Shortest Paths in a Dynamic Uncertain Domain

David Meir Blei
Artificial Intelligence Center
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
blei@ai.sri.com

Leslie Pack Kaelbling
Computer Science Department
Box 1910
Brown University
Providence, RI 02912
lpk@cs.brown.edu

Abstract

This paper describes solutions to finding shortest paths in stochastic graphs with partially unknown topologies. We consider graphs which are both static and dynamic. We solve the static problem by reduction to a Markov decision process and solve the dynamic problem by reduction to a partially observable Markov decision process. We show these solutions to be intractable and explore reinforcement learning as a method of approximation. Finally, we present empirical results of a reinforcement learning approach in this framework.

Suppose we are trying to deliver an important package to a town in a cluster of small islands. These islands have recently been struck by a terrible storm and we can't be sure of the status of each bridge. Some of them are intact but many of them washed away or are otherwise unusable. What can we do in such a situation?

Suppose now that there are efforts to fix some of the bridges while other bridges continue to fall apart due to additional rains. Now how can we plan to deliver the package?

These planning problems are interesting and difficult. We know something about the state of the world but need to observe it to be sure of that state. Furthermore, we want to deliver the package to the town as soon as possible and avoid wasting valuable time and resources on improbable and long paths through the islands.

1 Introduction

In this paper we explore algorithms for finding the shortest path to the goal island in the kinds of stochastic dynamic environments described above. We refer to the first kind of world as a *bridge problem* (BP). A BP is an undirected graph of islands and bridges (nodes and edges) where each bridge is assigned an initial probability of being intact (figure 1). Additionally, one island is considered the goal island and an agent traverses this graph trying to reach that goal.

Note that the probabilities on the bridges represent the agent's prior belief that the bridge is intact. After an agent tries to cross a bridge, it discovers the true state of that bridge. We assume all future attempts to cross it will yield the same results.

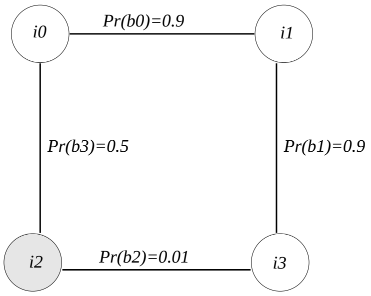


Figure 1: An example of a bridge problem. The goal island is *i2*.

A solution to the bridge problem is a behavior for the agent that minimizes the expected path length to the goal. The agent needs to consider its past discoveries about bridges as well as its current position in determining its next action.

Since the agent's current knowledge and position are based on its previous knowledge and last action, this problem lends itself to a solution using *Markov decision processes* (MDP's). We will show that this problem can be solved with an MDP, though its size grows too quickly with the size of the BP for this to be a practical solution.

The second scenario is represented as a *dynamic bridge problem* (DBP) where the status of a bridge can fluctuate even after an agent's discovery of whether it is intact. We will see that an MDP is no longer adequate to model an agent in a DBP. However, we can represent a DBP with a *partially observable Markov decision process* (POMDP). Unfortunately, exact solutions to POMDP's are intractable in this case. We use reinforcement learning techniques to approximate a solution.

2 The Bridge Problem

A BP is represented as a tuple $(\mathcal{I}, \mathcal{B}, g)$ where:

- $\mathcal{I}$ is a finite set of islands in the world.
- $\mathcal{B}$ is a set of bridges. Each bridge b is a tuple (i_0, i_1, p) where $i_0, i_1 \in \mathcal{I}$ are the islands connected to the bridge and $0 \leq p \leq 1$ is the probability that the bridge is

intact. We write b_{i_0},b_{i_1}, and b_p for the different parameters of each bridge. The bridges in this case are undirected. Note however that it is easy to extend the model to the directed case.

- $g \in \mathcal{I}$ is the goal island in the world.

Solving this problem means defining an agent that acts to minimize the expected length of its path to the goal. To find this behavior, we use an MDP to model the agent's movement between islands and discovery of whether bridges are intact or not. Acting optimally in such an MDP is the optimal behavior for an agent in the BP.

2.1 Overview of Markov decision processes

MDP's are well described by Puterman [1994]. Recall that a finite MDP is represented by a tuple, $(\mathcal{S}, \mathcal{A}, T, R)$ where:

- $\mathcal{S}$ is a finite set of states in the world.
- $\mathcal{A}$ is a finite set of actions.
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state-transition function. For each state and action, T gives a probability distribution over next states. $T(s, a, s')$ represents the probability of reaching state s' given that the agent began in state s and took action a. Since T is a probability distribution, $\sum_{s' \in \mathcal{S}} T(s, a, s') = 1$ for any given $s \in \mathcal{S}$ and $a \in \mathcal{A}$.
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the reward function. $R(s, a)$ represents the expected reward the agent receives for taking action a in state s.

A solution to an MDP is the policy π^*, a mapping of states to actions, that maximizes the expected discounted future reward,

$$E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

for every starting state. The discount factor $\gamma \in [0, 1]$ measures the degree to which long-term rewards are valued and r_t represents the given reward at time t.

2.2 Converting a BP to an MDP

We can find an optimal policy for our agent in the BP by reducing the problem to an MDP. We need a mapping from $(\mathcal{I}, \mathcal{B}, g)$ to $(\mathcal{S}, \mathcal{A}, T, R)$.

The BP can not be directly modeled as an MDP with states corresponding to islands. If we expand the notion of state however to include the agent's unfolding knowledge of the status of the bridges, then a BP can be reduced to a finite-state MDP. Our optimal policy in that MDP will combine directed movement to the goal and information gain from attempting to cross bridges.

States and Actions

The agent's state is a combination of its location in the world and what it currently knows about the bridges. Therefore, the state space of the MDP is the cross product of the islands with all the possible knowledge states about the bridges.

The set of knowledge states is all the possible instantiations of bridge probabilities where each bridge can have probability 1 (the bridge is intact), 0 (the bridge is not intact), or b_p (the agent does not know whether the bridge is intact). Formally, we describe the MDP states as

$$\mathcal{S} = \mathcal{I} \times \prod_{b \in \mathcal{B}} \{0, 1, b_p\}.$$

The actions of the MDP are the bridges of the BP, $\mathcal{A} = \mathcal{B}$.

Transition Function

In building the transition function, we need to model physical movement in the world and discovery about the world from taking an action. For example, suppose an agent in the BP in figure 1 is at *i1* and is considering taking bridge *b1*. It thinks that *b1* has probability 0.9 and tries to cross it. Discovering that the bridge is not there, it knows it has remained on *i1* and bridge *b1* has probability 0. Similarly, if it successfully crosses the bridge, it knows that it has changed location to *i3* and the bridge has probability 1. In both cases, the agent's state, a combination of location and knowledge state, has changed. The probability of reaching the state where the bridge is intact and the agent's location has changed is 0.9 (the probability of the bridge in the current knowledge state). The probability of reaching the state where the agent's location remains the same and the bridge is damaged is 0.1.

We formally describe the transition function as

$$T(s, b, s') = \begin{cases} s_{ks}(b) & \text{if } s'_l \in \{b_{i_0}, b_{i_1}\}, \\ & s_l \in \{b_{i_0}, b_{i_1}\}, \\ & s_l \neq s'_l, \text{ and} \\ & s'_{ks}(b) = 1. \\ 1 - s_{ks}(b) & \text{if } s'_l = s_l, \\ & s_l \in \{b_{i_0}, b_{i_1}\}, \\ & s'_{ks}(b) = 0. \\ 0 & \text{otherwise} \end{cases}$$

where s_l represents the location parameter in a state and $s_{ks}(b)$ is the probability of a bridge in the knowledge state of state s. Note that when a bridge is unconnected to the location of either state, $T(s, b, s') = 0$.

Reward Function

Finally, we calculate the reward function. For this function, we can disregard the knowledge states since we are not concerned with how much the agent knows as long as it somehow reaches the goal island. The states where the agent is at the goal have reward 1 and all other states have reward 0. Furthermore, we assume that the agent enters a zero-cost absorbing state once it reaches the goal. Formally,

$$R(s, b) = \begin{cases} 1 & \text{if } s_l = g \\ 0 & \text{otherwise.} \end{cases}$$

Analysis

Now, we can use a dynamic programming algorithm called *value iteration* to find the optimal policy of this MDP and compute a mapping from knowledge states and locations to bridges. An agent, based on what it knows about the bridges and where it is in the world, can attempt to take the optimal bridge. Furthermore, the optimal policy takes into account both the value of gaining information about uncertain bridges and the value of moving closer to the goal.

Note that this solution is better then a simple greedy method of repeatedly computing the most probable path to the goal and taking the first step of that path. The MDP agent has defined a policy for all contingencies in the world and can choose its action in constant time. It is faster than the greedy solution and implicitly takes into account future options (or lack of them) in its plan.

Unfortunately, it is intractable to solve reasonably sized bridge problems exactly. Though value iteration runs in time polynomial in $|S|$ and $|A|$ [Littman *et al.*, 1995a], the states in the MDP grow exponentially with the bridges in the BP. There are $3^{|B|}$ knowledge states in the BP since each bridge can have one of three values. Therefore, there are $|\mathcal{I}| * 3^{|B|}$ states in the constructed MDP and value iteration runs in time exponential in $|B|$.

3 The Dynamic Bridge Problem

The BP is limited to static worlds. The Dynamic Bridge Problem (DBP) is similar to a BP except that the underlying structure of the world can change. As a result of this, the certainty of an agent's knowledge about a bridge diminishes as time passes since it last observed that bridge. This aspect of the problem gives rise to a continuous space of knowledge states, and finite MDP's are no longer sufficient to represent the problem.

We represent a DBP by a tuple $(\mathcal{I}, \mathcal{B}, g)$ where:

- $\mathcal{I}$, g are as in the BP.
- $\mathcal{B}$ is a set of *dynamic* bridges. Each bridge is a tuple (i_0, i_1, p, f, c) where $i_0, i_1 \in \mathcal{I}$ are the islands connected to the bridge and $0 \leq p \leq 1$ is the probability that the bridge is initially intact (as in the BP). At each time step, f is the probability that the bridge is fixed and c is the probability that the bridge breaks (c stands for "crashes" or "crumbles").

The solution to the BP hinges on there being a finite number of knowledge states which we represent in the states of a finite MDP. In the case of the DBP however, each bridge can have an infinite number of values as its probability. Therefore, there are infinitely many knowledge states and it is no longer possible to solve using a finite MDP. A DBP can be reduced to a more powerful model, a partially observable Markov decision process (POMDP).

3.1 Partially Observable Markov Decision Processes

A POMDP is a model in which the agent is not sure of its state but makes observations as it acts in the world. To act optimally in a POMDP, the agent needs to select its action based its history of observations and actions [Kaelbling *et al.*, 1998].

Recall that a POMDP is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O)$ where

- $(\mathcal{S}, \mathcal{A}, T, R)$ is a Markov decision process.
- Ω is a set of observations. The agent can make one of these observations each time it takes an action.
- $O : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ is the observation function. For each action and state, O determines a probability distribution over all the observations. $O(s', a, o)$ is the probability of observing o given that we took action a and ended up in state s'. Since $O(s, a)$ is a probability distribution, $\sum_{o \in \Omega} O(s', a, o) = 1$ for some $s' \in \mathcal{S}$ and $a \in \mathcal{A}$.

Finding the optimal policy in a POMDP amounts to solving a continuous MDP where the state is a *belief state*, a probability distribution over $\mathcal{S}$. Methods of solving this MDP can be found in the overview by Kaelbling *et al.* [1998].

3.2 Representing a DBP as a POMDP

In order to reduce a DBP to a POMDP, we describe a mapping from $(\mathcal{I}, \mathcal{B}, g)$ to $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O)$.

States and Actions

In the bridge problem, we defined the knowledge state to model the uncertainty about the world. In the reduction of the POMDP to a continuous MDP, the belief states will implicitly model the agent's uncertainty. Therefore, the POMDP states simply need to model all the different possible worlds the agent *may* be in. In a given instance of the world, each bridge can either be intact or not. We define a *world configuration* to be a vector of length $|\mathcal{B}|$ of T's and F's. If the i^{th} element is T, the i^{th} bridge is intact and if the i^{th} element is F, the i^{th} bridge is not intact. The states of the POMDP are all the possible world configurations coupled with the possible locations. Formally, we define the world configurations, and states in the POMDP as

$$\begin{aligned} W &= \textstyle\prod_{b \in \mathcal{B}} \{\mathtt{T}, \mathtt{F}\} \text{ and} \\ \mathcal{S} &= \mathcal{I} \times W. \end{aligned}$$

As in the bridge problem, the set of actions is the set of bridges, $\mathcal{A} = \mathcal{B}$.

Observations

The POMDP agent is not sure of its state but can make observations on each action. In the DBP world, an agent observes that the bridge it just took was *in* or *out*. Therefore, $O = \{in, out\}$.

The observation function is deterministic because the agent will never make a false observation about a bridge. Assuming it is standing at one end of the bridge it wants to try, it will always observe *in* if that bridge is intact and *out* if it is not. Furthermore, the intactness of the bridge in the current world configuration of the agent should match the agent's observation.

$$O(s,s',a,o) = \begin{cases} 1 & \text{if } s'_w(a) = \mathtt{T},\ o = in, \\ & \text{and } s_l \neq s'_l \in \{a_{i_0}, a_{i_1}\} \\ 1 & \text{if } s'_w(a) = \mathtt{F},\ o = out, \\ & \text{and } s_l = s'_l \in \{a_{i_0}, a_{i_1}\} \\ 0 & \text{otherwise.} \end{cases}$$

where s_w(a) is the status of bridge a in the world configuration of s, s_l is the location of the agent in state s, and $\{a_{i_0}, a_{i_1}\}$ are the islands that bridge a connects. Notice that this observation function deviates from the formal definition of a POMDP since it takes into account the current state of the agent. We have described a mapping from $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \Pi(\Omega)$. The POMDP model requires an observation function of the form $\mathcal{S} \times \mathcal{A} \to \Pi(\Omega)$. If we square the state space and consider each possible pair (s, s') to be a state, we can easily reduce our observation function to the standard form.

The Transition Function

To construct the transition function, we first analyze which states are reachable from some state s with non-zero probabilities. If $s_l = a_{i_0}$ and $s_w(a) = \mathtt{T}$, the agent can reach all those states where $s_l = a_{i_1}$ and $s_w \in W$. In other words, if the bridge is intact in s and the agent is at one of the connecting islands of the bridge, it can reach any world configuration (since the world can arbitrarily change without the agent's knowledge) coupled with the other connecting island of that bridge. In all other cases, the agent can reach all world configurations coupled with its current location.

The actual probability of reaching state s' from state s depends on how the world configuration changes from the source state to the destination state. If, for some bridge, that bridge changes status from F to T we factor in the probability that the bridge is fixed. If it changes from T to F we factor in the probability that the bridge breaks. If it stays the same at T, we factor in the probability that the bridge is intact and doesn't break. Finally, if it stays the same at F we factor in the probability that the bridge is broken and is not fixed. Formally,

if $s'_l \neq s_l \in \{a_{i_0}, a_{i_1}\}$ and $s_w(a) = \mathtt{T}$
or if $s'_l = s_l \in \{a_{i_0}, a_{i_1}\}$ and $s_w(a) = \mathtt{F}$,

$$T(s,a,s') = \prod_{b \in \mathcal{B}} \begin{cases} b_f & \text{if } s_w(b) = \mathtt{F} \\ & \text{and } s'_w(b) = \mathtt{T} \\ b_c & \text{if } s_w(b) = \mathtt{T} \\ & \text{and } s'_w(b) = \mathtt{F} \\ 1 - b_f + b_f b_c & \text{if } s_w(b) = \mathtt{T} \\ & \text{and } s'_w(b) = \mathtt{T} \\ 1 - b_c + b_f b_c & \text{if } s_w(b) = \mathtt{F} \\ & \text{and } s'_w(b) = \mathtt{F} \end{cases}$$

Otherwise, $T(s,a,s') = 0$.

The Reward Function

The reward function, as in the BP, rewards only the set of states where the agent's location is the goal island.

$$R(s,a) = \begin{cases} 1 & \text{if } s_l = g \\ 0 & \text{otherwise.} \end{cases}$$

Analysis

Now that we have reduced a DBP to a POMDP, we can find an optimal policy for this problem using known methods of solving POMDP's. However, common methods for solving POMDPs can only handle cases of up to about 30 states. In our reduction, the number of states is exponential in the size of the DBP so we cannot easily solve problems of more than 5 bridges.

3.3 Factoring the Belief State Space

We can take advantage of the nature of the DBP to reduce the size of the belief state MDP that we constructed in the previous section. The state space of this MDP has dimension $|\mathcal{I}| * 2^{|B|}$ but we can factor out unnecessary information with two insights.

The agent always knows its location in the DBP. Only a set of world configurations coupled with one location can have non-zero probabilities at one time. Therefore, each belief state is a point in one of $|\mathcal{I}|$ spaces of dimension $2^{|\mathcal{B}|}$.

We call a point in one of these spaces a *world configuration belief state*; it essentially represents the agent's current beliefs about the status of each bridge. However, we encode it as a distribution over all possible configurations even though the bridges change independent of each other. In fact, we can compute the probability of an entire world configuration as the product of the probabilities of the states of the individual bridges. This allows us to represent the distribution with a single probability for each bridge reducing the state space to dimension $|B|$ from $2^{|B|}$.

In factoring the belief state space, the resulting belief state MDP becomes even more difficult to solve. The value of a belief state ψ is the sum of the values of the configurations weighted by their probabilities. In the unfactored state space, each configuration was represented with its own dimension. Thus, $\sum_{s \in \mathcal{S}} \psi(s)V(s)$ is linear in the space of probability distributions over configurations. This property lets us solve the continuous MDP [Kaelbling *et al.*, 1998]. In the factored state space, however, the probability of a given configuration is a product of the components of s. Therefore, V is no longer linear in ψ and we can no longer solve the POMDP using known exact methods.

4 Reinforcement Learning in the DBP

In previous sections, we showed exact solutions to the BP and DBP. These solutions are intractable for large problems and even for small problems in the case of the DBP. In this section we will demonstrate one method of approximating the optimal value function in reasonable time.

There have been many applications of value-function approximation to the solution of POMDPs [Hauskrecht, 1998; Littman *et al.*, 1995b; Parr and Russell, 1995]. The most successful take direct advantage of the known piecewise-linearity and convexity of the value function. These methods are not applicable to our factored belief-space representation so we use reinforcement learning with a nearest-neighbor approximation method.

Reinforcement learning (RL) is a method of approximating good behavior in a unpredictable world. RL is well described by Sutton and Barto [1998]. Using RL to approximate solutions to continuous state MDP's is described by Bertsekas and Tsitsiklis [1996].

We use RL in the DBP framework as follows. The learning agent uses a function approximator to estimate the value function of the factored belief state space (section 3.3). The agent simulates life in the bridge world for a fixed number of steps, keeping a record of its states and actions. At each step, the agent sometimes chooses its current idea of the optimal action (using the value function approximator) and sometimes chooses random actions. At the end of each iteration, the agent updates the function approximator with its new experiences and repeats.

Algorithm *LEARN*-DBP

1. $s \leftarrow \text{DBP}_{start}$
2. $\mathcal{T} \leftarrow \{\}$
3. **for** $i \leftarrow 1$ **to** n
4.   **if** we want to explore
5.   **then** $a \leftarrow$choose a random bridge
6.   from s
7.   **else** $a \leftarrow$approximate the best a
8. $s \leftarrow$simulate taking action a from s
9. $\mathcal{T} \leftarrow \mathcal{T} \cup \langle s, a \rangle$
10.   **if** $s = \text{DBP}_{goal}$ **or** we choose to restart
11.   **then** $s \leftarrow \text{DBP}_{start}$
12. **for** $\langle s, a \rangle$ **in** $\mathcal{T}$
13. $\mathcal{V}(s, a) \leftarrow R(s, a) +$
14. $\gamma \sum_{s' \in S} T(s, a, s')\mathcal{V}(s, a)$
15. **goto** step 1

In the algorithm above, DBP is a dynamic bridge problem where DBP_{start}, DBP_{goal} are the starting and goal islands respectively. The transition function T, reward function R, and discount factor γ are as defined by the reduction of DBP to a POMDP (section 3.2). Finally, $\mathcal{V}$ is a value function approximation architecture and $\mathcal{V}(s, a) \leftarrow \mathcal{R}$ notates updating that architecture for point (s, a) with the given value.

Lines 1 and 2 initialize the agent to be in the starting island with no current experience. In lines 3-11, the agent chooses an action, executes it, and records the result. Finally, the agent updates its value function in lines 12-14 and repeats.

This algorithm works given a good function approximator but is very impractical. The agent starts out with a completely blank slate in a possibly large and unpredictable domain. It will eventually find its way to the goal but there is little chance that it will learn such a route except after many trials of random actions. To reduce the agent's ignorance, we introduce a third option for selecting an action, a *mentor algorithm*. The mentor algorithm gives the agent an action which is the next step in a reasonably good but suboptimal initial solution to the problem. Initially, the agent relies heavily on its mentor to lead it to the goal several times to learn a good route. As it learns, it begins to use its value function and exploration to eventually find a behavior which is better than its mentor's. The agent architecture is illustrated in figure 2.

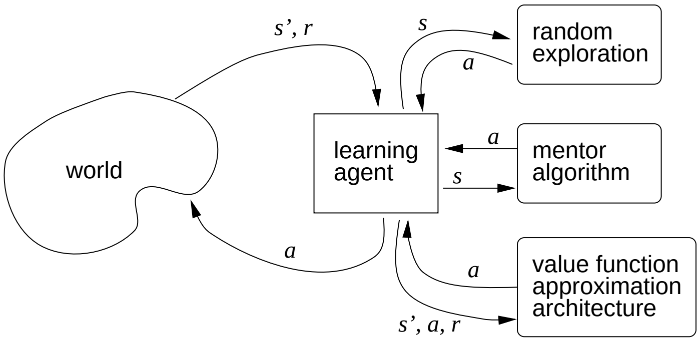


Figure 2: Reinforcement learning architecture in the dynamic bridge world. Note that the learning agent gets its next action from one of three sources.

In testing RL in the DBP framework we construct our agent as follows. The value function approximator is a nearest neighbor architecture. The value of a given point in the state space is simply the value of the closest point in Euclidean distance for which we have information. For a mentor algorithm, we use the next step on the currently most probable path to the goal. This step is computable in time $|B|$ but is not optimal since it does not take into account the dynamics of the bridges. The agent reduces its use of this algorithm linearly over the course of its learning. When not using the mentor algorithm, it explores 20% of the time and uses the value function approximator for the remaining 80%. Finally, we choose γ to be 0.95.

Our simulated agent navigated in Venice, Italy (figure 3) from and to islands on opposite sides of the city. There are 44 islands and 64 bridges and the parameters of each bridge, $\{b_p, b_f, bc\}$, were determined randomly. However, we fixed the mean values of these parameters so the world was parameterized by $\{\overline{b_p}, \overline{b_f}, \overline{b_c}\}$.

We tested the agent in 81 worlds with $\overline{b_c}$ and $\overline{b_f}$ ranging in values from 0.1 to 0.9. The agent trained by simulating 1000 steps as per the algorithm described above. In each training episode, it restarted after reaching the goal or 200 steps. The value function was updated with these 1000 points and the agent repeated this training 200 times.

To evaluate the trained agent, we ran 100 trials through the world. The agent stoped after reaching the goal or 200 steps and we measured the discounted reinforcement for each trial. The score of an agent in a

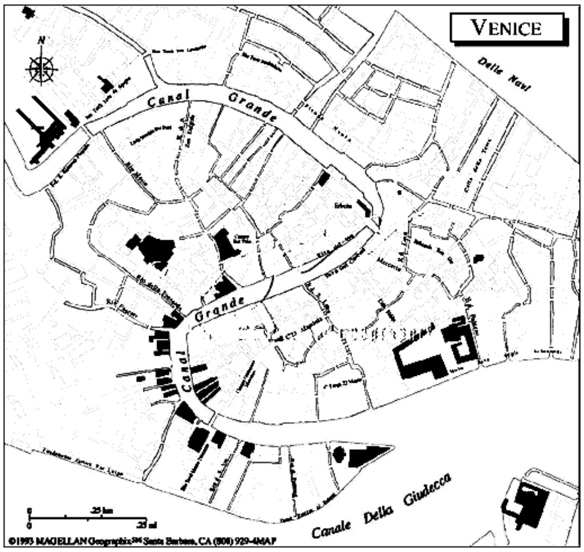


Figure 3: The testbed for the DBP learning agent: Venice, Italy.

given trial is $\sum_{i=1}^{200} \gamma^i r_i$ where r_i is the reward the agent received at time i. The total score is the sum of the 100 individual trial scores.

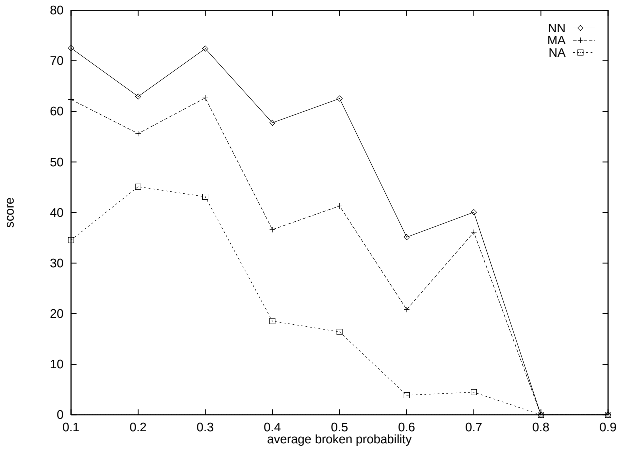
To evaluate the results, the learning agent's score is compared with the mentor algorithm and a similar but more naive algorithm. This naive algorithm works the same way as the mentor algorithm but doesn't take into account the actual probabilities of the bridges. Instead, each bridge can have one of three values: intact, not intact, or unknown. The naive agent scores paths to the goal based on these values and chooses the next step in the path with the highest score.

Generally speaking, the learning agent performs better than both algorithms. In figure 4, we present the performance of the three algorithms in worlds where bridges are often fixed ($\overline{b_f} = 0.7$), seldom fixed ($\overline{b_f} = 0.3$), and sometimes fixed ($\overline{b_f} = 0.5$). Note that in the case where $\overline{b_f} = 0.3$, the mentor algorithm and learning algorithm are closer in performance. Furthermore, as $\overline{b_c}$ increases all three graphs, the learning algorithm decreases in performance faster than the mentor algorithm.

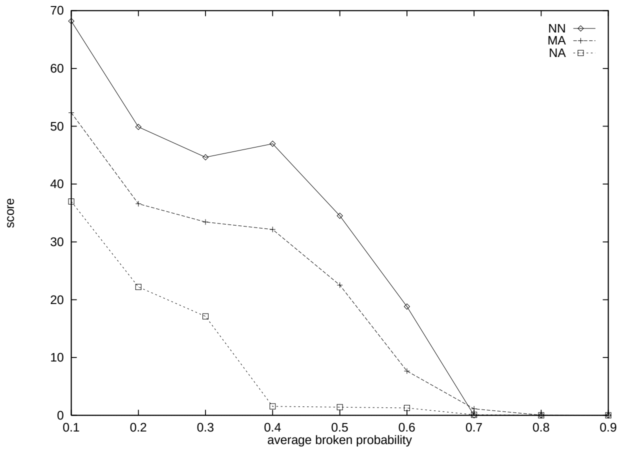
5 Conclusions and Future Work

This work demonstrates a practical use of MDP's and POMDP's in finding optimal behavior for an agent in a dynamic stochastic environment. Though exact solutions are intractable, we show that approximation methods such as reinforcement learning are effective.

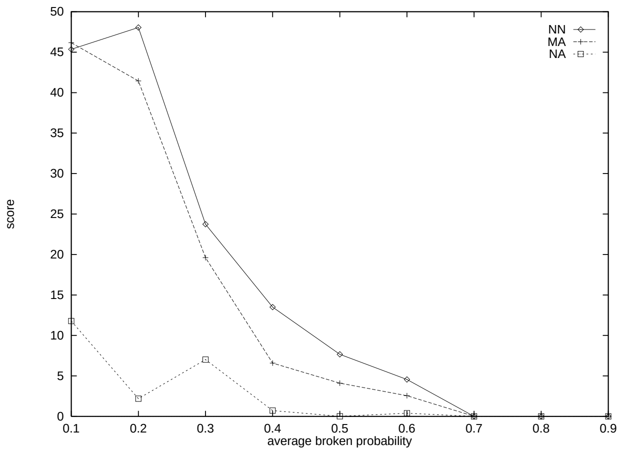
There are three areas of future work. First, it would be useful to find a different real-world domain, such as network packet routing, in which to implement and test this kind of reactive planning behavior. Second, it would be interesting to explore the problem of building a BP from experience in a stochastic world, computing an exploratory behavior to best induce the bridge probabil-



$\overline{b_f} = 0.7$



$\overline{b_f} = 0.5$



$\overline{b_f} = 0.3$

Figure 4: Results using RL in Venice. **NN** is the reinforcement learning score; **MA** is the mentor algorithm score; and **NA** is the score for the naive algorithm.

ities. Finally, we can make the model more expressive by removing the assumption that bridges are completely independent of each other. Instead, we would like to consider the case where sets of bridges are independent of each other but the bridges themselves have some degree of interdependence.

References

[Bertsekas and Tsitsiklis, 1996] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[Hauskrecht, 1998] Milos Hauskrecht. *Planning and Control in Stochastic Domains with Imperfect Information.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1998.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[Littman *et al.*, 1995a] Michael Littman, Thomas Dean, and Leslie Kaelbling. On the complexity of solving markov decision problem. In *Proceedings of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, 1995.

[Littman *et al.*, 1995b] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning.* Morgan Kaufmann, 1995.

[Parr and Russell, 1995] Ron Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes.* John Wiley and Sons, 1994.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew T. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.