equivalent to analyzing a linear dynamical system without the bias vector. The optimality condition:

$$X^* = A \cdot X^* + \tilde{b} \tag{16}$$

can be rewritten as $\alpha L \cdot X^* = \tilde{b}$. We use the graph Laplacian property that when G is connected, $\text{rank}(L) = n - 1$ with $\text{null}(L) = \mathbf{1}$. We can add an additional constraint to the system based on *mass conservation* property of the agent state: $\sum_i x_i(0) = \sum_i x_i^* = \mathcal{C}$ (since A is a row stochastic matrix, and $\sum_i \tilde{b}_i = 0$). The new linear system with the additional constraint becomes $\alpha L' \cdot X^* = \tilde{b}'$. Since the new constraint lies in the null space of L, $\text{rank}(L') = n$ and there exists a unique X^* for every initial condition $X(0)$. We subtract the optimality condition from Equation (6):

$$Y(t+1) = A \cdot Y(t) \tag{17}$$

where $Y(t) = X(t) - X^*$. The following proof follows the procedure of Olfati-Saber et al. (2007). Since L is a real symmetric matrix, the well-known Courant–Fischer theorem yields

$$\lambda_2(L) = \min_{\|x\|=1,\, x \perp \mathbf{1}} \frac{x^{\mathrm{T}} L x}{x^{\mathrm{T}} x}.$$

As $Y(t)^{\mathrm{T}} \cdot \mathbf{1} = 0$ for all t, we can utilize the results from Olfati-Saber et al. (2007) and show that

$$\max_{Y(t)} \frac{Y(t)^{\mathrm{T}} A Y(t)}{Y(t)^{\mathrm{T}} Y(t)} = \mu_2(A) < 1, \tag{18}$$

where $\mu_2(A)$ denotes the second largest eigenvalue of A. Define a Lyapunov function $V(t) = \sum_i (x_i - x_i^*)^2 = Y(t)^{\mathrm{T}} Y(t)$. Now, following Equation (18), we obtain

$$\begin{aligned} V(t+1) &= (AY(t))^{\mathrm{T}} (AY(t)) = Y(t)^{\mathrm{T}} A^2 Y(t) \\ &< (\mu_2(A))^2 Y(t)^{\mathrm{T}} Y(t) = (\mu_2(A))^2 V(t). \end{aligned} \tag{19}$$

Here $Y(t)$ converges to zero ($X(t)$ converges to X^*) with exponential rate at least $\mu_2(A) < 1$.

Appendix B. Proof of Theorem 2

The $A(t)$ matrix in Equation (9) can be written as $I - L_w(t)$ where $L_w(t)$ is a *weighted* graph Laplacian matrix. The properties of the weighted graph Lalpacian are similar to those of the standard Laplacian (Mohar 1991b). When G is connected, $\text{rank}(L_w(t)) = n - 1$ with $\text{null}(L_w(t)) = \mathbf{1}$. Since $\sum_i \tilde{b}(t) = 0$ and $A(t)$ is stochastic for all t, the mass conservation constraint still applies. We can solve X^* with the same procedure as the proof of Theorem 1 with respect to a particular $A(t)$. We note that the obtained X^* satisfies $x_j^* - x_i^* = \Delta_{ij}^*$ for all $a_i, a_j \in N_i$, the optimality condition $X^* = A(t) X^* + \tilde{b}(t)$ will hold for all[6] t. We can again obtain the new dynamics system:

$$Y(t+1) = A(t) Y(t).$$

Since $L_w(t)$ is a real symmetric matrix for all t. Applying the Courant–Fisher theorem to the analogously defined Lyapunov function, a similar derivation to Equations (18) and (19) yields

$$V(t+1) \leq (\mu_2(A(t)))^2 V(t) \leq (\max_t \mu_2(A(t)))^2 V(t).$$

Therefore, the convergence rate is at least the maximal value of the second largest eigenvalues among the $A(t)$.

Appendix C. Proof of Theorem 4

Expanding out the definition of 2-norm, we have

$$\begin{aligned} \|X(0) - X^*\|^2 &= \sum_i (x_i(0) - x_i^*)^2 \qquad (20) \\ &= \left(\sum_i x_i^2(0)\right) + \left(\sum_i (x_i^*)^2\right) \\ &\quad - 2 \sum_{i<j} x_i(0) x_j^*. \qquad (21) \end{aligned}$$

Since $x_i(t) > 0$ for all t, both $x_i(0)$ and x_i^* must be non-negative. Thus, $\sum_{i<j} x_i(0) x_j^*$ is non-negative. Similarly, $\sum_i x_i^2(t) \leq \left(\sum_i x_i(t)\right)^2$ for all t. Hence,

$$\|X(0) - X^*\|^2 \leq \left(\sum_i x_i(0)\right)^2 + \left(\sum_i x_i^*\right)^2.$$

By the conservation of mass principle mentioned above, $\sum_i x_i^* = \sum_i x(0)_i = \mathcal{C}$, so we have

$$\|X(0) - X^*\|^2 \leq 2\mathcal{C}^2.$$

Taking square roots of both sides of the above and substituting into Equation (11) yields the result.

Appendix D. Proof of Theorem 5

We have seen in Theorem 1 that we can always convert biased consensus dynamics, i.e. $\tilde{b} \neq \mathbf{0}$, to equivalent consensus dynamics with $\tilde{b} = \mathbf{0}$. We therefore only need to prove the case when $\tilde{b} = \mathbf{0}$.

We first construct collective dynamics as in Equation (6). We use a_k to denote the agent whose actuator fails in the process. It thus has a fixed state x^*. We use n to denote the number of agents in the system. For the sake of proof convenience, we switch the index of agent a_k with the last agent a_n. Here a_k and a_n's set of neighbors[7], N_k and N_n, are also correspondingly changed. a_n is now the agent with actuation failure. Since agent a_n's state is fixed at x^*, agents' states can be written as a n-dimensional column vector: $X(t) = (x_1(t), \ldots, x_{n-1}(t), x^*)'$.

We can further write the collective dynamics of the agents as follows:

$$X(t+1) = A \cdot X(t) = \prod_{m=1}^{t} A \cdot X(1). \tag{22}$$

This can be further expanded as follows:

$$\begin{aligned} \prod_{m=1}^{t} A \cdot X(1) &= \begin{pmatrix} F & L \\ 0 & 1 \end{pmatrix}^t X(1) \\ &= M(t) \cdot X(1) = \begin{pmatrix} P(t) & Q(t) \\ 0 & 1 \end{pmatrix} X(1), \end{aligned} \tag{23}$$

where F is an $(n-1) \times (n-1)$ matrix and $F_{ii} = 1 - \alpha \cdot |N_i|$, and $F_{ij} = \alpha$. Here L is an $n-1$ column vector and $L_i = \alpha$ if $a_i \in N_n$, where N_n is simply the set of neighbors of the failed agent. To prove the convergence property of Equation (22), we first define:

- matrix maximum norm, $\|M\|_{\max} = \max_i \sum_j M_{ij}$;
- matrix minimum norm, $\|M\|_{\min} = \min_i \sum_j M_{ij}$;
- $d_{\max}$, the maximal hop distance between an agent and the failed agent.

To prove convergence, we use the property that the product of row stochastic matrices is still row stochastic. We can see that A is a row stochastic matrix with a positive diagonal ($A_{ij} \geq 0$ and $\sum_j A_{ij} = 1$, for all i), thus $M(t)$ is also row stochastic with $\sum_j M_{ij}(t) = 1$, for all i. We can further expand

$$Q(t) = \sum_{m=1}^{t} F^{m-1} \cdot L. \tag{24}$$

Since the failed agent can still communicate, the communication graph G stays connected. From Equation (24), we can immediately see that $F^{d_{\max}+1} > 0$ and thus $\|Q(t)\|_{\min} > 0$ when $t \geq d_{\max} + 2$. We can also see from Equation (24) that $\|Q(t)\|_{\min}$ monotonically increases with t after time $d_{\max} + 2$. Since $M(t)$ is row stochastic, $\left(\sum_j P_{ij}(t)\right) + Q_i(t) = 1$, for all i. Therefore, $0 < \|P(t)\|_{\max} < 1$. Since $\|Q(t)\|_{\min}$ monotonically increases with t after time $d_{\max} + 2$ and $\|P(t)\|_{\max} + \|Q(t)\|_{\min} = 1$, the maximum norm of $P(t)$ decreases at least every $d_{\max} + 2$ time steps. Thus, $\lim_{t\to\infty} \|P(t)\|_{\max} = 0$. This leads to the conclusion that $\lim_{t\to\infty} Q(t) = \mathbf{1} \Rightarrow \lim_{t\to\infty} x_i(t) = x^*$, for all i. All agents will eventually have the same state x^* as the failed agent a_n.

Appendix E. Proof of Theorem 6

The proof of Theorem 6 follows the same procedure as Theorem 5. We now have more than one agent that is fixed at certain states due to actuation malfunction. Similarly, we index these k agents from $a_{n-k+1}, a_{n-k+2}, \ldots, a_n$ and their fixed states are $x_1^*, x_2^*, \ldots, x_k^*$. We can also write down the collective dynamics in the following form:

$$\begin{aligned} X(t+1) = A \cdot X(t) &= \prod_{m=1}^{t} A \cdot X(1) = \begin{pmatrix} F & L \\ 0 & I \end{pmatrix}^t X(1) \\ &= M(t) \cdot X(1) = \begin{pmatrix} P(t) & Q(t) \\ 0 & I \end{pmatrix} X(1). \end{aligned} \tag{25}$$

Different from the proof of Theorem 5, matrices F and $P(t)$ are now $(n-k) \times (n-k)$ matrices and L and $Q(t)$ are now $(n-k) \times k$. We denote $d_{\max}$ as the maximal hop distance between an agent and its closest failed agent in our graph G. We can follow the same procedure as Theorem 5 to show that $\|Q(t)\|_{\min} > 0$ for $t \geq d_{\max} + 2$. Using the stochastic matrix property, we again show $0 < \|P(t)\|_{\max} < 1$ for $t \geq d_{\max} + 2$. This allows us to show that $\lim_{t\to\infty} \|P(t)\|_{\max} = 0$ and $\sum_j Q_{ij}(t) = 1$ for $t \to \infty$.

Since $Q(t)$ is now multi-column, we need to show that each element in $Q(t)$ converges to a stable state: $Q_{ij}(t) \to Q_{ij}^*$, for all i,j. Owing to the fact that the bottom-right submatrix of $M(t)$ is a $k \times k$ identity matrix, we can get $Q_{ij}(t) \geq Q_{ij}(t-1)$ for all i,j; and $Q_{ij}(t)$ is monotonically nondecreasing with t for all i,j. Therefore, as $\sum_j Q_{ij}(t)$ approaches $\mathbf{1}$, $Q_{ij}(t)$ will also approach a fixed value Q_{ij}^*, for all i,j. Since each failed agent's fixed state can be different, each agent converges to a potentially different fixed state:

$$\lim_{t\to\infty} x_i(t) = \sum_j Q_{ij}^* x_j^*.$$

Appendix F. Proof of Theorem 7

The proof of Theorem 7 is the same as Theorem 6 of Olfati-Saber et al. (2007). Olfati-Saber et al. (2007) prove the case when the agent topology is dynamic and periodically connected, all agents' states will converge to a single

value. The periodically connected property (due to temporary communication failures) we assume here is the same as Olfati-Saber et al. (2007) which states that the union of all graphs over a finite sequence of intervals are connected.

Appendix G. Convergence Rate versus Topology

We provide several graph topological factors versus λ_2 ($\mu_2 = 1 - \alpha\lambda_2$). We define several factors that are not mentioned in the article. The mean distance between two vertices:

$$\bar{\rho} = \frac{1}{n(n-1)} \sum_{\forall u,v \in V(G), u \neq v} d(u,v).$$

The maximal degree sum of two connected vertices: $d^+_{\max} = \max\{\deg(u) + \deg(v) \mid uv \in E(G)\}$.

We can summarize λ_2's relationships with various topological factors from (Mohar 1991a; Chung 1994) as follows:

	Best Case	**Worst Case**
Number of agents (n)	$O(\sqrt[D]{n})$	$O(1/n)$
Diameter (D)	$O(1/D)$	$O(1/D)$
Mean distance ($\bar{\rho}$)	$O(1/\bar{\rho})$	$O(1/\bar{\rho})$
Maximal degree ($d_{\max}$)	$O(d_{\max})$	n/a
Maximal degree sum ($d^+_{\max}$)	$O(d^+_{\max})$	n/a