1160

Escaping Static and Cyclic Behavior in Autonomous Agents

Brian Yamauchi[1] and Randall Beer[1,2]

Department of Computer Engineering and Science[1]
Department of Biology[2]
Case Western Reserve University
Cleveland, Ohio 44106

yamauchi@alpha.ces.cwru.edu
beer@alpha.ces.cwru.edu

Abstract

Animals possess behaviors that have evolved to handle routine interactions between an organism and its environment. Likewise, most autonomous robots are controlled by a set of behaviors designed to deal with routine interactions between a robot and its environment. A fundamental difference between animals and robots is that animals also have strategies for dealing with the situations that arise when routine behavior fails, and most robots do not. We call these strategies *exploratory behaviors*, and we investigate two of these behaviors in this paper -- one for detecting and escaping behavioral stasis, and another for detecting and escaping behavioral cycles. These behaviors are applied to the task of finding food using chemotaxis in an environment containing obstacles, and we present experimental results describing the performance of agents using these behaviors.

Exploratory Behavior

We are interested in providing robots with the ability to deal with unexpected events that is common in most animals. Animals, like robots, have a repertoire of behaviors for dealing with their routine interactions with their environment. However, unlike most robots, animals also have behaviors for dealing with non-routine interactions with their environment.

We call these behaviors for non-routine interaction with the environment *exploratory behaviors*, and we would like to provide autonomous agents with similar capabilities. There are three main questions we would like to answer with regard to exploratory behavior:

1. How can you detect when routine behavior has failed?
2. Once routine behavior has failed, what can you do to explore the space of possible actions?
3. How can you learn which actions work and which do not -- without requiring excessive numbers of learning trials?

In this paper, we describe our first efforts at addressing questions one and two. We describe ways to detect unproductive behavior based on unchanging or oscillatory sensory input, and ways of exploring new behavior by adding increasing amounts of randomness to motor outputs. We then compare the performance of agents with and without these exploratory behaviors.

Detecting Behavioral Stasis

Intuitively, static behavior refers to behavior that is unproductive and unchanging. More formally, behavioral stasis is a condition of the agent/environment system such that the agent does not receive any reward and such that the rate of change in the agent's sensory input is low or zero.

The level of stasis can be represented by an internal state variable that is a function of the rate of change of an agent's sensory inputs and the reward received. The normalized rate of sensor change *ds/dt* is given by the following equation:

$$\frac{ds}{dt} = \frac{1}{n\Delta t}\sum_{i=1}^{n}\left|s_{i,t} - s_{i,t-\Delta t}\right|$$

where n is the number of sensors, $s_{i,t}$ is the value of sensor i at time t, and Δt is the duration of each simulation time step.

The rate of change in the stasis level $d\sigma/dt$ is defined by:

$$\frac{d\sigma}{dt} = \begin{cases} -k_1 & \text{if } \frac{ds}{dt} < \tau_s \text{ and } r = 0 \\ k_2 & \text{if } \frac{ds}{dt} \geq \tau_s \text{ and } r = 0 \\ -\sigma & \text{if } r \neq 0 \end{cases}$$

where σ is the current stasis level, τ_s is the sensor change threshold, r is the reward currently being received, and k_1 and k_2 are constants.

If the agent is being rewarded, its stasis level is reset to zero. If the agent is not being rewarded and the normalized rate of sensor change exceeds the threshold, the stasis level is decreased by k_1. Otherwise, the agent's stasis level is increased by k_2.

Detecting Behavioral Cycles

A behavioral cycle consists of a sequence of agent interactions with the environment that repeats at regular intervals. From the agent's point-of-view, this cycle appears as a repeating sequence of sensory input.

A simple way to detect such cycles is to consider the intervals between sensor events, where an event occurs when a sensor value exceeds a threshold. If the intervals between such events is constant (or near constant) over a long period of time, then the agent is likely to be trapped in a behavioral cycle.

A separate threshold exists for each sensor. The threshold value can be computed by integrating the sensor's input values over a long period of time:

$$\frac{d\tau_i}{dt} = \alpha_\tau\left(-\tau_i + s_i\right)$$

where τ_i is the threshold value for sensor i, s_i is the input value of sensor i at the current time, and α_τ is a time constant that determines how fast the threshold responses to changes in sensor input ($0 \leq \alpha_\tau \leq 1$). A small value of α_τ results in a threshold that changes slowly, while a large value indicates a threshold that changes rapidly. In general, α_τ should be chosen to be much larger than the expected cycle interval.

When a sensor value exceeds its threshold, the interval between the current time t and the previous event e_1 is compared with the interval between the previous event and the event e_0 that preceded e_1. If these intervals are sufficiently similar, the internal state variable for measuring the level of cyclic behavior is increased.

In addition, the internal state variable is also constantly decreasing by a small decay increment regardless of whether an event occurs. So the update rule for the cyclic level c is:

$$\frac{dc}{dt} = -k_3 + k_4 \sum_{i=1}^{n} \phi_i$$

$$\phi_i = \begin{cases} 1 \quad \text{if} & s_{i,t-\Delta t} < \tau_i \ \textit{and} \\ & s_{i,t} \geq \tau_i \ \textit{and} \\ & (t-e_1)-(e_1-e_0) < w \\ 0 & \textit{otherwise} \end{cases}$$

where k_3 is the decay rate, k_4 is the rate of increase, n is the number of sensors, $s_{i,t}$ is the value of sensor i at time t, Δt is the interval between simulation time steps, and τ_i is the threshold for sensor i. w is a constant determining the maximum difference in length between two successive event intervals. Intervals differing in duration by more than w will not cause an increase in the cyclic level.

Foraging Task

Environment

We have applied these ideas to an autonomous agent that forages for food in a (simulated) environment containing barriers. Food patches in this environment emit chemicals that can be detected by sensors on the agent's body. The barriers in the environment prevent motion, but are permeable to chemicals. The agent's task is to find all of the food, maneuvering around barriers as necessary. Figure 2 depicts the environment for the foraging task. The small circle is the agent -- the short line in the circle indicates the direction that the agent is facing. The large black circles are food patches, and the black lines are the barriers.

Static and Cyclic Behavior in Foraging

This task and environment were selected because a simple reactive strategy using chemotaxis is capable of generating both static and cyclic behavior. Chemotaxis refers to a behavior found in animals where the organism orients toward and moves in the direction of the chemical gradient. Using two sensors -- one on either side of the agent's body -- this can be accomplished by turning in the direction of the stronger chemical concentration and moving forward.

While this strategy can usually lead an agent to food, it can fail when barriers are introduced into the environment. Suppose the agent is separated from a food patch by a barrier. Since barriers are permeable to chemical signals, the agent will orient toward the food, move forward until it hits the barrier, and continue attempting to move forward forever. (An explicit sensor could have been added to detect barrier contacts, but our interest was on how to escape static behavior when such sensors are unavailable.)

Second, since the agent has a limited turn rate, and thus a non-zero turn radius, it can become trapped in loops around food patches, where it is constantly turning toward the food and constantly missing.

Neural Circuit Model

The agent is controlled by a dynamical neural network interfaced to a stasis detector and a cycle detector. The state equation for neuron *i* is:

$$\frac{dv_i}{dt} = \frac{-v_i + \sum^{j} w_{ij} f(v_j) + I_i}{T_i}$$

where v_i is the voltage of neuron *i*, w_{ij} is the weight of the link connecting neuron *j* to neuron *i*, I_i is the current injected into neuron *i* from external sources (i.e. sensors), T_i is the time constant of neuron *i*, and $f(v_i)$ is the firing rate of neuron *i*.

Each neuron has a linear threshold firing function:

$$f(v_i) = \begin{cases} 0 & v_i < \tau_i \\ v_i - \tau_i & \tau_i \le v_i \le \tau_i + 1 \\ 1 & v_i > \tau_i + 1 \end{cases}$$

where τ_i is the firing threshold of neuron *i*.

Sensor Model

The strength of the chemical signal from a food patch is proportional to the size of the patch, and drops off as the inverse square of the distance from the center of the patch. The strength of the signal received by the agent's sensors is equal to the sum of the signals from all of the food patches.

$$c_l = \sum^{i} \frac{p_i}{(x_i - x_l)^2 + (y_i - y_l)^2} \qquad c_r = \sum^{i} \frac{p_i}{(x_i - x_r)^2 + (y_i - y_r)^2}$$

where c_r and c_l are the inputs to the right and left chemosensors (RCS and LCS in Figure 2), p_i is the size of food patch *i*, (x_r, y_r) is the location of the right sensor, (x_l, y_l) is the location of the left sensor, and (x_i, y_i) is the location of food patch *i*.

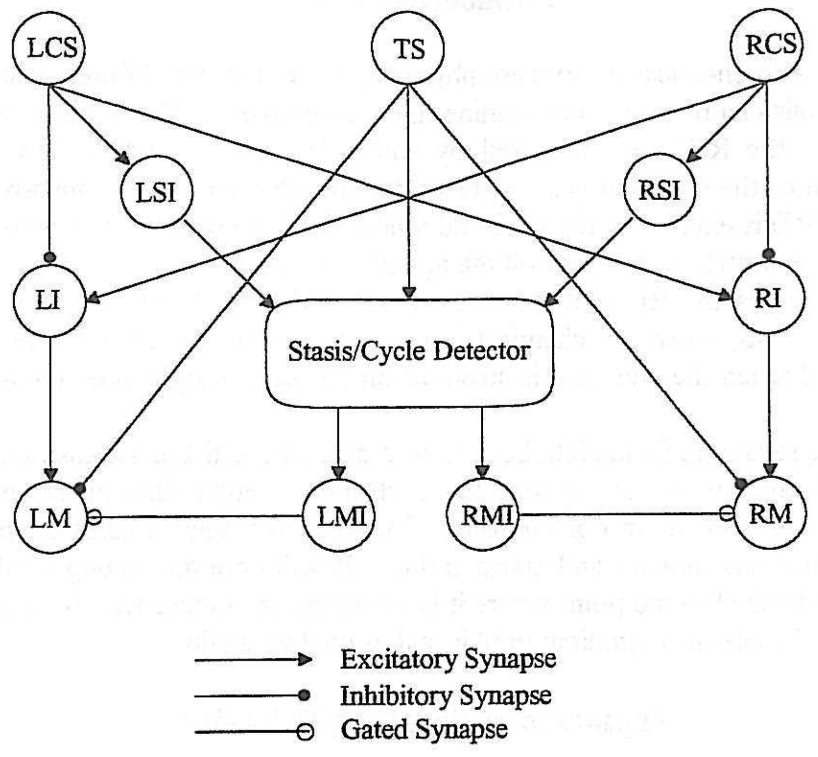

Figure 1: Neural Circuit

In addition, the agent has a taste sensor (TS) that detects when it is feeding. The agent feeds whenever it is in contact with a food patch. This sensor is set to 1 if the agent is feeding, 0 otherwise.

Locomotion Model

The agent's motion is controlled by two motor neurons (LM and RM) whose outputs control two motor effectors -- simulating wheels located on either side of the agent's body. The agent's velocity is the average of the two effector values, simulating inertialess motion on a high-friction surface:

$$v = k_v\left(\frac{m_l + m_r}{2}\right)$$

where v is the agent's velocity, m_l and m_r are the left and right motor neuron firing rates, and k_v is a constant.

The agent's change in heading is proportional to the difference between the left and right motor neuron firing rates:

$$\frac{d\theta}{dt} = k_\theta(m_l - m_r)$$

where θ is the agent's heading and k_θ is a constant.

Chemotaxis Circuit

Since the two chemical sensors are physically separated, the difference between the chemical signals can be used to determine the food gradient. The left interneuron (LI) is excited by the RCS and inhibited by the LCS, so it is active only when the concentration of the chemical is stronger on the agent's right side. Similarly, the right interneuron (RI) is excited by the LCS and inhibited by the RCS, so it is active only the chemical concentration is stronger on the agent's left side.

The LI excites the left motor neuron (LM) and the RI excites the right motor neuron (RM). So, when the chemical is stronger on the agent's left side, the agent turns left, and when the chemical is stronger on the agent's right side, the agent turns right.

The taste sensor (TS) inhibits both motor neurons, and the synapse between the touch sensor and the motor neurons has a stronger weight than those between the chemical sensors and the motor neurons. So, when the agent makes contact with a food patch, it stops moving and starts eating. It will continue eating until the food patch has diminished to the point where it is no longer in contact with the agent. Then it will follow the chemical gradient until it makes contact again.

Exploratory Foraging Behavior

Stasis/Cycle Detector

The left and right integrator neurons (LSI and RSI) integrate the output from the sensory neurons over time. These neurons filter out transient variations in the sensory input, to prevent interference with stasis/cycle detection.

The stasis/cycle detector applies the equations described above to the input from LSI, RSI, and TS to update the state variables for the stasis level and cyclic level. Both of these variables are initially zero.

Gated Random Current Injection

Random bursts of current are constantly injected into the left and right motor integrator neurons (LMI and RMI). These neurons integrate these bursts over time, allowing large differences in activation to build up slowly.

The synapses between the motor integrator neurons and the motor neurons (LM and RM) are gated by an exponential function of the sum of the current stasis and cyclic levels.

$$w = \begin{cases} w_{\max} c^{\lambda} & \sigma < \tau_\sigma \\ w_{\max}\left(c + \dfrac{\sigma - \tau_\sigma}{1 - \tau_\sigma}\right)^{\lambda} & \sigma \geq \tau_\sigma \end{cases}$$

where w is the weight of the synapse, w_{max} is the maximum synapse weight, σ is the stasis level, c is the cyclic level, τ_σ is the stasis threshold, and λ is a constant.

The strength of this synapse increases exponentially with the cyclic level and with the stasis level once it has exceeded a minimum threshold. The parameter λ controls the steepness of this increase.

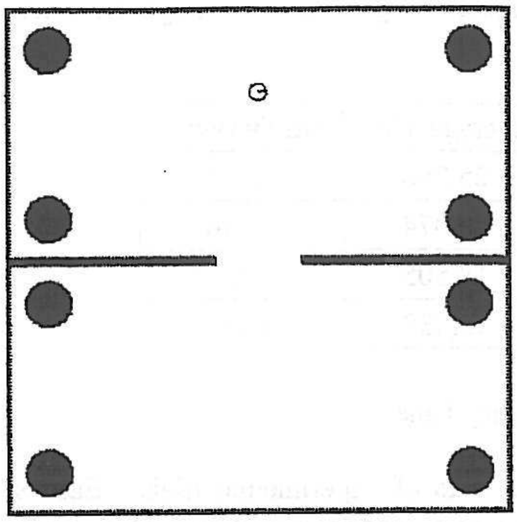
Figure 2: Foraging Environment

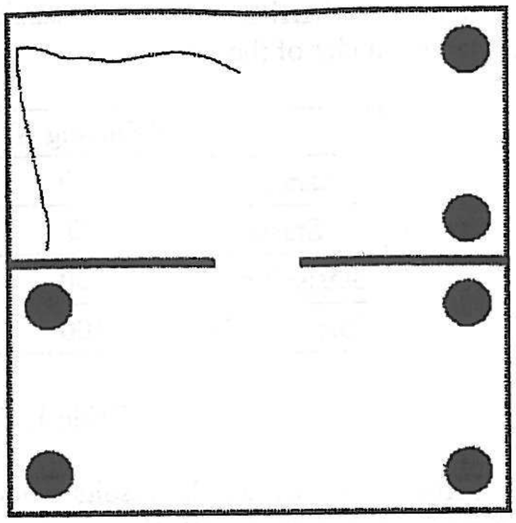
Figure 3: Reactive Agent Path

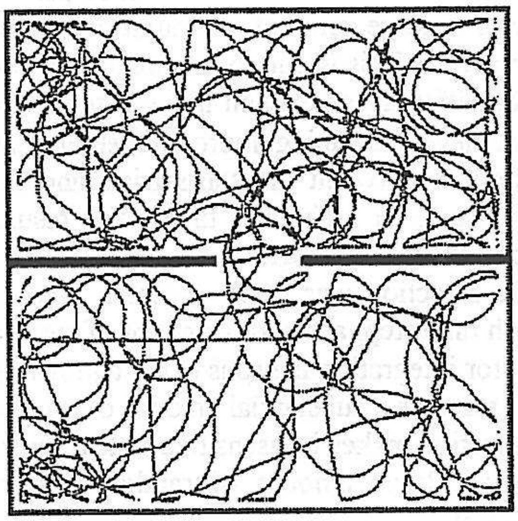
Figure 4: Agent Path
(Random Motor Noise)

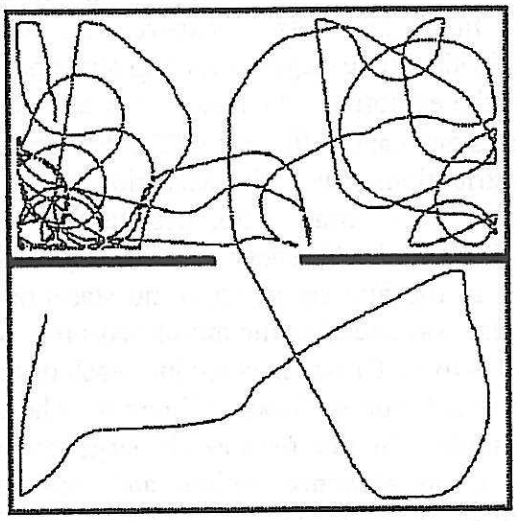
Figure 5: Agent Path
(Stasis/Cycle Detection)

So, when static or cyclic behavior is initially detected, only small variations will be induced in the motor outputs of the chemotaxis network. As the stasis and cyclic levels increase, the amount of random injection will increase. This allows the stasis/cycle detector to use the minimum amount of randomness necessary to escape static and cyclic behavior, while still allowing it to almost completely override the reactive network, if necessary.

Experimental Results

A purely reactive strategy, using only the chemotaxis network, was never able to find all four patches. Figure 3 depicts a typical run using the reactive strategy. The agent starts in the central upper section of the environment and follows the chemical gradient to consume the two food patches in the upper left region. Then the agent tries to follow the gradient toward the food in the lower section, but its motion is blocked by the barrier. The agent remains stuck, trying to move through the barrier, for the remainder of the run.

	# Finding All	Average Time	Std Deviation
Random	100	25,868	12,999
Stasis	90	15,074	9,476
Stasis+Noise	100	18,505	13,869
Stasis+Cyclic	100	15,438	8,587

Table 1: Foraging Time

Table 1 summarizes the results from four sets of experimental trials. Each set consisted of 100 runs with the agent starting in a random orientation at a random location in the environment. The average time is the time required to find and consume all four food patches. All time measurements are in simulation time steps.

In all four sets of experiments, there was a large amount of variation in the individual time required for a given experimental run. This is due to the random nature of the exploratory behavior. Nevertheless, the differences between the average times are significant at the $\alpha = 0.05$ (95% confidence) level (assuming approximately normal distributions), with the exception of the difference between the stasis trials and the stasis+cyclic trials. So, despite the large variance in individual times, the results indicate real differences in the average efficiency of these strategies.

In the first set of trials, no stasis or cycle detection was used. Instead, random noise was added to the motor neurons. At each time step, a random amount of current (-1.0 to +1 0) was injected into each of the motor integration neurons (LMI and RMI). A sample run is shown in figure 4. The agent requires a substantial amount of time to complete the run, because the large random injection makes it insensitive to low levels of chemical concentration, and thus requires a large amount of random search. However, the agent eventually finds all of the food.

In the second set of trials, stasis detection alone was added to the chemotaxis network In these experiments, the agent was able to find all four patches quickly in 90 out of 100 trials The time required was substantially less for the agent that used stasis detection than for the agent that used random noise. However, in 10 of the stasis detection trials, the agent was trapped in cyclic behavior -- looping around a food patch, trying to turn toward the center, but limited by the agent's minimum turn radius. This behavior continued until the maximum time limit (one million time steps) was exceeded.

In the third set of trials, small amounts of random motor noise (-0.05 to +0.05) were added to stasis detection and chemotaxis. Less noise was required than in the first set of trials. In the first set, sufficient noise was required to offset the strong chemical gradient compelling the agent to move across the barrier. In the third set, only a small amount of noise was required to disrupt the looping behavior. In this set, the agent was successful in all 100 trials.

In the fourth set of trials, both stasis and cycle detection were added to the chemotaxis network (without motor noise). This agent was able to find all food patches in all 100 trials, and it had the fastest average time of the agents that were 100% successful.

A sample run using stasis and cycle detection is displayed in figure 5. The agent starts in the upper portion of the environment, near to the left wall. It reactively follows the chemical gradient to consume the two food patches in the upper left region, then it tries to move down toward the food in the lower portion of the screen.

The agent's motion is blocked by the barrier, but the agent detects that it is trapped in static behavior, so it injects slowly increasing amounts of random current into its motor neurons. This triggers a random search behavior which eventually leads the agent to the upper right region. When its sensors detect the increase in chemical concentration, the random injection is decreased, and the agent follows the gradient to the two upper right food patches.

The agent then tries to move down again, toward the remaining food patches in the lower half of the environment. Again its motion is blocked, stasis is detected, and random search begins. Eventually, the agent wanders through the gap in the barrier into the lower region. Upon detecting the increase in chemical concentration, the reactive behavior takes control again, and the agent follows the gradient to consume the remaining food.

Related Work

Our research has the goal of building agents that can react adaptively to unpredictable situations. This is a goal shared by many of the research projects related to learning and evolution in autonomous agents.

Many of these projects have involved the use of reinforcement learning in agents whose control systems are decomposed on a behavioral basis. Maes has used reinforcement learning to coordinate multiple behaviors for legged locomotion [Maes 90] and to learn to control the interactions between multiple behavioral modules [Maes 91]. Mahadevan and Connell have used reinforcement learning to learn the behaviors involved in a box-pushing task [Mahadevan 90]. Kaelbling has used reinforcement learning to learn obstacle avoidance and phototaxis without an explicit decomposition into separate behaviors [Kaelbling 91].

In addition, Nehmzow and Smithers have used behavioral task decomposition in combination with perceptrons and Kohonen networks to learn a number of mobile robot behaviors, including obstacle avoidance, wall-following, and location recognition [Nehmzow 91].

Our research is also related to the work being done by Beer and Gallagher on using genetic algorithms to evolve dynamical neural networks for controlling autonomous agents [Beer 92]. In particular, our work relates to the questions of how these networks can generate behavior that can adapt to a wide variety of environments, and issues related to the role of agent internal state in learning and adaptation.

However, our research differs from all of these in that its primary focus is not on how to initially acquire behavioral capabilities -- whether through design, learning, or evolution -- but on how to deal with the situations that arise when these behaviors fail.

Conclusions

These experiments have shown that it is possible to use the algorithms described to detect simple forms of static and cyclic behavior in an agent controlled by a dynamical neural network and operating in a continuous, changing environment. These experiments also showed that it was possible to escape static and cyclic behavior through the injection of random motor current, and that the agent could operate more effectively when this random current was modulated by the level of detected static or cyclic behavior.

Constant injections of random motor current allowed the agent to escape static and cyclic behavior, but at the cost of constant interference with chemotaxis and substantially reduced foraging performance. In contrast, gated random injections allowed the agent to forage without interference under normal circumstances and also to escape static and cyclic behavior when either occurred. This resulted in the ability to deal with situations that the purely reactive chemotaxis agent was unable to handle, while incurring a very small performance cost in those situations that could be handled reactively.

Acknowledgments

We would like to thank the other members of the CWRU Autonomous Agents Group, Leslie Picardo, John Gallagher, and Alan Calvitti, who have provided helpful comments on this research.

References

Beer, Randall and Gallagher, John 1992. "Evolving Dynamical Neural Networks for Adaptive Behavior", *Adaptive Behavior*, Vol. 1, No. 1, Summer 1992.

Kaelbling, Leslie 1991. "An Adaptable Mobile Robot", *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Francisco Varela and Paul Bourgine, ed., Cambridge, MA: MIT Press.

Maes, Pattie and Brooks, Rodney 1990. "Learning to Coordinate Behaviors", *Proceedings of AAAI-90.*

Maes, Pattie 1991. "Learning Behavior Networks from Experience", *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Francisco Varela and Paul Bourgine, ed., Cambridge, MA: MIT Press.

Mahadevan, Sridhar and Connell, Jonathan 1990. "Automatic Programming of Behavior-Based Robots Using Reinforcement Learning", Research Report RC 16359, IBM T. J. Watson Research Center, Yorktown Heights: NY.

Nehmzow, Ulrich and Smithers, Tim 1991. "Using Motor Actions for Location Recognition", *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Francisco Varela and Paul Bourgine, ed., Cambridge, MA: MIT Press.