On Approximating the Best Decision for an Autonomous Agent

Gerardo I. Simari
Laboratorio de Investigación y Desarrollo
en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería
de la Computación
Universidad Nacional del Sur, Av. Alem 1253
(8000) Bahía Blanca, Argentina
gis@cs.uns.edu.ar

Simon Parsons
Department of Computer
and Information Science
Brooklyn College,
City University of New York
2900 Bedford Avenue,
Brooklyn, NY 11210, USA
parsons@sci.brooklyn.cuny.edu

Abstract

In this paper we are concerned with decision making in autonomous agents and, in particular, the tradeoff between the optimal solution provided by MDP*s and the more tractable approximation provided by the* BDI *model. In order to establish the relative performance of the approaches for the* TILEWORLD *domain, we have to first find approximations for the optimal* MDP *solution, and we demonstrate that even these approximations out-perform the* BDI *model for small domains. However, for large domains these approximations are less effective and the* BDI *approach performs better.*

1. Introduction

For any autonomous agent, whether a softbot playing a computer game [2], a web spider [16], or a robot playing soccer [8], the key thing it has to do is to decide what to do next [23]. As a result, the problem of establishing the best mechanism by which an agent can make this decision has been widely studied, and a number of approaches have been formulated. We can distinguish two broad classes of approach.

One class is that of *descriptive* approaches—approaches that are based on analysing the way that people or animals make decisions. These approaches include, for example, the belief/desire/intention (BDI) approach [5] and the behaviour-based approach [3]. Such approaches have often led to agent architectures which support the development of agents that use the approach to decision making. For example, one can look at the PRS [9] as an architecture for BDI decision making, and the subsumption architecture [6] as an architecture for behaviour-based [3] decision making.

Another class is that of *prescriptive* approaches—approaches which attempt to identify the optimal decision. These are typically based around decision theory [15], and one family of approaches within this class, a family that is currently the subject of much research interest, is the family of Markov decision processes (MDPs) [14].

Since the BDI model, and implementations thereof, have been widely used by agent developers, it is interesting to ask about the quality of the decisions that the model makes. It seems natural that this will depend upon the exact nature of the task, and this was experimentally validated by Kinny and Georgeff [10]. In particular Kinny and Georgeff showed that the performance of an agent depended upon the speed with which its environment changed, the amount of information the agent has at its disposal, and the likelihood of its actions having their intended effect.

Another of Kinny and Georgeff's findings was that the performance of the agent depended upon how often, broadly speaking, it considered whether it had made the right decision (its *commitment strategy* in the language of the BDI model). Following up on this, Schut and Wooldridge [17, 18, 19] considered a range of methods for making this meta-level decision about whether the last decision was still a good one, even using an MDP model [20] to optimise it.

All of this work, however, has only been able to compare different commitment strategies relative to one another rather than with any notion of what the optimum performance is. All that we know is that, as a heuristic approach, the BDI model is likely to be sub-optimal. We just don't know *how* sub-optimal. The trade-off, the reason we may be prepared to accept this sub-optimality, is that the BDI model is presumably much more tractable than an optimal solution, but since nobody has looked at the optimal solution, this is only a hypothesis.

Our aim in this paper is to investigate the trade-off.

Building on that of [20], we take the problem studied by both Kinny and Georgeff and Schut and Wooldridge, and provide a solution using an MDP. This can be considered optimal in a decision-theoretic sense (though other notions of optimality are possible). In fact, it turns out that, as hypothesized, the MDP solution is intractable for interesting-sized versions of the problem, so we have to resort to some novel approximations in order to make a meaningful comparison with BDI. We then provide comparisons between the MDP approximation and the BDI model for two different-sized versions of the TILEWORLD.

This paper is organized as follows: in Section 2, we present an introduction to the BDI architecture, MDPs, and then the TILEWORLD domain we will use for our experiments. Section 3 introduces two approaches to approximate MDP solutions which we need to adopt for the TILEWORLD, Section 4 discusses the experimental setup, and Section 5 gives our results. Finally, Section 6 summarizes our findings and discusses future work.

2. Background

Before we describe our contribution, we first describe the BDI architecture, MDPs, and then the TILEWORLD domain (and the way we can use the BDI architecture and MDPs to provide agents with a solution to it).

2.1. The BDI Model

The BDI model has its roots in the philosophical tradition of understanding *practical reasoning* [5]. This type of reasoning can be described as the process of deciding what actions to perform in order to reach a goal. Practical reasoning involves two important processes: decide *what* goals to try and reach, and *how* to reach them. The first process is known as *deliberation*, and the second as *means-ends* reasoning.

The practical reasoning process can be broken down into a number of basic components; the following are in general part of a BDI model [22]:

- A set of current *beliefs*, which represents the information the agent currently has about its environment.
- A *belief revision* function, which takes a perceptual input and the agent's current beliefs and, based on this, determines the new set of beliefs.
- An *option generation* function, which determines the options available to the agent based on the current beliefs about the environment and its current *intentions*. This function represents the agent's means-ends reasoning–the process of deciding how to achieve intentions. It maps a set of beliefs and a set of intentions into a set of *desires*.
- A set of *current options*, which represents the agent's possible courses of action.
- A *filter* function, which represents the agent's *deliberation* process, and which determines the agent's intentions based on its current beliefs, desires, and intentions.
- A set of current *intentions*, which represents the agent's current focus, *i.e.*, those states to which it is committed to arrive.
- An *action selection* function, which determines an action to perform based on the current intentions.

The desired behavior will have an impact on how each of these components is implemented. Such variety in possible implementations causes this architecture to be considered a *family* of models, rather than a rigid design method.

2.2. Markov Decision Processes

We are concerned with the problem of choosing optimal actions in complex *stochastic* environments, in other words environments in which actions have a *set* of possible effects, each having a *probability* of occurrence associated with it. These *transition probabilities* depend only on the state and the action carried out, not on the agent's previous history, and we will assume that agents always know *exactly* what state they are in. Markov decision processes are one approach to decision making in this kind of environment.

A Markov decision process can be formally defined [11] as a tuple $M = (S, A, T, R, \beta)$, where:

- S is a finite set of *states* of the environment.
- A is a finite set of *actions*.
- $T : S \times A \rightarrow \Pi(S)$ is the *state transition function*. It gives, for each state and action performed by the agent, a probability distribution over states ($T(s, a, s')$ is the probability of ending in state s', given that the agent started in state s and performed action a).
- $R : S \times A \rightarrow I\!R$ is the *reward function*, which gives the expected immediate reward gained by the agent for taking each action in each state.
- $0 < \beta < 1$ is a *discount factor*. This factor is used to represent that a reward obtained in the future is less valuable than an immediate one. Its use ensures that the reward obtained by a policy is not unbounded, and that the algorithm used to calculate the utilities converges.

The problem is, then, to find the best way to behave given this model of the environment. The same problem has been addressed in AI as *planning*, but the consideration of stochastic domains forces us to depart from the traditional

model and compute solutions in the form of *policies* instead of action sequences (plans). A policy is a complete mapping from states to actions; once a policy is calculated, it is trivial to decide what to do. The agent's decision function is represented explicitly by the policy: it describes a simple reflex agent.

One way to obtain an optimal policy is *value iteration* [14], which basically calculates the utility of each state using dynamic programming techniques, and then uses these values in the selection of an optimal action for each state. Because actions have no guaranteed effect, the calculation of utilities is not straightforward, but can be done to any degree of accuracy by using an iterative procedure.

The main drawback of algorithms for directly solving MDPs, is their intractability for even relatively small problems. A naïve approach for finding an optimal policy (one which tries every possible combination) would be $O(|A|^n)$, where n is the number of steps in the decision problem. This would preclude exhaustive search even for small values of $|A|$ and n.

In the dynamic programming approach [14], the cost of calculating the utility of one state is $O(|A|)$, and therefore the whole computation is $O(n|A||S|)$, or $O(|A||S|)$ if discounting is used. So, in general, each iteration of the value iteration algorithm takes $O(|A||S|)^2$ steps. The number of iterations required to reach an optimal policy can be proved to be bounded above by a polynomial in $|S|$, $|A|$, B, and $1/(1-\beta)$, where B is used to designate the maximum number of bits needed to represent any numerator or denominator of β, or one of the components of T or R. As a lower bound, value iteration has a worst case run time that grows faster than $1/(1-\beta)$ [11, 21].

2.3. The TILEWORLD Domain

The TILEWORLD testbed [13] is a grid environment occupied by agents, tiles, holes, and obstacles. The agent's objective is to score as many points as possible by filling up holes, which can be done by pushing the tiles into them. The agent can move in any direction (even diagonally); the only restriction is that the obstacles must be avoided. This environment is *dynamic*, so holes may appear and disappear randomly in accordance to a series of world parameters, which can be varied by the experimenter.

Because this environment, though simple to describe, is too complex for most experiments, we adopted the simplified testbed used in [10, 18]. The simplifications to the model are: tiles are omitted, so an agent can score points simply by moving to a hole; agents have perfect, zero-cost knowledge of the state of the world; and agents build correct and complete plans for visiting a single hole (they do not plan tours for visiting more than one hole). This domain, although simplistic, is useful in the evaluation of the effectiveness of situated agents. One of its main advantages is that it can be easily scaled up to provide difficult and unsolvable problems.

A BDI agent for the TILEWORLD can be easily implemented. The agent's beliefs consist of its perceptions of the locations of holes in the world. Various parameters dictate how accurate these beliefs are: *accessibility* (how many positions the agent can see from where it is standing), *dynamism* (how many steps the world takes for each step of the agent), *planning cost* (how many steps the agent must spend in order to build a plan), and the agent's *intention reconsideration* strategy. This last parameter is one of the most important because it has a great influence on how efficient the agent will be; if intentions are reconsidered too often or too soon, this will lead to a waste of effort [24].

The next component, *desires*, can be seen under this model as possible plans leading to a selected goal. In our case, any series of actions leading from the agent's current position to a hole constitutes a desire. On the other hand, an *intention* is a desire selected in order to reach a goal. The agent will select one such intention, and will use it to fill holes in the best possible way. Intention reconsideration in this domain corresponds to the frequency in which the agent revises its plans: *cautious* agents reconsider after each action taken, while *bold* ones wait until the current plan is completed to build a new one.

The computational costs associated to this model are low: plans can be built in time linear in the size of the world, and there is no off-line cost because all of the processing is done during execution time. However, this does not mean that BDI agents are *always* efective: the burden lies at the agent's *reconsideration strategy*, which can lead the agent to wasted efforts if it is sub-optimal.

For an MDP model, the world is modelled by taking into account *every* possible action in *every* possible state. For the simplified TILEWORLD, this means that for a world of size n (that is, an $n \times n$ grid) there is a set of 8 actions, n^2 possible positions for the agent, and 2^{n^2} possible configurations of holes. This last number is obtained by considering that every position in the grid may contain a hole or not. A *state* in this world consists of a pair (P, H), where $P = (i, j), 0 \leq i, j \leq n-1$, and H represents a given configuration of holes on the grid. Therefore, the total number of states in this case is $n^2 2^{n^2}$; Table 1 presents the number of states for some values of n.

These values show that even for a very small TILEWORLD, the MDP model requires an intractable amount of resources (both time and space) in order to compute an optimal policy. The limit for the tractability of direct calculation seems to be at $n = 4$, or $n = 5$ for a computer with reasonable resources, which is well below what is possible in the BDI model. Even with the many techniques that have been proposed for solving MDPs (for example [1, 4]) that

n	Number of states
3	4,608
4	1,048,576
5	838,860,800
6	2,473,901,162,496
7	$\approx 10^{16}$
8	$\approx 10^{21}$
9	$\approx 10^{26}$
10	$\approx 10^{32}$

Table 1. The number of states in the TILEWORLD

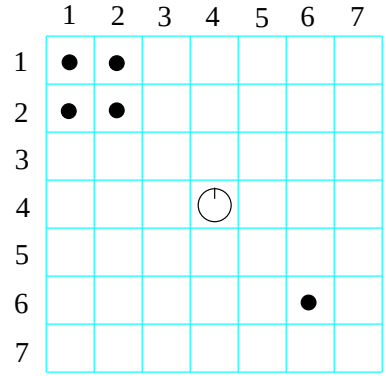

Figure 1. An example 7×7 TILEWORLD configuration

are more efficient than the direct calculation, intractability is going to be an issue. For example, the state reduction reported in [12] is less than a factor of 1000. Impressive though this is, it will not permit the solution to be scaled to the 7×7 TILEWORLD. This is one reason why the BDI model is interesting—it can easily handle much larger versions of the TILEWORLD with little problem.

However, these tractability issues don't mean that the MDP model cannot be used at all. The "explosion" in the number of states, as we have seen, depends largely on the amount of holes that can be present at a given moment, and this gives us a means of approximating the solution by pretending that there are fewer holes than there really are.

3. Approximate solutions

In this section we describe some techniques that we have examined as a way to approximate the MDP for intractable sizes of the TILEWORLD. We believe that these are interesting not only because of their role in our experiments, but also because similar approaches may be useable in other applications of MDPs. So far as we know, existing techniques for obtaining approximate solutions of MDPs [1] take a rather different approach.

3.1. Reducing State Information

One possible way of keeping the computation of policies within acceptable bounds is to consider a *reduced state space*. This means that the agent will no longer have complete information regarding the current state of the world, *i.e.*, the current state will be *hashed* into one of the states in the reduced space. Of course, this will generally mean that the agent will no longer be able to select the optimal action in each step. The action taken by the agent will be optimal with respect to the reduced state space, and sub-optimal with respect to the full state-space in general. Nevertheless, if the hashing is a good one, the action will approximate the optimal action for the full state space. This hashing can be done in a variety of ways trading-off between number of states and optimality. The following sections describe these proposals.

3.1.1. The Closest Hole The agent is only aware of the closest hole. Even though it is simple, this strategy cannot ensure good results because an agent will not be able, for example, to decide between two closest holes as in Figure 1 ($n = 7$). Here, even though (2,2) and (6,6) are both the closest to the agent, (2,2) is the best option because it is close to a group of holes. This behavior is captured by value iteration (position (2,2) will have a greater utility than (6,6)), but will be ignored by the closest hole approach, which effectively suffers from a version of the horizon effect—it cannot "see" far enough to pick the better solution. However, the advantage is that the number of states is only n^4, which is much smaller than the full state space.

3.1.2. The k Closest Holes The agent keeps track of the k closest holes. The parameter k can be varied in order to trade efficiency against cost: $k = 1$ yields the previous approach, while $k = n^2$ is the general case discussed above. This generalization is meant to deal with the difficulties that arise for $k = 1$, pushing back the search horizon, but still suffers from cases like that in Figure 2 ($n = 7$, $k = 4$) where the agent will regard the four closest holes as equal and will have to make a random choice among them, even though they all have different utilities. Here, the number of states grows to $n^2 \sum_{i=1}^{k} C_i^{n^2} \leq n^2 2^{n^2}$, where C_k^n represents the number of combinations of size k taken from a set of size n.

3.1.3. Local Density In a variation on the previous solution, the agent keeps track of the k closest holes along with additional information regarding the *local density* (in terms of holes) of the area in which the hole lies. This informa-

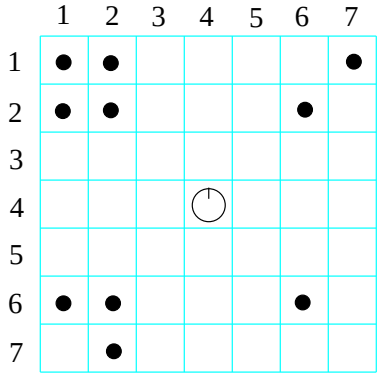

Figure 2. Another example 7×7 TILEWORLD configuration

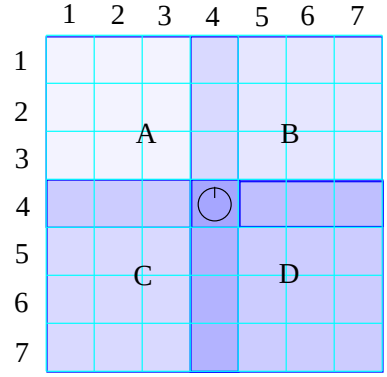

Figure 3. The moving window in the 7×7 TILEWORLD

tion can be used to make the right decision in cases like that of Figure 2.

The local density for a given position (i, j) that contains a hole can be calculated as the sum of the *influence* of every other hole in the world, where the influence is a value obtained as the product of the hole's reward and the inverse of the distance that lies between it and (i, j). More formally, the *density* of position (i, j) is

$$D_{i,j} = \sum_{(s,t)\in H,\ (s,t)\neq(i,j)} R'_{s,t} \frac{1}{Dist_{(i,j),(s,t)}}$$

where H is a set of ordered pairs containing the location of every hole in the grid, R' is the *reward function* on states, and $Dist_{(i,j),(s,t)}$. In this approach, the number of states is the same as in the previous section because the size of the hashed state space has not changed, only the way the hashing is done. The calculation of the density information yields the additional cost. The distance between holes and the reward function on states can be calculated in $O(1)$, so the complete density information for *one* state can be obtained in $O(n^2)$.

3.2. Using Optimal Sub-solutions

When reducing state state information we are pretending the problem is simpler than it is without changing its size. We can also just pretend the problem is smaller than it really is, making use of the solution to the biggest version of the problem that we can compute. One approach to doing this uses four 4×4 grids to make up a new 7×7 grid, which will be referred to as the *moving window*. This can be done by putting the four grids together so they overlap one row; we will call these sub-grids A, B, C, and D (see Figure 3). This 7×7 grid is a "moving window" because it follows the agent's moves, representing the limit of its perception. The agent occupies the center (*i.e.*, the SE corner of A, the SW corner of B, the NE corner of C, and the NW corner of D), and it will only be aware of the holes that fall inside this window. For example, if the agent is in the center of the grid, then it has full accessibility; now, if it moves North one position, it will not be able to see the holes to the far South.

The new policy is built using the solved 4×4 sub-grids. For each state, the agent will have four possibly different suggestions (one for each 4×4), given by the policies:

$$\pi_i(s) = \arg\max_a \sum_t T_i(s, a, t) U_i(t)$$

where $i = A, B, C, D$, T_i is the state transition function for grid i, and U_i is the utility function on states for grid i. The best action is then the one that has the highest expected utility:

$$\pi^*(s) = \pi_i(s)$$

for some i such that $\pi_i(s) = t_i$, where $U_i(t_i) = u_i$ and there is no u_j such that $\pi_j(s) = t_j$, $U_j(t_j) = u_j$, and $u_j > u_i$.

The computational cost of this approach is not significantly greater than the cost of solving an MDP for a TILEWORLD of size 4×4. Because the policy is built from the policy for a 4×4, the only difference lies in execution time, where the hashing and the selection of the best action among the four suggested take place. The hashing can be done in $O(n^2)$ because each position must be tested to see if it falls inside the moving window; the execution time of both operations are clearly in $O(1)$.

4. Experimental details

We have carried out some initial experimental investigation into the approximations described above. We started from the experimental framework developed by Schut [17],

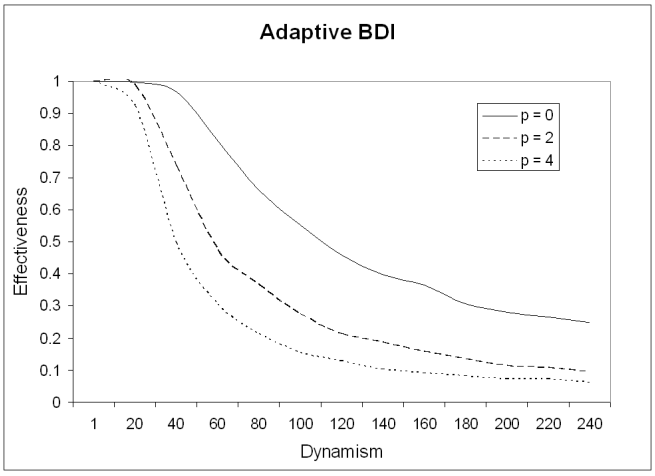

Figure 4. Results for the BDI model, 7×7 TILEWORLD

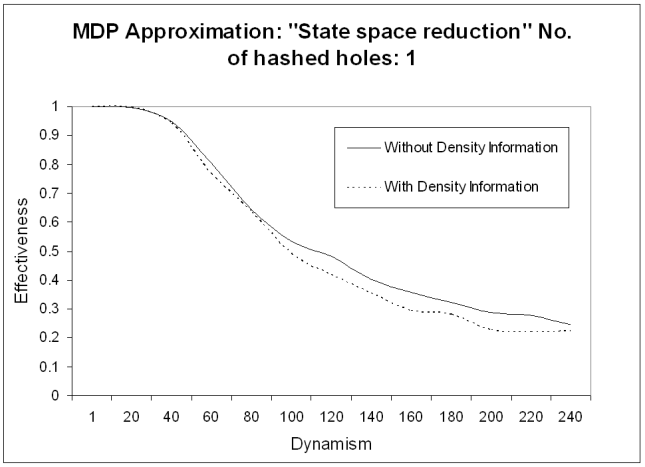

Figure 5. Results for the "one hole" MDP approximation, 7×7 TILEWORLD

using his implementation of the TILEWORLD and the adaptive BDI model [20] in JACK Intelligent Agents [7], and writing our own MDP solver. We then ran the BDI model and a selection of the MDP approximations (k closest holes for $k = 1, 2, 3$—each with and without the local density information—and the moving window) for a range of different values of dynamism. For each value of dynamism we ran 20 iterations of the same model. Initially, the experiments were carried out in a 7×7 TILEWORLD. The final results, however, are obtained from a generalization of the previous experiments, where the agents occupy a 20×20 TILEWORLD in order to see how the different models respond to growing state spaces.

5. Results

At this preliminary stage of our work, we have run two sets of experiments. The first consider a relatively small, 7×7 TILEWORLD to investigate the behaviour of the approximations introduced above on an example that is a little larger than that soluble by the current best approximations. The second examines a much larger, 20×20, TILEWORLD to see how the approximations fare on a much larger example.

5.1. A small example

The results for the 7×7 TILEWORLD are best summarised by Figure 4 to 7. We will explain each of these in turn. Figure 4 shows our replication of Schut's experiments. This shows that as the world becomes more dynamic, that is the holes appear and disappear with greater frequency, the effectiveness of the agent (measured as the proportion of all the holes that appear that the agent reaches) falls. This is to be expected—after a point many of the holes tend to disappear before the agent can reach them. Using the BDI approach the agent periodically has to replan. When this happens the agent pauses for a certain number of timesteps. These are the values of p that generate the three different curves.

Figure 5 shows the performance of one of our MDP approximations, in this case the $k = 1$ closest holes model, with and without local density information. Interestingly the local density information seems to detract from the performance of the model (the same trend was observed for all the approximations). The general shape of the curves is the same as for the BDI model. Figure 6 shows the performance of all the MDP approximations against one another. Though the relative performance of the models varies with dynamism, the $k = 3$ nearest holes approximation seems to consistently outperform the others[1].

Finally, figure 7 makes a comparison between the best performing MDP approximation and Schut's BDI model. This shows that the MDP model outperforms the BDI model even when the BDI model doesn't have to spend any time replanning, and thus outperforms the best known solution to the TILEWORLD.

These results indicate two things. First of all our approximations are sufficient to allow a naive approximate ap-

1 Clearly we need to run further experiments to establish if this really is the case.

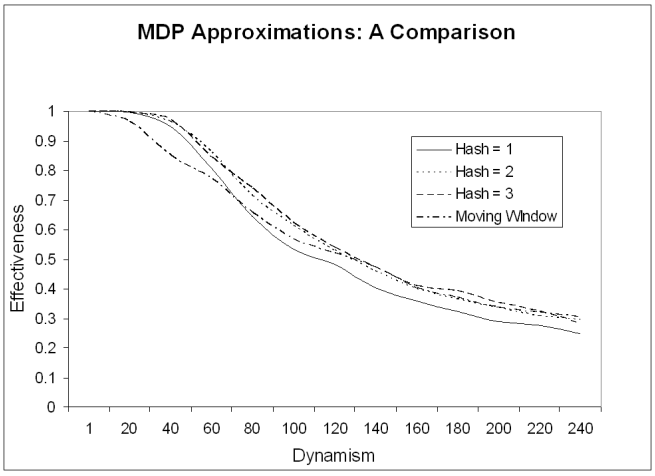

Figure 6. Comparing the MDP approximations for the 7×7 TILEWORLD

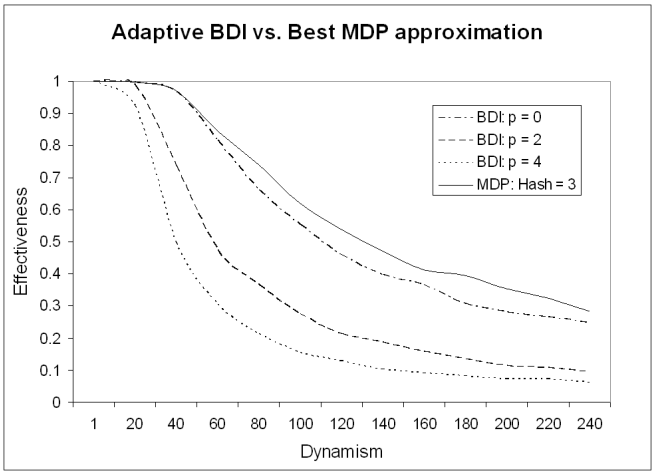

Figure 7. Comparing the BDI model with the best MDP approximation for the the 7×7 TILEWORLD

proach to solving the underlying problem as an MDP to be stretched to a world somewhat larger than would otherwise be soluble using state of the art techniques. Second, these approximations are sufficiently good that even though we know they are unable to give optimal performance, they outperform the best existing approach based on the BDI model. (This, of course, is not surprising, but it is pleasing that the approximations work reasonably well—it suggests that they may be a reasonable proxy for the unobtainable gold standard for this size of problem).

5.2. A Larger Example

As we have seen in the results so far, the MDP approximations consistently make better decisions than the BDI agent when the size of the world is small. However, observing the way in which they are built, we can see that the quality of these approximations can only deteriorate as the size of the world becomes larger. The final results we will report in this work show this for a TILEWORLD of size 20×20.

The agents in this case are based on the same models as in the previous experiments, except that they were adapted to occupy larger worlds. This is where the heuristic approach embodied by the BDI model pays off. Because it is not trying to be exact, the BDI agent is able to see every square in the grid—the model scales with no problems. However, this is not true for the MDP agent. Constrained to be an exact model, the largest grid we can solve for the MDP agent with a complete model is the same 4×4 as before. As a result, the best we can do in this case is to give the MDP agent is to implement it as a 7×7 "moving window" (with the 7×7 made up of four overlapping 4×4 grids), thus limiting it to being able to see just 3 cells in all directions. Alternatively we can use the other approximations to the 7×7 model that we discussed above.

Figure 8 shows a summary of the results for the 20×20 TILEWORLD. We have selected the BDI planning cost 2 curve as a representative of the heuristic model, and for the MDP model used the "moving window" based on the 4 overlapping 4×4 grids and the 3 hole reduced state space approximations. As the figure clearly shows, the BDI model now outperforms these MDP approximations. For low values of dynamism, the BDI model is far superior and is never worse than the MDP approximations. As dynamism reaches extreme values, the effectiveness of the agents becomes close; this is due to the online planning cost that the BDI agent must pay and the fact that the holes are now disappearing before the agent can reach them. Finally, we note that the two MDP approximations are almost equivalent in this case. This is due to the fact that the differences in their implementations make little impact in a large environment.

The important thing to note here is that the 20×20 TILEWORLD is enormously larger than anything that can be solved by the even the most powerful exact and approximate techniques for solving MDPs. We believe that this shows that the BDI model has a place in the armory of every agent designer since it makes it possible to build agents that perform better than those based on an optimal but insoluble, and thus necessarily badly approximated, model. Even existing approximations for solving MDPs will fail before some large problems that are soluble using the BDI model.

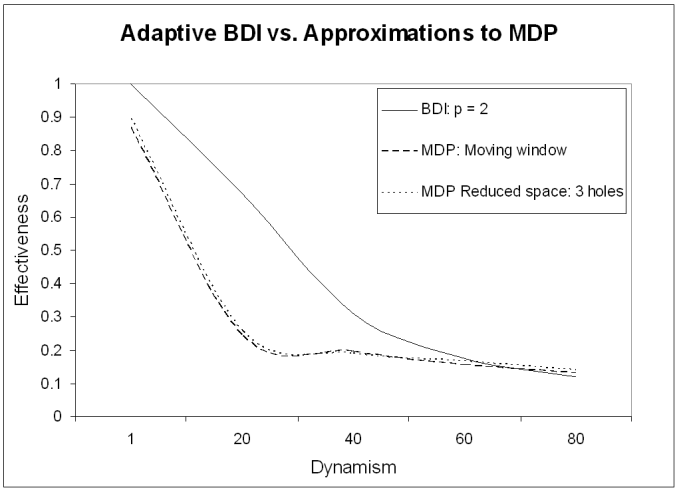

Figure 8. Comparing the BDI model with the best MDP approximation for a 20×20 TILEWORLD

6. Conclusions and future work

Though this work is preliminary, one can tentatively conclude that there is a tradeoff between the MDP and BDI models. For small worlds, the MDP approach—effectively giving an agent a complete conditional plan—outperforms the BDI approach even if the agent has a full set of linear plans that it switches between (which is the case we model when $p = 0$). While not surprising *per se*, since we are comparing a heuristic approach with decision-theoretic optimality, it is interesting that even an approximation to the MDP solution succeeds in this way. However, when the size of the world becomes larger, the performance of the MDP approximations becomes poorer, and the heuristic nature of the BDI planner is able to deal with the increase in world size without difficulty.

There are a number of directions in which we are taking this work. First of all, we need to carry out further experiments to flesh out the results we have presented here. Second, we acknowledge that the approximations we are using are somewhat crude, and it will be interesting to see if the results for more sophisticated techniques, like PolCA [12] are qualitatively the same. Of course, we will still not be able to solve the larger examples directly, but we may be able to take an exact solution to a 6×6 TILEWORLD and use that to create a better moving window solution to a 20×20 world. It seems unarguable that the solutions we obtain will be qualitatively the same for small TILEWORLDs, but the situation for larger worlds is unclear (especially since the BDI model can handle much larger worlds as well).

Finally, we are interested in establishing the formal correspondance between MDP and BDI models. We have established that there is a tradeoff between them, but in order to make a choice of which to use (especially when the choice is between approximation to MDP and BDI for a moderately sized problem) we would like to know more exactly what the tradeoff is.

Acknowledgements

The work described here was partially supported by *Comisión de Investigaciones Científicas, Gobierno de la Provincia de Buenos Aires*, the *Agencia Nacional de Promoción Científica y Tecnológica* (PICT 13096), and PSC CUNY 65395-00 34. We are very grateful to Martijn Schut, who gave us the source code for his implementation of the TILEWORLD and his BDI agent model, and to The Agent Oriented Software Group for a free license for JACK Intelligent Agents.

References

[1] D. Aberdeen. A (revised) survey of approximate methods for solving partially observable markov decision processes. Technical report, National ICT Australia, Canberra, Austalia, 2003.

[2] R. Adobbati, A. N. Marshall, A. Scholer, S. Tejada, G. A. Kaminka, S. Schaffer, and C. Sollitto. Gamebots: A 3d virtual world test-bed for multi-agent research. In *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, Montreal, Canada, 2001.

[3] R. C. Arkin. *Behavior-based Robotics*. MIT Press, Cambridge, MA, 1998.

[4] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[5] Michael E. Bratman, David Israel, and Martha Pollack. Plans and resource-bounded practical reasoning. In Robert Cummins and John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

[6] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), 1986.

[7] P. Busetta, R. Ronnquist, A. Hodgson, and A. Lucas. Jack intelligent agents — components for intelligent agents in Java. White Paper, Agent Oriented Software Group, 1999.

[8] V. Frias-Martinez, M. Marcinkiewicz, S. Parsons, and E. Sklar. The metrobots four-legged league team at robocup 2003. In B. Browning and D. Polani, editors, *Proceedings of RoboCup 2003*, page (to appear). 2003.

[9] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, 1987.

[10] D. N. Kinny and M. P. Georgeff. Commitment and effectiveness of situated agents. In Barbara Grosz and John Mylopoulos, editors, *Proc of the Twelfth International Joint Conference on Artificial Intelligence*, pages 82–88, San Mateo, California, 1991. Morgan Kaufmann.

[11] Michael Lederman Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Department of Computer Science, Brown University, Providence, RI, February 1996.

[12] J. Pineau, G. Gordon, and S. Thrun. Policy-contingent abstraction for robust robot control. In *Proceedings of the Conference on Uncertainty in AI*, Acapulco, Mexico, 2003.

[13] Martha Pollack and Marc Ringuette. Introducing the tileworld: experimentally evaluating agent architectures. In Thomas Dietterich and William Swartout, editors, *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 183–189, Menlo Park, CA, 1990. American Association for Artificial Intelligence, AAAI Press.

[14] M. L. Puterman. *Markov decision processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, 1994.

[15] H. Raiffa. *Decision Analysis: Introductory Lectures on Choices under Uncertainty*. Addison-Wesley, 1968.

[16] J. Rennie and A. K. McCallum. Using reinforcement learning to spider the Web efficiently. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning*, pages 335–343, San Francisco, CA, 1999. Morgan Kaufmann.

[17] M. C. Schut. *Intention Reconsideration*. PhD thesis, University of Liverpool, 2002.

[18] Martijn Schut and Michael Wooldridge. Intention reconsideration in complex environments. In Carles Sierra, Maria Gini, and Jeffrey S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 209–216, Barcelona, June 2000. ACM Press.

[19] Martijn Schut and Michael Wooldridge. Principles of intention reconsideration. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 340–347, Montreal, Canada, May 2001. ACM Press.

[20] Martijn Schut, Michael Wooldridge, and Simon Parsons. Reasoning about intentions in uncertain domains. In *Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Berlin, Germany, 2001. Springer Verlag.

[21] Paul Tseng. Solving H-horizon, stationary Markov decision problems in time proportional to $\log(H)$. *Operations Research Letters*, 9(5):287–297, 1990.

[22] M. Wooldridge. Intelligent Agents. In G. Weiss, editor, *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, chapter 1, pages 27–78. The MIT Press, Cambridge, Massachussetts, 1999.

[23] Mike Wooldridge and Nick Jennings. Agent theories, architectures, and languages: A survey. In Michael Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents*. Springer Verlag, Berlin, Germany, 1995.

[24] Mike Wooldridge and Simon Parsons. Intention reconsideration reconsidered. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Intelligent Agents V*, Heidelberg, Germany, 1999. Springer-Verlag.