Learning Dynamics: System Identification for Perceptually Challenged Agents

Kenneth Basye
Department of Mathematics and Computer Science
Clark University, Worcester, MA 01610

Thomas Dean* Leslie Pack Kaelbling†
Department of Computer Science
Brown University, Providence, RI 02912

Abstract

From the perspective of an agent, the input/output behavior of the environment in which it is embedded can be described as a dynamical system. Inputs correspond to the actions executable by the agent in making transitions between states of the environment. Outputs correspond to the perceptual information available to the agent in particular states of the environment. We view dynamical system identification as inference of deterministic finite-state automata from sequences of input/output pairs. The agent can influence the sequence of input/output pairs it is presented by pursuing a strategy for exploring the environment. We identify two sorts of perceptual errors: errors in perceiving the output of a state and errors in perceiving the inputs actually carried out in making a transition from one state to another. We present efficient, high-probability learning algorithms for a number of system identification problems involving such errors. We also present the results of empirical investigations applying these algorithms to learning spatial representations.

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601, by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436.

†This work was supported in part by a National Science Foundation National Young Investigator Award.

1 Introduction

System identification refers to inferring a model of the dynamics governing an agent's interaction with its environment. For instance, we might wish to infer a model of how fluctuations in the output of a parts supplier affect production for a factory or how an assembly robot interacts with the other devices in its work cell.

The inferred model might correspond to a system of differential equations, a set of production rules, or a set of states and transition probabilities for a stochastic process. The model is useful insofar as it enables the agent to predict consequences of performing actions in its environment. Such predictions might be used in planning, spatial inference, or diagnostic reasoning.

System identification has been studied in a variety of disciplines including control theory, neural networks, and automata theory. We focus on learning representations of environments that can be characterized as deterministic finite-state automata. There is a large literature even on this restricted problem, a portion of which is summarized in this paper. Our results address the effects of uncertainty on computational complexity. Our objective is to produce learning algorithms that infer accurate models with high probability in polynomial time when faced with noise in observing the inputs and outputs that determine the agent's interaction with its environment.

We are interested in how agents interact with their environments and, in particular, how uncertainty in observation complicates such interactions. We have chosen to focus on system identification as it appears to be critical in facilitating a wide range of interactions. It is clear that system identification is a means and not an end; however, we believe that studying system identification in isolation provides insight into many problems in which such identification plays a supporting role. Our basic findings are that uncertainty in observation is annoying and requires somewhat more bookkeeping but asymptotically it is not that hard to cope with. However, learning is hopeless in environments lacking any structure. Useful structure in the form of reasonably distributed landmarks or short sequences of distinctive features makes learning relatively easy.

It is certainly possible to function adequately even optimally without the use of a model. There are environments, however, in which having some sort of a model can help enormously. Perhaps, the clearest example of the utility of a model is in learning maps of large-scale

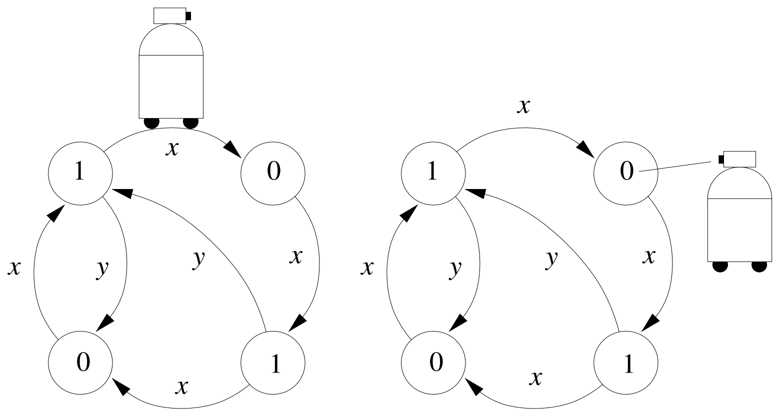


Figure 1: An agent interacting with its environment

space to support path planning. A dynamical model can also speed learning plans [Sutton, 1990] by allowing an agent to simulate its actions and the environment's reactions. Clearly there are tradeoffs involved in learning dynamical models; exactly what is worth learning will depend on the tasks of the agent. Again, we avoid addressing those tradeoffs in this paper (but see [Dean *et al.*, 1990]) in order to focus on basic issues in how uncertainty in observation affects learning.

2 Modeling Dynamical Systems with Automata

We model dynamical systems as deterministic finite-state automata (DFAs). Figure 1 shows the state-transition graph for a DFA in which the inputs to the DFA are the agent's actions and the outputs from the DFA are the agent's perceptual inputs. We are interested in learning the *discernable* structure of real environments, where discernable is defined in terms of the agent's perceptual capabilities. Discernability does not require that, from the information available in a state, the agent can uniquely identify that state, but rather that there exists some sequence of actions and observations that can be used by the agent to distinguish any two states.

For instance, the states of an automaton might correspond to the agent being in one of many locations in an office building, in which locations correspond to junctions where hallways meet. In this case, the observed outputs might correspond to the number of hallways incident on a junction, and the inputs to actions for traversing incident hallways. As another example, consider learning the structure of a voice-mail system. Here the states might correspond to various menus and services, actions to keys pressed by the user, and outputs to the announcements made at each state.

The DFA need not represent the whole of the agent's interaction with its environment; separate models could be used for different aspects of the interaction. We assume the state space has been reduced to a manageable size by careful choice of perception and action primitives. Actions are encapsulated abstract behaviors that serve to limit the agent's options for response in a given state. The set of possible observations is kept small through the use of perceptual apparatus that act as filters, thereby introducing equivalence relations on perceptual experiences.

Historically, AI researchers have kept the state space implicit, specifying only a set of state variables or fluents. In our view, the agent's observations need not correspond to observations of state variables and, even if this is desirable, the set of states of the automaton need not correspond to the cross product of the sets of values for all the state variables. We see the world as consisting of a relatively small number of perceptually distinguishable states. We admit that determining such state-space-reducing perception and action primitives requires a great deal of insight into the problem, and we offer no general advice on how to obtain such primitives. We claim, however, that without such primitives, learning will be very difficult.

Even in the deterministic case, uncertainty arises due to the fact that the observations available in a state do not uniquely determine that state. In this paper, we are particularly interested in stochastic sources of uncertainty. We allow there to be a noise process that occasionally results in the agent observing something other than the true output at a state or realizing one action while attempting to execute some other action. Figure 2 illustrates both sorts of errors. Our polynomial-time performance results apply for quite extreme forms of uncertainty, but predict fairly poor performance in relatively benign environments. Our empirical investigations, however, indicate that our algorithms perform much better than our current theoretical bounds predict for benign environments of the sort we expect robots to encounter in the real world.

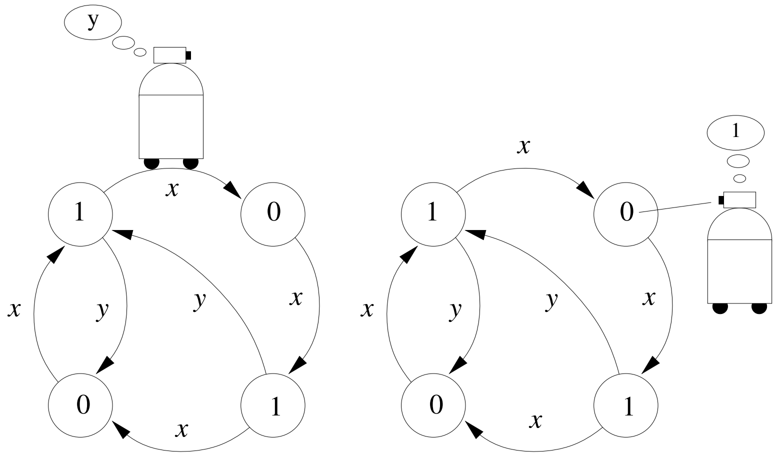


Figure 2: Noisy observations of inputs and outputs

3 Formal Model

In order to model system identification as inferring the structure of finite state automata, we now introduce an extension of the familiar definition of finite state automata. Recall that there are two varieties of finite state automaton; in both versions, output follows some action, and the state reached by the action depends on the action taken and the previous state. In the Moore model, output depends only on the state reached by the action, whereas, in the Mealy model, output depends on both the previous state and the action taken. Alternatively, one may think of the Mealy model as having outputs that depend on the current state and the previous action. In terms of system identification, which model is appropriate depends on the nature of the agent's sensing systems and in particular on whether sensations depend in some way on the previous action. We use Moore automata for our model, which implies that our sensations for a given state are the same regardless of how we got to that state.

In this model, we explicitly distinguish between the actual states and actions of the automaton and the agent's possibly erroneous view of them. Thus, in any given true state of the environment, the agent observes a *label*, which may or may not be an accurate reflection of the state.[1] Similarly, an agent generates *commands* to the environment; these

[1] It would be more technically correct to define an output alphabet for the automaton, give a function

commands have a nominal correspondence to real actions, but the corresponding action is not always taken.

In order to make use of probabilistic functions as a means of modelling uncertainly, we introduce the following notation: for any finite set S, let

$$F_S = \{f | f : S \rightarrow [0,1], \sum_S f(s) = 1\} \ ,$$

the set of probability density functions (PDFs) over S.

The structure of the agent's environment and its interaction with that environment is specified by the tuple $\mathcal{E} = (Q, B, L, C, \delta, \phi, \psi)$, where

- Q is a finite nonempty set of states,
- B is a finite nonempty set of basic actions,
- L is a finite nonempty set of observable labels,
- C is a finite nonempty set of executable commands,
- $\delta : Q \times B \rightarrow Q$, is the *state transition function*,
- ϕ is the probabilistic observation function, $\phi : Q \rightarrow F_L$, mapping each state into a distribution over possible observed labels, and
- ψ is the probabilistic action function, $\psi : C \rightarrow F_B$, mapping each executable command into a distribution over possible actions.

We write n for $|Q|$. We write $A = B^*$ for the set of all finite sequences of basic actions from B. We extend the state transition function δ to such sequences in the usual way by defining $\delta(q, \lambda) = q$, and $\delta(q, ab) = \delta(\delta(q, a), b)$ for all $q \in Q$, $a \in A$, and $b \in B$, where λ is the empty sequence. We write qa as shorthand for $\delta(q, a)$, the state resulting from execution of sequence a from state q.

mapping states to outputs, then describe how the outputs generate observable labels that are sometimes not correct. For simplicity in the following treatment, we have chosen to collapse these processes into one, giving an account of how states give rise directly to possibly erroneous labels.

We assume that every action may be executed in every state, so the function δ induces a set E of labelled edges (q_1, b, q_2) for every $q_1, q_2 \in Q$ and $b \in B$. An automaton is *strongly connected* if for any two states $q_1, q_2 \in Q$, there is some sequence $a \in A$ such that $q_1 a = q_2$.

We write $q\langle a\rangle$ to denote the sequence of outputs of length $|a|+1$ resulting from executing the sequence a starting in state q, beginning with the output at state q. For example, if $a = b_0 b_1 b_2 \ldots b_n$, then $q\langle a\rangle = \langle \phi(q), \phi(qb_0), \phi(qb_0b_1), \ldots, \phi(qa)\rangle$.

Q, B, and δ specify the system itself apart from any agent. Note that the state transition function is deterministic; we are concerned with worlds or systems that have a fixed structure. The remaining elements represent the agent's ability to observe and act in its environment. When the agent reaches some state $q \in Q$, it observes a label drawn from the distribution $\phi(q)$. Similarly, when it is in state q and attempts to perform command c, it performs an action drawn from the distribution $\psi(c)$.

We assume that there is some "correct" observation at each state, and some "correct" action for each command. It is therefore useful to distinguish between a "correct" action or observation function and a "noisy" version of that function. We write ϕ^* (ψ^*) for a deterministic action (observation) function, which may be thought of as mapping each state (command) to a PDF in which some element has probability 1. Clearly, our inference procedures will have wider applicability if they require few assumptions about the stochastic functions governing observation and action. However, we certainly cannot expect to be able to learn in the presence of arbitrarily malicious noise. Instead, we exclude such situations by assuming that observation and action functions behave correctly with probability above some threshold value and that the distributions governing the errors made in the remaining cases are stationary (that is, they do not change during the lifetime of the agent). In some cases, it is also necessary to impose additional restrictions on the error distribution; these will be made explicit where they are required.

When each state has a different output under the correct observation function, we say that the observation function is *unique*. While not all environments present agents with unique observations, we shall see that unique observation functions provide additional structure that allows agents to infer automata in the presence of stochastic forms of uncertainty. Even in environments without unique observation functions, it may be the case that some states have unique outputs. We refer to such states as *landmarks*. In order to make use of landmarks the agent must not only be able to make a unique observation, it must also be capable of determining that the observation made *is* unique, that is, be capable of recognizing landmarks as landmarks.

When there are no landmarks, it becomes necessary to distinguish states by considering the outputs of sequences of states. In the following discussion, sequences of outputs are understood to arise from deterministic observations, that is, from the observation function ϕ^*. A sequence $a \in A$ is said to *distinguish* q_1 and q_2 if and only if $q_1\langle a\rangle \neq q_2\langle a\rangle$. The notion of distinguishability is extended to automata in the following way: M_1 and M_2 are distinguishable if there is a state q of M_1 (or M_2) and some action sequence a such that for all states q' of M_2 (or M_1), $q\langle a\rangle \neq q'\langle a\rangle$. An automaton is said to be *reduced* if, for all pairs of states $q_1 \neq q_2 \in Q$, there exists an action sequence that distinguishes them. The class of automata indistinguishable from a given strongly-connected automaton M has a unique (up to isomorphism) reduced member that is also strongly connected [Moore, 1956]; this automaton has the minimum number of states. We consistently assume that the environments we are attempting to learn are reduced, since there would be no experiment we could perform that would tell us otherwise. Note that this requirement does not mean there is one sequence that distinguishes all pairs of states. However, if there is a single sequence $a \in A$ that distinguishes all non-identical pairs of states, then a is called a *distinguishing sequence* for M. More precisely, $a \in A$ is a distinguishing sequence if, for all $q_1, q_2 \in Q$, $q_1\langle a\rangle = q_2\langle a\rangle \Leftrightarrow q_1 = q_2$. The outputs resulting from the execution of a distinguishing sequence provide a unique signature for each state in an automaton, but note that the signature for a given state can only be determined by leaving the state, thus providing a way of knowing where you were when you began executing the sequence. A *homing* sequence is a sequence that provides a unique signature for the state reached at the end of its execution. More precisely, a sequence $a \in A$ is a homing sequence for M if and only if for all $q_1, q_2 \in Q$, $q_1\langle a\rangle = q_2\langle a\rangle \Rightarrow q_1 a = q_2 a$. Every distinguishing sequence is also a homing sequence, because knowing where you were when you started executing the distinguishing sequence (which is the result of having executed the sequence) implies that you also know where you are at the end of the sequence. Both distinguishing and homing sequences may be either *preset* or *adaptive* [Gill, 1961]. An adaptive sequence is one in which the next action in the sequence is determined by the previous outputs, thus it is really a tree of actions whose branches correspond to possible outputs. A preset sequence is fixed, and is executed "blindly," without regard for the outputs along the way.

The usual definition for finite state automata includes a start state, where the machine is assumed to be prior to any actions. Some systems may have this feature, and if the agent has the ability to return to this state at will, we say that it has a *reset*. For many environments, however, the assumption of a reset is not realistic.

4 Theoretical Results for the Purely Deterministic Case

The main results of this paper are methods for learning automata in the presence of stochastic input and observation functions. In order to put those results in context, in this section we review some important previous results concerning the inference of automata in the purely deterministic case.

Some writers have made a distinction between inferring an automaton that behaves identically to the observed automaton and inferring an automaton that is isomorphic to the observed automaton. For any sequence of input/output data, there is a trivial automaton that agrees with the data that is constructed by building a chain of states as long as the data. If multiple sequences are allowed, this construction can build a tree with the start state as the root. For this reason, research has concentrated on finding the smallest automaton (in terms of $|Q|$) that agrees with a given set of data. Moore [Moore, 1956] showed that if the input/output pairs are assumed to have come from a reduced, strongly connected automaton, then inference of the smallest consistent automaton yields a result that is isomorphic to the original automaton. Thus, in this case, behaviorally correct inference and isomorphic inference are the same.

Gold [1972] provides a method for inferring automata in the limit from their input/output behavior. The algorithm samples the automaton by generating inputs and recording the resulting outputs and periodically produces a description of the automaton. Inference in the limit means that the sequence of descriptions produced by the algorithm is guaranteed to converge on the description of the correct automaton eventually, but in this case there is no way to detect that this has happened. Gold's algorithm relies on the learner having the ability to reset the automaton to the initial state at any time.

The general problem of inferring the smallest automaton consistent with a given set of input/output pairs is NP-complete [Angluin, 1978, Gold, 1978]. Indeed, even finding an automaton polynomially close to the smallest is intractable assuming P $\neq$ NP [Pitt and Warmuth, 1989].

Angluin [1987], building on the work of Gold, provides a polynomial-time algorithm for inferring the smallest automaton given the ability to reset the automaton and a source of counterexamples. In this model, at any point, the algorithm can hypothesize an automaton and the source of counterexamples indicates whether it is correct and, if it is not, provides a sequence of inputs on which the hypothesized and actual automata generate different outputs.

Rivest and Schapire [1989] show how to make use of a homing sequence as a substitute for a reset and how to dispense with both the reset and the source of counterexamples in the case in which a distinguishing sequence is either provided or can be learned in polynomial time [Rivest and Schapire, 1987, Schapire, 1991].

Several researchers have approached learning finite-state systems using neural networks. For example, Servan-Schreiber *et al.* [Servan-Schreiber *et al.*, 1989] used a recurrent network to learn finite-state grammars, and Bachrach [Bachrach, 1992] used a neural net to implement one of the the Rivest and Schapire algorithms mentioned above. This work has not stressed performance issues, nor has the issue of noise in inputs or outputs been considered.

5 Theoretical Results for the Stochastic Case

In this section we provide polynomial-time algorithms for automata identification in three different stochastic settings. In the first, we assume that there is no error in the agent's actions, but allow error in its observations of outputs. In the second, we assume that observations of special landmark states are unique and perfect, but allow error in the action function. In the final case, we allow error in both the observations and actions of the agent, but impose the restriction that every state's nominal label be unique.

5.1 Deterministic Actions and Stochastic Observations

In this section, we consider a situation in which there may be no uniquely labelled states. We show that by relying on a correct action function and knowledge of a distinguishing sequence, automata with non-unique observation functions can be learned even when the output function is noisy.[2]

5.1.1 Structural and Interaction Properties

Structurally, the requirements for this algorithm are quite weak: we require only that the automaton to be learned be strongly connected. We thus avoid the possibility that the agent becomes trapped in some part of the environment from which it cannot reach other parts. We also assume that the agent knows some upper bound on the number of states in the environment.

With regard to interactions, we assume that the agent moves deterministically, that is, that execution of actions is perfect. Observations, however, are assumed to be noisy, with the restrictions that the correct observation is made with probability greater than $\frac{1}{2}$ and that observations are independent events. Finally, we assume that the learner is provided with a preset distinguishing sequence for the environment. For many man-made and natural environments it is straightforward to determine a distinguishing sequence. For example, in most office environments, a short sequence of turns will serve to distinguish all junctions in the environment.

5.1.2 Algorithms

The algorithm we present here uses as a subroutine a procedure that moves the agent in the environment, collecting statistics on the labels it observes. The procedure provides as output a signature for the state reached at the end of the movement. Recall that a homing sequence is one for which the output uniquely determines the state reached at the end of the sequence and a distinguishing sequence is one for which the output uniquely determines the state from which the sequence was begun. In an environment with stochastic observations, homing sequences cannot be used because their signatures will not, in general, be observed correctly. This algorithm makes use of a procedure, called LOCALIZE, that, given a distinguishing sequence, achieves the effect of having a probably correct homing sequence. That is, it terminates with the automaton in a state and returns a signature for

[2]The work described in this section was carried out jointly with Dana Angluin and Sean Engelson, and is described in more detail in [Dean *et al.*, 1992a].

that state that is probably correct. The procedure is parameterized in such a way that it can be run longer in order to guarantee correctness with a higher probability.

We begin by explaining the LOCALIZE procedure, and then show how it can be used to learn environments with the properties discussed above.

The localization procedure works by exploiting the fact that movement is deterministic. The basic idea is to execute the given distinguishing sequence repeatedly until the agent is certain to be in a cycle, then execute it some number of times after that and collect the output. By finding the period of the cycle of locations, we can separate the observed outputs and use them as statistics on the outputs observed at each state. These statistics can then be used to determine (with high probability) the correct outputs at each state in the cycle, and hence to localize the agent by supplying the signature that would be returned by the distinguishing sequence in the deterministic case for the state the agent is in.

In order to determine the period of repetition of the walk with high probability, we keep statistics for alternative hypotheses for the period of the cycle. After the walk, these statistics are analyzed to determine with high probability the period of the cycle.

For an environment $\mathcal{E}$ with states $Q = \{q_1, q_2, \ldots, q_n\}$, let m be the given upper bound on $|Q|$. The set of outputs is $L = \{l_0, l_1, \ldots, l_k\}$. Let P_{ji} denote the probability of observing symbol l_j given that the agent is in state q_i. Let P denote a lower bound on all the P_{ji} for $i \neq j$.

Let $s = b_1 b_2 \cdots b_{|s|}$ be a preset distinguishing sequence for $\mathcal{E}$ consisting of one or more actions. For any integer $i > 0$, let s^i represent the sequence s repeated i times. Let $q\langle i\rangle$ be the state reached after executing the sequence s^{m+i}. The first m repetitions are sufficient to guarantee that the agent is in a cycle. Thus, the sequence of states $q\langle 0\rangle, q\langle 1\rangle, q\langle 2\rangle, \ldots$ is periodic. Let p denote the least period of the cycle; this is the value we wish to find.

As we execute the second part of the walk (after the s^m prefix), we keep track of our position in the sequence, and keep statistics separately for each position. For each offset $\ell = 0, \ldots, |s| - 1$, let $q^\ell\langle i\rangle$ be the state reached from $q\langle i\rangle$ by executing the first ℓ actions of s, that is, $q\langle i\rangle b_1 b_2 \cdots b_\ell$. For each ℓ, the sequence $q_0^\ell, q_1^\ell, q_2^\ell, \ldots$ is also periodic of period p.

For each ℓ, consider the sequence of (correct) outputs from the states $q^\ell\langle i\rangle$:

$$\phi_i^\ell = \phi^*(q^\ell\langle i\rangle).$$

Step #	States visited
0	0 4 2 0 4 2 ...
1	1 5 3 1 5 3 ...
2	2 0 4 2 0 4 ...
3	3 1 5 3 1 5 ...

Table 1: Sequences of visited states for $\pi = 4$

The output sequence

$$\phi_0^\ell, \phi_1^\ell, \phi_2^\ell, \ldots.$$

is also periodic, of some least period p_ℓ dividing p. Since outputs are not necessarily unique, we may have $p > p_\ell$. However, because s is a distinguishing sequence, p will be the least common multiple (LCM) of all the p_ℓ's. Thus, it would suffice to find each of the values p_ℓ, and take their LCM. In fact, what we will do is to find (with high probability) values q_ℓ such that p_ℓ divides q_ℓ and q_ℓ divides p, so that the LCM of the q_ℓ is also p. We describe the procedure for the sequence $q_0, q_1, q_2, \ldots$; it is analogous for the others.

Consider any candidate period $\pi \leq m$, and let $g = \gcd(p, \pi)$. For each $0 \leq i \leq \pi - 1$, consider the sequence of states visited every π repetitions of s, starting with $m+i$ repetitions of s. This will be the sequence of states

$$q\langle i\rangle, q\langle i+\pi\rangle, q\langle i+2\pi\rangle, q\langle i+3\pi\rangle, \ldots.$$

Since $q\langle i\rangle$ is periodic of period p, this sequence visits each state of the set $\{q\langle i+kg\rangle : k = 0, 1, \ldots, p/g - 1\}$ in some order, and then continues to repeat this cycle of p/g states.

Table 1 shows an example with $p = 6$, $\pi = 4, g = 2$. In this case, row r gives the indices of the states visited by repeating s, starting from $q\langle r\rangle$, assuming that the cycle of states visited by repeating s is $q_0, q_1, \ldots, q_5$.

In the special case $\pi = p$, shown in Table 2, each row r will consist exclusively of visits to state q_r. It is this case that we wish to distinguish from the others.

We cannot observe the states themselves, but we can observe the labels at each state the agent visits. The algorithm will repeat the distinguishing sequence s a total of $\mathcal{N}$ times, with $\mathcal{N}$ chosen to ensure that, with high probability, our observed frequencies are close to

Step #	States visited
0	0 0 0 0 0 0 ...
1	1 1 1 1 1 1 ...
2	2 2 2 2 2 2 ...
3	3 3 3 3 3 3 ...
4	4 4 4 4 4 4 ...
5	5 5 5 5 5 5 ...

Table 2: Sequences of visited states for $\pi = 6$

the probabilities of the sampled distribution. For each candidate period $\pi \leq m$, we form a table with π rows, numbered 0 to $\pi - 1$, and k columns, one for each possible label l_j. During the second part of the walk, we increment the table in row r, column j each time we observe label l_j.

After the second part of the walk, we compute the frequency of each entry (label) relative to the other entries in the same row. Let $\rho = (P - \frac{1}{2})$, the *separation* between the lower bound on correct observation and $\frac{1}{2}$. We use the value $\frac{1}{2} + \frac{1}{2}\rho$ as a threshold. When every row in the table for some π has a value that is above the threshold, the table is said to be *plausible*. For each plausible table, we take the sequence of π outputs determined by the largest value in each of the π rows and find the minimum period π' of the sequence. We find the LCM of all π'; this is our candidate for p.

We now present the procedure in a precise manner.

Procedure LOCALIZE

1. For simplicity, we assume that all the possible outputs are known and correspond to the integers $1, \ldots, k$. Build a table $T(\pi, \ell, r, j)$ of size $m \times |s| \times m \times k$. Initialize all the table entries to zero.[3]

2. Execute the sequence s^m to ensure that the agent is in a closed walk that it will continually traverse for as long as it continues to execute s.[4]

[3] If the labels are not known, then the table can be constructed incrementally, adding new labels as they are observed.

[4] Following Step 2, the next action should be the first action in s.

3. Initialize the sequence counter $R \leftarrow 0$ and the step counter $c \leftarrow 0$.

4. Execute s at least $\mathcal{N}$ times, incrementing R after each time. After executing each individual step, do the following:

   (a) Let $\ell = c \bmod |s|$, and j be the label observed immediately following execution.

   (b) For each $\pi = 1, 2, \ldots, m-1$, increment the table entry $T(\pi, \ell, R \bmod \pi, j)$ by 1.

   (c) Increment the step counter: $c \leftarrow c + 1$.

5. Let
$$F(\pi, \ell, r, j) = \frac{T(\pi, \ell, r, j)}{\sum_{j=0}^{k} T(\pi, \ell, r, j)}.$$

6. Initialize the period list $\mathcal{L} \leftarrow \{\_\}$. For each π and each ℓ, consider the two-dimensional table $F(\pi, \ell, \cdot, \cdot)$. If, for each $r < \pi$, row r in this table contains an element larger than $\frac{1}{2} + \frac{1}{2}\mathbf{sep}$, build the sequence of outputs of length π by taking the outputs corresponding to the large elements, $\arg\max_j F(\pi, \ell, r, j)$ for $r = 0, 1, \ldots, \pi - 1$. Find the period of this sequence and add it to $\mathcal{L}$.

7. Let Π be the LCM of all $\pi' \in \mathcal{L}$.

8. Conclude that the agent is currently located at the last state before row $r = R \bmod \Pi$ in the three-dimensional table $F(\Pi, \cdot, \cdot, \cdot)$, and return, as the hypothesis for the correct outputs of the distinguishing sequence s from this state, the sequence of outputs $\arg\max_j F(\Pi, \ell, r, j)$ for $\ell = 0, 1, \ldots, |s| - 1$ concatenated after the single output $\arg\max_j F(\Pi, |s| - 1, (r-1) \bmod \Pi, j)$.

Table 3 shows the tables that resulted from a run of LOCALIZE in the environment shown in Figure 3. The distinguishing sequence given to the procedure was $\langle bb \rangle$. Recall that π is the conjectured period length (number of executions of s), l is an index into the sequence s, r is an index into the cycle of length π, and j is the observed label. We can see that the tables for $\pi = 2$ and $\pi = 4$ are all plausible. When $\pi = 2$ and $\ell = 0$, we have the sequence 01, which has period 2; for $\pi = 2$ and $\ell = 1$, the period is 1; for $\pi = 4$ and $\ell = 0$ the period is 2; and for $\pi = 4$ and $\ell = 1$, the period is 1. The LCM of this set of periods is 2, so $\Pi = 2$. In the example, the agent starts in q_0, and so finishes either in q_0 (R is even) or in q_2 (R is odd). In the first case $r = 0$ and the signature returned by the algorithm is that of q_0: $\langle 000 \rangle$. In the other case $r = 1$ and the signature is $\langle 010 \rangle$.

$\pi = 1, \ell = 0$		$\pi = 1, \ell = 1$	
$j = 0$	$j = 1$	$j = 0$	$j = 1$
26(0.52)	24(0.48)	44(0.88)	6(0.12)
0	0	0	0
0	0	0	0
0	0	0	0
$\pi = 2, \ell = 0$		$\pi = 2, \ell = 1$	
$j = 0$	$j = 1$	$j = 0$	$j = 1$
21(0.84)	4(0.16)	**21(0.84)**	4(0.16)
5(0.2)	**20(0.8)**	**23(0.92)**	2(0.08)
0	0	0	0
0	0	0	0
$\pi = 3, \ell = 0$		$\pi = 3, \ell = 1$	
$j = 0$	$j = 1$	$j = 0$	$j = 1$
7(0.412)	10(0.588)	15(0.882)	2(0.118)
10(0.588)	7(0.412)	16(0.942)	1(0.0588)
9(0.562)	7(0.438)	13(0.812)	3(0.188)
0	0	0	0
$\pi = 4, \ell = 0$		$\pi = 4, \ell = 1$	
$j = 0$	$j = 1$	$j = 0$	$j = 1$
11(0.846)	2(0.154)	**13(1)**	0
2(0.154)	**11(0.846)**	**12(0.923)**	1(0.0769)
10(0.833)	2(0.167)	**8(0.667)**	4(0.333)
3(0.25)	**9(0.75)**	**11(0.917)**	1(0.0833)

Table 3: Tables built by LOCALIZE for the environment of Figure 5.3

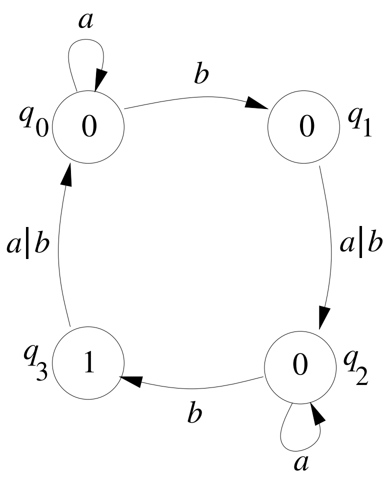


Figure 3: A simple environment with a short distinguishing sequence

5.1.3 Analysis

A formal proof of the correctness of the algorithm and a proof of the complexity in terms of the number of steps appears in a paper by Dean *et al.* [1992b]. We cite the final result concerning this complexity, and refer the reader to the paper for a complete proof.

Theorem 1 *In order to provide a correct signature with probability* $1 - \epsilon$, LOCALIZE *must execute the distinguishing sequence s at least*

$$m + \frac{8m^2}{(2P-1)^2} \ln \frac{2|s|km^3}{\epsilon}$$

times. The number of steps taken is thus

$$O\left(\frac{|s|m^2}{(P-\frac{1}{2})^2} \ln \frac{|s|km^3}{\epsilon}\right).$$

Recall that $|s|$ is the length of the distinguishing sequence, m is an upper bound on the number of states in the automaton, $P > \frac{1}{2}$ is a lower bound on the probability of the correct output from a state, k is the number of possible outputs, and ϵ is a bound on the probability that the procedure will fail.

The complete learning procedure which employs a distinguishing sequence s and LOCALIZE as subroutine is relatively straightforward and hence only sketched here. The detailed algorithm and proof of correctness is in [Dean *et al.*, 1992b]. Suppose for a moment that LOCALIZE always returns the agent to the same state and that the agent can always determine when it is in a state that it has visited before. In this case, the agent can learn the connectivity of the underlying automaton by performing what amounts to a depth-first search through the automaton's state transition graph. The agent does not actually traverse the state-transition graph in depth-first fashion; it cannot manage a depth-first search since, in general, it cannot backtrack. Instead, it executes sequences of actions corresponding to paths through the state transition graph starting from the root of the depth-first search tree by returning to the root each time using LOCALIZE. When, in the course of the search, a state is recognized as having been visited before, an appropriate arc is added to the inferred automaton and the search 'backtracks' to the next path that has not been completely explored.

The actual inference algorithm is more complicated because our localization procedure does not necessarily always put the agent in the same final state and because we are not able to immediately identify the states we encounter during the depth-first search. The first problem is solved by performing many searches in parallel, one for each possible (root) state that LOCALIZE ends up in. Whenever LOCALIZE is executed the agent knows (with high probability) what state it has landed in and can take a step of the depth-first search that has that state as the root node. The second problem is solved by using a number of executions of the distinguishing sequence from a given starting state to identify that state with high probability.

It can easily be shown that it is possible to learn an automaton with high probability given a polynomial number of visits to a fixed starting state. LOCALIZE may not return the agent to the same starting state every time, but in a polynomial number of times executing LOCALIZE the agent will return to some state the required number of times. Of course, the agent is never absolutely sure that it has returned to the same state, but the agent can achieve any required probability of success in a number of steps polynomial in the reciprocal of the probability and the other measures of problem size.

The requirement that a distinguishing sequence be provided seems unlikely to be avoided in general. In particular, Dean *et al.* [1992b] show that learning such a sequence is not possible in polynomial time for environments as general as those we are considering here.

Further, the results of Yannakakis and Lee [Yannakakis and Lee, 1991] indicate that even computing preset distinguishing sequences for general environments is hard, although they provide an efficient way to compute adaptive distinguishing sequences. However, it is clear that an adaptive sequence will not work in place of a preset one in the LOCALIZE procedure. The procedure relies on the ability to follow the sequence deterministically, so that it is certain to be in the right states, even if it perceives incorrect outputs along the way. With an adaptive sequence, correct movement would no longer be assured, since the action to perform at any point would depend on the previous, possibly incorrect, output.

5.2 Stochastic Actions and Deterministic Observations

A *landmark environment* is one in which certain states have unique labels that can be detected as such. That is, not only are the labels unique, but one can think of the agent as having a detector that allows it to determine whether or not an observed label is unique. Although this seems to be a fairly strong condition, it is certainly something that people are quite good at. Indeed, people select landmarks precisely because they are sure that they present a unique aspect, and this surety comes without having examined all other locations. For example, almost anyone would identify the Transamerica pyramid in San Francisco as a good landmark, based on background knowledge of building styles that assures them that it is unique in appearance. This ability amounts to having learned what is unusual enough to count as a landmark; in this section we assume that agents have this ability. The problem of how this ability is learned or programmed is an interesting one, but is outside the scope of this work. [5]

5.2.1 Structural and interaction properties

In addition to the landmark property, we assume that the environment to be learned is reversible. This means that for every action that does not result in a self-transition, there is another action that reverses the effect of the first. While the agent's observation is perfect, no restriction is made on the structure of the labelling other than the landmark property. Thus, it is possible that all states that are not landmarks have the same label. With regard

[5] The work described in this section was carried out jointly with Jeff Vitter and is described in more detail in [Basye *et al.*, 1989a].

to movement, we assume that the agent takes the intended action with probability $\theta > \frac{1}{2}$, but allow any static distribution over actions that are in error. Finally, we assume that the agent has perfect knowledge of the action that would reverse its last action, that is, it knows perfectly from which way it arrived at its current location. We call the latter requirement *reverse movement certainty.* We assume that the agent has the ability to detect actions that will fail, that is, that will cause self transitions. This obviously implies that the agent will also know which actions will succeed, and how many of these there are; we will sometimes refer to this as the degree of the state.

For convenience, we define D to be the subset of Q consisting of the landmark states and I to be the subset of Q consisting of the non-landmark states. We define the *landmark distribution parameter*, r, to be the maximum distance from any state in I to the nearest landmark (if $r = 0$, then I is empty and all states are landmarks). We say that a procedure learns the *local connectivity within radius* r of some $q \in D$ if it can provide the shortest path between q and any landmark within a radius r of q. We say that a procedure learns the *global connectivity of an environment* $\mathcal{E}$ *within a constant factor* if, for any two states q_1 and q_2 in D, it can provide a path between q_1 and q_2 whose length is within a constant factor of the length of the shortest path between q_1 and q_2 in $\mathcal{E}$. The path will be constructed from paths found between locally connected landmarks (see Figure 4).

Thus, we may summarize the agent's capabilities as follows. The agent's action function is not perfect, but serves to move the agent in the intended direction more than half the time. At each state, the agent knows what action to take to reverse the last action taken, even if it was not the intended action. In addition, the agent knows whether the state is a landmark, and if so, what its unique name is.

5.2.2 Algorithms

We begin by presenting a procedure that learns the local connectivity of an environment. We then show that the multiplicative error incurred in trying to answer global path queries can be kept low if the local error can be kept low and that the transition from a local uncertainty measure to a global uncertainty measure does not increase the complexity by more than a polynomial factor. We conclude that it is possible to build a procedure that directs exploration and map building so as to answer global path queries that are accurate and within a small constant factor of optimal with high probability.

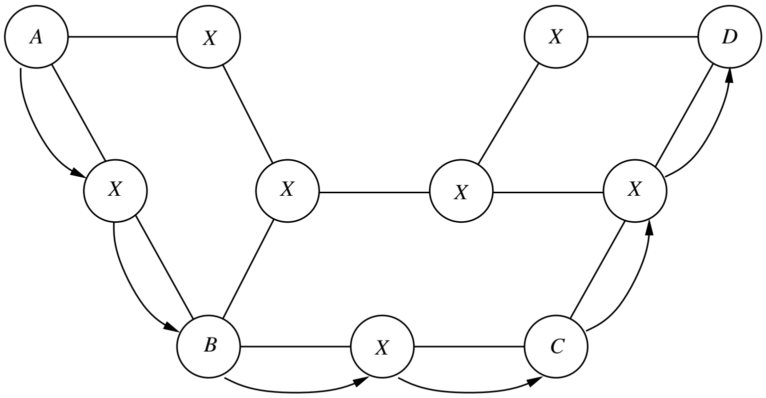


Figure 4: A path found between landmarks A and D

The procedure for learning local connectivity begins with a search of the environment to locate all the landmarks. Once they have been found, the algorithm looks for short (less than cr, where $c > 2$ is an integer) paths between landmarks. This process has two phases: in the first, candidate paths are located, and in the second, these paths are verified. The need for this two-stage process arises from the possibility that some combination of movement errors could result in paths that appeared to connect two landmarks, but that, in fact, did not. We show that by exploiting reverse certainty we can statistically distinguish between the true paths and errors. By attempting enough traversals, the procedure can ensure with high probability that the most frequently occurring sets of directions corresponding to perceived traversals actually correspond to paths in $\mathcal{E}$.

The learning algorithm can be broken down into three steps: a landmark identification step in which the agent finds and identifies a set of landmarks, a candidate selection step in which the agent finds a set of candidate paths in $\mathcal{E}$ connecting landmarks, and a candidate filtering step in which the agent determines which of those candidate paths correspond to actual paths in $\mathcal{E}$.

We now present the procedure for learning local connectivity.

Procedure: Connect

1. (Landmark location) A uniform random walk is made in the environment and landmarks encountered are added to a list.

2. For each landmark A in the list from step 1:

   (a) For each sequence of directions of length cr, make multiple traversals of the path defined by the sequence, starting from A, recording the intended direction and reverse direction at each step. After each traversal, return to A with a random walk. Add any path which reaches some other landmark to the list of candidate paths.

3. For each candidate path in the list from step 2, execute the path multiple times, beginning at the landmark at the head of the list and comparing reverse directions observed with those recorded for the path in step 2. Any failure in these comparisons is a failed traversal. If the landmark is reached without failure, the traversal is successful. If the landmark is not reached, return to the beginning landmark with a random walk. For each path, maintain a count of successful traversals.

4. Return all paths from the list whose count from step 3 is above a threshold.

5.2.3 Analysis

We now state without proof (the detailed proof can be found in [Basye, 1992]) a series of lemmas leading up to the result that this algorithm is correct and has polynomial sample complexity.

Lemma 1 *For any $\epsilon_l > 0$, the* CONNECT *procedure learns the local connectivity within cr of each state in any landmark environment with probability $1 - \epsilon_l$ in time polynomial in $\frac{1}{\epsilon_l}$, $\frac{1}{1-2\theta}$, and the size of $\mathcal{E}$, and exponential in cr.*

Lemma 2 *Let $\mathcal{E}$ be a landmark environment with distribution parameter r, and let c be any integer, $c > 2$. Given a procedure that, for any $\epsilon_l > 0$, learns the local connectivity within cr of any landmark in $\mathcal{E}$ in time polynomial in $\frac{1}{\epsilon_l}$ with probability $1 - \epsilon_l$, it is possible to learn the global connectivity of $\mathcal{E}$ with probability $1 - \epsilon_g$ for any $\epsilon_g > 0$ in time polynomial in $\frac{1}{\epsilon_g}$ and the size of the environment. Any global path returned as a result will be at most $\frac{c}{c-2}$ times the length of the optimal path.*

Theorem 2 *It is possible to learn the global connectivity of any landmark environment with probability* $1 - \epsilon$ *in time polynomial in* $\frac{1}{\epsilon}$, $\frac{1}{1-2\theta}$, *and the size of* $\mathcal{E}$, *and exponential in* r.

Theorem 2 is a simple consequence of Lemmas 1 and 2. It has an immediate application to the problem of learning the global connectivity of an environment where all the states are landmarks. In this case, the parameter $r = 0$, and we need only explore paths of length 1 in order to establish the global connectivity of the environment. Because each candidate path has length one, this process works even if there is no reverse certainty.

Corollary 1 *It is possible to learn the connectivity of an environment* $\mathcal{E}$ *with only distinguishable locations with probability* $1 - \epsilon$ *in time polynomial in* $\frac{1}{\epsilon}$, $\frac{1}{1-2\theta}$, *and the size of* G, *even if there is reverse uncertainty.*

The notion of global connectivity defined above does not require that the environment be *completely learned*, (*i.e.,* to recover the structure of the entire environment). It is assumed that the indistinguishable states are of interest only in so far as they provide directions necessary to traverse a direct path between two landmarks. But it is easy to imagine situations where the indistinguishable states and the paths between them are of interest. For instance, the indistinguishable states might be partitioned further into equivalence classes so that one could uniquely designate a state by specifying its equivalence class and some radius from a particular global landmark (*e.g.,* the bookstore just across the street from the Chrysler building). In [Basye *et al.*, 1989b], we show how to modify the above approach and try to completely learn the environment by first completely learning local neighborhoods of each landmark.

5.3 Stochastic Actions and Observations

In this section, we consider the case in which there is error in both the agent's actions and observations. In order to provide an algorithm with polynomial sample complexity, we are required to make restrictions on the environment. We present those restrictions, then give the algorithm and sketch a proof of its correctness and complexity.

5.3.1 Structural and interaction properties

Recall that an environment is reversible if, whenever there is an action a leading from q_1 to q_2 and $q_1 \neq q_2$, there is also a corresponding action from q_2 to q_1. Virtually all navigational environments have this property. Even one-way streets normally have corresponding parallel streets running in the opposite direction, providing an essentially undirected environment. Here, we assume the environment is reversible, but make no assumption that the agent knows the action that would reverse a given action.

The *conductance* of an environment is, informally, a measure of how many ways there are to get from one part of the environment to another. If there are a lot of bottlenecks, it is harder to learn the environment through random exploration. The algorithm presented here will work on environments of any conductance, but the lower the conductance (the more bottlenecks), the longer it will take.

We assume that the labelling of the environment is unique, so know that $|Q| \leq |L|$ (the number of states is less than or equal to the number of labels) and for simplicity of presentation we will assume that $|Q| = |L|$ and that for all i, label l_i is nominally correct for state q_i. Let $P_i = \phi(q_i)(l_i)$ be the probability that the agent correctly observes label l_i when it is in state q_i. For $i \neq j$, let $P_{ji} = \phi(q_i)(l_j)$ be the probability that the agent mistakenly observes label l_j when it is in state q_i. Finally, let P be a lower bound on all probabilities of correct observation, P_i. It is intuitively apparent that the higher the probability of correct observation is, the sooner the agent will be able to correctly identify its underlying environment.

The relationship between commands and actions is analogous to that between labels and states. We assume that $|A| = |C|$. Let $\theta_i = \psi(c_i)(a_i)$ be the probability that the agent correctly performs action a_i when it executes command c_i. Let θ be a lower bound on all the θ_i. In addition, we assume that the action function ψ is such that the agent can choose commands in such a way as to choose actions uniformly, allowing the agent to generate a random walk in the environment.

With regard to observation, we assume that the observation probabilities are *reflexive*; that is, for all i, j, $P_{ij} = P_{ji}$. The agent is just as likely to mistake state q_i for state q_j as it is to mistake state q_j for state q_i. As will be seen later, the point of this requirement is to limit the frequency with which a given label can be seen in those states where the observation is incorrect.

5.3.2 Algorithms

Given the restrictions from the previous section, the underlying structure of the environment can be learned by a very simple algorithm. The result of the algorithm will be, with high probability, an environment that is isomorphic to the original environment.

The algorithm uses a random strategy to explore the graph and records, for each pair of labels, (l_i, l_k), and each command, c_j, the number of times that an observation of l_i followed by performing c_j resulted in an observation of l_k. After enough exploration, a graph can be extracted from these statistics. If l_k is the most frequently observed label after doing command c_j when observing label l_i, then we assume that there is an edge for action a_j from state q_i to state q_k in the underlying graph.

More formally, the state-transition graph can be learned in the following way:

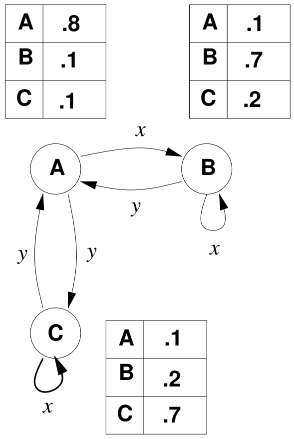
Algorithm CMFO (Choose Most Frequent Observation)

1. For each command $c \in C$, construct a two-dimensional table T_c indexed in each dimension by the labels in L.
2. Initialize all of the entries of all tables to 0.
3. Begin executing a uniform random walk in the environment. Whenever a transition is made from a state in which label l_i is observed to a state in which label l_j is observed by using command c, increment the value of $T_c(l_i, l_j)$.
4. After n action steps, stop and return edge set E' containing edge (q_i, a_h, q_j) only if $T_{c_h}(l_i, l_j) > T_{c_h}(l_i, l_k)$ for all k not equal to j.

Figure 5 shows the results of a sample run of this algorithm on a very simple graph. The small tables specify the perception probabilities at each vertex; the large tables indicate, for each action, the frequency of sequential label pairs. The underlying graph is encoded by the largest element in each row of each table, which is in bold-face type.

5.3.3 Analysis

In this section we state specific conditions under which algorithm CMFO above succeeds and provide a bound on the number of observations the agent must make in order to have a given confidence of identifying the entire graph correctly.



0

T_x	A	B	C
A	12	**109**	37
B	10	**89**	55
C	56	26	**92**

T_y	A	B	C
A	35	19	**108**
B	**125**	23	17
C	**81**	54	52

Figure 5: Results of running the graph-identification algorithm in the simple environment shown for 1,000 steps. ($\theta = 0.8$)

In the following, we abuse notation slightly and let q_i stand for the event of visiting vertex q_i, l_i for the event of perceiving label l_i, and so on; in addition, we let $\circ q_i$ stand for the event of visiting q_i as a result of executing some action and $\circ l_i$ stand for the event of perceiving l_i as a result of executing some action. The algorithm can be seen as constructing estimates of the probabilities that a label l_j is seen after executing command c, given that the label l_i was seen just prior; this probability is notated as $\Pr(\circ l_j \mid l_i \wedge c)$. The observation probabilities depend on the details of the random walk being executed by the agent. This will be addressed later in the proof.

An important quantity for bounding the necessary number of trials is the *separation* of the probabilities $\Pr(\circ l_j \mid l_i)$ and $\Pr(\circ l_k \mid l_i)$, given that command c was executed. For a particular i, j, and k, the separation is defined as $s_i(j,k) = \Pr(\circ l_j \mid l_i) - \Pr(\circ l_k \mid l_i)$; [6] for a given label i, $s_i = s_i(j,k)$ where l_j is the "correct" successor label to label l_i (that is, that for action a corresponding to command c, $\delta(q_i, a) = q_j$) and l_k is the most likely incorrect successor label (that is, l_k is the label not equal to l_j that maximizes $\Pr(\circ l_k \mid l_i)$); a lower bound on all the s_i is written as s. If the value of s is high, then we are much more likely to see "correct" transitions than to see "incorrect" ones. For the CMFO algorithm

[6] Because the separation is always with respect to a particular command, we suppress the parameter throughout.

to work correctly, we must guarantee that the separation is always positive, so that no incorrect transition is more likely to be observed than the corresponding correct one. The requirement that observation probabilities be reflexive derives from this need. Consider the sum of the probabilities of making a particular observation in error: $\sum_{i \neq k} P_{ki}$. Without any other restriction, it is possible to construct situations in which this is as high as $|Q|(1-P)$, and in which positive separation is impossible for all but unrealistically high values of P. The reflexivity requirement is one of several possible requirements that eliminate such situations. In Section 6 we discuss one set of noise models that satisfy this requirment.

We can characterize the number of observations required by the algorithm as a function of the separation. The proof is omitted, but can be found elsewhere [Kaelbling *et al.*, 1992, Basye, 1992]. It uses Hoeffding's inequality to show that after a large enough number of samples, it is very likely that the observation with the largest sample frequency is also the observation with the largest true frequency.

Lemma 3 *If the separation s is greater than 0, then the output of algorithm* CMFO *is correct with probability at least $1-\epsilon$ after each vertex has been visited at least N times, where N is polynomial in $1/s$, $1/\epsilon$, $|Q|$, and $|A|$.*

We will use a random walk to explore the environment, so we must turn our attention to the question of how long a walk is needed. Not only must we guarantee with high probability that the states are all visited enough times, we must also be able to characterize the distribution of state visitations, because it affects the transition probabilities, and, hence, the separation.

A *uniform* random walk in the environment is one in which actions are also chosen equiprobably; by our earlier assumption, the agent knows some distribution over commands that allows this. We can describe the walk by a Markov process with transition matrix R, defined by $r_{ij} = \frac{g}{|A|}$, where g is the number of edges in E from q_i to q_j. In a reversible environment, there are as many edges from q_i to q_j as there are from q_j to q_i, so R is symmetric; this implies that the columns, as well as the rows, sum to 1, making the matrix *doubly stochastic*. Any matrix that is doubly stochastic has a uniform stationary probability distribution [Feller, 1970]; that is, in the limit each state is visited with probability $1/|Q|$.

The next lemma concerns the rate at which the distribution of state visitations approaches the uniform stationary distribution. Let $\bar{x}_i(t)$ be the probability that the process is in state q_i at time t and let π_i be the stationary probability of state q_i. Define the discrepancy, ζ_t, to be the L_2 norm of the difference between π and $\bar{x}(t)$. This result is a direct consequence of Mihail's result [Mihail, 1989] on the convergence of Markov chains.

Lemma 4 *Let t be the number of time steps needed to guarantee that the discrepancy between the state distribution at time t and its stationary distribution is less than ζ, when executing the process determined by the transition matrix R. Then t can be bounded above by*

$$t \leq \frac{2}{\Phi^2} ln \frac{1}{\zeta} \ ,$$

where Φ is the merging conductance *of the process. It is defined by*

$$\Phi = \min_{S \subset Q : \sum_{q_i \in S} \pi_i \leq 1/2} \Phi(S) \ ,$$

where

$$\Phi(S) = \frac{\sum_{j_1 \in S} \sum_{j_2 \in Q-S} \sum_i r_{j_1 i} r_{j_2 i}}{|S|} \ .$$

Now, we give a bound on the separation, assuming a bound on the discrepancy.

Lemma 5 *Let ζ be an upper bound on the discrepancy after t steps, let $z = \frac{1+|Q|\sqrt{\zeta}}{1-|Q|\sqrt{\zeta}}$, let P be a lower bound on the probability of making a correct observation, P_i, and let θ be a lower bound on the probability of taking a correct action, θ_i. If the probabilities of making incorrect observations are reflexive, that is, if $\forall x, y, P_{xy} = P_{yx}$, then for all actions and all initial labels l_i, after a random walk of length t,*

1. *$s_i > 0$ if $P \geq \frac{1}{4\theta}(1 - z + \sqrt{z^2 + 8\theta z - 2z + 1})$, and*
2. *$s_i \geq \frac{1}{z} P(2\theta P - 1) + P - 1$.*

Note that for a long enough walk it is reasonable to approximate z by 1, which yields the simple requirement that $P > 1/\sqrt{2\theta}$; this has the intuitively pleasing consequence that θ, the lower bound on correct execution of commands, must be greater than $\frac{1}{2}$. In this case, the separation is bounded below by $2\theta P^2 - 1$, so the exploration complexity of algorithm CMFO contains a factor of $1/(2\theta P^2 - 1)$. Figure 6 shows a plot of this factor for values of

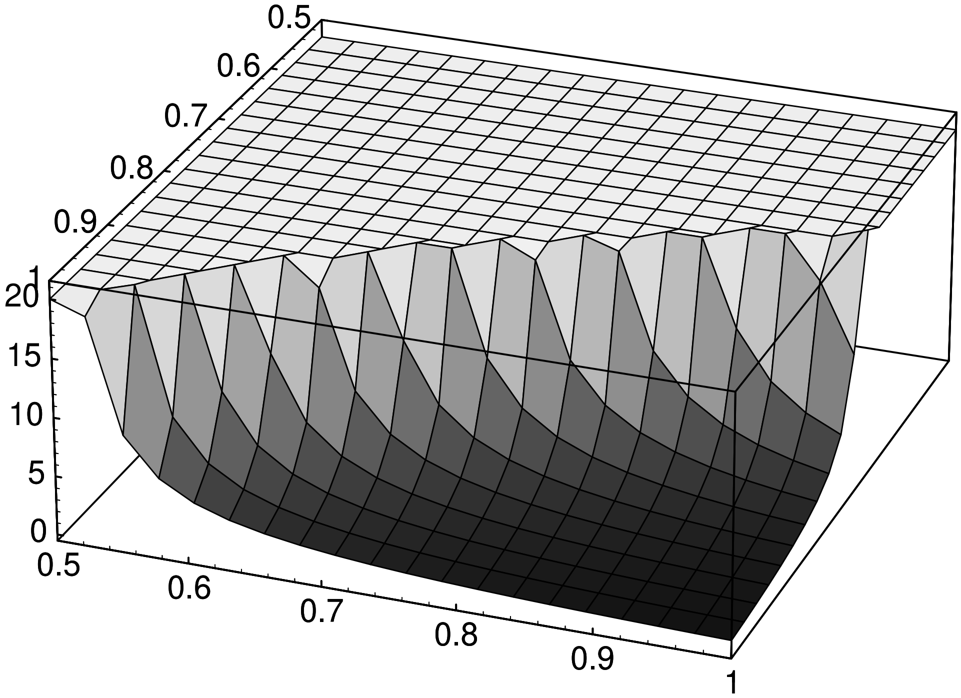


Figure 6: A plot of $1/(2\theta P^2 - 1)$ in the area of positive separation

P and θ from $\frac{1}{2}$ to one in the area where the separation is positive. The "plateau" area in the figure represents the portions of (P, θ)-space for which the bound on the separation is negative.

Theorem 3 *The output of the* CMFO *algorithm is correct with probability at least* $(1-\epsilon)(1-\epsilon_2)$ *after a uniform random walk of length polynomial in*

$$1/(\frac{1}{z}P(2\theta P - 1) + P - 1) \ ,$$

$1/\epsilon$, $1/\epsilon_2$, $1/\Phi$, $|Q|$, *and* $|A|$, *where* P *is the lowest probability of correct observation,* θ *is the lowest probability of correct action,* Φ *is the conductance,* $z = \frac{1+|Q|\sqrt{\zeta}}{1-|Q|\sqrt{\zeta}}$, *and* ζ *is an upper bound on discrepancy of the state distribution, whenever*

1. $P > \frac{1}{4\theta}(1 - z + \sqrt{z^2 + 8\theta z - 2z + 1})$,
2. *a uniform distribution on* C *induces a uniform distribution on* A,
3. $\forall x, y, P_{xy} = P_{yx}$, *and*
4. *the environment is reversible.*

6 Empirical Results

In the following sections we develop a particular class of noise model and present results of empirical simulations of our algorithms from Sections 5.1 and 5.3. In both cases, the results demonstrate that in automata representing plausible real-world environments, the actual number of samples needed is far less that predicted by the theoretical results.

6.1 Noise models

Two of the results of Section 5 require that the probability of correct action be above some threshold. Two others require that the probability of correct observation be above some threshold. In addition, algorithm CMFO requires that the probability of error be reflexive for observation. A large number of possible noise models satisfy these constraints, and the choice of noise model will certainly affect the actual performance of the algorithm. In our experiments, we have used one noise model for actions, and several different noise models for observations.

Our error model for actions is a simple *uniform* model. When an incorrect action is taken, it is chosen uniformly at random from all incorrect actions. For observations, we have developed a general class of noise models called *similarity partition* noise models. Such a model is constructed as follows. The set Q of states is partitioned into subsets $Q_1, Q_2, \ldots, Q_k$. Intuitively, the elements of the partition represent sets of states that look alike. Each state in a given partition, Q_i, has an observation function that gives the correct answer with probability P, and gives the label of some other state in the partition with probability $(1-P)/|Q_i|$. The uniform error model is one special case of this scheme; it occurs when the partition has only one element that covers all of Q.

6.2 Deterministic Actions and Stochastic Observations

The polynomial functions we have shown to bound the performance of the algorithms described in Section 5.1 are pessimistic. We now describe the results of experiments with LOCALIZE that indicate that this is so. There are similar results for the complete automaton inference algorithm provided in [Dean *et al.*, 1992b].

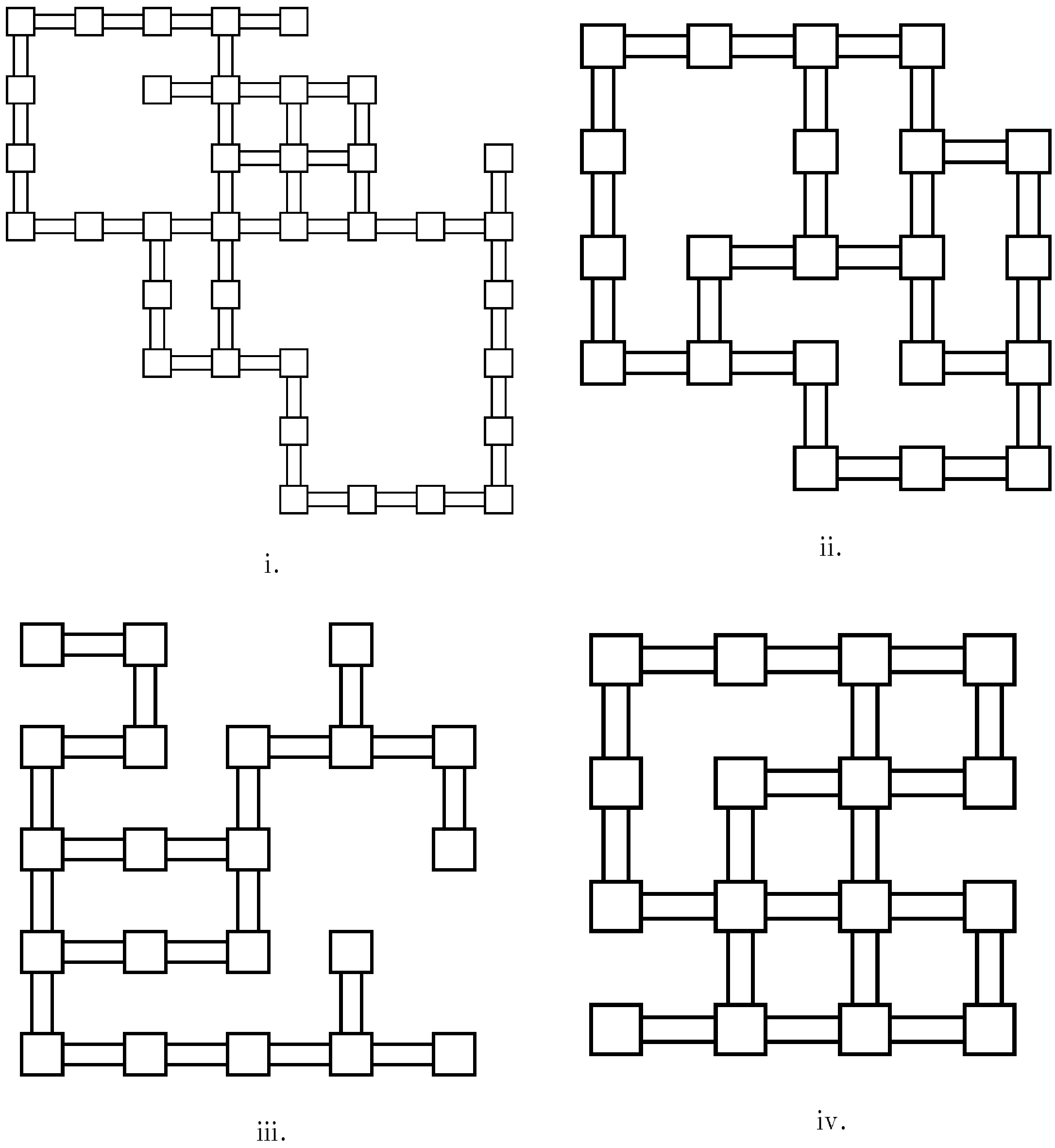


Figure 7: Graphs for hallway environments

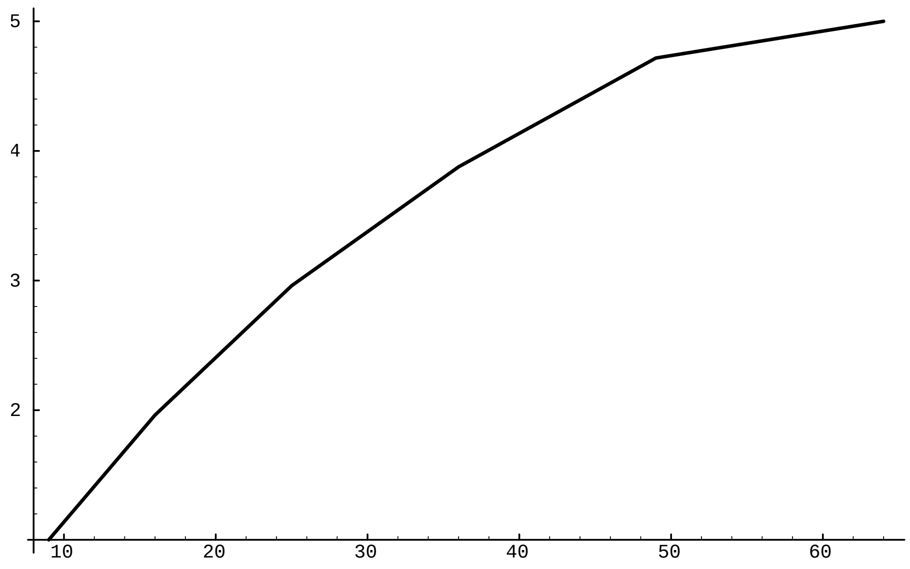


Figure 8: Length of the shortest distinguishing sequence as a function of the number of states in the environment

Our result requires environments with distinguishing sequences. We hypothesize that many natural environments, and hallway environments in particular, possess short distinguishing sequences. To test this hypothesis, we constructed a variety of hallway environments and determined the length of the shortest distinguishing sequence, assuming that such a sequence existed. Figure 7.i depicts the state-transition graph for the the fifth floor of the Brown CS Department. Three other graphs typical of the ones that we used in our experiments are shown in Figure 7.ii through 7.iv. The length of the shortest distinguishing sequence for Figure 7.i is four. The lengths of the shortest distinguishing sequences for Figures 7.ii, 7.iii, and 7.iv are two, three, and two respectively.

We generated a large number of graphs by starting with a $d \times d$ grid of locations and constructing a graph of n edges by selecting n pairs of adjacent locations according to a uniform distribution without replacement. The actions available at a location consisted of movement in each of the four directions (*i.e.,* N, E, W, S) along axes of the grid; if there was not an edge in a particular direction, the action corresponded to a self transition. The labels for locations encoded the junction type (*e.g.,* L-shaped or T-shaped) and orientation (*e.g.,* facing N, E, W, or S) for a total of sixteen labels, including the degenerate label

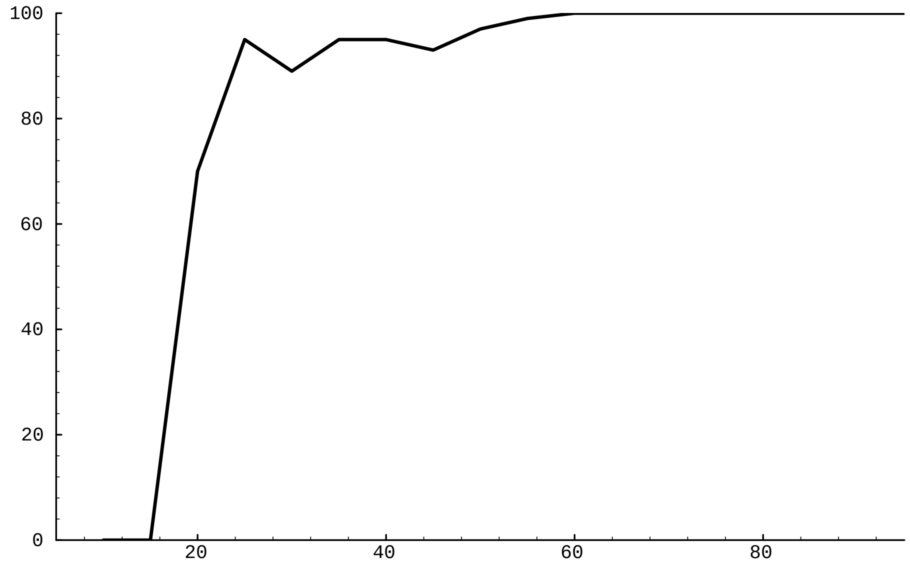


Figure 9: Percentage of correct state identifications for LOCALIZE as a function of the number of repetitions of the distinguishing sequence

corresponding to a location with no adjacent corridors. A uniform error model was used, so the probability that the agent observed the correct label was P and the probability that it observed a label other than the correct one was $\frac{(1-P)}{15}$. For fixed d with n in the range of d to d^2, the length of the shortest distinguishing sequence was nearly constant. For the graphs that we have looked at, the length of the shortest distinguishing sequence seemed to increase roughly as the square root of the number of states. Figure 8 shows the length of the shortest distinguishing sequence as a function of the number of states in the environment, averaging over sets of environments.

The theoretical results indicate that for a DFA consisting of 21 states LOCALIZE needs as many as 76206 steps for $P = 0.8$. In our simulations, however, LOCALIZE is successful 100% of the time with no more than 50 steps using a distinguishing sequence of length three. We also observed that the performance of LOCALIZE is largely insensitive to P, continuing to perform with 100% accuracy having executed 50 steps with P as low as 0.5. We believe this is largely due to the fact that errors are distributed uniformly over the incorrect labels; it is straightforward to construct alternative error distributions that require a lot more

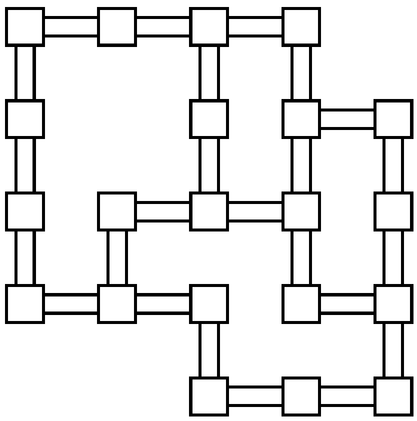


Figure 10: An abstract environment used to test the CMFO algorithm

work on the part of LOCALIZE. Figure 9 shows a graph of the percentage of correct state identifications for LOCALIZE running on the environment of Figure 7.i as a function of the number of repetitions of the distinguishing sequence. This graph typifies the performance of LOCALIZE running on the range of graphs that we considered in our experiments.

6.3 Stochastic Actions and Observations

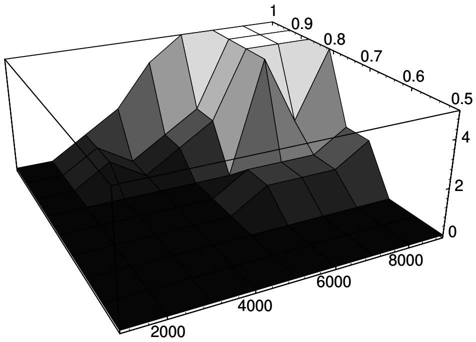
In this section, we consider the performance of the CMFO algorithm in simulation. The environments used by the simulation are also constructed as abstract models of hallway environments. These environments have four actions, corresponding to moving North, South, East, and West. As before, actions that are not applicable for a given location result in self-transitions. Figure 10 shows the environment, called *CIT 4*, used in our experiments; it models one floor of the Computer Science Department at Brown University and has 21 states. The experiments were performed using a range of values for the parameters P and θ.

With regard to the CMFO algorithm, the uniform error model is quite benign. This is because uniform distribution over a large number of states virtually guarantees that the most frequent observation will be correct. Indeed, under the uniform noise model, increasing the size of the problem actually helps in this regard, although the increased size also requires a longer walk to gather enough data. By using smaller partitions, more pernicious noise models may be created. For example, by partitioning Q into pairs, a significant competitor to the correct answer, in terms of frequency, is assured. It is easy to see that all similarity partition noise models satisfy the reflexivity requirement stated in the proof.

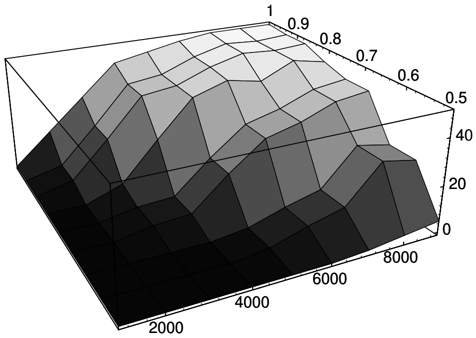
In these experiments, three different similarity partition error models were used. The first error model was the uniform model, the second and third were more complicated partitions based on the hallway structure of the environment. In these models, locations were partitioned according to the type of junction they represented in the world. For example, *CIT 4* has corner junctions, T junctions and hall junctions. If the agent is able to detect the type of a junction reliably, then its sensors will conform to a noise model that partitions locations by junction type. Junction types may further be partitioned by considering the orientation of the junction. For example, a corner junction with South and West hallways might be distinguished from one with North and East hallways. In the *CIT 4* environment, the oriented junction-type partition results in a number of singleton elements, but also results in a number of pairs and triples.

6.3.1 Results

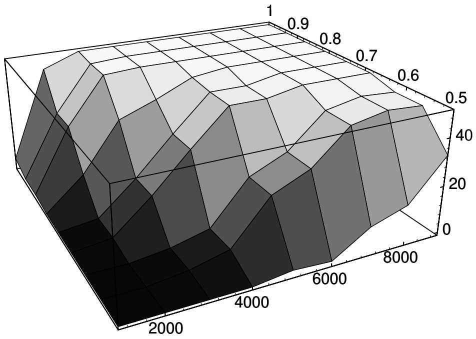
The experiments consisted of multiple runs of algorithm CMFO on the *CIT 4* environment. For each of the three error models, the algorithm was run in simulation with different values for the probability of correct action, θ, probability of correct observation, P, and number of steps. In these simulations, each time the algorithm issued a command, a random value from 0 to 1 was generated and compared against θ. If the value generated was greater than θ, an random incorrect action was chosen uniformly and executed, otherwise the correct action was executed. After each execution, a second number was generated and compared against P. If the number generated was greater than P, a random incorrect observation was returned from the similarity partition element of the current state. Otherwise, the correct label of the current state was returned. Values of θ and P from 0.5 to 1 in 0.1 increments were used; the walk length was varied from 1000 to 10,000. The walk length interval was chosen to give a range in which the algorithm's overall performance went from bad to nearly perfect. For each combination of walk length, θ, and P, the algorithm was given 20 tries to construct a map (50 tries per combination were used for the uniform error model).



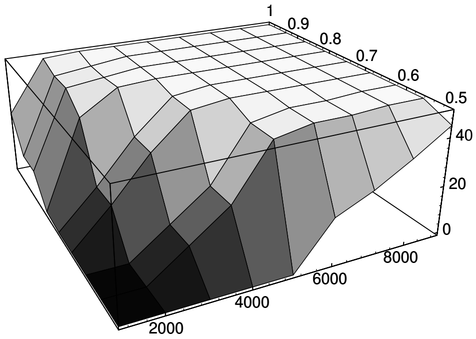
$\theta = 0.5$



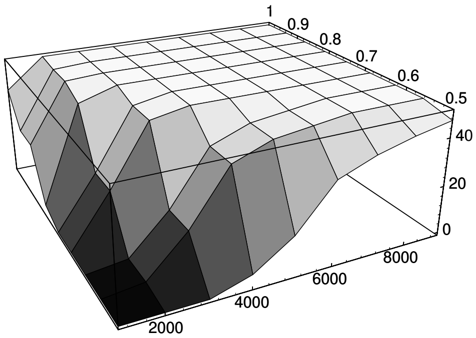
$\theta = 0.6$



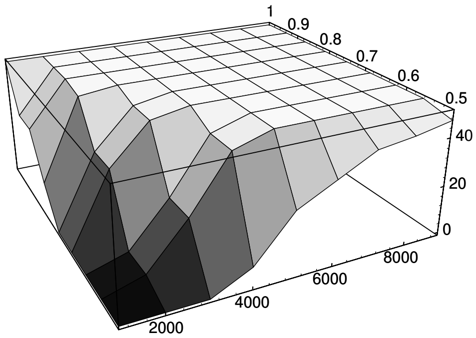
$\theta = 0.7$



$\theta = 0.8$



$\theta = 0.9$



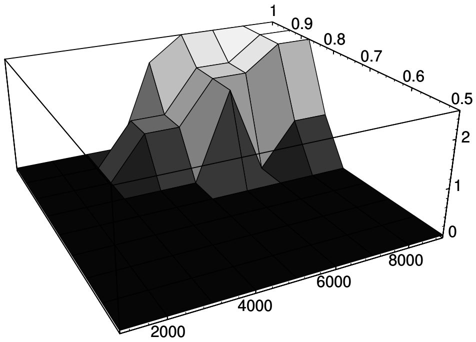
$\theta = 1.0$

Figure 11: Number of successes plotted as a function of p and the walk length for different values of θ for algorithm CMFO on the *CIT 4* environment with uniform error model

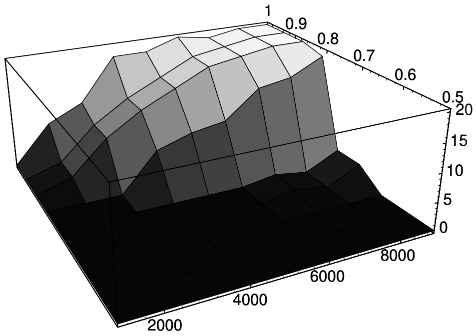
Figures 11, 12, and 13 show the performance of algorithm CMFO on the *CIT 4* environment with the uniform, oriented junction-type and unoriented junction-type error models, respectively. Each figure shows that the algorithm's performance inproves steadily as the uncertainty in action and in observation decrease and as the number of steps increases. The figures also provide a comparison of the different error models. Although the theorem guarantees the performance of the algorithm only when $P > 1/\sqrt{2\theta}$, data was collected with values of P and θ beginning at $1/2$. In the case of the uniform error model, complete success is obtained even for combinations of these parameters that are disallowed by the theorem. This is a result of the benign nature of the uniform noise model. In the oriented partition model, performance is very poor until the requirements are met, (in particular until P reaches $1/\sqrt{2}$) because this model comes much closer to the pessimistic assumptions used in the theorem. The unoriented model performs much more closely to the uniform model, and this is attributable to the fact that the partitions are large in this model, none smaller than size 6. In addition, the walk lengths used in the simulation were much shorter than those suggested by the theorem, by roughly 2 orders of magnitude. The success of the algorithm on shorter walks is due both to the factors just mentioned and to the looseness of a result concerning random walks used in our proof. Figure 14 shows the data gathered in a different form. Here, the number of steps (in thousands) needed by the algorithm to infer a correct map in each of twenty trials is shown for different values of P and θ. The "plateau" areas of these plots represent areas in which the algorithm failed to get twenty perfect answers with less than 10,000 steps. Comparison of these figures with Figure 6, reproduced in the bottom right plot, shows that these failures are not unexpected; they occur in a region that may have very low or negative separation.

6.4 Application to a Real Mapping System

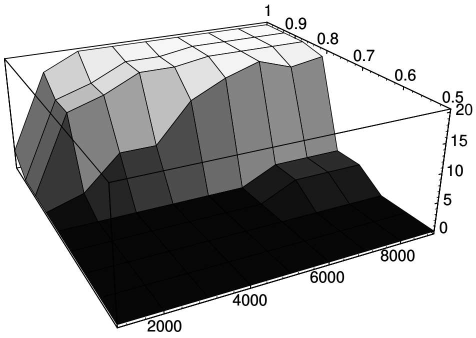
In the earlier analyses, simplifications were made for the sake of mathematical tractability. For example, we assume independent observation and movement errors and ignore sources of systematic error. Despite these simplifying assumptions, we believe that our models are relevant to a variety of interesting tasks and environments. Our simulations were based on the problem faced by mobile robots in learning about their spatial environment. In this section we briefly describe a real system for robotic map building based on the CMFO algorithm from Section 5.3; more detail is available elsewhere [Basye, 1992].



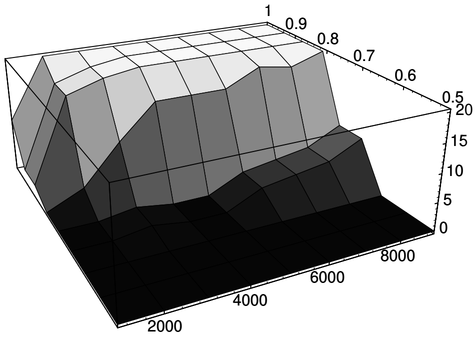
$\theta = 0.5$



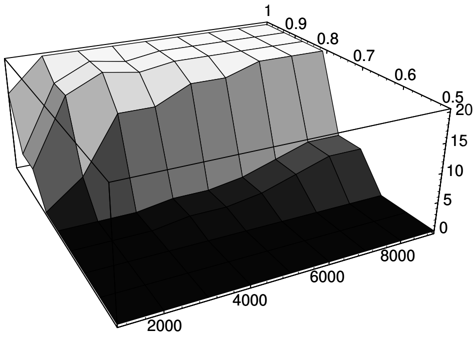
$\theta = 0.6$



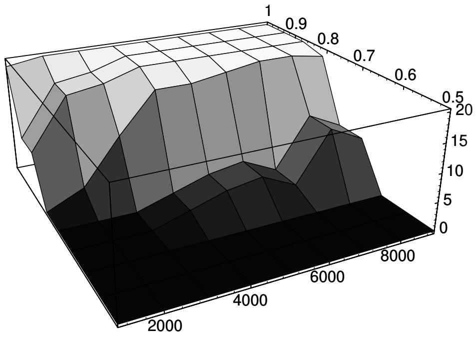
$\theta = 0.7$



$\theta = 0.8$

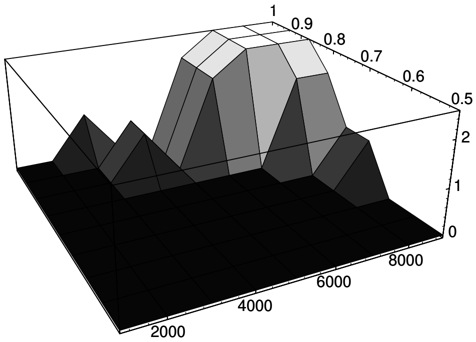


$\theta = 0.9$

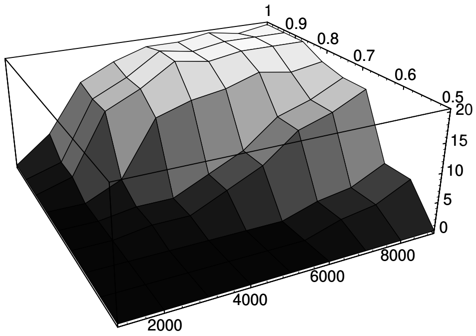


$\theta = 1.0$

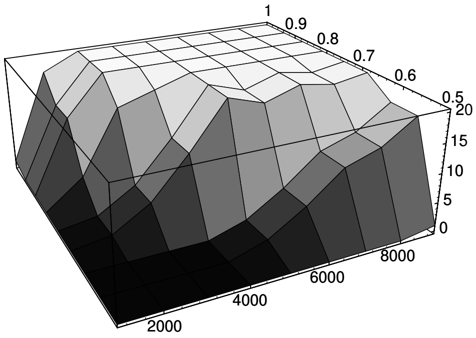
Figure 12: Number of successes plotted as a function of p and the walk length for different values of θ for algorithm CMFO on the *CIT 4* environment with the oriented junction-type similarity partition error model



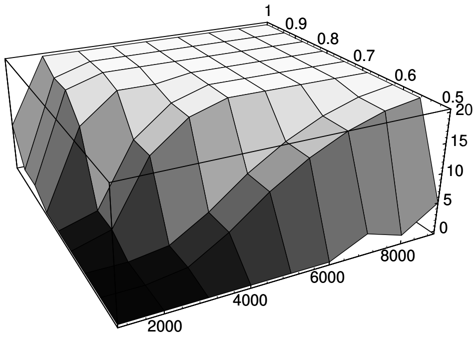
$\theta = 0.5$



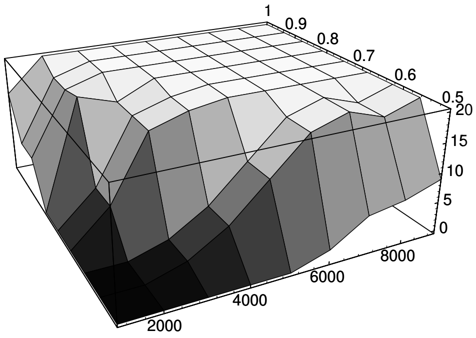
$\theta = 0.6$



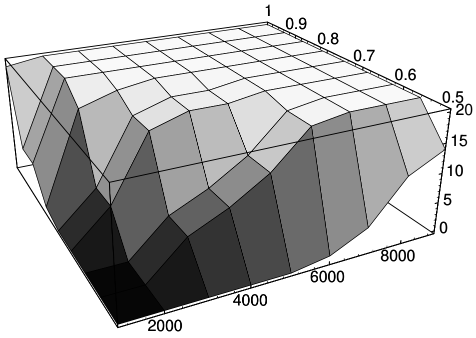
$\theta = 0.7$



$\theta = 0.8$

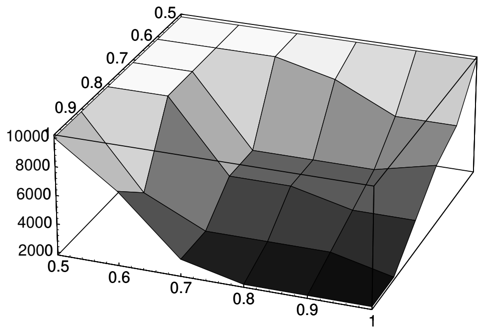


$\theta = 0.9$

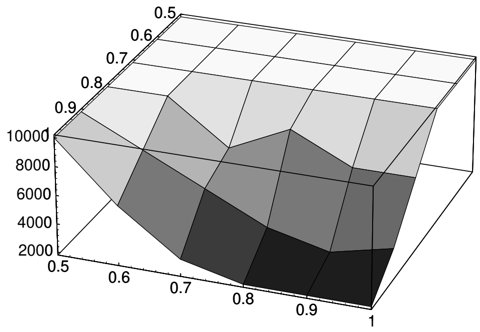


$\theta = 1.0$

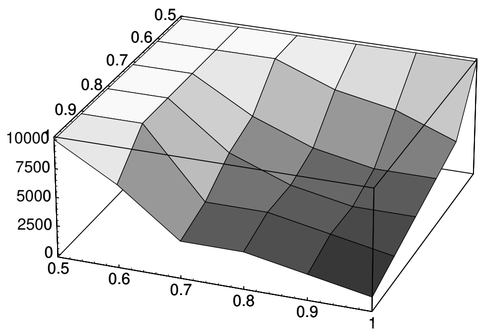
Figure 13: Number of successes plotted as a function of p and the walk length for different values of θ for algorithm CMFO on the *CIT 4* environment with unoriented junction-type similarity partition error model



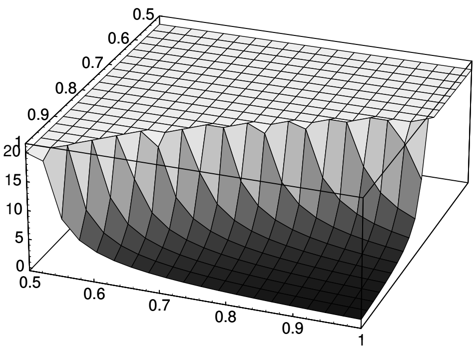
CIT4 Unoriented



CIT4 Oriented



CIT4 Uniform



$1/s$

Figure 14: Number of steps for first 100% success (N = 20) plotted as function of P and the walk length for different values of θ for algorithm CMFO on the *CIT 4* environment with three similarity partition error models; the bottom right plot is the $1/s$ factor from Theorem 2.

Recall that the CMFO algorithm required, in addition to several structural features, a nominally unique output at each state which whose observation probability was above some threshold. This requirement was satisfied by combining information about orientation, junction type and position at each location. The robot used was equipped with eight sonar transducers and an odometer which were used to generate this information. Robust high-level movement procedures were implemented as combinations of simpler traversal strategies; these procedures were used as the algorithm's actions. The robot's sensing and movement procedures were designed to operate in common hallway environment such as that corresponding to the *CIT 4* simulation environment. The CMFO algorithm was modified slightly from the version presented above; these modifications allowed more efficient exploration and avoided taking actions which could be predicted to fail.

The goal of the modifications and development of sensing and movement strategies was to allow correct identification of the environment using a number of steps well below the number required by the analysis of Section 5.3, on the order of a small constant times the number of states in the environment. This goal was achieved; The robot successfully built models using $3|Q|$ steps.

7 Discussion and Open Problems

The system identification problems discussed here can be thought of as points in a large space of such problems, each with its own set of structural and interaction properties. Our goal in this work has been to provide solutions to several representative problems from a portion of this space characterized by noisy inputs and/or outputs. Our interest in this part of the space derives directly from our interest in solving identification problems in the real world, where such noise is unavoidable.

These results provide indications about which problems might be realistically approached with current robotic systems. For example, although we have not provided direct comparisons of the complexity of the problems explored here, both our theoretical and empirical results suggest that having nominally unique labels is an enormous advantage. The robotic system described earlier was designed with this knowledge in mind. With regard to problems involving spatial exploration, there may be other assumptions that can be made which allow even better results.

The solutions to problems involving non-unique outputs generally involve sequences, in particular distingushing sequences. Gill [Gill, 1961] provides a way to construct preset distinguishing sequences when they exist, but these may have length exponential in the number of states and the algorithm requires a complete description of the automaton. Yannakakis and Lee [Yannakakis and Lee, 1991] show that the problem of determining whether an automaton has a preset distinguishing sequence is PSPACE-complete, but also give an efficient, constructive algorithm for determining whether an automaton has an adaptive distinguishing sequence. As regards uncertainty in perception, the following two important questions remain to be resolved.

- Suppose distinguishing sequences exist and observation and movement are uncertain? Are adaptive sequences still easy to find?
- Suppose the agent is given an adaptive distinguishing sequence. Is it easy to identify the underlying automaton with observation and movement uncertainty?

8 Conclusions

This paper investigates algorithms for agents to identify the input/output behavior of the dynamical system corresponding to their environment. In our model, the system is represented as a deterministic finite automaton with a relatively small number of states and actions. While we admit that the real world cannot be modeled by such automata to any high degree of predictive accuracy, expediency requires and nature has provided the means of simplifying the huge amounts of data available to our senses. Biological systems appear to be equipped with robust perceptual and motor routines that serve to abstract and considerably reduce the complexity of the real world. We believe that, lacking such routines, learning is impossibly hard. In addition, our approach does not require an agent to learn one automaton to represent the full range of its interaction with the environment; rather, different aspects of that interaction would be modeled with different automata. We claim then that, given appropriate perceptual and action routines, it makes sense to model system identification in terms of inferring automata.

In this paper, we address the problem of dealing with the inevitable errors that occur in perception and movement. Given our model, we show that errors in perception that do not affect movement are rather easy to contend with if the goal is polynomial-time, high-probability approximation. A general method of dealing with errors in both observation and movement without any means of establishing ground truth as a basis for filtering hypotheses has so far eluded us. We have, however, provided algorithms that work for the case in which there are landmarks distributed throughout the environment and the agent has some means of determining how it got somewhere without necessarily knowing where it came from. We have also provided algorithms for the case in which all states have unique signatures but both observation and movement are noisy.

In additional to our theoretical results, we have performed extensive experimental studies that indicate that, for a class of relatively benign but nevertheless realistic environments, our bounds are quite conservative. Ultimately, we are seeking algorithms that can learn a high-probability approximation to the correct, underlying environmental model in time some small constant factor of the size of the underlying model even in the presence of occasional errors. In our work on real mobile robots, we are approaching that goal.

Acknowledgments

Dana Angluin and Sean Engelson provided insights and corrections to the algorithm and proof of section 5.1. Oded Maron and Evangelos Kokkevis also participated in discussions of these algorithms and provided helpful simulation results. Jeffrey Vitter participated in the development of the landmark algorithm. Philip Klein provided useful pointers to the literature of random walks in graphs. Several anonymous reviewers also provided helpful suggestions.

References

[Angluin, 1978] Dana Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

[Angluin, 1987] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[Bachrach, 1992] Jonathan R. Bachrach. Connectionist modeling and control of finite state environments. Technical Report 92-6, University of Massachusetts at Amherst Department of Computer and Information Science, 1992.

[Basye *et al.*, 1989a] Kenneth Basye, Thomas Dean, and Jeffrey Scott Vitter. Coping with uncertainty in map learning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 663–668. IJCAI, 1989.

[Basye *et al.*, 1989b] Kenneth Basye, Thomas Dean, and Jeffrey Scott Vitter. Coping with uncertainty in map learning. Technical Report CS-89-27, Brown University Department of Computer Science, 1989.

[Basye, 1992] Kenneth J. Basye. *A Framework for Map Construction.* PhD thesis, Brown University Department of Computer Science, Providence, Rhode Island, 1992.

[Dean *et al.*, 1990] Thomas Dean, Kenneth Basye, Robert Chekaluk, Seungseok Hyun, Moises Lejter, and Margaret Randazza. Coping with uncertainty in a control system for navigation and exploration. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1010–1015. American Association for Artificial Intelligence, 1990.

[Dean *et al.*, 1992a] Thomas Dean, Dana Angluin, Kenneth Basye, Sean Engelson, Leslie Kaelbling, Evangelos Kokkevis, and Oded Maron. Inferring finite automata with stochastic output functions and an application to map learning. In *Proceedings of the Tenth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence, 1992.

[Dean *et al.*, 1992b] Thomas Dean, Dana Angluin, Kenneth Basye, Sean Engelson, Leslie Kaelbling, Evangelos Kokkevis, and Oded Maron. Inferring finite automata with stochastic output functions and an application to map learning. Technical Report CS-92-27, Brown University Department of Computer Science, 1992.

[Feller, 1970] William Feller. *An Introduction to Probability Theory and its Applications.* Wiley, New York, third edition, 1970. revised printing.

[Gill, 1961] Arthur Gill. State-identification experiments in finite automata. *Information and Computation*, 4:132–154, 1961.

[Gold, 1972] E. Mark Gold. System identification via state characterization. *Automatica*, 8:621–636, 1972.

[Gold, 1978] E. Mark Gold. Complexity of automaton identification from given sets. *Information and Control*, 37:302–320, 1978.

[Kaelbling *et al.*, 1992] Leslie Kaelbling, Kenneth Basye, and Thomas Dean. Learning labelled graphs from noisy data. In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*. Center for Systems Science, 1992.

[Mihail, 1989] Milena Mihail. Conductance and convergence of Markov chains: A combinatorial treatment of expanders. In *Proceedings of the 31st ACM Symposium on Foundations of Computer Science*, pages 526–531, 1989.

[Moore, 1956] Edward F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, pages 129–153. Princeton University Press, Princeton, New Jersey, 1956.

[Pitt and Warmuth, 1989] Leonard Pitt and Manfred K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*, pages 421–432, 1989.

[Rivest and Schapire, 1987] Ronald L. Rivest and Robert E. Schapire. Diversity-based inference of finite automata. In *Proceedings of the 29th ACM Symposium on Foundations of Computer Science*, pages 78–87, 1987.

[Rivest and Schapire, 1989] Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the Twenty First Annual ACM Symposium on Theoretical Computing*, pages 411–420, 1989.

[Schapire, 1991] Robert E. Schapire. The design and analysis of efficient learning algorithms. Technical Report MIT/LCS/TR-493, MIT Laboratory for Computer Science, 1991.

[Servan-Schreiber *et al.*, 1989] David Servan-Schreiber, Axel Cleeremans, and James L. McClelland. Learning sequential structure in simple recurrent networks. In David Touretzky, editor, *Advances in Neural Information Processing*, volume 1. Morgan Kaufman, 1989.

[Sutton, 1990] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings 7th International Conference on Machine Learning*, 1990.

[Yannakakis and Lee, 1991] Mihalis Yannakakis and David Lee. Testing finite state machines. In *Proceedings of the 23rd ACM Symposium on Theoretical Computing*, pages 476–485. ACM Press, 1991.