Biologically-Inspired Control for Self-Adaptive Multiagent Systems

A dissertation presented

by

Chih-Han Yu

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

May 2010

Thesis advisor
Radhika Nagpal

Author
Chih-Han Yu

Biologically-Inspired Control for Self-Adaptive Multiagent Systems

Abstract

Biological systems achieve robust and scalable group behaviors, such as flocking, through local interactions amongst vast numbers of unreliable agents. In these systems, each agent acts autonomously and interacts only with its neighbors, yet the global system exhibits coordinated behavior. Large-scale multi-agent systems, e.g. distributed robot systems, are similar to these biological systems, in that their overall tasks must be achieved by coordinating many independent agents. One important question to ask is: How can we program large numbers of agents to achieve collective tasks, and at the same time adapt to dynamic conditions like living systems?

In this thesis, I develop a biologically-inspired control framework for multi-agent systems to achieve coordinated tasks in a scalable, robust, and analyzable manner. I focus on the type of tasks in which agents must use distributed sensing to solve collective tasks and to cope with changing environments. This task space can be formulated more generally as distributed constraint-maintenance on a networked multi-agent system such that it can capture various multi-agent tasks. I show how one can exploit the locality of this formulation to design nearest-neighbor agent control, based on simple sensing, actuation and local communication.

I further theoretically determine several important properties of this decentralized control, such as convergence, scalability, and reactivity. Using these results, we can provide precise statements on how the approach scales and how quickly agents can adapt to perturbations. Practically, I demonstrate this approach with various modular robots, a type of robotic system composed of many connected and autonomous modules. With this approach, a diverse set of modular robot challenges can be similarly addressed, including environmentally-adaptive shape formation, grasping, and collective locomotion. Altogether, the thesis provides examples of how one can systematically design decentralized control for multi-agent tasks and autonomous robot control, and provides a deeper understanding of the contrast between centralized and decentralized algorithms.

Contents

Citations to Previously Published Work

Chapters 2, 3, and 4 have appeared in the following four papers:

C. -H. Yu, F. -X. Williems, D. Ingber, R. Nagpal. "Self-organizing Environmentally-adaptive Shapes on a Modular Robot", In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), pages 2353–2360, 2007

C. -H. Yu and R. Nagpal. "Sensing-based Shape Formation Tasks on Modular Multi-robot Systems: A Theoretical Study", In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS), pages 71–78, 2008

C. -H. Yu and R. Nagpal. "A Self-organizing Framework for Modular Robot Adaptive Tasks: Theory and Application", *in submission*, 2010

C. -H. Yu and R. Nagpal. "Biologically-Inspired Control for Multiagent Self-Adaptive Tasks", In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (AAAI), 2010

Chapter 5 and 6 are primarily based on the following three papers:

C. -H. Yu, K. Haller, D. Ingber, R. Nagpal. "Morpho: A Self-deformable Modular Robot Inspired by Cellular Structure", In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), pages 3571–3578, 2008

C. -H. Yu, and R. Nagpal. "Self-Adapting Modular Robotics: A Generalized Distributed Consensus Framework", In *Proceedings of International Conference on Robotics and Automation* (ICRA), pages 1881–1888, 2009

C. -H. Yu, J. Werfel, R. Nagpal. "Coordinating Collective Locomotion in an Amorphous Modular Robot", In *Proceedings of International Conference on Robotics and Automation* (ICRA), 2010

Chapter 7 has been published in the following paper:

C. -H. Yu, J. Werfel, R. Nagpal. "Collective Decision Making in Multi-Robot Systems by Implicit Leadership", In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS), 2010

Acknowledgments

I sincerely thank my advisor, Radhika Nagpal, for her tremendous support and advice over the past few years. As a true advisor, she stimulated my passion in science and successfully guided me to find and maximize my potential in research. Under her guidance, I learned not only how to conduct research but also how to communicate scientific ideas. Her input has positively influenced my professional capabilities as well as my personal characteristics.

I also thank my thesis committee members Profs. David Parkes, Manuela Veloso, and Robert J. Wood. David's tremendous intelligence and broad knowledge provided me with the best possible research consultation. Manuela's guidance allowed me to think about my research in a much broader content. Rob's advice on practical issues of my research allowed me to think much more deeply about the application side of my work. I also sincerely appreciate Dr. Justin Werfel for not only being a great collaborator but also a true mentor and friend. His comments and suggestions on my work over the last few years have greatly helped me to shape and crystallize my research work. Prof. Donald Ingber has provided me with many biological insights and established a great interdisciplinary research environment – the Wyss Institute. I also thank Prof. H. T. Kung for his valuable comments on my qualifying exam, which subsequently led to fruitful theoretical results in Chapter 4 of this thesis. I also want to thank my former research advisors, Profs. Andrew Ng, Yi-Ping Hung, and Chu-Song Chen, for guiding me to the field of Artificial Intelligence.

I am proud to be part of the Harvard SSR group. I thank SSR group members for fun memories and valuable research feedback, including Julius Degesys, Tyler Gibson, Nick Hoff, Ankit Patel, Chandana Paul, Kirstin Petersen, Ian Rose, Michael Rubenstein, and Dan Yamins. I am fortunate to work with several talented master's and undergraduate researchers, including François-Xavier (FX) Willems, Becky Belisle, and Kristina Haller. I particularly thank FX, who offered tremendous support in the early stages of this thesis project. I also thank Drs. Leia Stirling and Eugene Goldfield for their collaboration on the soft orthotics project.

My graduate student life would not be so colorful and interesting without all my lovely friends: Chia-Lin Chang, Chia-Yung Su, Dori Lin, Chia-Hua Lee, Nelson Lin, Chi-Hung Cheng, Kobe Chien, Bor-Rong Chen, Jr-Shin Li, Tsung-Han Lin, Anna Huang, Georgios Theocharous, Jason Yu, Miki Lee, Lino Huang, Joyce Yang, Chuan-Heng Hsiao, Mirko Bordignon, Kai-min Chung, Sharon Lee, Angie Lee, Akio Ishiguro, Kuo-Hua Huang, Hung-Chieh Chou, Samuel Chang, Guille Canas. I also thank all my Harvard AIRG colleagues who have constantly given me great research insights.

I am fortunate to have had the company of Wan-Ling Winnie Lee during my graduate student life. Winnie has provided me with her full love and support during my long journey in the US, and her persistence in research also deeply motivates me to pursue excellence. Her wisdom is my best oracle and always helps me to overcome obstacles along my way.

Finally, I am also fortunate to have been born and raised in the most supportive family. My parents' and brother's love and support really make me feel like I am at home even when I am physically on the other side of the earth. In particular, I thank my dad, Chao-Tang Thomas Yue, for being a perfect role model for my life in both career and family aspects. My wish is that I can contribute to society as much as he has someday in my career.

Dedicated to my parents – Chao-Tang Yue and Chiu-Yuan Huang
and Wan-Ling Winnie Lee

Chapter 1

Introduction

In nature, biological systems achieve robust and scalable group behaviors through simple local interactions amongst vast numbers of unreliable agents. These systems exhibit an incredible ability to adapt and optimize group function as the environment changes, despite the fact that each individual agent has only limited sensing and motion capabilities. One common technique by which biological systems achieve such amazing group behaviors consists of each agent executing a simple and decentralized approach. For example, each bird in a migrating flock decides its traveling direction based on its local neighbors' traveling directions; in this manner, the whole group can travel cooperatively for thousands of miles and adapt to changing environments along the way (Fig. 1.1 (a)). In plants, each cell senses external light and changes its local shape while causing the overall plant structure to deform to maximize photosynthesis (Fig. 1.1 (b)). Such a strategy also allows these living multiagent systems to be more robust because there is *no* single agent that plays a critical role in the group task and that can consequently become a potential point of failure for the system.

Modern manufacturing technologies have allowed low-cost production of large-scale multiagent systems that are composed of numerous autonomous agents, from modular robots composed of many identical modules with sensing and actuation capabilities [83], to swarm robot systems consisting of many autonomous mobile robots [40], to sensor networks formed by vast numbers of sensor nodes that can compute and communicate [1]. These systems have similar characteristics to those of the biological groups I describe above because their overall tasks are necessarily achieved by coordinating the actions of independent agents. For such systems to act autonomously, there is a key challenge to be addressed: *how do we program a vast number of agents to achieve global functions cooperatively while adapting to external environments dynamically*? In this context, traditional algorithmic techniques for coordinating small numbers of powerful and robust agents usually cannot be applied.

(a) (b)

Figure 1.1: Examples of biological group behaviors. (a) Birds flocking: Each individual decides its motion based on only local observation while the whole group can travel together and be extremely adaptive to external perturbations. (b) Plant's light adaptation: Each cell senses light and changes its shape, and the global shape of the plant deforms and adapts to the light source. (Image source: Harvard SSR group)

If there is an algorithmic framework that can achieve this goal, one can potentially program many large-scale multiagent systems to act autonomously while achieving *robustness* and *adaptability* similar to those of biological systems. One can imagine designing a modular robot that can autonomously transform its shape and locomotion strategy to efficiently traverse different terrains, a modular robot bridge that can become a "living structure" that senses and autonomously adapts to a sudden environmental impact such as an earthquake, or even a "living (programmable) material" that can transform its structure based on its distributed sensing of the external environment.

One approach is to designate a centralized coordinator to collect information from all agents, then compute and disseminate agent states to the whole group. However, such a strategy interferes with the system's scalability and robustness: the coordinator can easily become a communication bottleneck, and it is also a potential point of failure for the system. On the other hand, simple and decentralized strategies adopted by biological systems can potentially provide effective solutions to address this challenge. In this thesis, I propose a decentralized framework based on self-organizing principles in biology by means of which large-scale agent systems can achieve coordinated tasks. To utilize such biological principles in designing an effective algorithmic approach requires addressing several open questions:

- How can we effectively translate self-organizing principles in biology into an

algorithmic framework?

- Can we obtain a theoretical guarantee that the desired global state will result? Furthermore, what are the factors that affect task performance? These are important questions to ask when implementing decentralized strategies because all agents act based on only purely local information in parallel and can potentially affect each other.

- Finally, how can we express various multiagent tasks and goals with this framework? What are the scope, strength, and limitations of this decentralized approach?

I address the above questions in the context of multiagent *self-adaptive* tasks in which a large number of agents utilize their distributed sensors and actuators to solve tasks and cope with dynamic environmental changes. I demonstrate that many multiagent tasks can be formulated with this framework; my proposed solution allows these systems to achieve these tasks in a *scalable*, *robust* and *analyzable* manner. This thesis can also serve as an example or road map for translating a biological concept into an effective algorithmic approach.

1.1 Thesis Accomplishments and Approach

The overall goal of this thesis is to *develop a unified framework that allows various multiagent systems to collectively achieve self-adaptive tasks and gain a thorough theoretical understanding of the approach.* By self-adaptive tasks, I refer to the type of tasks in which agents autonomously exploit local sensed information to solve tasks in cooperation with other agents and to cope with dynamic conditions.

Towards this goal, I first formulate this problem more generally as distributed constraint maintenance on a networked multiagent system. Such a formulation allows the framework to model and simplify tasks on a variety of distributed agent applications. It also allows us to exploit locality of the task to derive decentralized agent control that is based on a simple sensing and actuation strategy (The concept diagram for this approach is shown as Fig. 1.2). I theoretically analyze important aspects of this decentralized approach, including convergence, scalability, and robustness. I also outline sufficient conditions that guarantee agent control laws' correctness for different networked multiagent systems. This analytical framework allows one to have a concrete understanding of the strengths, limitations, and scope of this class of decentralized approaches.

In practice, I have built several modular robot systems and have used them to achieve various self-adaptive tasks within this control framework. I demonstrate that

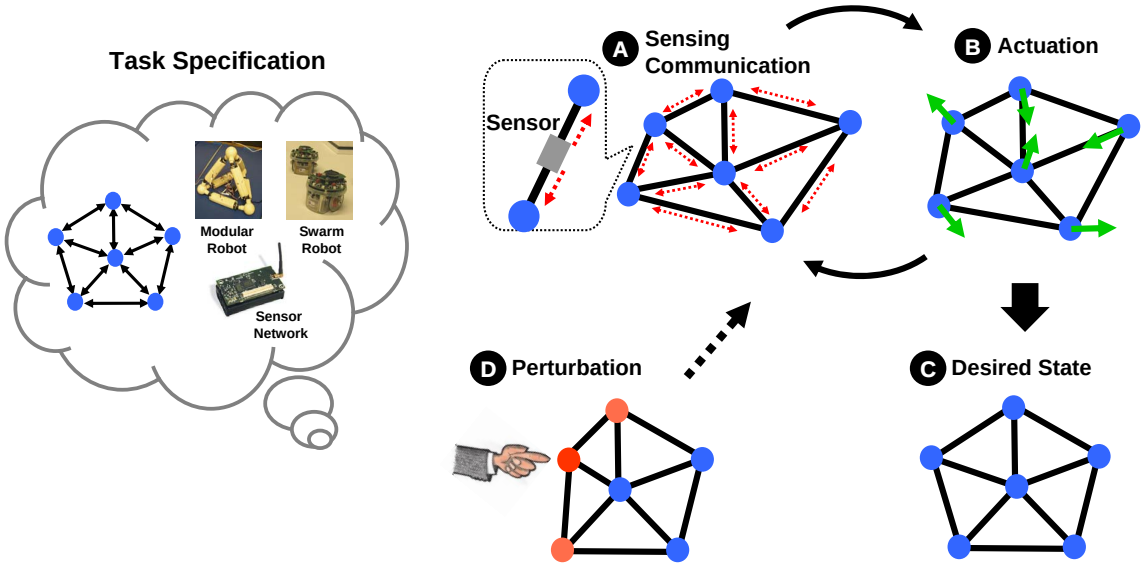


Figure 1.2: Concept diagram of the approach. A vertex indicates an agent and an edge indicates a neighborhood communication link. The global task is specified by inter-agent relationships, and such task specification scheme can be used to describe tasks in many multiagent systems, e.g., modular robots, swarm robots, and sensor networks. Each agent iteratively sense and communicate with neighbors (A) and perform action to change its state (B) until all agents' local constraints are satisfied and the desired state is reached (C). When some agents are perturbed, agents restart the process until the desired state is reached again (D).

this approach allows modular robots to cooperatively carry out various tasks, for example, the formation of self-adaptive structures and adaptive locomotion. I further describe how the framework applies to a wider area of applications, e.g., adaptive medical devices and smart architecture. Finally, I broaden the scope of this framework by extending it to scenarios in which (1) agents are required to solve a sequence of similar tasks and (2) agents have different levels of importance in accomplishing group tasks.

With respect to both its theoretical and its practical aspects, my thesis suggests that this decentralized approach outperforms centralized algorithms in tasks that require constant adaptation to dynamic environments. This thesis also provides a deeper understanding of the contrast between centralized and decentralized algorithms in multiagent tasks and autonomous robot control.

1.2 Specific Contributions

In this thesis, I make original contributions in the areas of algorithm, theory, and application. More specifically, the contributions of this thesis are as follows:

- **A Self-organizing Framework for Various Self-adaptive Tasks**: To develop this framework, I first show that a diverse set of agent self-adaptive tasks can be expressed as distributed constraint-maintenance on a multiagent system. The global task can be expressed as a set of local constraints between neighboring agents. When each agent satisfies its local constraints, the system achieves the desired task. This largely simplifies the complexity of solving collective tasks amongst agents. I then propose a simple agent control law that allows agents to carry out the desired tasks by exploiting locality in this task formulation. I show that this algorithm is a generalization of a class of algorithms in control theory called distributed consensus which is a process by which a network of agents can come to an agreement via distributed communication [48].

- **Generalization to Broader Range of Applications**: I extend this framework's scope by proposing a generalized distributed consensus (GDC) framework and deriving unified principles for designing controllers in distributed sensing-actuation applications. These principles provide clear guidelines for the application of this approach to novel situations. This generalization also significantly enlarges the application area of traditional distributed consensus-type framework and allows various systems that can be viewed as sensor-actuator networks to be captured by this framework.

- **Theoretical Study on Correctness, Scalability, and Robustness of the Algorithms**: I first show that interactions between agents can be modeled as a collective linear dynamical system. I then prove the correctness of this approach and show that all agents will converge to the desired states at an exponential rate using results from spectral graph theory [10]. I further analyze the scalability of this approach by characterizing how various parameters related to agent topology, for example, agent network diameter and the number of nodes, determine the system's convergence speed by extending results from [42]. Finally, I study robustness of the algorithm by analyzing how agents behave in the face of communication and actuation failures using properties of stochastic matrices products. This theoretical study also increases our understanding of the type of tasks for which this class of algorithms is well-suited. The theoretical and algorithmic contributions of this thesis lead to a unified framework that provably allows distributed agents to achieve various self-adaptive tasks with a common decentralized control strategy.

- **Empirical Verification on Robot Applications**: I apply this framework to several modular robot hardware prototypes and simulations. I show that the modules can collectively form adaptive structures such as a self-balancing table and a terrain-adaptive bridge (Fig. 1.3 (a - b)). I further demonstrate that this framework can be extended to agents (modules) that employ indirect mapping between sensors and actuators, for example, a modular gripper that can dynamically adjust its grasping posture and maintain uniform pressure on the target object (Fig. 1.3 (c)). I demonstrate through several examples that this framework can be extended to dynamic agent tasks, such as modular robots' locomotion. I show that a modular tetrahedral robot can perform terrain-adaptive locomotion by completing a sequence of self-adaptations (Fig. 1.3 (d)). I further show that a 2D modular robot (a single module shown as Fig. 1.3 (e)) assembled with arbitrary geometry can collectively achieve coordinated locomotion through modules' reaching locomotion pattern consensus. Finally, I provide examples of the application of this framework to many distributed autonomous systems beyond modular robotics.

- **Extension to Broader Multiagent Decision-Making Scenarios**: In the original framework, I assume that all agents contribute equally to the global tasks. Here I relax this assumption and extend this framework to capture multiagent decision-making scenarios in which some agents, which I call *implicit leaders*, obtain privileged information. These implicit leaders are required to play more important roles in group consensus. This also includes a generalization of our original algorithms to cases in which different agents have different degrees of influence on the global tasks. I further show how this extended framework can address conflicting implicit leaders and achieve fast decision-making in rapidly changing environments. This approach provides a simple way of allowing the agents to reach collective decisions regarding dynamic information.

1.3 An Illustrative Example

I now provide a simple example to illustrate (1) how this algorithmic framework can be used to solve a distributed agent task, and (2) how various theories developed in this framework can contribute to one's understanding of various task properties. Similar algorithmic and analytical procedures can be applied to all tasks captured by this framework, such as the examples shown in Fig. 1.3.

Let us assume that we have a bridge or walkway that is formed by many identical robotic modules (Fig. 1.4). In this structure, each leg (module) is an agent, and there is a tilt sensor mounted between each pair of neighboring agents to capture inter-agent relationship. The global task of the bridge is to maintain its surface level at all times,

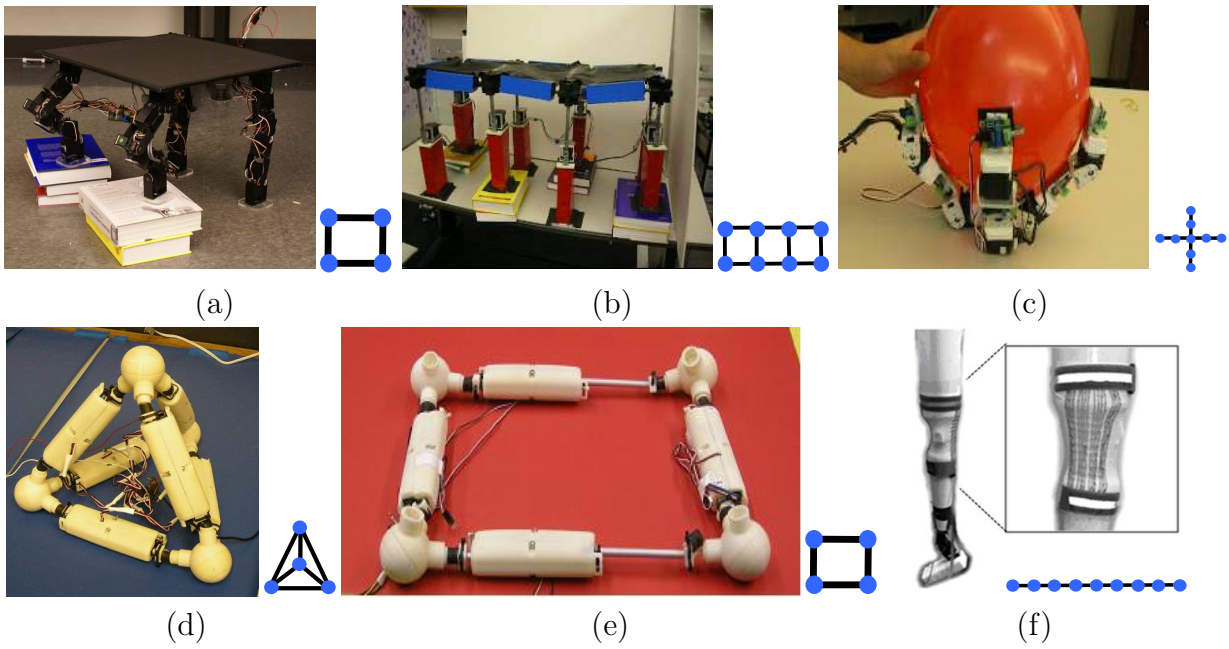


Figure 1.3: Different self-adaptive applications that I demonstrate with this thesis. Their corresponding agent graph topologies are shown on the right. (a) Self-balancing table. The module-formed legs are capable of maintaining table surface level irrespective of tilt changes. (b) Terrain-adaptive modular bridge. The bridge is able to achieve a flat bridge surface when it is placed on a rough terrain. (c) A modular gripper. The modules can cooperatively form a configuration that grasps a fragile object, e.g., a balloon. (d) A modular tetrahedral robot. The robot can perform locomotion by a sequence of modules' self-adaptations to the terrain. (e) 2-D modular robot that performs coordinated locomotion. (f) Adaptive orthotic. The orthotic contains a sensor-actuator network and can deform its shape to correct pathological gaits.

irrespective of underlying uneven terrain (right figure of Fig. 1.4 (a)). A skeletal view of the overall system is shown in the left diagram of Fig. 1.4 (b); the global task is specified by assigning each agent to maintain zero tilt angles with respect to each of its neighbors. In fact, this system can be abstractly represented by an agent graph in which each vertex indicates an agent and each edge indicates a neighborhood link.

Correctness: In this decentralized framework, each agent executes an identical control law by performing iterative sensing and actuation. One can design the agent control law by following guidelines provided in this thesis and obtain an associated correctness guarantee: the derived agent control law allows agents to complete the desired tasks from all initial conditions when the agent graph is *connected*. In our bridge robot example, the bridge's agent graph is a connected grid graph (right diagram of the above figure).

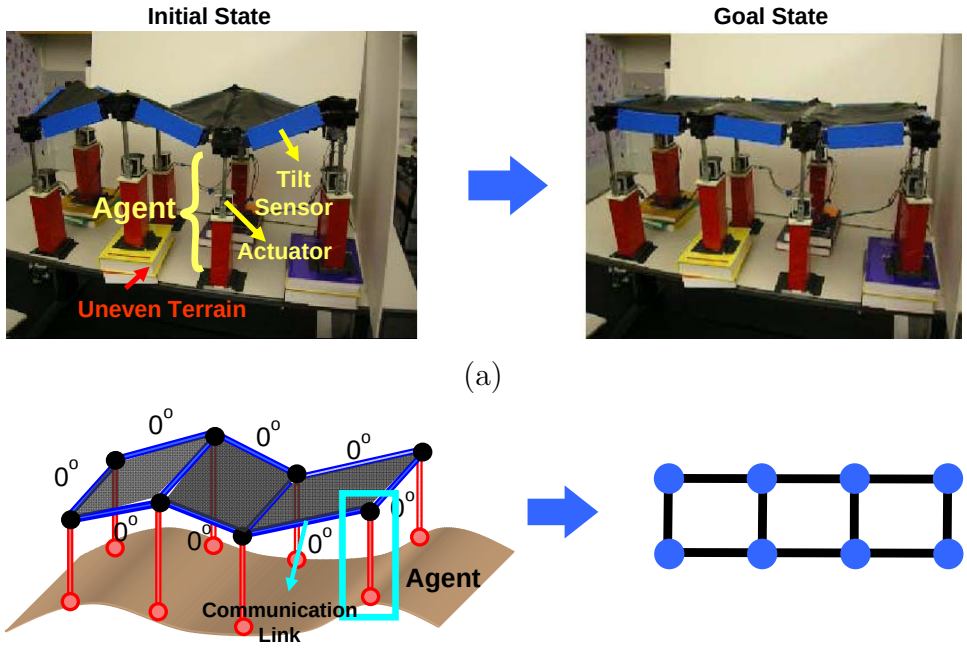
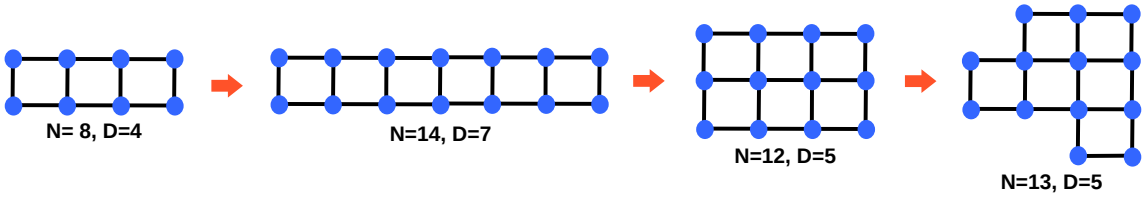


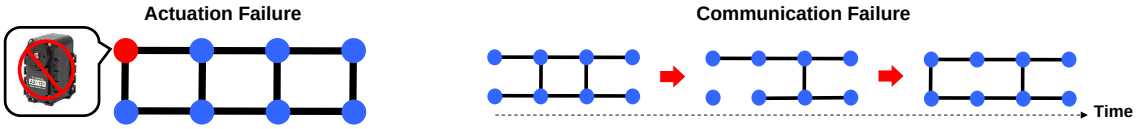
Figure 1.4: (a) A terrain-adaptive bridge that is formed by robotic modules, and there is a sensor mounted between two neighboring agents. The global task is to maintain the bridge surface level (right). (b) Skeletal view of the system (left) and the corresponding agent topology (right).

Scalability: One major advantage of a reconfigurable multiagent system is that it can transform to different configurations based on different scenarios, for example, elongating the bridge or widening the bridge by connecting with more modules (agents). Two key questions relevant here are whether the whole system will still produce correct results and how system performance changes based on different agent graphs (representative graphs are shown in the following figure). The theoretical analysis presented in this thesis provides a precise statement that the correct global behavior will emerge in *arbitrary connected* agent graphs. One can also determine how the number of time steps required to complete the task varies with the number of agents in the system (N) or the diameter of the agent graph (D).



Robustness: There are several potential failure modes in the system. For exam-

ple, some agents in the bridge robot might fail to actuate their motors during task execution and therefore might be unable to change their states (left diagram in the following figure); agents' communication links might also fail temporarily, causing the agent communication topology to change over time (right diagram in the following figure). In the face of such failures, a key question to ask is whether the global task can still be completed. If not, what state will the agents eventually be led to? Using theories developed in this thesis, one can characterize the robustness properties of this approach and estimate how the system will behave while experiencing different failure modes.



Reactivity: In a dynamic environment, agents can be constantly perturbed even when they have reached the desired state (shown as the following figure). For example, the underlying terrain of the bridge can potentially change, e.g., due to an earthquake. I prove that this approach allows agents to achieve quick adaptation and that it performs particularly well in tasks that require constant adaptation to a changing environment. Using this result, one can compare the tradeoffs and identify scenarios in which this decentralized method is advantageous.

What I have just described is a sample application to which this approach is suitable. In fact, there are many multiagent tasks that can be formulated with this framework, thereby acquiring the associated correctness, scalability, robustness, and reactivity properties described above. I now summarize how this approach can be applied to various examples of autonomous robot applications, some of which we have already demonstrated with real robots.

1.4 Application Areas

This framework has wide applicability to autonomous systems that exploit sensor-actuator networks as underlying architectures. By sensor-actuator networks, I mean networks that are composed of many independent and identical units that have sensing, actuation, computation, and local communication capabilities. In addition to those systems that are originally manufactured as sensor-actuator networks, many

systems can be turned into such networks by engineering each modular component with these capabilities. Here I illustrate several application areas to which I have either successfully applied this framework or within which I have implemented preliminary prototypes.

Modular Robot Self-adaptive Tasks: Modular robots are a class of robotic systems that can potentially assemble into a wide range of configurations and perform tasks in different environments. One main challenge in the field is to design a unified control strategy that allows robots to perform a variety of tasks with different configurations while adapting to dynamic environments. I apply this self-organizing control framework to two different types of modular robots, i.e., chain-style and struts-based, and show that various self-adaptive tasks can be solved with this approach. The experimental results show that robots of these three types can collectively and robustly achieve various tasks, including (1) forming environmentally-adaptive structures that can dynamically adapt to constantly changing terrains (Fig. 1.3 (a - b)), (2) achieving an adaptive gripper that can autonomously conform to and grasp fragile objects (Fig. 1.3 (c)), (3) performing locomotion and climbing tasks while autonomously adapting to the external environment (Fig. 1.3 (d)), and (4) performing locomotion while autonomously adjusting to different robot assemblies (Fig. 1.3 (e)).

Adaptive Architecture: One new direction in architecture design is the incorporation of sensing, computation, and motion capabilities in architecture such that it can adapt to environment changes, e.g., different daylight conditions or even earthquakes. Such architecture can also be viewed as containing an embedded sensor-actuator network. I apply this framework to an adaptive bridge hardware prototype (Fig. 1.3 (b)) and to a simulated adaptive building structure and show that these architectures can cope with sudden terrain change and minimize environmental impact.

Multi-Robot Collective Decision Making: Multi-robot teams have been shown to be advantageous in several types of tasks; for example, such teams can collectively explore a wider area of the environment than can a single robot. In these systems, a major challenge is how to achieve collective decision-making (for example, how they decide on a common traveling direction) by these spatially-distributed robots quickly and accurately. I extend this framework to address challenges in this direction, including methods of allowing robots to achieve fast decision propagation while facing emergency situations and letting some better-informed robots guide the overall decision-making process. I illustrate the applications of this extension by showing how it solves several challenging scenarios in a multi-robot simulated flocking task, such as tracking a fast-moving target. I also show that this approach can be generalized to solve tasks on wireless sensor networks, modular robots, and multi-robot teams.

Adaptive Medical Devices: In collaboration with medical and bioengineering researchers, we built preliminary adaptive knee and ankle orthotics for cerebral

palsy patients. In contrast to traditional orthotics that have fixed configurations, the embedded sensor-actuator network in our adaptive orthotic allows the orthotic to change its shape based on sensing of the subject's current state (Fig. 1.3 (f)). Our initial experimental results show that this control framework can potentially coordinate multiple sensors and actuators to achieve this goal.

1.5 Thesis Organization

The rest of this thesis is organized as following:

- **Chapter 2** summarizes work related to the subject of this thesis.
- **Chapter 3** describes the basic algorithmic framework of this work and the multiagent model.
- **Chapter 4** describes the theoretical results of this work, including correctness and how various factors, e.g., agents' communication topology, determine the performance of this approach.
- **Chapter 5** provides experimental results on various robotics applications in which I have used this approach, including self-adaptive modular robot structures and an adaptive modular gripper.
- **Chapter 6** shows how this framework can be extended to solve more dynamic tasks, such as adaptive locomotion and climbing tasks performed by a modular robot.
- **Chapter 7** describes the generalization of this framework to capture scenarios in which some agents obtain privileged information relevant to group tasks and to show several applications in the areas of multi-robots, sensor networks, and modular robots.
- **Chapter 8** summarizes contributions of the thesis. I then describe several potential applications in which this work can be further developed and some future promising directions.
- **Appendices A to E** provide proofs of theorems and details of robot hardware.

Chapter 2

Related Work

This thesis is related to several areas of multiagent applications and algorithms. Here I briefly summarize their relationships to this work; a detailed description is provided in Section 2.1, 2.2, and 2.3 of this chapter.

- **Modular and Swarm Robotics**: The main application area demonstrated in this thesis is the modular robot, a type of robotic system that is composed of many autonomous modules [55]. While various decentralized algorithms have been developed in the field of modular robotics, most approaches lack theoretical treatment; furthermore, most are specialized for particular tasks and difficult to generalize to other modular robot tasks or configurations. The main contribution of this thesis to the area of modular robotics is to propose a unified control framework that is applicable to a diverse set of modular robot tasks and provide theoretical guarantees of its correctness. I further show that an in-depth theoretical understanding allows us to identify the scope of this approach and to further generalize it to a broader application area. This framework also contributes to the field of swarm robotics by providing a simple yet effective way for a swarm robot team to solve consensus decision-making and flocking tasks in changing environments. A detailed literature review is provided in Section 2.1.

- **Cooperative Control in Multiagent Systems**: The approach presented in this framework is closely related to the field of distributed consensus (DC) [3, 48, 53] in cooperative control. This work contributes to this area by extending both the *task* and *application* scopes of DC. I further leverage results of spectral graph theory [10, 65] and stochastic matrix theory [23, 35] to analyze this class of algorithms such that one can predict the performance of multiagent systems in the face of different agent topologies, assigned tasks, and sensor-actuator networks. In Section 2.2, I present this thesis' contributions to the field of

cooperative control in detail.

- **Distributed Problem Solving**: The agent task formulation of this framework shares similarities with distributed constraint optimization (DCOP) [64, 85]. The main distinction is that DCOP uses discrete variables for agent states, while agent states in this framework are continuous variables that represent control parameters. In Section 2.3, I compare this framework with other distributed problem-solving techniques.

2.1 Modular and Swarm Robotics

A modular robot belongs to the class of robots that are composed of many independent modules [16, 55, 83, 84]. 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 [4, 17, 19, 45, 49, 61, 75, 82, 87]. The design of modular robots has been greatly influenced by multicellular systems in biology. Each module of a modular robot can be compared to a cell in a multicellular organism that can potentially adapt to different environments by changing either its internal structure or its external connectivity with other "cells". The ultimate goal of modular robotics is to allow such a system to perform a task adaptively according to its external environment, either by controlling actuator parameters or by changing the overall connectivity of the modules.

There are three main classes of modular and self-reconfigurable robots. The first type is the "chain-based" modular robot [45, 62, 58, 80, 81] in which individual modules can change their own shapes through individual actuation (e.g., changing internal angles) and/or reconfigure to change their connectivity with neighboring modules. Another common style is the "lattice-based" modular robot in which overall shape change is achieved only through changes in the local connectivity of modules [17, 19, 27, 32, 55, 56, 67]. One major limitation of lattice-based systems is that shape change can only be achieved through changing modules' connectivity, which is slow in hardware implementation. The third type is the "strut-based" modular robot [21, 39, 86, 87], which is composed of linear modular links. Using node-like interfacing connectors, one can potentially assemble these modules into a variety of geometries. The shape change of this type of robot generally relies on actuation of the linear modules.

Although the modular nature of these robotic systems makes it possible for them to sense and adapt to environments like multicellular organisms, the control strategies that allow robots to perform tasks adaptively based on environmental conditions are rarely addressed. Most groups have focused on static predefined goals that do not depend on sensed environmental conditions for shape transformation and loco-

motion [45, 55, 94]. This usually results in difficulties during operation of the robots in a dynamic environment. Only a few groups have addressed self-adaptive tasks using centralized or decentralized control [45, 55, 61, 63, 82]. In chain-based robots, Yim et al. demonstrated robot locomotion that conforms to the environment via a hand-designed gait table and distributed force feedback [82]. These authors showed that such an approach allows the robot to successfully negotiate different obstacle environments.

Another type of adaptive locomotion strategy for chain-based robots is based on the use of a Central Pattern Generator (CPG). Kamimura et al. and Sproewitz et al. demonstrated such an approach in the M-TRAN [28] and YaMoR [41] modular robots, respectively. Unfortunately, most of the strategies in this category require a tedious design process. In addition, there is no theoretical guarantee that the control laws will correctly lead all modules to achieve the desired task. Rus et al. demonstrated distributed algorithms for locomotion over obstacles [55], while Bojinov et al. 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 [5]. Although Bojinov et al. provide a framework for describing adaptive tasks, it is difficult to generalize their method to other systems and to theoretically prove its correctness. Furthermore, lattice-based systems have one major disadvantage in speed: shape change can only be achieved through changing modules' connectivity, which is slow in its hardware implementation. In this work, we mainly consider a chain-based system that can achieve fast adaptation.

Despite the fact that various decentralized algorithms described above have been developed for modular robots, these approaches usually focus on a particular type of task and robot configuration and are difficult to generalize to other tasks or other modular robotic systems. Furthermore, the theoretical properties of these algorithms are rarely studied, although some studies based on simplified 2D modular robot models have been carried out [52, 76]. General understanding of the strengths, weaknesses, and scopes of these approaches is thus rather limited. In this thesis, I present a thorough analytical framework of my proposed approach and show how an in-depth theoretical understanding allows us to assess its scope and further generalize it to a broader area of application [88, 89, 90, 92, 93].

In swarm robotic systems [6, 40], it is important for a team of spatially distributed robots to make consensus decisions [33, 50] and perform group tasks such as flocking [2, 15, 47, 72]. In changing environments, it can be tedious to establish coordination mechanisms by means of which agents can integrate unpredictably distributed information to accomplish specific tasks. In this thesis, I show that the proposed framework can be extended to solve these tasks in dynamic environments and that it provides a simple alternative to address this challenge [91].

2.2 Cooperative Control in Multiagent Systems

In the cooperative control community, there has been some success in analyzing biologically-inspired decentralized algorithms, for example, in flocking [18, 37, 47, 69, 72], multi-robot formation tasks [14, 15, 25, 48], and communication protocol [8, 30, 79]. In particular, the proposed control strategy is related to distributed consensus (DC) algorithms in multiagent systems [3, 48]. DC is considered to be an underlying principle in many biological group phenomena, from flocking of birds to synchronization of fireflies. It has been widely applied in distributed (robotic) systems, including autonomous vehicle formation control [15, 25, 47], sensor network coordination [13, 38, 79], and the sensor coverage problem [59].

This thesis contributes to the area of cooperative control in the following three ways: (1) In the algorithmic aspect, I 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]. I 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. (2) In the application aspect, I 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, I formulate a framework that allows one to view many sensor-actuator tasks as generalized distributed consensus (GDC) tasks. In addition to proving convergence of both DC and GDC algorithms, this thesis also contributes to analyzing how various factors, e.g., agent topology and task complexities, influence convergence speed, such that it can predict the performance of the algorithm for a given modular robot connectivity and programmed task.

The implicit leadership algorithm presented in Chapter 7 also relates to studies on networked agents with leaders. Some recent studies have shown how an entire group's behavior can be controlled by controlling specific agents designated as leaders, whether a single agent [25], multiple agents [26, 70], or even virtual leaders that do not correspond to actual agents [37]. Such studies generally assume that leaders are pre-designated and they all have the same goals. In a distributed multiagent system, agents that can sense important aspects of the environment might naturally provide guidance to the group without possessing an explicitly privileged role as leaders. Recent studies show that animal groups can achieve effective decentralized decision-making without explicit announcement of leaders or elaborate communication [11, 12, 68]. Inspired by these findings, I propose and analyze an implicit leadership framework that can capture such scenarios and further show how these mechanisms

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

can resolve conflicts between leaders and achieve fast decision-making.

2.3 Distributed Problem Solving

The task formulation in this framework shares similarities with distributed constrained optimization (DCOP) [64, 85] because both frameworks use inter-agent constraints as a way of expressing tasks. One major difference between two frameworks is that agent states of this framework are expressed as continuous variables, while, in DCOP, agents' states are discrete variables. Because agents' sensing and control states are usually continuous variables, this distinction makes the approach considered in this thesis more suitable than the DCOP framework for solving control tasks in robotic systems. On the other hand, DCOP can model multiagent decision-making tasks, e.g., scheduling tasks [64], well because agent states in these scenarios are usually expressed as discrete decision variables. Another important distinction between this framework and DCOP is that DCOP mainly focuses on computing solutions under static problem formulations. It is rarely addressed that how it can be extended to solve scenarios where it is necessary to cope with dynamic environments like tasks considered in this framework. Our approach extends distributed constrained task formulation to achieving tasks that require constant adaptation to current environments.

Particle Swarm Optimization (PSO) also shares certain similarities with my proposed agent control laws in terms of mathematical formulation [51, 74]. However, the overall framework presented in this thesis is quite distinct. PSO is an optimization technique in which each particle (agent) represents a solution to the objective function, while this framework addresses distributed multiagent decision-making tasks with each agent representing an independent decision maker.

Chapter 3

Multiagent Model and Self-Organizing Control Algorithms

In this chapter, I describe the basic self-organizing multiagent model. The system considered in this framework is composed of many *identical* and *autonomous* agents with computation, actuation, sensing, and communication capabilities. There are many multiagent applications that satisfy this model, such as modular robots and distributed robot systems. No global communication is required in this framework; instead, the global task is achieved by having each agent *communicate and coordinate with a small set of neighboring agents*.

Programming such systems can be challenging because agents are distributed in space and have only local information about the system, while the global task must be completed by coordination of all agents. One approach is to designate a leader agent to collect information from all agents, then compute and disseminate agent states to the whole group. However, such a strategy interferes with the system's scalability and robustness: it can be costly to deliver messages between the coordinator and all agents, and the coordinator becomes a potential point of failure for the system. A decentralized approach in which each agent interacts and communicates with only local neighbors has the potential to be more robust and scalable. In this approach, there is no leader agent that can potentially become a communication bottleneck or a failure point for the whole system. If an agent fails, its failure affects only a small area of the system and such a failure will not propagate to the whole system. Since no agent needs to coordinate with any agents other than its immediate neighbors, this approach is more scalable to the number of agents.

On the other hand, the decentralized approach also encounters a major challenge: how do we ensure that the desired global outcome will be achieved when all agents have only local views of the system? Here I also propose a task-specification scheme that describes global tasks as local inter-agent constraints. When each agent satisfies

its local constraints, the whole system achieves the desired task. This largely simplifies the complexity of solving collective tasks amongst agents and allows us to use simple local rules to solve global tasks in these systems. I will also show that a wide variety of tasks can be modeled using this task specification scheme.

In the second part of this chapter, I will present two agent control algorithms. In these two algorithms, each agent receives local sensing information from its neighbors and controls its state or actuator based on such information at every time step. The control laws *do not* depend on the history of agent actions or states. I will first describe the basic algorithm called the *distributed homeostasis* algorithm. I will then show how such a control framework can be extended to capture a wide range of sensing and actuation scenarios by presenting the *generalized distributed consensus* algorithm.

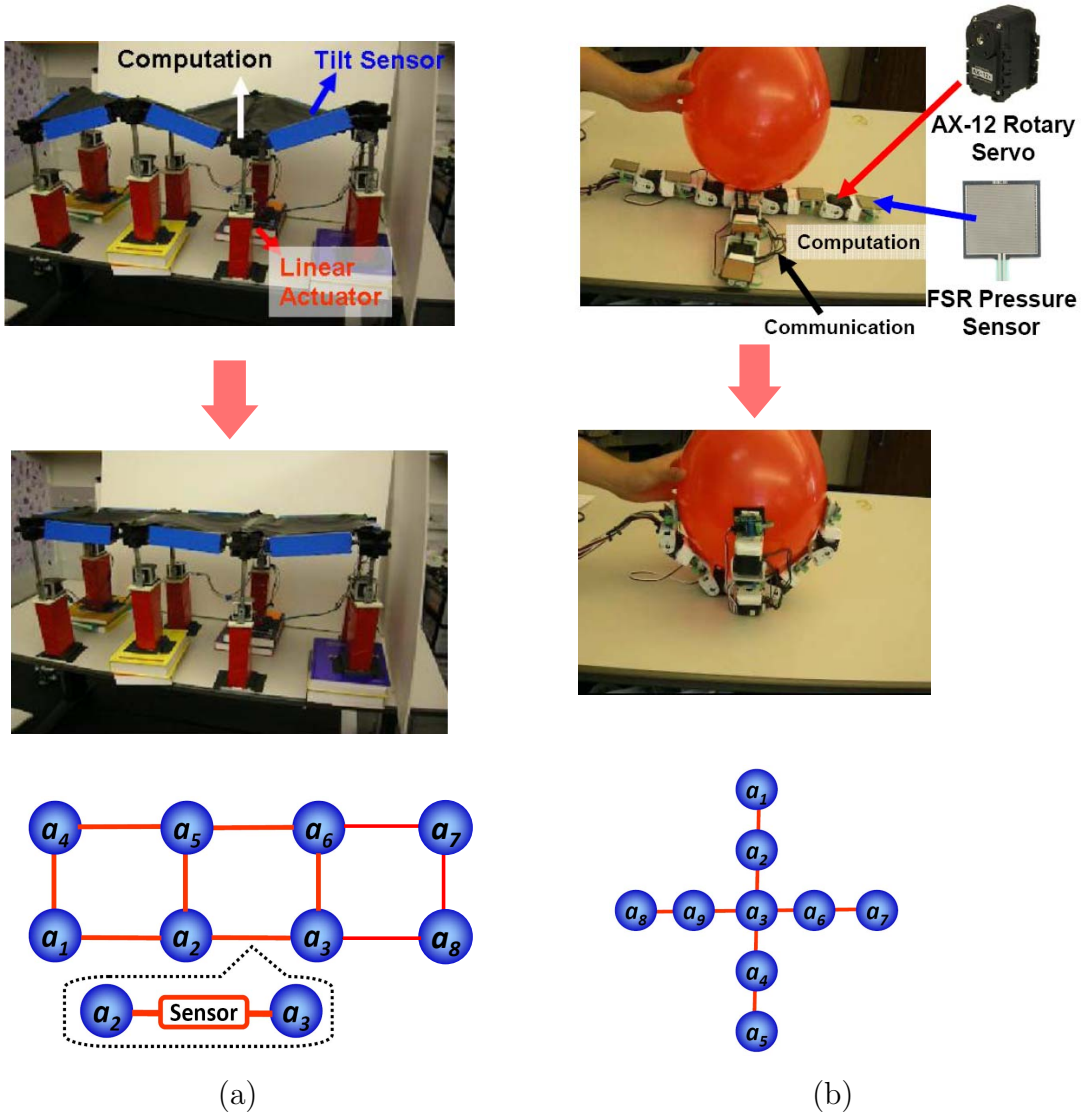
3.1 Multiagent Model and Task Specification

In this section, I describe the multiagent model and the capabilities assumed in its framework. The primary focus of this framework is on distributed agent systems in which the whole system is composed of many independent and autonomous agents. Throughout this chapter, I will use the following two example systems in modular robotics to illustrate the model and algorithms considered in this framework.

Figure 3.1 (a) shows a terrain-adaptive bridge that is composed of many modules. Each module is a pillar of the bridge and can vary its length via the module's actuation. There is a sensor mounted between each pair of neighboring agents. The desired global task is specified by a set of desired tilt sensor angles (constraints) between two neighboring modules. Each module can only observe its neighboring tilt sensor outputs and has multiple constraints to satisfy. When the robotic bridge is placed in an unknown environment (e.g., on a rough terrain), modules coordinate to change their heights in such a way that all desired constraints are satisfied and the bridge surface becomes level.

In the second example application, modules are connected to form a gripper (Fig. 3.1 (b)). The goal of modules is to collectively grasp an unknown object while maintaining equal force applied to the object by each module. Each module forms a segment of the gripper and can change the rotational angle of a joint by actuation. Each module is also equipped with a pressure sensor that can sense force it applies to the object. Since modules are physically connected, they can potentially affect each other's states. Modules coordinate to form a grasping posture, with each module applying force to the object equal to that applied by its neighbors.

In the following subsections, I describe the multiagent model considered in this framework using the above applications as examples.



(a) (b)

Figure 3.1: (a) 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 (top) and is programmed to achieve a flat surface (middle). The graph representation of the system is shown as the bottom diagram. (b) Modular gripper. Each segment of the gripper is a module with rotary servo, pressure sensor, computation unit, and communication channel. The gripper is programmed to grasp around the object with each segment applying equal pressure to the object. The system can be more abstractly represented as a graph (bottom).

3.1.1 Agent

In this framework, an agent is defined as an independent unit with the following capabilities:

- **Sensor**: Each agent is equipped with one or multiple sensors of the same type suited to different robotics applications. Sensors are used to measure the current state of the agent or the relationship between a neighboring pair of agents. In most applications, each agent is mounted with a single sensor to measure its current state. In some applications, for example, the terrain bridge in Fig. 3.1 (a), sensors can also be shared between two neighboring agents.
- **Actuator**: Each agent is equipped with an actuator or several redundant actuators that can provide the same function. I consider both linear (Figure 3.1 (a)) and rotary actuators (Figure 3.1 (b)) in this framework. In some applications, for example, sensor networks, the "actuator" can be viewed as the capability of changing agents' internal states, such as the timer in a sensor node. In this framework, I consider each agent has only one (or one type of) actuator(s), eliminating the possibility that each agent can potentially choose among multiple actuators.
- **Computation**: Each agent is capable of performing simple computations such as addition and multiplication.
- **Communication**: In this framework, agents are only required to communicate with their one-hop nearest neighbors. In a connected system, such as a modular robot, each agent is also able to communicate with immediate neighbors that are physically connected to it. If agents are not connected, each agent is required to be capable of communicating with neighbors that are within its communication radius. Here I assume that every agent can communicate with its neighbors at every discrete time step, i.e., round synchronization. In Chapter 4, I will present the scenarios that occur when agents cannot achieve round synchronization.

Most of the current distributed robotic systems have the stated capabilities. In modular robotics, these examples include M-TRAN [45], Odin [39], SUPERBOT [61], and various swarm robotic systems [40].

3.1.2 Graph Representation of a Networked Multiagent System

In this 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* multiagent 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 Fig. 3.1 (bottom), edges correspond to the communication links between two neighboring agents in the bridge of Fig. 3.1 (a). The neighbor relationship between agents is symmetric; therefore, the edges in graph G are undirected (Fig. 3.1).

3.1.3 Self-Adaptive Tasks as Distributed Constraint Maintenance

In this framework, the multiagent 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 multiagent self-adaptive tasks can be formulated under this task specification. For example, in the terrain-adaptive bridge example (Fig. 3.1 (a)), 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 diagram of Fig. 3.2 (a). 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 (Fig. 3.2 (b), see more details in Chapter 5).

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 finger tips). We can also specify a modular surface robot to form more a complex shape by specifying the desired inter-agent constraints (in Fig. 3.2 (c)). Even in cases where agents are not equipped with actuation capabilities, we can specify the task in terms of desired state relationships between agents. For example, in a sensor network time synchronization task (Fig. 3.2 (d)), we can simply program each agent to maintain the same timer as its neighbors (agents that are within its communication radius).

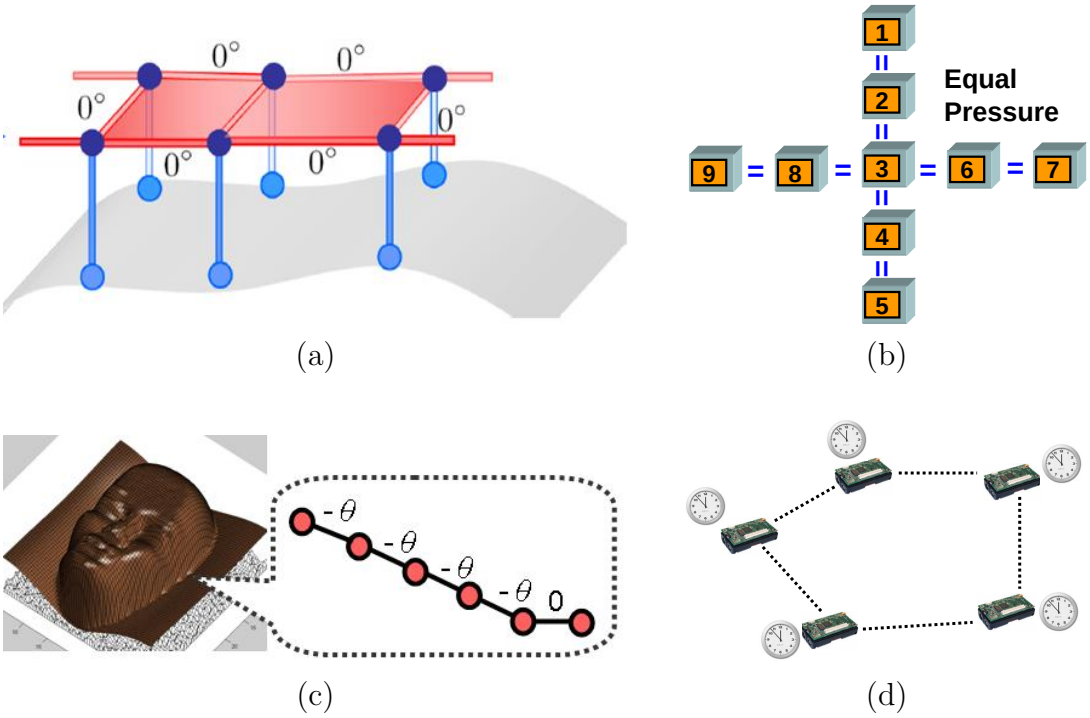


Figure 3.2: A diverse set of tasks can be specified with inter-agent constraints. The global task is achieved when all agents satisfy their local constraints. (a) Terrain-adaptive bridge. Each module is programmed to maintain zero tilt angles to its physically-connected neighbors. (b) A Modular gripper. Modules in the gripper is program to maintain equal pressure as its neighbors. (c) Module-formed 3D Relief Display. If agents are specified with appropriate constraints, they can collectively achieve more complex tasks, e.g. forming more complex shapes. (d) Sensor network time synchronization. Each agent is programmed to achieve the same timer as its neighbors.

3.2 Control Algorithm

In this section, I present two control algorithms for solving self-adaptive tasks. The first algorithm, the distributed homeostasis algorithm, is able to solve complex self-adaptive networked multiagent systems where there is a known function relating each agent's sensor and actuator states. In this case, the problem is closely related to distributed consensus in control theory [48].

The second algorithm, called generalized distributed consensus (GDC), generalizes the distributed homeostasis algorithm to address systems in which the exact relationship between each agent's sensor and actuator states is unknown. Taking the modular gripper as an example, there is no direct function that converts each agent's

rotational angle to the pressure sensor state. I derive three conditions for deriving the agent control law for such scenarios. As long as the derived control laws satisfy these conditions, one can still solve complex self-adaptive tasks. This generalized framework allows us to extend the control law to a wide range of applications, which I will cover in Chapter 5. This extension also broadens the application scope of distributed consensus-type algorithms.

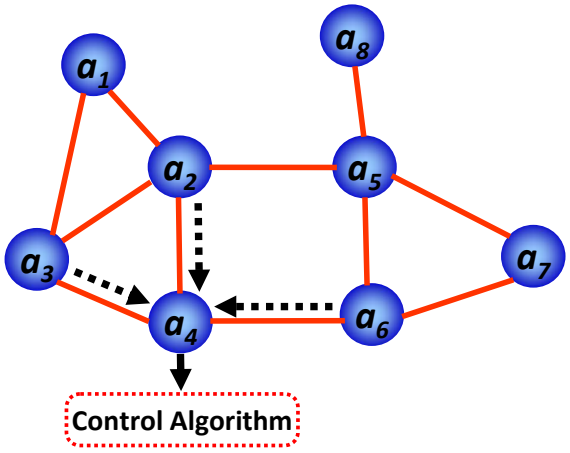


Figure 3.3: Each agent's control algorithm takes all neighboring sensor readings as inputs to compute the agent's new state.

3.2.1 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 is illustrated in Fig. 3.3 and can be formally written as:

ALGORITHM 1: **Distributed Homeostasis**

$$x_i(t+1) \quad = \quad x_i(t) + \alpha \sum_{a_j \in N_i} \left(x_j(t) - x_i(t) - \Delta^*_{ij} \right) \tag{3.1}$$

where a_i indicates agent i, and $x_i(t)$ and $x_i(t+1)$ are actuation states[1] of agent i at time step t and $t+1$, respectively. N_i indicates the set of all one-hop neighbors

[1]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.

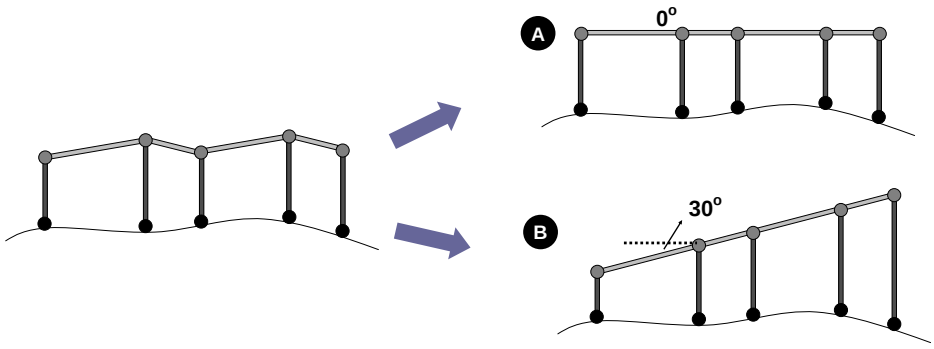


Figure 3.4: In the terrain-adaptive bridge task, one can specify different tasks with different desired inter-agent relationships Δ^*_{ij}. If $\Delta^*_{ij} = 0$ for all i and $j \in N_i$, modules form a level bridge surface over rough terrain. If $\Delta^*_{ij} = \pm 30°$ for all i and $j \in N_i$, modules form an incline slope.

of a_i. α is a small constant, and is sometimes called the damping factor. The inter-agent constraint variable Δ^*_{ij} is the desired state difference between agents a_i and a_j. As I described previously in Section 3.1, the inter-agent constraints, i.e. $\Delta^*_{ij}, \forall i$ and $j \in N_i$, are specified when we assign the task to the robot.

If $\Delta^*_{ij} = 0, \forall i$ and $j \in N_i$, agents are programmed to achieve the same state; this process is called *distributed consensus* (DC). The mathematical formulation is shown as the following Eq. 3.2:

$$\begin{aligned} x_i(t+1) &= x_i(t) + \alpha \sum_{a_j \in N_i} (x_j(t) - x_i(t)) \\ &= (1 - \alpha|N_i|) \cdot x_i(t) + \alpha \sum_{a_j \in N_i} x_j(t) \end{aligned} \tag{3.2}$$

On the other hand, when $\Delta^*_{ij} \neq 0$, this is sometimes also called *biased* consensus. Fig. 3.4 shows that one can use different Δ^*_{ij} specifications to achieve different shape formation tasks with the terrain-adaptive bridge. When $\Delta^*_{ij} = 0, \forall i$ and $j \in N_i$, the bridge forms a level surface. When $\Delta^*_{ij} = 30°$ if agent j is the right neighbor of agent i (and vice versa for the left neighbor), the bridge forms a walkway with 30° incline.

If an agent's sensory and actuation states are different, the distributed homeostasis control law requires it to convert sensory information from its neighbors to actuation state differences between it and its neighbors $(x_j(t) - x_i(t))$. For example, as shown in Fig. 3.5, if the distance between an agent a_i and its neighbor a_j , d_{ij}, is known, one can easily calculate the state (height) difference from the tilt sensor reading $\theta_{i,j}(t)$: $x_j(t) - x_i(t) = d_{ij} \tan(\theta_{ij}(t))$.

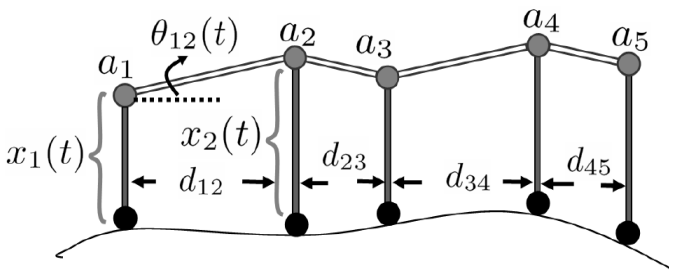


Figure 3.5: In some scenarios, the inter-agent sensor readings can be mapped to inter-agent state differences. For example, when the distance between each pair of neighboring legs is known (d_{ij}), one can calculate $x_j(t) - x_i(t) = d_{ij} \tan(\theta_{ij}(t))$.

There are three main assumptions associated with the distributed homeostasis algorithm:

1. Each agent is capable of directly observing its state and its neighbors' states or can map sensor readings to agent states.
2. Each agent is capable of freely driving itself to a new state $x_i(t+1)$.
3. The state variable $x_i(t)$ is assumed to be scalar and is continuous.

In the following, I will relax some of these assumptions and present a generalized algorithm within which this framework can be applied to a wider variety of tasks.

3.2.2 Generalized Distributed Consensus (GDC) Algorithm

In many cases, the relationship between sensor space and an agent's actuation state is not precisely known. For example, in the modular gripper example in Fig. 3.1 (b), each module is equipped with a rotary motor and a pressure sensor. The mapping between the actuator's actuating angle and the agent pressure sensor value cannot be directly computed. Similarly, in the terrain-adaptive bridge example, if the distances between legs are unknown, the function relating sensor readings and actuator states cannot be exactly expressed.

Therefore, I 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 (GDC)**

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} \left(g(\theta_i, \theta_j) - \theta^*_{ij} \right) \tag{3.3}$$

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 function of the sensory readings that capture relationship between agents a_i and a_j , and $\theta^*_{ij}, \forall i$ and $j \in N_i$ are task-specific inter-agent sensory constraints, similar to Δ^*_{ij} in Eq. 3.1. Note that Eq. 3.3 assumes agents perform update with round synchronization, i.e. each agent updates according sensory feedback at every discrete time step. As I will prove in the next chapter, there are three important requirements for designing $g(\theta_i, \theta_j)$ to ensure that all agents will correctly complete the desired task

$$\textbf{Condition 1} \;:\; g(\theta_i, \theta_j) - \theta^*_{ij} = 0 \Leftrightarrow \theta_j - \theta_i = \theta^*_{ij} \tag{3.4}$$

$$\textbf{Condition 2} \;:\; sign(x_j(t) - x_i(t) - \Delta^*_{ij}) = sign(g(\theta_i, \theta_j) - \theta^*_{ij}) \tag{3.5}$$

$$\textbf{Condition 3} \;:\; g(-\theta_i, -\theta_j) = -g(\theta_i, \theta_j) \tag{3.6}$$

Intuitively, condition 1 (Eq. 3.4) means that g only "thinks" the system is solved when it actually is; condition 2 (Eq. 3.5) 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 (Eq. 3.6) means that g is *anti-symmetric*.

Eq. 3.3 allows us to formulate many different tasks in sensor-actuator networks. 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., $\theta^*_{ij} = 0, \forall i \& j \in N_i$, can be viewed as reaching a consensus state: $\theta_i = \bar{\theta}, \forall i$.

In order to utilize the GDC algorithm to solve different tasks, we must address the following challenge: we need to appropriately design the function g such that each agents will correctly drive its state $x_i(t)$ towards satisfying all θ^*_{ij}. In fact, Eq. 3.4 - 3.6 represent three simple principles that can guide us to efficiently design the g function. In practice, it is most challenging to design control feedback to satisfy condition 2 (Eq. 3.5) since it is usually non-trivial to discover relationships between agent's sensing and actuation components. In the modular gripper example, it is not straightforward to design g such that the rotational angle state $x_i(t)$ is driven towards correct direction in the next time step ($x_i(t+1)$) such that $g(\theta_i, \theta_j) - \theta^*_{ij}$ is minimize for all agents a_i and a_j pairs ($a_j \in N_i$). For more details, please see Section 5.3 of Chapter 5. I will show several examples on how one can utilize these conditions to design controllers for various sensor-actuator network applications in Chapter 5.

Distributed consensus-type control has been widely applied in solving many multiagent cooperative control tasks, such as vehicle team formation and sensor network

time synchronization. However, its task space is for the most part constrained to achieving the same state amongst agents ($\Delta_{ij}^* = 0$). The distributed homeostasis algorithm allows the same type of control to apply to scenarios in which there are some desired relationships ($\Delta_{ij}^* \neq 0$) to be achieved between agents. Such tasks are sometimes called *biased consensus* [47]. Furthermore, the generalized distributed consensus algorithm enlarges the application area that this type of control by extending to a wide variety of systems in which each agent's sensor-actuator relationship is not precisely known.

3.3 Discussion

To utilize this framework to solve multiagent tasks, one needs to decompose the global tasks into state or sensory constraints between neighboring agents. In practice, I find that global tasks are easiest to decompose in this way if the tasks require agents to achieve *the same states*, for example to achieve and maintain the same sensory readings. In such tasks, one can simply specify that each agent should maintain the same states or same sensory readings as its neighbors. For tasks that require more complex inter-agent state or sensory constraints to achieve the global tasks, some prior knowledge is usually required about what local constraints will lead to desired global behaviors. We will see more examples in Chapters 5 and 6.

Note that there are potentially multiple solutions that satisfy all local constraints. For example, in tasks that require agents to achieve the same states as their neighbors, agents can complete such tasks with any state values as long as they achieve the same states as their neighbors.

This task specification scheme also shares similarities with distributed constrained optimization (DCOP) [64, 85] One major difference between the two frameworks is that agent states in this framework are expressed as continuous variables, while, in DCOP, agent states are discrete variables. Because agents' sensing and control states are usually continuous variables, this distinction makes the approach considered in this thesis more suitable than the DCOP framework for solving control tasks in robotic systems. Another important distinction between this task specification scheme and that of DCOP is that DCOP tasks mainly focus on computing solutions under static problem formulations. It is rarely addressed how DCOP could be extended to solve scenarios where it is necessary to cope with dynamic environments, as with several tasks considered in this thesis. My approach naturally extends to solve such scenarios by formulating constraints as continuous objective functions. When agents are perturbed from the goal states by dynamic conditions, they autonomously adjust their states toward satisfying the constraints again.

3.4 Summary

In this chapter, I have described the agent assumptions in this framework, a task specification scheme for self-adaptive tasks, and two agent control laws. The rest of this thesis is tightly related to this chapter. Specifically,

- Chapter 4 presents the convergence properties of both the distributed homeostasis algorithm and the generalized distributed consensus algorithm. I will also analyze how the agent graph G affects the speed of convergence. These properties relate to the correctness and scalability of the algorithm.
- Chapter 5 shows how one can design the g function and apply this framework to many different multiagent tasks. I also demonstrate several real-world applications with hardware prototypes and their implementation details.
- Chapter 6 shows how to expand the task space of this framework by generalizing it to solve a dynamic task that requires achieving a sequence of self-adaptations.
- Chapter 7 presents methods for extending the control algorithms such that this framework can capture scenarios in which agents have different levels of importance in solving the global tasks.

Chapter 4

Theoretical Analysis of the Self-Organizing Control Algorithms

In this chapter, I present a theoretical analysis of the multiagent algorithms presented in the previous chapter, including their correctness, scalability, robustness, and reactivity. I show that this class of problems relates strongly to a class of problems in decentralized control theory called *distributed consensus*, which includes several canonical bio-inspired algorithms such as flocking and firefly synchronization. By exploiting this connection, several important aspects of the algorithms can be thoroughly analyzed, and one can extend the algorithmic approach to new types of sensor-actuator agents and arbitrary multiagent network topologies.

The main contributions of this analysis are as follows: I first show that interactions between agents can be formulated as collective linear dynamical systems, and that one can analyze various aspects of the algorithms by studying this system. With this formulation, I prove that the algorithms converge to the correct user-described goal for any undirected connected graph of agents. This allows us to guarantee that the desired task will be correctly completed under arbitrary connected agent topologies and any initial agent condition.

I also derive the convergence rate and show how different factors of the agent network topology, e.g., number of agents and network diameter, determine the convergence time. This allows us to reason analytically about the scalability of the decentralized approach. I also quantify the impact of initial state and goal state and show that the convergence time depends logarithmically on the distance between the two. This allows us to bound the worst-case number of iterations and compute how quickly the system reacts to small perturbations. Via this analysis, we can understand how this approach scales to the number of agents, generalizes to different tasks and adapts to perturbations. I also prove how the algorithms behave in the face of agent actuation and communication failures. Based on these results, we can estimate

how the system will behave while encountering different failures. Finally, I use these results to show that decentralized algorithms have an advantage over tree-based centralized algorithms when it is important that the network adapt quickly to constant environment changes.

4.1 Correctness of the Algorithms

In the control law I presented in the previous chapter, 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 I show that (1) the dynamics of all agents can be modeled as a single collective dynamic system, and (2) based on this formulation, correctness of the algorithms can be characterized for arbitrary connected graphs G and desired goals in both distributed homeostasis and generalized DC algorithms.

4.1.1 Proofs for Distributed Homeostasis Algorithm

I first demonstrate how to aggregate all agent control laws into *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, I prove the convergence property for this dynamical system.

Collective Dynamics: We first consider the case for standard distributed homeostasis algorithm (Eq. 3.1). Let $X(t)$ represent the ensemble of all agents' states at time t:

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

We now recap Eq. 3.1:

$$x_i(t+1) \;=\; x_i(t) + \alpha \sum_{a_j \in N_i} \left(x_j(t) - x_i(t) - \Delta_{ij}^*\right) \tag{4.1}$$

We can aggregate all agents' update equations and write the collective dynamics of all agents as:

$$X(t+1) = A \cdot X(t) + \tilde{b} \tag{4.2}$$

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*. Note that A is a stochastic matrix since each row sums to 1 [60]. In addition, $\sum_i \tilde{b}_i = 0$, since $\Delta^*_{ij} = -\Delta^*_{ji}$ for all i, j.

Eq. 4.2 can be rewritten as

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

where $A = I - \alpha \cdot L$ and 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 thoroughly analyzed. Olfati-Saber et al. ([48]) 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:[1] The key result here is that the presence of a nonzero bias vector $\tilde{b}$ does not affect the eigenvalue analysis, so that the results of Olfati-Saber et al [48] apply here as well.

The sketch of the proof is as follows: I first show that agents' distance to the desired goal depends on the second largest eigenvalue of A. I then show that eigenvalues of A and L matrices are related. Since L encodes the agent topology, we can ensure that the second largest eigenvalue of A falls between 0 and 1 when the agent topology is connected. Specifically, let λ_i denote the i^{th} *smallest* eigenvalue of the graph Laplacian L and let μ_i be the i^{th} *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 [42], when the coordination graph G is *connected*, the *second* smallest eigenvalue of L, λ_2, is strictly

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

larger than 0. Thus, the second largest eigenvalue μ_2 of A is strictly smaller than 1 and greater than or equal to zero.

Let X^* denote our desired goal state; we can prove (shown in Appendix A.1) that under the iteration of Eq. 4.1, agents' distance from the current state $X(t)$ to the desired state X^* satisfies the following inequality:

$$\|X(t) - X^*\|^2 \leq \mu_2^{2t} \|X(0) - X^*\|^2 \tag{4.3}$$

Intuitively, since $0 \leq \mu_2 < 1$, agents are ensured to get closer to the desired state X^* at each iteration. This result leads to the following theorem:

Theorem 1 (Convergence, Distributed Homeostasis Algorithm)**.** *Let X^* be the desired shape state that $x_j^* - x_i^* = \Delta_{ij}^* \;\; \forall a_i$ and $a_j \in N_i$ and 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.1. □

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

From Eq. 4.3 the convergence proof, we can further express how agents' distance to the desired state changes over time as the following inequality:

$$\|Y(t+1)\| \leq \mu_2 \|Y(t)\| \tag{4.4}$$

where $Y(t) = \|X(t) - X^*\|$ represents the current distance from the desired goal. From Inequality 4.4, we can see that the value of μ_2 determines how quickly agents approach the goal state at each time step (when it is closer to 0, the whole system converges faster). In Section 4.2, I will further analyze how various other factors, e.g., the agent topology, determine μ_2's value.

4.1.2 Proofs for Generalized Distributed Consensus Algorithm

Now let us recapt the case of the generalized distributed consensus algorithm in Eq. 3.3:

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} \left(g(\theta_i, \theta_j) - \theta_{ij}^* \right) \tag{4.5}$$

Let

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

one can then reexpress Eq. 4.6 as

$$x_i(t+1) = x_i(t) + \sum_{a_j \in N_i} \phi_{ij}(t)(x_j(t) - x_i(t) - \Delta^*_{ij}). \tag{4.6}$$

Intuitively, comparing Eq. 4.6 with Eq. 4.1, 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. The relationship between the step length and the actual agent state differences may be *non-linear*. Following the same procedure in Section 4.1.1, we can rewrite the system dynamics as

$$X(t+1) = A(t) \cdot X(t) - \tilde{b}(t) \tag{4.7}$$

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

$$\mathbf{a}_{ij}(t) = \begin{cases} \phi_{ij}(t) & \text{if } a_j \in N_i \text{ and } i \neq j \\ 1 - \sum_{a_j \in N_i} \phi_{ij}(t) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

and $\tilde{b}_i(t) = \sum_{a_j \in N_i} \phi_{ij}(t) \Delta^*_{ij}$. 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 I used in the original case required the generating matrix A to be row stochastic and symmetric with positive diagonal elements. That is, (i) $A_{ij} \geq 0$ for $i \neq j$; (ii) $A_{ii} > 0$ 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 I outline in Chapter 3:

$$g(\theta_i, \theta_j) - \theta^*_{ij} = 0 \Leftrightarrow \theta_j - \theta_i = \theta^*_{ij} \tag{4.9}$$

$$sign(x_j(t) - x_i(t) - \Delta^*_{ij}) = sign(g(\theta_i, \theta_j) - \theta^*_{ij}) \tag{4.10}$$

$$g(-\theta_i, -\theta_j) = -g(\theta_i, \theta_j) \tag{4.11}$$

$A(t)$ is guaranteed to be stochastic and symmetric for all t. To see why, notice that for *off-diagonal* elements, $A_{i \neq j}(t) = \phi_{i \neq j}(t)$ for all a_i and $a_j \in N_i$. But because of the condition on g in Eq. 4.10, 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, notice that $A_{ii}(t) = 1 - \sum_j \phi_{ij}(t) > 0$ for all i. But this can always be

ensured as long as α is chosen small enough. Finally, since $g(-x,-y) = -g(x,y)$ (Eq. 4.11), we have $\phi_{ij}(t) = \phi_{ji}(t)$, therefore the symmetry condition is ensured.[2]

The guarantee of stochasticity and symmetry allows us to prove:

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 Eq. 4.7 will ensure that $X(t)$ converges to X^* for all initial conditions with convergence rate at least:*

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

Proof. see Appendix A.2. □

Intuitively, we understand from this result that the analogue of the original result holds, except that the guarantee becomes somewhat looser, as we must now 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.

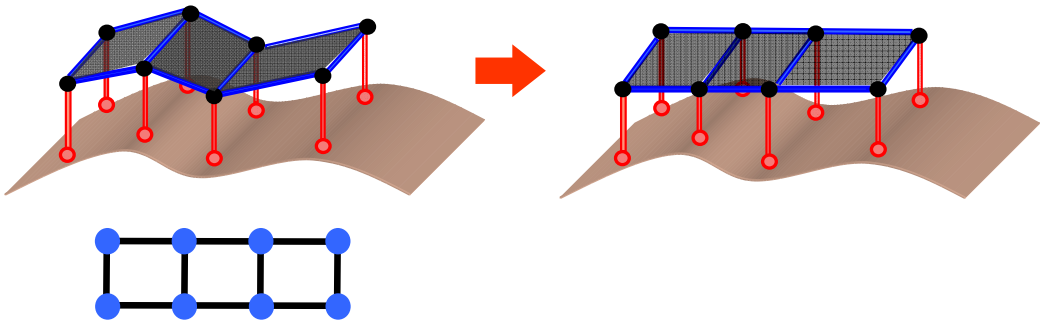
4.1.3 A Sample Task

I use the terrain-adaptive modular bridge described in the previous chapter as a simple illustrative example to show how one can use the correctness results presented here to understand the multiagent system.

The following figure shows a skeletal view of the bridge and the agent connectivity topology. Each agent is a pillar of the bridge and its state $x_i(t)$ represents its current height. We program all agents to maintain the same height, so the desired state difference $\Delta_{ij}^* = 0$ for all i and $j \in N_i$ (biased vector $\vec{b} = \vec{0}$). Intuitively, the distributed homeostasis (DH) algorithm and Theorem 1 tell us that if each agent reacts to local state disagreements, the system will converge to a solution as long as the agent topology remains connected. In fact, the solution that agents converge to while executing DH would be the average height of the agents.

If agents cannot compute local state disagreements and can only access tilt sensors, one can potentially program them to execute the generalized distributed consensus (GDC) algorithm. Theorem 2 intuitively tells us that as long as the three conditions

[2]Note that the assumption that $A(t)$ is symmetric (Eq. 4.11) can be relaxed, but in that case the upper bound on the convergence rate is less tight. The proof of this case is based on the theory of nonhomogenous stochastic matrix products [60]; the product $A(t)\cdots A(2)A(1)$ will converge to a rank 1 matrix at an exponential rate. The recent result in [9] explicitly determines an upper bound on the convergence rate.



outlined in Eq. 3.4 to Eq. 3.6 are satisfied in agent control law, the system will still converge to the desired state X^*.

For a reconfigurable multiagent system, one can potentially change agents' connectivities based on different tasks. One would potentially want to program them to achieve different tasks (for example, program the bridge to form a steady inclining slope). During task execution, agents might fail to communicate with their neighbors or fail to actuate themselves. In the following section, I analyze how these various factors affect the performance of agents in completing the desired task.

4.2 Factors Affecting Convergence Speed

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

1. **Scalability and impact of multiagent topology**: How does the time to complete the desired task increase as one increase the number of agents and the diameter of the networked multiagent system?

2. **Task complexity**: How does the complexity of the task affect the time to achieve the task?

3. **Agent Failures**: How does the multiagent system perform when some agents encounter actuation and/or communication failures?

4. **Reactivity**: How do the agents react to perturbations from the desired state?

In this section, I provide precise answers to these questions, such that one can predict agents' performance given agent topology and their assigned task.

4.2.1 Formula for Convergence (Task) Time

Let us discuss how various factors affect agents' convergence (task) time. I specifically note that task time discussed here is based on *the number of iterations required and is not the absolute time* [3]. In developing a formula for convergence (task) time, 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, I mean:

Definition: The task achieved by the agents at state $X(t)$ is an $\epsilon-$*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 and that some level of inaccuracy must therefore be tolerated. From Theorem 1 and 2, we know that the agents approach the desired state X^* at an exponential rate. I 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^*:

$$\begin{aligned} \|X(t^*) - X^*\| &\leq \mu^{t_{\max}} \|X(0) - X^*\| \leq \epsilon \\ \Rightarrow t_{\max} &\leq \log_{\mu_2} \left(\frac{\epsilon}{\|X(0) - X^*\|} \right) \end{aligned} \tag{4.12}$$

We can see from Inequality 4.12 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).
- 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.

4.2.2 Scalability and Topology

Assuming that the initial distance from the desired goal is fixed, the number of iterations depends on μ_2. I showed in Section 4.1.1 that $\mu_2 = 1 - \alpha\lambda_2$, where

[3] If agents' speed of executing one iteration of update is faster, the overall *absolute time* to achieve the desired task is also proportionally increased.

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 I show that there are several important bounds on λ_2 that will provide us with a concrete understanding of how the algorithm scales as one scales up the size of the multiagent system.

Theorem 3 (Mohar, 1991 [42])**.** *Let L be the graph Laplacian of G. Then the second eigenvalue λ_2 of L satisfies*

$$\lambda_2 \geq \frac{4}{N \cdot D}$$

where $N = |G|$ and $D = diam(G)$ 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.

$$\mu_2 \leq \mu_2^+ = 1 - \frac{4\alpha}{N \cdot D} \tag{4.13}$$

For example, assume that the multiagent 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 goes down by a factor of 4. Hence, the runtime of the algorithm is worst-case bounded by $O(N^2)$ for a line.

However, in reality, the convergence rate can be much superior to this upper bound. Fig. 4.1 shows the case when the multiagent topology is an n by n grid where n is varied from 2 to 15. I fix $Y(0) = 50\epsilon$ in order to examine only the effect of topology. The upper curve shows t_{bnd}, the estimated t_{max} of Ineq. 4.13 with μ_2^+, the middle curve shows test, the estimated t_{max} with real μ_2, and t_{sim} is the average number of iterations in simulation computed from 1000 different initializations of $X(0)$ for each topology. We can see that although the worst case t_{max} is quite high, the convergence time in reality can be much faster.

Note that there are many forms of upper and lower bounds on λ_2 that can provide tighter estimates of the convergence time. Here I use one of the simpler bounds that is easy to reason from; in Appendix A.7, I provide additional examples of known bounds based on different graph topological properties. At the same time, one can use the same analytical procedure to obtain an understanding of the convergence time for a specific multiagent system.

The analytical results presented here allow us to predict a networked multiagent system's convergence time based on its topology. Take the terrain-adaptive bridge as an example: when we change its modules' connectivity based on different environmental conditions, e.g., in situations in which elongation or widening of the bridge is

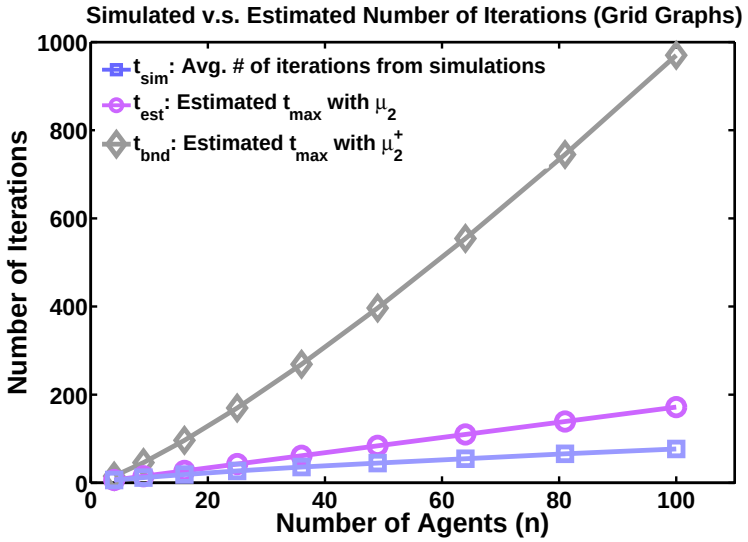
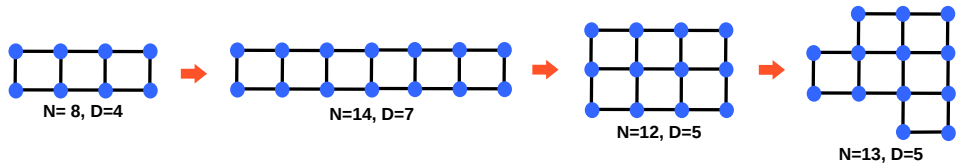
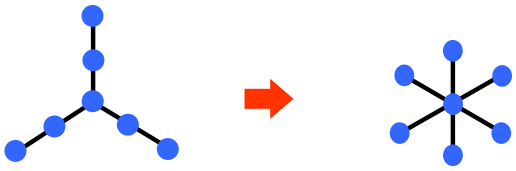


Figure 4.1: Number of modules vs (1) average number of iterations in simulations (t_{sim}), (2) estimated bounds using μ_2 and μ_2^+ on number of iterations (t_{est} and t_{bnd}). Simulation suggests that although the worst case t_{bnd} is quite high, the convergence time in reality can be much faster.

required (shown as the following diagram (a)), we can then estimate its performance based on its topological variable changes (e.g., diameter (D) and number of agents (N)). Using these results, one can also reconfigure the existing multiagent system to a configuration that would be expected to yield better performance. For example, one can change the topology of a three-finger gripper to a six-finger configuration to decrease its topology diameter and increase its connectivity (shown as the following diagram (b)).



(a)



(b)

4.2.3 Task Complexity

Another important question is how the desired task impacts the speed of convergence. From Inequality 4.12, if the agent topology G is fixed, 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, for a given multiagent system with topology G, 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 $x_i(t) \in R^+ \; \forall i, t$ 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 $\epsilon-$approximation of the desired goal is at most:*

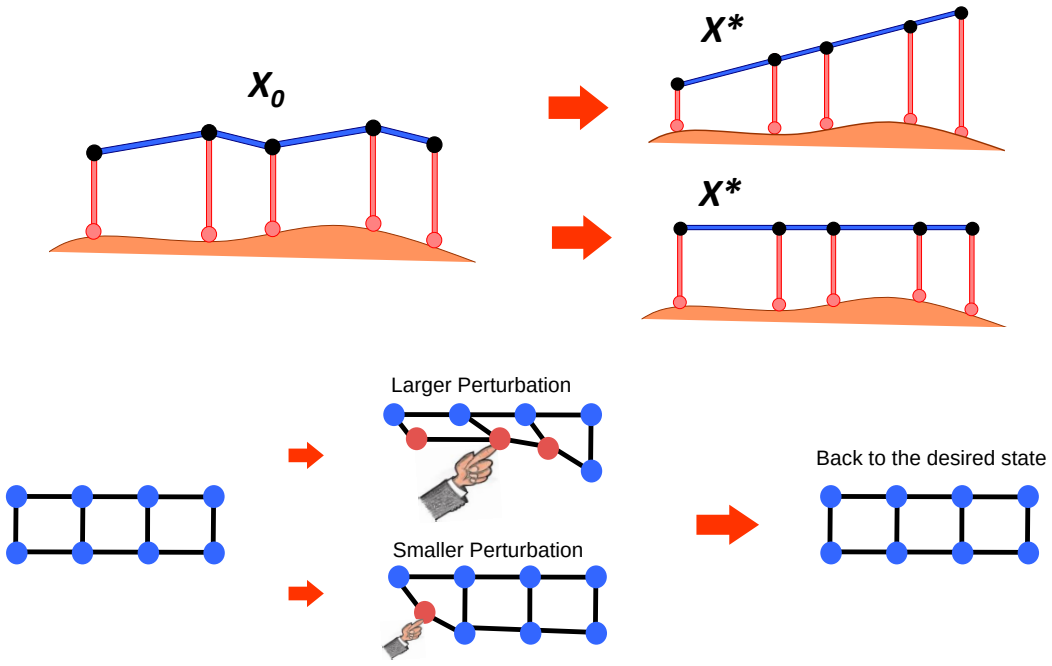
$$t_{\max} = \lceil \log_{\mu_2} \left(\frac{\epsilon}{\sqrt{2}\mathcal{C}} \right) \rceil$$

Proof. see Appendix A.3. □

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

Using the results presented here, one can estimate agents' convergence time to the desired goal. For example, if we program the terrain-adaptive bridge to form different shapes (e.g., level surface or steady incline, shown as the following diagram), we can use the distance between the initial state and the desired state X^* (the summation of "degrees" by which agents violate the inter-agent constraints) to compute the expected iteration to achieve the task. If we know the initial states of the agents, we can also compute the maximal number of iterations required to form any shape based on Theorem 4.

After agents have achieved the desired state, we can also calculate the time required to converge back to the desired state when the system is perturbed by an external force (shown as the following figure). From Inequality 4.12, we also know that the convergence time is logarithmically proportional to the degree of perturbation.



4.2.4 Robustness to Agent Failures

In this section, I examine the correctness of the algorithm while having one or more agents fail in the process. The first type of failure we will see here is actuation failure. In this case, some agents are unable to drive themselves to the desired states by controlling their equipped actuators. They thus remain in the state they are in when the failures initially occur. I show that if there is one agent with an actuation failure, the complete system will still converge and achieve the desired task X^*. I 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 the other agents execute either the distributed homeostasis or the GDC algorithm. The agents will eventually achieve the desired state X^*.*

Proof. see Appendix A.4. □

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 in which there are multiple agent failures.

Theorem 6 (Multiple Agents Actuation Failures)**.** *Let* $\mathbf{F}$ *be the set of all failed agents and* $|\mathbf{F}| > 1$*. Let all the other agents executes either the distributed homeostasis or*

the 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 A.5. □

The above two theorems tell us that when the networked multiagent 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 the manner in which multicellular systems increase their robustness with redundant cells. An agent will only undergo complete actuation failure when all actuators fail.

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. Note that if agents cannot execute control algorithm and update their states at the same rate (asynchronous update), we can also consider them as agents with temporarily failures. The agent communication graph is thus time-varying, denoted as $G(t)$. Let us 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 A.6. □

To prove Theorem 7, I first show that the multiplication of agents' generating matrices over a finite time period t to $t + \Delta t$ is a stochastic matrix and that this is true for any starting time point t. Then, as in the proof of Theorem 2, I can use the convergence property of stochastic matrices to prove that all agents will converge to the desired state.

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. Empirically, this control approach is also robust toward zero mean sensory noises since such noises can possibly be decreased when each agent aggregates and add all sensory information from all neighbors. One interesting direction for future analysis is to obtain a theoretical guarantee on this approach's robustness under different sensory noises.

4.2.5 Reactivity

Another important criterion is how the networked multiagent system reacts to perturbations. From Inequality 4.12, 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 4.2 shows the system's reactivity for square grid topologies ($1 \times 1, 2 \times 2, \cdots, 15 \times 15$). Red lines indicate larger perturbations ($Y(0) = 250\epsilon, 500\epsilon, 1000\epsilon$), and blue lines indicate smaller perturbations ($Y(0) = 2\epsilon, 5\epsilon, 7\epsilon$, and 10ϵ). Each point on Fig. 4.2 represents the mean number of simulation rounds required of 1000 random initial $X(0)$ that satisfy the given $Y(0)$. I further examine the case with random connected topologies that have the same number of edges and nodes as the corresponding grid graphs but in which the connections between vertices are randomly generated[4]. For Fig. 4.3, each point represents the mean from 20 random topologies and 500 different initializations. We can see from the figures that the networked multiagent system's reactivities toward small perturbations (blue lines) scale well with the number of agents in both regular and irregular topologies. 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 described in the next chapter.

4.3 Self-Organizing vs. Centralized Approach

An important question in networked multiagent systems is whether to use a decentralized self-organizing approach, such as the one 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 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 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 multiagent systems, each agent has fixed communication and computation power. For the kinds of tasks we consider here, com-

[4]I remove the graph with ≥ 2 connected components and restart the process until the generated graph is connected.

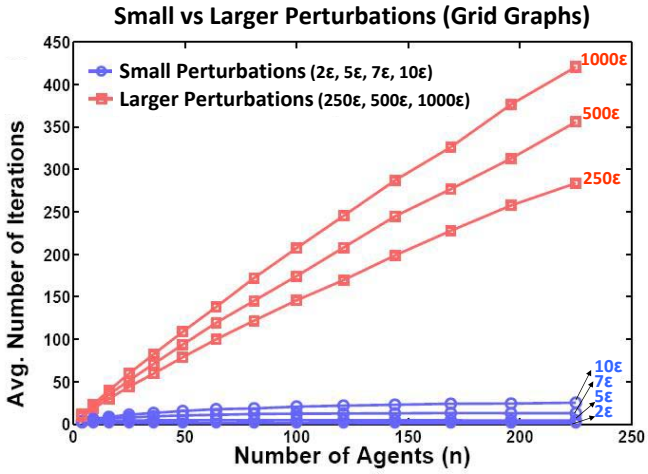


Figure 4.2: Number of iterations required to complete the task versus number of agents, large and small perturbations with self-organizing (decentralized) approach. In our experiment, agents are connected to form a grid graph. I increase the number of agents of the grid graph from $1 \times 1, 2 \times 2, \cdots, 15 \times 15$ (X axis). I apply small perturbations $(2\epsilon, 5\epsilon, 7\epsilon, 10\epsilon)$ and large perturbations $(250\epsilon, 500\epsilon, 1000\epsilon)$ in each graph configuration. We can see that, in small perturbation cases, the number of iteration required stays low even when I increase the number of agents in the graph to several hundreds.

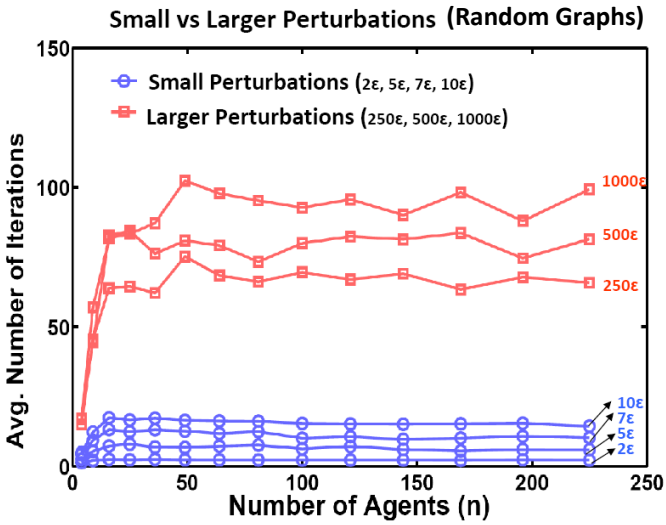


Figure 4.3: Number of iterations required to complete the task versus number of agents in randomly-generated topologies.

munication 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. Fig. 4.2 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 self-adaptive tasks, 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. Finally, they are more robust and less complex to implement in situations where agent errors and topology changes are common.

4.4 Summary

I have presented and analyzed both distributed homeostasis and generalized distributed consensus algorithms. I prove the convergence properties of such algorithms and characterize how various factors, such as the topology of the agent cooperation graph and initial states of the agents, are related to their performance. I further prove robustness of this approach and show how agents behave in the face of actuation and communication failures. In comparison with a centralized approach, this approach has strong advantages in robustness and reactivity and is well-suited for tasks that require constant adaptation to changing environments. In Chapter 5, I will demonstrate how to apply this approach to various tasks that fall within this category.

Chapter 5

Self-Adaptive Multiagent Applications

In the previous chapter, I introduced distributed homeostasis (DH) and generalized distributed consensus (GDC) algorithms and showed how to analyze the correctness, robustness, and scalability of this decentralized approach. Here, I show how this algorithmic framework can be applied to a wide variety of modular robot self-adaptive tasks and hardware.

In this chapter, I provide control algorithms and experimental results for these tasks. The detailed design of the modular robot systems is provided in Appendix C. More specifically, I demonstrate the following: (1) a module-formed table and bridge that can quickly adapt to external perturbation while maintaining the table surface and bridge surface level; (2) a modular pressure-adaptive column that is capable of maintaining "pressure-consensus" by reconfiguring itself to absorb uniform pressure, with this application directed toward the goal of adaptive building structure; and (3) a modular gripper that is capable of grasping a fragile object while each module applies identical force on the balloon, allowing computation of the grasping posture to be distributed amongst the modules that form the gripper.

I first present how to apply the proposed algorithm in solving each of the above modular robot tasks. In these tasks, agents are equipped with pressure (force feedback) sensors and tasks are described as satisfying inter-agent pressure sensory relationships. To design agent control laws in these tasks, it it easy to establish conditions 1 (Eq. 3.4) and 3 (Eq. 3.6) with GDC control law. However, it is challenging to design feedback functions such that agents drive their actuation states toward the correct direction to satisfy inter-agent sensory relationships across all time steps (condition 2, Eq. 3.5), since there is no straightforward mapping between agent pressure sensory states and and actuation states.

With various real robot experiments, I evaluate the approach's performance in several different aspects, including the following: (1) its capacity to adapt to external perturbations as well as internal faults; (2) how different initial conditions and different robot configurations would affect the time required to complete the desired tasks; and (3) its scalability to the number of modules. In Section 5.5, I describe several self-adaptive tasks in simulations that are implemented with this approach, including a 3D relief display that can transform to various shapes and an adaptive building structure that can adapt to different terrain conditions. Finally, I also provide a discussion about several possible application extensions and limitations of this framework in Section 5.6.

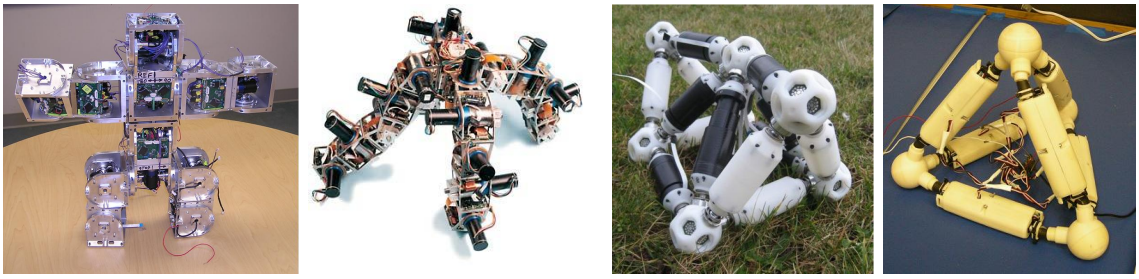
The robot hardware experiments and simulations presented in this chapter show the following:

- The DH and GDC algorithms allow us to systematically and easily program agents to achieve complex and correct global behaviors.
- This self-organizing approach is inherently error-correcting and able to achieve global tasks robustly even when agents are equipped with cheap and noisy sensors and actuators.
- A diverse set of multiagent applications can be abstractly viewed as the same task and tackled with this framework.
- The theoretical results presented in the previous chapter coincide with our empirical results, thereby validating that one can use theories developed here to estimate real-world system performance.

This empirical study also contributes to providing a deeper understanding of the limitations and strengths of decentralized approaches and how one can potentially combine such approaches with a centralized planner.

5.1 Modular Robot Systems

The main application area I demonstrate in this chapter is the modular robot – a class of robots that are composed of many interconnected and autonomous modules [16, 55, 83, 84]. In a modular robot system, each module is usually equipped with its own independent computation, sensing, communication, and actuation capabilities and can thus be viewed as an independent agent. Each module can usually also send messages to other modules that are physically connected to it. Modular robots have three main advantages over traditional robots: (i) They are capable of changing their configurations to become different structures or shapes based on different tasks. (ii) In



(a) (b) (c) (d)

Figure 5.1: Two types of modular robots considered in this thesis. (a – b) **Chain-style modular robots**, SUPERBOT (a) and Polybot (b). Each square unit is an independent module, and there is a rotary motor mounted on each module. (c – d) **Strut-based modular robots**, Odin (c) and our own designed robot (d). Each link/node module is an independent module; each link module can perform linear actuation to elongate or contract its length.

most modular robots, there are distributed sensors in the whole systems because each module is equipped with a sensor. Such sensor-rich robots can potentially perform tasks more adaptively and reactively in changing environments than other robots with limited sensors. (iii) Because all modules are identical, the whole system is more robust with regard to module failures as long as we program each module in the system with an identical controller.

Although point (i) has been widely explored in various reconfigurable modular robots, points (ii) and (iii) are rarely addressed in the research community. If (ii) and (iii) can be achieved, one can potentially program modular robots to perform tasks or form structures while adapting to dynamic environment efficiently and being fault-tolerant. In this chapter, I program modular robots with this self-organizing framework and show how such a strategy allows the robots to achieve goals (ii) and (iii).

While the main contributions of this thesis are algorithmic and theoretical aspects, I also implement this approach on several modular robot hardware prototypes. Our systems are inspired by the following two styles of modular robots: chain-style modular robots (a – b) and strut-based modular robots (c – d). Fig. 5.1 (a) shows a chain-style modular robot, SUPERBOT [58], and Fig. 5.1 (b) shows another chain-style modular robot, Polybot. In this type of system, each module is equipped with a rotary motor. The whole structure can deform via the modules' collective actuation or changing of connectivity. The modules we use in self-balancing table and adaptive gripper (described later in this chapter) fall in the category of chain-style modules.

Fig. 5.1 (c) and (d) show Odin strut-based modular robot system and our own designed strut-based system. Such systems are based on a node/link design in which

each link module interfaces with other links by connecting to node modules. Each link module is equipped with linear actuator(s), and the whole structure can deform via the link modules' actuation. In terrain-adaptive bridge and pressure-adaptive column, we use strut-based modules to construct these structures.

5.2 Self-Balancing Table and Terrain-Adaptive Bridge

In this section, I describe two self-adaptive structures formed by modular robots: a terrain-adaptive bridge (shown in Fig. 5.3 (a)) and a self-balancing table (Fig. 5.3 (b)). In both applications, the goals are to form a level surface over bumpy terrain and to maintain levelness over time even if the terrain changes. If we can program modules to form such a "living" structure, we can potentially place such a structure on arbitrary terrain to form a flat walkway while adapting to an external environmental impact (e.g., an earthquake). Additionally, modules can potentially form a self-balancing table that maintains levelness on a boat or airplane.

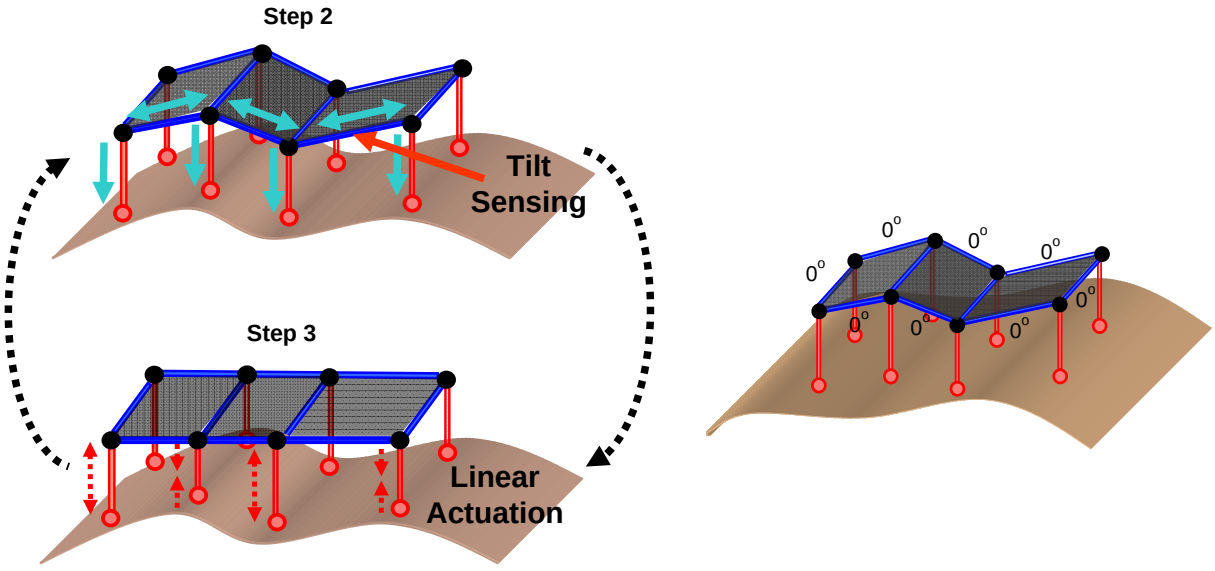
Modular Structure and Initialization: In both structures, each leg of the structure is formed from one or more modules and serves as an independent agent that can change its height by compressing and extending its actuator(s) (as shown in Fig. 5.2). In the terrain-adaptive bridge structure, we use a single link module as a leg of the bridge (Fig. 5.3 (a)). In the self-balancing table, multiple chain-style modules are connected together to form a leg (Fig. 5.3 (b)). There is a tilt sensor mounted in between each pair of legs to measure neighboring agents' tilt relationships [1].

Task Specification : The global task of the bridge is to maintain its surface level at all times, irrespective of underlying uneven terrain. Using constraint-maintenance task specification, it can be specified for each agent to maintain zero-tilt angles with respect to all of its neighbors (shown as Fig. 5.2 (b)).

Algorithmic Procedure: Each agent is programmed to execute the generalized distributed consensus algorithm by iteratively computing its height based on neighboring tilt sensory feedback. More specifically, our algorithmic procedure can be divided into the following three steps:

- **Step 1 (Initialization)**: Agents start sending messages to their neighbors. Based on these messages, agents identify their local neighbors and sensors.
- **Step 2 (Sensing):** At each iteration, each agents start by accessing sensor readings between themselves and all of their neighbors (shown as Fig. 5.2 (a)).

[1]Each tilt sensor also captures the relative height relationship between two neighboring agents



(a) (b)

Figure 5.2: Concept diagram of self-adaptive structure. Each module (leg) of the structure is an independent agent. Modules iteratively sense environment and perform linear actuation until a level surface is achieved.

(a) (b)

Figure 5.3: (a) Terrain-adaptive bridge prototype. Each pillar is a linear module and can be viewed as an independent agent. (b) Self-balancing table hardware prototype. Each leg is an independent agent and composed of three chain-style modules. The compression (or elongation) of a leg is achieved by individual modules acting in an "accordion-like" fashion by simultaneous rotating their motors.

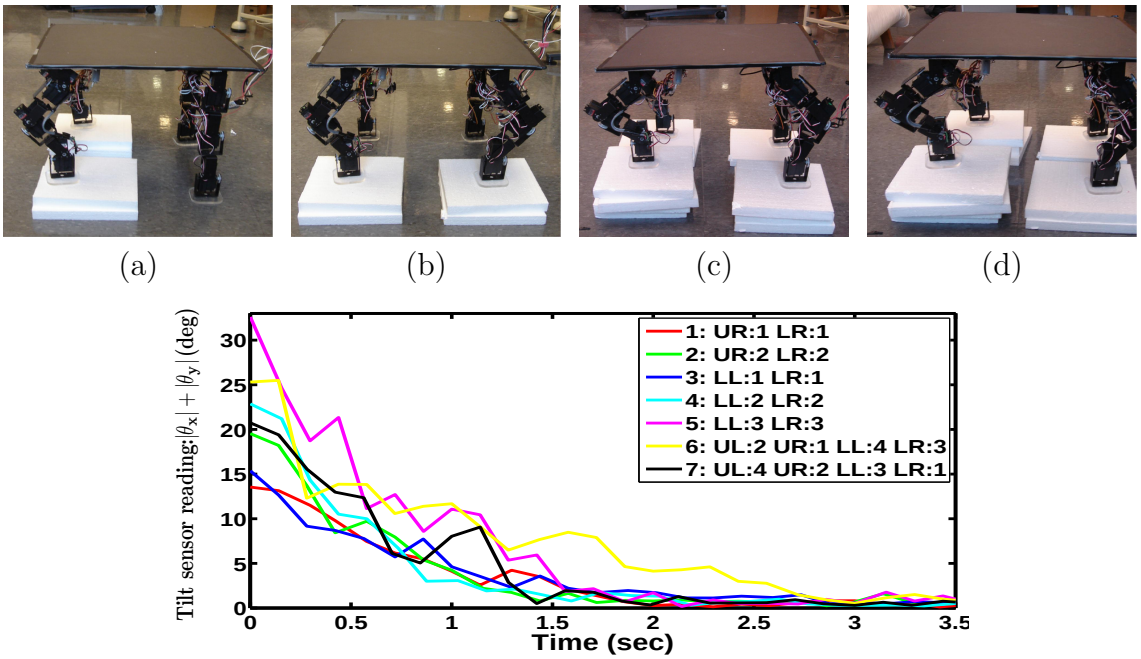


Figure 5.4: (a - d) The robot is placed on obstacles of different height. It adapts to different obstacles and forms a flat surface. (Bottom) The robot's response time to different initial conditions. In seven out of eight experiments, it achieves $|\theta_x| + |\theta_y| \leq 3^\circ$, within 2 seconds (30 iterations).

- **Step 3 (Actuation):** After receiving this sensory information, each agent uses the data as input from which to compute its actuation length. Each agent's state at t, $x_i(t)$, is used to denote the *height of the leg*. The Agents' control law is derived from the conditions described in Eq. 3.4 – 3.6 and can be formally written as:

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} \theta_{ij} \tag{5.1}$$

- **Step 4 (Iteration):** This process reiterates until all agents have achieved zero tilt angles with respect to all of their neighbors.

Now, the generalized feedback function $g(\cdot)$ in Eq. 5.1 is simply θ_{ij}, the tilt angle of the surface between agents a_i and a_j. It is in fact easy to show that $g(\cdot)$ satisfies Eq. 3.4 – 3.6: (1) $\theta_{ij} = 0$ only if the inter-agent tilt sensor constraint is satisfied. (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. In the following, I describe experimental results for both the self-balancing table and the terrain-adaptive bridge.

5.2.1 Self-Balancing Table Experiments

The self-balancing table follows the abstract model of the self-adaptive structure in Fig. 5.2 (a). Each leg (agent) is composed of three chain-style modules that can collectively change the leg height via actuation, and four legs are attached to a rigid table surface. Modules in each leg are connected together, and each pair of neighboring modules' motors has opposite rotational directions. The compression (or elongation) of a leg is achieved by individual modules acting in an "accordion-like" fashion by simultaneously actuating their motors. We mounted a two-axis (x and y) tilt sensor (Analogue Devices ADXL311 accelerometer) on the table surface. Each agent can receive data from this sensor instead of having their own tilt sensors (because the table surface is rigid). All agents execute the control law in Eq. 5.1. Here, I describe several experiments that test the effectiveness of the algorithm with a self-balancing table:

Different Initial Conditions: In the first experiment, we examine how the robot responds to different rough terrains. As shown in Fig. 5.4 (a - d), the robot's four legs were placed on four obstacles of different heights. Each foot position was adjusted with several bricks (a brick's thickness is 2.5 cm). Through different combinations of bricks, different rough terrains can be generated. The robot's upper left supporting group position is denoted as UL, lower left is LL, upper right is UR, and lower right is LR. The number following each denotes how many bricks were placed at that position.

Agents are programmed to maintain a level surface (i.e., the tilt angles in the x axis and y axis, θ_x and θ_y, are 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 5.4's bottom diagram shows the robot's response time to achieving levelness under different terrains. In the first five experiments, two legs on one side are lifted. In the last two experiments, the robot's four legs are lifted to simulate fully irregular terrain scenarios. As shown in the figure, the robot is capable of converging to levelness ($|\theta_x| + |\theta_y| \leq 3\,^\circ$) within 2 seconds ($\sim$ 40 iterations) in most of the cases. Experiment 6 is the only case in which the robot requires > 2 seconds to achieve levelness because its setup requires all agents to collaborate with their neighbors rigorously.

Environment Response Experiments: In the second experiment, we examined how quickly and accurately the self-balancing table responds to consistent, rapid environmental changes. In this 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 (Fig. 5.5 (a - d)). 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 five seconds of Fig. 5.5 (bottom)).

Figure 5.5's bottom diagram shows the results of the experiment. We can see that

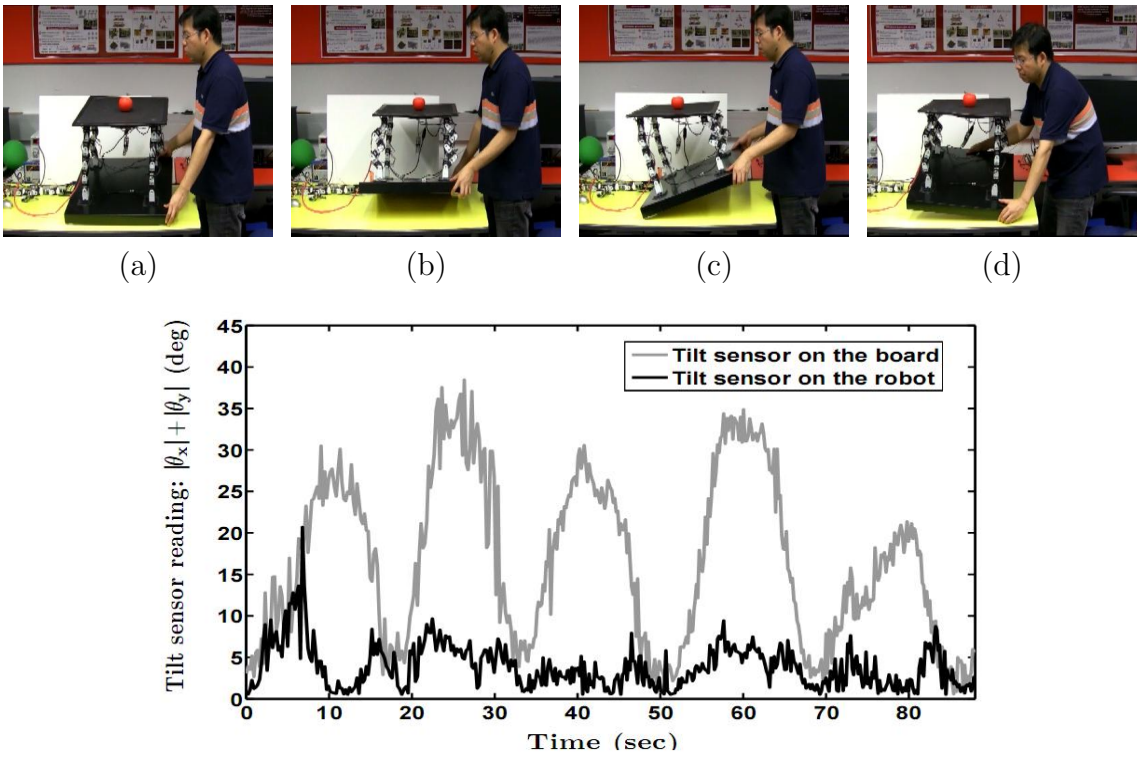


Figure 5.5: (a - d) The robot is fixed to a board. The robot keeps its surface level when we repeatedly change the orientation of the board. (Bottom) The self-balancing table response time to repeated environment changes. The robot was able to maintain its surface's tilt angle within 5 degree $\sim$ 8 degree (the average noise level of the sensor is 3 degrees) even when the board is tilted 30 degree $\sim$ 40 degree over a few seconds.

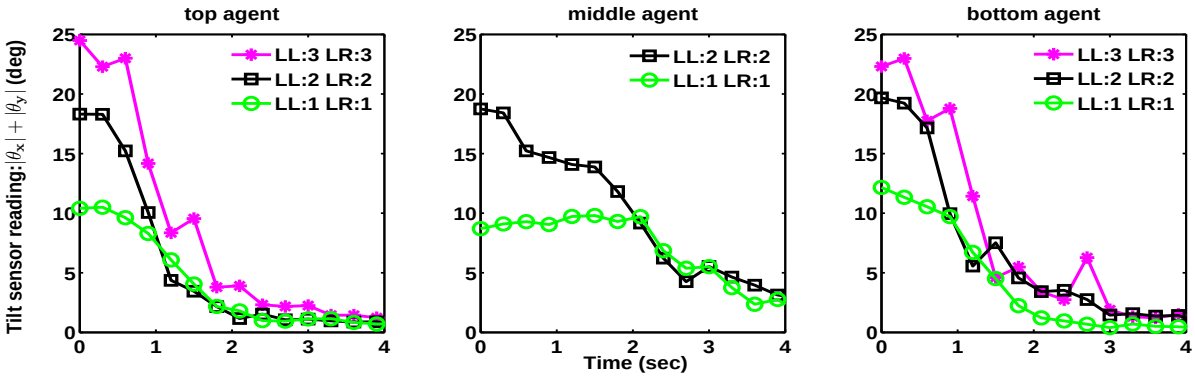


Figure 5.6: Robustness test results when one of the agents does not respond. The left, middle, and right figures are scenarios when top, middle, and bottom modules do not respond respectively. It is most critical when the middle module fails.

the table is able to respond quickly and keep the surface level even when the tilt angle of the floor is changed by 30 degrees to 40 degrees over a few seconds. Even though the average sensor noise level is 3 degrees, the table never tilts more than 5 degrees to 8 degrees.

Robustness Experiments: We also perform robustness experiments by observing the robot's reaction when one of the modules in a leg (agent) fails. In this case, this agent has a lower range of actuation than its neighbors. We tested it under different task difficulties and in different positions in the group. Two situations in which an agent fails to respond are examined: (1) the module's motor is disabled and becomes a passive link, so it freely takes on any angle with no resistance to movement; (2) the module's motor remains stuck at the zero-degree position at all times. We discovered that the first case does not affect the effectiveness of the algorithm, whereas the second case affects a few scenarios. This result implicitly demonstrates that the algorithm is robust to hardware failure in the first case. Here, we only discuss the second case.

Figure 5.6 shows the robot's responding time to achieve levelness when different modules do not participate in the task. One side of the robot is lifted to a height of one, two, and three bricks. At each height, one of the three modules (top, middle, or bottom) in the legs that need to be compressed is disabled. This process is repeated four times, and Fig. 5.6 shows the average of the robot's tilt angle across time. We can see from Fig. 5.6 that the middle module's failure is more critical than is that of either the top or bottom modules. When the middle module fails, the robot generally falls over in the third experiment (three bricks high). When the obstacle is one or two bricks high, it achieves $< 4^\circ$ of tilt angle in approximately four seconds.

We also observe that this approach allows modules in the same leg as the failed module to compensate for the failed module without explicit communication. This is primarily because modules in an agent are always acting proportionally to the "error" of satisfying, constraints. Having one failed module in the group would naturally make this error larger and thus other normal modules compress or elongate their heights more. Our results also empirically show that (1) this approach is robust toward partial agent failures and (2) increasing the redundancy of actuators also allows the system to be more robust.

5.2.2 Self-Adaptive Bridge Experiments

We also examine the terrain-adaptive bridge's adaptiveness to terrains of different roughness levels. The modules were assembled in the same bridge-like configuration as Fig. 5.3 (a). 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

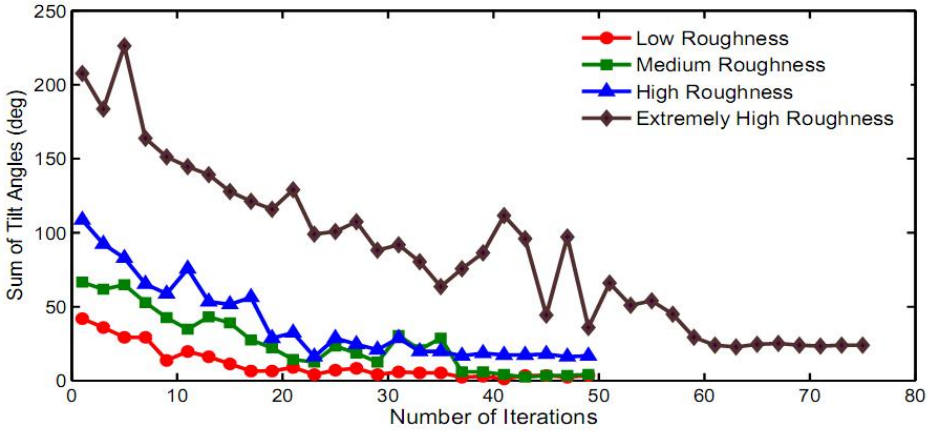


Figure 5.7: The bridge structure's response time to achieve its surface's levelness. The average initial sensor tilt angles for low, medium, high, and extremely high roughness are 5, 8 , 10, and 20 degrees respectively (the maximal tilt angel each link can achieve is ~ 25 degrees. In low, medium, and high roughness cases, it is capable of achieving the average of passive links' tilt angles ≤ 3 degree after 40 iterations. In extremely high roughness case, it achieves the same level of levelness after 60 iterations.

increase the roughness of the terrain by adding more underlying bricks. Because 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 the levelness measure, shown as the following equation:

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

Figure 5.7 shows the number of iterations required to achieve levelness versus different levels of roughness of the underlying terrain. Because the tilt sensors used in experiments are somewhat noisy, we define the bridge as having achieved levelness when each passive link's tilt angle is on average smaller than 3 degrees. We can see from the Fig. 5.7 that the bridge surface is capable of achieving levelness after running the control algorithm for 40 iterations in low, medium, and high roughness cases (the x-axis represents the number of iterations, and the y-axis is the summation of the tilt angles from all of the inter-agent tilt sensors). In the extremely high roughness case, the table achieves levelness after approximately 60 iterations.

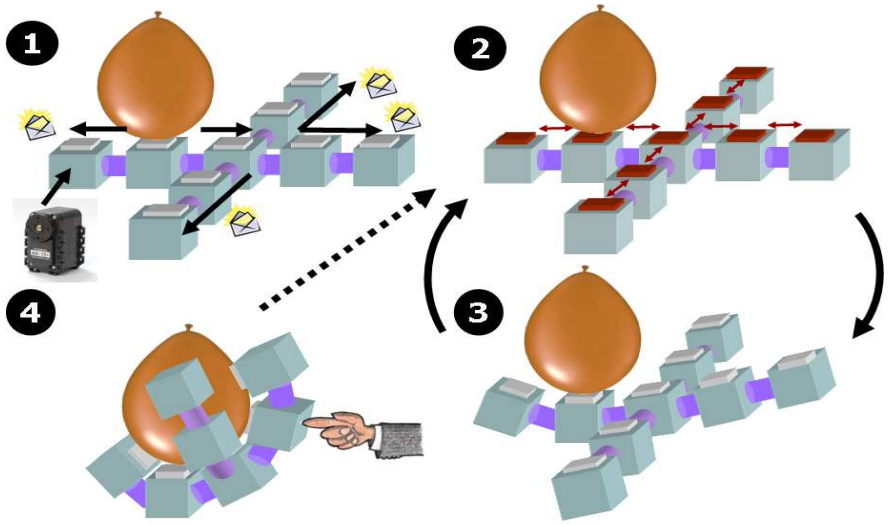


Figure 5.8: The algorithmic overview of the grasping task. Step 1: The first module starts sensing the presence of the object. It starts sending messages to it neighbors. Steps 2 and 3: Agents perform iterative sensing and actuation until they converge to the desired state. Step 4: When the robot is perturbed by exogenous force, it goes back to Step 2.

5.3 Modular Gripper

In this section, we illustrate another application: a modular gripper. The gripper is capable of reconfiguring itself to grasp an object using distributed sensing and actuation. The goal of the task is for modules to collectively grasp around an object while each module applies identical force to the object. If the gripper is perturbed, modules are required to quickly adapt and conform to another posture that ensures grasping stability. If this can be achieved, it can potentially simplify the complexity of grasping tasks by allowing each individual segment of the gripper to execute simple sensing and actuation.

The control law design follows a similar procedure as described in the previous examples. 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.

Task Specification: In this task, we view each module as a single agent. As shown in Fig. 5.8 (1), a modular gripper is composed of a chain of modular agents (chain-style), where each agent is equipped with a rotary motor 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 $\theta_p (\theta_{\min} \leq \theta_p \leq \theta_{\max})$. Therefore, each agent is

programmed to maintain equal pressure with respect to its connected neighbors.

Algorithmic Procedure: The illustration of the algorithmic procedure is shown in Fig. 5.8. 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 Fig. 5.8 (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 Fig. 5.8 (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:

$$x_i(t+1) = x_i(t) + \alpha \cdot (\theta_{R_i} - \theta_i) \tag{5.2}$$

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 returns to Step 2.

Convergence Analysis: The control law we showed in Eq. 5.2 satisfies condition 1 because sensory feedback $g(\cdot) = \theta_{R_i} - \theta_i = 0$ only when agent a_i's sensor reading is equal to that of its neighbor a_{R_i}. In addition, g is also anti-symmetric. However, it is nontrivial 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. The details of the proof are in the Appendix B.2. We can state the following theorem for the Eq. 5.2 control law:

Theorem 8. *Let Θ^0 be the initial sensory condition of the gripper and Θ^* be the desired sensory state such that $\theta_j = \theta_i$, for each agent a_i and its neighbor a_j. If configurations between Θ^0 and Θ^* are reachable, the control law Eq. 5.2 will lead the gripper to apply uniform pressure on the object, and the system's state will converge to Θ^*.*

Proof. see Appendix B.2. □

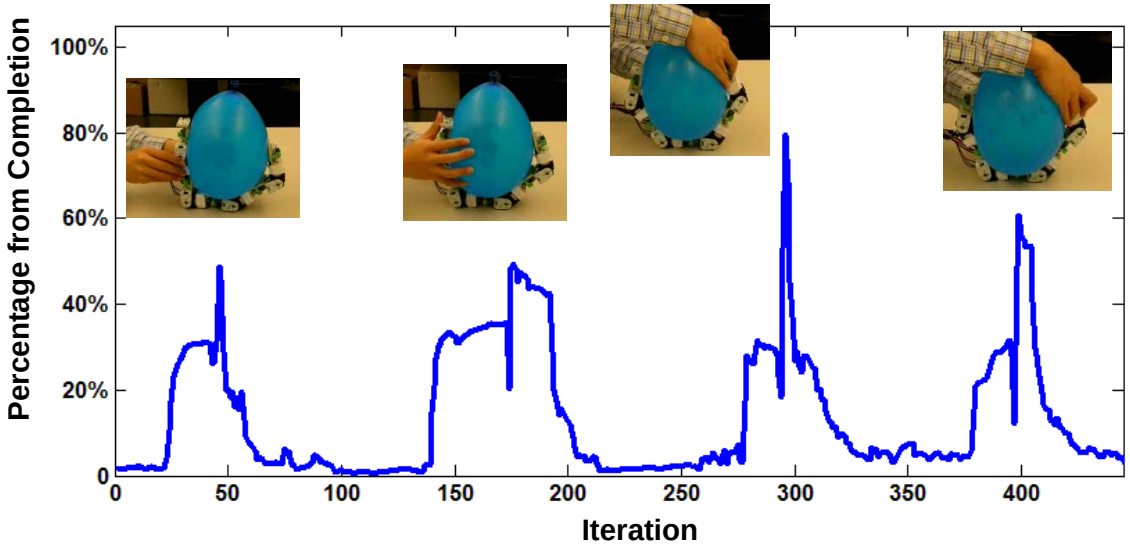


Figure 5.9: After the robot has reached the desired state, we constantly perturb the gripper by applying force. The robot is able to re-adapt after being perturbed.

Most of the controllers designed for grasping tasks have used a centralized architecture. This decentralized approach has the following merits: (1) the proposed decentralized and modular robot approach allows the whole system to adapt to local perturbations more efficiently; (2) given any initial contacting module, the gripper is able to form a grasping configuration that conforms to the shape of the object; (3) this control scheme is also applicable to different kinds of gripper configurations. These capabilities are demonstrated in the following experimental results.

The modular gripper adaptive grasping task presented here is also inspired by Hirose's work on a distributed feedback controller for snake robots [22]. He demonstrated that the snake robot can coil around an object using a distributed controller and force feedback. When applied to a coiling task, the decentralized framework presented here further allows each element of the robot to achieve equal pressure or any desired pressure distribution around the object. In addition, the control law provably converges to the desired state.

5.3.1 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 Eq. 5.2's convergence properties 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. Fig. 5.9 shows ϵ vs 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 decentralize 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 motors we use here have better precision than the linear actuators.

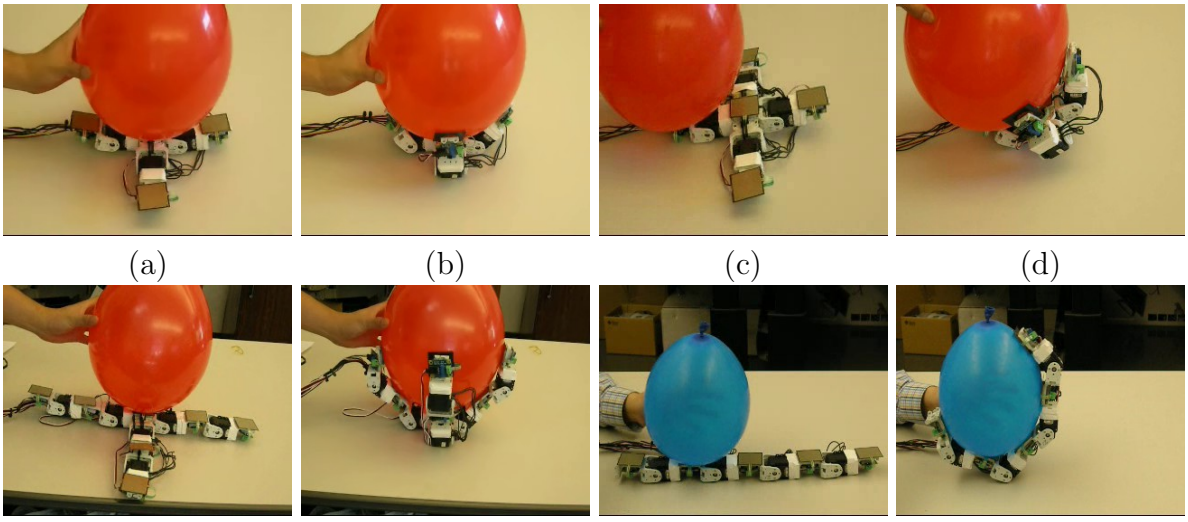
Different Initial Conditions and Configurations: We connect the agents to form a "cross" configuration as shown in Fig. 5.10 (a–d). We let different agents start to touch the balloon to examine the system's behavior under different initial conditions. Fig. 5.10 (a–h) shows a sequence of robot configurations 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 5.11 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 to 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 Fig. 5.12, we increase the number of agents from 5 to 9. We can see from the figure that there is no significant increase in convergence time when we increase the number of agents. 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 5-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 4.2 that decreasing the diameter of the agent network can increase convergence speed.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.10: (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.

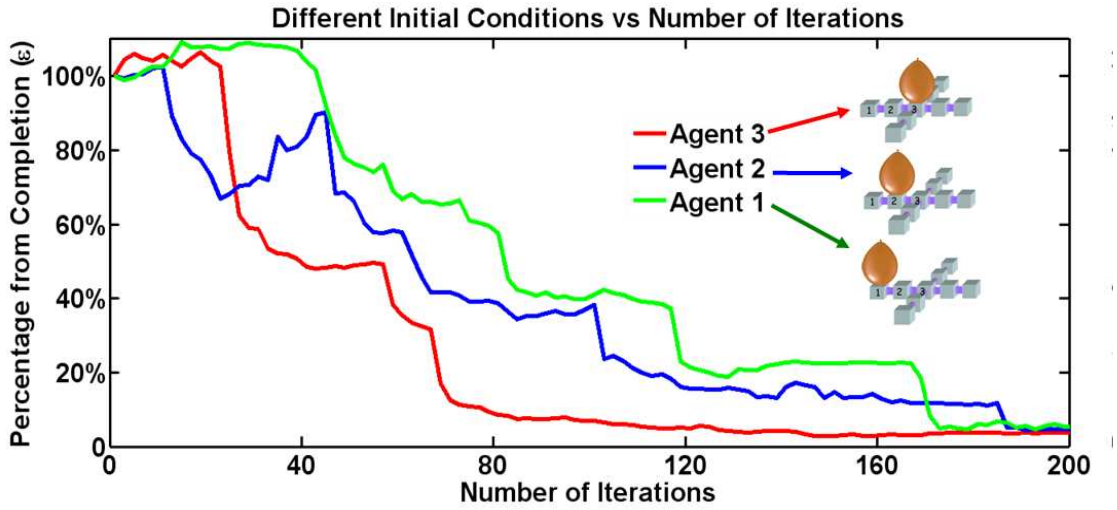


Figure 5.11: Experiments with different initial conditions. After $\sim$ 180 iterations, agents are capable of achieving less 3% distance from the consensus state in all three cases.

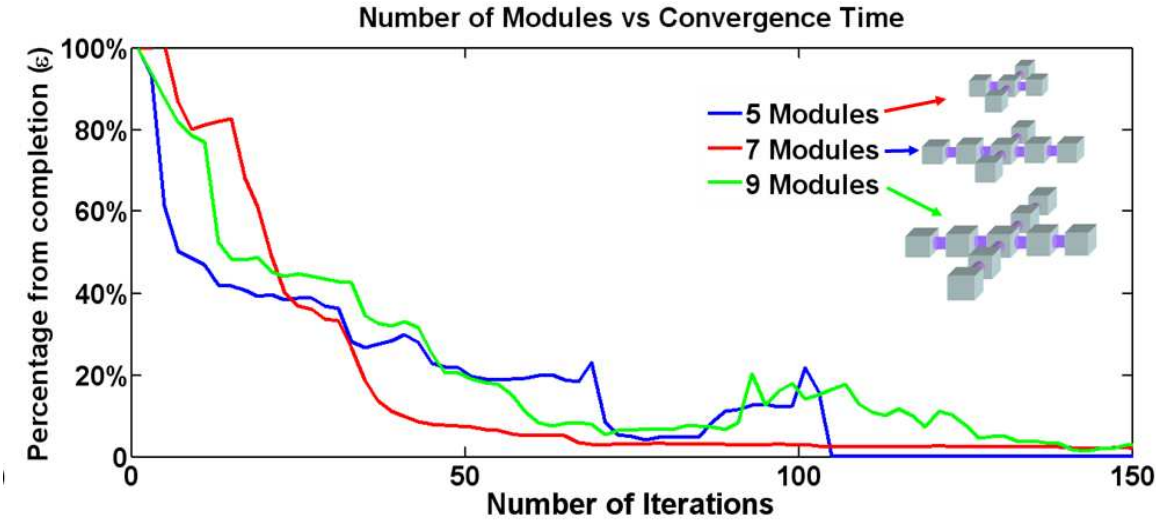


Figure 5.12: Scalability experiment. The decentralized algorithm is scalable with the number of agents. On the other hand, the network structure might affect the convergence speed. In the 7 and 9 agents cases, the diameter of network is 2 and it leads to longer time for the robot to complete the task.

5.4 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.

Task Specification: As shown in Fig. 5.13 (1), each modular 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.

Algorithmic Procedure: The algorithmic overview of the self-adaptive process is shown in Fig. 5.13:

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

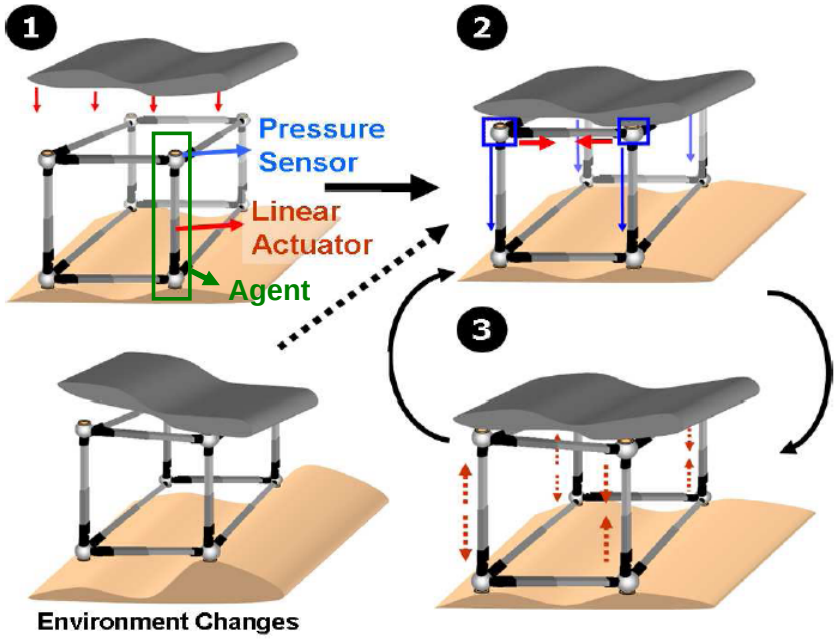


Figure 5.13: The algorithmic overview of the pressure-adaptive column. Step 1: an unknown object is placed on the robot. Step 2: Each agent sends its pressure reading to neighbors. Step 3: Step 3: Each agent continuously adapts based on its neighbor's states.

- **Step 2 (Sensing):** Each agent starts exchanging current pressure sensor feedback with its neighbors.
- **Step 3 (Actuation):** Each agent computes its actuators new parameters based on the sensor feedback that it receives from all 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:

$$x_i(t+1) = x_i(t) + \alpha \cdot \sum_{a_j \in N_i} (\theta_j - \theta_i) \tag{5.3}$$

Convergence Analysis: Here, the feedback function g is simply $g(\theta_j, \theta_i) = \theta_j - \theta_i$. g satisfies conditions Eq. 3.4 – Eq. 3.6, 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

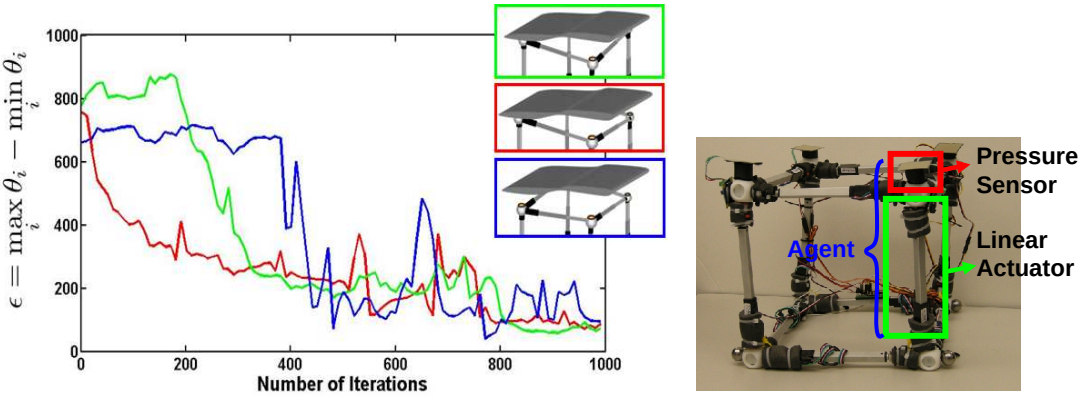


Figure 5.14: Pressure-adaptive column with different initial conditions. We let the number of initial contacting agents to be one (green), two (red), and three (blue) respectively and examine how the column respond with different initializations. After 1000 iterations of running the control law, the distance measure, ϵ, decreases to less than 100. Right: Physical prototype of the pressure-adaptive column.

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 (Eq. 5.3) will allow the robot to converge to the desired state. This leads to the following theorem:

Theorem 9. *Let Θ^0 be the initial condition of the robot and Θ^* be the desired state, so that $\theta_j = \theta_i$ for every agent a_i and its neighbor a_j. If configurations between Θ^0 and Θ^* are reachable, the control law Eq. 5.3 will lead all agents to converge to Θ^* at an exponential rate.*

Proof. see Appendix B.1. □

The proof the above theorem requires one to show that the linear actuator's control state $x_i(t) \to x_i(t+1)$ is driven toward the correct direction by feedback function $g(\cdot)$ at all time.

5.4.1 Experimental Results

In the pressure-adaptive column experiment, we test 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 pound. The robot starts in three different configurations,

such that the number of initial contacting agents is different, ranging from one to three[2]. We define

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

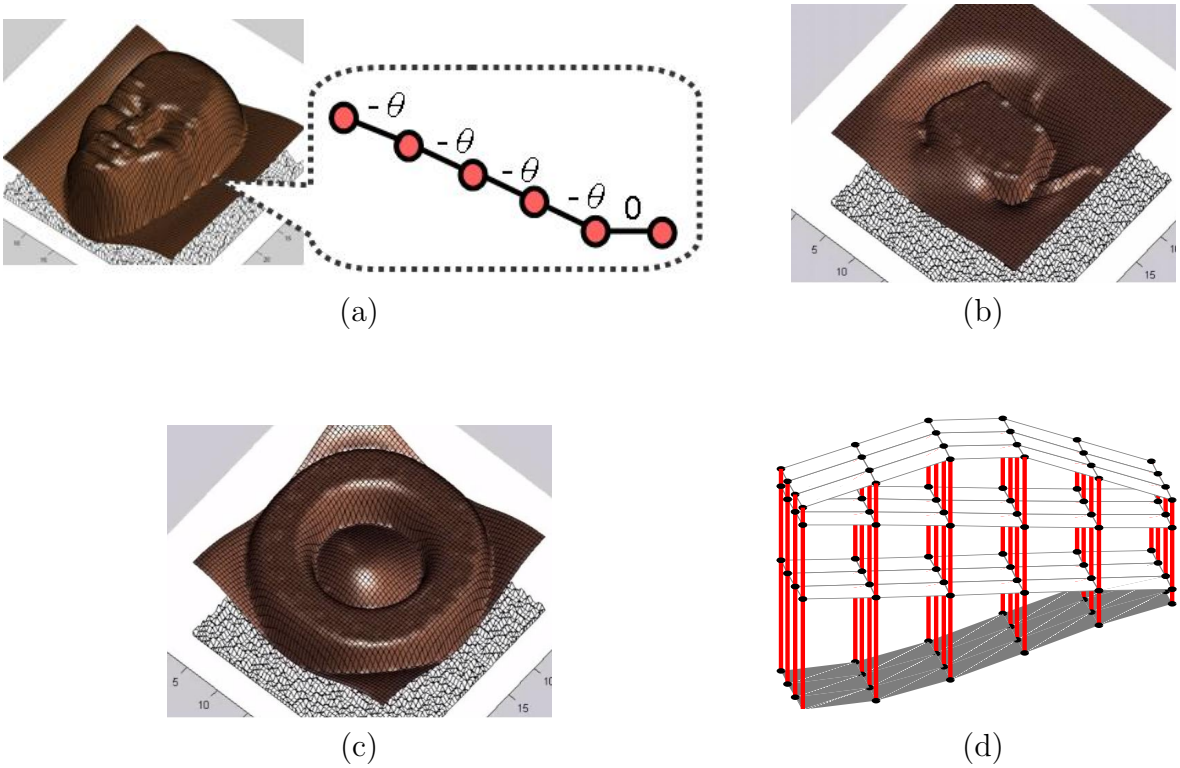
the difference between maximal and minimal sensory reading among agents, as a measure of distance from reaching consensus. We can see from Fig. 5.14 that ϵ decreases from 800 to around 100 after 1000 iterations ($\sim$ 10 sec. in real time) in all three cases. Note that the sensor we use is very noisy and sensitive to slight perturbations of the linear actuators. Therefore, α in Eq. 5.3 is set to be a very small constant to avoid the column from being over-sensitive to perturbations. This naturally leads to a longer convergence time. We also note that the larger fluctuations in the blue curve is primarily due to the object significantly shifted its center of mass when more agents contact it.

5.5 Other Self-Adaptive Tasks

The framework I propose here can be used to solve more tasks than consensus-type tasks, as presented in the previous subsections. In Eq. 5.1, 5.3, and 5.2, we set either $\theta^*_{ij} = 0$ or $\Delta^*_{ij} = 0$, $\forall i$ and $j \in N_i$ such that agents eventually achieve the same (sensory) states. In fact, if $\theta^*_{ij} \neq 0$ or $\Delta^*_{ij} \neq 0$. This allows one 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., the self-balancing table or terrain-adaptive bridge) to three-dimensional connectivity. Here, we illustrate several 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 Fig. 5.15 (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. Fig. 5.15 (a - c) shows several different complicated shapes that are rendered in simulation when we have 16,000 modules in our system. We can also see from Table 5.1 that the algorithm scales quite well to the number of modules. When

[2]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.



(a) (b)

(c) (d)

Figure 5.15: This 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. The figure shows an adaptive-building structure formed by the modules, and the grey region indicates uneven terrain.

	human face	donut	teapot
mean	317.9	64.6	19.0
std.	14.42	4.74	0.67

Table 5.1: Mean and Standard deviation of number of iterations for 12288 agents in different 3D shapes (ϵ is 10% of the initial error.

the numbers of modules have increased ~ 1000 times, the number of iterations required only increases on the order of 10.

- **Adaptive Building Structure:** In the previous applications, we connected modules to form a surface. In fact, we can extend modules' connectivity to three dimensions to form an adaptive building structure. In this simulation (Fig. 5.15 (d)), modules form several surfaces that correspond to different levels of a building; some linear modules form the supporting pillars of the building. From the figure, we can see that the building floors autonomously adapt to maintain levelness while the underlying terrain changes.

- **Potential Extensions:** When modules are equipped with different sensors and actuators, many other applications can be generalized from this framework. Here, I 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 involves viewing dynamic tasks as a sequence of self-adaptations.

5.6 Discussion and Lessons Learned

In the previous experiments, we have seen that one can easily specify tasks and program agent control laws by following the framework presented in Chapter 4. The applications shown here represent a small subset of what is achievable within this framework. One can potentially apply this approach to solve tasks in various systems that can be viewed as sensor-actuator networks. Further, many existing robotic systems can be turned into sensor-actuator networks if we interface each component with the appropriate capability (e.g., sensing). These systems can then utilize this decentralized approach, thereby acquiring the associated scalability and robustness.

We can also see from various experiments that our empirical results coincide with various theories I presented in Chapter 4. From all three applications, agents converge to the desired state because their agent topologies are all connected. One can also set ϵ in inequality 4.12 appropriately based on distinct sensor and actuator noise levels in different applications. From the experiments in Section 5.3.1, we can also empirically

see how agent topology, especially diameter, affects convergence speed. As shown in the robustness experiments in Section 5.2.1, one can prevent an agent's complete actuation failure by equipping each agent with redundant actuators; this verifies our discussion in Section 4.2 of Chapter 4. From the simulation results, we also see that this approach empirically scales quite well to systems with a large number of agents.

Our various experimental results validate that the theoretical results I presented in the previous chapter are a good indicator of how systems would perform in practice. In the following, I discuss potential limitations and several interesting extensions for this study.

5.6.1 Potential Challenges and Limitations

In this Chapter, we have seen that this framework can be applied to solve various applications that can be viewed as sensor-actuator networks. It nevertheless has its limitations: the current scope of this approach only allows it to be applied to solving tasks that can be formulated as a single self-adaptive task. In Chapter 6, we will see how this approach can be adapted to solve modular robot dynamic locomotion tasks by a sequence of repetitive self-adaptations.

To extend this approach to more complex scenarios (e.g., tasks that require solving by many different types of self-adaptive tasks), a global "task compiler" that can automatically generate a sequence of (potentially different) self-adaptive tasks and their corresponding inter-agent relationships will be required. Hence, similar control laws can still apply. This is beyond the scope of this thesis but would be an interesting direction for future research.

5.6.2 Centralized Planning vs. Decentralized Control

While the type of decentralized approaches presented here can scale to a vast number of agents, the task representation scope is usually more constrained than that of centralized planners, which can potentially express a wider range of tasks across a longer time horizon. In contrast, the main limitation of centralized planning is its scalability in degrees of freedom (e.g., the number of agents or number of actuators) that it can coordinate simultaneously (also referred to as the 'curse of dimensionality'). Although many approximate centralized planning strategies have been developed to address such a challenge [34, 36], these strategies are usually difficult to implement and are constrained to specific task domains.

Biological systems have evolved to intelligently leverage the strengths of both approaches. For example, human beings use their brains to plan sophisticated tasks (e.g., arranging daily schedules) in a centralized manner, whereas millions of cells

regulate our body in a decentralized way. It would be interesting to explore a mixed strategy that is composed of both a centralized planner and decentralized controllers. We see that decentralized strategies show strengths in tasks that require constant adaptations to a changing environment, which helps us to identify scenarios that they can solve more effectively. One can then design a centralized planner that oversees and coordinates decentralized tasks. For example, a humanoid robot utilizes a centralized planner to plan a trajectory to reach an object, and decentralized controllers run on a gripper that exploits distributed sensors to grasp the object and self-adapt to sudden external impacts.

5.7 Summary

In this chapter, I have presented how one can utilize unified principles to design agent control laws for various applications through several real robot applications, including (1) self-adaptive modular structures that can autonomously adjust to terrain changes via distributed sensing, (2) a pressure-adaptive column that can adjust to external pressure to maintain uniform pressure on an object, (3) an adaptive modular gripper that can grasp around an object with a desired pressure distribution, and (4) other sensor-actuator network applications (e.g., an adaptive prosthetic device). Experimental results show that such a control scheme allows these modular robots to correctly achieve the desired goal state, which coincides with the theoretical results presented in the previous chapter. In addition, our experimental results also show that such an approach is robust toward internal actuator failures and external perturbations. There also many interesting directions that one can further pursuit based on this study, such as interfacing a decentralized agent control with a centralized planner to leverage the advantages from both decentralized and centralized approaches.

Chapter 6

Self-Organizing Strategies for Adaptive Locomotion

In Chapter 5, we have seen that the proposed self-organizing strategy allows many multiagent systems to achieve statically-defined self-adaptive tasks. In this chapter, I show that this approach can be generalized to more dynamic scenarios with several different modular robot locomotion tasks.

Modular robots, built from many identical and interchangeable components, have a potential advantage over traditional robots in performing locomotion with different *configurations* and *strategies* that adapt to different environments. Towards this goal, there are two main challenges to be addressed:

- *Configuration adaptation:* How can modules find a controller appropriate for any given configuration? Ideally, we would like to have a unified control strategy that allows robot to locomote efficiently in each possible configuration.
- *Environment adaptation:* How can modules collectively perform adaptive locomotion based on different environment conditions? Animals can intelligently adjust their locomotion or climbing strategies based on different terrain conditions. How can we program modules to achieve such a level of sophistication?

I apply this self-organizing approach to permit configuration and environment adaptation in two modular robot locomotion tasks. In the configuration adaptation task, I show that 2D modular robots made of square modular agents assembled with an arbitrary configuration can effectively achieve collective and directed locomotion with no centralized controller. Individual agents communicate locally and *provably* achieve consensus in coordinating movement in a common travel direction (shown as Fig. 6.1). In the environment adaptation task, I demonstrate that a modular tetrahedral robot is capable of performing terrain-adaptive locomotion with a sequence

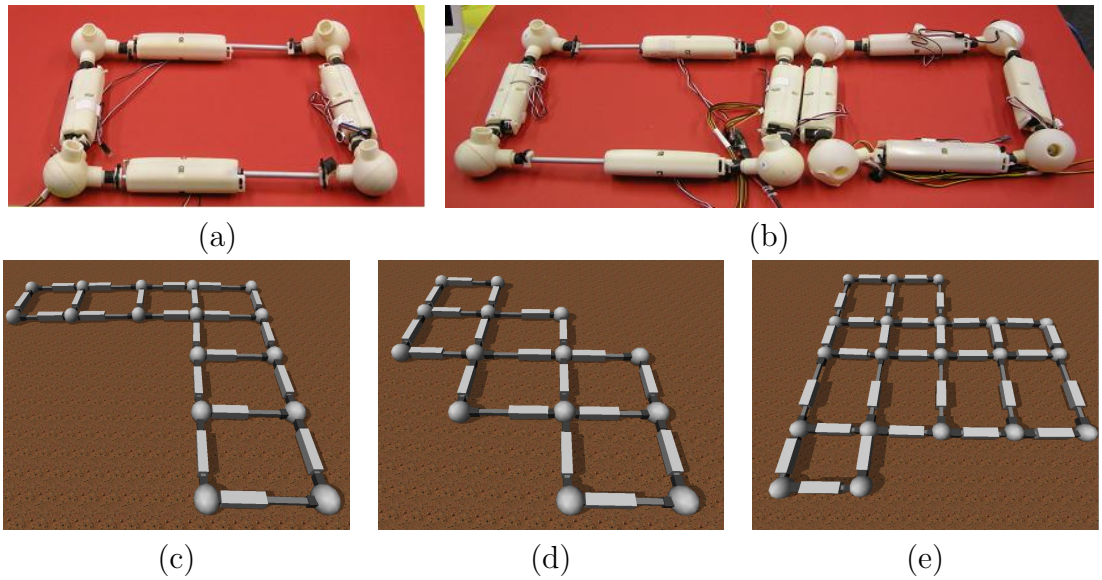


Figure 6.1: : **Configuration-adaptive Locomotion**: Modules of different configurations, including (a) single hardware agent, (b) two linked hardware agents, (c) L-shape configuration, (d) Ladder-like configuration, and (e) arbitrary assembly, can achieve efficient locomotion with this framework.

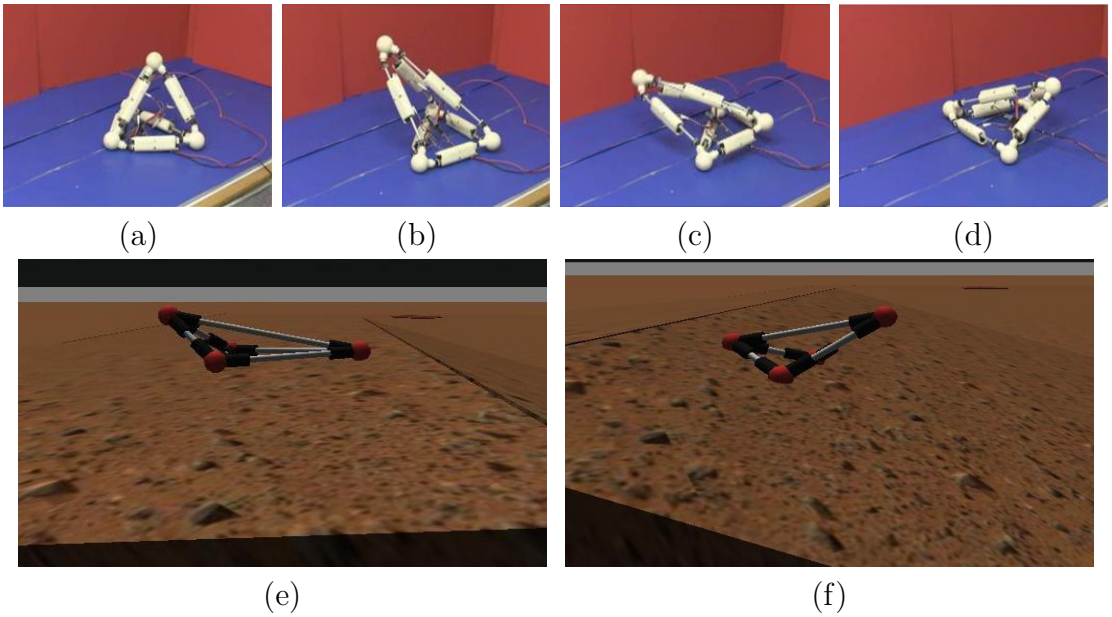


Figure 6.2: **Environment-adaptive Locomotion**: (a-d) Real robot motion sequence. A modular tetrahedral robot is capable of performing terrain-adaptive rolling locomotion that adapts to (e) inclining and (f) declining slopes with a sequence of self-adaptive tasks. The simulation is implemented with Open Dynamics Engine.

of self-adaptive tasks. This approach allows the robot to adjust its locomotion cycle adaptively based on different slope conditions (e.g., when the robot is on a declining slope, modules exploit gravity and collectively adjust the locomotion cycle to become shorter (shown as Fig. 6.2)). I further show how this environmentally-adaptive strategy generalizes to the modular robot's autonomous climbing tasks.

This chapter is organized as follows. In Section 6.1, I describe the modular robot hardware and simulation environment that I use to implement the modular robot tasks in this chapter. In Section 6.2, I describe the algorithmic procedure of applying this self-organizing framework to address configuration adaptation tasks. In Section 6.3, I also present various experiments that validate this approach and can generate efficient locomotion in complex and asymmetric assemblies. In Section 6.4, I describe how to model rolling-type locomotion with a modular robot as a sequence of self-adaptive tasks and several experiments that verify the effectiveness of the approach. In Section 6.5, I show how this approach generalizes to modular robot vertical climbing tasks

6.1 Modular Robot Design

For this work, we developed a new modular robot design, inspired by NASA's tetrawalker [57], Odin's deformable modular robot [39], and our own Morpho modular robot [87] (based on the Tensegrity concept [24]). In this section, I describe the mechanical design of our strut-based modular robot. In this design, the robot is built by connecting many linear modules that can change their body lengths. Each module takes the form of a telescoping link consisting of a pair of linear actuators. Modules are connected together through spherical joints to form different configurations. Compliant connectors are attached to the end of the modules. The whole structure can deform and return to its original shape while modules perform linear actuation. Here, I provide a detailed description of the module and the spherical joint.

6.1.1 Module Hardware Design

The node/link design of our modular robot is inspired by previous deformable modular robots that have used such components to form other architectures, such as tetrahedra [39, 57, 87]. Our module design is composed of the following three components:

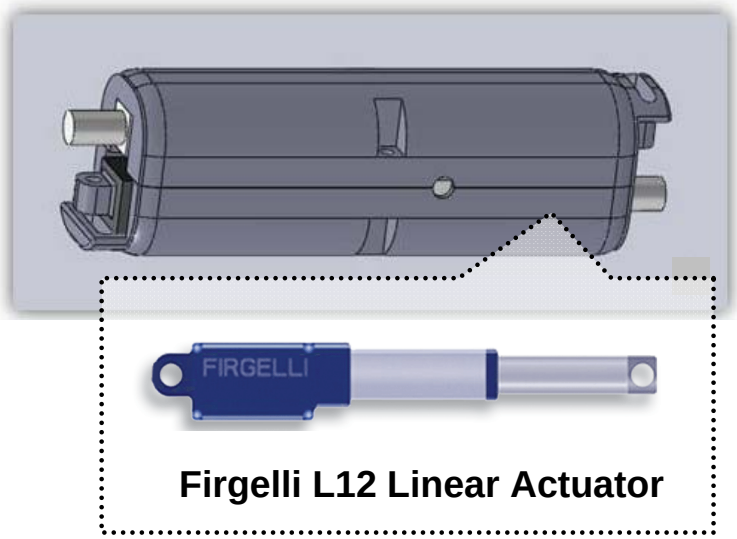


Figure 6.3: Each module consists of a pair of linear actuators, bound together and pointing in opposite directions so as to achieve an approximately 3:1 extension capability if both actuators are extended simultaneously. (Design and Photo by Rebecca Belisle)

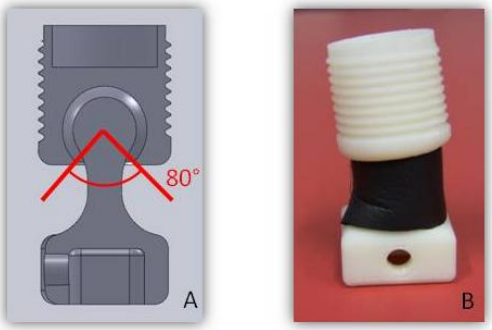


Figure 6.4: The compliant connector consists of a ball-and-socket joint (A) inside a foam tube (B) that gives it restoring force to maintain a preferred position. (Design and Photo by Rebecca Belisle)

- **Linear Actuator:** Each module consists of a pair of Firgelli linear actuators that can elongate or contract at a maximal speed of 2.3 cm/sec. The linear actuators are bound together and placed in ABS plastic housing. The two actuators point in opposite directions so as to achieve an approximately 3:1 extension capability if both actuators are extended simultaneously. The detailed design of a single module is shown in Fig. 6.3.

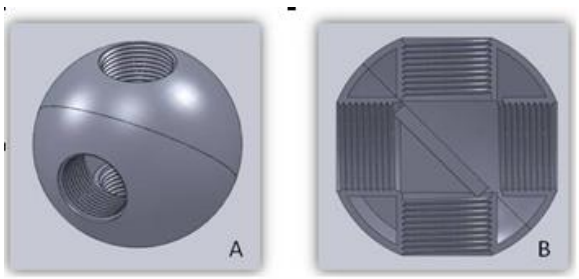
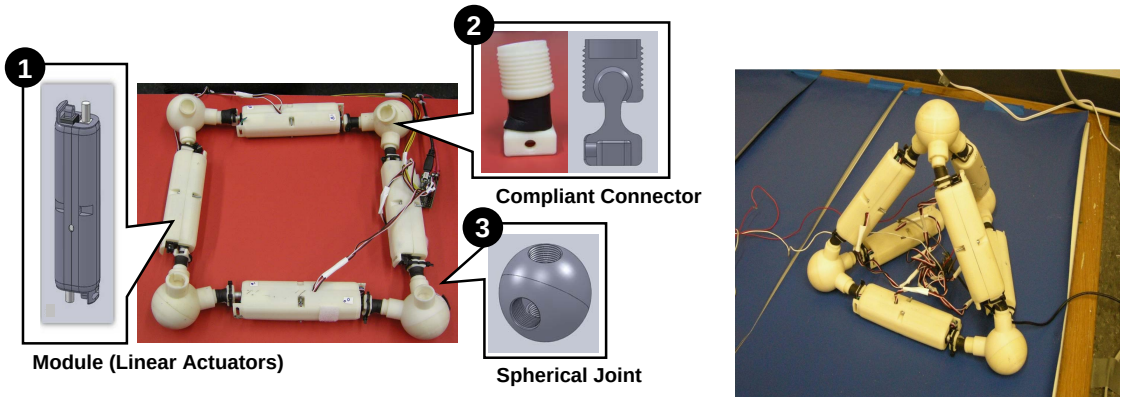


Figure 6.5: Spherical joint design. View A shows the exterior view of the joint and view B shows cross section. (Design and Photo by Rebecca Belisle)



(a) (b)

Figure 6.6: Modules can be connected together with compliant connectors and spherical joints to form different configurations, such as a square (a) and a tetrahedron (b).

- **Compliant Connector:** Each linear actuator is attached with a compliant connector to interface with a spherical joint. There are two primary requirements for the connectors. First, they are required to supply sufficient compliance to allow the overall structure to deform when modules perform actuation. Second, they need to maintain rigidity to sustain the load of the overall structure. In our design, we use a ball-and-socket joint inside a foam tube that provides sufficient flexibility and restoring force to maintain structural rigidity (shown in Fig. 6.4).

- **Spherical Joint:** The joints are designed to connect the different modules together. To hold the modules, the joints were designed with internal thread to interface with the compliant connectors (shown as Fig. 6.5).

One can connect modules together to form various different configurations. Fig. 6.6 shows two example configurations formed by our modules. The main challenge for this modular robotic system is then to design an algorithmic approach for the robot to perform tasks (e.g., locomotion) in all possible configurations.

6.1.2 Simulation Environment

In addition to the hardware prototype, we also construct a physics-based simulation (shown in Fig. 6.1 (c – e) and 6.2 (e, f)) with Open Dynamics Engine (ODE). This simulator allows us to examine the behavior of this control approach for large numbers of modules. The dimensions and mass of each simulated component are set based on the real robot. Several environmental parameters, e.g., parameters for the ground contact model, are optimized based on real robot behavior to provide high fidelity simulation.

6.2 Collective Locomotion in an Amorphous Modular Robot

In this section, I describe the procedure that generalizes this self-organizing framework by solving locomotion tasks for a 2D modular robot with an arbitrary configuration. The approach is inspired by how natural systems consisting of many interchangeable agents perform locomotion with purely decentralized controllers. Cellular slime molds, a classic example, have a stage in their life cycle in which many individual amoebae aggregate into single multicellular slugs of variable morphology, which can then crawl as though a single coordinated unit [7].

Such examples led us to consider modular robots with units connected in any arbitrary geometry (i.e., amorphous robots) and distributed controllers that allow for coordinated movement in a direction determined by the group as a whole.

In this section, we consider a system where each agent is composed of four modules connected in a square. The agent can travel in any of the four cardinal directions, using a simple periodic actuator (discrete CPG). When multiple agents assemble to form an aggregate robot, we show how the distributed homeostasis algorithm can allow the group to agree on a common direction of travel and to coordinate their actuation cycles to travel in that direction.

6.2.1 Approach Overview

The overall goal is to develop an amorphous modular robot system that is composed of many self-mobile agents that can travel and act independently (e.g., travel and explore an area). When multiple such agents join together into an aggregate robot as required (e.g., for efficient locomotion), the new assembled robot needs to be able to adapt to its new configuration and travel in a new way.

Towards this goal, I assemble four modules together to become a square agent that can perform locomotion. Each square agent has independent sensing, motion, communication, and computation capabilities. A single agent's hardware prototype is shown in Fig. 6.7 (a).

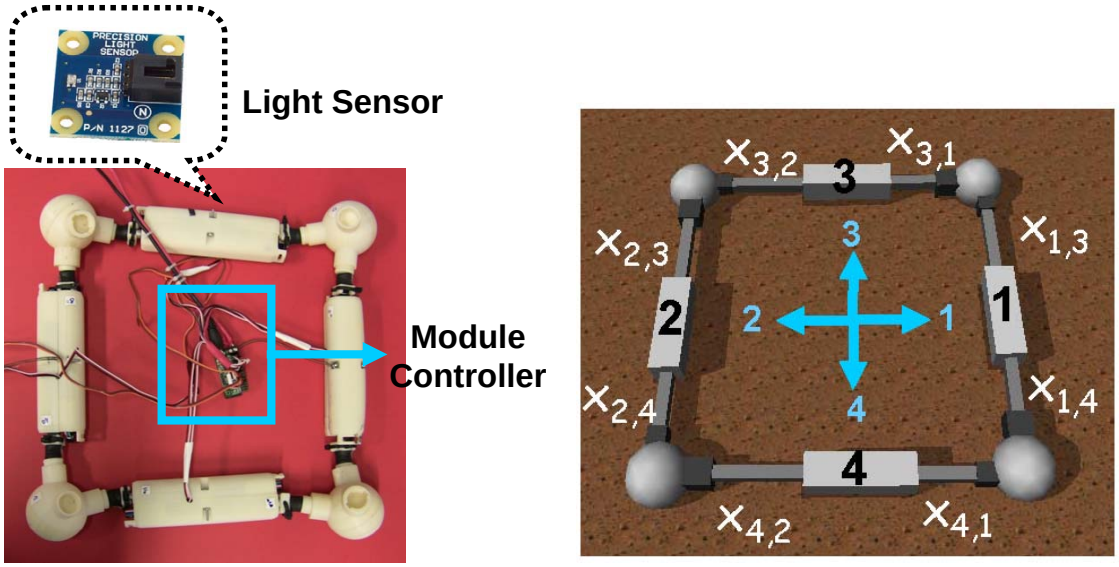
When multiple such square agents are assembled together to form a new geometry (some example geometries are shown in Fig. 6.1 (c - e)), agents in the new configuration then use the distributed homeostasis algorithm (Eq. 3.1) to coordinate their locomotion phases and frequencies. A key advantage of using the DH algorithm is that it is completely *decentralized* - there is no leader agent that determines coordination or direction. As I have proved that the DH leads agents to achieve the desired state in any connected topology, this approach can effectively achieve configuration-adaptive locomotion. It does not rely on prior knowledge of agent topology and can be applied to any connected configuration of agents.

This approach is verified with both hardware and simulation experiments. Using real robots, I show that agents can achieve coordinated and directed locomotion, even with initially conflicting goal directions among agents. Using physics-based simulations, I show that this coordination method can generate efficient locomotion in complex assemblies. Coordination is essential for effective locomotion: robots composed of ineffectively coordinated agents trying to travel in the same direction can barely move, whereas those using the proposed method can travel at least as fast as individual agents. Finally, I also show that this approach is scalable with robot size, with convergence time for direction agreement and locomotion coordination being proportional to the connectivity graph diameter and number of agents, respectively.

In the following, I first describe a single agent's locomotion algorithm (Section 6.2.2). I then describe the algorithmic procedure for a group of assembled agents to achieve coordinated locomotion (Section 6.2.3 – 6.2.5).

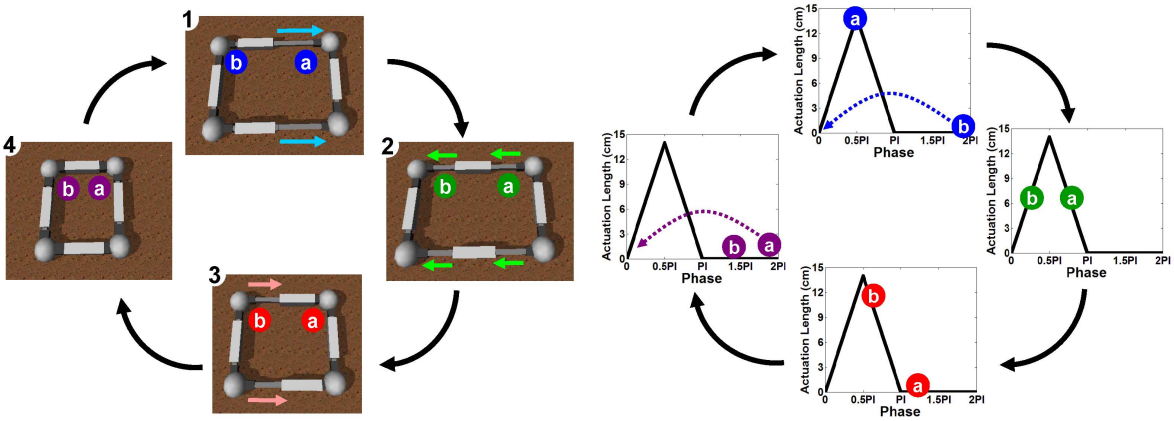
6.2.2 Single-Agent Directed Locomotion Algorithm

Here, I describe a control algorithm based on a discrete CPG that allows a single agent to perform directed locomotion. Before I describe the algorithmic procedure, I first present the agent model:



(a) (b)

Figure 6.7: (a) A single agent hardware prototype. There is a module controller that can simultaneously control four modules. There is a light sensor mounted on each module. (b) Linear actuator indexing scheme. The four modules, and corresponding cardinal directions, are indexed numerically. Linear actuators within each module are indexed by the labels of their own module and the adjacent module.



(a) (b)

Figure 6.8: (a) Rightward movement steps: Locomotion in any of the four cardinal directions can be achieved by a cycle of actuation like this one for rightward movement. (b) The corresponding phases of linear actuators in each module. X axis indicates phase and Y axis indicates the length of actuator extension.

Module Connection: A single agent consists of four modules connected in a square configuration by four nodes (Fig. 6.7 (a)). Four spherical connectors at the corners of the square configuration connect the modules together. Compliant connectors attached to the corner nodes let the frame deform and return to its original shape.

Module Controller and Sensors: There is a module controller that can simultaneously control the actuation of the four modules. There is a light sensor mounted on each module, and the module controller can determine an agent's traveling direction based on sensor values from the four modules.

Communication: As is typical for modular robotic systems [28, 39], I assume that each agent can exchange messages with its physically-connected neighboring agents.

For single-agent locomotion, the asymmetry produced by extending only one actuator or the other in a module opens up the possibility of using such elements for locomotion in a desired direction. The following sequence of actions for the top and bottom modules moving in unison produces net movement to the right, for instance (Fig. 6.8 (a)):

- Step 1: Extend the right arm only.
- Step 2: Retract the right arm while simultaneously extending the left arm.
- Step 3: Retract the left arm.
- Step 4: The cycle starts and ends with a configuration in which all actuators are retracted.

At each step, elements pushed to the right are fewer and correspondingly lighter than those pushed to the left; the net motion is thus to the right. Movement in any other cardinal direction can be achieved analogously.

Here, I describe the controller formally. Each module is identified by a numerical index, which also corresponds to a robot-relative direction (Fig. 6.7 (b)). The actuation state (length) of each of the two linear actuators in a module is denoted by $x_{i,j}$, where i is the index number of the module and j is the index of the neighboring module. This index scheme allows us to uniquely specify the state of each linear actuator.

Let d indicate the index of the desired traveling direction and d' indicate the opposite direction. Γ is a set of two modules that enables the agent to move while being actuated. For instance, if the desired traveling direction is 1 ($d = 1$), then d' is 2 and Γ contains two modules that are parallel to the traveling direction ($\Gamma = \{3, 4\}$).

The modules that are not in Γ remain static. The actuation state function for modules in Γ can be written as follows:

$$\forall i \in \Gamma, x_{i,d}(t) = f(\theta \bmod 2\pi) \quad (6.1)$$

$$x_{i,d'}(t) = f((\theta - \pi/2) \bmod 2\pi) \quad (6.2)$$

where $\theta = \omega \cdot t$ is the phase of the actuator; ω is a parameter determined by actuation speed; and actuation function f (Fig. 6.8 (b)) is formally defined as follows:

$$f(z) = \begin{cases} \lambda \cdot z & \text{if } z \leq \frac{\pi}{2} \\ \lambda \cdot (\pi - z) & \text{if } \frac{\pi}{2} < z \leq \pi \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

where $\lambda = 2L/\pi$, with L the maximal actuation length. The two actuators in a module go through the same cycle with phase difference $\pi/2$. Fig. 6.8 (b) shows the sequence of phases corresponding to the locomotion cycle described above. In Section 6.3, I show real robot experiments and simulations to validate this mode of locomotion.

An agent's direction of travel may be chosen based on sensor feedback; for instance, an agent equipped with light sensors may exhibit phototaxis by choosing the direction with the highest light level.

6.2.3 Assemblies of Agents

In this subsection, I describe how agents coordinate their movements when they are aggregated together. A key aspect of the approach is that *it does not assume any specific connectivity*. This procedure can be described in the following three steps:

- Initially, each square agent has some preferred traveling direction (Fig. 6.9 (1)).
- Next, it communicates with local neighbors to reach an agreement on a common direction (Fig. 6.9 (2)).
- Finally, agents start coordinating their movement cycles until they achieve an efficient traveling wave (Fig. 6.9 (3, 4)).

Here, I do not address how agents come together and attach to form aggregate robots, which is itself an important topic of research [20, 44]. Instead, I focus on coordinating the resulting collective: how does an arbitrary assembly of agents act as a single unified unit? I first describe assumptions about agent connections.

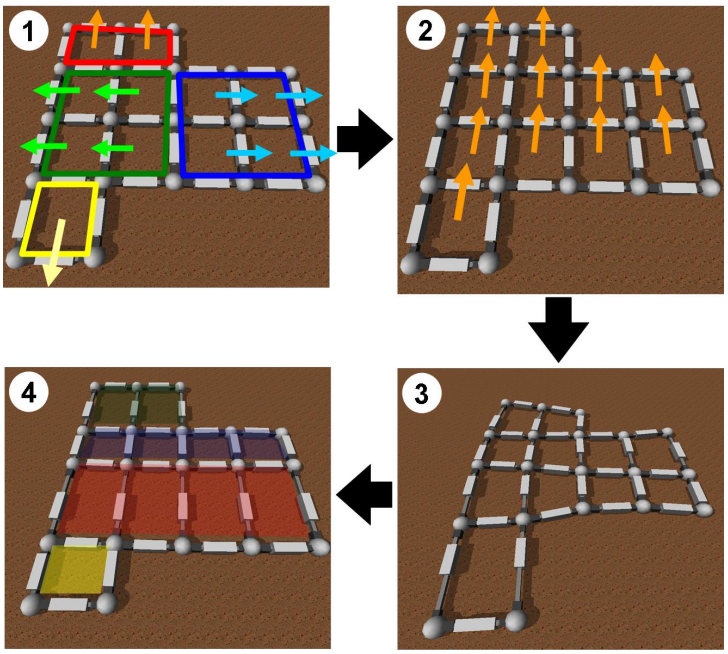
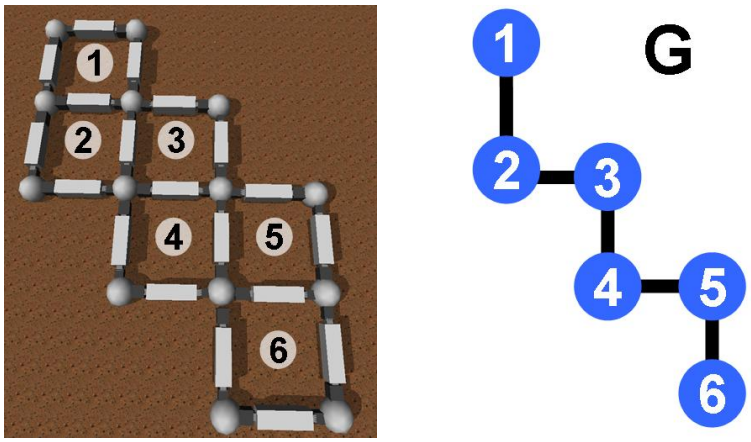


Figure 6.9: 1: An asymmetric assembly of agents, with different initial preferred traveling directions. 2: Modules communicate with local neighbors and reach a global agreement on a traveling direction. 3: Modules start traveling toward the agreed direction with uncoordinated movement cycles. 4: Modules achieve coordinated movement and locomote efficiently. Color blocks represent locomotion phase of the agents. Each agent maintains the same phase as its horizontally-connected neighbors; phase in adjacent rows differs by π.



(a) (b)

Figure 6.10: Graph representation (b) of a robot assembly configuration (a).

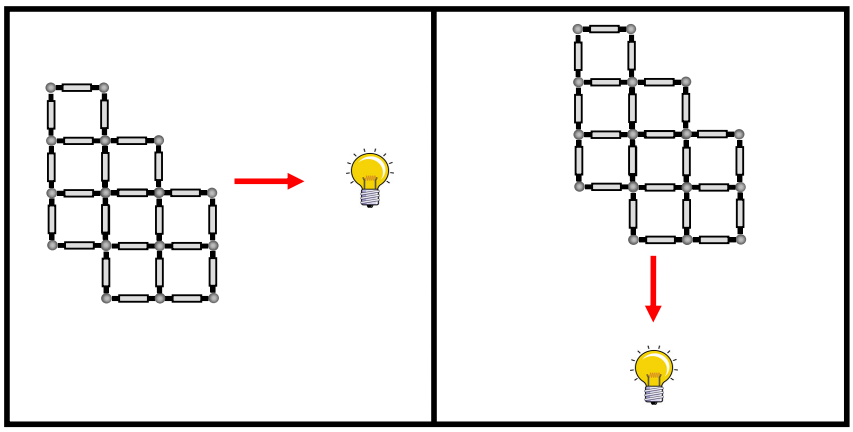


Figure 6.11: Distributed sensors on agents allow the whole assembly to agree on a consensus direction towards light source, exhibiting phototaxis behavior.

An aggregate robot is composed of agents attached edge-to-edge, so as to effectively occupy sites on a square grid to form a contiguous structure of arbitrary geometry. In simulations, I implement a shared edge as a single module with doubled mass, rather than doubling the modules along each edge by literally putting two agents side by side. The proposed framework does not rely on agents to assemble into specific configurations, and it allows arbitrary robot assembly configurations to achieve coordinated movements.

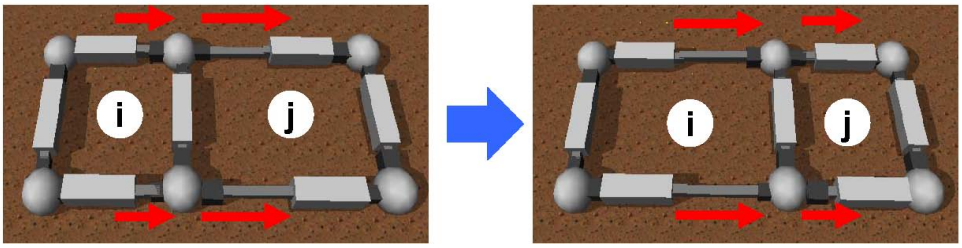
Each agent in the aggregate robot maintains an independent controller,[1] and is able to communicate real values bi-directionally with its physically-connected neighboring agents. I assume that the two controllers of a pair of agents that share an edge can agree on a common value for the extension of the corresponding linear actuators. An undirected graph G is used to represent such a communication topology by representing an agent with a node and a communication link as an edge (Fig. 6.10).

In the following, I present the approaches associated with agreement regarding a common direction of travel (Section 6.2.4) and the coordination of movement cycles (Section 6.2.5).

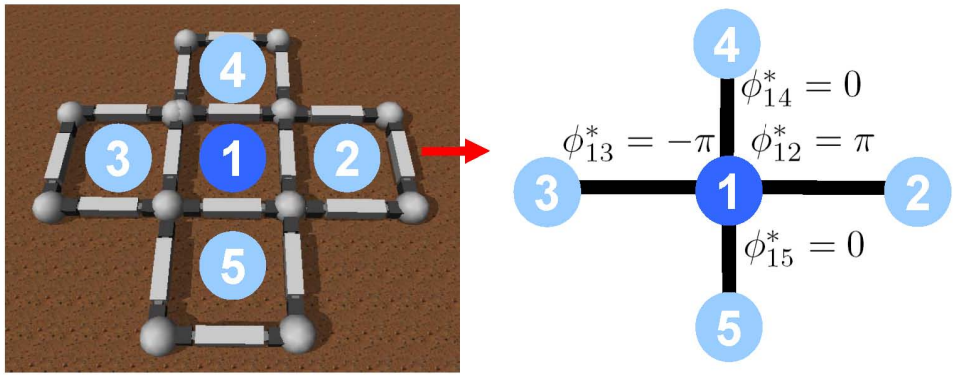
6.2.4 Collective Decision: Direction

In order for the amorphous robot to locomote effectively, all agents must agree on a single direction. Agents may have their own initial preferences (e.g., based on sensor feedback as above). The goal for the aggregate robot may be to travel in the

[1]The hardware prototype actually simulates a decentralized set of independent controllers using a single centralized controller.



(a)



(b)

Figure 6.12: (a) The cycles of two linked agents need to be offset so that both move the "passive" edge they share at the same time. (b) The desired phase differences between agent 1 and its neighbors. Red arrow indicates the robot's traveling direction. Agent 1 needs to achieve $\phi^*_{12} = \pi$ with its front neighboring agent, $\phi^*_{13} = -\pi$ with its back neighboring agent, and $\phi_{14} = \phi_{15} = 0$ with other agents.

direction of the most intense stimulus sensed by any unit, such that the robot exhibits a behavior like collective phototaxis or chemotaxis based on distributed sensing (Fig. 6.11).

In such a case, each agent will have an initial preferred direction with an associated confidence level. A simple algorithm can be used to reach global agreement: each agent repeatedly updates its direction and confidence to match that of the highest-confidence unit in the set consisting of itself and its neighbors. If an agent has the same confidence as a neighbor but a different preferred direction, it can break the tie by adding a small random value to its own confidence (with mean 0 so that either direction is equally likely to win). Such an algorithm will reach global agreement in a number of iterations equal to the maximum distance between the highest-confidence non-outlier agent and any other agent (i.e., in the worst case, the graph diameter; in the best case, half of the graph diameter). If updates are asynchronous, fewer

iterations may be required because a value may be able to propagate through multiple agents in a single iteration if the updates happen to take place in the right order.

Because sensed quantities of interest are likely to change slowly over space compared to the distance between agents, neighboring agents will typically start with confidence values that are not too dissimilar. Thus, a value much higher than any neighboring values is likely to reflect a sensor malfunction rather than a highly localized phenomenon of interest and should accordingly be discarded. This can be implemented as follows: if an agent finds that its confidence value is much higher than any of its neighbors, it ignores its own state and adopts that of its highest-confidence neighbor; if an agent finds that exactly one of its neighbors has a very high confidence that may indicate an error, it waits one update cycle before adopting its neighbor's state to see whether that state persists or is discarded.

The pseudocode implementation of the above algorithm is shown in Appendix D.2. Although this approach addresses only how agents arrive at a consensus direction at once, it can be applied to track a constantly changing target by running the above algorithm periodically. While its correctness is tedious to prove formally, this algorithm is supported empirically by its success in thousands of trials (Section 6.3).

6.2.5 Collective Locomotion: Movement Consensus

Here, I present the algorithm that allows agents to achieve coordinated movement using the distributed homeostasis algorithm described in Chapter 3 after agreeing on a common traveling direction.

I first consider the following two-agent locomotion case. Consider a "passive" edge shared by two agents (i.e., one perpendicular to the direction of locomotion) that does not move during the locomotion cycle. For the best coordination and minimum of interference between the two agents, it is clear that the cycles should be coordinated so that one agent pulls this edge forward while the other pushes it in the same direction (Fig. 6.12 (a)).

Thus, the two agents' locomotion phases need to be offset by π (with the front agent being ahead of the back agent) if the agents are connected along their traveling direction. I can modify the module actuation function (Eqs. 6.1 and 6.2) by replacing θ with $\theta + \phi_i$, where ϕ_i is agent i's phase variable:

$$\forall i \in \Gamma, x_{i,d}(t) = f((\theta + \phi_i) \bmod 2\pi) \quad (6.4)$$

$$x_{i,d'}(t) = f((\theta + \phi_i - \pi/2) \bmod 2\pi) \quad (6.5)$$

I define ϕ^*_{ij} as the desired phase offset between agent i and its neighbor j. If two agents i and j are connected along their traveling direction (Fig. 6.12 (a)), then

Algorithm 1 Pseudocode for agents to achieve coordinated movement. Initially, each agent i has a random ϕ_i and starts by identifying the desired phase offsets ϕ^*_{ij} between it and and its neighbors. Each agent then iteratively modifies its phase ϕ_i until all ϕ^*_{ij} are satisfied. In line 11, α is a damping factor and needs to be set as: $0 < \alpha < \frac{1}{\max_i |N_i|}$.

```
// Each agent i initializes desired phase offsets with neighbors
for all j ∈ N_i do
  if agent j is in front of i in the traveling direction then
    φ*_ij = π
  else if agent j is at back of i then
    φ*_ij = −π
  else
    φ*_ij = 0
// Each agent i iteratively communicates with its neighbors and updates its phase φ_i
loop
  φ_i ← φ_i + α · Σ_{j∈N_i} (φ_j − φ_i − φ*_ij)
```

$\phi^*_{ij} = \phi_j - \phi_i = \pi$. Agents that are connected perpendicular to their traveling direction should maintain synchronized locomotion phases ($\phi^*_{ij} = 0$). Fig. 6.12 (b) shows an example of the desired relationships between a single agent and its neighbors.

The challenge is for agents to coordinate to set ϕ_i such that all $\phi_j - \phi_i = \phi^*_{ij}$ are satisfied, when *each agent has only a local view and its choice of ϕ_i can potentially affect all other agents*. In fact, one can utilize the distributed homeostasis algorithm (line 11 in Algorithm 1) to solve this challenge. Let us denote the set of agent i's neighbors as N_i. Algorithm 1 shows a fully decentralized approach for each agent to iteratively modify ϕ_i based on feedback from neighbors, such that all ϕ^*_{ij} are eventually satisfied.

At every time step, each agent computes the differences between the current and desired phase offsets to its neighbors and acts proportionally to the summation of all such differences. By establishing a connection with the analysis in Chapter 4, convergence for Algorithm 1 can be guaranteed.

Theorem 10 (Convergence)**.** *Let ϕ_i be the phase of agent i and ϕ^*_{ij} be the desired phase offset between i and its neighbor j. If agent i updates ϕ_i based on Algorithm 1 and the communication graph G is connected, ϕ_i will eventually converge to a value such that $\phi_j - \phi_i = \phi^*_{ij}$ for all agents i and $j \in N_i$.*

Proof. see D.1. □

The movement coordination algorithm presented here not only applies to the

"traveling wave" locomotion strategy that I describe in this section but also can potentially be generalized to other coordinated locomotion patterns formulated as inter-agent relationships.

6.3 Collective Locomotion Experimental Results

In this section, I present various experimental results from applying this framework in both real and simulated robots:

- The efficiency of the proposed locomotion strategy is tested one and two hardware robot agents. In the two-agent case, I also compare locomotion efficiency with and without coordinated phase offsets.
- I use physics-based simulators to test the proposed coordinated locomotion algorithm in several complex and irregular assemblies, evaluating: (a) how coordination scheme affects locomotion efficiency; (b) how shape of the robot affects locomotion speed; (c) how locomotion efficiency changes with number of agents.
- I test the convergence speed of both direction consensus and agent movement coordination algorithms as the number of agents increases.

Experimental results show that this approach allows agents of various different connectivities to achieve efficient locomotion in both simulated and real robots. In addition, the agent movement coordination algorithm is scalable to the number of agents in the system. When more agents are aggregated together, locomotion efficiency in aggregated agents is as or more efficient than individual agents.

6.3.1 Real Robot Experiments

Here I use the hardware prototype I previously described in Section 6.1. The length of each module is 16 cm when it is fully contracted. I set linear actuators' speed to their maximal speed 2.3 cm/s. The traveling speed is used as a metric to evaluate agents' locomotion efficiency. To define a measure that is invariant to the scale of the agents, I define speed as percentage of an agent's body length that the robot can travel in each locomotion cycle. To evaluate the effectiveness of the algorithm, I first assemble two agents together to compare their locomotion capability with and without phase coordination. The agents start with random desired traveling directions, and they reach agreement on a common direction in two communication cycles. After agents decide on their phases, they start locomoting in the agreed traveling direction.

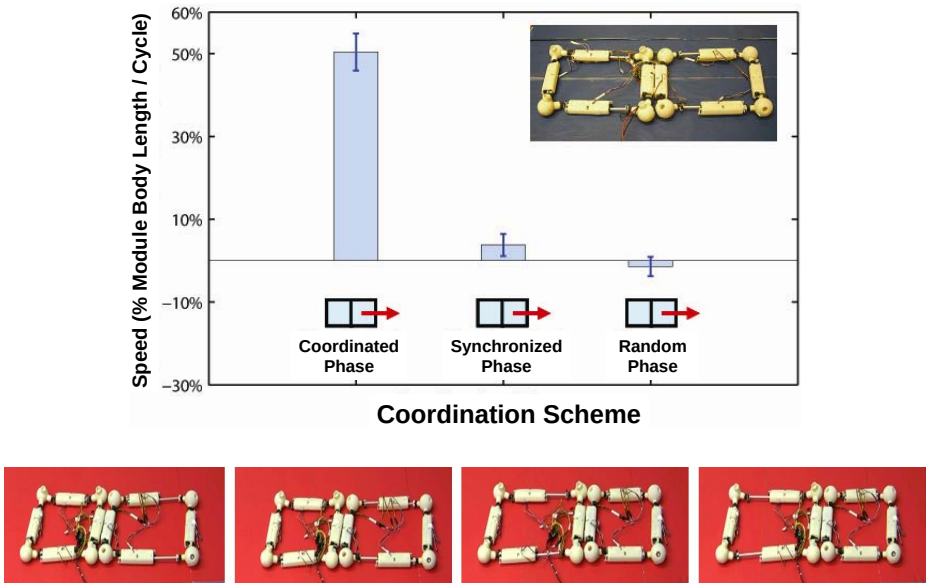


Figure 6.13: Hardware experiments: Comparison of robot locomotion speed with different agent phase coordination schemes. Locomotion speed is significantly greater when agents achieve phase coordination with Algorithm 1 (left). When two agents have synchronized phases or maintain random phases, the agents can barely move. (Bottom) A sequence of steps in a locomotion cycle with coordinated phase offsets for a two-agent robot.

Figure 6.13 shows a comparison of different phase offsets between agents. The agents' locomotion speed is averaged over 12 locomotion cycles, and the error bars show standard deviation of the speed among cycles. When agents use Algorithm 1 to compute their phase offsets, they can locomote effectively (left bar). When agents have the same phases or maintain random phase offsets (the data is averaged from 6 random phases), the agents show very limited locomotion abilities, even though each executes an actuation cycle that would let it move effectively if on its own.

I also evaluate speed of locomotion with one and two agents and in different traveling directions. Fig. 6.14 shows that a single agent's locomotion speed has higher variability (error bars) compared to two-agent configurations. The intuition behind this is that an additional agent can help the robot stabilize its movement. We can also see that if two agents are attached along their traveling direction, the robot's locomotion speed is also higher than the speed of travel in the orthogonal direction.

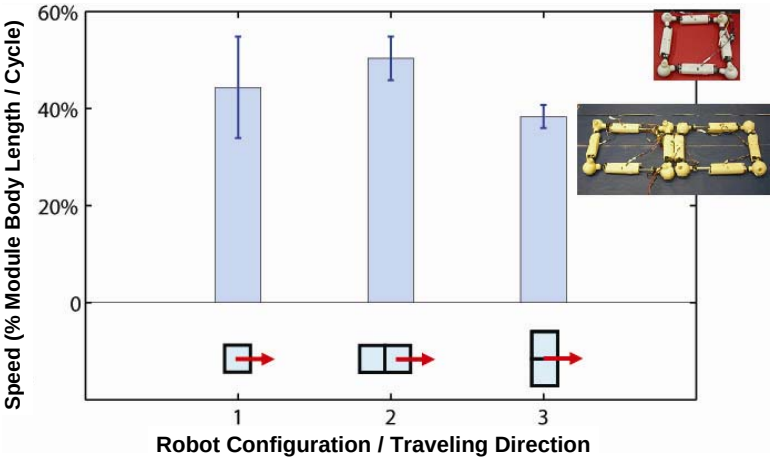


Figure 6.14: Hardware experiments: The robot's locomotion speed becomes more consistent if a second agent is attached to the first (reduced variance, left vs middle and right bars). We can also see from this result that an additional agent attached along the traveling direction can also help to increase the robot's locomotion speed (middle vs right bar).

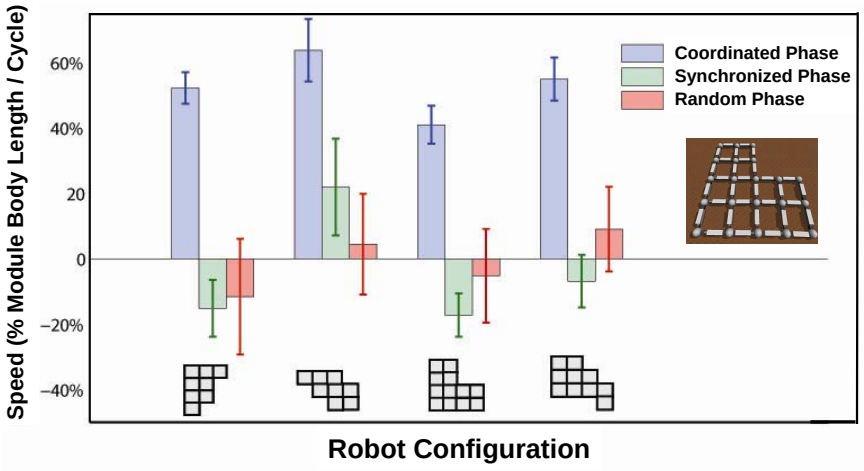


Figure 6.15: Comparison between different ways of setting phase offsets between agents. If phase offsets are computed based on Algorithm 1 (coordinated phase), it leads to much more efficient locomotion for the robot. On the other hand, the robot can move very limited distances with synchronized and random phase offsets.

6.3.2 Arbitrary Connectivities

With the ODE simulator, I investigate agents' locomotion efficiency in a wide range of robot configurations. Again, let's compare agents' locomotion efficiency with and without the movement coordination (Fig. 6.15). For coordinated and synchronized phase experiments, results show mean and standard deviation from 40 locomotion cycles. For random phase experiments, the graph shows results from 5 different random initializations and 20 locomotion cycles in each case (100 cycles in total). The most efficient locomotion results when all agents compute their phases based on Algorithm 1 ("traveling wave"). This is consistently true for different irregular agent connectivities.

From Fig. 6.15, we can also see that the robot's moving speed varies with different shapes. This inspired us to further explore how shape might potentially affect the robot's locomotion. Fig. 6.16 shows how varying robot shape and traveling direction affects locomotion efficiency. In these experiments, when front rows of the robot have more agents than back rows, the robot travels faster (bars 1 and 3). This suggests that a robot's locomotion speed may improve when its center of mass is closer to the front end.

6.3.3 Efficiency vs Module Aggregation

The independent agents that make up an aggregate robot have the potential to act in conflict, interfering with their collective locomotion. I thus next investigate how the attainable traveling speed is affected as the number of agents increases. Ideally, one would like for self-mobile agents not to lose speed when they join a larger robot.

Fig. 6.17 shows the speed of robots comprising one to four agents with coordinated phases, for both real and simulated robots. Simulations tested varying levels of ground friction to explore different terrain conditions.

In all cases, the speed of robots with two or more agents was at least as great as that of a single agent. Simulations showed that under some conditions, adding a second agent significantly increased the speed of the robot; even in the high-variance cases where the increase was not significant, the data suggest an overall increase rather than decrease. The mechanism appears to involve the fact that some components can slide backwards during parts of the movement cycle; the second agent provides increased stability and reduces sliding away from the travel direction. Adding agents beyond the second had no significant further effect on speed.

While Algorithm 1 leads to effective locomotion, there may well exist locomotion patterns that produce still greater speed, potentially attainable only with larger assemblies or with heterogeneity among agent controllers. An interesting future di-

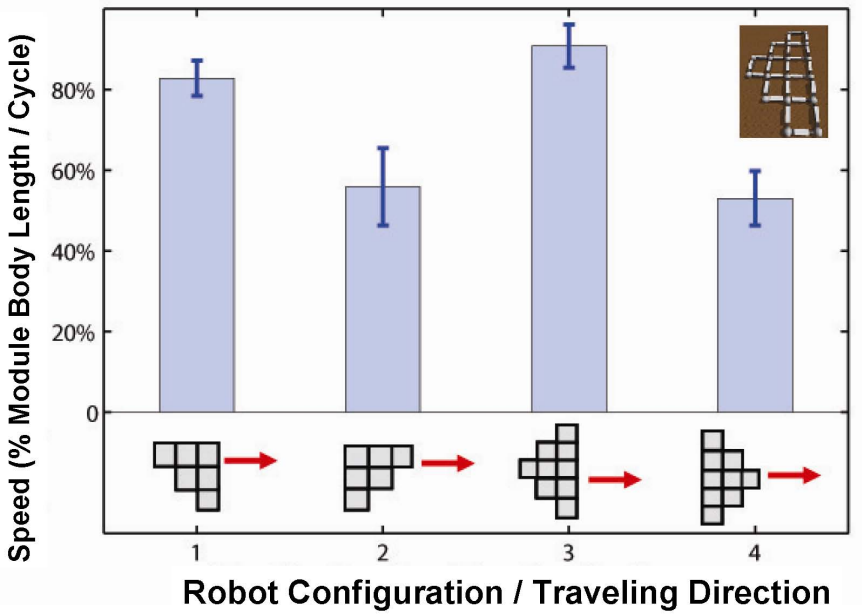


Figure 6.16: Different shapes and traveling directions vs speed: If the robot's center of mass is at the front, locomotion is faster (first and third bars).

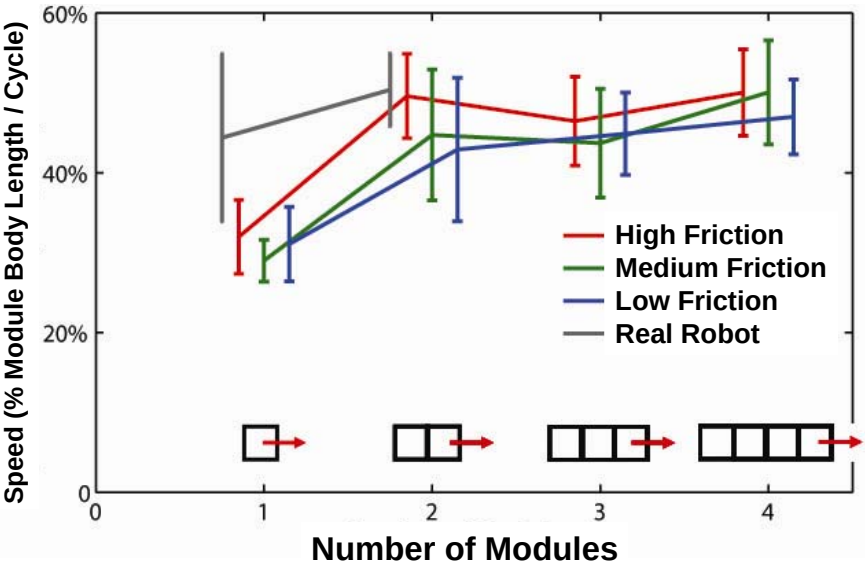


Figure 6.17: Number of agents vs. robot speed. Adding a second agent to the first can increase net travel speed; adding further agents has no significant effects.

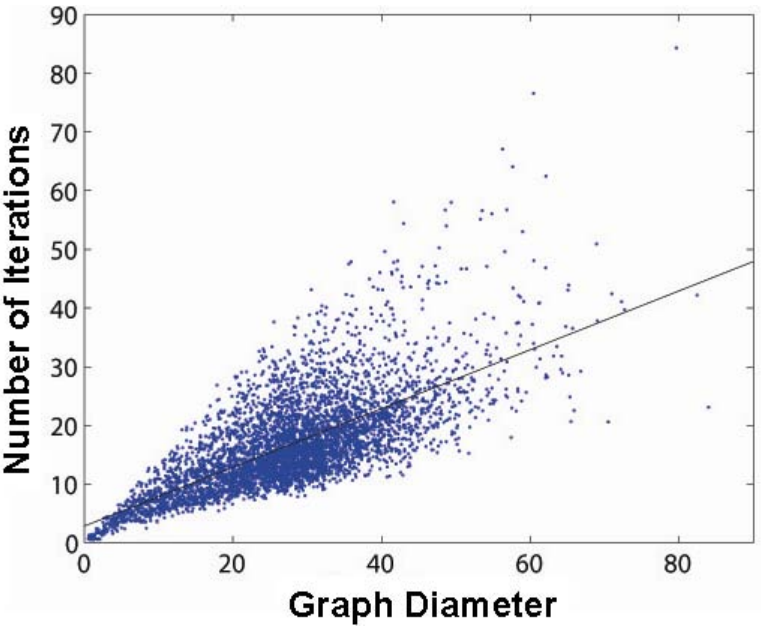


Figure 6.18: Number of iterations n required for randomly generated robot to reach global agreement on direction, as a function of diameter d of its connectivity graph. Results show 5000 trials using asynchronous updating, with best-fit line minimizing squared error $n = 0.50d + 2.8$.

rection is to compute an optimal locomotion pattern for any given geometry.

6.3.4 Scalability Experiments

Here I describe how the direction agreement and movement coordination algorithms of Section 6.2.4 and Section 6.2.5 scale with the size of the system.

Consensus direction: Fig. 6.18 shows the number of iterations required for randomly generated, randomly initialized robots to come to global agreement on direction using an implementation in Appendix D.2 of the algorithm outlined in Section 6.2.4, with asynchronous updating. Tie-breaking random values were chosen to be small compared to the discretization of confidence values.

In 5000 out of 5000 trials, the robot reached global agreement on the direction associated with the highest non-outlier confidence. The typical number of iterations n was proportional to the graph diameter d, with best-fit line $n = 0.50d + 2.8$. When agents execute the algorithm synchronously, convergence to the appropriate direction is still reliably correct but somewhat slower ($n = 0.86d + 2.5$), since unlike the asynchronous case values can propagate at most one agent per time step.

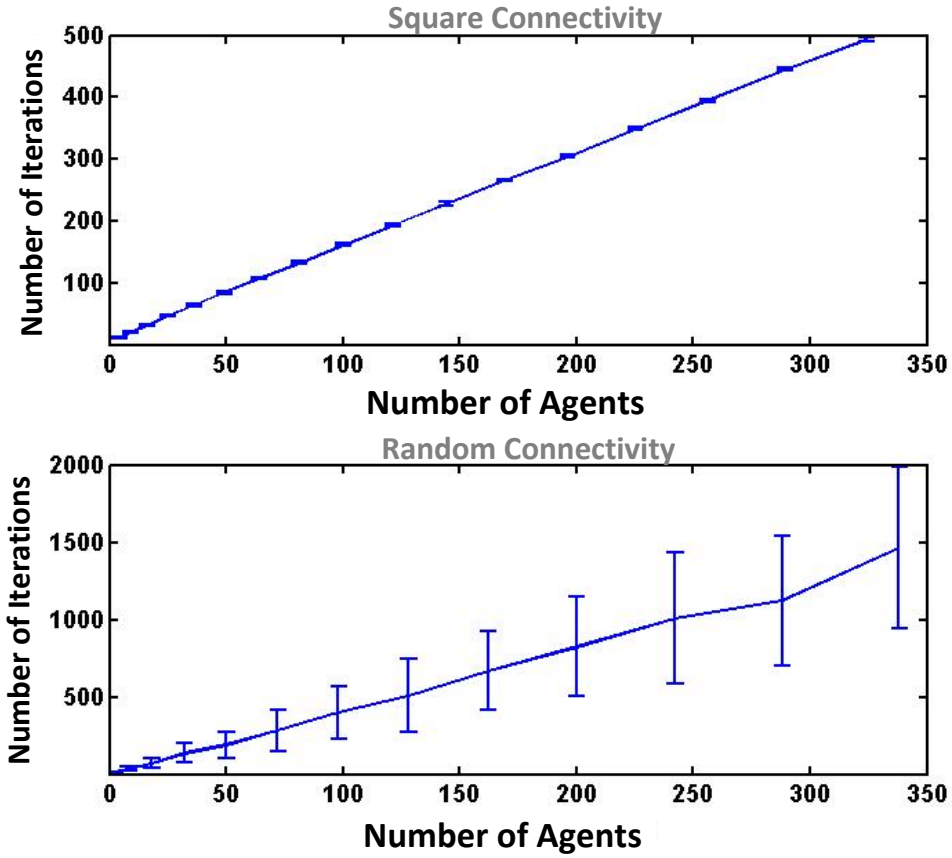


Figure 6.19: Convergence speed vs. number of agents. The number of agents is sequentially increased in both square (top) and random (bottom) connectivities. Each point is computed from 100 random initializations. The average number of iterations required for convergence is linearly proportional to the number of agents.

Movement coordination: Let us define $\epsilon(t) = \sum_i \sum_{j \in N_i} |\phi_{ij}(t) - \phi^*_{ij}|$ as a measure of convergence to coordinated movements, where $\phi_{ij}(t)$ is the phase offset between agents i and j at time t. The number of iterations required for convergence is defined as $t_{\min} = \min_t \frac{\epsilon(t)}{\epsilon(0)} < 5\%$. Fig. 6.19 shows averaged $t_{\min}$ versus number of agents in configurations with square connectivity (100 different initializations) and random connectivity (100 different connectivities and initializations). We can see $t_{\min}$ is linearly proportional to size of the system. Further, square connectivities on average require fewer iterations than random connectivities. Since square connectivities generally have smaller diameters than random connectivities, these results coincide with theories presented in Chapter 4.

6.4 Terrain-Adaptive Locomotion on a Modular Tetrahedral Robot

In the previous two sections, we have seen how one can model an agent's locomotion pattern as a discrete CPG function and use the distributed homeostasis algorithm to allow agents to reach a phase consensus. The tasks I described so far (e.g., forming self-adaptive structures tasks in Chapter 5 and the collective locomotion tasks described in this chapter) all belong to the category of achieving *a single adaptive task*. In this section, I show *how agents can achieve more complicated tasks by performing a sequence of self-adaptive tasks*. In particular, I show an example of rolling motion by a modular tetrahedral robot. Each module in the tetrahedral robot can sense pressure and light, and modules can cooperatively control the whole structure's direction of traveling and adapt the gait cycle to the terrain.

I first show that a tetrahedral robotic structure formed by modules can be viewed as a multiagent network. By formulating the modular tetrahedral robot's locomotion task as a sequence of self-adaptive tasks, I demonstrate that the robot is capable of performing rolling-type locomotion with terrain adaptation capabilities. This approach can be potentially applied to many other robot locomotion tasks that exploit the robots' special structure as a way of achieving adaptive locomotion; example robotic systems include SUPERBOT [61] and ChemBot [66].

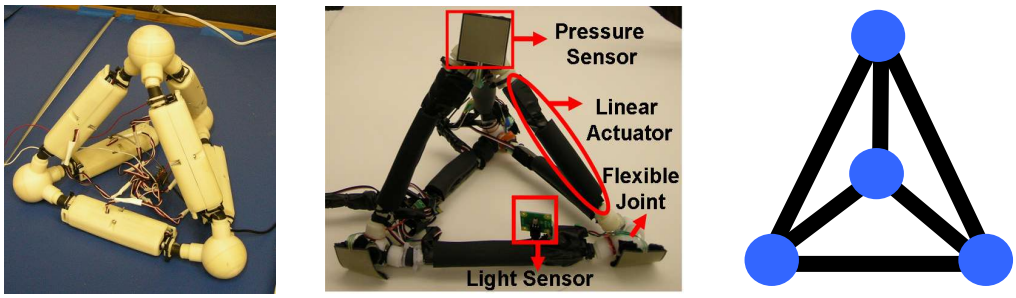
6.4.1 Tetrahedral Modular Robot Model

The modular tetrahedral robot is a multiagent system where we view each *spherical joint* (node) as an agent if it is equipped with a pressure sensor and if it is capable of controlling the actuators that are connected to it. I also establish a communication link (along the module between two spherical joints) between each pair of neighboring agents. One can further represent such a robot as an agent graph, as shown in Fig. 6.20.

6.4.2 Locomotion as a Sequence of Self-Adaptations

Here, I describe the algorithmic procedure for adaptive locomotion. The concept diagram for this locomotion strategy is shown in Fig. 6.21. At each locomotion step, a subset of agents on the surface with highest light-sensor reading is selected to achieve a self-adaptive task (i.e., achieving (nearly) equal pressure readings by actuating and rolling the whole structure towards that surface). This process repeats until the robot has reached the final destination.

Before we start describing the algorithmic procedure, I define several notations



(a) (b) (c)

Figure 6.20: (a - b) Two modular tetrahedral robots used in implementing the proposed algorithm. In these systems, each spherical joint is viewed as an agent, and each module serves as a communication link between two agents. (c) Each robot can be viewed as a networked multiagent system and can be represented as a graph with vertices indicate agents and lines indicate communication links.

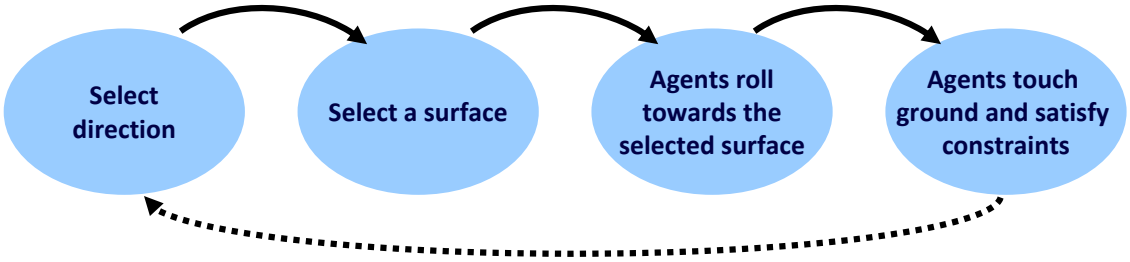


Figure 6.21: The overview of tetrahedral modular robot locomotion. The robot goes through the cycle of choosing a direction, selecting a surface, rolling towards the surface, completing a cycle, and repeat.

related to agent indices. I assume that there is a linear actuator mounted between two agents and denote x_{ij} as the linear actuator mounted between agent a_i and a_j. If there is more than one actuator mounted between a_i and a_j (e.g., the module shown in Fig. 6.3 has two actuators), x_{ij} is used to denote the length of all actuators. In the example in Fig. 6.22, agent a_1 can control actuators $x_{1,2}$, $x_{1,3}$, and $x_{1,4}$. I use a light source as the robot's traveling destination. In addition, a light sensor is mounted on each surface of the tetrahedron (Step 2 of Fig. 6.22), and agents on the surface can access the sensor readings.

The detailed algorithmic steps are as follows:

- **Step 1: (Initialization)** Each agent starts passing messages to its neighbors, allowing it to identify its neighboring agents and the linear actuators between them.

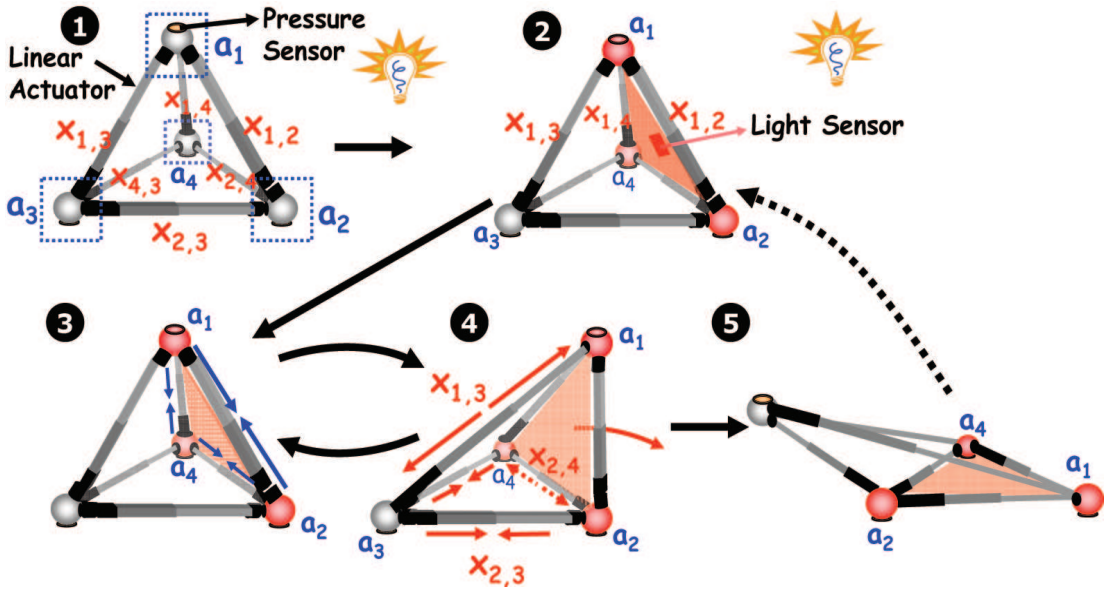


Figure 6.22: A single step of the tetrahedral robot locomotion task. Step 2: one of the surfaces is triggered by light. Step 3: the activated agents exchange sensor readings with their neighbors. Step 4: Agents control actuators to let the triggered surface lean forward until agents reach the pressure consensus state. Step 5: The pressure consensus state has been reached, the robot goes back to the default configuration, and a new surface is triggered.

- **Step 2: (Choose a Direction)** The surface that is closest to the light source is triggered[2]. I denote the subset of agents on the triggered surface as Ω. In the example of Fig. 6.22, $\Omega = \{a_1, a_2, a_4\}$.
- **Step 3: (Sensing)** The activated agents start sending pressure readings to their activated neighbors.
- **Step 4: (Actuation)** I denote linear actuators that are on the triggered surface as surface actuators, and those attached to the triggered surface are denoted as supporting actuators (e.g., agent 1 in Step 2 of Fig. 6.22 has surface actuators x_{12} and x_{13} and supporting actuator x_{14}). In this step, each agent actuates the supporting linear actuator (the linear actuator that it is connected to but not on the triggered surface) by running the following control law:

[2]In my implementation, I set a threshold $\bar{\theta}$ to determine whether a surface is triggered. In a tetrahedral structure, only one surface is nearly perpendicular to the light direction, so only one surface will be triggered. In some other structures where ambiguities might arise, a maximum value determination algorithm described in Section 6.2.3) can be run to identify which surface is to be triggered.

$$x_{ik}(t+1) = x_{ik}(t) + \alpha \cdot \sum_{a_j \in N_i \cap a_j \in \Omega} (\theta_j - \theta_i) \tag{6.6}$$

where x_{ik} is a_i's supporting actuator.

This control law will allow the activated surface to lean forward until the tetrahedron rolls over to put all three activated agents in contact with the ground. In the Fig. 6.22 example, this procedure is achieved with $x_{2,3}$ and $x_{4,3}$'s contraction and $x_{1,3}$'s expansion (Step 4 of Fig. 6.22)[3]. Agents iterate between Steps 3 and 4 until they converge to the consensus state.

- **Step 5: (Cycle Complete)** The self-adaptive task is achieved when all activated agents have contacted the ground and $\|\theta_j - \theta_i\| \leq \epsilon$ for all agents a_i and their neighbors a_j[4]. After agents have achieved consensus, they reset to the default configuration (Step 2); a new surface is then triggered.

The verification of sufficient conditions for completing each locomotion step with the control law Eq. 6.6 is similar to that of Theorem 9 in Chapter 5. To avoid repetition, I omit the details here.

The generalization of single self-adaptive task to a sequence of such tasks allows this framework to be extended from solving static shape/structure adaptations to dynamic tasks like locomotion. Utilizing agents' satisfying sensory reading constraints provides a way for modular robots to adapt to different environmental conditions.

In the tetrahedral robot example, the cycle time of locomotion is determined by the pressure states of the agents. This approach allows the robot to adapt its locomotion cycle based on different terrain slopes. When the environmental condition allows agents to reach a consensus state sooner (e.g., when the robot is on a steeper slope), the robot then exploits gravity; its locomotion cycle time will adapt to become shorter.

---

[3]In my hardware implementation, all actuators are fully contracted in the default state; thus, $x_{2,3}$ and $x_{4,3}$ are not able to further contract. Alternatively, I can program agents in contact with the ground (a_2 and a_4) to actuate the surface actuator between them ($x_{2,4}$). The modified control law is then:

$$x_{il}(t+1) = x_{il}(t) - \alpha \cdot \sum_{a_j \in N_i \cap a_j \in \Omega} (\theta_j - \theta_i)$$

where a_i and a_l are agents that have contacted the ground. I note that this control law will allow surface actuator x_{il} to expand, achieving the same effect as contracting two supporting linear actuators.

[4]In this application, a larger ϵ is allowed to identify whether a consensus has been reached.

6.4.3 Modular Tetrahedral Robot Experimental Results

I also implement the locomotion algorithm as described in Section 6.4.2 on tetrahedral robot hardware prototypes (shown in Fig. 6.20). As shown in the previous Fig. 6.20 (b), each agent is equipped with a pressure sensor and each surface a light sensor. The total height of the tetrahedron is ~ 20 cm.

Figure 6.23 shows two sequences of the robot's locomotion actions on a 10-degree declining slope (top) and on flat ground (bottom). As shown in Fig. 6.23, agents on the surface that is closest to the light source are activated in each cycle. The average locomotion cycle time is 5 sec, and the robot is capable of moving towards the light source at a speed of 10 cm/sec.

I also constructed a simulation environment using Open Dynamics Engine to examine how the tetrahedral robot performs terrain-adaptive locomotion on different slopes. When the robot is placed on a declining slope of -54 degrees (Fig. 6.24 (b)), the locomotion cycle time becomes much shorter: its average cycle time is 48% of the average cycle time when it locomotes on an $+18$-degree inclining slope (Fig. 6.24 (a)). The robot travels in a more efficient way when it can exploit gravity to assist locomotion.

6.5 Generalization to Modular Robot Climbing Tasks

Robot vertical climbing tasks generally require special adhesive mechanisms on robot feet [31]. On the other hand, animals can usually exploit environments, e.g., rocks, to climb when adhesive mechanisms are not available to them. Here, I describe how this idea can be achieved by combining the pressure-adaptive control task described in Chapter 5 and the locomotion function described in the previous section.

In this vertical tunnel climbing task, modules execute the same locomotion algorithm as described in Section 6.2 to climb vertically. Meanwhile, a pressure sensor and computation units are also present on each spherical joint. To stabilize the robot in the tunnel, each spherical joint acts as an independent agent and executes a control law similar to that of the pressure-adaptive column described in Section 5.4. Each spherical joint agent is programmed to achieve the same pressure as its neighbor while maintaining at least θ_p force applied to the tunnel (if the joint is not contracting or extending to make the whole robot move).

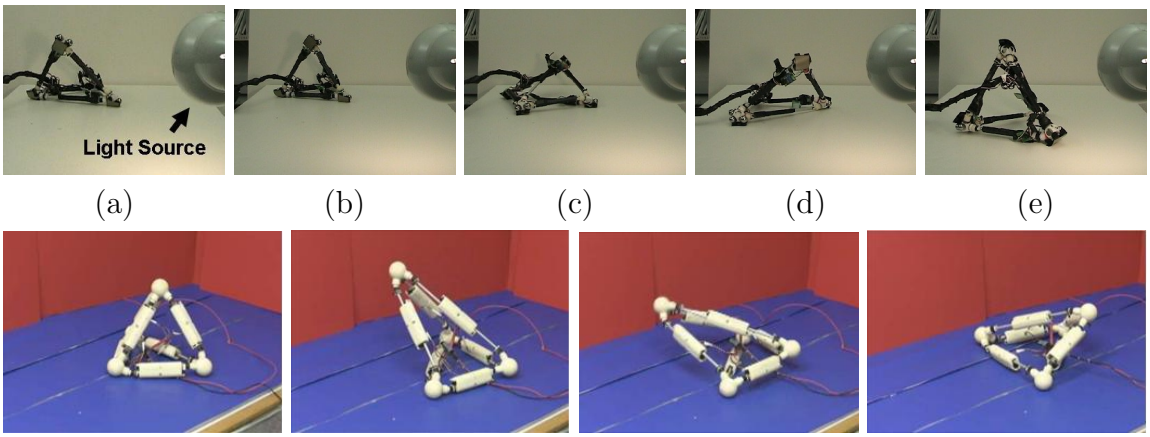


Figure 6.23: Modular tetrahedral robot adaptive locomotion. (a-e) Agents coordinate with each other to allow the robot to locomote toward light source on a 10 degrees declining slope. (f-i) A sequence of steps that allow the robot to complete a full rolling locomotion cycle. Agents coordinate to let the robot travel on a flat terrain.

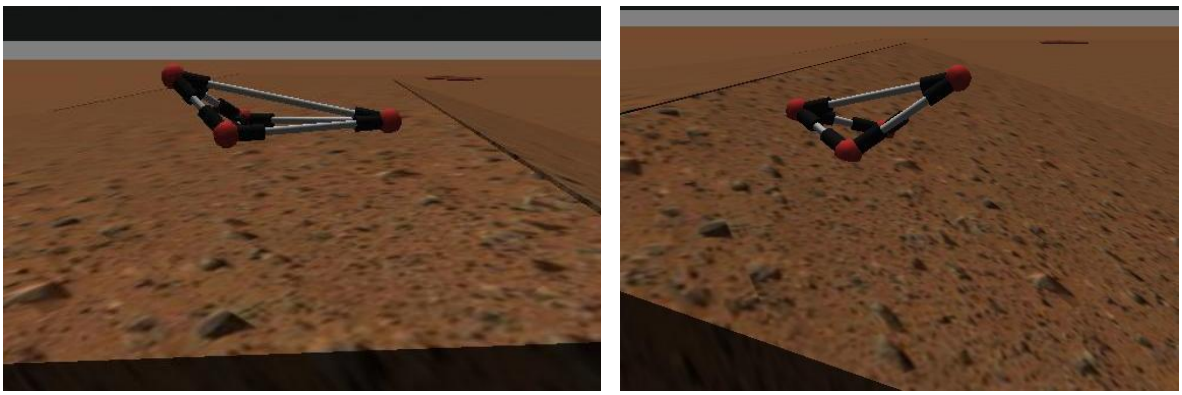


Figure 6.24: Modular tetrahedral robot locomotion simulation. The robot's locomotion cycle adaptively becomes ~ 2 times faster when it traverse on a declining slope (b) than on a inclining slope (a).

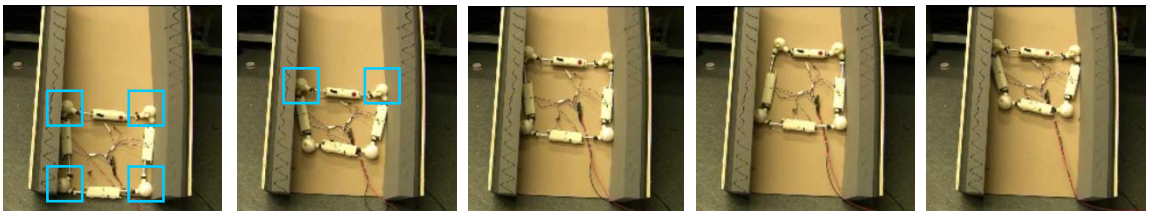


Figure 6.25: A sequence of movements of the modular robot climbing task. Blue boxes indicate the agents that are executing pressure-adaptive tasks to push against the wall and maintain robot's stability. The robot is able to climb up at a speed of 0.22 cm/sec.

Figure 6.25 shows a sequence of movement of the modular robot climbing on a vertical wall. Blue boxes indicate the spherical agents that are executing pressure-adaptive tasks to maintain the stability of the structure. The robot is able to climb up at a speed of 0.22 cm/sec.

6.6 Summary

In this chapter, I have presented how this framework can be generalized to solve two adaptive locomotion tasks. I first demonstrated the distributed coordination schemes through which the modules of an amorphous modular robot can agree on a travel direction and synchronize their movement to achieve effective group locomotion. No central controller, global knowledge, or a priori known connectivity is required. I then showed how a modular tetrahedral robot's locomotion task can be modeled as a sequence of pressure-adaptive tasks. This approach can potentially be generalized to many robot rolling-type locomotion tasks. I have investigated the correctness and effectiveness of these locomotion schemes analytically and experimentally both in simulations and with hardware.

Cellular slime molds, which spend part of their life locomoting as individual cells and part locomoting as a multicellular slug, serve as the biological inspiration for this adaptive locomotion task [7]. For future applications, I envision developing a decentralized robotic system of self-mobile agents that can act independently (e.g., for effective coverage of a large area), join together into an aggregate robot at need (e.g., for more effective movement), and perform an adaptive locomotion strategy based on different environment conditions (e.g., for terrain-adaptive locomotion). Such a system would require the capabilities I have discussed in this chapter, and the work presented in this chapter is a first step toward that larger goal.

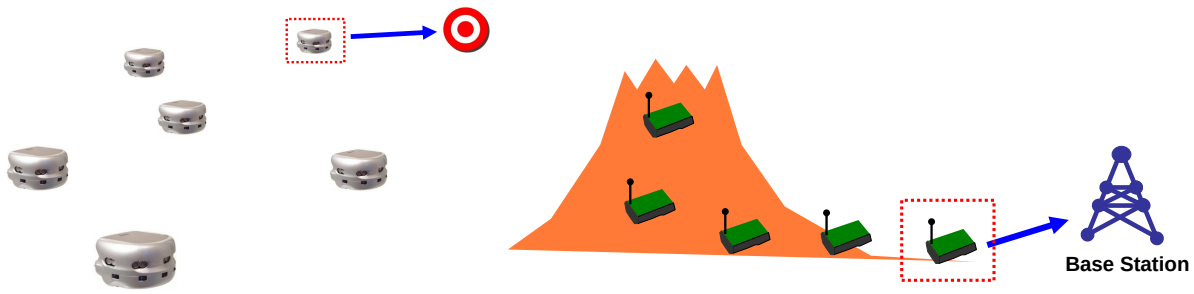
Chapter 7

Multiagent Decision-Making by Implicit Leaderships

In the previous chapters, I consider situations where all agents play *equal* important roles in the collective tasks. Here, I show a simple generalization of the agent control law that will allow us to address decision-making scenarios where *some agents obtain privileged information and need to play a more important role in the group task*. Such task scenarios are particularly common when the agent group is distributed in space. For example, in a multi-robot search task, robots need to agree on a common traveling direction. Robots close to a target may be able to detect or track it better than those farther away (Fig. 7.1 (a)). In sensor network time synchronization tasks, sensor nodes are required to reach a consensus on a common timer. Nodes closer to the base station can provide more precise timing information than those far away (Fig. 7.1 (b)).

Solving consensus decision-making tasks can be particularly challenging in large-scale agent systems where agents are spatially distributed in a wide space. One approach that can incorporate this privileged information is to designate a centralized coordinator to collect information from all agents and then disseminate a decision to the whole group. However, such a hierarchical approach interferes with the system's scalability and robustness: the coordinator can easily become a communication bottleneck and may also be a potential point of failure for the system.

In decentralized animal groups, recent biological studies have shown that a few individuals without privileged roles can guide the entire group to collective consensus on matters like travel direction through simple local interactions. Inspired by these findings, I propose an *implicit leadership algorithm* based on simple and decentralized interactions. This algorithm also generalizes the agent control law presented in Chapter 3 to cases where agents have different confidence levels in their preferred states.



(a) (b)

Figure 7.1: Example scenarios where some agents obtain important group information and need to play more important roles. (a) In a team of mobile robots, robots close to the group target can track it better than those farther away. (b) In a sensor network, sensor nodes closer to the base station can provide more precise timing information than those far away

In this chapter, I start by presenting the biological inspiration for, and overview of, this study. I then describe the agent model and compare this model with the standard agent model presented in Chapter 3. In Section 7.3, I present an implicit leadership algorithm and *prove* that this generalized control law reliably allows all agents to agree on a decision that can be determined by one or a few better-informed agents through purely local interactions. In addition, I also analytically show how the number of informed agents and their confidence levels affects the consensus process. In Section 7.4, I address cases where informed agents (implicit leaders) share a common goal or have conflicting goals, and I further present an extension that allows for fast decision-making in a rapidly changing environment in Section 7.5. Finally, I show how the framework can be applied to a diverse variety of applications, including mobile robot exploration, sensor network clock synchronization, and shape formation in modular robots, in Section 7.6.

7.1 Biological Inspirations and Approach Overview

This approach is inspired by recent biological studies on how natural systems arrive at a consensus decision efficiently by simple and decentralized control laws (example shown in Fig. 7.2). Couzin et al. have proposed a model for animals in groups, such as migrating herds, in which only a few informed individuals influence the whole group's direction of travel [12]. In this model, each informed agent simply decides its movement direction based on the observed directions of its neighbors and its preferred direction. Informed individuals are able to transmit their preferred direction to the whole group. Ward et al. report nonlinear decision mechanisms in which fish

Figure 7.2: In animal groups, such as fish schools, a few informed or experienced individuals can potentially guide the whole group to travel in a correct direction through simple local interactions. (Image source: Harvard SSR group)

choose movement directions based on the actions of a quorum of their neighbors, some of whom may have privileged information [68]. These studies provide examples of effective, decentralized decision-making under conditions of information asymmetry, in which there is *no* explicit announcement of leaders or elaborate communication; each individual simply modifies its behavior based on that of its neighbors.

Although these models are simple and effective, one disadvantage of these studies from an engineering standpoint is that they are purely empirical, without guarantees regarding the observed behavior or analytic treatment of the conditions under which the results will hold. Designing effective controllers for artificial systems requires the further investigation of several open questions.

- How do we prove that the informed agents correctly influence the rest of the group?
- Can we characterize factors related to the speed with which the group can reach a decision?
- How do we resolve cases involving conflicting goals?

Here I propose an implicit leadership framework by which distributed agents can reach a group decision efficiently. I address *the above theoretical challenges and generalize the approach to more complex scenarios.* In this approach, each agent's control algorithm is a weighted combination of the average states of its neighbors within its

communication range [1] and its sensed goal state. This approach is that of distributed consensus algorithms, with a critical departure: while traditional distributed consensus studies make the assumption that all agents have equally valid information, here I allow agents to be informed to varying degrees. This generalization allows this self-organizing approaches to be applied to a wider range of scenarios. I characterize the algorithm's theoretical properties and provide several important extensions:

- I prove that the algorithm indeed leads to the desired group behavior: a subset of informed agents with the same goal can correctly lead the whole group to adopt that goal, achieving a consensus decision.
- I characterize how factors such as the fraction of informed agents in the group and confidence levels of informed agents affect the speed with which the group arrives at a decision.
- I generalize the algorithm to the case where informed agents have conflicting goals, by having them modify their confidence levels according to the states of their neighbors (*implicit leadership reinforcement*).
- Inspired by the quorum response model in [68], I further extend the framework to improve group performance and convergence speed in rapidly changing environments (*fast decision propagation*).
- Finally, I demonstrate how the framework can be applied to a diverse variety of distributed systems applications, including multi-robot exploration, sensor network clock synchronization, and adaptive shape formation in modular robots.

7.2 Multiagent Model

The multiagent model considered here is similar to that of Chapter 3, except that this model also captures each agent's different importance in the group task. The agent model comprises a set of independent agents a_i, each of which can control certain of its state variables x_i (e.g., heading direction in Fig. 7.3), and determine the states of neighbors within some fixed range. This determination may be passive, via sensing, or active, via communication.

Some agents have privileged knowledge, e.g., due to better sensing capabilities or a location close to an object of interest. These informed agents have a goal state x_i^* that represents its believed goal that the whole group should reach, and a confidence factor w_i that represents its confidence on this belief (Fig. 7.3 (a)).

[1] The local average part of the control algorithm can also be substituted with the right hand side of Eq. 3.2

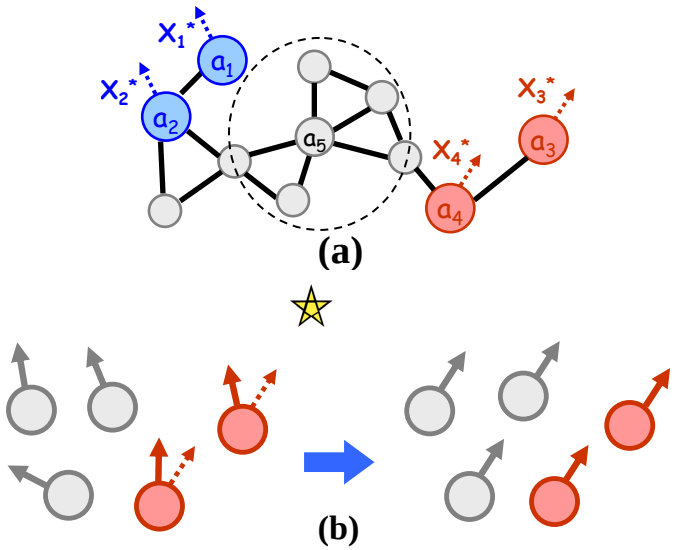


Figure 7.3: (a) A node indicates an agent and an edge indicates a neighborhood link. The dotted line indicates agent a_5's sensor/communication range. Informed agents(colored) have their own goal states (arrows). (b) Agents start with random headings and try to make a consensus decision.Solid/dotted arrows for informed agents represent current/goal directions, respectively.

Neighborhood relationships among agents can be described by a connectivity graph (Fig. 7.3 (a)). Nodes represent agents, and edges indicate that agents are within sensor or communication range. This relationship is assumed to be symmetric; hence the graph is undirected. Since agents can move, the graph can potentially be time-varying. This scenario is referred as a *dynamic topology*. If the graph remains static, I refer to it as a *static topology*.

Throughout the following sections, I use *collective pursuit* in multi-robot systems as a motivating example: there exists some moving target whose heading direction the whole group needs to match (Fig. 7.3 (b)). Such a task may arise in, e.g., patrol scenarios (finding and pursuing an intruder), exploration (where the group moves in the direction of a feature of interest), or human-robot interaction (where a user needs to guide a swarm to a desired location). In this example, an agent's (robot's) direction is its relevant state variable.

In Section 7.6, I discuss applying this approach to a wide variety of tasks.

7.3 Implicit Leadership Algorithm

Here I present the algorithm that achieves collective decision-making in the mul-

Algorithm 2 Equation for state update with implicit leadership. All agents execute this update at each time step. Informed agents ($w_i > 0$) influence the rest of the group without explicit commands. The two terms represent behaviors (a) and (b) as described in the text.

loop

$$x_i(t+1) \leftarrow \underbrace{\frac{1}{1+w_i} \cdot x_{\text{avg}}(t)}_{\text{Behavior (a)}} + \underbrace{\frac{w_i}{1+w_i} \cdot x_i^*}_{\text{Behavior (b)}}$$

tiagent system.

Inspired by Couzin et al. [12], the proposed agent control law incorporates two components: (a) a tendency to align its state with those of its neighbors, and (for informed agents) (b) an attempt to achieve its goal state. In fact, part (a) of the control law is similar to Eq. 3.2, and the control law can be viewed as an extension of my original proposed control law (with an extension of part (b)).

At each time step, each agent computes its new state as a function of its old state and the states of its neighbors. The control algorithm can be formally written as in Algorithm 2. In that equation, a_i indicates agent i and $x_{\text{avg}}(t)$ is simply the average of the states of agent a_i and its neighbors:

$$x_{\text{avg}}(t) = \frac{x_i(t) + \sum_{j|a_j \in \mathcal{N}_i(t)} x_j(t)}{|\mathcal{N}_i(t)| + 1} \tag{7.1}$$

where $\mathcal{N}_i(t)$ is the set of a_i's neighbors at time step t. The factor $w_i/(1+w_i)$ captures the degree to which agent a_i *drives itself to the goal state* x_i^*, and $1/(1+w_i)$ the degree to which it *follows the local average*. If a_i is an informed agent, its confidence w_i is positive; otherwise $w_i = 0$. In the rest of this section, I assume that w_i and x_i^* are constant over time. One can also replace Eq. 7.1 with distributed consensus equation (Eq. 3.2):

$$x_{\text{avg}}(t) = x_i(t) + \alpha \sum_{a_j \in N_i} (x_j(t) - x_i(t))$$

and the analysis is similar to that of Eq. 7.1.

All agents execute the simple rule of Algorithm 2, with no need for informed agents to take specific actions to coordinate others or communicate its goal state. For instance, when a robot team starts with randomly initialized heading directions and only some robots have knowledge of a unique global goal direction, the latter set w_i to be positive to reflect their confidence and x_i^* to match that goal. *All* agents repeatedly update their heading direction $x_i(t)$ according to Algorithm 2, and the *informed agents iteratively influence others to reach the desired consensus.* Each agent knows it has reached a stable decision state when the maximal state difference between it and its neighbors is less than a small number.

7.3.1 Algorithm Analysis

Here I analytically prove properties of Algorithm 2's correctness and performance. In this section, I only present theorems, and defer all proofs to the Appendix E.

First, in the absence of informed agents ($w_i = 0 \, \forall a_i$), this control formulation then becomes identical to that of *distributed consensus* [25, 48, 89]. If the communication graph is connected or periodically-connected[2], all agents will converge to the same state: $\lim_{t\to\infty} x_i(t) = \bar{x} = \frac{1}{N}\sum_i x_i(0) \, \forall i$, where N is the number of agents. Distributed consensus analyses have been applied to many areas, e.g., sensor network synchronization, and the theoretical properties have been widely explored [48].

Next, let's consider how the decision of the group is influenced when there are informed agents all with the same goal state ($x_i^* = x^* \, \forall i \text{ such that } w_i > 0$). More specifically, we ask:

- Will all agents converge to x^*?
- How does convergence speed relate to the confidence factors w_i and the number of informed agents?
- Will agents converge to the same state in both dynamic and static topology cases?

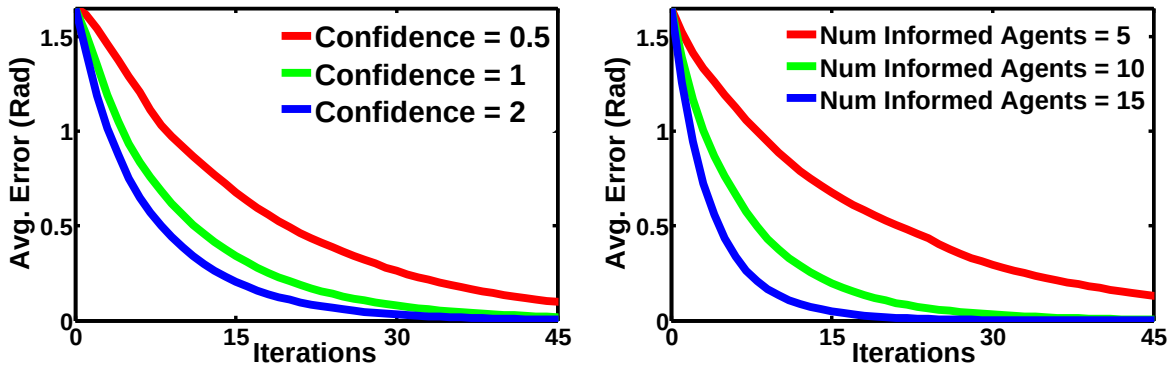
I first introduce two definitions: (1) Informed agent group: A set of informed agents $\mathcal{L}_k$ with the same goal state $\bar{x}_k^*$. (2) Informed agent-connected: A condition for dynamic topologies in which a path always exists between each non-informed agent and some informed agent, and this path persists without modification for $d_{\max} + 2$ time steps. Here $d_{\max}$ is the maximum among all non-informed agents of the shortest hop distance to an informed agent.

Theorem 11. *(Convergence) Let there be only one informed agent group with goal state x^*, with all agents executing control Algorithm 2. Let the communication topology be either (1) static and connected, or (2) dynamic and informed agent-connected. Then*

$$\lim_{t\to\infty} x_i(t) = x^* \, \forall i$$

Proof. See Appendix E.1. □

[2]A graph with dynamic topology is periodically-connected if, for all pairs of nodes $\{a_i, a_j\}$, some path P between them exists and persists without modification for a number of time steps $\geq$ length$(P) + 1$.



(a) (b)

Figure 7.4: Average error in heading direction within the group over time, for (a) different confidence levels and (b) number of informed agents. The y-axis gives agents' average distance from the goal state. Convergence speed increases when the confidence and/or number of informed agents increases. Each data point is averaged from 50 random initializations.

Intuitively, Theorem 11 tells us that if the agent topology is either connected or informed agent-connected, all agents will eventually be able to follow a single informed agent group.

Next let's consider how the group size and confidence weights of informed agents can affect the convergence rate.

Theorem 12. *Under the same conditions as Theorem 11, the system's convergence rate increases if more agents become informed, or if existing informed agents increase their confidence levels, in both connected (static) and informed agent-connected (dynamic) topologies.*

Proof. See Appendix E.2. □

7.3.2 Experimental Results

I also explore these relationships in simulation. The simulation model is composed of 50 agents that are initially randomly distributed within a 10×10 area (arbitrary units). They can freely travel in 2D space without boundaries. Each agent moves a distance of 0.05 per iteration and has a communication radius of 4. An agent's state is characterized by its heading direction $x_i \in [0, 2\pi)$. All agents begin in random states, and informed agents are given a common goal direction. The evaluation metric is defined as agents' average distance from the goal state. Fig. 7.4 shows how the convergence speed changes when (a) all informed agents' confidence weights are

increased (I fix the number of informed agents at 10), or (b) more agents are selected as informed agents (informed agents' confidence is fixed at 2). The convergence speed to the informed agents' goal direction increases in both cases.

Finally let's consider the case where informed agents do not all share the same goal state—i.e., multiple informed agent groups.

Theorem 13. *(Multiple Informed Agent Groups) Let there be multiple informed agent groups $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_q$ with different goal states $\bar{x}_1^*, \bar{x}_2^*, \cdots, \bar{x}_q^*$. Let agent topology be static and connected. Then each agent's state will converge to some value, but this value can be different for different agents.*

Proof. See Appendix E.3. □

Consider an example with agents arranged in a line, where the two agents on the ends are informed and have opposite goal directions. The agents between them will then converge to states in between the goal states of the two informed agents. This convergence holds for static topologies, but not for dynamic topologies; in the latter the agents may never converge to a fixed state[3].

Given these results, if a single group consensus is to be reached and there are initially multiple informed agent groups, it is necessary for them to become a single group with a common goal state. In the following, I describe how to extend Algorithm 2 to achieve this.

7.4 Implicit Leadership Reinforcement (ILR) Algorithm

In many tasks, it is necessary for a single informed agent group to emerge. For example, in a multi-robot search and rescue task, informed robots at different locations may observe different clues and thus have different opinions about where rescue subjects are located, yet the whole group may eventually need to agree on a single rescue target to secure enough robot force to complete its task. Here I show one can achieve this aim by replacing the constant w_i of informed agents with a time-varying variable $w_i(t)$. I call the mechanism that decides how $w_i(t)$ should change *implicit leadership reinforcement* (ILR).

Algorithm 3 describes ILR. The idea is that informed agents have their confidence increase if the states of a sufficiently large fraction $\bar{\rho}$ of their neighbors are sufficiently

[3]To see this, suppose that each agent's state has converged to a fixed value, and that the topology then changes such that a non-informed agent a_i goes from having an informed neighbor in $\mathcal{L}_j$ to instead having one in $\mathcal{L}_k (j \neq k)$. Then a_i's state will clearly change.

Algorithm 3 Implicit leadership reinforcement procedure. Running this update along with that of Algorithm 2 lets informed agents change their confidence so that the entire group can reach a single consensus. The function $1\{C\}$ evaluates to 1 if condition C is true and 0 otherwise (line 3). Agents execute this reinforcement routine along with the standard implicit leadership algorithm (Algorithm 2). I set $\delta = 0.6$, $\bar{\rho} = 0.4$, $\bar{t} = 5$, and $\alpha = 0.5$.

loop
  if a_i is an informed agent **then**
    if $\sum_{j|a_j \in \mathcal{N}_i} \frac{1\{\|x_j(t) - x_i^*\| \leq \delta\}}{|\mathcal{N}_i|} > \bar{\rho}$ **then**
      $w_i(t+1) = w_i(t) + \alpha$
    else
      $w_i(t+1) = \max(w_i(t) - \alpha, 0)$
    if w_i has been 0 for $\bar{t}$ time steps **then**
      a_i becomes non-informed ($w_i = 0$ permanently)

close to their own goal states (within δ); otherwise the confidence decreases, and if it reaches 0 for long enough then the agent loses its "informed" status.

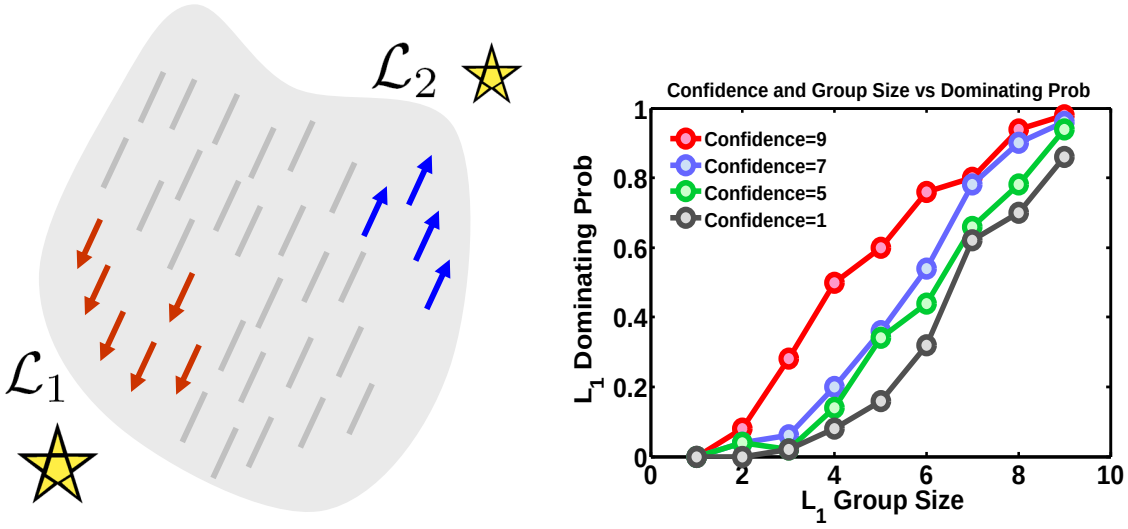
This approach has the following advantages:

- While each agent has only a local view, the result that emerges nevertheless captures *global statistics*. The informed agent group with a larger population or higher confidence dominates the decision with a high probability.
- At any time, agents can join or leave an informed agent group or change confidence while observing more instances, and the system will accommodate these updates without the process needing to restart.
- Only a few iterations are required for a single informed agent group to emerge.

An informed agent group is called a *dominating group* if at some point it contains all remaining informed agents—i.e., all agents in other groups have become non-informed. If this occurs, then all agent states will converge to the dominating group's goal state so long as the connection graph is connected or informed agent-connected (Theorem 11).

7.4.1 Experimental Results

Here I investigate ILR's performance experimentally. I use the same simulation setup as in Section 7.3.1. Initially, two informed agent groups $\mathcal{L}_1$ and $\mathcal{L}_2$ are constructed by choosing a total of ten informed agents forming two spatial clusters (Fig.



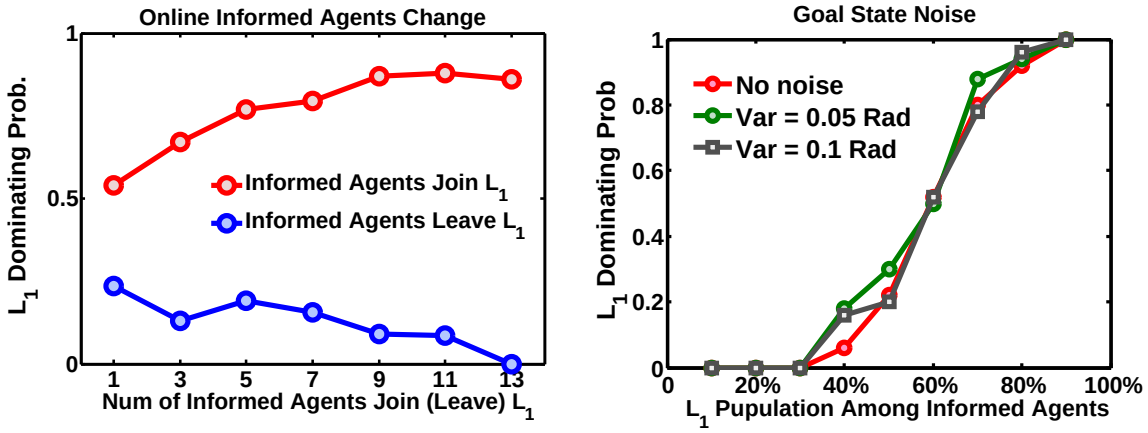
(a) (b)

Figure 7.5: (a) $\mathcal{L}_1$ and $\mathcal{L}_2$ observe different targets and have distinct goal states. (b) When informed agents in $\mathcal{L}_1$ increase their confidence (different curves) or the size of $\mathcal{L}_1$ increases (x-axis), the probability that $\mathcal{L}_1$ dominates increases (i.e., ultimately becomes the only remaining informed agent group).

7.5 (a)). The two groups are given distinct goal headings, set in opposite directions. In 1800 experiments, the average number of iterations required for a dominating group to be established is 21. After a further 20 iterations on average, all agents arrive at a consensus heading direction with an average error within 0.05 rad of the informed agents' goal.

Informed agent group size and confidence: Here I investigate the probability that $\mathcal{L}_1$ becomes a dominating group, as a function of initial group size and confidence level. Fig. 7.5 (b) shows the results of varying the number of agents in $\mathcal{L}_1$ from 1 to 9 ($|\mathcal{L}_2|$ correspondingly varies as $9 \rightarrow 1$), and of varying the confidence of the agents in $\mathcal{L}_1$ from 1 to 9 (weights of agents in $|\mathcal{L}_2|$ correspondingly change from 9 to 1). Each data point represents 50 randomly initialized trials. Increasing the number or confidence of informed agents in a group generally yields a higher dominating probability.

Online changes in informed agents: During the consensus process, some initially non-informed agents might have the opportunity to make new observations that convert them into informed agents. Conversely, initially informed agents might lose their informed status, e.g., if they lose track of the group goal. I thus consider the case where the size of $\mathcal{L}_1$ changes while the ILR algorithm is running. Initially,



(a) (b)

Figure 7.6: (a) When more agents join $\mathcal{L}_1$ during the course of a run, $\mathcal{L}_1$'s dominating probability increases; similarly, the probability decreases when more agents leave $\mathcal{L}_1$. (b) The probability that $\mathcal{L}_1$ dominates is not appreciably affected by slight variation among the goal states of agents within the group.

the number of agents in each group is set as $|\mathcal{L}_1| = |\mathcal{L}_2| = 14$, and agents' confidence factors are set as $w_1 = w_2 = 5$. An agent is randomly selected to leave or join $\mathcal{L}_1$ every five time steps. The total number of selected agents to join or leave the group varies from 1 to 13. Fig. 7.6 (a) shows that a group's probability of dominating (averaged from 200 trials) increases as more agents join the group, and decreases as more agents leave.

Goal variability within groups: In some cases, informed agents that are properly considered part of the same group may have slightly different goal states. For instance, noisy observations or perception difference may prevent different agents observing the same target from setting identical goals. Because ILR makes no reference to group identity, such variability need not interfere with the algorithm. Experiments that add zero-mean Gaussian noise to each informed agent's goal state demonstrate that a group's probability of dominating is not significantly affected by moderate variability of this sort (Fig. 7.6 (b)).

7.4.2 Other Reinforcement Conditions

The condition presented in line 3 of Algorithm 3, which evaluates whether a large enough fraction of an agent's neighbors have similar states, is experimentally effective for cases where informed agents are localized into spatial clusters. Other conditions may be more appropriate for other scenarios. For instance, my preliminary results

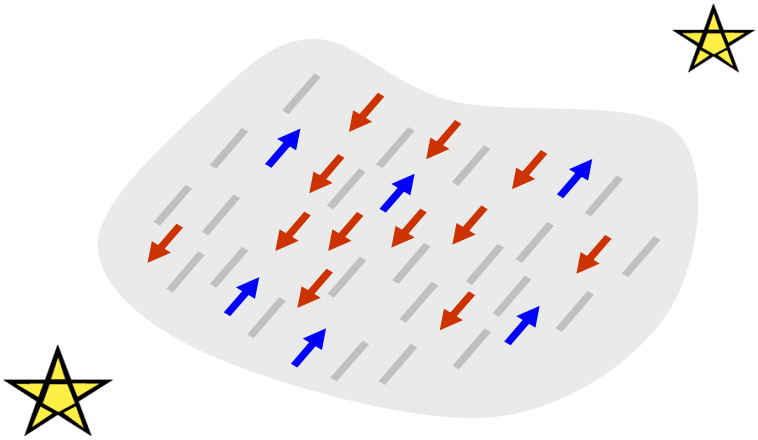


Figure 7.7: Example scenario when informed agents are spatially-mixed in space.

suggest that if informed agents are randomly distributed in space (as shown in Fig. 7.7), then the condition

$$\|x_i(t) - x_i^*\| \leq \delta$$

that is, informed agents increase their confidence if they are able to travel sufficiently close to their goal direction—results in more effective performance.

We also note that there is also the probability that multiple leader groups still exist or no leader group exists ($< 15\%$ in most cases). One solution to this is to have leaders update their weights based on new observations and run ILR until the whole group has reached a consensus decision.

7.5 Fast Decision Propagation

The control strategy I have presented so far requires agents to iteratively approach a group consensus decision. If the iteration time is long, consensus can be slow in coming, since tens of updates may be required. In nature, groups of animals can react very quickly to sudden environmental changes. For example, when a flock of starlings maneuvers to evade a predator's pursuit, starlings on the far side of the flock from the predator react swiftly even without direct perception of the threat. Inspired by a process by which a school of fish arrives at a quorum decision quickly [68], here I further extend this framework in a way that lets a group of agents reach consensus quickly for coping with rapid environmental changes.

The basic concept that enables fast propagation is that I allow non-informed agents to acquire informed status and preferred goal states through interactions with neighbors. In this way the size of the informed group increases. An uninformed

Algorithm 4 Pseudocode for agents to achieve fast decision propagation. $\mathcal{N}_i^*$ is the set of neighboring agents whose states differ from x_i by at least $\bar{\phi}$. r is a random number drawn from a uniform distribution between 0 and 1. Constants are chosen as: $k_1 = 4 \sim 6$, $k_2 = 2$, $\rho = 0.4$, and $\bar{\phi} = 0.4$. Agents execute this fast decision propagation routine along with the standard algorithm (Algorithm 2).

loop
  if $r \leq \frac{|\mathcal{N}_i^*|^{k_1}}{|\mathcal{N}_i^*|^{k_1}+|\mathcal{N}_i-\mathcal{N}_i^*|^{k_2}}$
  and $\max_{\{j,k\}\in\mathcal{N}_i^*} ||x_j - x_k|| < \rho$ **then**
    $x_i^*(t+1) = \frac{1}{|\mathcal{N}_i^*|}\sum_{j|a_j\in\mathcal{N}_i^*} x_j^*(t)$
    if a_i is non-informed **then**
      a_i becomes informed
      $w_i(t+1) = \frac{1}{|\mathcal{N}_i^*|}\sum_{j|a_j\in\mathcal{N}_i^*} w_j(t)$

agent changes its status to informed when it observes that a significant fraction of its neighbors have goal states different from its own, and that all of those neighbors have goals similar to each other. Already-informed agents can update their goal state x_i^* under the same conditions. Because the convergence speed increases with the number of informed agents (Theorem 12), This extension to result in faster consensus.

Algorithm 4 shows the fast decision propagation algorithm. I define $\mathcal{N}_i^*$ as the set of neighbors of a_i whose states differ from x_i by at least some amount $\bar{\phi}$. If all agents in $\mathcal{N}_i^*$ have similar states ($\max_{\{j,k\}\in\mathcal{N}_i^*} ||x_j - x_k|| < \rho$), x_i changes to be the average of the $\mathcal{N}_i^*$ agents' goal states with probability

$$P = \frac{|\mathcal{N}_i^*|^{k_1}}{|\mathcal{N}_i^*|^{k_1} + |\mathcal{N}_i - \mathcal{N}_i^*|^{k_2}}$$

If a_i was non-informed, it also becomes informed, setting its weight to be the average of the $\mathcal{N}_i^*$ agents' weights. The choice of expression for P is based on a study of how fish respond to their neighbors in quorum decision-making for travel direction [68]. In my implementation, variables are set as $k_1 = 4 \sim 6$ and $k_2 = 2$, which yields $P \sim 1$ when the fraction of $\mathcal{N}_i^*$ among a_i's neighbors is greater than 0.5. This formulation also prevents the propagation of a few agents' bad decisions: $P \sim 0$ when $\frac{|\mathcal{N}_i^*|}{|\mathcal{N}_i|} < 0.3$.

Experimental Results

I evaluate Algorithm 4 in a challenging tracking task using the same simulation framework as the other experiments reported above. A target of collective pursuit (Fig. 7.8) moves distance 0.05 per time step and rotates its heading direction counter-clockwise by a random amount between 0.3–0.9 radians every 20 steps. Agents within a distance of 5 can perceive the target, and accordingly act as informed agents. To

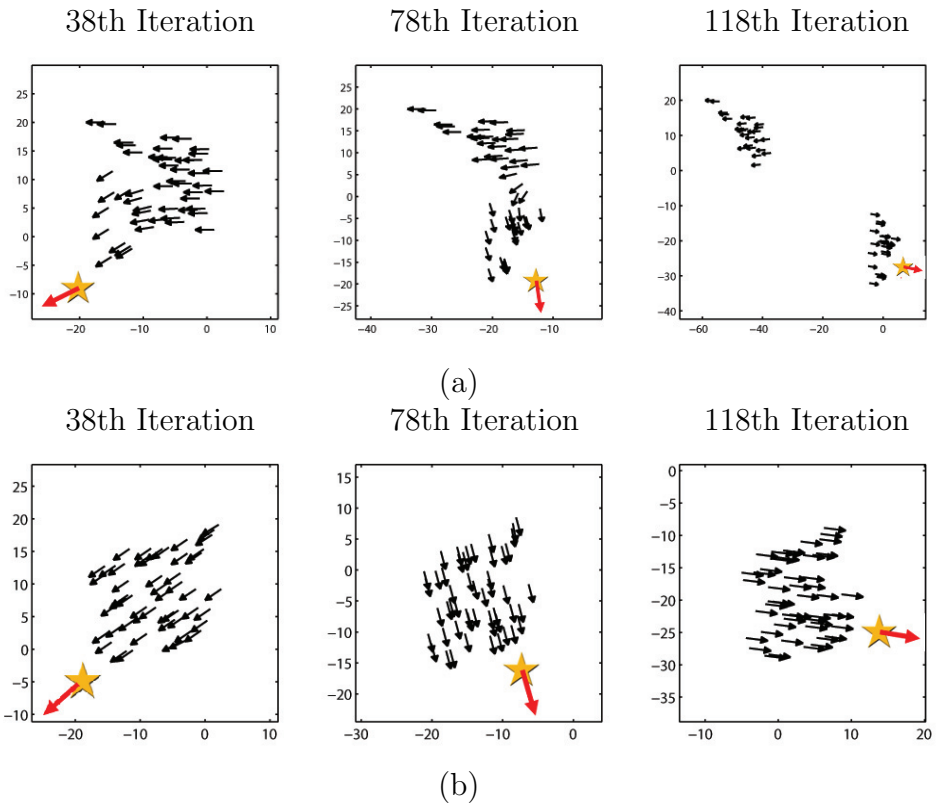


Figure 7.8: Snapshots of a multiagent system (black arrows) performing collective pursuit of a target (star, red arrow). The target changes its heading direction to a new random value every 20 time steps. (a) Agents use Algorithm 2 only. The system fragments into two separated groups after 78 time steps in this run. (b) Agents use Algorithms 2 and 4 (fast decision propagation). The group remains cohesive for 500 time steps, at which point the experiment ended.

effectively follow the target's rapidly changing trajectory, the whole group needs to propagate the information of the informed agents as fast as possible.

Experiments used 100 different target trajectories of 500 time steps, tested with each of the following two control laws. Agents programmed with the basic implicit leadership algorithm (Algorithm 2) alone fragmented into disconnected groups in *every* trial (75% fragmented in the first 100 steps), while groups of agents using fast decision propagation (Algorithms 2+4) remained connected until the end of the run in 31% of trials. Fig. 7.8 shows snapshots from one example trial. In (a), we can see that a large number of agents end up traveling in a different direction from that of the target, resulting in two spatially separated groups that cannot communicate with each other. By contrast, in (b), the fast decision propagation algorithm much more consistently lets the group reach consensus quickly enough to track the target's

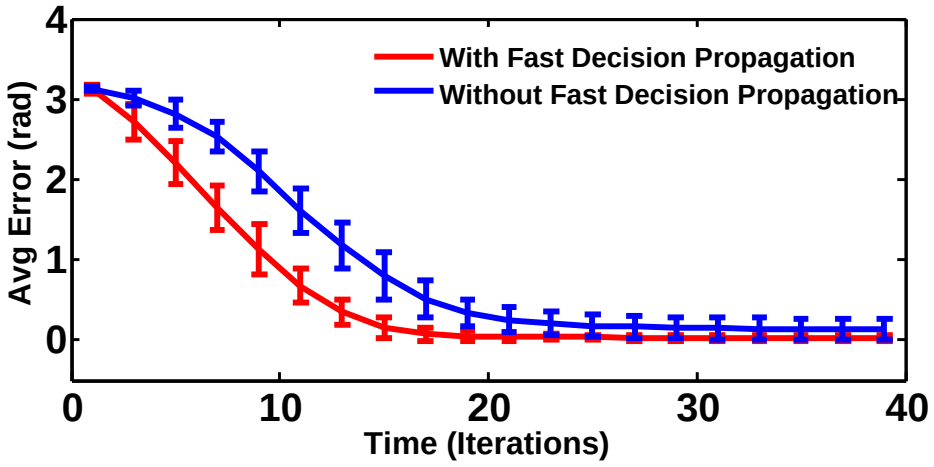


Figure 7.9: In a task in which all agents are initially aligned with a target that then instantaneously reverses direction, agents consistently reach consensus on the new direction more quickly when using fast decision propagation (Algorithms 2+4, red curve) than when using the implicit leadership algorithm (Algorithm 2, blue curve) alone. Averages are over 100 trials.

movement as a single cohesive unit.

I also test how fast the group can react to a sudden reversal of target direction. Fig. 7.9 shows the results of experiments from 100 trials in which all agents and the target are initialized with identical direction and random position, and the target's direction is instantaneously reversed by π. Systems using fast decision propagation consistently adjust to the new direction significantly faster than systems using the basic implicit leadership algorithm alone.

7.6 Applications

In this section, I demonstrate applying this framework to three diverse applications. (1) In a multi-robot exploration task, robots search for a target in a 2D space. Robots close enough to the target to localize it act as informed agents to allow the whole group to reach consensus on moving toward the target. (2) In a sensor network time synchronization task, sensor nodes close to a base station can access a global clock and thus better serve as informed agents. The whole group can then achieve more accurate clock settings than if a standard information-symmetric distributed consensus approach is used. (3) In a modular terrain-adaptive bridge task, robotic modules at the ends of a chain act as informed agents with their positions determined by the environment, and when the modules in between reach consensus, a straight

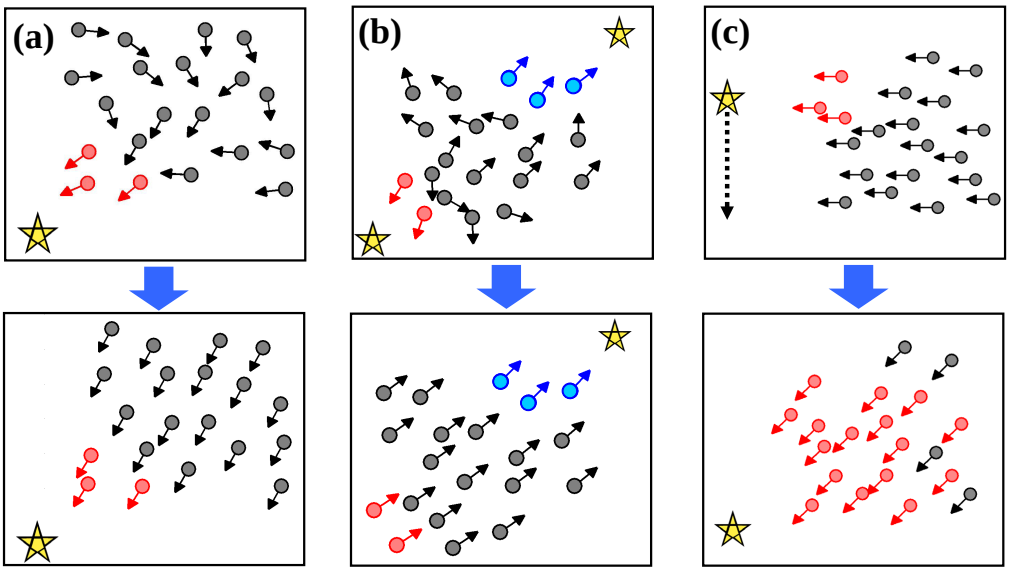


Figure 7.10: Multi-robot exploration tasks. (a) Red agents localize the target and the whole group iteratively achieves a collective decision. (b) Two groups simultaneously localize two different targets. After running ILR, the blue informed agents dominate the red informed agents and guide the whole group towards their target. (c) The target swiftly changes its direction. Fast decision propagation lets more agents become informed agents to allow the group to track the target quickly.

bridge is formed.

7.6.1 Multi-Robot Exploration Task

Here let's consider a team of mobile robots searching for a target. Each robot agent follows a movement direction $d_i(t)$ based on Reynolds's flocking rule [54]:

$$d_i(t) = \alpha_x \cdot x_i(t) + \alpha_c \cdot d_c(t) + \alpha_s \cdot d_s(t)$$

where x_i is the heading direction computed by the control algorithm Algorithm 2; $\alpha_x, \alpha_c, \alpha_s$ are constants; d_c and d_s are functions of the positions of neighboring agents, with d_c associated with maintaining a minimum distance to avoid collisions and d_s associated with attraction between more distant agents. At each time step, each agent communicates $x_i(t)$ to its neighbors.

Figure 7.10 (a) shows how agents explore the area with different heading directions (indicated as arrows) until several agents sense the target. The agents set their goal states as the direction pointing towards the target, and their confidence w_i to positive

values. All agents proceed to reach a group decision of moving toward the target. Fig. 7.10 (b) shows how ILR allows all agents to reach a single consensus when two different targets are located simultaneously. If the target is mobile, the fast decision propagation algorithm allows the group to track the target more effectively (Fig. 7.10 (c)).

7.6.2 Sensor Network Time Synchronization

In wireless sensor networks, maintaining synchronized clocks between nodes is important. To save power, sensor nodes are usually programmed to stay in sleep mode most of the time. Thus it is important for neighboring nodes to synchronize their timers so they can wake up at the same time and communicate with each other.

One strategy is to use a decentralized synchronization protocol inspired by coupled oscillator models in biology [38]. In this model, each node maintains a phase $\theta_i(t) = \omega t + \phi_i(t)$, where $\theta_i(t), \phi_i(t) \in [0, 2\pi]$ $(0 \equiv 2\pi)$ and ω is a constant related to frequency of oscillation. $\phi_i(t)$ is the phase shift variable, and each node's phase cycles periodically. When $\theta_i(t) = 2\pi$, the node wakes up and "fires". If all nodes can achieve identical $\phi_i(t)$, the network is synchronized. The control law in [38] is similar to that of distributed consensus, and each node constantly exchanges its phase shift variable with its neighbors. The discrete time step version of the protocol can be written as:

$$\phi_i(t+1) \leftarrow \phi_i(t) + \alpha \sum_{a_j \in \mathcal{N}_i} (\phi_j(t) - \phi_i(t))$$

where α is a small constant.

One major drawback of this approach is that it fails to take into account that some sensor nodes' timers are more accurate than others. In real-world sensor network deployment, some nodes close to a base station can access and synchronize to Coordinated Universal Time (UCT) via GPS. With this framework, one can set those nodes' desired phase shift to match the UCT timer, denoted as ϕ^*, and their confidence to be positive. The control law then becomes:

$$\phi_i(t+1) \leftarrow \frac{1}{1+w_i} \cdot \phi_{\text{avg}}(t) + \frac{w_i}{1+w_i} \cdot \phi^*$$

where $\phi_{\text{avg}}(t)$ is the average phase shift of agent i and its neighbors. Figure 7.11 shows an example scenario for this time synchronization task. The two leftmost nodes can access UCT and thus act as informed agents. The bottom figure shows the phase shift variables to which each node converges, which eventually match those of the informed agents $(\phi_i(t) = \phi^* = 0 \, \forall i)$.

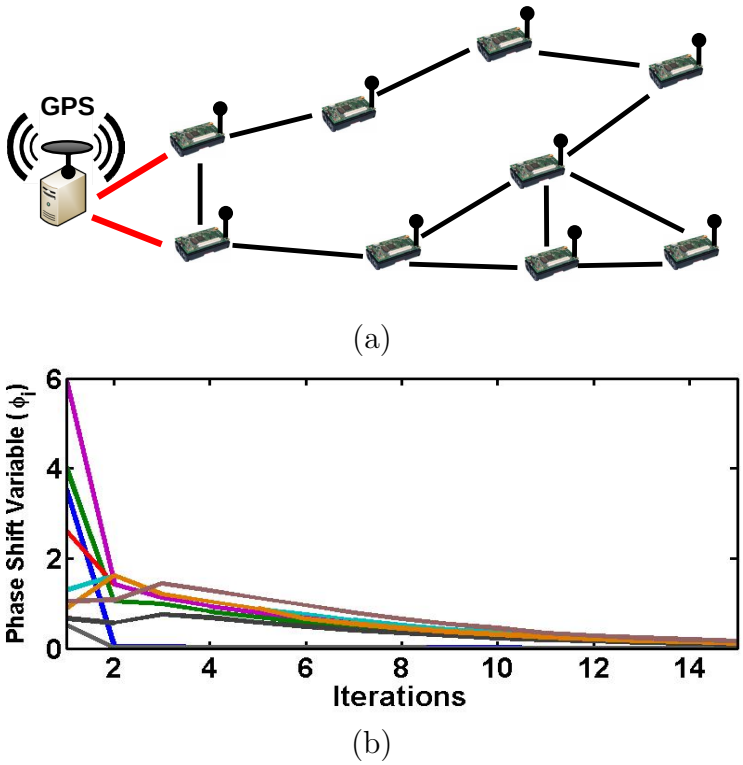


Figure 7.11: Sensor network time synchronization tasks. (a) Two sensor nodes that can access UCT are informed agents. (b) The whole group eventually achieves the same phase shifts as the informed agents ($\phi_i(t) \to \phi^* = 0\,\forall i$).

7.6.3 Modular Robot Shape Formation Task

In Chapter 5, I describe the design and control of modular robots that adapt to the environment. One example is a modular bridge, which is composed of actuated modular "legs" connected to each other by a flexible bridge surface. When the bridge is placed on an unknown terrain, each modular leg can sense the tilt angles of the local surface elements and adaptively change its height until a level surface is achieved. They show that a distributed consensus strategy can be used to solve this problem. One problem with the approach presented Chapter 5 is that while it allows all the modules to arrive at the same height, it cannot be applied to cases where a subset of agents need to maintain specific desired heights.

Using the approach presented here, one can easily interact with this system to incorporate desired heights; e.g., by simply fixing the height of one of the modules, the entire system will converge to agreement at that height if feasible. The control

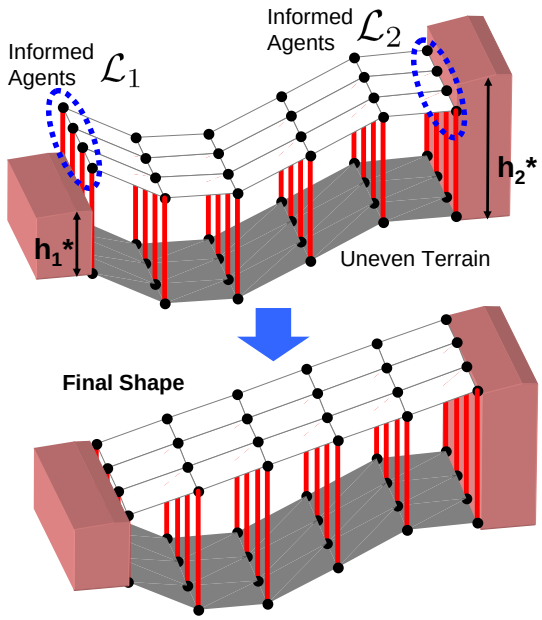


Figure 7.12: Modular robot shape formation task. Modules at the two ends form two informed agent groups ($\mathcal{L}_1$ and $\mathcal{L}_2$). Agents in $\mathcal{L}_1$ and $\mathcal{L}_2$ set their goal states to be the heights of the adjacent cliffs (h_1^* and h_2^*). The robot converges to a shape forming a straight slope (bottom).

law can be written:

$$h_i(t+1) \leftarrow \frac{1}{1+w_i}\left(h_i(t) + \alpha \sum_{a_j \in \mathcal{N}_i} \theta_{ij}\right) + \frac{w_i}{1+w_i} h_i^*$$

in where $h_i(t)$ and h_i^* represent current and desired heights of the agents (modules) and θ_{ij} represents the tilt sensor reading between agents i and j. By fixing the end agents at two different heights we can further create smooth inclines. Fig. 7.12 shows a simulation of the modular bridge where the agents at the two ends are fixed to the respective heights of the cliffs; the system converges to a smooth incline (Theorem 13). This type of implicit leadership provides a simple way for influencing and interacting with the modular robotic system and equally applies to the other applications presented in Chapter 5 such as the pressure-adaptive column and gripper.

7.7 Potential Algorithmic Extensions

The control rule presented in Algorithm 2 can be extended in various ways; implicit leadership reinforcement and fast decision propagation (Algorithms 3 and 4) can be seen as two such extensions. Future work may explore further variations, which additionally may be used in combination; e.g., while I considered Algorithms 3 and 4 separately above, both can be applied to the same multiagent system simultaneously. The preliminary experiments have indicated that systems using both extensions can achieve effective performance in both resolution of conflicting goals and rapid convergence to consensus.

7.8 Summary

In this chapter I have presented an implicit leadership algorithm based on a simple, purely local control law. This extension allows all agents to follow a single rule and efficiently reach a common group decision without complex coordination mechanisms. I have characterized the algorithm's theoretical properties both analytically and experimentally, and presented extensions that handle multiple conflicting goals among informed agents and achieve fast decision propagation. In a distributed multiagent system, it can be tedious to establish infrastructure and coordination mechanisms for agents to integrate unpredictably localized information to make collective decisions. The biologically inspired framework presented here provides a simple alternative that allows the group to reach consensus regarding distributed and dynamic information.

I have shown how this framework can be applied to several diverse application areas. Other applications may likewise benefit from incorporating this approach. Many existing control rules can be modified to incorporate a decision state that is observed or explicitly communicated between agents, as demonstrated with the example of Reynolds's rules for flocking above. The implicit leadership algorithm can also be seen as a useful tool that can be exploited in various multiagent coordination tasks, with the advantage that controllers for those tasks thereby acquire the associated convergence guarantees.

Chapter 8

Conclusions

8.1 Summary of Contributions

In this thesis, I have presented a biologically-inspired decentralized control framework for distributed multiagent systems to achieve collective tasks. I have also analyzed the theoretical properties of this approach such that one can ensure its correctness under connected agent topologies and predict the performance of the system while applying this framework. I further built several modular robots and demonstrate applicabilities of this framework with various autonomous tasks. I conclude by summarizing the contributions of this thesis by its algorithmic, theoretic, and application aspects.

Algorithm: I have proposed a self-organizing multiagent framework in which global tasks are specified by distributed constraint maintenance between local agents. I have also shown that this formulation can capture many multiagent *self-adaptive* tasks in which a large number of agents utilize their distributed sensors and actuators to solve tasks and cope with dynamic environmental changes. Exploiting this decentralized formulation, I have presented distributed homeostasis (DH) and generalized distributed consensus (GDC) algorithms (Chapter 3). The DH algorithm broaden the standard distributed consensus algorithm [48] in terms of task specification. The GDC algorithm extends the distributed consensus approach to a richer set of sensor-actuator systems.

To capture scenarios in which some agents obtain privileged information for group decisions, I extend this approach by incorporating an implicit leadership component (Chapter 7). This extension makes the approach effective for consensus decision-making among spatially-distributed agents, such as swarm robot systems and sensor networks. Finally, I demonstrate how this algorithmic approach can be adapted to solve various locomotion challenges by organizing them as sequences of self-adaptive

tasks. One important algorithmic contribution I make in this thesis is showing how one can generalize the simplest form of nearest-neighbor rules toward several variants to solve a much wider set of distributed agent tasks.

Theory: In the study of emergent behaviors from decentralized agents, it is important to address whether a correct global behavior will eventually result. I have theoretically proved the correctness of this approach and have shown that the parameters associated with agent topology (e.g., diameter and degrees of connection) are important indicators for the speed at which tasks are completed (Chapter 4). I also characterize how task complexity and external perturbations can potentially affect the system. My results show that this approach has a strong advantage over tree-based centralized approach when adapting to continuous perturbations and is thus well suited for tasks that require constant adjustments to dynamic environments. One advantage of large-scale decentralized agent systems is their robustness, and the theoretical analysis presented in this thesis also addresses how such agents would behave in the face of actuation and communication failures.

The theoretical results developed within this framework also contribute to one's understanding how to program this approach on various multiagent systems to operate within this framework to achieve self-adaptive tasks. One can potentially predict the multiagent system's performance when carrying out an assigned task based on agent topology, task complexity, and intensity of external perturbations. One can also reconfigure the agent system (e.g., changing the agent topology) based on this theoretical study to yield better performance.

Application: I have demonstrated how various modular robot applications can be formulated solved with this framework, examples include (1) self-adaptive structures that adapt to changing terrains (2) a modular gripper that can perform decentralized grasping; (3) modular robots that can perform dynamic locomotion with configuration and terrain adaptation capabilities. The applications shown in this thesis represent a small subset of what is achievable within this framework. Using unified controller design principles presented in Chapter 4, one can easily design control laws based on the GDC algorithm and verify correctness for various different agent systems.

There are many useful lessons that we have learned from these applications. First, our empirical results show that this approach is error-correcting and robust toward real world sensor and actuator noises. Second, various multiagent applications can be abstractly viewed as the same self-adaptive tasks, and they can be solved similarly within this framework. These tasks can range from statically-defined self-adaptive structures to the dynamics of modular robot locomotion. Finally, the theoretical results developed in this thesis provide accurate predictions of the performance of this approach in real world applications.

8.2 Future Work

There are many promising directions for future research and applications. Here, I outline several potential extensions.

Automated Task Compiler: One limitation of this approach is that the agent task must be specified in advance. Additionally, the agent tasks I have demonstrated in this thesis are limited to either single self-adaptive tasks or a sequence of similar self-adaptive tasks. One interesting direction would be the construction of an automated task compiler that can recognize the current world and autonomously construct a sequence of (potentially different) self-adaptive tasks for agents to achieve. With such a task compiler, one could potentially use this approach to solve a wider range of tasks across a longer time horizon. Several important research questions along this line include:

- How do we utilize the distributed sensors of agents to collectively recognize the current state of the world?
- How do agents exploit this task compiler and reach a consensus on a sequence of tasks?
- How can we allow such a compiler to automatically generate local constraints from global objective functions?[1]
- How can an agent language framework be developed such that one can easily coordinate different agent networks and interface with different tasks?

Large scale "Micro"-Agent Networks: The real world applications I have demonstrated in this thesis are composed of tens of physical agents. From Chapter 5's simulations, we have seen that this approach can potentially scale to thousands of agents while remaining reactive to perturbations. Future actuator and sensor technologies may allow us to construct applications consisting of thousands or even millions of "micro"-agents to form programmable materials (e.g., the Chembot project [66]).

Toward this direction, we have investigated the idea of building a soft and adaptive orthotic with a "programmable micro sensor-actuator network." The goal for such an orthotic is to use its distributed sensor to infer the current gait cycle of the patient and to utilize the shape change of the material to correct pathological gaits (e.g., cerebral palsy gaits). In this manner, the orthotic can provide variable assistance depending

[1] One example of such global functions is the total actuation energy consumption to achieve the desired global state.

on the time in the gait cycle, the activity level, and the needs of the wearer. The subject will then have individualized control, causing the muscles to be used more appropriately, which could possibly lead to a re-education of the motor system and eventual independence from the orthotic system.

Figure 8.1: The soft and adaptive orthotic prototype is embedded with many mini-actuators constructed by shape memory alloy (SMA) wires.

As shown in Fig. 8.1, the orthotic prototype is embedded with many mini-actuators constructed by shape memory alloy (SMA) wires. Our initial results suggest that such a device can potentially correct pathological gaits if it is actuated properly. One main challenge here is how to embed sensors and computation in this device such that the physical agents can collectively recognize the current state of the patients.

In addition to static agent networks, the implicit leadership algorithm developed in this thesis (Chapter 7) can also be applied to coordinate decision-making tasks in dynamic (mobile) micro-robot networks. For example, one may apply the algorithm to solve a direction consensus reaching among numerous aerial robobees.

Animal Network Analysis: The algorithms developed in this thesis have been largely inspired by individual behaviors in animal groups. Through the formally-defined multiagent model and the aggregation of interactions among agents in a matrix form, I was able to leverage graph and matrix theories to thoroughly analyze these artificial networks. One interesting and challenging direction is how one can utilize the theoretical framework developed here to reexamine animal or even human networks. For example, one could examine whether the diameter of a network is also an important factor in determining the coordination performance of animal networks similar to the artificial networks that I have analyzed. One fundamental difference between an artificial network and an animal network is the "self-interested" component. In an artificial agent network, we can safely assume all agents are cooperative. However, both the species of animals and the type of tasks they can potentially lead

to significantly different levels of self-interest. A systematic approach that generalizes the theoretical analysis in this thesis and adapts it to such scenarios would be an interesting direction for future exploration.

Appendix A

Proofs of the Basic Algorithms

A.1 A. Proof of Theorem 1

I first show that analyzing Eq. 4.2 is equivalent to analyzing a linear dynamical system without the bias vector. The optimality condition:

$$X^* = A \cdot X^* + \tilde{b} \tag{A.1}$$

can be rewritten as: $\alpha L \cdot X^* = \tilde{b}$. I 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 Eq. 4.2:

$$Y(t+1) = A \cdot Y(t) \tag{A.2}$$

where $Y(t) = X(t) - X^*$. The following proof follows the procedure of [48]. Since L is a real symmetric matrix, the well-known Courant-Fischer Theorem yields:

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

As $Y(t)^T \cdot \mathbf{1} = 0$ for all t, we can utilize the results from [48] and show that:

$$\max_{Y(t)} \frac{Y(t)^T A Y(t)}{Y(t)^T Y(t)} = \mu_2(A) < 1 \tag{A.3}$$

$\mu_2(A)$ denotes the 2nd largest eigenvalue of A. Define a Lyapunov function $V(t) = \sum_i (x_i - x_i^*)^2 = Y(t)^T Y(t)$. Now, following Eq. A.3, we get:

$$\begin{aligned} V(t+1) &= (AY(t))^T (AY(t)) = Y(t)^T A^2 Y(t) \\ &< (\mu_2(A))^2 Y(t)^T Y(t) = (\mu_2(A))^2 V(t) \end{aligned} \tag{A.4}$$

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

A.2 Proof of Theorem 2

The $A(t)$ matrix in Eq. 4.7 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 [43]. 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 w.r.t. a particular $A(t)$. 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 t.[1] 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 analagously-defined Lyapunov function, a similar derivation as Eq. A.3 and A.4 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 $A(t)$.

A.3 Proof of Theorem 4

Expanding out the definition of 2-norm, we have:

$$||X(0) - X^*||^2 = \sum_i (x_i(0) - x_i^*)^2 \tag{A.5}$$

$$= \left(\sum_i x_i^2(0)\right) + \left(\sum_i (x_i^*)^2\right) - 2\sum_{i<j} x_i(0) x_j^*. \tag{A.6}$$

[1] The i^{th} element of X^*: $x_i^* + \sum_{a_j \in N_i} \phi_{ij}(t)(x_j^* - x_i^*) - \sum_{a_j \in N_i} \phi_{ij}(t)\Delta_{ij}^* = x_i^*$

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, $\sum_i x_i^2(t) \leq (\sum_i x_i(t))^2$ 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 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 Eq. 4.12 yields the result.

A.4 Proof of Theorem 5

We have seen in Theorem 1 that one 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 $\tilde{b} = \mathbf{0}$.

I first construct collective dynamics as that of Eq. 6. Let us denote a_k as the agent whose actuator fails in the process. It thus has a fixed state x^*. Let us denote n as the number of agents in the system. For the sake of proof convenience, I switch the index of agent a_k with the last agent a_n. a_k and a_n's set of neighbors[2], 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: $X(t) = (x_1(t), \cdots, x_{n-1}(t), x^*)'$.

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

$$X(t+1) = A \cdot X(t) = \prod_{m=1}^{t} A \cdot X(1) \tag{A.7}$$

This can be further expanded as following:

$$\begin{aligned} \prod_{m=1}^{t} A \cdot X(1) &= \begin{pmatrix} F & L \\ 0 & 1 \end{pmatrix}^t X(1) \\ &= M(t) \cdot X(1) = \begin{pmatrix} P(t) & Q(t) \\ 0 & 1 \end{pmatrix} X(1) \end{aligned} \tag{A.8}$$

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

where F is an $(n-1)\times(n-1)$ matrix and $F_{ii} = 1-\alpha\cdot|N_i|$, and $F_{ij}=\alpha$. 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 Eq. A.7, I 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, I 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 $\sum_j A_{ij}=1, \forall i$), thus $M(t)$ is also row stochastic with $\sum_j M_{ij}(t)=1, \forall i$. We can further expand

$$Q(t) = \sum_{m=1}^{t} F^{m-1}\cdot L \tag{A.9}$$

Since the failed agent can still communicate, the communication graph G stays connected. From Eq. A.9, 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 Eq. A.9 that $\|Q(t)\|_{\min}$ monotonically increases with t after time $d_{\max}+2$. Since $M(t)$ is row stochastic, $\left(\sum_j P_{ij}(t)\right)+Q_i(t)=1, \forall i$. 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 $\lim_{t\to\infty} Q(t) = \mathbf{1} \Rightarrow \lim_{t\to\infty} x_i(t) = x^*, \forall i$. All agents will eventually have the same state x^* as the failed agent a_n.

A.5 Proof of Theorem 6

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

$$\begin{aligned} X(t+1) &= A\cdot X(t) = \prod_{m=1}^{t} A\cdot X(1) = \begin{pmatrix} F & L \\ 0 & I \end{pmatrix}^t X(1) \\ &= M(t)\cdot X(1) = \begin{pmatrix} P(t) & Q(t) \\ 0 & I \end{pmatrix} X(1) \end{aligned} \tag{A.10}$$

Different from Theorem 5's proof, matrices F and $P(t)$ are now $(n-k)\times(n-k)$ matrices and L and $Q(t)$ are now $(n-k)\times k$. Let us 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: $Q_{ij}(t) \to Q^*_{ij}\,\forall i,j$. Due to the fact that the bottom-right submatrix of $M(t)$ is a $k\times k$ identity matrix, we can get $Q_{ij}(t) \geq Q_{ij}(t-1)\,\forall i,j$; 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 $Q^*_{ij}\,\forall i,j$. Since each failed agent's fixed state can be different, each agent converges to a potentially different fixed state:

$$\lim_{t\to\infty} x_i(t) = \sum_j Q^*_{ij}x^*_j$$

A.6 Proof of Theorem 7

The proof of Theorem 7 is the same as Theorem 6 of [48]. In [48], Olfati-Saber et al. 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) I assume here is the same as [48] which refers to the union of all graphs over a finite sequence of intervals are connected.

A.7 Convergence Rate vs. Topology

I provide several graph topological factors v.s λ_2 ($\mu_2 = 1-\alpha\lambda_2$). I define several factors that are not mention in the main text of the thesis. Mean distance between two vertices: $\bar{\rho} = \frac{1}{N(N-1)}\sum_{\forall u,v\in V(G),u\neq v} d(u,v)$. Maximal degree sum of two connected vertices: $d^+_{\max} = \max\{deg(u)+deg(v)|uv \in E(G)\}$.

I can therefore summarize λ_2's relationships with various topological factors from [42, 10] 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

Appendix B

Proofs of Convergence in Different Self-Adaptive Applications

B.1 Pressure-Adaptive Column

I denote $\theta_i(t)$ as the sensor reading of agent a_i at time t. Let us first recap the control law run on each agent:

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

where $x_i(t)$ is denoted as the length of the linear actuator (actuation state) of a_i. Because the height of agent a_i is decided by the length of the linear actuator, $x_i(t)$ is also the height of the agent. Δ^*_{ij} is defined as the unknown and desired height difference such that ai and a_j achieve equal pressure state to a_i, $\theta_i(t) = \theta_j(t)$. Let

$$\phi_{ij}(t) = \alpha \frac{\theta_j - \theta_i}{x_j(t) - x_i(t) - \Delta^*_{ij}}$$

The control law can then be rewritten as:

$$x_i(t+1) = x_i(t) + \sum_{a_j \in N_i} \phi_{ij}(t)(x_j(t) - x_i(t) - \Delta^*_{ij}). \tag{B.2}$$

I use a column vector $X(t)$ to denote all agents' heights at time t. Following a procedure similar to Chapter 4, the dynamics of all agents can be represented with the following linear system:

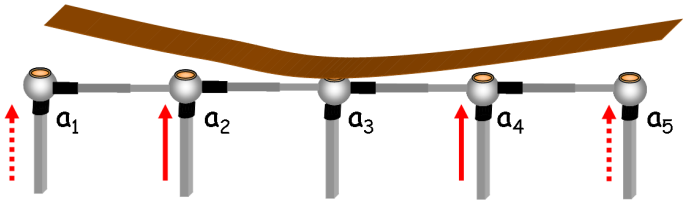


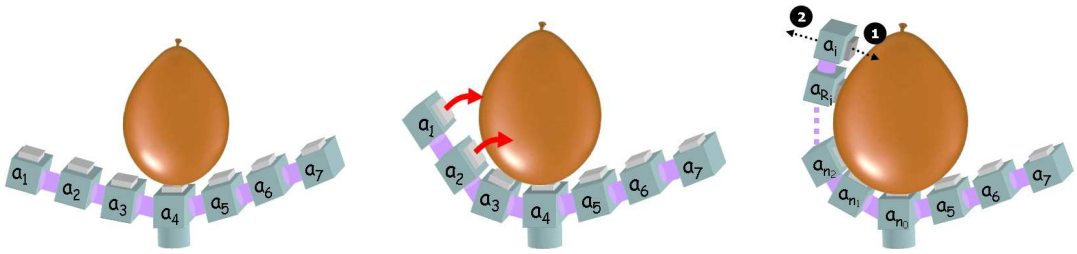
Figure B.1: The control law allows all agents to eventually contact the object. Initially, agent a_3 starts contacting the object, and the pressure differences allow both agents a_2 and a_4 to start moving upward. After a_2 and a_4 have contacted the object, a_1 and a_5 are further induced to move upward.

$$X(t+1) = A(t) \cdot X(t) - \tilde{b}(t) \tag{B.3}$$

In Chapter 4, I have shown that as long as $\phi_{ij}(t) > 0$ and $\sum_{a_j \in N_i} \phi_{ij}(t) < 1$, the system of Eq. B.3 will converge to the desired state at an exponential rate. We can see from Eq. B.2 that $\theta_j(t) - \theta_i(t)$ and $x_j(t) - x_i(t) - \Delta^*_{ij}$ need to have the same sign to ensure $\phi_{ij}(t) > 0$. For discussion, let us separate it into two cases:

- Case 1: Agents a_i and a_j have contacted the object, and both have a nonzero pressure reading. $\theta_i(t)$ increases monotonically with height $x_i(t)$ (because it is pushing the object further). In addition, Δ^*_{ij} represents the height difference while $\theta_j(t) = \theta_i(t)$. Therefore, $\frac{\theta_j - \theta_i}{x_j(t) - x_i(t) - \Delta^*_{ij}} > 0$. We also note that the sensor reading difference $\theta_j(t) - \theta_i(t)$ is a monotonic increasing function with $x_j(t) - x_i(t) - \Delta^*_{ij}$, and the slope of the function can be bound by a constant: $0 < \frac{\theta_j - \theta_i}{x_j(t) - x_i(t) - \Delta^*_{ij}} < C, \forall t$. We can therefore set constant α to be between 0 and $\frac{1}{n \cdot C}$, where n is the maximal number of neighbors for all agents to ensure that $\sum_{a_j \in N_i} \phi_{ij}(t) < 1$.
- Case 2: At least one agent between a_i and a_j has not contacted the object. We can show that they will eventually contact the object and become Case 1 as long as one agent in the system starts contacting the object. We use Fig. B.1 as an illustration. Initially, only a_3 has contacted the object and has nonzero pressure reading. By running the control law, a_2 and a_4 will increase its length due to their pressure differences with a_3. After a_2 and a_4 have reached the object, their sensor differences with a_1 and a_5 will let a_1 and a_5 contact the object eventually. Therefore, as long as the agents' nearest neighbor graph is connected, all agents will eventually contact the object and become Case 1.

I have shown that Case 1 satisfies the converging conditions we outlined in Chapter



(a) (b) (c)

Figure B.2: (a) Initial configuration of the gripper (b) Agents' actuation may have a long-range effect, e.g., changing a_3's actuation can affect both a_1 and a_2 (c) Agents $a_{n_0}, a_{n_1}, \cdots a_{n_k}, a_{R_i}$ have converged to the equal pressure state. In this case we can ensure that $\phi_{iR_i}(t) > 0$.

4 and that Case 2 will eventually become Case 1. Thus, the system will converge to the desired state at an exponential rate.

B.2 Modular Gripper

This analysis follows a procedure that is similar to the previous proof. The control law can be written in the following form:

$$x_i(t+1) = x_i(t) + \phi_{i,R_i}(t)(x_{R_i}(t) - x_i(t) - \Delta^*_{iR_i}). \tag{B.4}$$

where

$$\phi_{i,R_i}(t) = \alpha \frac{\theta_{R_i} - \theta_i}{x_{R_i}(t) - x_i(t) - \Delta^*_{iR_i}}$$

and all agents' dynamics can again be written in the form of Eq. B.3. We need to ensure that $\phi_{iR_i}(t) > 0$ and $\sum_{a_j \in N_i} \phi_{iR_i}(t) < 1$ as in the previous proof. Nevertheless, the agent's actuation no longer effects only its own sensor state. I use Fig. B.2 (a - b) as an example: When the gripper has reached the object, the first agent to contact the object is fixed (i.e., agent a_4, in our example). When each agent tries to control its own actuator, it can potentially change the sensor state of the other agents. For example, when agent a_3 makes its servo rotate inward, it is possible for it to change the states of agents a_1 and a_2. Because all agents are connected in a chain, agents a_1 and a_2 will move with a_3 when it rotates. Instead of ensuring $\phi_{iRi}(t) > 0$ for all i, we first show that it is true for some i.

Because agents are connected in a chain, the forward kinematics of agent ai can be written in the following form:

$$y_i(t) = T_i^i(x_i(t)) \cdot T_{R_i}^i(x_{R_i}(t)) \cdot T_{n_{k-1}}^{n_k}(x_{n_{k-1}}(t)) \cdots T_{n_0}^{n_1}(x_{n_0}(t)) \cdot y_{n_0}(t) \tag{B.5}$$

where agent y_{n_0} is the coordinate of the fixed agent a_{n_0} at time t. As shown in Fig. B.2 (c), a_{R_i} is the agent that a_i receives the message from, and $a_{n_0}, a_{n_1}, \cdots, a_{n_k}$ represent agent indices along the chain from the fixed agent a_{n_0}. T_i^i indicates agent a_i's coordinate change according to its own (rotational) actuation. $T_j^i(x)$ in a 4 matrix indicates a frame transformation between agents a_j and a_i with (rotational) actuation parameter x. Eq. B.5 allows us to see that there is indeed a long range effect from each agent's actuation [1].

First, when agents along the chain $a_{n_1}, a_{n_2}, \cdots a_{n_k}$ have converged to the desired state, agent a_i's control law will allow $\phi_{iR_i}(t) > 0$. In this case, control law Eq. B.4 will make agent a_i rotate inward if its pressure reading is smaller than a_{R_i} (case 1 of Fig. B.2 (c)). This will allow the sensor difference between agent a_i and a_{R_i} to be minimized. On the other hand, if agent a_i's pressure reading is larger than a_{R_i}, the control law will make agent a_i rotate outward. In this case, then, the control law drives agent a_i toward the right direction to equalize pressure readings between a_i and a_{R_i}. Therefore, $\phi_{iR_i}(t) > 0$.

The dynamics of all agents can be rewritten in the following form:

$$\begin{bmatrix} \vdots \\ \tilde{X}(t+1) \\ \vdots \end{bmatrix} = \begin{pmatrix} \ddots & & \\ & \tilde{A}(t) & \\ & & \ddots \end{pmatrix} \cdot \begin{bmatrix} \vdots \\ \tilde{X}(t) \\ \vdots \end{bmatrix} + \begin{bmatrix} \vdots \\ \tilde{b}(t) \\ \vdots \end{bmatrix}$$

where

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

indicates the dynamics of a subset of connected agents that can ensure $\phi_{iR_i}(t) > 0$ for all agents a_i. This include the subset of agents have converged to the equal pressure state (I denote this subset as Ω) and one immediate neighbor. For example, in Fig. B.2 (c), $\Omega = a_{n_0}, a_{n_1}, \cdots a_{R_i}$, and a_i is the immediate neighbor. It is clear that a_i's pressure state will eventually converge to the same state as the agents in Ω, because Eq. B.4 with $0 < \phi_{iR_i}(t) < 1$ will eventually allow $x_i(t) - x_{R_i}(t) = \Delta_{i,R_i}^*$. Therefore, ai will eventually converge to the equal pressure state as, and Ω will increase by one element: $\Omega = \Omega + \{a_i\}$.

As long as the nearest neighbor graph of all agents is connected, one can ensure Ω will eventually include all agents in the system by following the same procedure as above. Therefore, all agents will converge to the equal pressure state.

---

[1] For example, agent a_{n_0}'s parameter x_{n_0} can potentially affect agents $a_{n_1}, a_{n_2}, \cdots a_{n_k}$, and the end agent a_i

Appendix C

Robot Hardware

In this appendix, I provide detailed description of a struts-based modular robot, *Morpho*, and a chain-based modular robot that we built to explore this algorithmic framework's applicabilities in real world applications.

The Morpho struts-based modular robot design is inspired by the creation of complex structures and functions in biology via deformation (Fig. C.1). Our design is based on the *Tensegrity* model of cellular structure, where active filaments within the cell contract and expand to control individual cell shape, and sheets of such cells undergo large-scale shape change through the cooperative action of connected cells. This deformation process plays a role in many processes, e.g. early embryo shape change and lamprey locomotion. Modular robots that replicate the basic deformable multicellular structure have the potential to quickly generate large-scale shape change and create dynamic shapes to achieve different global functions.

Based on this principle, our design includes four different modular components: (1) active links, (2) passive links, (3) surface membranes, and (4) interfacing cubes. We show several self-deformable structures that can be generated from these components, including a self-deformable surface, expandable cube, terrain-adaptive bridge. In simulation, we show that these components can be configured into a variety of bio-inspired robots, such as an amoeba-like robot and a tissue-inspired material.

The design of our chain-based modular robot is similar to that of Superbot [61] and M-TRAN [45]. I also provide a description on each hardware component we used to construct our chain-based modular robot and different example configurations that it can achieve.

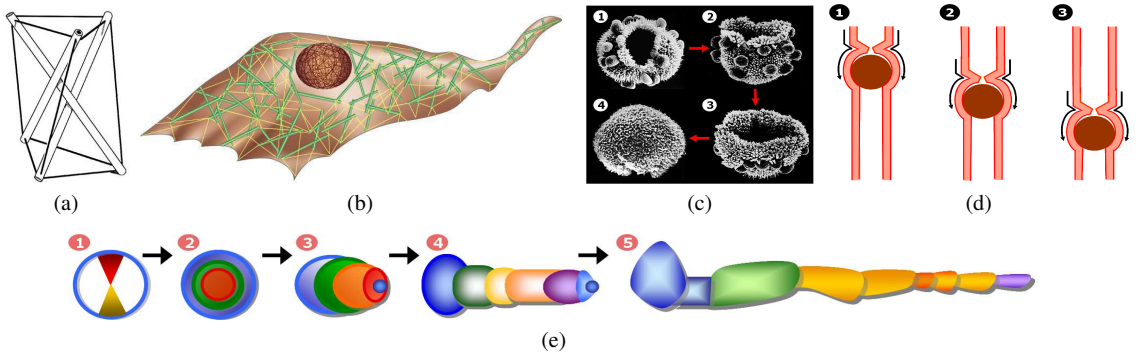


Figure C.1: Self-deformation in biology: (a) The Tensegrity structure. The model provides an explanation of how cells get their shapes. (b) A cell and its contractile filaments (colored green), which provide inner pulling force (photo courtesy of Matt Pickett). (c) The volvox algae form a hollow spherical colony that can invert itself through local contractions of individual cells (photo courtesy of David Kirk and Ichiro Nishii). (d) Pertistolic waves in the esophagus are achieved through distributed contraction of cells. (e) The imaginal disc of a fruit fly transforms from a flat sheet to a more complex 3D shape, a leg.

C.1 Morpho: A Self-deformable Modular Robot Inspired by Cellular Structure

The design and control of modular robots has been widely influenced by multicellular behaviors in biological systems, from module design to tasks like self-repair and self-replication [94]. Several research groups have demonstrated sophisticated mechanical designs that allow modular robots to transform their shape through changing the connectivity of modules [55], [4]. This process usually involves either self-reconfiguration (modules actively attaching, detaching, and moving to assume different positions) or self-assembly (modules moving via external vibrations or forces and selectively attaching). One can think of these processes as analogous to creating shapes by cell migration, cell growth, and even cell death in biology. The rearrangement of connectivity allows one to achieve a large range of shapes, even from a few modules. However, the shape change process is often slow in hardware implementation.

Self-deformation is another shape changing process observed in multicellular biological systems. Unlike growth and migration, this depends on a fixed connected topology of cells where individual cells exert actuation forces (expansion, compression) on the whole structure. This type of multicellular behavior can create complex shapes and achieve complex functions: e.g. gut formation in early animal embryos,

contractions in the heart, traveling waves in the intestines, and locomotion in lampreys and manta rays. Ingber et al have proposed a Tensegrity-like model [77], [24], of how individual cells and multicellular tissues achieve this shape change, through the control of contractile filaments embedded in each cell. This model of shape change has been explored in depth in biology literature [78].

Modular robotic systems that replicate the basic deformable multicellular structure have the potential to quickly generate large-scale shape change and create time-varying shapes to achieve different global functions. However, self-deformation has only been explored to a limited extent in modular robotics; e.g. chain-based modular robots which use module angle change to create locomotion and shape change but also rely on attachment/detachment [29], [61], [81]. Only recently have robots based on contracting linear modules been explored [39]. Our research is inspired by this work. We seek to define a modular robot system aimed at exploring shape change through self-deformation.

Here we present a self-deformable modular robot design. Our design is based on the Tensegrity model of cellular structure, where active filaments within the cell contract and expand to control individual cell shape, and the forces between connected cells create shape change within the tissue. Our system incorporates four different modular components: (1) active links: linear structures whose overall length can be controlled. (2) passive links: linear structures that can be passively compressed or expanded. (3) surface membranes: square structures that are capable of expanding. (4) interfacing cubes: an connector that can interface with passive/active links and surface membranes. We show that with different combinations and connections between the four components, many robotic configurations can be created, including two dimensional surfaces, volumetric structures, and several bio-inspired robots.

C.1.1 Morpho Modular Robot Design

In this section, we describe the design of each component of Morpho. There are four different modular components, including active links, passive links, interfacing cubes, and surface membranes. Geometrically, the active/passive links can be viewed as lines, interfacing cubes can be viewed as points and surface membranes can be viewed as planes. The main function of active links is to enable deformation of the structure by varying its physical length. Passive links allow the structure to resolve different geometries by passively expanding and contracting in accordance with the deformation generated by the active links. The surface membrane provides a flexible 3D plane to convert the skeleton structure to a volume, when attached to other modules. This allows the structure to interact with objects and form deformable volumes. The interfacing cubes provide a connection mechanism for all of the other modular components. As shown in Fig. C.2, through different combinations and configurations

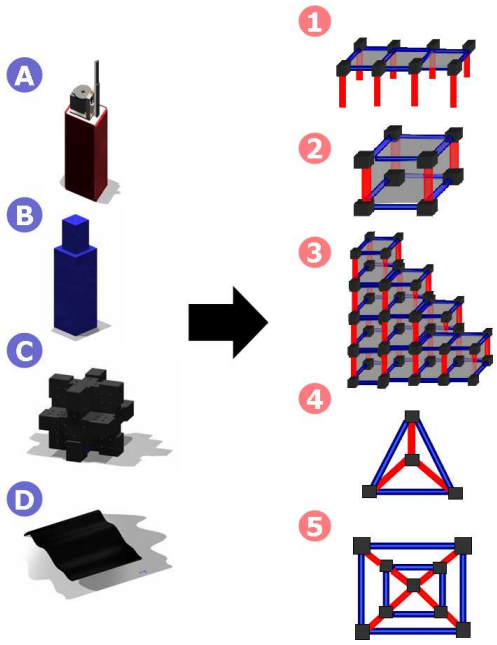
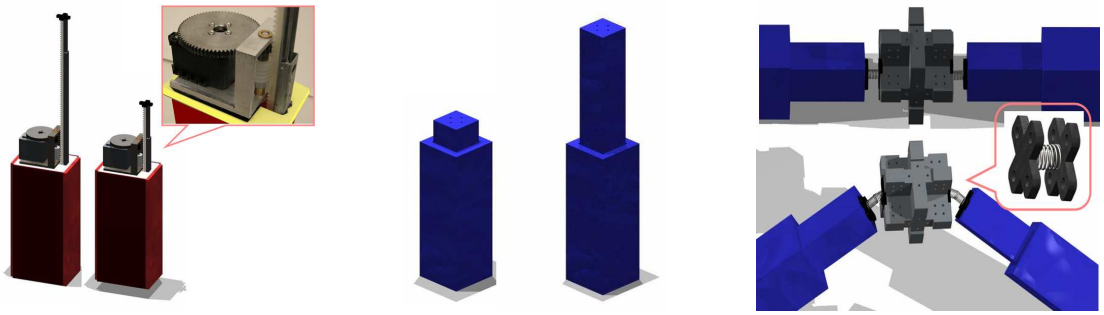


Figure C.2: Morpho has four different types of reconfigurable modules that can be combined to form structures like those shown on the right: (1) A surface structure capable of forming a terrain adaptive bridge or transporting an object. (2 – 3) A cubic structure capable of volumetric deformation. Cubes can be combined to form larger structures, such as a staircase. (4 – 5) Links can also be combined to form non-cubic structures, such as this tetrahedron and this pyramid (as viewed from above)

of these modules we are capable of forming different types of 2D structures and 3D volumes, including surfaces, cubes, tetrahedrons, and compositions of those structures (Fig. C.2 (1 – 5)). The Morpho modules are generic enough to be reconfigured into many bio-inspired robots. We will illustrate some of them in Section V. We will now provide a detailed description of each hardware element.

- **Active Link:** The active links can controllably change the skeleton of the structure, as shown in Fig. C.3 (a). The mechanical functionality of the active link is similar to a linear actuator. The main difference is that its physical length can be precisely controlled. In contrast, most linear actuators can only provide fully expanded/compressed configurations 1. In addition, the active links are also equipped with connecting mechanisms to attach to interfacing cubes.

  As shown in Fig. C.3 (a), our current hardware implementation of an active link uses a AX-12+ servo. The AX-12+ provides position, speed, and load feedback. Each servo is connected to a large spur gear, coupled with a smaller



(a) (b) (c)

Figure C.3: (a) The expanded (left) and contracted (right) state of the red active links. Active links expand/contract when a servo drives a vertically mounted rack up/down. As they expand and contract they force (b) the blue passive links to contract and expand to accommodate new geometries. (c) Interfacing cubes are used to connect to links and membranes. Since some geometries require a change in the angle between a passive link and a connecting cube, the spring interface between the two gives extra flexibility.

spur gear, to increase the speed at which the links are able to move, giving the system an overall 1:6 gear ratio. The small spur gear is attached to a worm gear that raises and lowers a vertically mounted rack. These continuous rotation actuators allow the active links to expand and compress with an overall six inch height differential.

The maximal toque an AX-12+ can generate is 16.5 kgf.cm, and this generates a maximal payload of 2 kg for a single active link. To further increase the speed of expansion and contraction we also reduced friction in the system by mounting the vertical rack on a ball-bearing slider. When the motor rotates at full speed the active link can be extended/compressed at the speed of 2 inch/sec. A given length can be precisely achieved.

- **Passive Link:** Passive links are components that can be expanded or contracted while active links perform actuation. These passive links provide a supporting framework for the surface membrane. The detailed design of the passive link is shown in Fig. C.3 (b) Each link consists of two concentric rectangular shells. The inner shell has a larger step at the end that slides along the inner dimension of the outer shell, using it as a bearing surface. To reduce friction between moving components, the shells for both the passive and active links were made out of acrylic, which has a low friction coefficient and can be rapidly machined with a laser cutter.

- **Interfacing Cube:** The function of the universal interfacing cube is to provide

convergence points for links. As shown in Fig. C.3 (c), the interfacing cubes provide six different attachment points for both passive and active links in addition to eight different surface membrane attachment points along all major axes. This allows links to connect in different ways according to different geometries. The surface area to which the outer membrane attaches is maximized in order to provide more stability and keep the membrane in tension.

A flexible connector is used to attach passive and active links with the interfacing cubes, as seen in Fig. C.3 (c). The two cross pieces screw into the cube and link. The spring allows for two consecutive nodes to reach different heights without over-constraining the system while providing enough restoring force to maintain linearity between the links and the connectors.

- **Surface Membrane:** The function of the surface membrane is to cover the 3D skeleton, changing the overall structure into a volume or a surface. Therefore, the material needs to be stretchable while maintaining rigidity. This allows the surface or volume to deform while maintaining its overall shape.

  Material selection was based on its bulk modulus (a measure of elasticity) and Youngs Modulus (a measure of rigidity). Neoprene provides enough rigidity to minimize deformation under external pressure and can be stretched up to 300% its original size. Another potential selection of material is latex, which can stretch up to 780size. The surface membranes attach flexibly along the edges of the links as well as the interfacing cubes.

- **Communication/Control:** We built our communication structure and protocol on the existing communication framework of the AX-12+ actuators. The maximal communication speed of the AX-12+ actuators is around 1Mbps. The control algorithm is run on a laptop computer (2GHZ CPU) that simulates purely distributed control in our simulation environment. At each iteration, the simulator takes sensor input from the real robot through serial port interface. The control algorithm then computes control parameter (desired expansion or contraction length) for each active link (actuator). The control output is then sent to all active links through serial interface. This procedure allows the same control trajectory to be simultaneously executed in simulation and on the real robot.

- **Simulation Environment:** Our simulation environment is constructed with the Open Dynamics Engine (ODE) which can simulate the physics of Morpho in different configurations (as shown in Fig. C.4 (a – d)). We note that ODE is not capable of simulating deformable material, like the surface membrane in our module components. In our simulation, we use a sequence of rigid objects to approximate the property of such materials.

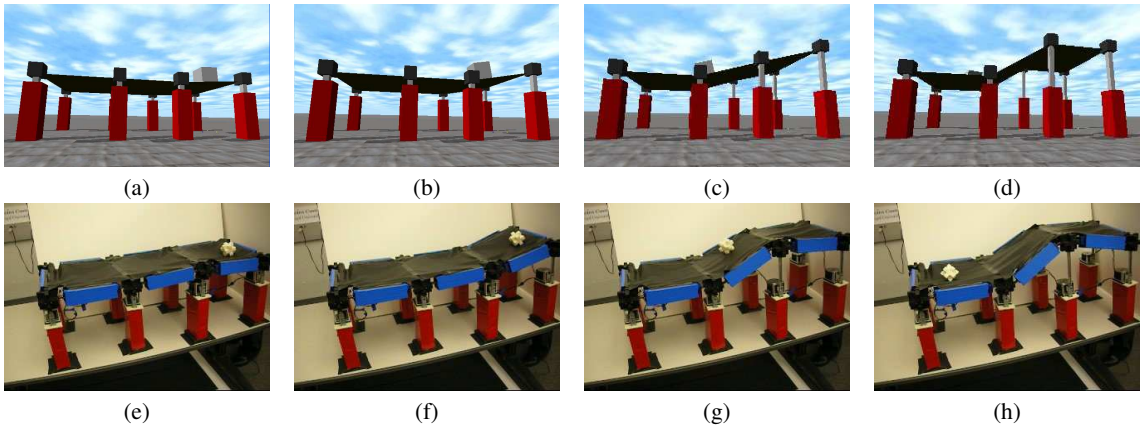


Figure C.4: (a–d) Control trajectories, computed in Open Dynamics Engine, of Morpho transporting a cubic object across three cells (e–h) the execution of computed control trajectories on the Morpho hardware platform.

C.1.2 Applications

We describe several applications configured from the Morpho hardware and simulation platform:

Object Transportation on Flexible Surface: In the first application, we perform object transport on a flexible surface. To construct the flexible surface, the top surface is connected with passive links, spanned by three surface membranes to provide flexibility. Active links provide the vertical supports which can expand and contract to controllably deform the surface. The object being transported is a 2" × 2" × 2" ABS plastic cube.

We view each of the active links as a single agent. Each agent gets its neighborhood topology via message passing [46]. For every surface membrane an agent is connected to, it is programmed with the desired direction of transport to allow the object to reach the goal position. This allows us to implement a simple cooperation mechanism on them: when four such agents sense the object is in the surface connecting them, two of them will extend in order to roll the object in the direction of the goal.

These control trajectories are first computed in ODE simulation as shown in Fig. C.4 (a – d), then executed on the Morpho. Fig. C.4 (e – h) shows that the control trajectory successfully transports the object from the first cell to the third cell. The active links were capable of generating a 20 degree tilt angle in approximately three seconds, i.e. each cell was capable of deforming 1.3 times its original size in approximately three seconds. This allows us to successfully transport objects between surface

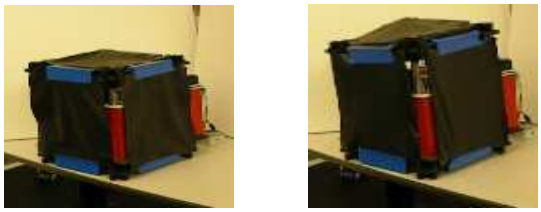


Figure C.5: Morpho configured into a cubic structure. In this configuration, with active links along one axis, the cube is capable of expanding to 1.3 times its original volume. Left: Original volume. Right: After expansion.

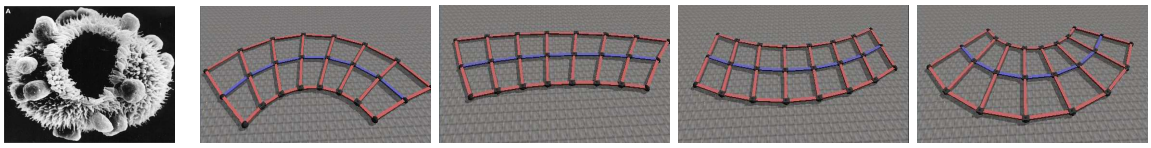
membranes. This experiment shows that the design and construction of Morpho is capable of tasks that require fast shape-deformation, such as transporting an object (that is within weight limit) across a deformable surface. We further test the maximal payload of each cell. We discover that one limitation of our design is the surface membrane is not rigid enough to hold object that is more than ~ 3 kg, although each active link is capable of lifting an object of 2 kg.

Expandable Cube: We also test the Morpho modules configured into a cube, with passive links on two axes and active links on the third. Surface membranes are added to all six sides of the cube. We then perform a volumetric deformation task on the cubic structure. The active links expand, maximizing the volume of this configuration of modules.

Fig. C.5 (a) and (b) show the volume of the cube before and after deformation. From the figure, the resulting volume is 1.3 times larger than the original volume. The stretchability of the surface membrane is the limiting factor in increasing the volume. With different materials and providing actuating links on all edges, (latex for example), or a thinner piece of neoprene the volume could potentially expand up to 8 times its original volume before the length of the expanded links becomes the limiting factor.

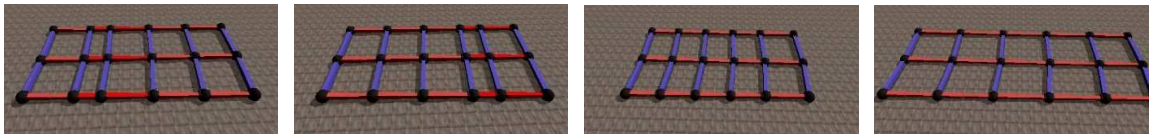
Biologically-Inspired Robots: There are many interesting biological examples in nature that can perform self-deformation. Here we simulate dynamics of the Morpho modules with Open Dynamics Engine and illustrate several robots that mimic biological processes:

- *Volvox Inversion*: The volvox is a chlorophyte that takes the shape of a sphere. It is capable of inverting its geometry by expanding and contracting different areas along its perimeter, as shown in Fig. C.6 (a). The Morpho, when configured appropriately, can reproduce this action. As shown in Fig. C.6 (b), active



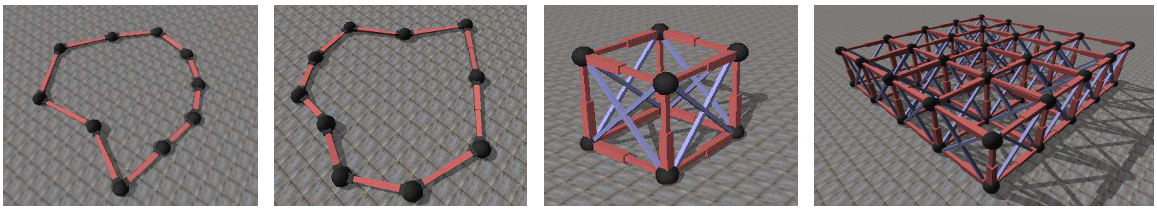
(a) (b) (c) (d) (e)

Figure C.6: (a–d) Control trajectories, computed in Open Dynamics Engine, of Morpho transporting a cubic object across three cells (e–h) the execution of computed control trajectories on the Morpho hardware platform.



(a) (b) (c) (d)

Figure C.7: A robotic structure that can perform locomotion by changing its shape. (a – b) The active links sequentially perform periodic motion; dark red links represent actuated links. (c – d) The robot traverse in horizontal direction by actuating active links simultaneously.



(a) (b) (c) (d)

Figure C.8: (a – b) Amoeboid robot. The robot changes its shape while active links perform actuation. (c) A basic cube of the tissue-inspired programmable material (d) A programmable material that is formed by a grid of 4×4 basic cubes.

links are placed along the inner and outer perimeter, with concentric inner arcs composed of passive links. The cross struts are made from active links because throughout the inversion process they maintain a constant length instead of being passively stretched. Inversion will occur when the active links along the inner arc expand while the active links along the outer arc contract. Fig. C.6 (b – d) shows a sequence of shapes generated by such collective actuation.

- *Locomotion via Self-Deformation*: There are many biological creatures that perform locomotion by changing their shapes. For example, inchworm and lamprey perform locomotion via periodic self-deformation. Here we illustrate a robotic structure that can perform locomotion via this self-deformation. As shown in Fig. 11, the horizontal links are placed with active links and the vertical links are placed with passive links. This configuration allows the robot to traverse

along the horizontal direction while active links periodically contract and expand. Fig. C.7 (a) and (b) show the case when active links sequentially perform such periodic actuation, and the robot is capable of performing locomotion. On the other hand, Fig. C.7 (c) and (d) show the case when all active links simultaneous actuate.

- *Amoeboid Robot*: The amoeboid is a unicellular creature that can perform locomotion by transforming its shape. Our design is inspired by [73], and the robot's perimeter are placed with active links. Two consecutive active links are connected with a interfacing cube. The design of the interfacing cubes allows the robots shape to change with the length of active links.

- *Tissue-Inspired Programmable Material*: Morpho modules can also be connected to form programmable material structure that was previously proposed in [46]. Fig. C.8 (c) shows a basic cube of the programmable material, and Fig. 12 (d) demonstrates a programmable material that is formed by 4×4 cubes. Same basic cube can be used to form other animal structures, e.g. robotic fish simulation in [71] use four cubes to form body of the fish.

Appendix D

Appendix for Self-Organizing Strategies for Adaptive Locomotion

D.1 Proof of Theorem 10

Let $\phi_i(t)$ be module i's phase offset variable at time step t. We first collect all $\phi_i(t)$ into a column vector:

$$X(t) = [\phi_1(t), \phi_2(t), \cdots, \phi_n(t)]'$$

The update equation in Alg. 1 can be rewritten as:

$$\phi_i(t+1) = \phi_i(t) + \alpha \cdot \sum_{j \in N_i} (\phi_j(t) - \phi_i(t) - \phi^*_{ij})$$

Aggregating all modules' update equations, we can write the collective dynamics of phase offset variable updates in matrix form:

$$X(t+1) = A \cdot X(t) + \tilde{b} \tag{D.1}$$

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

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

and where $\tilde{b}_i = \alpha \cdot \sum_{j \in N_i} \phi^*_{ij}$ is a *bias vector*. Note that A is a stochastic matrix since each row sums to 1. In addition, $\sum_i \tilde{b}_i = 0$, since $\phi^*_{ij} = -\phi^*_{ji}$ for all i, j. We can write the optimality condition (when all $\phi_j - \phi_i = \phi^*_{ij}$) as:

$$X^* = A \cdot X^* + \tilde{b} \Rightarrow \alpha \cdot L \cdot X^* = \tilde{b}$$

where $L = 1/\alpha(I - A)$ is a Laplacian matrix with $\text{rank}(L) = n-1$ and $\text{null}(L) = \mathbf{1}$. Since $\sum_i \tilde{b}_i = 0$ and A is a stochastic matrix, the summation of state variables is conserved over time: $\sum_i \phi_i(t) = C, \forall t$. This additional constraint falls in the nullspace of L. Let us define L' as the matrix whose first n rows are L and whose last row is the new constraint; we can then show $\text{rank}(L') = n$ and there exists a unique solution X^* for $\alpha L' X^* = \tilde{b}$. Therefore, there is a unique optimal solution X^* (satisfying $\phi_j - \phi_i = \phi^*_{ij}$, $\forall i$ and $j \in N_i$) for any initial condition $X(0)$. Subtracting the optimality condition $X^* = AX^* + \tilde{b}$ yields the following dynamics:

$$X(t+1) - X^* = A \cdot (X(t) - X^*) \Rightarrow Y(t+1) = A \cdot Y(t) \tag{D.2}$$

Eq. D.2 has the same form as average consensus [48]; we can then follow the proof procedure in [48]. Let $X^* = [\phi^*_1, \phi^*_2, \cdots \phi^*_n]$ with $\phi_j - \phi_i = \phi^*_{ij}$ for all i and $j \in N_i$. We define Lyapunov function $V(t) = \sum_i (\phi_i(t) - \phi^*_i)^2 = (X(t) - X^*)^T (X(t) - X^*)$. Following the same procedure as Theorem 4 in [88], we can show:

$$V(t+1) < (\mu_2(A))^2 V(t)$$

where $\mu_2(A)$ is the 2nd largest eigenvalue of A. Using the theory of graph Laplacians, we have $0 \leq \mu_2(A) < 1$ if the module communication graph G is *connected*. We can then show $\lim_{t \to \infty} V(t) = 0$ and $X(t)$ converges to X^* with exponential rate.

D.2 Pseudo Code for Direction Agreement

Pseudocode for agents to agree on direction (all agents run this routine in parallel). Based on locally shared information about preferred direction (D) and associated confidence level (C), all agents will adopt the direction of globally maximum confidence, with ties randomly broken. Local confidence outliers exceeding their neighbors' values by $\geq T$ are discarded. Updates may be synchronous or asynchronous, so long as all agents update once at the start before any of their neighbors update a second time.

Algorithm 5

```
    H ← 0
    loop
      R ← 0
      k ← agent with max C among self and neighbors
 5:   if k is self then
        h ← agent with max C among neighbors only
        if C(self) − C(h) > T then
          k ← h
        if C(k) = C(h) then
10:       // C agreement; check for D disagreement
          if C(k) = C(j) and D(k) ≠ D(j) for any neighbor j then
            R ← small random value (mean 0)
      if C(k) − C(self) > T then
        if H = 0 then
15:       // wait to see if potential outlier is discarded
          H ← C(k)
          k ← self
        else if H = C(k) then
          H ← 0
20:   else
        H ← 0
      C(self) ← C(k) + R
      D(self) ← D(k)
```

Appendix E

Proofs for Implicit Leadership Algorithms

E.1 Convergence Proof

I first aggregate all agents' state update dynamics into a single collective dynamic system. Here I consider the case of a single informed agent group $\mathcal{L}_1$ with common goal state x^*. Add an auxiliary agent a^* assumed to connect to all informed agents. Agent states can be aggregated into an $(n+1)$-dimensional vector:

$$X(t) = [x_1(t), \cdots, x_n(t), x^*]^T$$

The system's collective dynamics can written as:

$$\begin{aligned} X(t+1) &= A(t) \cdot X(t) && \text{(E.1)} \\ &= \prod_{m=1}^{t} A(m) X(1) && \text{(E.2)} \\ &= \prod_{m=1}^{t} \begin{pmatrix} F(m) & H(m) \\ 0 & 1 \end{pmatrix} X(1) && \text{(E.3)} \\ &= \begin{pmatrix} P(t) & Q(t) \\ 0 & 1 \end{pmatrix} X(1) && \text{(E.4)} \end{aligned}$$

where $F(m)$ is a $n \times n$ matrix, with $F_{ij}(m) = \frac{1}{(1+w_i)|\mathcal{N}_i(m)+1|}$ if $a_j \in \mathcal{N}_i(m)$ and $a_j \in \mathcal{L}_1$, $F_{ij}(m) = \frac{1}{|\mathcal{N}_i(m)+1|}$ if $a_j \in \mathcal{N}_i(m)$ and $a_j \notin \mathcal{L}_1$, and $F_{ij}(m) = 0$ otherwise. $\mathcal{N}_i(m)$ indicates the set of agent i's neighbors at time m. $H(m)$ is a n-dimensional column vector, $H_i(m) = \frac{w_i}{(1+w_i)}$ if $a_i \in \mathcal{L}_1$ and $H_i(m) = 0$ otherwise. I further define:

1. Matrix maximum norm: $\|M\|_\infty = \max_i \sum_j M_{ij}$;

2. Matrix minimum norm: $\|M\|_{-\infty} = \min_i \sum_j M_{ij}$;

3. $d_{\max} = \max_i {}_{\text{dist}}(a_i, n_i^*) \, \forall a_i \notin \mathcal{L}_1$, the maximal distance between a non-informed agent a_i and its closest informed agent n_i^*.

Static Topology

In this case, $A(m)$, $F(m)$, and $H(m)$ are time-invariant. We can let $A(m) = \bar{A}$, $F(m) = \bar{F}$, $H(m) = \bar{H}$, and

$$X(t+1) = \prod_{m=1}^{t} \bar{A} X(1) = \prod_{m=1}^{t} \begin{pmatrix} \bar{F} & \bar{H} \\ 0 & 1 \end{pmatrix} X(1) \tag{E.5}$$

Since $\bar{A}$ is a row stochastic matrix all of whose main-diagonal elements are positive, all elements in $\bar{A}$ are nonnegative ($\bar{A}_{ij} \geq 0$) and $\sum_j \bar{A}_{ij} = 1 \, \forall i$. We can show that $(\bar{F}^{d+1})_{i,j} > 0$ for any $\{i, j\}$ that are d hops apart, and the connectivity graph G is connected. Since

$$Q(t) = \sum_{m=1}^{t} \bar{F}^{m-1} \bar{H}$$

and

$$\bar{H}_i > \, 0 \forall i | a_i \in \mathcal{L}_1$$

we can ensure $Q_i(t) > 0 \, \forall i$ after $d_{\max} + 2$ time steps. Therefore, $\|Q(t)\|_{-\infty} > 0$. Since

$$\|P(t)\|_\infty = 1 - \|Q(t)\|_{-\infty}$$

we can derive $0 < \|P(t)\|_\infty < 1$ after $d_{\max} + 2$ time steps. The maximum norm of $P(t)$ decreases at least every $d_{\max} + 2$ time steps. Thus,

$$\lim_{t \to \infty} \|P(t)\|_\infty = 0$$

and

$$\lim_{t \to \infty} P(t) = \mathbf{0}, \lim_{t \to \infty} Q(t) = \mathbf{1} \Rightarrow \lim_{t \to \infty} x_i(t) = x^* \, \forall i$$

.

E.1.1 Dynamic Topology

Let us define

$$\bar{G} = \bigcup_{t'}^{t'+\Delta} G(t)$$

as a union of connectivity graphs over a finite time period Δ. If agents' topology is informed agent-connected during a finite period τ, then for each non-informed agent a_i, there exist at least $d_{\max} + 2$ instances within τ of a $\bar{G}$ that connects a_i with some informed agent. Thus, there are at least $d_{max} + 2$ instances of $A' = \prod_{t'}^{t'+\Delta} A(t)$ within τ. We expand

$$Q(\tau) = \sum_{m=1}^{\tau} \left(\prod_{k=1}^{m-1} F(k) \right) H(m)$$

Since all elements on the main diagonal of $A(t)$ are always positive and there are at least $d_{max} + 2$ instances of A', we can derive $Q_i(t) > 0 \, \forall i$ and $t > \tau$. Following a similar procedure as in the static topology case, we can show $0 < \|P(t)\|_\infty < 1$ after τ time steps and the maximum norm of $P(t)$ contracts. Thus,

$$\lim_{t\to\infty} P(t) = \mathbf{0}, \lim_{t\to\infty} Q(t) = \mathbf{1} \Rightarrow \lim_{t\to\infty} x_i(t) = x^* \, \forall i$$

E.2 Proof for Factors Affecting Convergence Speed

I first demonstrate that the system's convergence speed increases when a previously non-informed agent becomes informed, increasing the number of informed agents from m to $m + 1$. I first show this is true in the static topology case. As shown in the proof of Theorem 4.1, the speed at which the maximum norm $\|P(t)\|_\infty$ approaches zero determines the system convergence speed. I start by adding one additional informed agent to the system. Let $\mathcal{L}'_1 = l \cup \mathcal{L}_1$ denote the new informed agent group after adding agent l to the original $\mathcal{L}_1$. Let us use $\bar{F}'$, $P'(t)$, $Q'(t)$ to denote new agent matrices for the system with informed agent group $\mathcal{L}'_1$.

Let $Q_l(t)$ and $Q'_l(t)$ be elements of Q corresponding to l, in the m-informed agent and $(m + 1)$-informed agent cases respectively. We can derive matrix elements at $t = 2$ from Eq. E.4:

$$Q'_l(2) \;=\; \left(\sum_j \bar{F}'_{lj} Q'_j(1) \right) + \frac{w_l}{1 + w_l} > \sum_j F_{lj} Q_j(1) = Q_l(2)$$

With the same procedure, we can show $Q'_k(3) > Q_k(3)$ for all agents k that are neighbors of l, $Q'_r(4) > Q_r(4)$ for all r that are neighbors of k, etc. Thus $Q'_i(d' + 1) > Q_i(d' + 1) \, \forall i$, where $d' = \max_i \text{dist}(l, i)$.

Thus, $\|P'(d'+1)\|_\infty < \|P(d'+1)\|_\infty$ and agents converge faster in the $\mathcal{L}'_1$ case. Adding multiple informed agents at the same time is a trivial generalization of this proof.

A similar proof applies to the case of changing weight(s) of informed agent(s) from w_l to $w'_l > w_l$. Let $Q'_l(t)$ be an element of Q after the weight increase, and we can write

$$Q'_l(2) = \left(\sum_j \bar{F}'_{lj} Q'_j(1)\right) + \frac{w'_l}{1+w'_l} > \left(\sum_j F_{lj} Q_j(1)\right) + \frac{w_l}{1+w_l} = Q_l(2)$$

. We can then show $Q'_k(3) > Q_k(3)\ \forall k \in \mathcal{N}_l$, etc. After $d'+1$ time steps, we have $Q'_i(d'+1) > Q_i(d'+1) \forall i$.

With dynamic topology, if there exists a path between a_l and *all* non-informed agents, we can similarly show that

$$Q'_l(2) = \left(\sum_j \bar{F}'_{lj} Q'_j(1)\right) + \frac{w_l}{1+w_l} > \sum_j F_{lj} Q_j(1) = Q_l(2)$$

. If such a path exists $d'+1$ times in a finite period τ, we can show $Q'_i(\tau) > Q_i(\tau) \forall i$ and $\|P'(\tau)\|_\infty < \|P(\tau)\|_\infty$. Therefore, adding one more informed agent makes the system converge faster.

E.3 Proof for Multiple Informed Agent Groups

With q different informed agent groups, I introduce auxiliary agent states $\bar{x}^*_1 \cdots \bar{x}^*_q$, with agents in each informed agent group $\mathcal{L}_i$ connected to auxiliary agent with state $\bar{x}^*_i$. We can formulate the collective dynamics of all agents as Eq. E.5 of Theorem 4.1 with a different generating matrix:

$$\bar{A} = \begin{pmatrix} \bar{F} & \bar{H} \\ 0 & I \end{pmatrix}$$

$$\prod_{m=1}^{t} \bar{A} = \bar{A}^t = \begin{pmatrix} P(t) & Q(t) \\ 0 & I \end{pmatrix}$$

where $\bar{F}$ and $P(t)$ are $n \times n$ matrices, and $\bar{H}$ and $Q(t)$ are $n \times q$ matrices. Following the same procedure as for Thm. 4.1, we can show that the maximum matrix norm of $P(t)$ approaches zero:

$$\lim_{t\to\infty} \|P(t)\|_\infty = 0 \Rightarrow \sum_j Q_{ij}(t) = 1 \forall i$$

Since $Q(t)$ is multi-column, we show that each element in $Q(t)$ converges to a stable state: $Q_{ij}(t) \to Q^*_{ij} \, \forall i, j$. We decompose

$$\bar{A}^t = \begin{pmatrix} P(t) & Q(t) \\ 0 & I \end{pmatrix} = \begin{pmatrix} P(t-1) & Q(t-1) \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} \bar{F} & \bar{H} \\ 0 & I \end{pmatrix}$$

Since the bottom right submatrix of $\bar{A}$ is a $q \times q$ identity matrix, we can get $Q_{ij}(t) \geq Q_{ij}(t-1) \, \forall i, j$; 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 $Q^*_{ij} \, \forall i, j$. Since each informed agent group's state is different, each agent converges to a potentially different fixed state:

$$\lim_{t \to \infty} x_i(t) = \sum_j Q^*_{ij} \bar{x}^*_j$$

.

Bibliography

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

[2] T. Balch and R. C. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14:926–939, 1999.

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

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

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

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[7] J. T. Bonner. *The Social Amoebae*. Princeton University Press, 2009.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings of IEEE INFOCOM*, pages 1653–1664, Miami, FL, USA, 2005.

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

[10] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1994.

[11] I. D. Couzin. Collective cognition in animal groups. *Trends in Cognitive Sciences*, 13(1):36 – 43, 2009.

[12] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516, 2005.

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

[14] J. A. Fax. *Optimal and cooperative control of vehicle formations*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, 2001.

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

[16] T. Fukuda, S. Nagasawa, Y. Kawauchi, and M. Buss. Structure decision method for self organizing robots based on cell structure–cebot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 698–700, Scottsdale, AZ, USA, 1989.

[17] K. Gilpin, K. Kotay, and D. Rus. Miche: Modular shape formation by self-disassembly. In *Proc. of International Conference on Robotics and Automation*, 2007.

[18] F. Gokce and E. Sahin. To flock or not to flock: Pros and cons of flocking in long-range migration of mobile robot swarms. In *Proceedings of the 8th international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pages 65–72, Budapest, Hungary, 2009.

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

[20] R. Groß, M. Bonani, F. Mondada, and M. Dorigo. Autonomous self-assembly in swarm-bots. In *IEEE Transactions on Robotics*, 2006.

[21] G. J. Hamlin and A. C. Sanderson. *Tetrobot: A Modular Approach to Reconfigurable Parallel Robotics*. Springer, 1997.

[22] S. Hirose. *Biologically-Inspired Robots*. Oxford Sciences Pubublishing, 1993.

[23] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[24] D. Ingber. The architect of life. *Scientific American*, pages 48–57, Jan 1998.

[25] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[26] M. Ji. *Graph-Based Control of Networked Systems*. PhD thesis, Georgia Institute of Technology, 2007.

[27] M. W. Jorgensen, E. H. Ostergaard, and H. H. Lund. Modular atron: Modules for a self-reconfigurable robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2068–2073, Sendai, Japan, 2004.

[28] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji. 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, pages 2370–2377, Sendai, Japan, 2004.

[29] A. Kamimura, E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji. A self-reconfigurable modular robot (mtran) - hardware and motion planning software. In *Proc. of DARS*, 2002.

[30] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, page 482, Washington, DC, USA, 2003. IEEE Computer Society.

[31] S. Kim and S. Heyneman B. Santos D. Cutkosky M.R. Spenko, M. Trujillo. Smooth vertical surface climbing with directional adhesion. *IEEE Transactions on Robotics*, 24(1):65–74, February 2008.

[32] E. Klavins, S. Burden, and N. Napp. Optimal rules for programmed stochastic self-assembly. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[33] J. R. Kok, M. T. J. Spaan, and N. Vlassis. Multi-robot decision making using coordination graphs. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*, pages 1124–1129, University of Coimbra, Portugal, 2003.

[34] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[35] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM Series on Statistics and Applied Probability, 1999.

[36] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2004.

[37] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of IEEE Conference on Decision and Control*, pages 2968–2973, Orlando, Florida, USA, 2001.

[38] D. Lucarelli and I. J. Wang. Decentralized synchronization protocols with nearest neighbor communication. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 62–68, New York, NY, USA, 2004.

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

[40] J. Mclurkin and J. Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Toulouse, France, 2004.

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

[42] B. Mohar. Eigenvalues, diameter, and mean distance in graphs. *Graphs and Combinatorics*, 7:53–64, 1991.

[43] B. Mohar. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, pages 871–898. Wiley, 1991.

[44] S. Murata, K. Kakomura, and H. Kurokawa. Docking experiments of a modular robot by visual feedback. In *Proc. IROS*, 2006.

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

[46] R. Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2002.

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

[48] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[49] A. Pamecha, D. Stein, and G. Chirikjian. Design and implementation of metamorphic robots. In *in Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, Irvine, CA, USA, 1996.

[50] C. Parker and H. Zhang. Cooperative decision-making in decentralized multi-robot systems: the best-of-n problem. *IEEE/ASME Transactions on Mechatronics*, 14:240–251, April 1999.

[51] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, June 2007.

[52] J. Reif and S. Slee. Optimal kinodynamic motion planning for 2D reconfiguration of self-reconfigurable robots. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[53] W. Ren and R. W. Beard. *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*. Springer Publishing Company, 2007.

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

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

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

[57] S. Curtis et al. Tetrahedral robotics for space exploration. *IEEE Aerospace and Electronic Systems Magazine*, pages 22–30, 2007.

[58] B. Salemi, M. Moll, and W.-M. Shen. SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006.

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

[60] E. Seneta. *Non-negative Matrices and Markov Chains*. Springer Verlag, 1981.

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

[62] W. M. Shen, B. Salemi, and P. Will. Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *IEEE Transactions on Robotics and Automation (ICRA)*, 18:700–712, 2002.

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

[64] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, NY, USA, 2009.

[65] D. A. Spielman. Spectral graph theory and its applications. In *Tutorial in 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Providence, RI, USA, 2007. http://www.cs.yale.edu/homes/spielman/sgta.

[66] E. Steltz, A. Mozeika, N. Rodenberg, E. Brown, and H. M. Jaeger. JSEL: Jamming skin enabled locomotion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5672–5677, St. Louis, MO, USA, 2009.

[67] J. W. Suh, S. B. Homans, and M. Yim. Telecubes: Mechanical design of a module for self-reconfigurable robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4095–4101, Washington, DC, USA, 2002.

[68] D. J. T. Sumpter, J. Krause, R. James, I. D. Couzin, and A. J. W. Ward. Consensus decision making by fish. *Current Biology*, 18(22):1773 – 1777, 2008.

[69] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. volume 52, pages 863–868, 2007.

[70] H. G. Tanner and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20:443–455, 2003.

[71] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. of ACM SIGGRAPH*, 1994.

[72] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin. Self-organized flocking with a mobile robot swarm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pages 39–46, Estoril, Portugal, 2008.

[73] T. Umedachi, T. Kitamura, and A. Ishiguro. A fully decentralized control of an amoeboid robot by exploiting the law of conservation of protoplasmic mass. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[74] F. Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Pretoria, South Africa, South Africa, 2002. Supervisor-Engelbrecht, A. P.

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

[76] J. E. Walter, J. L. Welch, and N. M. Amato. Distributed reconfiguration of metamorphic robot chains. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing (PODC)*, pages 171–180, New York, NY, USA, 2000. ACM.

[77] N. Wang, J. Butler, and D. Ingber. Mechanotransduction acrtoss the cell surface and through the cytoskeleton. *Science*, 260:1124–1127, May 1993.

[78] L. Wolpert, R. Beddington, J. Brockes, T. Jessel, P. Lawrence, and E. Meyerowitz. *Principles of Development*. Oxford University Press, 1998.

[79] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN)*, pages 9–16, Piscataway, NJ, USA, 2005. IEEE Press.

[80] M. Yim. *Locomotion with unit-modular reconfigurable robot*. PhD thesis, Stanford University, Stanford, CA, USA, 1995.

[81] M. Yim, D.G. Duff, and K.D. Roufas. Polybot: a modular reconfigurable robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1381 – 1386, San Francisco, USA, 2000.

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

[83] M. Yim, W. M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems – challenges and opportunities for the future. *IEEE Robotics and Autonomation Magazine*, March:43–53, 2007.

[84] M. Yim, Y. Zhang, and D. Duff. Robotics: modular robots. *IEEE Spectrum*, 39(2):30–34, 2002.

[85] M. Yokoo. *Distributed constraint satisfaction: foundations of cooperation in multi-agent systems*. Springer-Verlag, London, UK, 2001.

[86] Y. Yoon and D. Rus. Shady3D: A robot that climbs 3D trusses. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4071–4076, Rome, Italy, 2007.

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

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

[89] C.-H. Yu and R. Nagpal. Self-adapting modular robotics: A generalized distributed consensus framework. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3051–3058, Kobe, Japan, 2009.

[90] C.-H. Yu and R. Nagpal. Biologically-inspired control for multi-agent self-adaptive tasks. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI), Nectar Track*, Atlanta, GA, USA, 2010.

[91] C.-H. Yu, J. Werfel, and R. Nagpal. Collective decision making in multi-robot systems by implicit leadership. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Toronto, Canada, 2010.

[92] C.-H. Yu, J. Werfel, and R. Nagpal. Coordinating collective locomotion in an amorphous modular robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, 2010.

[93] C.-H. Yu, F.-X. Willems, D. Ingber, and R. Nagpal. Self-organization of environmentally-adaptive shapes on a modular robot. In *Proceedings of the*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2353–2360, San Diego, USA, 2007.

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