# Real-time Lookahead Control Policies

**Vadim Bulitko** and **Ilya Levner** and **Russ Greiner**
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1
CANADA
{bulitko|ilya|greiner}@cs.ualberta.ca

## Abstract

Decision-making in practical domains is usually *complex*, as a coordinated sequence of actions is needed to reach a satisfactory state, and *responsive*, as no fixed sequence works for all cases – instead we need to select actions after sensing the environment. At each step, a *lookahead control policy* chooses among feasible actions by envisioning their effects into the future and selecting the action leading to the most promising state.

There are several challenges to producing the appropriate policy. First, when each individual state description is large, the policy may instead use a low-dimensional *abstraction* of the states. Second, in some situations the quality of the final state is not given, but can only be learned from data.

Deeper lookahead typically selects actions that lead to higher-quality outcomes. Of course, as deep forecasts are computationally expensive, it is problematic when computational time is a factor. This paper makes this accuracy/efficiency tradeoff explicit, defining a system's effectiveness in terms of both the quality of the returned response, and the computational cost. We then investigate how deeply a system should search, to optimize this "type II" performance criterion.

**Keywords:** Decision-making, control policy, lookahead state tree expansion, abstraction, responsive image recognition, real-time best-first heuristic search, search horizon.

## 1 Real-time Decision-Making

We will start with a motivating practical example. Having detailed inventories of forest resources is of tremendous importance to forest industries, governments, and researchers. It would aid planning wood logging (planting and cutting), surveying against illegal activities, and ecosystem and wild-life research. Given the dynamic nature of forest evolution, the task of forest mapping is a continuous undertaking, with the objective of re-mapping the estimated 344 million hectares of Canadian forests 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.

We have been developing such a forest inventory mapping system (FIMS) using a set of sophisticated computer vision operators. FIMS problem-solving state is represented using a layered shared data structure as shown in Figure 1. Initially, aerial images and LIDAR data are deposited at the sensory input layer. As the computer vision operators are applied, higher layers become populated with extracted data. Finally, the top layer gains the derived 3D interpretation of the forest scene. At this point the scene can be rendered back onto the sensory layer to be compared with the original imagery to assess the quality of the interpretation.

In nearly every problem-solving state several computer vision operators can be applied. Furthermore, each of them can be often executed on a subregion of interest in its input datum. Escalating the choice problem complexity even higher, the operators sometimes map a single input to several alternative outputs to select from.

In order to address the numerous choice problems a control policy guiding operator application is necessary. As demonstrated in (Draper et al. 2000), a dynamic control policy can outperform manually engineered fixed policies by taking into account feedback from the environment. In each cycle the control policy chooses among applicable operators by *selectively* envisioning their effects several plies ahead. It then evaluates the states forecasted using its *heuristic value function* and selects the operator leading to the most promising state.

Deeper lookahead can often increase the quality of the approximate value function used to compare predicted problem-solving states (Korf 1990). Unfortunately, the type II performance (Good 1971) that explicitly takes computational costs into account is negatively affected as the number of envisioned states grows rapidly with the lookahead depth. As we will show in section 6, an adaptive lookahead depth selection might be necessary to optimize the overall type II performance and make the system runnable in *real-time* (e.g., on-board a surveillance aircraft).

In FIMS operators are applied to traverse the problem-solving state space. Furthermore, immediate rewards can be assigned to operator applications. 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.

This paper presents work in progress with the following research objectives: (i) creating a theory and a practi-

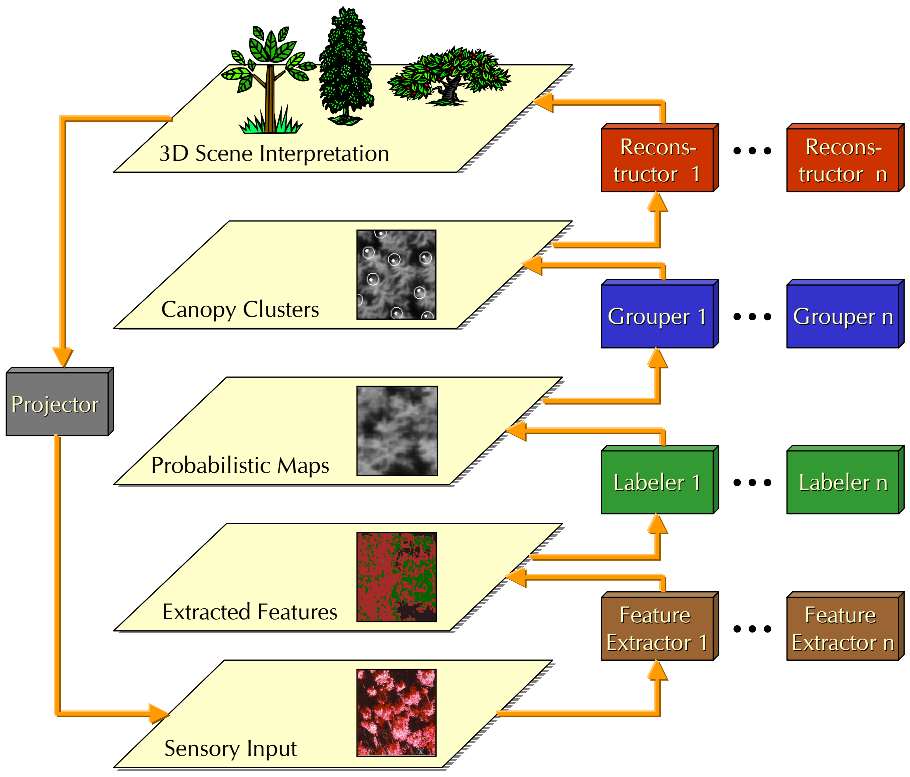

Figure 1: Forest Inventory Mapping System (FIMS) is based on a layered data representation. Raw sensory data such as aerial images are deposited at the bottom layer. As various computer vision operators are applied higher layers become populated with derived data. The output of the system is formed at the top layer and is then projected back into 2D for quality assurance. In addition to deciding among operators (e.g., feature extractor #5 vs. feature extractor #12) their inputs often need to be decided on as well (e.g., image subregion #1 vs. image subregion #87).

cal implementation of an adaptive lookahead depth selection module, (ii) studying machine learning methods for value function and domain model approximation within the large-scale FIMS application, (iii) comparing the adaptive envisionment-based control policy for FIMS with traditional hand-crafted scene interpretation strategies.

The rest of the paper is organized as follows. Section 2 describes the challenges we faced in producing this system – many due to our efforts to make the real-world system, with its mega-byte state information, run efficiently. We then (section 3) present a best-first lookahead control policy guided by an approximate MDP value function. The section discusses its advantages over optimal goal-seeking algorithms such as A* as well as the challenges in the form of value function inaccuracies. Section 4 discusses type II optimality which explicitly combines performance quality and computational time and poses the specific problem we will address in this paper: automatically determining the appropriate depth of the search. Section 5 then provides an empirical study to help us address this problem. Here we first present the artificial ”grid world” domain, and explain why this domain is rich enough to capture the challenges of the FIMS system, while remaining sufficiently constrained that we can obtain precise numbers. We then draw several important conclusions in section 6 and consider related research in section 7.

## 2 Challenges

Like many real-world systems, the FIMS control policy is faced with several formidable challenges including:

**Ill-defined goal states.** In FIMS, a goal system state contains an accurate interpretation of the forest scene observed. Unfortunately, such a state is impossible to recognize since the sought forest maps are typically unknown. The 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 requires extensions beyond the classical MDP.

**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). Section 3 describes several kinds of value function errors in more detail.

**Limited decision-making time.** Value function inaccuracies can often be compensated by carrying out a deeper lookahead (Hsu et al. 1995). On the other hand, the run time of state expansion can be exponential in the number of plies, thereby severely limiting the depth possible under time constraints (Korf 1990). This trade-off becomes especially pressing if real-time performance is desired (e.g., on-board a surveillance aircraft).

**Inaccurate domain models.** Advanced computer vision operators involving EM algorithms (Cheng et al. 2002) can take hours to compute on large images. The exponential number of operator applications often needed for the lookahead makes the actual operators unusable for envisionment. Thus, approximate versions of such operators comprising the domain model 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 approximate state values thereby off-setting the benefits of looking ahead.

**Unknown lookahead depth.** Shallow lookahead can lead to suboptimal operator applications that will have to be undone (e.g., (Draper et al. 2000), Figure 2). Overly deep lookahead severely harms type-II performance by taking excessive amounts of computational resources. Therefore, the optimal ply depth lies somewhere in between the extremes. Unfortunately, it cannot be reliably determined by a mere inspection of the changes in lookahead tree node values. Producing automated methods for *dynamic* optimal ply depth determination is one of the primary objectives of this research.

## 3 Best-First Heuristic Search

### 3.1 Basic Algorithm

Best-first heuristic real-time search has been successfully used in response to the challenges outlined above (Korf 1990; Schaeffer et al. 1992; Hsu et al. 1995; Draper et al. 2000). FIMS control policy uses a depth-limited heuristic search algorithm (called MLVA*) that expands the state tree in the best-first fashion guided by an approximate MDP value function $\widetilde{V}^*$ and action costs.

As with most best-first search algorithms, we evaluate inner nodes of the tree and, thus, expand the most promising branches first. This suggests gradual improvement of the expected quality of the running solution. MLVA*, therefore, does **not** require: (i) well-defined goal states, (ii) an exponential time to produce a meaningful solution, (iii) a precise value function, (iv) a perfect domain model, and (v) fully observable states.

### 3.2 Value Function Approximation

Value function plays a critical role in the best-first search by guiding it towards *supposedly* more promising states. In fact, given an optimal value function $V^*$ a policy can select the best action ($a^*$) with a 1-ply lookahead:

$$a^* = \arg\max_a \sum_{s'} \delta(s,a,s')\,[r(s,a,s') + V^*(s')] \quad (3.1)$$

here probabilistic state transition (or domain model) function $\delta(s,a,s') = P(s'|s,a)$ defines the probability of arriving at state $s'$ by applying action $a$ in state $s$. The reward of getting to state $s'$ by taking action $a$ in state $s$ is represented by $r(s,a,s')$.

Unfortunately, even in well-defined domains such as board games, the actual $V^*$ is usually unavailable. Thus, an approximate $\widetilde{V}^*$ (manually designed or machine learned) is used instead. In the following we consider two important kinds of errors found in approximate value functions.

### 3.3 Inaccuracies due to State Abstraction

A non-trivial feature function $\mathcal{F}$ abstracts irrelevant state details (i.e., is a many-to-one mapping) and is used to reduce an unmanageably large actual state space $S$ to a more manageable abstracted state or feature space $\mathcal{F}(S)$. In doing so, $\mathcal{F}$ will unavoidably put several raw states into a single abstracted state bucket. The bucketing process introduces the first kind of inaccuracies into the $V^*_{\mathcal{F}}(\mathcal{F}(\cdot))$ function – namely, merging two raw states that have different $V^*$ values into one abstracted state:

$$\exists s_1, s_2 [V^*(s_1) \neq V^*(s_2) \ \ \& \ \ \mathcal{F}(s_1) = \mathcal{F}(s_2) \ \ \& \\ V^*_{\mathcal{F}}(\mathcal{F}(s_1)) = V^*_{\mathcal{F}}(\mathcal{F}(s_2))]. \quad (3.2)$$

### 3.4 Inaccuracies due to Machine Learning

State abstraction via feature function $\mathcal{F}$ brings the decision process into a more manageable abstracted state space $\mathcal{F}(S)$. On the negative side, it makes it even more difficult to hand-engineer value function $V^*_{\mathcal{F}}(\mathcal{F}(\cdot))$. Often, machine learning methods are then used to induce an approximation to $V^*_{\mathcal{F}}(\mathcal{F}(\cdot))$. Consequently, the second type of inaccuracies is caused by the application of machine learning algorithms. In other words, even though the feature function $\mathcal{F}$ may map $V^*_{\mathcal{F}}$-distinct raw states $s_1$ and $s_2$ to *distinct* abstracted states $\mathcal{F}(s_1)$ and $\mathcal{F}(s_2)$, a machine-induced value function $\widetilde{V}^*_{\mathtt{ML}}$ may still misestimate their values. For instance:

$$\exists s_1, s_2 [V^*(s_1) < V^*(s_2) \ \ \& \ \ \mathcal{F}(s_1) \neq \mathcal{F}(s_2) \ \ \& \\ \neg(\widetilde{V}^*_{\mathtt{ML}}(\mathcal{F}(s_1)) < \widetilde{V}^*_{\mathtt{ML}}(\mathcal{F}(s_2)))]. \quad (3.3)$$

Remarkably, a lookahead search is able to remedy some of these inaccuracies as illustrated in Figure 2 and also discussed in (Korf 1990).

## 4 Automatic Lookahead Depth Selection

Many (Korf 1990; Schaeffer et al. 1992; Hsu et al. 1995) have argued and experimentally verified that deeper lookahead can improve the quality of action selection. On the

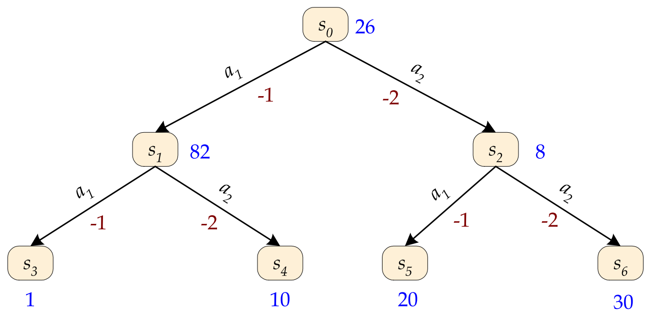

Figure 2: Deeper lookahead can address inaccuracies in the approximate value function $\widetilde{V}^*_{\mathcal{F}}$. The actual values of the optimal value function $V^*$ are displayed by the nodes while the action costs are shown by the edges. Feature function $\mathcal{F}$ fails to distinguish between distinct states $s_1$ and $s_2$ ($\mathcal{F}(s_1) = \mathcal{F}(s_2)$) resulting in $\widetilde{V}^*_{\mathcal{F}}(\mathcal{F}(s_1)) = \widetilde{V}^*_{\mathcal{F}}(\mathcal{F}(s_2))$. Consequently, a single-ply expansion (i.e., generating only $s_1$ and $s_2$) will choose less expensive $a_1$ since $\widetilde{V}^*_{\mathcal{F}}(\mathcal{F}(s_1)) = \widetilde{V}^*_{\mathcal{F}}(\mathcal{F}(s_2))$. Clearly, $a_1$ is a *suboptimal* choice as the maximum reward reachable via $a_1$ is 7 while $a_2$ can deliver as much as 26. Suppose the differences between $s_3, s_4, s_5$, and $s_6$ are pronounced enough for $\mathcal{F}$ to map them into different buckets. Thus, by expanding the tree *two* plies deep, the control policy will *optimally pick* a *more expensive* action ($a_2$) since it leads to a higher reward.

negative side, the number of states expanded can be as high as $b^p$ where $b$ is the effective branching factor and $p$ is the ply depth. Furthermore, even an efficient implementation, operating in $O(b^p)$ time, quickly incurs severe penalties on computational resources as $p$ increases. Additionally, deeper envisionment amplifies noise in the approximate versions of the operators (i.e., $\widetilde{\delta}$) used for state tree expansion. Running $\widetilde{V}^*$ on inaccurate envisioned states adds to the overall error of the value function.

Therefore, a core control problem in using a best-first any-time search is the expansion depth as a function of approximate value function and domain model inaccuracies. Ideally, the lookahead search has to be conducted just deep enough to enhance the $\widetilde{V}^*$ values and therefore increase the quality of the action selected without taking an excessive amount of time and introducing further inaccuracies. Section 7 links this concept to limited rationality.

Knowing the optimal ply depth for a lookahead policy would optimize the lookahead search performance. Unfortunately, the fine interplay between computational costs of $\delta$ and $\widetilde{\delta}$ and inaccuracies of $\widetilde{V}^*$ and $\widetilde{\delta}$ makes the "golden middle" a complex function of many domain-specific parameters. Thus, it would be very useful to have a meta-control module that dynamically selects the ply depth for a best-first lookahead action-selection policy.

This paper takes a step towards creating the theory and implementation of such an adaptive ply-depth selector by investigating the relationship between the degree of state abstraction, lookahead depth, and the overall performance of best-first search.

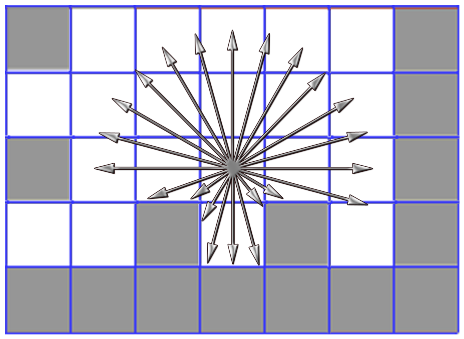

Figure 3: Agent's actions in the maze domain: $\lambda = 24, \tau = 2$. Cells occupied by walls are inaccessible (shown in grey).

## 5 Experimentation: The Maze Domain

Many Reinforcement Learning projects have used the grid world domain as a testbed (Sutton et al. 2000). In the following we refine the classical definition by introducing state abstraction in a particular fashion compatible with the real-world FIMS domain. This compatibility enables further scalability studies via transitioning from the grid world to the actual FIMS system. The refined grid world is referred to as the Maze Domain.

- The *maze* is represented by an $N \times N$ two-dimensional matrix $M$ with each cell being in two possible states: *empty* $M(x, y) = 0$ or *wall* $M(x, y) = 1$. There is a single *agent* in the maze that can occupy any of the empty cells. Additionally, one of the empty cells $(x_g, y_g)$ contains a *goal*. The maze is surrounded by a solid wall preventing the agent from wandering off the map.
- The proportion of the wall-occupied cells is called the density of the maze and denoted by $d$.
- An *agent's raw state* is a pair of coordinates $(x, y)$ with $0 \leqslant x, y \leqslant N - 1$. Formally: $S = \{0 \ldots N - 1\} \times \{0 \ldots N - 1\}$.
- The set $A$ of agent's actions is comprised of $\lambda$ equally spaced directions and a special action 'quit'. Each of the $\lambda$ move actions transports the agent along a ray shot from the agent's current location at the angle of $\frac{360 \cdot a}{\lambda}$ degrees where $a \in [0, 1, \ldots, \lambda - 1]$. The Euclidean distance travelled is deterministically upper-bounded by $\tau$ and walls encountered (Figure 3). We represent this with a deterministic state transition function $\delta_d(s, a) = s'$. The probabilistic transition function is 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 an a priori specified Normal distribution centered over action $a$.
- The MDP *immediate reward* function is defined as:

  $$r(s, a, s') = \begin{cases} \mathcal{R}(s'), & \text{if } a = \text{'quit'}, \\ -\|s, s'\|, & \text{otherwise.} \end{cases} \tag{5.1}$$

  Here the cost $\|s, s'\|$ of action $a$ is defined as the Euclidian distance along the shortest path between the new ($s'$) and

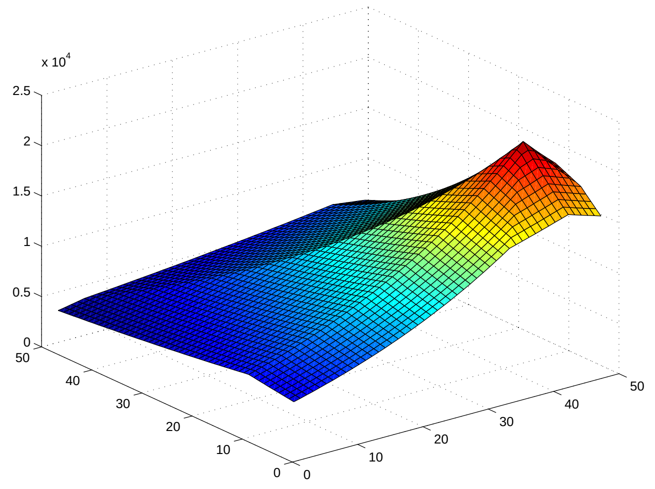

Figure 4: Terminal reward $\mathcal{R}$ for a $48 \times 48$ maze of $d = 0.0$.

the old ($s$) cells. Figure 4 illustrates the terminal reward $\mathcal{R}(s)$ of quitting in state $s$ defined as:

$$\mathcal{R}(s) = e^{\Theta - \|s, s_g\|} \tag{5.2}$$

where $\Theta$ is a constant and $s_g = (x_g, y_g)$.

- An episode terminates whenever the agent executes the 'quit' action or a move quota is exhausted.
- We define the optimal value function $V^*$ on state $s$ in the standard fashion as the maximum expected cumulative reward the agent can get from state $s$ to the end of the episode. The top plot in figure 6 demonstrates $V^*$ for a particular maze.

## 5.1 Policy Evaluation

**Policy score.** As motivated above, a policy is evaluated according to the total reward it collects during an episode ($R_g$) as well as the computational resources it takes ($NG_g$). The combined policy score for an episode is defined as:

$$\mathcal{S}_g(V, p) = \frac{R_g(V, p)}{NG_g(V, p)}. \tag{5.3}$$

Here the total reward for an episode can be computed as:

$$\begin{aligned} R_g(V, p) &= \sum_{j=1}^{J} r_j \\ &= \mathcal{R}(s_J) - \sum_{j=1}^{J-1} \|s_j, s_{j+1}\| \\ &= e^{\Theta - \|s_J, s_g\|} - \sum_{j=1}^{J-1} \|s_j, s_{j+1}\|, \end{aligned} \tag{5.4}$$

where $s_1 \to s_2 \to \cdots \to s_J$ is the actual sequence of states visited by the agent during the game episode. Computational resources taken during the game episode are defined in terms of the number of nodes expanded by the lookahead policy:

$$NG_g(V, p) = \sum_{j=1}^{J} NG_j \tag{5.5}$$

where $NG_j$ is the number of the lookahead tree nodes expanded to select action $a_j$.

**Policy error.** The relative error of the lookahead policy defined with the value function $V$ and the lookahead depth $p$ is computed as:

$$\eta(V, p) = \frac{E[R_g(V^*, 1)] - E[R_g(V, p)]}{E[R_g(V^*, 1)]} \tag{5.6}$$

where $E[R_g(V, p)]$ is the expected episode reward the agent would gain by starting at a random state and using the $p$-ply lookahead policy based on a value function $V$. Likewise, $E[R_g(V^*, 1)]$ is the expected episode reward the agent would gain by starting at a random state and using a 1-ply lookahead policy that follows the *optimal* value function $V^*$.

## 5.2 State Abstraction

Within the maze domain, the state abstraction is simulated via *fixed tiling*. Specifically, if the raw state is $s = (x, y)$ the abstracted state is given as:

$$\mathcal{F}_k(x, y) = \left( \left\lfloor \frac{x}{k} \right\rfloor \cdot k, \left\lfloor \frac{y}{k} \right\rfloor \cdot k \right) \tag{5.7}$$

where the floor function $\lfloor\ \rfloor$ returns the integer part of its argument. Parameter $k = 1, 2, \ldots, N$ is the *degree of abstraction*. Effectively, the entire maze is divided into non-overlapping rectangular $k \times k$ tiles.

The maze domain is fairly simple and compact and doesn't require state abstraction to make its state space manageable. Rather, the underlying motivation of this particular abstraction scheme is compatibility with state abstraction in FIMS. Namely, in a computer vision system like FIMS and ADORE (Draper et al. 2000), most operators add an information datum of a different category to the problem-solving state (e.g., add extracted edges to an image). The state abstraction features used for various types of data vary considerably (e.g., average brightness for images and mean curvature for extracted edges) making it unlikely that an operator application leaves the agent in the same abstraction tile. On the other hand, several similar operators can move the agent to a single tile (different from the current one). Correspondingly, as shown in Figure 3, several actions along neighboring rays can end up in the same tile. Furthermore, some computer vision operators are not applicable in certain problem-solving states which is modelled by certain angles being blocked off by the maze walls.

## 5.3 Inaccuracies in $\widetilde{V}^*$ due to Fixed-tiling

The abstracted version of the value function $\widetilde{V}^*$ is defined as follows:

$$\begin{aligned} \widetilde{V}^*_{\mathrm{FT},k}(\mathcal{F}_k(s)) &= \underset{s_t \in \mathcal{N}_{\mathrm{FT},k}(s)}{\operatorname{avg}} V^*(s_t) \\ &= \frac{1}{|\mathcal{N}_{\mathrm{FT},k}(s)|} \sum_{s_t \in \mathcal{N}_{\mathrm{FT},k}(s)} V^*(s_t). \end{aligned} \tag{5.8}$$

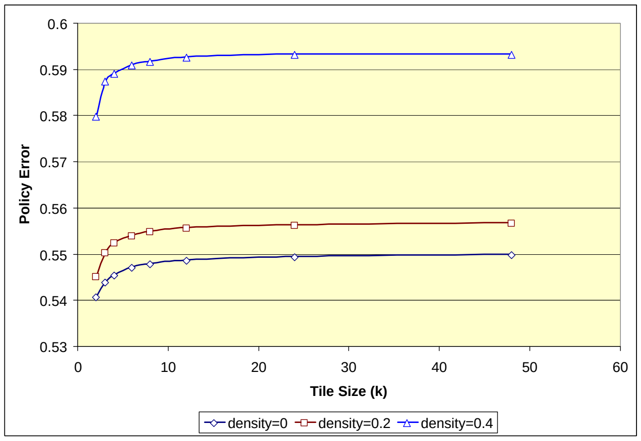

Figure 5: Policy error $\eta(\widetilde{V}^*_{\mathtt{FT},k}, 1)$ increases as the tile size $k$ goes up.

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.

Intuitively, $\widetilde{V}^*_{\mathtt{FT},k}$ suffers from state abstraction *only* and, thus, lower values of $k$ bring $\widetilde{V}^*_{\mathtt{FT},k}$ closer to the optimal value function $V^*$. Figure 5 presents the increase in policy error $\eta(\widetilde{V}^*_{\mathtt{FT},k}, 1)$ as the degree of abstraction $k$ goes up (section 5.1 defines the policy error formally). Figure 6 demonstrates the step-function nature of the approximation.

## 5.4 Inaccuracies in $\widetilde{V}^*$ due to Machine Learning

One of the common approaches to machine learning an approximation to value function $V^*$ is to expand the state tree completely on each pre-labelled training datum during a designated off-line learning phase (Draper et al. 2000). The complete off-line expansion delivers a body of training data in the form of pairs $\{\langle \mathcal{F}(s), V^*(s)\rangle\}$. Supervised machine learning methods can be then used to generalize the set into a value function approximation:

$$\widetilde{V}^*_{\mathtt{ML},k}(\mathcal{F}_k(\cdot)) = \mathbf{ML}\left(\{\langle \mathcal{F}_k(s), V^*(s)\rangle\}\right) \qquad (5.9)$$

In our experiments presented we used backpropagation for Artificial Neural Networks (Haykin 1994). Note that, unlike $\widetilde{V}^*_{\mathtt{FT},k}$, the induced approximation $\widetilde{V}^*_{\mathtt{ML},k}$ suffers from *both* state abstraction and generalization inaccuracies (e.g., caused by an insufficient training data set).

Figure 6 presents a plot of ANN-induced value function ($\widetilde{V}^*_{\mathtt{ML},k}$) for a $48 \times 48$ maze with no walls. A comparison with the fixed-tiling approximation (in the same figure) reveals *additional* inaccuracies due to the machine learning process.

## 6 Analysis

In spite of the exponential nature of $\mathcal{R}$, deeper lookahead is not necessarily an advantage since: (i) the quality of the solution found is upper-bounded by the quality of the optimal solution, (ii) the resources consumed can grow virtually unlimitedly (Figure 7), and (iii) the accuracy of the envisioned future states can degrade rapidly given a noisy domain model approximation $\widetilde{\delta}$.

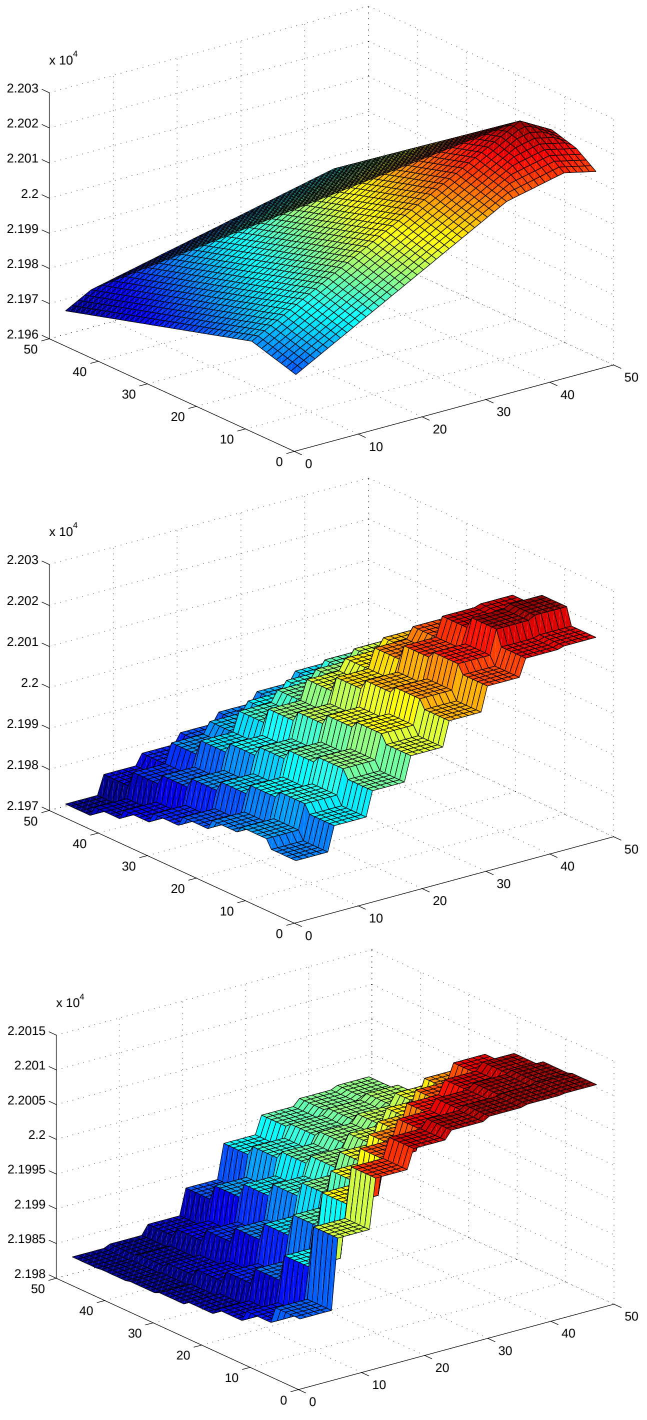

Figure 6: Inaccuracies in value function approximations. The plots demonstrate (top to bottom): the original $V^*$, fixed-tiling averaged $\widetilde{V}^*_{\mathtt{FT},k}$, and ANN-learned $\widetilde{V}^*_{\mathtt{ML},k}$.

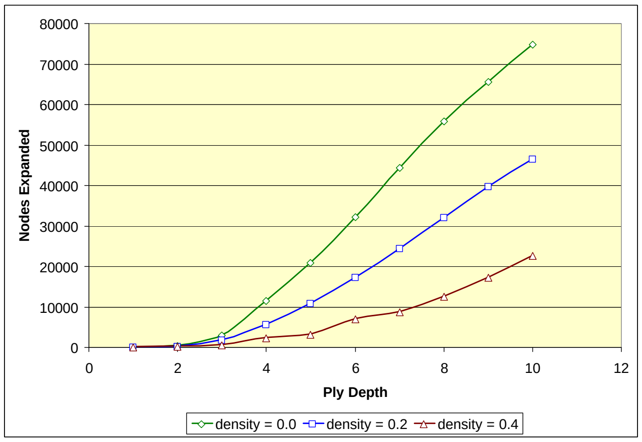

Figure 7: The number of nodes expanded by the lookahead search algorithm (MLVA*) grows rapidly as the lookahead depth increases. Note that the combinatorial explosion is more noticeable in mazes of lower density $d$ as more cells are unoccupied increasing the number of feasible actions and thereby the effective branching factor. Also note that the growth is sub-exponential due to the fact that only $O(p^2)$ maze cells are reachable with $p$-ply lookahead.

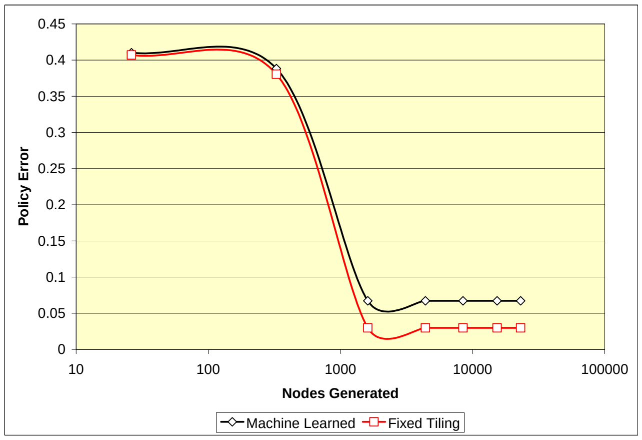

Figure 8: Policy errors $\eta(\widetilde{V}^*_{\mathtt{FT},k}, p)$, $\eta(\widetilde{V}^*_{\mathtt{ML},k}, p)$ plotted vs. the number of nodes generated $NG_g$. Each marker on the curves corresponds to one ply in the lookahead search.

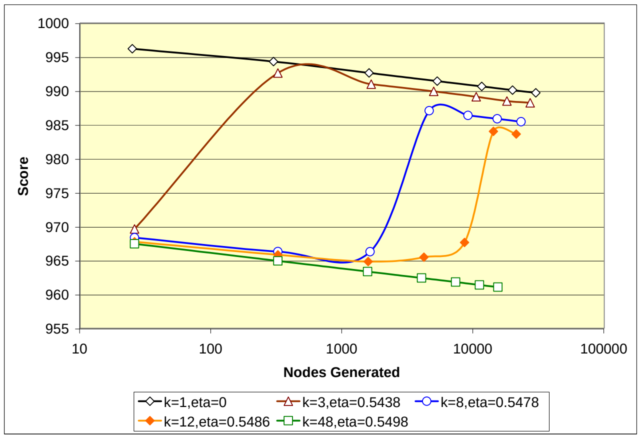

Figure 9: Fixed-tiling: overall score $\ln \mathcal{S}_g(\widetilde{V}^*_{\mathtt{FT},k}, p)$ vs. the number of nodes generated $NG_g$. Each marker on the curves corresponds to one ply in the lookahead search. In this plot we ignored the effects of action costs.

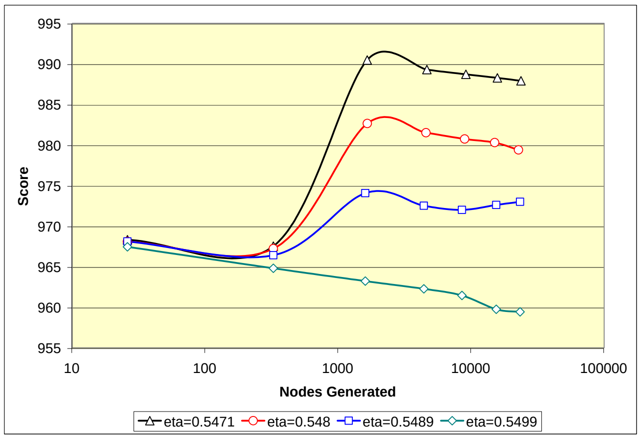

Figure 10: Machine-learning: score $\ln \mathcal{S}_g(\widetilde{V}^*_{\mathtt{ML},k}, p)$ vs. the number of nodes generated $NG_g$. Each marker on the curves corresponds to one ply in the lookahead search. In this plot we ignored the effects of action costs.

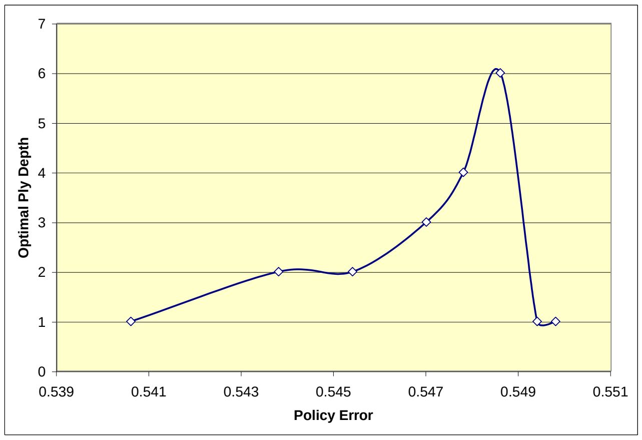

Figure 11: Optimal ply depth $p^*$ vs. the initial ($p = 1$) policy error $\eta(\widetilde{V}^*_{\mathtt{FT},k}, 1)$.

We are currently working on an analytical study of the empirical evidence. Preliminary results are as follows:

1. As expected, lookahead reduces the policy error. Figure 8 presents the empirical results for the policies guided by a fixed-tiling abstracted approximation $\widetilde{V}^*_{\mathtt{FT},k}$ and a machine-learned approximation $\widetilde{V}^*_{\mathtt{ML},k}$.
2. Lookahead initially improves overall policy performance in the presence of *inadmissible* value function inaccuracies induced by *state abstraction* or *machine learning* (Figures 9 and 10). While deeper values of lookahead do decrease the policy error (Figure 8) they also result in the larger numbers of nodes expanded (Figure 7) thereby adversely affecting the overall score.
3. The optimal lookahead search horizon ($p^*$) is reached fairly rapidly. Figure 11 presents the empirical results for a fixed-tile approximation to $V^*$. Note, as the value function approximation $\widetilde{V}^*_{\mathtt{FT},k}$ becomes progressively more inaccurate (i.e., its policy error $\eta(\widetilde{V}^*_{\mathtt{FT},k},1)$ increases), the optimal ply depth decreases again partly due to the combinatorial explosion of $NG_g$.
4. Remarkably, the nature of ANN-induced inaccuracies in $\widetilde{V}^*_{\mathtt{ML},k}$, does *not* call for deeper lookahead as the policy error increases. Indeed, figure 10 demonstrates that the best score is always reached at $p^* = 3$ regardless of the initial $\eta(\widetilde{V}^*_{\mathtt{ML},k},p)$. The attributes of $\widetilde{V}^*_{\mathtt{ML},k}$ leading to this phenomenon are yet to be better characterized.

## 7 Related Work

As our evaluation criteria depends on both the quality of the outcome, and the computational time required, it is clearly related to "type II optimality" (Good 1971), "limited rationality" (Russell et al. 1991), and "meta-greedy optimization" (Russell et al. 1993; Isukapalli et al. 2001). Our results differ as we are considering a different task (depth of search in the context of an image interpretation task), and we are isolating the problem of adaptively selecting the optimal lookahead depth.

### 7.1 ADORE

Adaptive image recognition system ADORE (Draper et al. 2000) is a well-known MDP-based control policy implementations in computer vision. The FIMS prototype can be considered as a testbed for several significant extensions over ADORE. The extension most relevant to this paper is the emphasis on internal lookahead. In other words, ADORE selects operators in a greedy (i.e., the lookahead of depth 1) fashion guided by its MDP value function. As a result, the existing inaccuracies in the value function caused by state abstraction or machine learning deficiencies, often force ADORE to undo its actions and backtrack. On the other hand, FIMS attempts to predict effects of *action sequences* and is expected to be less sensitive to value function inaccuracies.

### 7.2 Minimax and Minimin

Most board game-playing programs use a lookahead search. In particular, the first computer program to ever win a man-machine world championship, Chinook (Schaeffer et al. 1992), and the first system to defeat the human world-champion in chess, Deep Blue (Hsu et al. 1995) utilized a minimax-based lookahead search. Like FIMS, they compensate for the insensitivity of $\widetilde{V}^*$ via envisioning the future states many plies ahead. The core differences between minimax and minimin (Korf 1990) algorithms and FIMS include minimax's easily recognizable goal states, perfect domain model $\delta$, fully deterministic actions, and the use of full unabstracted states for the lookahead. Additionally, some algorithms (e.g., branch and bound or minimax with alpha pruning) leave out entire search subtrees by assuming an admissible heuristic function which is unavailable in FIMS. An interesting study of minimax pathologies is presented in (Nau 1983).

### 7.3 Minerva

Real-time ship-board damage control system Minerva (Bulitko 1998) demonstrated a 318% improvement over human subject matter experts and was successfully deployed in the US Navy. It selects actions dynamically using an approximate Petri Nets based domain model ($\widetilde{\delta}$) and a machine-learned decision-tree-based value function ($\widetilde{V}^*$). The key distinctions between Minerva and FIMS include a fixed lookahead depth and fixed state abstraction features used for envisionment. Therefore, the focus of this paper – the interplay between the lookahead depth and the degree of state abstraction – is not considered in the research on Minerva.

## 8 Summary and Future Research

This paper describes the practical challenges of producing an efficient and accurate image interpretation system (FIMS). We first indicate why this corresponds to a Markov decision process, as we apply a *sequence* of operators to go from raw images to an interpretation, and moreover must decide on the appropriate operator *dynamically*, based on the actual state.

At each stage, FIMS uses a real-time best-first search to determine which operator to apply. Due to the size of individual states, FIMS actually uses *abstractions* of the states, rather than the states themselves making the entire process POMDP. The heuristic value function, used to evaluate the quality of the predicted states, depends on the correctness of the final interpretation; as FIMS is dealing with novel images, this information is not immediately available. Instead, we first learn an approximation to this heuristic function given a small set of pre-labelled training images. In addition, as FIMS must select the appropriate operator quickly, we want a search process that is both accurate and efficient.

This paper investigates the effectiveness of the lookahead policy given these challenges. In particular, we take a step towards determining, automatically, the appropriate lookahead depth we should use, as a function of inaccuracies due to the state abstractions as well as the machine-learned estimates of the heuristic, with respect to our specific "type II optimality". Applications of this research range from time-constrained game playing and path-finding to real-time image interpretation and ship-board damage control.

The empirical study in a FIMS-compatible variant of the grid world testbed gives us a better understanding of the fine interplay between various factors affecting performance of a lookahead best-first control policy. It therefore sets the stage for developing a theory and an implementation of a dynamic and automated lookahead depth selection module. One of the immediate open questions is the relation between inaccuracies in the *approximate* domain model $\tilde{\delta}$, the approximate value function $\widetilde{V}^*$, and the optimal lookahead search depth. Likewise, better understanding of the optimal lookahead depth in the presence of machine-learning inaccuracies is needed.

## Acknowledgements

Guanwen Zhang and Ying Yuang have assisted with the development of the maze domain environment at the University of Alberta. We thank Terry Caelli, Omid Madani, and Alberta Research Council staff for an insightful input and versatile support. The paper also benefited from the comments by anonymous reviewers. Finally, we are grateful to the funding from the University of Alberta and NSERC.

## References

Bulitko, V. 1998. *Minerva-5: A Multifunctional Dynamic Expert System*. MS Thesis. Department of CS, Univ. of Illinois at Urbana-Champaign.

Cheng, L., Caelli, T., Bulitko, V. 2002. Image Annotation Using Probabilistic Evidence Prototyping Methods. *International Journal of Pattern Recognition and Artificial Intelligence*. (in preparation).

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

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

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

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

Hsu, F.H., Campbell, M.S., Hoane, A.J.J. 1995. Deep Blue System Overview. *Proceedings of the 9th ACM Int. Conf. on Supercomputing*, pp. 240-244.

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

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

Larsen, M. and Rudemo, M. 1997. Using ray-traced templates to find individual trees in aerial photos. *In Proceedings of the 10th Scandinavian Conference on Image Analysis*, volume 2, pages 1007-1014.

Nau, D.S. 1983. Pathology on Game Trees Revisited, and an Alternative to Minimaxing. *Artificial Intelligence*, 21(1, 2):221-244.

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

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.

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

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

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.

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