ANALYZING ABSTRACTION AND APPROXIMATION WITHIN MDP/POMDP ENVIRONMENT

ILYA LEVNER AND MAGDALENA JANKOWSKA

May 6, 2002

ABSTRACT. Markov Decision Process (MDP) has been used as a theoretical framework to solve AI problems for many decades. However, thus far most of the results cannot be effectively applied to most real world domains, which have large state spaces, indeterminant (fuzzy) goal states, incorrect guiding functions, and in general are littered with occlusions and inaccuracies. This paper explores abstractions, approximations that occur in most real world domains and their effects on the performance of the agent within such an environment.

CONTENTS

1. INTRODUCTION

A Markov Decision Process (MDP) characterizes an agent within a system. An MDP process is one where the actions taken by an agent depend solely on the current state of the system and not how that state was achieved. While MDP has been studied for many decades, the theoretical results obtained thus far are only applicable to trivial problems within trivial domains. Most real world domains have huge state spaces, which may also be only partially observable or identifiable. The goal states, which the agent is trying to reach, may not be well defined, and a further myriad of inaccuracies generally prevent the construction of optimal policies based on classical methods. The aim of this paper is to study the abstractions and approximations within Markov Decision Processes (MDP's) and Partially Observable Markov Decision Processes (POMDP's). In particular the paper looks at how inaccuracies found in value functions and state transition functions, as well as inaccuracies due to state abstraction effect the agents performance. Furthermore, the research addresses the topic of lookahead within MDP/POMDP environments as a technique to overcome the various perturbations outlined above. As motivation we describe the Forestry Information Management System (FIMS) next, a real-world domain exhibiting all of the aforementioned abstraction and approximation problems. The rest of the paper is organized as follows. The FIMS overview is succeeded by the formalization of a MDP's/POMDP's followed by a historical overview of the field (2). We then present the Maze Domain and the empirical studies carried out within(section 4). We conclude the paper with a brief summary (section 5) of analytical and experimental results presented.

1.1. **Motivation - Forestry Information Management System.**

Canadian forests span an estimated 344 million hectares, and need to be inventoried on a 10-20 year cycle. Remote-sensing based approaches appear to be the only feasible solution to inventorizing the estimated 10^{11} trees. Despite numerous previous attempts [Gougeon 1993, Larsen et al. 1997, Pollock 1994], no robust forest mapping system exists to date. FIMS is yet another attempt at developing such a system using a set of sophisticated computer vision operators and a control policy guiding their application by taking into account feedback from the environment (Figure 4). In each cycle the control policy chooses among applicable operators by *selectively* envisioning their effects several plies ahead. It evaluates the states forecasted using its *heuristic value function*, $\tilde{V}^*$ and selects the operator leading to the most promising state [Bulitko et al 2002, Madani et al 2002].

In FIMS, a *sequence* of operators is needed to go from raw images to a 3D interpretation and they have to be selected *dynamically* based on the state observed. Therefore, FIMS can be thought of as an extension of the classical Markov Decision Process (MDP) [Sutton et al. 2000] framework along the following directions: (i) feature functions for state space reduction, (ii) machine learning for the value function and domain model, and (iii) explicit accuracy/efficiency tradeoff for performance evaluation.

Ill-defined goal states: In FIMS, a goal system state contains a geo-referenced 3D forest map with the minimum possible amount of error. Unfortunately, such a state is impossible to recognize since the sought forest maps are typically unknown. The

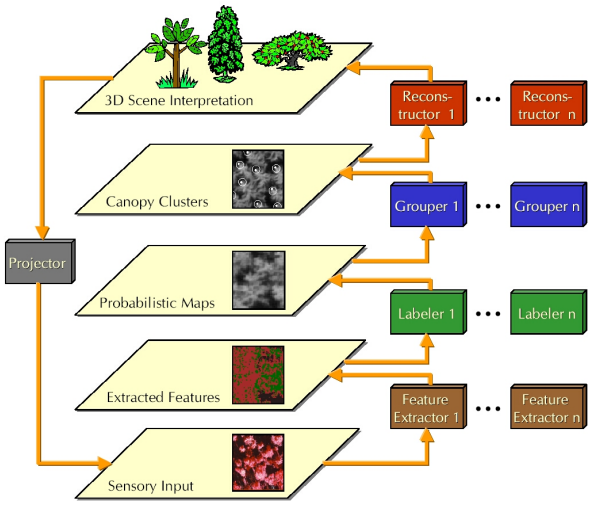

FIGURE 1. Operator Layers within Forestry Information Management System (FIMS).

lack of recognizable goal states means that we cannot use optimal goal-searching algorithms such as A*, IDA*, RBFS, etc. [Korf 1990].

Partial state observability: Raw states are often enormous in size – FIMS requires on the order of 10^7 bytes to describe a *single* problem-solving state. Thus, the raw state space is infeasible to handle. A common remedy is to employ a *state abstraction function* $\mathcal{F}$ that maps large raw states to smaller abstracted states specified via extracted feature values. ADORE [Draper et al. 2000] uses feature functions to reduce multi-megabyte raw images to a handful of real-valued parameters such as the average image brightness or edge curvature. As a result, the control policy operates over the abstracted state space which usually casts the entire problem as a Partially Observable Markov Decision Process (POMDP) [Sutton et al. 2000]. POMDPs entail several difficulties as detailed below.

Value function inaccuracies: In order to help select an operator to apply we need a value function which maps a problem-solving state to a numeric score, thereby setting a preference relation over the set of reachable states. It is typically defined as the reward the system would get from that state on by acting optimally. If the control policy had access to the actual optimal value function V^* [Sutton et al. 2000] it would be able to act optimally in a simple greedy fashion. Unfortunately, as the true value function is usually unavailable, we use an approximation $\widetilde{V}^*$ (denoted with a tilde). Various inaccuracies in the approximate V^* often lead to sub-optimal action choices and force back-tracking [Draper et al. 2000].

Inaccurate domain models: Advanced computer vision operators can take hours to compute on large images. The exponential number of operator applications needed for the lookahead makes the actual operators unusable for envisionment. Thus, approximate versions of such operators comprising the domain model $\widetilde{\delta}$ are employed for the lookahead. Consequently, such simplified models are inaccurate

and, therefore, unable to foresee future states precisely. Sequential noisy predictions introduce errors into envisioned states and thus into $\widetilde{V}^*$ values thereby offsetting the benefits of looking ahead.

FIMS clearly exemplifies the typical problems commonly encountered in real world domains. The need for state abstraction to reduce the state space, the existence of value functions inaccuracies, and errors in the state transition function disable the applicability of classical methods. Before reviewing the classical MPD methods and their shortcomings we formally define the Markov Decision Process framework used in this paper.

2. MDP/POMDP OVERVIEW

Our formulation of the Markov Decision Process is based on the work of [Boutilier et al.99]. Over the years MDP has been used for problem formulation in many fields of Science. This paper focuses on two closely related streams which use the MDP formulation extensively, namely Decision-Theoretic Planning and Reinforcement Learning. In these and all other fields where MDP is employed the task is conceptually very simple.

> *Given a description of the current state of some system, a set of actions that can be performed on the system and a description of a goal set of states for the system, find a sequence of actions that can be performed to transform the system into one of the goal states.* [Blythe99]

This section will formalize the above notion in greater detail.

2.1. **MDP Framework.**

An Markov Decision Process is typically characterized by a 4-tuple (S, A, T, R). Where

- $S = s_1, ..., s_n$**:** is a (finite) set of states the system (environment+agent) can be in. Typically a Markovian assumption is made about states, implying that information about the current state is all that is necessary for the agent to make optimal action choices. Strictly speaking, the Markov property holds if state transition probabilities depend only on the current state and action taken. Thus the agent need not concern with the history of states and actions but can compute an optimal action just from the current state information (percept) alone.
- $A = a_1, ..., a_m$**:** is a (finite) set of actions available to the agent. In general actions are stochastic, meaning that the execution of an action in the same state may not produce the same results over two trials.
- $T : S \times A \mapsto S$**:** is the state transition function. In this paper we represent the state transition function as $\delta(s, a, s') = P(s'|s, a)$, meaning that with some probability the agent can end up is state s' from state s by taking action a.
- $R : S \times A \mapsto \mathbb{R}$**:** is the reward an agent gets for being in state s and taking action a.

The agent's goal then, is to maximize rewards over some, possibly infinite, time interval (*horizon*). This paper deals with *finite-horizon* problems where the agent has a limited number of actions to maximize its reward. To evaluate the agents performance we simply add up all the rewards received and subtract all the costs incurred. Formally we define a value function for the agent's history h as:

$$V(h) = \sum_{i=1}^{T-1} R(s_i, a_i) + R(s_T) \tag{1}$$

[Bellman57], [Boutilier et al.99]. Where the agent prefers history h' over h if $V(h) \leq V(h')$.

Reworded, to maximize reward, the agent must construct a plan or policy (π) consisting of a sequence of actions that will yield maximum V(h) if that sequence of actions was to be carried out. In theory if the agent has full knowledge of (S, A, T, R) then an optimal value function $V^*(s_i)$ can be computed, in turn producing an optimal policy, π^*.

2.2. **POMDP Extension.** Partial observability occurs when the agent is unable to determine the exact system state. If absolutely no state related information is ever given to the agent then the system is characterized as a Non-Observable MDP (NOMDP). However, in general some kind of observation *is* provided to the agent, which may give some notion of its current state. This is the case of a partially observable Markov decision process, characterized by a 5-tuple (S, A, T, R, O), where O is a finite set of observations. We use $P(s|o)$ to denote the probability of being in state s given observation o was sensed by the agent. If the agent does not know its current state the Markovian property no longer holds. I.e. state transitions do not depend only on the current observation and action. As a result the agent may be forced to keep the whole history of percepts (observations) to make optimal planning choices.

3. Previous work and Research Issues

3.1. **Previous Work.**

3.1.1. *Regarding Lookahead.* The limited lookahead (real-time search) has been often used to improve the performance of a system. The examples include such successful systems as the chess program Deep Blue [Hsu et al. 1995], the checker program Chinook [Schaeffer et al. 1992] and the backgammon program TD-Gammon [Tesuaro95]. Combined with updating a heuristic function, real-time search algorithms can become learning algorithms that improve their performance over several trials (e.g Learning Real Time A* [Korf 1990]). The task of robot navigation is an example of a domain where learning algorithms related to real-time search [Koening et al. 01, Koening01] have been successfully applied. The real-time search has been also used as a tool for solving POMDP problems [Geffner et al.98].

3.1.2. *General Uses of State Abstraction.* The state abstraction within MDP problems has been studied with respect to overcoming the curse of dimensionality. The research in this field is especially related to domains that exhibit considerable structure leading to a compact representation. Adaptive abstraction schemes have been studied with relation to value iteration and policy iteration and different methods of performing state aggregation that change with iterations have been proposed by [Boutilier97, Hoey et al. 1999]. Another approach has been presented in [Boutilier et al.97, Boutilier et al.99], also related

to domains with compact factored representations (where states are not explicitly enumerated). A single fixed abstraction is produced by deleting variables, from the problem description, that are deemed irrelevant. The method of choosing irrelevant variables leads to an abstract MDP that can be solved by standard/classical methods. Once again, state aggregation was also applied to solve MDP problems for robot navigation [Laroche et al.98].

Research has also been done on applying reinforcement learning methods (such as Q-learning) directly to non-Markovian decision problems resulting from state aggregation [Singh et al.94, Jiang et al.98]. The general challenges include possibilities that learning algorithms can fail to converge on a stable policy and the fact that the optimal policies for a non-Markovian problem can be stochastic or even worse non-stationary.

3.2. **Analysis of Inaccuracies within MDP/POMDP domains.** This section presents analysis of three kinds of inaccuracies commonly found in MDP models, namely imperfect value functions, state abstraction and errors in a transition model. We focus on questions regarding Markovian property and the possible use of lookahead.

3.2.1. *Value function inaccuracies.* First, we consider a case restricted to inaccuracies related only to the value function used by the agent. Consider a value function for each state: $\widetilde{V}^*(s)$ that is sub-optimal (I.e. $\widetilde{V}^*(s) \neq V^*(s)$). Obviously, regardless of the value function used, the system is still Markovian in nature.

Now lets turn our attention to a particular inaccuracy due to state abstraction. Meaning that state abstraction is considered only with respect to the value function. The agent knows perfectly its state in the original MDP state space, but the value of each state is affected by state aggregation. Every state within one abstracted tile has the same value, which is $V^*(s)$ averaged over all original states s in a given abstracted tile. The purpose of the defined fixed tiling $\widetilde{V}^*_{\mathtt{FT}}$ function is to provide a way to analyze a small aspect of state abstraction (which here affects only the value function).

Of course, if every abstracted state groups together only states with equal V^* values, there is no inaccuracy present in $\widetilde{V}^*$ and no lookahead is needed (exact abstraction is exhibited where no information loss exists due to the abstraction process). But in the case of non-exact abstraction process the $\widetilde{V}^*_{\mathtt{FT}}$ becomes inadmissible, meaning that real value of states can be underestimated due to averaging of V^* values for states that were better than the derived average together with states that are worse than the average. Admisibility is an important property because a heuristic function used in a local search must be admissible [Koening01]. It is especially important if we would like to combine lookahead with learning. There is a family of algorithms that incorporate learning into local (real-time) search (as Learning Real Time A* [Korf 1990] for deterministic transition models or Real Time Dynamic Programming for stochastic transition models [Barto et al.95]. These algorithms update the heuristic functions while performing a search. If a heuristic function *is* admissible, LRTA* or RTDP converge to the optimal solution.

Another important property of $\widetilde{V}^*_{\mathtt{FT}}$ is related to the fact that all states inside one abstracted state have the same value. If a state aggregation has a property that a longer sequence of actions is more likely to lead outside an abstracted state, a deeper lookahead

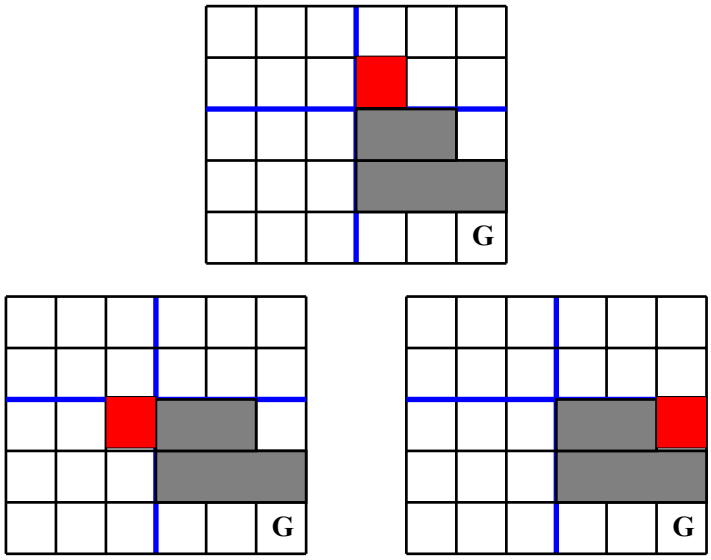

FIGURE 2. **Top Center:** A maze domain partitioned into four state abstraction tiles (purple line). The agents starting position is the red cell and the goal position is marked with a "G". Walls are shown as grayed out cells. **Bottom Left:** The final position of the agent using a lookahead depth of $p = 1$. **Bottom Right:**The final position of the agent using a lookahead depth of $p = 3$.

plays an important beneficial role. By considering only one action or a short sequence of actions the agent can observe that all reachable states have the same values. Thus in such a case deeper lookahead increases the probability of the agent "seeing" outside the current state abstraction tile and seems to be quite beneficial. Therefore, lookahead provides the agent with some guidance, not a perfect one, but very likely a better one then a random policy. A simple example to work with is a metric state space domain, where states are clustered together if they are close to each other and actions correspond to short steps in the state space. But even with this simple assumption, is deeper lookahead *always* beneficial? The answer is most certainly not. Figure 2, illustrates an example of adverse effects of a deeper lookahead in a simple maze domain. There are four deterministic actions: up, down, left, right, available to the agent. The immediate reward for each state is -1 with the goal state being the exception, where the reward is 10. The agent has perfect knowledge about its position within the maze but the value function is the same for all states in each abstracted state (or tile), as previously defined. Lets assume that the agent starts in the red square of the top grid in Figure 2. A *greedy* policy considers only a lookahead of 1 ply deep. Choosing an action under such policy leads the agent to the state denoted by the red square in the Bottom Left grid of figure 2 (which is a local maximum provided depth 1 lookahead). But if the agent considers sequences of 3 actions (a lookahead of 3 plies deep) it ends up in the state denoted by a red square in the bottom right grid of figure 2. In this domain this is the worst state. Thus deeper lookahead results in

a policy leading the agent away from the goal rather than bring in closer. This example illustrates adverse impact abstracting together states with very different V^* values.

3.2.2. *Full Effects of State Aggregation.* The state space S of the original MDP can be partitioned into abstracted states $X_1, X_2, ..., X_M$, $1 \leq M \leq |S|$ such that every state $s \in S$ belongs to one and only one abstracted state X_i.

Our first question is as follows: Is the decision problem based on the abstracted states still Markovian? The answer in this case is no. The Markov property holds if the output of an action does not depend on previous actions and visited states (history), but only on the current state. In our definition of MDP it is expressed by the fact that the state transition function δ depends only on the current state, the future state and the current action. Similarly the reward function depends only on current state and action.

Provided that the agent is in state s belonging to an arbitrary abstraction tile X_i, the probability of reaching an abstracted state X_j by taking an action a is

$$P\left(X_j|s,a\right) = \sum_{s' \in X_j} \delta\left(s,a,s'\right) \tag{2}$$

Given some probability distribution $p\left(s\right)$ of being in a state s over all states from an abstracted state X_i we have for the probability of reaching the abstracted state X_j:

$$P\left(X_j|X_i,p\left(\cdot\right),a\right) = \sum_{s \in X_i} p\left(s\right) P\left(X_j|s,a\right) = \sum_{s \in X_i} \sum_{s' \in X_j} p\left(s\right) \delta\left(s,a,s'\right) \tag{3}$$

Similarly the probability of getting a reward R' by taking an action a in an abstracted state X_i given the probability distribution $p\left(s\right)$ of being in a state s over states inside X_i is:

$$P\left(R'|X_i,p\left(\cdot\right),a\right) = \sum_{s \in X_i} p\left(s\right) P\left(R'|s,a\right) \tag{4}$$

In general the probability distribution $p\left(s\right)$ over underlying MDP states inside one abstracted state depends on the previous actions and previously visited abstracted states and that means that the Markov property is not satisfied:

$$P\left(X_{t+1} = X_i|X_t,a_t,X_{t-1},a_{t-1},X_{t-2},a_{t-2},....\right) \neq P\left(X_{t+1} = X|X_t,a_t\right) \tag{5}$$

$$P\left(R_{t+1} = R'|X_t,a_t,X_{t-1},a_{t-1},X_{t-2},a_{t-2},....\right) \neq P\left(R_{t+1} = R'|X_t,a_t\right) \tag{6}$$

where X_t, a_t denotes abstracted states and actions at time t.

Thus in general, a decision problem based on state abstraction is *not* Markovian. It is a special case of POMDP, defined by a special set of observations, O and the probability distribution of making an observation $o \in O$ given the agent is in state s after taking an action a: $P\left(o|a,s\right)$. Namely abstracted states defines a POMDP where:
$O = X_1, X_2, ..., X_M$ is the set of observations
and

$$P(X_i|a,s) = P(X_i|s) = \begin{cases} 1 & if \quad s \in X_i \\ 0 & if \quad s \notin X_i \end{cases}$$

is the probability of making an observation X_i, and for any $i \neq j$ $X_i \cap X_j = \emptyset$, for any $s \in S$ there is $i \leq M$ such that $s \in X_i$.

Before we consider ways to restore Markov property, let us present our analysis of a special case in which the Markov property for a problem based on abstracted states is satisfied.

Markov property is related to immediate rewards and state transitions. First we consider the rewards. If every abstracted state X_i groups underlying states that are indistinguishable with respect to immediate rewards, then the problem based on such abstracted states is still Markovian. This assumption means that for every action a and every abstracted state X_i, if $s_1, s_2 \in X_i$ then for every R', $P(R'|s_1,a) = P(R'|s_1,a)$. Therefore, if the reward function R is deterministic then for every action a and every abstracted state X_i, if $s_1, s_2 \in X_i$ then $R(s_1,a) = R(s_2,a)$. This condition is satisfied for example, if the reward function R can be partitioned into a function that depends only on states (and this function is identical for every state in one abstracted state) and a cost of actions that does not depend on states. We found that the dynamics of the system is Markovian given a weaker condition than the indistinguishable states with respect to the state transition. Consider for any action a and any pair of abstracted states X_i, X_j

$$\text{if } s_1, s_2 \in X_i \text{ then } \sum_{s' \in X'_j} \delta(s_1, a, s') = \sum_{s' \in X_j} \delta(s_2, a, s') \tag{7}$$

.

This condition means that the output of an action taken in two different states s_1, s_2 belonging to one abstracted state X_i may be different, but the sum of probabilities of ending up in a state s' over all states of each abstracted state is the same. If we define for such a case $\delta(X_i, a, X_j)$ as a value of the above sum, that is $\delta(X_i, a, X_j) \equiv \sum_{s' \in X_j} \delta(s, a, s')$ for an arbitrary $s \in X$, then the expression 3 has the form

$$P(X_j|X_i, p(\cdot), a) = \delta(\mathrm{X}_i, a, X_j) \sum_{s \in X_i} p(s) = \delta(\mathrm{X}_i, a, X_j) \tag{8}$$

Thus the above probability does not depends on $p(s)$ and we have an history-independent transition between abstracted states.

Such a special case is important even if rewards are not Markovian for abstracted states. Expression 8 means that the output of an action in terms of abstracted states depends only on the current abstracted state. In such a case if we have some heuristic function defined on abstracted states (for example $\widetilde{V}^*_{\mathrm{FT}}$), we can use the transition $\delta(X, a, X')$ from one abstracted state to another in a heuristic search and we do not need any estimation of the agent's position in the underlying MDP space.

An example of creating an aggregation that satisfies the condition described by equation 7 is presented in [Boutilier et al.97].

Next we consider ways of restoring the Markov property in general case of state abstraction. If the underlying MDP environment is known the most common approach

given a POMDP problem, is to take into account the whole history of the agent by calculating a belief state, i.e. the probability distribution over all underlying states. We can rewrite the general formulas for updating a belief state in our specific case of POMDP created by state aggregation as follows:

Let $b(s)$ be the current probability distribution over individual states. Notice that given current abstracted state X_i, a nonzero probability $b(s)$ exists only if $s \in X_i$. Then the prediction of a belief state given an action a is

$$b_a^{'}(s') = \sum_{s \in X_i} b(s)\, \delta(s, a, s') \tag{9}$$

And the new belief state resulting from observing a new abstracted state X_j is given by

$$b'(s) = \begin{cases} \frac{b_a^{'}(s)}{\sum_{s' \in X_j} b_a^{'}(s')} & if \quad s \in X_j \\ 0 \qquad if \quad s \notin X_j \end{cases} \tag{10}$$

So now we can apply actions to belief states and predict output of actions in terms of new belief states. In this way we transform the partially observable problem to fully observable MDP, where the states are probability distributions and immediate rewards are the average rewards for taking an action:

$$R(b(\cdot), a) = \sum_{s} b(s)\, R(s, a) \tag{11}$$

Given this fully observable belief state MDP, one approach is to use exact algorithms for solving MDP's. Of course, this approach is often intractable because belief MDP's involve state spaces that are infinite and continuous (leaving a bit of room for future research).

Another approach is to use the $V^*_{MDP}(s)$ (or $Q_{MDP}(s, a)$) of underlying MDP states as a heuristic[Litman et al. 1995]: the heuristic value for a belief state $b(\cdot)$ is the average value of states: $\widetilde{V}_{MDP}(b) = \sum_{s \in S} b(s)\, V^*_{MDP}(s)$. The problem with such a heuristic is that it assumes that all uncertainties will vanish in the future and thus the policy based on it will never perform information-gathering actions.

One way to, at least partially, overcome the aforementioned difficulties is to use a local search (lookahead) in the belief state space combined with learning. RTDP-Bel [Geffner et al.98] is an algorithm that performs local search in the belief state space using an admissible heuristic function. During the search the algorithm updates the heuristic values of belief states. In [Geffner et al.98] belief states are discretized so that the values can be stored in a table. $\widetilde{V}_{MDP}(b)$ is a good heuristic for such an algorithm because it never underestimates the value of a belief state (as it represents the value in a situation when the agent has more information than it actually has)

A belief state is an accurate summary of all the relevant past information but requires knowledge about the underlying MDP model (the model before state aggregation). This approach may be unrealistic. If we want to restore the Markov property without the access to underlying MDP environment, we have to encode the (relevant) history (e.g. by

tree structures [Kearns et al.00]). An interesting question in such a case is how long is the relevant history? The answer is domain-dependent but in general for problems based on abstracted states possibly the whole history is relevant.

3.2.3. *Errors in the State Transition Model.* The last error we consider is error in the transition function. That means that an agent has imperfect knowledge about how its own actions affect the system. We present an example that clearly demonstrates the potentially disastrous effects of error in the transition function.

Assume that no error due to value function approximation or state abstraction is present and consider the one-dimensional MDP presented at the top of figure 3. In this domain two deterministic actions, left and right, are available to the agent, leading to the next left and the next right cell, respectively (middle of figure 3). The reward for being in the goal cell (G) is 10 and for every other cell -1. But the agent uses the incorrect transition model presented on the bottom of the same figure. Notice that according to the incorrect transition function the most probable output of every action is still the desired direction of travel. Even if the agent knows the perfect V^* for each cell and has the full information about its position, if it is using this incorrect transition function, it always chooses the worst possible policy: in each cell it chooses the action 'left' instead of the correct action, 'right'.

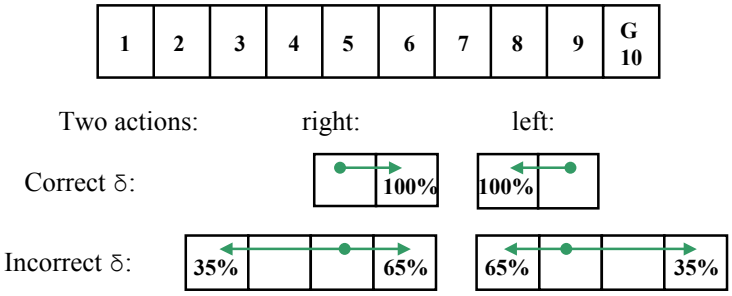

FIGURE 3. **Top:** An MDP domain with 10 cells. The number in each cell is the V^* value for that cell. The agents starting position can be any cell except goal cell, marked with a "G". **Middle:** Correct state transition function (deterministic in this case) moving the agent a distance of 1 in the desired direction with 100% certainty. **Bottom:** Incorrect state transition function moving the agent a distance of 1 in the desired direction with probability of 65% and with a 35% chance of moving the agent opposite the desired direction a distance of 2 cells. Observe that if the agent starts in cell 5, then the value of taking action 'left' is 5.05 and the value of taking action 'right' is 4.95. Thus the agent will choose the travel away from the goal, rather than towards it.

The example shows that given a perfect V^* and no state abstraction, but an incorrect transition function, even assuming the highest probability of correct action outcome, can

result the worst possible policy being adopted. This effectively demonstrates the possibility of a severe consequences of using an incorrect transition model for choosing the policy.

4. EMPIRICAL WORK

To study the effects of state abstraction and V^* perturbations the Maze Domain environment was used. The experiments were divided into three categories.

- The effects of State Abstraction alone.
- The effects of approximation errors in V^* alone.
- Grouped effects of State abstraction and V^* approximation.

We begin with a brief description of the Maze Domain followed by the experimental results.

4.1. **Experimental Setup - Maze Domain.** The Maze Domain Environment is an empirical test bed created to emulate FIMS. Based on the grid world commonly used in MDP research, the maze domain has two basic components, the maze and the agent. The agent's task is to reach a goal cell located within the maze. Briefly, the maze domain is described as follows [Bulitko et al 2002, Madani et al 2002]:

- The maze is an $N \times N$ grid of cells producing a state space $S = \{0 \dots N-1\} \times \{0 \dots N-1\}$. The agent's *state* is described by a coordinate pair (x, y) with $0 \leqslant x, y \leqslant N-1$ indicating its position within the maze. Each cell can be either empty or can contain a wall, with the agent being able to occupy only the empty cells. The maze perimeter is surrounded by walls in order to contain the agent within the maze. At the start of a batch, containing m individual games, one *empty* cell is randomly chosen to be the *goal* cell (x_g, y_g), while at the beginning of every game a random *starting* cell is chosen such that a path exists to the goal cell. An example of a maze is shown in figure 4.
- The set of actions $A = \{a_0, ..., a_{\lambda-1}, a_{quit}\}$ available to the agent consists of λ evenly spaced directional rays plus a *quit* action. Each move action a_i transports the agent along a ray projected from the agent's current location at the angle of $\frac{360 \cdot i}{\lambda}$ degrees where $0 \leq i < \lambda$. The Euclidean distance travelled is upper-bounded by τ and walls encountered (Figure 4, Right). A deterministic state transition function can be represented as $\delta_d(s, a) = s'$, while the probabilistic transition function can then defined appropriately as:

$$\delta(s, a, s') = P(s'|s, a) = \sum_{a' \in A} P(a'|a) I\{s' = \delta_d(s, a)\},$$

  where probabilities $P(a'|a)$ are based on a Normal distribution centered over action a. The execution of a quit action terminates the game.
- The MDP *immediate reward* function is defined as:

$$r(s, a, s') = \begin{cases} \mathcal{R}(s') = e^{\Theta - \|s, s_{goal}\|}, & \text{if } a = \text{'quit'}, \\ -\|s, s'\|, & \text{otherwise.} \end{cases} \tag{12}$$

  Where $\|s, s'\|$ is the Euclidean distance of the shortest *path* from s to s'.

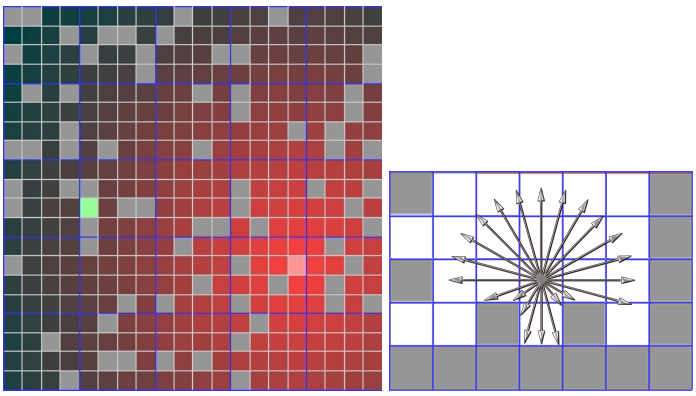

FIGURE 4. **Left:** A 20x20 maze with abstraction of $k = 4$ and density $d = 0.2$. The green cell is the starting cell and the pink cell is the goal cell. Walls are displayed in gray, while the empty cells vary in color from black to red depending on their proximity to the goal cell. **Right:** Actions an agent can take within the maze domain

4.2. **Policy Evaluation.** In order to evaluate the experiments that follow two policy error measures are defined, η_1 and η_2. The first error measure is based on the following reward per game scoring function.

$$S_g(V,p) = \mathcal{R}(s_J) - \sum_{j=1}^{J-1} \|s_j, s_{j+1}\| = e^{\Theta - \|s, s_{goal}\|} - \sum_{j=1}^{J-1} \|s_j, s_{j+1}\| \tag{13}$$

where $s_1 \rightarrow ... \rightarrow s_J$ is the sequence of states visited by the agent during a game. Using the above scoring function we define η_1 as:

$$\eta_1(V,p) = \frac{E[S_g(V^*,p)] - E[S_g(V,p)]}{E[S_g(V^*,p)]} \tag{14}$$

where $E[S_g(V^*,p)]$ is the optimal reward averaged over every possible starting position using a lookahead of p. Obviously $E[S_g(V^*,p_i)] = E[S_g(V^*,p_j)]$, $\forall i,j \in [1,\infty)$, by definition of V^*. Similarly, $E[S_g(V,p)]$ is the reward averaged over all starting cells using a sub-optimal value function. However, if $V \neq V^*$ then $\exists i,j,(i \neq j)$ s.t. $E[S_g(V,p_i)] \neq E[S_g(V,p_j)]$. Unfortunately in practice this measure becomes intractable as ply depth p increases. Therefore, we are limited to the use of η_1 for $p = 1$ to give and initial policy rankings and a global picture of policy performance. In order to evaluate error at plies deeper than $p = 1$ we define an empirical quality measure of the policy induced by a value function V as:

$$\eta_2 = \frac{\|s_{goal}, s_{quit}\|}{\max_{s \in S} \|s_{goal}, s\|} \tag{15}$$

The second measure is used for computing the decrease in policy error as lookahead is increased under a fixed $\widetilde{V}^*$ value function.

4.3. **State Abstraction.** State abstraction within the maze domain is done via a Fixed tiling scheme. The real coordinates of the agent are abstracted by:

$$\mathcal{F}_k(x, y) = \left(\left\lfloor \frac{x}{k} \right\rfloor \cdot k, \left\lfloor \frac{y}{k} \right\rfloor \cdot k \right) \tag{16}$$

where $k = \{1, ..., N\}$ is the degree of abstraction. Thus state abstraction divides the maze into non-overlapping $k \times k$ tiles. An actuality State Abstraction was never directly used in the experiments that follow. However, the presence of state abstraction can be indirectly felt within the $\widetilde{V}^*$ inaccuracies which are presented next.

4.4. $\widetilde{V}^*$ **inaccuracies.** Two different inaccuracies can be present in $\widetilde{V}^*$. One, defined as $\widetilde{V}^*_{\mathtt{FT},k}$, due to fixed tiling from state abstraction, defined by

$$\widetilde{V}^*_{\mathtt{FT},k}(\mathcal{F}_k(s)) = \underset{s_t \in \mathcal{N}_{\mathtt{FT},k}(s)}{\text{avg}} V^*(s_t). \tag{17}$$

Here $\mathcal{N}_{\mathtt{FT},k}(s)$ is the non-wall subset of the $k \times k$ fixed tile that state $s = (x, y)$ belongs to. The averaging function is defined as: $\text{avg}_A f = \frac{1}{|A|} \sum_{x \in A} f(x)$. Figure 5, demonstrates the step-function nature of the approximation.

The other inaccuracy in the $\widetilde{V}^*$ is due to machine learning. In our experiments we used artificial neural networks [Haykin 1994] to learn $\widetilde{V}^*_{\mathtt{ML},k}$ function, which introduces inaccuracies due to the ANN's inability to accurately learn the $\widetilde{V}^*_{\mathtt{ML},k}$ function. Figure 7, provides an example of a $\widetilde{V}^*_{\mathtt{ML},k=1}$ function learned by a neural network.

By combining inaccuracies from fixed tiling and machine learning we obtain a third family of $\widetilde{V}^*$ functions. Figure 9, provides an example of a $\widetilde{V}^*_{\mathtt{ML},k=6}$ function learned by a neural network.

4.5. **Experimental Results.** The experiments conducted in the maze domain consisted of 3 distinct series, with the aim of establishing the effectiveness of lookahead given various $\widetilde{V}^*$ functions. For all the experiments conducted the following parameters were kept constant: Maze Size $N = 48$; density $d = 0.2$; Number of Actions $\lambda = 24 + quit$; Maximal ray length for each non-terminal action $\tau = 2\sqrt{2}$; Batch Size consisted of 10 games, each with a random starting position.

4.5.1. *Fixed-tiling Inaccuracies.* Fixed-tiling experiments were done by varying the abstraction tile size, $k = 1, 2, 3, 4, 6, 8, 12, 24, 48$ and running a batch of games with an agent using a lookahead of $p = 1 - 7$ for each tile size.

The results are displayed in figure 6. As hypothesized prior to experiments, larger and larger fixed tiling inaccuracies can be overcome though deeper and deeper lookahead.

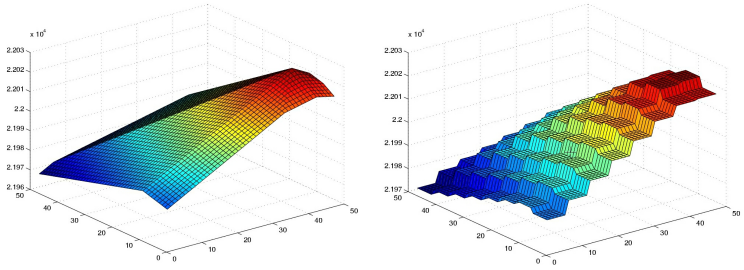

FIGURE 5. **Left:** True V^* function. **Right:** A graph of the resulting $\widetilde{V}^*_{\texttt{FT},k=6}$ function. function.

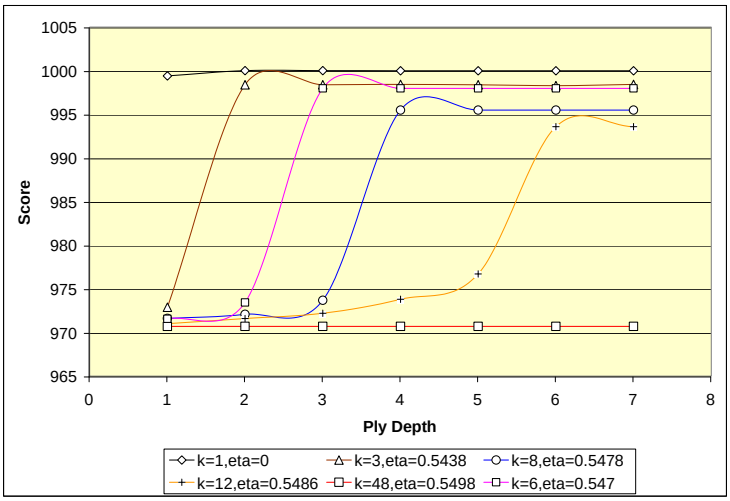

FIGURE 6. Score vs. Ply Depth for experiments using $\widetilde{V}^*_{\texttt{FT},k}$. The experiment was ran with 10 random starting points for each k. Clearly lookahead does help, with optimal ply depth (resulting in maximal reward) proportionally rising with the increase in k.

4.5.2. *Machine Learning Inaccuracies.* Various policies containing machine learning inaccuracies were created by training a $2 - h - 100$ neural network and varying the number of hidden units h from $1 - 20$. The inputs were the real x, y coordinates of cells and the target output was the true V^* value. An example of an output produced by one of the neural networks is shown in figure 7 and the experimental results are shown in figure 8. Unlike the results for fixed tiling inaccuracies, lookahead does not seem to help when inaccuracies due to machine learning are present. A possible explanation for this result is that the inaccuracies in the $\widetilde{V}^*_{\texttt{ML},k=1}$ function are not distributed throughout the whole state space but are clustered at the goal. Indeed since the Reward function grows exponentially as the agent gets closer and closer to the goal small inaccuracies close to the goal produce drastic differences in the performance of the agent captured by η_1. Thus as η_1 increases, ie. policy error rises, the agents performance gets worse and worse independent of ply

depth and starting position. The drop in performance can be clearly seen in figure 8 and is perfectly explained by the rising inaccuracies in the vicinity of the goal cell.

A question of grater importance is why this is the case. One hypothesis is that the constructed ANN module uses sigmoid activation function and as a whole also behaves like a sigmoid function. Visually in figure 7, the function learned by the ANN indeed resembles a sigmoid function. This observation, of course, is far from a proof but does warrant enough interest to prompt further empirical testing as well as analytical work.

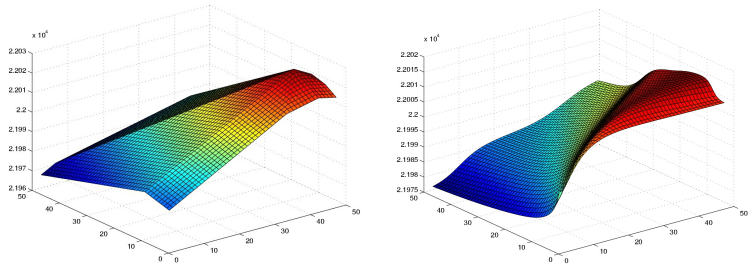

FIGURE 7. **Left:** True V^* function. **Right:** An example of the resulting $\widetilde{V}^*_{\texttt{ml},k=1}$ function learned by the neural net function.

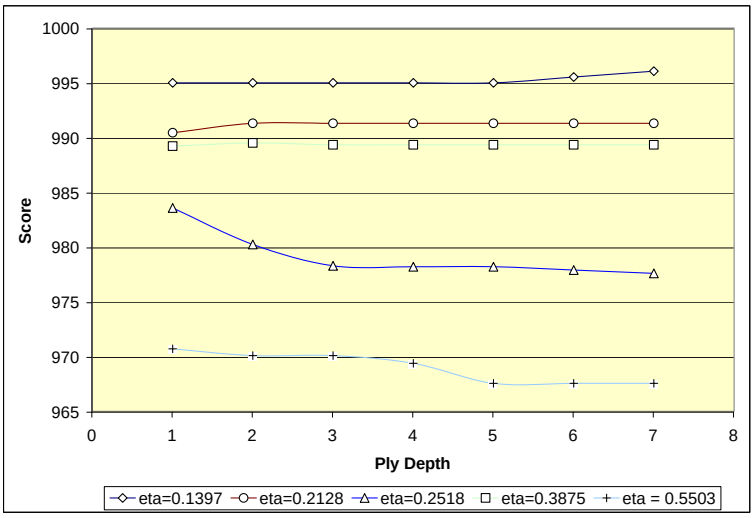

FIGURE 8. Score vs. Ply Depth for experiments using $\widetilde{V}^*_{\texttt{FT},k=1}$. Only pure machine learning inaccuracies are present since no abstraction is taking place. It is clear that lookahead does not help for pure machine learned inaccuracies.

4.5.3. *Combining Fixed tiling and Machine Learning Inaccuracies.* The Last Experiment performed combined both fixed tiling and machine learning inaccuracies together. This time the ANN was trained on the abstracted coordinates $(\widetilde{x}, \widetilde{y})$, for $k = 6$ tile size, and the output still being the real V^*. The an example of the resulting $\widetilde{V}^*_{\texttt{ML},k=6}$ function is displayed in

figure 9 and the experimental results are summarized in figure 10. The results turn out to be quite obvious in light of insights gained from the previous two experiments. Lookahead is only helpful in overcoming the inaccuracies due to fixed tiling and does not help for errors due to machine learning.

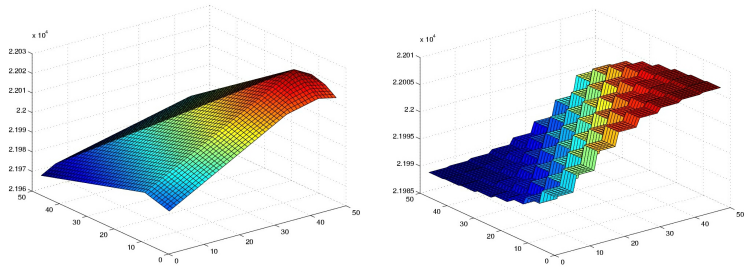

FIGURE 9. **Left:** True V^* function. **Right:** An example of the resulting $\widetilde{V}^*_{\texttt{ml},k=6}$ function learned by the neural network function.

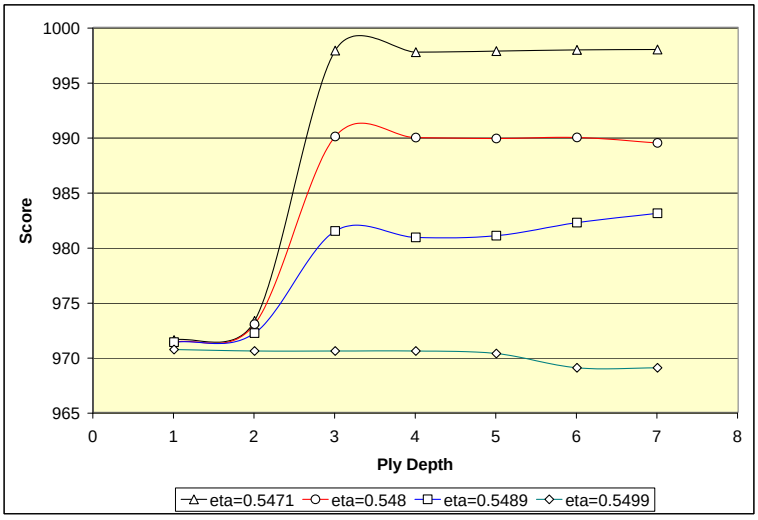

FIGURE 10. Score vs. Ply Depth for experiments using $\widetilde{V}^*_{\texttt{ML},k=6}$. Both inaccuracies due to tiling and machine learning are present. Clearly the lookahead helps in overcoming tiling inaccuracies. Referring back to Figure 6 one can see that for $k = 6$ tile size a ply depth of 3 is needed to attain optimal score, which is also the ply depth that is needed to reach optimal score when both fixed tiling inaccuracies and machine learning inaccuracies are present. However, the increase in machine learning error simply decreases the maximum score achievable regardless of the depth of lookahead.

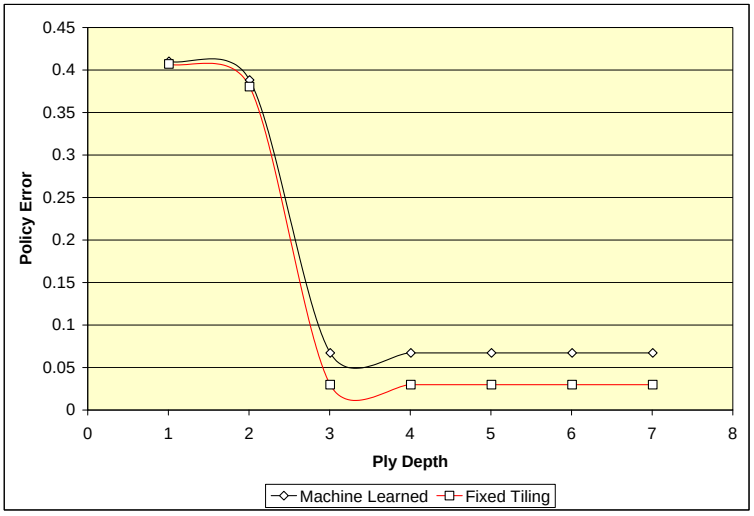

FIGURE 11. Change in the η_2 policy error with respect to lookahead ply depth. The addition of machine learning error results in the increase in policy error. However deeper lookahead is unable to overcome this added machine learned noise since the errors are localized about the goal.

5. CONCLUSIONS

This paper presented analytical and empirical results pertaining to inaccuracies typically found in MDP/POMDP domains. Clearly some inaccuracies hinder the performance of the agent to a larger extent than others. For example, we demonstrated that even small errors in the state transition model can lead to disastrous consequences for the agent. Moving along, we saw that in general state abstraction recasts an MPD problem into a POMDP, but special cases were presented where Markov property was preserved despite state abstraction. Furthermore, in the cases where the problem was transformed into a POMDP classical methods could be applied in theory but are often too weak in practice for large state spaces due to intractability issues. With regards to the value function inaccuracies, we analyzed the benefits and performance of lookahead policies in the presence of state abstraction and machine learning errors. Experimental results demonstrated cases where lookahead was extremely beneficial, cases where the use of lookahead had no effect and analytical results presented an example where deeper lookahead was actually detrimental to the agent's performance. In general we found that perturbations are problem dependant and each case requires an individual approach. No generic solutions exist to tackle all inaccuracies, however further research may prove otherwise. Regarding future research directions, we believe emphasis should be places on the inaccuracies within the value function. Of particular interest are still the questions "When does lookahead help to overcome inaccuracies in $\widetilde{V}^*$ function?" and "How can we efficiently evaluate the policies learned by an agent?".

5.1. **Acknowledgements.** Deepest thanks to Prof. Vadim Bulitko and Dr. Omid Madani, first of all for their patience and support, and for the detailed discussions and suggestions

they have provided over the lifetime of this project. Special thanks again to Prof. Vadim Bulitko for letting us use the FIMS and Maze Domain formulations and the accompanying figures.

REFERENCES

[Levner02a] I.Levner, *CMPUT 610: Robust Tracking techniques*, Final Project Paper, University of Alberta, `http://www.cs.ualberta.ca/~ilya/courses/HandEye/Project/c610_Project_paper.pdf`, 2002.

[Levner02b] I.Levner, *CMPUT 615: SHAPE DETECTION, ANALYSIS AND RECOGNITION*, Final Project Paper, University of Alberta `http://www.cs.ualberta.ca/~ilya/courses/c615_PerceptualSystems/c615_Final_paper.pdf`, 2002.

[Madani et al 2002] O.Madani, V.Bulitko, I.Levner, R. Greiner, *On Models of Control and Lookahead Search for Image Interpretation* SARA, 2002.

[Bulitko et al 2002] V.Bulitko, I.Levner, R.Greiner, *Real-time Lookahead Control Policies*, Joint Workshop on Real-Time Decision Support and Diagnosis Systems, aaai, 2002.

[Koening et al. 01] S.Koening, B.Szymanski, Y.Liu, *Efficient and inefficient ant coverage methods*, Annals of Methematics and Artificial Intelligence-Special Issue on Ant Robotics 31:41-76, 2001.

[Koening01] S. Koening. *Agent-Centered Search*, Artificial Intelligence Magazine, 22, (4), 109-131, 2001.

[Isukapalli et al. 2001] R.Isukapalli, , R.Greiner, *Efficient Interpretation Policies*, Proceedings of IJCAI'01, pp. 1381–1387, 2001.

[Shimbo et al.00] M.Shimbo, T.Ishida, *Towards Real-Time Search with Inadmissible Heuristics*, European Conference on Artificial Intelligence (ECAI-2000), IOS Press/Ohmsha, pp. 609-613, 2000.

[Kearns et al.00] M.Kearns, Y.Mansour and Andrew Y. Ng *Approximate planning in large POMDPs via reusable trajectories*, In NIPS 12, 2000.

[Sutton et al. 2000] R.S.Sutton, A.G.Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2000.

[Draper et al. 2000] Draper, B., Bins, J., Baek, K. 2000. ADORE: Adaptive Object Recognition, *Videre*, 1(4):86–99.

[Blythe99] J. Blythe, *Decision-Theoretic Planning*, AI Magazine, 1999.

[Boutilier et al.99] C.Boutilier, T.Dean, S.Hanks. *Decision-Theoretic Planning: Structural Assumptions and Computational Leverage*, Journal of Artifcial Intelligence Research 11, 1999.

[Hoey et al. 1999] J.Hoey, R.St-Aubin, A.Hu, C.Boutilier, *SPUDD:Stochastic planning using decision diagrams*, In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence Stockholm, 1999.

[Laroche et al.98] P.Laroche, F.Charpillet, *State aggregation for solving markov decision problems : An application to mobile robotics.* In 10th IEEE International Conference on Tools with Artificial Intelligence. ICTAI'98, 1998.

[Geffner et al.98] H.Geffner, B.Bonet, *Solving Large POMDPs by Real Time Dynamic Programming*, Working Notes Fall AAAI Symposium on POMDPS. 1998.

[Jiang et al.98] G.Jiang, C.Wu, G.Cybenko. *Minimax-based Reinforcement Learning with State Aggregation*, The 37th IEEE Conference on Decision and Control, pp.1236-1241, Florida, Dec.,1998.

[Boutilier et al.97] C.Boutilier, R.Dearden, *Abstraction and approximate decision theoretic planning*. Artificial Intelligence, 89, 219-283, 1997.

[Boutilier97] C.Boutilier, *Correlated action effects in decision theoretic regression*. In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, pp. 30-37 Providence, RI, 1997.

[Larsen et al. 1997] M.Larsen, M.Rudemo, *Using ray-traced templates to find individual trees in aerial photos*, Proc. 10th Scandinavian Conference on Image Analysis, volume 2, pages 1007-1014, 1997.

[Newborn 1997] M. Newborn, *Kasparov vs. Deep Blue: Computer Chess Comes Out of Age*. Springer-Verlag, 1997.

[Mitchell97] T.M.Mitchell, *Machine Learning*, McGraw-Hill Companies, Inc. 1997.

[Kaelbling96] L.P.Kaelbling, M.L.Littman, A.W.Moore. *Reinforcement Learning: A Survey*, Journal of Artifcial Intelligence Research 4, 1996.

[Hsu et al. 1995] F.H.Hsu, M.S.Campbell, A.J.J.Hoane, *Deep Blue System Overview*, Proc. 9th ACM Int. Conf. on Supercomputing, pp. 240-244, 1995.

[Russell95] S.Russell, P.Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc., 1995.

[Littman et al.95] M.Littman, A.Cassandra, L.Kaelbling *Learning Policies for Partially Observable Environments:Scalling Up*, Proceedings of the Twelfth International Conference on Machine Learning, p. 362-370, 1995.

[Tesuaro95] G.Tesauro,*Temporal Difference Learning and TD-Gammon.* Communications of the ACM, Vol. 38, No. 3, 1995.

[Barto et al.95] A.Barto, S.Bradtke, S.Singh, *Learning to act using real-time dynamic programming*, Artificial Intelligence, 72:81-138, 1995.

[Haykin 1994] S.Haykin, *Neural Networks: A Comprehensive Foundation*. Macmillian College Pub. Co, 1994.

[Pollock 1994] Pollock, R.J. 1994. A Model-based Approach to Automatically Locating Tree Crowns in High Spatial Resolution Images. *Image and Signal Processing for Remote Sensing*. Jacky Desachy, editor.

[Boutilier et al.94] C.Boutilier, R.Dearden, *Using abstractions for decision-theoretic planning with time constraings.* In Proceedings of the Twelft National Conference on Artificial Intelligence, pp. 1016-1022 Seattle, WA, 1994.

[Singh et al.94] S.Singh, P.Satinder, T.Jaakkola, M.I. Jordan *Learning Without State-Estimation in Partially Observable Markovian Decision Processes*, ICML: 284-292, 1994.

[Gougeon 1993] F.A.Gougeon, *Individual Tree Identification from High Resolution MEIS Images*, Proc. of the International Forum on Airborne Multispectral Scanning for Forestry and Mapping, Leckie, D.G., and Gillis, M.D. (editors), pp. 117-128, 1993.

[Russell et al. 1993] Russell, S., Subramanian, D., Parr, R. 1993. Provably Bounded Optimal Agents. *Proceedings of IJCAI'93*.

[Schaeffer et al. 1992] Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., Szafron, D. 1992. A World Championship Caliber Checkers Program. *Artificial Intelligence*, Volume 53, Number 2-3, pages 273-290.

[Russell et al. 1991] S.Russell, E.Wefald, *Do the right thing : studies in limited rationality*. Artificial Intelligence series, MIT Press, Cambridge, Mass, 1991.

[Korf 1990] R.E.Korf, Real-time heuristic search. *Artificial Intelligence*, Vol. 42, No. 2-3, pp. 189-211, 1990.

[Good 1971] I.J.Good, *Twenty-seven Principles of Rationality*. Foundations of Statistical Inference, Godambe V.P., Sprott, D.A. (editors), Holt, Rinehart and Winston, Toronton, 1971.

[Bellman57] R.Bellman, *Dynamic Programming* Princeton University Press, Princeton, NJ, 1957.

APPENDIX A. ON-LINE MATERIAL

A copy of this paper can be found at
`http://www.cs.ualberta.ca/~ilya/courses/c551/`
E-mail address: `ilya|magdalen@cs.ualberta.ca`