Learning in Multiagent Control of Smart Matter

Oliver Guenther, Tad Hogg and Bernardo A. Huberman

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, U.S.A.
{guenther,hogg,huberman}@parc.xerox.com

Abstract

Embedding microscopic sensors, computers and actuators into materials allows physical systems to actively monitor and respond to their environments. This leads to the possibility of creating smart matter, i.e., materials whose properties can be changed under program control to suit varying constraints. A key difficulty in realizing the potential of smart matter is developing the appropriate control programs. One approach to this problem is a multiagent control system, which can provide robust and rapid response to environmental changes. To improve the performance of such systems, we describe how the agents' organization can adapted through simple learning mechanisms. As a specific example, we consider maintaining a physical system near an unstable configuration, a particularly challenging application for smart matter. This shows how the system's organization can adapt to the local physical structure to improve performance.

keywords: distributed control, learning organizational structure, smart matter

Introduction

Embedding microscopic sensors, computers and actuators into materials allows physical systems to actively monitor and respond to their environments in precisely controlled ways. This is particularly so for microelectromechanical systems (MEMS) [2, 3, 5] where the devices are fabricated together in single silicon wafers. Applications include environmental monitors, drag reduction in fluid flow, compact data storage and improved material properties.

In many such applications the relevant mechanical processes are slow compared to sensor, computation and communication speeds. This gives a *smart matter* regime, where control programs execute many steps within the time available for responding to mechanical changes. A key difficulty in realizing smart matter's potential is developing the control programs. This is due to the need to robustly coordinate a physically distributed real-time response with many elements in the face of failures, delays, an unpredictable environment and a limited ability to accurately model the system's behavior. This is especially true in the mass production of smart materials where manufacturing tolerances and occasional defects will cause the physical system to differ somewhat from its nominal specification. These characteristics limit the effectiveness of conventional control algorithms, which rely on a single global processor with rapid access to the full state of the system and detailed knowledge of its behavior.

A more robust approach for such systems uses a collection of autonomous agents, that each deal with a limited part of the overall control problem. Individual agents can be associated with each sensor or actuator in the material, or with various aggregations of these devices, to provide a mapping between agents and physical location. This leads to a community of computational agents which, in their interactions, strategies, and competition for resources, resemble natural ecosystems [10]. Distributed controls allow the system as a whole to adapt to changes in the environment or disturbances to individual components [8].

Multiagent systems have been extensively studied in the context of distributed problem solving [4, 6, 11]. They have also been applied to problems involved in acting in the physical world, such as distributed traffic control [12], flexible manufacturing [16], the design of robotic systems [13, 17], and self-assembly of structures [15]. However, the use of multiagent systems for controlling smart matter is a challenging new application due to the very tight coupling between the computational agents and their embedding in physical space. Specifically, in addition to computational interactions between agents from the exchange of information, there are mechanical interactions whose strength decreases with the physical distance between them.

In this paper we examine how a simple learning mechanism can improve a multiagent control strategy for unstable dynamical systems. This is a particularly challenging problem, for in the absence of controls, the physics of an unstable system will drive it rapidly away from the desired configuration. This is the case, for example, for a structural beam whose load is large enough to cause it to buckle and break [1]. In such cases, weak control forces, if applied properly, can counter departures from the unstable configuration while they are still small. Successful control leads to a virtual strengthening and stiffening of the material. Because the detailed characteristics of the system can vary due to failures and environmental changes, any initial organization for the agents may not remain appropriate for long. Hence, the addition of a learning mechanism can also help the agents respond and adapt to these changes by altering their organizational structure.

Dynamics of Unstable Smart Matter

The devices embedded in smart matter are associated with computational agents that use the sensor information to determine appropriate actuator forces. The overall system dynamics is a combination of the behavior at the location of these agents and the behavior of the material between the agent locations. In mechanical systems, displacements associated with short length scales involve relatively large restoring forces, high frequency oscillations and rapid damping. Hence, they are not important for the overall stability [9]. Instead, stability is primarily determined by the lowest frequency modes. We assume that there are enough agents so that their typical spacing is much smaller than the wavelengths associated with these lowest modes. Hence, the lower frequency dynamics is sufficiently characterized by the displacements at the locations of the agents only. The high-frequency dynamics of the physical substrate between agents serves only to couple the agents' displacements.

The system we studied, illustrated in Fig. 1a, consists of *n* mass points connected to their neighbors by springs. In addition a destabilizing force proportional to the displacement acts on each mass point. This force models the behavior of unstable fixed points: the force is zero exactly at the fixed point, but acts to amplify any small deviations away from the fixed point. This system can be construed as a linear approximation to the behavior of a variety of dynamical systems near an unstable fixed point, such as the inverted pendulums shown in the Fig. 1b. In the absence of control, any small initial displacement away from the vertical position rapidly leads to all the masses falling over. In this case, the lowest mode consists of all the pendulae falling over in the same direction and is the most rapidly unstable mode of behavior for this system. By contrast, higher modes, operating at shorter length scales, consist of the masses falling in different directions so that springs between them act to reduce the rate of falling.

The system's physical behavior is described by

1. the number of mass points *n*
2. the spring constant *k* of the springs
3. a destabilizing force coefficient *f*
4. a damping force coefficient *g*

We also suppose the mass of each point is equal to one. The resulting dynamics of the unstable chain is given by[1]

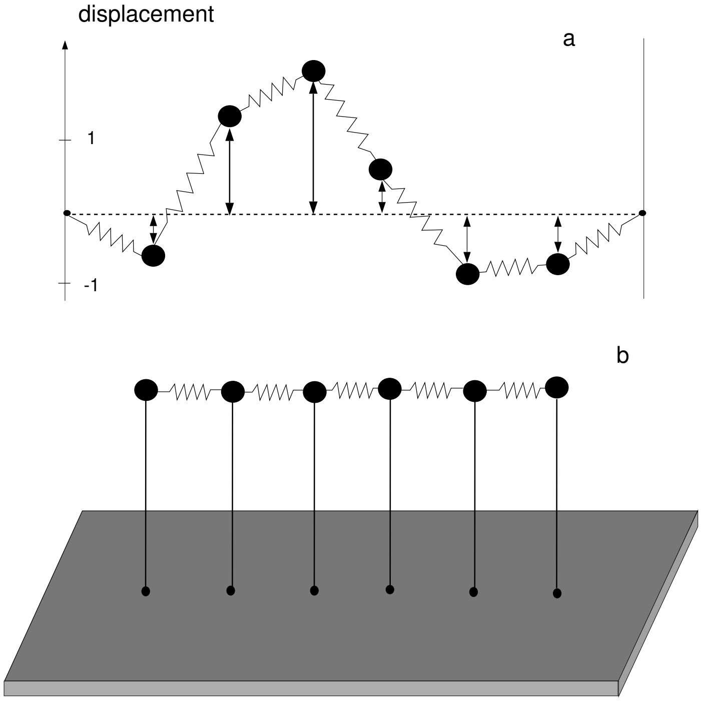

Fig. 1. An unstable dynamical system. a) The unstable chain with the mass points displaced from the unstable fixed point which is indicated by the horizontal dashed line. The masses are coupled to their neighbors with springs, and those at the end of the chain are connected to a rigid wall. b) A chain of upward-pointing pendulums connected by springs as an example of an unstable spatially extended system.

the following equations of motion [7]:

$$\begin{aligned} \frac{dx_i}{dt} &= v_i \\ \frac{dv_i}{dt} &= k(x_{i-1}-x_i) + k(x_{i+1}-x_i) + fx_i - gv_i + H_i \end{aligned} \tag{1}$$

where x_i is the displacement of mass point *i*, v_i is the corresponding velocity, and $x_0 = x_{n+1} = 0$ is the boundary condition. The H_i term in Eq. (1) is the additional control force produced by the actuator attached to mass point *i*.

For these systems, the long time response to any initial condition is determined by the eigenvalues of the matrix corresponding to the right hand side of Eq. (1). Specifically, if the control force makes all eigenvalues have negative real parts, the system is stable [9]. The corresponding eigenvectors are the system's modes. Thus to evaluate stability for *all* initial conditions, we can use any single initial condition that includes contributions from all modes. If there are any unstable modes, the displacements will then grow. We used this technique to evaluate stability in the experiments described below.

A Multiagent Control System

The control problem is how hard to push on the various mass points to maintain them at the unstable fixed point.

[1]We used a standard ordinary-differential-equation solver [14] to determine the controlled system's behaviors.

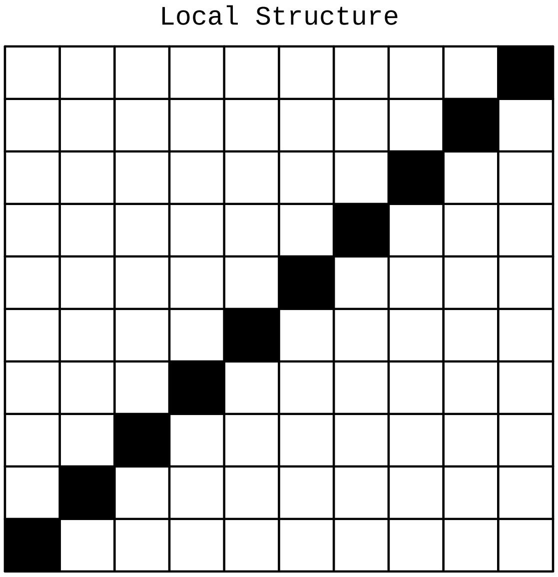

Fig. 2. Purely local organizational structure where every agent considers only his own displacement information.

This problem can involve various goals, such as maintaining stability in spite of perturbations typically delivered by the system's environment, using only weak control forces so the actuators are easy and cheap to fabricate, continuing to operate even with sensor noise and actuator failures, and being simple to program, e.g., by not requiring a detailed physical model of the system.

To focus on the multiagent learning aspects of this problem, we consider a simple control program where each agent exerts a control force against a linear combination of the displacements of the individual agents. Specifically, $H_i = -X_i$ where

$$X_i = \sum_{j=1}^{n} a_{ij} x_j \qquad (2)$$

is a linear combination of the displacements of all mass points that provides information about the chain's state. The values of the a_{ij}'s determine which displacements individual agents pay attention to in determining their response. These values can be viewed as defining the organization of this multiagent system with a_{ij} giving the influence of information at location *j* on the agent at location *i*. We assume the existence of a communication network that allows any agent i to access sensor information at any position j within the system.

As a measure of the control effort employed by the agents, we consider the power P_i used to produce the control force H_i which we take to be simply given by

$$P_i \; = |\, H_i \,| \qquad (3)$$

For example, a purely local organization (Fig. 2) has $a_{ij} = 1$ when $i = j$ and is 0 otherwise. In this case, agent *i* considers only its own displacement x_i. While this organization is very simple to implement, it has the drawback of making the agents respond equally to high and low frequency disturbances. For the unstable chain, generally only the low frequency modes are actually unstable: control force acting against high frequency modes is unnecessary, thus forcing the agents to unnecessarily expend power and exert strong forces. By contrast, if the values of the system parameters are accurately known, a global model of the system can determine the shape of the modes and arrange for the agents to act only against the unstable ones [9]. Unfortunately, in practice, this amount of detailed information will seldom be available, and can change due to environmental contamination of parts of the structure or damage to some of the actuators. Hence it is desirable to arrange a learning mechanism that can automatically adjust the a_{ij}'s to continually improve performance in the face of these changes, and without the requirement of a global analysis of the system's modes. In this context, the agents might start with a simple structure based on estimated or nominal system parameters. They could then learn to improve performance by adjusting their interactions to better fit the particular environment they are in.

Learning Method

In this section, we describe a simple, specific learning mechanism the agents can use to adjust their organizational structure. Specifically, we start with a local structure (Fig. 2) and simulate the control process for a given period of time. Changing some of the a_{ij}'s in the interaction matrix will result in a different controlling performance. By comparing the overall power usage of the given and the changed structure we apply a hill-climbing algorithm and take the structure that performs better for the next learning iteration step.

For the decision which a_{ij}'s to change and how strongly, we apply a method that uses a randomly chosen center a_{ij} within the interaction matrix. We change the magnitude of this interaction by a fixed amount δ, with the sign of the change chosen randomly. We also change its neighborhood in the matrix using a linearly decreasing function with distance, which we measure using the 1–norm, or "taxicab" metric. Specifically, the magnitude of the change applied at location *kl* is

$$f_{kl} = \delta(1 - \epsilon(|k - i| + |l - j|)) \qquad (4)$$

provided this is positive (so the changes in the neighborhood of the center all have the same sign). After each change we normalize the interaction matrix to ensure that the sum of all elements a_{ij} stays constant during the learning process.

Every learning iteration results in a comparison of two different structures. Which structure will be taken is

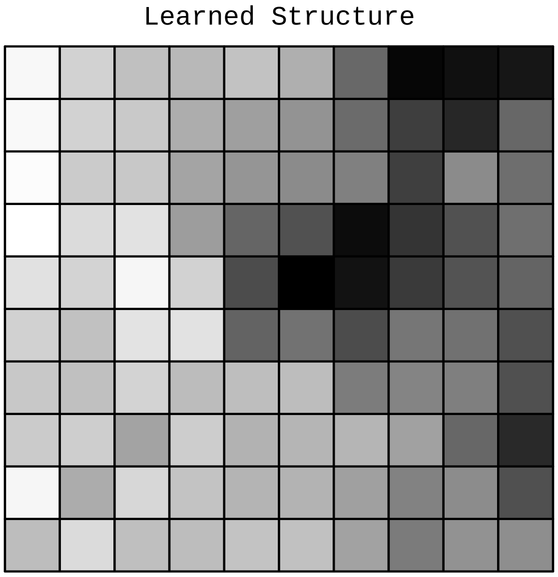

Fig. 3. Final structure, learned by the system after 1500 iteration steps.

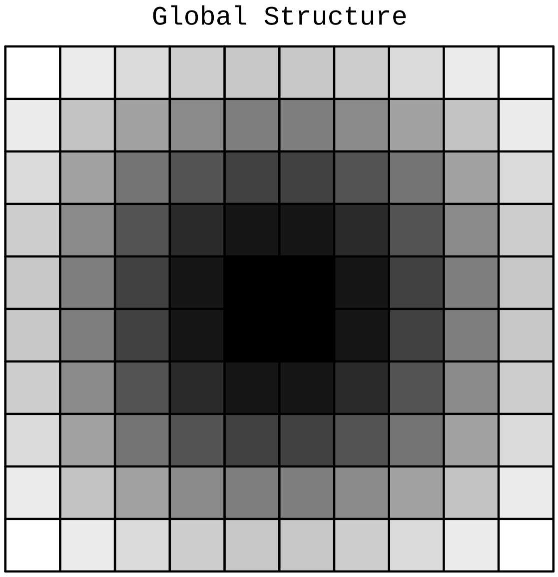

Fig. 4. Global structure that includes a model to push against the unstable first mode of the example system..

simply based on the amount of overall power used to control the system, i.e., $\sum_i P_i$. The improvement in performance depends on the parameters δ and ϵ. Too little changes or the change of only one a_{ij} at a time require a lot of iteration steps. Too many changes on the other hand may represent a new structure where several actions compensate each other and therefore the learning will be inefficient.

Because of the physical coupling within smart matter systems, the method of changing several a_{ij}'s around a given center at one time provides a reasonable compromise between having a fast convergence in performance and a small enough scale for changes to prevent compensating effects. Neighboring a_{ij}'s in the interaction structure correspond to neighbors in physical space. Since physical interactions generally decrease with distance, this choice for updating the agent interactions exploits the locality of interactions in physical space. It thus represents a learning mechanism that is particularly well suited for multiagent systems tightly embedded in physical space.

Results

In this section we present an example of the learning method. Fig. 3 shows the final structure that the system learned after 1500 iteration steps, starting with a local structure (Fig. 2) with amplitudes

$$a_{ii} = 0.025 \tag{5}$$

as an initial condition and parameter values

$$f = 0.01,\ g = 0.1,\ k = 0.05 \tag{6}$$

and

$$\delta = 0.0015,\ \epsilon = 0.6 \tag{7}$$

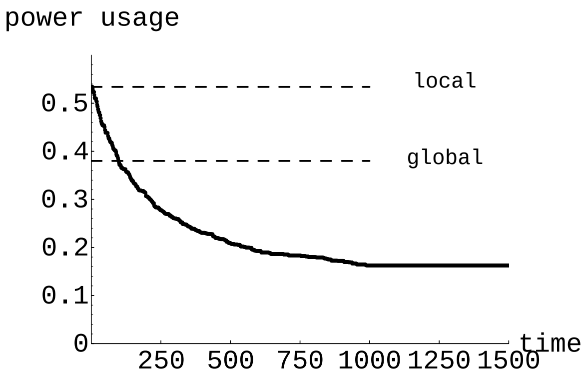

Fig. 5. Decrease in power usage as the system learns to better use its resources.

We found that these values of δ and ϵ led to a significant performance improvement. The power consumption of the overall system is shown in Fig. 5. In every iteration step we integrated Eq. (1) over 25 time units. The final structure has a power usage of 0.16 which is less than one third of the power consumption of the initial local structure. For comparison we also calculated the power usage of a global structure (Fig. 4) that uses a global model where the agents push only against the unstable first mode [9]. This value is about 0.38 and also more than twice as large as the final structure's performance value.

Discussion

In this paper we presented a simple learning mechanism for a multiagent system controlling unstable dynamical systems. This learning mechanism, which takes advantage of the physical locality of agent interactions, improves the performance of simple control strategies. This may be particularly useful for more complex instances of smart matter where conventional global controllers become infeasible due to the difficulties of modeling the system.

There are a variety of ways this work could be extended. First, one appeal of a multiagent approach to controlling smart matter is the ability of such systems to deal with failures. For instance, a relatively good organization for a fully functional system may no longer be appropriate when some actuators break. Thus an interesting experiment in this context would be to see whether the learning mechanism could improve the performance in spite of the failures. More generally, it would be useful to understand the limits of this learning mechanism, e.g., types of smart matter control tasks for which convergence to better organizations becomes very slow.

Although we have chosen a particular performance measure for the learning, there are many other possibilities, such as faster recovery from sudden changes and minimizing the number of active actuators.

The power of multiagent approaches to control lies in the fact that relatively little knowledge of the system to be controlled is needed. This is in stark contrast to traditional AI approaches, which use symbolic reasoning with extremely detailed models of the physical system. However, while providing a very robust and simple design methodology, the distributed nature of the agent-based system suffers from the lack of a high level explanation for its global behavior. An interesting open issue is to combine this approach with the more traditional AI one.

References

[1] A. A. Berlin. *Towards Intelligent Structures: Active Control of Buckling*. PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1994.

[2] A. A. Berlin, H. Abelson, N. Cohen, L. Fogel, C. M. Ho, M. Horowitz, J. How, T. F. Knight, R. Newton, and K. Pister. Distributed information systems for MEMS. Technical report, Information Systems and Technology (ISAT) Study, 1995.

[3] Janusz Bryzek, Kurt Petersen, and Wendell McCulley. Micromachines on the march. *IEEE Spectrum*, pages 20–31, May 1994.

[4] E. H. Durfee. Special section on distributed artificial intelligence. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 21. IEEE, 1991.

[5] K. J. Gabriel. Microelectromechanical systems (MEMS). A World Wide Web Page with URL http://eto.sysplan.com/ETO/MEMS/index.html, 1996.

[6] Les Gasser and Michael N. Huhns, editors. *Distributed Artificial Intelligence*, volume 2. Morgan Kaufmann, Menlo Park, CA, 1989.

[7] Herbert Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, MA, 2nd edition, 1980.

[8] Tad Hogg and Bernardo A. Huberman. Controlling chaos in distributed systems. *IEEE Trans. on Systems, Man and Cybernetics*, 21(6):1325–1332, November/December 1991.

[9] Tad Hogg and Bernardo A. Huberman. Controlling smart matter. Technical report, Xerox PARC, 1996. preprint http://xxx.lanl.gov/abs/cond-mat/9611024.

[10] Bernardo A. Huberman and Tad Hogg. The behavior of computational ecologies. In B. A. Huberman, editor, *The Ecology of Computation*, pages 77–115. North-Holland, Amsterdam, 1988.

[11] Victor Lesser, editor. *Proc. of the 1st International Conference on Multiagent Systems (ICMAS95)*, Menlo Park, CA, 1995. AAAI Press.

[12] Kai Nagel. Life times of simulated traffic jams. *Intl. J. of Modern Physics C*, 5(4):567–580, 1994.

[13] A. C. Sanderson and G. Perry. Sensor-based robotic assembly systems: Research and applications in electronic manufacturing. *Proc. of IEEE*, 71:856–871, 1983.

[14] L. F. Shampine and M. K. Gordon. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*. W. H. Freeman, San Francisco, 1975.

[15] Elisabeth Smela, Olle Inganas, and Ingemar Lundstrom. Controlled folding of micrometer-size structures. *Science*, 268:1735–1738, 1995.

[16] David M. Upton. A flexible structure for computer controlled manufacturing systems. *Manufacturing Review*, 5(1):58–74, 1992.

[17] Brian C. Williams and P. Pandurang Nayak. Immobile robots. *AI Magazine*, 17(3):17–35, 1996.