# Discovering Conditions for Intermediate Reinforcement with Causal Models

Irvin Hwang

May 18, 2005

**Abstract**

Learning to perform a task in an environment with sparse feedback is a difficult problem. While several approaches for increasing feedback during learning have been taken, these methods suffer from the dependency on human knowledge and engineering to find good solutions. We propose using causal models to increase the amount of feedback that will improve learning. This approach does not require domain-specific human engineering because causal models can be constructed directly from the environment using empirical data. Preliminary experiments and results show causal models can be used to automatically discover conditions for applying intermediate feedback that accelerate learning.

## 1 Introduction

Reinforcement learning (RL) is a field of artificial intelligence that deals with the design and implementation of trial and error learning algorithms. Having programs that can learn through interaction with an environment rather than through explicit instruction reduces human effort required to solve problems and also opens up the possibility of discovering better than human solutions [1, 2]. The lack of human interaction makes RL difficult because feedback during the learning process can be very sparse and may not contain a lot information. Often times tasks are formulated such that reward (a single numerical value indicating correctness) is only received after successful completion of a complicated task. Our approach to improving RL is to increase the amount of feedback received during the learning process while keeeping the amount of human interaction to a minimum. We increase the amount of feedback received in each episode of a task by exploiting dependence relationships between environment variables. Human engineering is minimized by having the agents automatically build graphical models of the dependence relations through exploration and observation of their environment.

The notion of increasing feedback during the learning process is not new and has been shown to be beneficial for RL. Methods such as progress estimators, heterogeneous rewards , and hierarchical task decomposition [3, 4] improve learning through increased feedback, but require a priori knowledge of when and how to apply intermediate reinforcement, which goes against the direction of RL in minimizing human engineering. More recent work on increasing feedback approach this problem by finding critical bottleneck states that must be reached

before the goal state [5]. Our approach differs in that we focus on the relations between individual variable values and reward, which is more appropriate for certain tasks than using bottleneck states. Consider the domain of keepaway soccer. A behavior that you might want to positively reinforce is having a ball-keeper maximize her distance from the taker. In this case, there does not exist a single particular sub-goal state for which intermediate reinforcement should be applied upon reaching; instead the condition for feedback is dependent on a single state feature (distance to the ball) and so a broad sub-space of the entire state space contains good candidate "sub-goal" states. The idea of using environment variable values as conditions for applying feedback captures this idea.

The use of graphical models for understanding the relationships between variables has a rich background as evidenced by the prevalence of Bayesian networks [6]. The structure and orientation of probabilistic graphical models can be interpreted as reprsentation of causal relationships between variables [7, 8]. Research into the field of causality has produced algorithms for automatically determing graphical model structure from empirical data. The main difficulty with learning in sparse feedback environments is described as the temporal credit problem. The temporal credit problem deals with how to assign credit to critical actions in achieving a task given an entire sequence of relevant and possibly irrelevant actions. This problem can naturally be reframed as a question of causality i.e. what actions cause success in a task.

The main issue explored in this work is whether applying causal models to reinforcement learning is a promising approach to the temporal credit problem and if so how might one implement a solution. Preliminary experimental results show causal models can identify important dependence relationships between variables in simple domains. The paper is structured as follows: we discuss background material on reinforcement learning, graphical models, and statistical testing in section 2. Section 3 describes the approach used for applying graphical models in RL, we examine experiments and their results in section 4, and we end with conclusions and discussion in section 5.

# 2 Background

In this section we present brief introductions to relevant techniques from the fields of reinforcement learning, graphical models, and statistics. Each description consists of brief idea sketches for intuition along with references for further understanding. Algorithms for the techniques actually used in this project have been included in the appendix.

## 2.1 Reinforcement Learning

The purpose of reinforcement learning is to find a policy for choosing actions that maximize the agent's reward. A common description of reinforcement learning is as follows:

- A set of states $S$
- A set of actions $A$

- A reward signal $REWARD$

Each state is defined by the values the environment variables (also called state variables) take on in a given instance. The reward signal is a function, $REWARD : S \longrightarrow \mathbb{R}$, from the set of states to the reals. Our goal can be defined as learning a policy, $\pi : S \longrightarrow A$, that determines what action an agent should take given its current state. A common approach to learning such a $\pi$ is to assign values to state-action pairs based on the expected reward. So for each state we give values to each of the actions that may be taken from the state, with higher values given to actions that expect to produce a bigger reward and lower values given to actions expected to return less of a reward. We denote the state-action values as $Q(s, a)$ and store them in what is known as a $Q$-table. The policy function, $\pi(s)$, works by choosing the $a$ that maximizes $Q(s, a)$.

We use a fairly standard method of temporal-difference learning called SARSA for assigning the $Q(s, a)$ values. A description of the SARSA algorithm from [9] can be found in the appendix. More information on this method and reinforcement learning in general can be found in the excellent sources [10, 9], but will not be gone into more detail here as it is not the focus of this work.

## 2.2 Graphical Models

We use graphical models in our approach to represent the dependence relations between variable values. A common probabilistic graphical model is the Bayesian network. In a Bayesian network each node in the graph represents a different variable and the directed edges between nodes denote what needs to be factored into calculating the distribution of a variable. A simple example would be the directed graph consisting of two nodes $X$ and $Y$. If $Y$ is the parent of $X$ then the directed edge tells us the distribution of $X$ is dependent on $Y$ i.e. we have the conditional probability distribution $P(X|Y)$. The utility of the graphs comes from the Markov property between parent and child nodes. This means that the distribution of each node is independent of all its non-descendents given its parents and thus creates an abbreviated joint probability distribution over all variables. In order to attain complete information about a single variable/node, only the parent nodes need be examined as opposed to the whole graph.

The idea that a node in a probabilistic graphical model is only dependent on its parents leads us to the causal interpretation of the graph structure and orientation of edges. This interpretation allows us to apply properties of causality (e.g. transitive, asymmetric, non-reflexive) to help determine the structure of a graphical model from empirical data. The causal properties constrain the possible configurations of structure and orientation. These constraints combined with conditional independence statements about the domain are used in several algorithms to discover the causal structure between variables [7, 8, 11]. The general idea is to use pair-wise dependence tests between variables to identify neighboring nodes in the graph (if two variables are independent then they should not be connected). Once this initial structure is laid down, the constraints of causality are used to orient edges and prune the structure. We use the grow-shrink(GS) algorithm for constructing causal models in our application to reinforcement learning. A description of the algorithm from [11] can be found in the appendix and proofs of correctness and complexity analysis can be found in [11].

The constraint-based algorithms for determining graphical model structure are reliant on conditional indepenedence tests between variables. The fact that RL data comes in the form of a time-series (an agent's periodic observations of its environment) is problematic for the calculation of conditional probability. We discuss the problem and present a solution based on statistical testing in section 3.2. A brief presentation of the statistical tests used is covered in the next subsection.

## 2.3 Statistical Testing

Correlated sample tests are statistical methods for determining the significance of the difference between the means of two correlated samples. These tests are useful for examining a population before and after the occurence of some variable in order to determine whether the variable had a significant effect upon a population.

A simple illustration of this idea is answering the question of whether people are taller with their shoes on as opposed to with their shoes off [12]. The obvious answer is yes, but as we will shortly see, a simple misapplication of statistical tests can give us a very non-intuitive answer. In this example we have two samples; sample $A$ consists of the height measurements for people without shoes. Sample $B$ is the set of height measurements for the same people except with shoes. If we try and apply a standard student t-test for determining the difference between the means of the two samples, we will probably find that the difference is quite insignificant. This is because the small mean difference is washed out by the large internal variability of the samples. Since we know the samples have paired data (the measurements of the same individual from both samples) we can overcome this problem by examining the difference between the pairs. If the two samples truly have the same mean then the difference across all pairs should be zero. This fact is used to determine the significance of the difference between correlated samples.

Two tests for performing this comparision are the paired student t-test and the Wilcoxon signed rank test. The difference between these methods is the student t-test is parametric (i.e. it makes assumptions about the parameters of the population distribution) and the Wilcoxon signed rank test is non-parametric (i.e. it does not make these assumptions). The assumptions made for the student t-test are that the scale of measurement is an equal-interval scale, the differences between the paired values have been randomly drawn, and the source population of the differences can be assumed to have a normal distribtuion [12]. The Wilcoxon signed-rank test is an alternative for when these assumptions do not hold. The nice thing about both these tests is that as the sample size passes a small number (these tests are designed for small sample testing) they perform virtually the same so we can choose either test for our purposes. We use the paired student t-test in our application and the algorithm can be found in the appendix (taken from [12]). More details on these tests can be found in [12].

# 3 Applying Graphical Models to Reinforcement Learning

In this section we present a method for applying causal models to reinforcement learning. The first subsection gives the high level application of how a causal model is used during the learning process along with an algorithm for application. The second section describes the difficulties of applying standard model construction algorithms to the RL domain and presents a possible approach using statistical testing.

## 3.1 Internal Intermediate Reinforcement

We approach the problem of increasing feedback during learning by exploiting dependence relationships in the environment. The general idea is to determine what state variable values have influence on reward then use those values as conditions for applying intermediate feedback. Intuitively, we can think of this process as first finding the causes of reward and creating sub-goals around those causes (e.g. if our goal is $g$ and $c$ causes $g$ then make $c$ a sub-goal). In theory, the process can be applied recursively so that we find intermediate feedback for the sub-goals ad infinum until the problem has been completely decomposed (e.g. if $c$ is a subgoal because it causes the original goal $g$, determine what causes $c$ and make that a sub-sub-goal).

We call this *internal intermediate* reinforcement because in standard RL, reward is allocated externally by the agent's environment. The feedback we propose for achieving the sub-goals (described above) comes in between the external rewards, hence the *intermediate*, and is applied by the agent *internally* to itself. The significant difference between these two types of feedback is that the set of conditions for external reward must be defined using human knowledge of the domain whereas the conditions for the internal rewards are dependent on the agent's interaction and observation of the environment. This means the definition of conditions for internal rewards do not require additional human engineering.

The intuition behind using causal models in RL is straight-forward. Learning to complete a complicated task with sparse feedback is difficult because we are searching in a very large state space for reward state(s). By finding out what state values increase the probability of receiving reward, we learn what parts of the state space have high probability of being close to the goal state. Learning to states with these dependent values is essentially learning to move to a part of the state-space that is close to the goal state, which makes searching for goal easier.

An algorithm we shall call CMRL (for Causal Model Reinforcement Learning) is shown in Figure 1. This algorithm applies to RL a graphical model representing dependence relationships between variable values. Each node in the graph represents a variable value and the directed edges represent a dependence relation between two variable values. The general idea is for the agent to check the model upon entering a new state and see if it should apply intermediate reward to itself. The model contains information on what variable values directly or indirectly increase the probability of high reward occuring in the future (in our case high reward is 1 and low/no reward is 0). The nodes

that positively influence reward (we shall call these "influential nodes") are on directed paths in the graph that end with the high reward node. The agent applies reward to itself if it reaches a state who has a variable value equal to one of the influential nodes. The algorithm also describes maintenance operations such as recording every state visited as data for re-constructing the model later on. After the agent has determined whether reward should be applied or not, the behavior is based on the RL algorithm generating the agent's policy. In our case an action is chosen using the $Q$-table and the agent transitions to a new state starting the process anew.

## 3.2 Modifying Causal Model Construction for the RL Domain

We present the main technical difficulties of applying causal models to RL in this section. The first adaptation is graphical models represent relationships between individual variable values as opposed to entire variables. We do this because our goal is to find conditions for when to apply intermediate reward and those conditions are defined as environment variable values. The second change deals with the fact that state variable data comes as a time-series. The difficulty with time-series is that conditional probability is defined in such a way that the calculation is not meaningful when data is in a time-series. Determining conditional independence relies heavily on the idea of conditional probability and this proves to be the most challenging aspect of adapting causal models to RL.

Modeling relationships between individual variable values is a natural way to introduce intermediate reward (as described in section 3.1). These graphs are a generalization of standard models because nodes representing values from the same variable can be grouped together to create a standard model.

The constraint-based algorithms for constructing graphical models rely on conditional independence tests. The formula $X \perp Y|S$ states that variables $X$ and $Y$ are independent given the set of variables $S$. Conditional independence can also be described by the formula $P(X|S) = P(X|Y \bigwedge S)$. Empirical data from reinforcement learning naturally comes in the form of a time series (evenly spaced observations of the environment by the agent over time). This is not conducive to the standard formula of calculating conditional probability so we present an alternative method using correlated samples tests.

Dependence between two variables is normally defined in terms of conditional probability with the formula $P(X = x) = P(X = x|Y = y)$. The intuition is if the probability of $X = x$ is not affected by the occurrence of $Y = y$ then the two values are independent. Calculating $P(X = x|Y = y)$ is not effective with time series data because the event of $Y = y$ may not co-occur with $X = x$ even if $Y = y$ influences $X = x$. Imagine a light bulb being turned on always causes a door to open at some time in the future, but the light bulb can be turned off before the door is actually open. Under these conditions it is quite conceivable that $P(door = open|light = on) = P(door = open)$ despite the fact $door = open$ and $light = on$ are dependent.

We approach this problem by examining the intuitive notion of dependence. If we are trying to determine whether two variable values $X = x$ and $Y = y$ are dependent, we would like to see if the occurence of one value changes the behavior of the other value. Behavior in this case is defined by frequency so we

*CMRL algorithm*

Let $\pi : States \rightarrow Actions$ be a policy for choosing an action given the current state.
Let $V = \{V_1 = v_{1_1}, V_1 = v_{2_1} \ldots, V_q = v_{n_q}\}$ be the set of all state variable value pairs where there are $q$ different variables and $n$ possible variable value pairs. Note each $v_i$ represents a unique variable value pair.
Let $G = V, E$ be a graphical model of the causal relationships in the environment, where $E$ is a set of weighted edges.
Let $I = \{< v_i, d_1 >, \ldots, < v_j, d_m >\}$ be a set of pairs used for defining intermediate reward. The pair consists of a variable value pair $v_i \in V$. The second value is an integer $d$ representing the distance from the reward node in the graph.
Let $S =< s_1, \ldots, s_T >$ be the ordered state history, where each $s = \{V_1, \ldots, V_q\}$ is a state defined by the state variables in $V$. Each $s_i \in S$ is a state that has been visited by the agent.
Let $R$ be the reward node in $G$. This node represents $reward = 1$

Initialize $I$, $S$, and $G$ to be empty.

For each episode
1. Initialize agent to be in state $s$.
2. Let $G'$ be the subset of $G$ consisting of nodes that have a direct path to $R$ that when conditioned upon increase probability of reward. $\forall v \in G'$ add $< v, d >$ to $I$, where $d$ is the distance of $v$ from $R$ in the graph.

*While goal state not reached...*
3. Add $s$ to $S$. If $s$ is the goal state end the episode. Else $\forall < v, d > \in I$ check if any of the variables defining $s$ has the value $v$. If any part of $s$ has value $v$ apply a reward inversely proportional to $d$ to the agent and remove $v$ from $I$.
4. Use action $a = \pi(s)$ to transition to a new state $s'$. Let $s = s'$.
*After the episode ends...*

5. Call the GS algorithm to re-construct $G$ using $S$.

Figure 1: CMRL algorithm.

look at how $P(X = x)$ changes before and after an event $Y = y$. In practice we measure the significance of behavior change using paired statistical tests on samples containing $P(x)$ before and after occurences of $Y = y$ collected from the data. If the sample means differ significantly then we know (with a certain level of confidence) that $x$ and $y$ are dependent. We apply this idea in algorithms for testing dependence relations and calculating conditional probabilities in time series data (see Figure 2 and Figure 3).

The algorithm presented in Figure 2 uses an agent's observations of the envrionment to determine whether two values $X = x$ and $Y = y$ are independent given the set of values $Z$. The first step is to identify each occurence of $Y = y$ in the state history. Once the states where $Y = y$ are recorded, we calculate the conditional probability $P(X = x|Z)$ in time intervals before and after each occurence. The time intervals are defined using heuristic. Suppose the occurence of $Y = y$ currently being examined is at time $t_i$. We define the before time interval as half way between $t_i$ and $t_{i-1}$ where $t_{i-1}$ is the most previous occurence of $Y = y$. We define the interval for calculating $P(X = x|Z)$ after occurence $t_i$ as $t_i$ to $t_{i+1}$. The reasoning behind this choice is if $Y = y$ does in fact influence $P(X = x|Z)$ then the amount of influence in a short time interval right before $Y = y$ will be different with respect to the time period after $Y = y$ occurs. If $Y = y$ does not influence $P(X = x|Z)$ then the time intervals should not matter and $P(X = x|Z)$ should be the same before and after $Y = y$ occurs. Each conditional probability calculation before and after a $Y = y$ occurence makes up a pair for the paired student t-test. After each pair is found for each occurence of $Y = y$, we use the test to determine the confidence level that $P(X = x|Z)$ is significantly different before and after an occurence of $Y = y$.

The method we use for calculating conditional probability over a specified time interval (as described in Figure 3) is to record the time right after all of the conditioning values have occurred at least once. We call this time *startCount*. The probability of $X = x$ is then calculated over the subinterval beginning at *startCount* and ending at the end of the specified interval. Probability is calculated by dividing the number of $X = x$ occurences by the length of the subinterval.

While this is an intuitively appealing idea for determining conditional independence relationships there is no formal backing for its correctness. Many assumptions are made such as the duration of influence a variable value has after it occurs. Defining what time interval to calculate probabilities over is a non-trivial problem and is dealt with in a heuristic manner for this approach. That being said, the experiments performed in simple domains show some success at recovering important dependence relationships in the environment.

Now that we have a method for performing conditional independence tests with time series data, we can apply the GS algorithm to an agent's observations and learn the structure of a causal model. The model can be applied to improve RL as described in section 3.1.

# 4 Experiments and Results

In this section we present preliminary experiments and results that support the idea of using causal models in reinforcement learning. Our experiments show

*Conditional Independence Test algorithm:*

Input: $X = x$, $Y = y$, $Z = \{Z_1 = z_1, \ldots, Z_n = z_n\}$, where $X$, $Y$, and $Z_i$ are state variables.
Output: Confidence value that $X = x$ is indepedent of $Y = y$ given set $Z$.

Let $S =< s_1, \ldots, s_T >$ be the ordered state history, where each $s_i$ is a state visited by the agent.
Let $S_y = \{s_{t_1}, \ldots, s_{t_p}\}$ be a set of states, $s_{t_j} \in S$, with $Y = y$ and $1 \leq t_j \leq T$.
Let $P_{before} = \{p_1, \ldots, p_k\}$ be the set of probability values $P(X = x|Z)$ before occurences of $Y = y$.
Let $P_{after} = \{p_1, \ldots, p_k\}$ be the set of probability values $P(X = x|Z)$ after occurences of $Y = y$.

1. $\forall s \in S$ check whether $y_s = y$, where $y_s$ is the value for $Y$ in state $s$. If $y_s = y$ then add $s$ to $S_y$.
2. Define $P_{t_i \to t_j}(X = x|Z)$ to be the conditional probability measured over the time interval $[t_i \ldots t_j]$. Then $\forall s_{t_j} \in S_y$ calculate $p_{before} = P_{[\frac{(t_{(j-1)}+t_j)}{2}] \to t_j}(X = x|Z)$ and $p_{after} = P_{t_j \to t_{(j+1)}}(X = x|Z)$. Add $p_{before}$ to $P_{before}$. Add $p_{after}$ to $P_{after}$. These observations make up the paired samples for the statistical tests.
3. Perform the paired student t-test (see appendix) using $P_{before}$ and $P_{after}$. Return the confidence level of that test.

Figure 2: Conditional Independence Test algorithm.

*Conditional Probability algorithm:*

Input: $t_{begin}$, $t_{end}$, $X = x$, $Z = \{Z_1 = z_1, \ldots, Z_n = z_n\}$, where $t_i$ are integers and $X$ and $Z_i$ are state variables.
Output: Probability of $X = x$ conditioned on $Z$ over the interval $t_{begin} \rightarrow t_{end}$ (i.e. $P(X = x)$ given $Z$ has occurred).

Let $S_{sub} = \{s_{t_begin}, \ldots, s_{t_end}\}$ be a subset of the state history $S$.
Let *counter* be an integer keeping track of instances of $X = x$.
Let *startCount* be an integer marking the beginning of the interval for calculating $P(X = x)$.

Initialize $counter = 0$, $startCount = t_{begin}$.

1. Check $S_{sub}$ for the first occurences of conditional values $Z$. Suppose these first occurences are at $\{s_{t_1}, \ldots, s_{t_n}\}$ where $s_{t_i}$ is the state where value $z_i$ occurred. then let $startCount = t_j$ where $t_j = max\{t_i\}$. Else return 0 because $P(X = x|Z)$ is vacuously true. This essentially means wait for all the conditioning values to occur at least once and when that happens then begin calculating the $P(X = x)$.
2. $\forall s_i \in S$; for $startCount \leq i \leq t_{end}$, check whether $x_{s_i} = x$, where $x_{s_i}$ is the value for $X$ in state $s_i$. If $x_{s_i} = x$ then $counter = counter + 1$. Here we calculate the occurences of $X = x$ in the specified time interval, which we will eventually divide by the total interval to get $P(X = x)$.
3. Return $\frac{counter}{|S_{sub}|}$. This is $P(X = x)$ given $Z$ has occurred.

Figure 3: Conditional Probability algorithm.

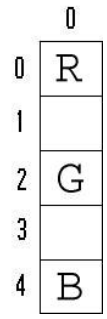


Figure 4: Simple Taxi

the modified GS algorithm can discern important dependence relationships in simple environments. The conditions for applying intermediate reinforcement found by this algorithm also show improvement in learning within the domain. We describe the domain, experimental setup, and results in the subsections.

## 4.1 Simple Taxi

The domain for testing the algorithm is a simple retrieval task set in a grid-world environment. The grid-world consists of adjacent cells/grids the agent can move amongst using the actions UP, DOWN, LEFT, or RIGHT. Taxi is an episodic task where the objective of the agent is to pickup passengers at pre-designated locations and drop them off at their destinations. We use a simplified "corridor" variation where the agent moves along a 1-by-5 strip (see Figure 4). The world contains a set of landmarks, $L = \{R, G, B\}$, at designated locations on the strip.

At the beginning of each episode a passenger appears at one of the landmarks and must be delivered to a different landmark. The agent's state, $s = \{X, Y, PL, D\}$, is defined by the variables X location ($X = \{0\}$), Y location($Y = \{0, 1, 2, 3, 4\}$), passenger location($PL = \{R, G, B, I\}$), and destination ($D = \{R, G, B\}$). Passenger location can be any of the landmarks or inside the taxi; destination can be any of the landmarks. Actions the taxi can take are defined by the set $A = \{UP, DOWN, PICKUP, PUTDOWN\}$. Two variations of the task are used in the experiments. In the first version the passenger location (PL), destination (D), and taxi starting position are the same for every episode (we refer to this set-up as "defined taxi"). The other version starts each episode with PL, D, and taxi start position randomly chosen (we refer to this set-up as "random taxi").

## 4.2 Experimental Setup

The following experiments test model construction in the simple taxi domain. The objective is to determine whether important dependence relationships are discovered using the algorithms described in section 3. We do this by observing direct and indirect dependence relations in the environment, discovered by the algorithm, in regards to select variable values. The variable values are chosen with respect to their relevance to reward.

| REWARD=1 | |
|---|---|
| number of episodes | dependent values |
| 5 | Y=0,PL=R |
| 25 | Y=0,PL=R |
| 50 | Y=0,PL=R |
| 100 | Y=0,PL=R |

Figure 5: Table 1.1

| PL=R | |
|---|---|
| number of episodes | dependent values |
| 5 | Y=1 |
| 25 | Y=1,PL=I |
| 50 | Y=0,PL=I |
| 100 | Y=0,PL=I |

Figure 6: Table 1.2

We test the algorithm in both defined taxi and random taxi. In defined taxi, the starting state for each episode is $s_0 = \{X = 0, Y = 3, PL = R, D = B\}$. The agent uses a random action-selection policy in both experiments. Each experiment is run for 5, 25, 50, and 100 episodes. The results are summarized in tables 1.1-1.4 and table 2.

### 4.3 Defined Taxi Results

We begin by observing what variable values influence reward in defined taxi. Table 1.1 shows that in the defined taxi domain reward is dependent upon $Y = 0$ and $PL = R$. These relationships are apparent after 5 episodes of experience and remain the same after 25, 50, and 100 episodes. The dependence relation of reward and $Y = 0$,$PL = R$ makes sense for this task because $Y = 0$ is the grid coordinate for $PL = R$ and the passenger location at the start is always $PL = R$. These values would be useful as intermediate reward because the taxi must pickup the passenger at $Y = 0$ in order to complete the task. Experiments using similar conditions for intermediate reinforcement (see section 4.5) suggest learning to go to $Y = 0$ will accelerate learning of the whole task.

Since reward is dependent on $PL = R$ and $Y = 0$, we would like to know what these values might depend upon in the environment. The algorithm determines $PL = R$ to have dependence relations to $Y = 1$ and $PL = I$ after a 5-25 episodes (as shown in Table 1.2), but with more information we settle on $PL = R$ having dependence relations to $PL = I$ and $Y = 0$. These relationships make sense because the passenger must get into the taxi ($PL = I$) at the landmark $R$ and the landmark $R$ is located at $Y = 0$.

The results in Table 1.3 describe the dependence relations for $Y = 0$. $Y = 0$ is found to be dependent with $PL = R$ and $PL = I$. The direct dependence with $PL = R$ and $PL = I$ exists because $Y = 0$ is the grid coordinate for $PL = R$ and the passenger must get inside at $Y = 0$.

The final variable value we examine in the defined taxi domain is $PL = I$

| Y=0 | |
|---|---|
| number of episodes | dependent values |
| 5 | PL=R |
| 25 | PL=R,PL=I |
| 50 | PL=R,PL=I |
| 100 | PL=R,PL=I |

Figure 7: Table 1.3

| PL=I | |
|---|---|
| number of episodes | dependent values |
| 5 | PL=R |
| 25 | Y=0 |
| 50 | Y=0,PL=R |
| 100 | Y=0,PL=R |

Figure 8: Table 1.4

(see Table 1.4). We find that $PL = I$ is dependent with $Y = 0$ and $PL = R$, which is to be expected from the previous tables.

## 4.4 Random Taxi Results

The results for random taxi are more difficult to interpret than in defined taxi. The dependence relations for selected variables are summarized in Table 2. Reward is found to have dependence relationships with multiple values of $Y$ and $PL = I$ over the course of episodes 5-50. The variable value reward most consistently has direct dependence with is $PL = I$. This is an important relationship because $PL = I$ is a condition for reward independent of the passenger starting position or destination. $PL = I$, as a condition for applying intermediate reinforcement, is also important because it has been shown to accelerate learning in the domain (see Figure 10). After 100 episodes the dependence relationships make less sense as reward is found to have dependence with $PL = G, PL = Y$, and $D = G$. These values are explored in more depth, but no intuitive explanation is apparent.

## 4.5 Learning in Simple Taxi

We verify the dependence relationships found in section 4.2 and 4.3 can accelerate learning by applying them as conditions for intermediate reinforcement in the simple taxi domain. The experiment compares learning with the standard SARSA algorithm and SARSA combined with intermediate reinforcement applied when $PL = I$ (i.e. when the passenger get into the car). $PL = I$ was discovered to be an important variable value when constructing the causal model in both defined and random taxi. The following results are not from a complete integration of causal models and reinforcement learning, but they do demonstrate relationships discovered by the construction of a causal model can benefit learning.

| REWARD=1 | |
|---|---|
| number of episodes | dependent values |
| 5 | Y=0,PL=I |
| 25 | Y=1,Y=3,Y=4,PL=I |
| 50 | Y=1,PL=I,D=Y |
| 100 | PL=G,PL=Y,D=G |
| PL=G | |
| 100 | PL=Y |
| PL=Y | |
| 100 | Y=4,PL=G |

Figure 9: Table 2

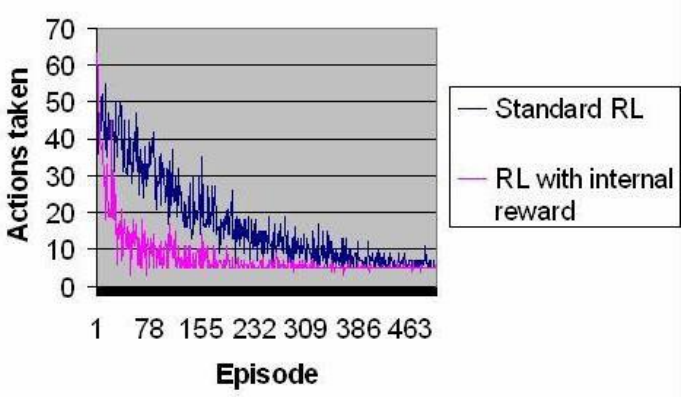


Figure 10: Graph of learning in Simple Taxi with and without intermediate reinforcement given when PL=I.

Results are summarized in Figure 10 where each point is the median of running the simulation 20 times for 500 episodes each. Each episode starts with the taxi at a set position in the world and the passenger located at a random special landmark with a random destination. The domain was first learned using SARSA as presented in [9] then learned such that the agent would apply a reward to itself the first time it reached a state with $PL = I$. The results show this intermediate reinforcement accelerates the learning process in simple taxi.

# 5 Conclusion and Discussion

We have presented an approach for improving reinforcement learning by automatically increasing useful feedback. Our proposal is to exploit the causal structure of an environment; more specifically we use environment values that increase the probability of reward as conditions for applying intermediate reinforcement. This method minimizes the amount of human engineering required because causal structure is determined through an agent's observation of its

environment. We presented modifications to established structure-learning algorithms in order to apply them in the RL domain. The general idea is to reformulate conditional independence as a comparison (using paired statistical tests) of a variable before and after the occurence of a conditioning value. This method inherently has a heuristic nature to it, but experiments in simple domains showed dependence relations that improve learning can be automatically discovered.

While the particular approach presented in this paper is probably limited by the weaknesses of the method for determining conditional independence in a time series, the main contribution of this work is a demonstration that using causal models may be a good approach to the reinforcement learning problem. Theoretically, a non-action-value RL method could be devised by learning a causal graph that represents the environment in its totality (including the action model). If complete understanding of what causes what were known, then a policy could be generated by recursively sub-goaling until only atomic actions are needed for the lowest level goals' completion. This is still a lofty endeavor as the machinery for causal model construction requires much more development before the idea can be feasible. Possibilities for more immediate future work would be to apply already established methods for graphical model construction from time-series data [13] and seeing how well they aid learning. Making the models for continuous domains and applying them to benchmarks such as keepaway [14] is also an important task. The application of causal models in reinforcement learning is a rich field with potential for future successes.

# 6 Acknowledgements

This project was supervised by Peter Stone and funded by the University of Texas Mathematics Department VIGRE program. Risto Miikkulainen and Calvin Lin reviewed the work and served on the undergraduate thesis committee. Additional statistical code was provided by Nick Jong and the learning agent framework code was provided through [14]. Many thanks go to all those involved.

# A SARSA

The SARSA algorithm from [9].

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize $s$
  Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
  Repeat (for each step of episode):
    Take action $a$, observe $r$, $s'$
    Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma Q(s', a') - Q(s, a)\big]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until $s$ is terminal

# B Paired Student T-Test

The paired student t-test from [12].

**Step 1.** Go forward one page ple of N values of $D_i$, where each instance of $D_i$ is equal to $X_{Ai}-X_{Bi}$, calculate the mean of the sample as

$$M_D = \frac{\Sigma D_i}{N}$$

and the sum of squared deviates as

$$SS_D = \Sigma D^2_i - \frac{(\Sigma D_i)^2}{N}$$

---

**Step 2.** Estimate the variance of the source population as

$$\{s^2\} = \frac{SS_D}{N-1}$$

---

**Step 3.** Estimate the standard deviation of the sampling distribution of $M_D$ as

$$\text{est. } \sigma_{M_D} = \text{sqrt}\left[\frac{\{s^2\}}{N}\right]$$

---

Note that Steps 2 and 3 can be combined into the more streamlined formula

$$\text{est. } \sigma_{M_D} = \text{sqrt}\left[\frac{SS_D/(N-1)}{N}\right]$$

---

**Step 4.** Calculate **t** as

$$t = \frac{M_D}{\text{est. } \sigma_{M_D}}$$

---

**Step 5.** Refer the calculated value of **t** to the table of critical values of **t** (Appendix C), with **df**=N−1. Keep in mind that a one-tailed directional test can be applied only if a specific directional hypothesis has been stipulated in advance; otherwise it must be a non-directional two-tailed test.

# C Grow-Shrink(GS) Algorithm

The Grow-Shrink algorithm from [11].

1. **[ Compute Markov Blankets ]**
   For all $X \in \mathcal{U}$, compute the Markov blanket $\mathbf{B}(X)$.
2. **[ Compute Graph Structure ]**
   For all $X \in \mathcal{U}$ and $Y \in \mathbf{B}(X)$, determine $Y$ to be a direct neighbor of $X$ if $X$ and $Y$ are dependent given $\mathbf{S}$ for all $\mathbf{S} \subseteq \mathbf{T}$, where $\mathbf{T}$ is the smaller of $\mathbf{B}(X) - \{Y\}$ and $\mathbf{B}(Y) - \{X\}$.
3. **[ Orient Edges ]**
   For all $X \in \mathcal{U}$ and $Y \in \mathbf{N}(X)$, orient $Y \rightarrow X$ if there exists a variable $Z \in \mathbf{N}(X) - \mathbf{N}(Y) - \{Y\}$ such that $Y$ and $Z$ are dependent given $\mathbf{S} \cup \{X\}$ for all $\mathbf{S} \subseteq \mathbf{T}$, where $\mathbf{T}$ is the smaller of $\mathbf{B}(Y) - \{X, Z\}$ and $\mathbf{B}(Z) - \{X, Y\}$.
4. **[ Remove Cycles ]**
   Do the following while there exist cycles in the graph:
   - Compute the set of edges $\mathbf{C} = \{X \rightarrow Y$ such that $X \rightarrow Y$ is part of a cycle$\}$.
   - Remove from the current graph the edge in $\mathbf{C}$ that is part of the greatest number of cycles, and put it in $\mathbf{R}$.
5. **[ Reverse Edges ]**
   Insert each edge from $\mathbf{R}$ in the graph in reverse order of removal in Step 4, reversed.
6. **[ Propagate Directions ]**
   For all $X \in \mathcal{U}$ and $Y \in \mathbf{N}(X)$ such that neither $Y \rightarrow X$ nor $X \rightarrow Y$, execute the following rule until it no longer applies: If there exists a directed path from $X$ to $Y$, orient $X \rightarrow Y$.

The algorithm for finding the Markov Blanket [11].

1. $\mathbf{S} \leftarrow \emptyset$.
2. While $\exists Y \in \mathcal{U} - \{X\}$ such that $Y \not\perp X \mid \mathbf{S}$, do $\mathbf{S} \leftarrow \mathbf{S} \cup \{Y\}$. **[Growing phase]**
3. While $\exists Y \in \mathbf{S}$ such that $Y \perp X \mid \mathbf{S} - \{Y\}$, do $\mathbf{S} \leftarrow \mathbf{S} - \{Y\}$. **[Shrinking phase]**
4. $\mathbf{B}(X) \leftarrow \mathbf{S}$.

# References


[1] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang, Inverted autonomous helicopter flight via reinforcement learning, in: International Symposium on Experimental Robotics, 2004.

[2] G. Tesauro, TD-gammon, a self-teaching backgammon program, achieves master-level play, Neural Computation 6 (2) (1994) 215–219.

[3] M. J. Mataric, Reward functions for accelerated learning, in: Proceedings of the Eleventh International Conference on Machine Learning, Cambridge, Massachusetts, 1994, pp. 181–189.

[4] T. Dietterich, Hierarchical reinforcement learning with the MAXQ value function decomposition, Journal of Artificial Intelligence Research 13 (2000) 227–303.

[5] A. McGovern, A. G. Barto, Accelerating reinforcement learning through the discovery of useful subgoals, in: Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2001, 2001.

[6] M. I. Jordan, Graphical models, Statistical Science (Special Issue on Bayesian Statistics) 19 (2004) 140–155.

[7] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction, and Search, 2nd Edition, The MIT Press, 2000.

[8] J. Pearl, Causality: Models, Reasoning, and Inference, Cambridge University Press, Cambridge, UK, 2000.

[9] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998, http://www-anw.cs.umass.edu/~rich/book/the-book.html.
URL http://www-anw.cs.umass.edu/~rich/book/the-book.html

[10] L. Kaelbling, M. Littman, A. Moore, Reinforcement learning: A survey, Journal of Artificial Intelligence 4 (1996) 237–285.

[11] D. Margaritis, Learning bayesian network model structure from data, Ph.D. thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, available as Technical Report CMU-CS-03-153 (May 2003).

[12] R. Lowry, Concepts Applications of Inferential Statistics, 1998, http://faculty.vassar.edu/lowry/webtext.html.
URL http://faculty.vassar.edu/lowry/webtext.html

[13] F. R. Bach, M. I. Jordan, Learning graphical models for stationary time series, IEEE Trans. Signal Process. 52 (8) (2004) 2189–2199.
URL http://www.cs.berkeley.edu/ jordan/papers/650.ps.gz

[14] P. Stone, R. S. Sutton, G. Kuhlmann, Reinforcement learning for RoboCup-soccer keepaway, Adaptive BehaviorTo appear.