On Partially Observable MDPs and BDI Models

Martijn Schut[1], Michael Wooldridge[2], and Simon Parsons[2,3]

[1] Department of Artificial Intelligence, Vrije Universiteit Amsterdam,
1081 HV Amsterdam, The Netherlands,
schut@cs.vu.nl
[2] Department of Computer Science, University of Liverpool,
Liverpool L69 7ZF, United Kingdom,
m.j.wooldridge@csc.liv.ac.uk
[3] Center for Coordination Science, Sloan School of Management, MIT,
Cambridge, MA 02142, USA,
sparsons@csc.liv.ac.uk

Abstract. Decision theoretic planning in ai by means of solving Partially Observable Markov decision processes (pomdps) has been shown to be both powerful and versatile. However, such approaches are computationally hard and, from a design stance, are not necessarily intuitive for conceptualising many problems. We propose a novel method for solving pomdps, which provides a designer with a more intuitive means of specifying pomdp planning problems. In particular, we investigate the relationship between pomdp planning theory and belief-desire-intention (bdi) agent theory. The idea is to view a bdi agent as a specification of an pomdp problem. This view is to be supported by a correspondence between an pomdp problem and a bdi agent. In this paper, we outline such a correspondence between pomdp and bdi by explaining how to specify one in terms of the other. Additionally, we illustrate the significance of a correspondence by showing empirically that it yields satisfying results in complex domains.

1 Introduction

Designing autonomous agents that are to operate in uncertain environments has been the focus of substantial research in various sub-areas of ai. These agents have to deal with executing actions that may not have the intended results, with environments that change while the agent is operating, and with making observations that might not be completely accurate. Much research effort has gone into specifying such agents by means of Markovian planning. In this respect, agents are implemented as solutions to Markov Decision Problems: they are, as such solutions, mappings from states to optimal actions. Although theoretically very appealing, the Markov planning framework poses some important problems when put into practice. For example, computing optimal solutions of mdps is computationally very hard. Fast close-optimal solution algorithms and various abstraction techniques have been proposed to solve this problem. We propose an alternative technique. If we are able to map mdp components to bdi components, we can use the bdi architecture to design a bounded optimal mdp agent. Then we utilise the theoretical rigorousness of the mdp framework, combined with the practical utility of the bdi framework.

M. d' Inverno et al. (Eds.): UKMAS 1996–2000, LNAI 2403, pp. 243–259, 2002.

In this paper, we investigate the correspondence between the theory of Markov decision processes for planning in partially observable stochastic domains (pomdp) and the belief-desire-intention (bdi) architecture for programming situated agents. The motivations for obtaining this correspondence are diverse. Firstly, we show that it is possible to utilise an important characteristic of intentions – the constraint of reasoning – in practice by using it to solve pomdps. Secondly, because solving pomdps is inherently intractable, our approach contributes to dealing with this intractability by utilising tractable corresponding bdi models. Thirdly, whereas pomdp models take away part of the burden of explicitly programming agents, the identification of relevant problem structure often proves to be very hard and unintuitive from a design point of view. bdi models seem to be easier to specify, and if we can establish this corespondance and so build pomdp models from bdi models, we may be able to simplify the construction of pomdp models.

This paper does not address all these issues. Here we just point out the correspondence between the bdi and pomdp models and demonstrate empirically that the performance of a bdi model approximates the performance of a discrete mdp model. Although we present a general formulation of the correspondence, the experiments are still for the specific case of mdp problems and we are currently working on pomdp experiments. This paper summarises the prerequisites for the construction of initial formal and empirical correspondences between the two models. Thus the payoff of the work presented here is in the future, but for now we have provided the first detailed comparison between the models.

The paper is structured as follows. In the following Section, we provide some background information on the bdi agent architecture and we show how to specify bdi agent programs. Section 3 presents the Markov decision framework upon which our approach builds. Section 4 explains the correspondence between the bdi architecture and partially observable mdps. In Section 5 we empirically evaluate our approach with respect to effectiveness and computational leverage. Finally, in Section 7 we present our conclusions and describe related and future work.

2 Belief-Desire-Intention Agents

The idea of applying the concepts of beliefs, desires and intentions to agents originates in the work of [4] and [9]. In this paper, we use the conceptual model of bdi agency as developed by Wooldridge and Parsons [14]. This model is shown in Figure 1. The model distinguishes three main data structures in an agent: a *belief* set, a *desire* set and an *intention* set. An agent's beliefs represent information that the agent has about its environment, and may be partial or incorrect. Desires can be seen as states of affairs that an agent ideally would want to accomplish. Intentions are those desires that an agent has committed to bringing about. The behaviour of the agent is generated by four main components: a *next-state* function, which updates the agent's beliefs on the basis of an observation made of the environment; a *deliberation* function, which constructs a set of appropriate intentions on the basis of the agent's desires, and its current beliefs and intentions; an *action* function, which selects and executes an action that ultimately satisfies one or more of the agent's intentions; and a *meta-level control* function, the sole purpose of which is to decide whether to pass control to either the deliberation or

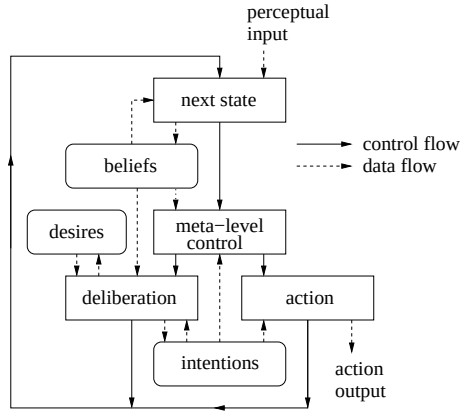

Fig. 1. An abstract bdi agent architecture.

action subsystems. On any given control cycle, an agent begins by updating its beliefs through its next-state function, and then, on the basis of its current beliefs, the meta-level control function passes control to either the deliberation function (in which case the agent expends computational resources by deliberating over its intentions), or to the action subsystem (in which case the agent acts). As a general rule of thumb, an agent's meta-level control system should pass control to the deliberation function when the agent will change intentions as a result; otherwise, the time spent deliberating is wasted.

We present a simple formal model of bdi agents. First, we have to consider that agents are situated in *environments*; an environment denotes everything that is external to the agent. Let P be a set of *propositions* denoting environment variables. In accordance with similar proposition based vector descriptions of states, we let environment states be built up of such propositions. Then E is a set of *environment states* with members $\{e, e', \ldots\}$, and $e = \{p_1, \ldots, p_n\}$, where $p_i \in P$. Let A denote the set of actions that an agent can execute. A state transition function $\tau : E \times A \rightarrow \Pi(E)$ manages the probabilistic transition of environment states, based on doing some action $a \in A$ in state $e \in E$.

The internal state of an agent consists of beliefs, desires and intentions. Let $Bel : E \rightarrow [0, 1]$, where $\sum_{e \in E} Bel(e) = 1$, denote the agent's *beliefs*: we represent what the agent believes to be true of its environment by defining a probability distribution over the possible environment states. The agent's set of *desires*, Des, is a subset of the set of environment variables: $Des \subseteq P$. Finally, we denote the set of intentions by Int. An intention denotes a number of different means to achieve a certain desire. This is represented here by letting an intention be a stack of partially instantiated plans, i.e., plans in which some variables have been instantiated (as in [9]). We assume that a plan consists of some trigger event, a context and a series of actions. The context is a series of propositions that are evaluated true (for achievement plans) or false (for maintenance plans) after executing the specified series of actions. Let the *head* of a plan be a trigger event and context. Then a plan that is intended typically contains a head that includes

some true or false belief that the agent wants to bring about. This belief literal is an environment proposition.

Note that an intention is a sequence of actions in a partially instantiated plan. This is also the key to the way that bdi approximates pomdp: a bdi agent chooses a pre-compiled plan (which is why the online computation is quick) which is nearest to being optimal (which is why we only ever approximate the optimal solution).

An internal state s is then $s = \langle Bel, Des, Int \rangle$, where $Bel : E \rightarrow [0,1]$ is a probability distribution over the agent's beliefs, $Des \subseteq P$ a set of desires and Int a set of intentions. Let S be the set of all internal states. For a state $s \in S$, we refer to the beliefs in that state as Bel_s, the desires as Des_s and to the intentions as Int_s. We use subscript S to refer to beliefs, desires and intentions for all states; for example, Bel_S refers to the beliefs for all states $s \in S$. We refer to an $i \in Int_s$ as a *background intention* of state $s \in S$. We assume that it is possible to denote values and costs of the outcomes of intentions[1]: an *intention value* $V : Int \rightarrow \mathbb{R}$ represents the value of the outcome of an intention; and *intention cost* $C : Int \rightarrow \mathbb{R}$ represents the cost of achieving the outcome of an intention. The *net value* $V_{net} : Int \rightarrow \mathbb{R}$ represents the net value of the outcome of an intention; $V_{net}(i)$, where $i \in Int$, is typically $V(i) - C(i)$. We denote the *quality* of a state by a function $Q : S \rightarrow \mathbb{R}$, which we assume to be based on the net values of the outcomes of the intentions in a state. Moreover, we assume that if $\forall s, s' \in S, \forall p \in Int_s, \forall p' \in Int_{s'}, V_{net}(p) \geq V_{net}(p')$, then $Q(s) \geq Q(s')$. In the empirical investigation discussed in this paper, we illustrate that a conversion from intention values to state qualities is feasible, though we do not explore the issue here[2]. Finally, A denotes the set of actions the agent is able to perform; with every $\alpha \in A$ we identify a set of propositions $P_\alpha \subseteq P$, which includes the propositions that change value when α is executed. (In the remainder of this paper, we label the various bdi components with label bdi.)

3 Partially Observable Markov Decision Processes

A partially observable Markov Decision Process (pomdp) can be understood as a system that at any point in time can be in any one of a number of distinct states, in which the system's state changes over time resulting from the performance of actions and in which the current state of the system cannot be determined with complete certainty [2]. pomdps satisfy the Markov assumption in that knowledge of the current state renders information about the past irrelevant to making predictions about the future [2]. In a pomdp, we represent the fact that the knowledge of the agent is not complete by defining a probability distribution over all possible states. An agent then updates this distribution when it observes its environment.

[1] We clearly distinguish intentions from their outcome states and we do not give values to intentions themselves, but rather to their outcomes. For example, when an agent *intends* to deliver coffee, an *outcome* of that intention is the state in which coffee has been delivered.

[2] Notice that this problem is the inverse of the utilitarian *lifting problem*: the problem of how to lift utilities over states to desires over sets of states. Discussing the lifting problem, and its inverse, is beyond the scope of this paper, and therefore we direct the interested reader to the work of Lang et al. [7].

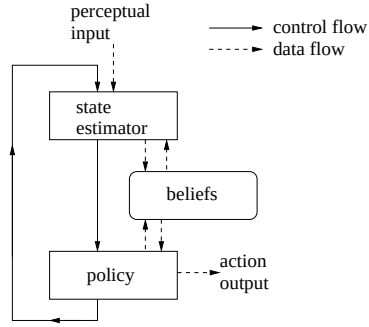

Fig. 2. Components of a pomdp agent.

Let a set of states be denoted by S and a set of actions be denoted by A. An agent might not have complete knowledge of its environment, and must thus *observe* its surroundings in order to acquire knowledge: let Ω be a finite set of observations that the agent can make of the environment. Then an *observation function* $O : S \times A \rightarrow \Pi(\Omega)$ defines a probability distribution over the set of observations; this function represents the observations an agent can make resulting from performing an action $a \in A$ in a state $s \in S$. The agent receives rewards for performing actions in certain states: this is represented by a *reward function* $R : S \times A \rightarrow \mathbb{R}$. Finally, a *state transition* function $\tau : S \times A \rightarrow \Pi(S)$ defines a probability distribution over states resulting from performing an action in a state – this enables us to model non-deterministic actions. (In the remainder of this paper, we label the pomdp components with subscript mkv[3].)

Figure 2 shows the components of a pomdp. Unlike a discrete mdp, a pomdp model includes a *state estimator* SE, which controls the belief state transitions, based on the last action, the current observation and the previous belief state. This component is not necessary in a discrete mdp, since there the agent's policy is based on external states that always accurately reflect the current state of the environment. The state estimator computes a new belief state from basic probability theory, as explained in [5]. The output of the state estimator is used in the agent's state transition function by assigning a probability of 1 to belief state b' resulting from executing action a in belief state b and making observation o if $SE(b, a, o) = b'$ and a probability of 0 otherwise.

Having defined the sets contained in a pomdp, we solve a pomdp by computing an *optimal policy*: an assignment of an action to each possible state such that the expected sum of rewards gained along the possible trajectories in the pomdp is a maximum. An mdp has either an infinite horizon, which renders the policy to be a mapping from states to actions, or a finite horizon, which makes the policy a mapping from states and time to actions. In finite horizon pomdps it thus matters *when* an action is executed. In this paper, our concern is mainly with infinite horizon mdps. Optimal policies can be computed by applying dynamic programming methods to the pomdp, breaking the

[3] Note that both the discrete mdp and continuous pomdp are Markov processes, hence the acronym mkv.

problem up into one-step decision problems using Bellman's equations [1]. The standard dynamic programming algorithms are based on backwards induction; value iteration and policy iteration are the most well known algorithms to solve pomdps. A major drawback of applying pomdps is that these kinds of algorithms tend to be highly intractable.

Traditional approaches that attempt to tackle the computational complexity of solving mdps are either aimed at reducing the state space by exploiting the space structure, e.g., by means of abstraction and aggregation; or the focus is on designing algorithms that are faster than value and policy iteration. Research on computing optimal policies for pomdps have focused on problems with finite horizons. For some finite horizon problems, for example, the Tiger problem in [5], the optimal policy turns out to be an infinite horizon policy, i.e., a policy that does not depend on time. Computing infinite horizon policies for pomdps turns out to be extremely hard[4].

4 Correspondence between bdi and pomdp

The belief-desire-intention model can be used to specify partially observable Markov decision processes. In this Section we show how bdi models correspond to pomdp models and what this means in terms of offline and online computation time and effectiveness.

The objective of our approach is to demonstrate that it is possible to identify a correspondence between the structure of pomdps and structure of an existing agent model, in this case the bdi model. The main motivation behind our approach is the fact that, viewed at its most abstract, both the pomdp and bdi models ultimately model decision making by mapping perceptual inputs to actions; all other components in the pomdp model and the bdi model are there *in the service* of this abstract decision making function. This can be easily observed by comparing Figures 1 and 2.

In this Section, we first explain what the problem of finding correspondence encompasses, in particular in relation to the bdi agent model. We do this by letting both the pomdp model and bdi model be instantiations of an abstract generic agent function. Secondly, we explain the correspondence in computing agent runs in both models and actually running the models.

Agent Functions

Both the pomdp and bdi model can be represented on some level of abstraction, so that they both correspond to some abstract agent function $ag : S \rightarrow A$ that maps agent states to agent actions. This agent function can then be implemented by either a pomdp or bdi model. As shown above, we define a pomdp as a tuple $\langle S_{\text{mkv}}, A_{\text{mkv}}, \Omega, R, \tau_{\text{mkv}} \rangle$. A bdi model is defined as $\langle S_{\text{bdi}}, A_{\text{bdi}}, Bel, Des, Int \rangle$ with the bdi control functions as described earlier. Let the implementation of ag by a pomdp be denoted by ag_{mkv} and the bdi implementation of ag by ag_{bdi}. We show here how the components of ag_{mkv} and ag_{bdi} map into each other.

[4] Algorithms for solving finite horizon pomdps utilise the fact that in this case the value function is piecewise-linear and convex. However, in an infinite pomdp, the value function is convex, but not necessarily piecewise-linear.

Firstly, we identify the following obvious mappings between the bdi and pomdp models:

- *Actions* – The sets of external actions that an agent has at its disposal in the bdi and pomdp model are identical: $A_{\text{mkv}} \equiv A_{\text{bdi}}$. In the bdi model these actions can be collected and represented more expressively through the concept of plans (or intentions).
- *States* – Because it is assumed that the environment is only partially observable in both the bdi and pomdp model, agent states are belief states rather than environment states. The sets of belief states are identical: $S_{\text{mkv}} \equiv Bel_S$, where Bel_S refers to the bdi set of beliefs. Thus the set of pomdp states is identical to the set of bdi states when we exclude the desires and intentions in every bdi state. But because desires and intentions are internal data structures, this issue is not a major obstacle to form state correspondence and thus we let $S_{\text{mkv}} \equiv S_{\text{bdi}}$.
- *Transition* – The external transition functions, as defined over the environment states and external actions, are identical, because such functions are external to the agent: $\tau_{\text{mkv}} \equiv \tau_{\text{bdi}}$. The internal transition functions are identical as well: in the bdi model this is the next state function and in the pomdp model the state estimator controls internal transitions: $nextState \equiv SE$. As such, the bdi next state function can be implemented as a pomdp state estimator.

This leaves us with some mappings between components that are somewhat more convoluted: rewards on the pomdp side and desires and intentions on the bdi side. We relate desires to rewards and intentions to a combination of rewards and actions. As mentioned above, *rewards* are received for executing some action in a particular state and are thus defined over action and state combinations. *Desires* are states of affairs that the agent wants to bring about, and thus define some kind of ordering over the set of states. Currently, we are not concerned with how this ordering is exactly realised; in this paper, we define desires simply as a subset of the environment propositions. But, for example in [7], this ordering is based on the individual utilities of the environment propositions. Let $D : S \rightarrow \mathbb{R}$ be a function that represents the ordering of desires over the state space. On the pomdp side, we can distill the rewards in such a way that they are defined only over states[5]. Again, we do not prescribe how this should be done, but merely utilise the fact that it can be done. An example conversion, that works for our experimental testbed as described below, would be to define the *worth* of a state, denoted by $W : S \rightarrow \mathbb{R}$, as the maximum reward of all actions that can be executed in a state $s \in S$:

$$W(s) = \max_{a \in A_s} R(s, a),$$

where $A_s \in A$ denotes the set of all actions that can be executed in s. Then this concept of state worth corresponds to the ordering on desires: $W \equiv D$. From this we conclude that rewards correspond with desires.

Finally, we identify the concept of *intentions* with a combination of rewards and actions. An intention is a stack of partially instantiated plans: it specifies a sequence

[5] We claim that this conversion can be done in general without any loss of information, but cannot currently support this claim with conclusive proof. Research is ongoing on this issue.

of actions which, when executed, fulfills some desire of the agent. There is thus an action as well as a desire aspect to intentions. First, we explore the desires part of this plan definition of intentions. The set of desires is a subset of the set of environment propositions. As mentioned above, the head of an intended plan contains an environment proposition that the agent wants to be either true or false: this is thus a desire. In terms of a pomdp, this first part of intentions relates to rewards, because desires correspond to pomdp rewards, as described previously.

The second part of intentions concerns the sequence of actions. In terms of a pomdp, this clustering of actions into intentions is some form of *action abstraction*. It is this abstraction which gives bdi approach its computational edge, but also means it may only approximate optimal actions. Because pomdps generate complicated plans progressively by mere execution of single actions rather than to build and – partially or completely – execute complex plans, we cannot simply utilise this similarity as a proper correspondence. However, these plans are not ordinary plans, but organised in intentions. Intentions have particular characteristics, under which most importantly representing a number of different means to fulfill the same desire. Based on the differences between traditional plans and the characteristics of intentions, we claim a valid correspondence between this part of intentions and actions. We further have to distinguish between deterministic and stochastic actions. An optimal agent decides the stability of its intentions based on the degree of determinism of its actions, the degree of observability of the environment, the rate of change of the environment [12] and the agent's own changing preferences. As for deterministic actions, this leaves intention stability to depend on the other three factors. However, we are concerned with stochastic domains and this renders the agent's actions stochastic. In that case, we have to take this non-determinism into account by expressing it on the level of intentions rather than the level of actions.

To summarise, we have discussed the following correspondences between the bdi and pomdp models. Firstly, the action spaces and transition functions (both internal and external) of the models are identical. Secondly, the pomdp state space and the belief parts of internal bdi states are the same. Thirdly, the bdi desires correspond to the rewards. And finally, bdi intentions correspond to a combination of pomdp rewards and actions.

Agent Runs

Assuming that the above correspondences are valid, we can identify the correspondence between running ag_{mkv} and ag_{bdi}. In both models, a run is a sequence of states connected by the actions executed by the agent. The method of choosing such a run in the pomdp model is based on the policy, as computed when solving the pomdp. In the bdi model, such a run defines the optimality of the agent; an optimal agent generates an optimal run. Computing runs for particular implementations of both models involves an *offline* and *online* component. Offline computation takes place outside of the environment in which the agent is to be situated and thus before executing actions. This computation results in an optimal policy for the pomdp case, or in an agent program in the bdi case. The online computation involves executing the policy or program. In case of a policy, this boils down to looking up the most believed state given the observations in the policy and executing the optimal action for that state. In case of a program, the online computation concerns the whole process that happens between receiving perceptual input and executing action

output. Thus ag_{mkv} and ag_{bdi} correspond to each other, though we have to be aware of potential differences in online and offline computation times.

An important issue to keep in mind is the Markov property: it is not necessary to maintain an action history. Obviously, ag_{mkv} obeys this property. However, in the bdi model it often happens that selection of an optimal action is based on the history leading up to the current state. Similar to approaches that turn non-Markovian processes into Markovian ones, we assume for now that the bdi history is contained in the current state. Since we have shown above that the belief states correspond and a belief state in ag_{mkv} is updated using the previous belief state, we can safely state that we can make ag_{bdi} obey the Markov property.

Finally, we mention the role of observability in the correspondence specification. In ag_{mkv}, observations are not only used to contain *physical* types of observations, but *informational* as well. In this way, it is possible to capture notions of resource-bounded information gathering or obtaining the value of information. We can use this correspondence in ag_{bdi}. An important issue when designing situated agents concerns the dynamism of the agent's environment, i.e., the world changes while the agent executes its policy. We can use the concept of observability to represent dynamism. In this way, we move to another type of pomdp in which there is no uncertainty about the current environment state, but there is uncertainty over state transitions and non-determinism of actions. We return to this issue in Section 7.

5 Empirical Validation

In this Section we apply our model in the Tileworld testbed [8]. The results of our experiments support the suggested benefit of our model in two ways. Firstly, we demonstrate that the increase in effectiveness of a mdp agent over a bdi agent is small. Secondly, we show that when the problem size grows, one cannot compute an mdp solution any more. This is mainly due to the intractability of solving complex mdps. This issue is discussed and illustrated below by indications of some offline computation times for solving an mdp representing the Tileworld.

The Tileworld [8] is a grid environment on which there are agents and holes. An agent can move up, down, left, right and diagonally. Holes have to be visited by the agent in order for it to gain rewards. The Tileworld starts in some randomly generated world state and changes over time with the appearance and disappearance of holes according to some fixed probability distributions. An agent moves about the grid one step at a time.

The Tileworld testbed is easily represented as an mdp. Let L denote the set of locations, i.e., $L = \{i : 1 \leq i \leq n\}$ represents the mutually disjoint locations, where n denotes the size of the grid. A proposition p_i then denotes the presence ($p_i = 1$) or absence ($p_i = 0$) of a hole at location i. An environment state is a pair $\langle \{p_i, \ldots, p_n\}, m \rangle$, where $\{p_i, \ldots, p_n\}$ are the propositions representing the holes in the grid, and $m \in L$ is the current location of the agent.

We computed the optimal infinite horizon mdp policy, using value iteration, for an agent situated in a Tileworld. An environment state in this mdp is a combination of the current location of the agent and the locations of present holes; the possible actions are up, right, down, left and stay; an action succeeds with probability 0.9 – failure means

Table 1. Offline computation times of running a value iteration algorithm for an mdp specification of the Tileworld with discount rate = 0.9 (measured performance of Java2 on PentiumIII-500Mhz, 128MB RAM).

Tileworld size (length $\times$ width)	$\lvert S_{\text{mkv}} \rvert$	# Iterations before optimal	Iteration duration (msec)
3×3	81	5	84
4×4	256	7	840
5×5	625	9	5,200
6×6	1296	11	23,750
7×7	2401	13	92,700
8×8	4096	15	250,000

that another action is chosen with equiprobability. Because the Tileworld is dynamic, we have to take into account that every cell is either occupied by a hole or not. Combining this fact with the current location of the agent, makes the state space of size $2^n \times n$. In order to render the necessary computations in some degree feasible, we *abstracted* the Tileworld state space. In the Tileworld domain, we abstract the state space by letting an environment state e be a pair $\langle p_1, p_2 \rangle$, where p_1 refers to the location of the hole which is currently closest to the agent, and p_2 refers to the current location of the agent. We deem this knowledge sufficient for the agent to choose an appropriate action. This abstraction means that the size of the state space is now reduced to n^4.

We plotted some statistics of these mdp solution computations for a number of Tileworlds of different size in Table 1 by means of value iteration. The results merely illustrate that even for a simplistic application such as the Tileworld, the offline computation times are exorbitant[6]. Although this approach renders the online computation times negligible, it is clearly not a realistic method for the design of agents in more complex settings. Moreover, as we keep increasing the size of the Tileworld, at some point it becomes impossible to compute mdp solutions (simply because of the intractability of solving mdps). From this point onwards, it pays off for certain to use a bdi approach – even if it only gives marginal results. (Currently, research is ongoing on exactly where this point is for the Tileworld testbed and how well the bdi approach performs from that point onwards.)

Whereas obtaining an optimal mdp policy in the Tileworld is computationally hard, we observe that this optimal policy is a simple fixed control strategy: the agent executes the action with highest success probability that brings it closer to the nearest hole. This strategy is easily implemented and we have done so. The effectiveness of this strategy is shown in Figure 3.

[6] We mention explicitly that these results have *only* been inserted for illustrative purposes. We are aware that performance can be increased dramatically by choosing a more efficient algorithm or even a faster programming language or machine. However, this does not refute our claim that computation times are unacceptable for such a simple domain as the Tileworld and will become impossibly long for sufficiently large Tileworld scenarios.

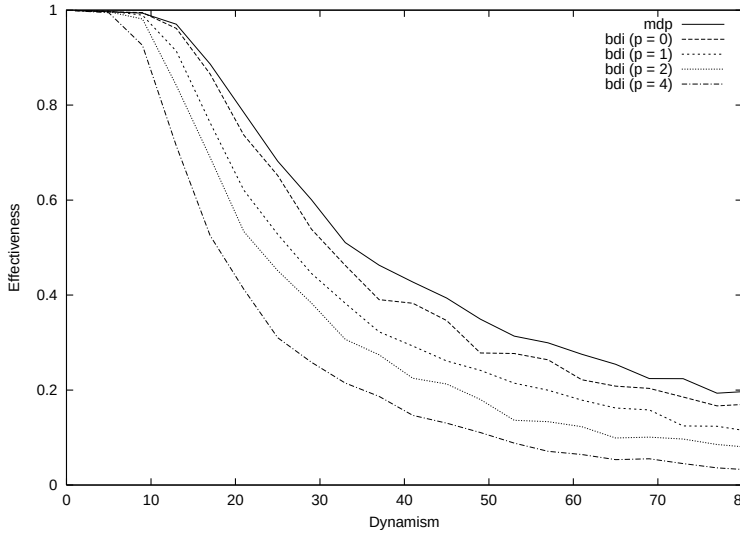

Fig. 3. Overall effectiveness of an mdp and bdi agent. Effectiveness is measured as the result of a varying degree of dynamism of the world. The four curves show the effectiveness for the bdi agent at planning costs (denoted by p) from 0 to 4.

In the experiments, the Tileworld has dimensions 20×20, thus there are 400 unique locations ($n = 400$). Environments were varied by changing the degree of dynamism (γ). Dynamism is denoted by an integer in the range 1 to 80, representing the ratio between the world clock rate and the agent clock rate. If $\gamma = 1$, then the world executes one cycle for every cycle executed by the agent and the agent's information is guaranteed to be up to date; if $\gamma > 1$ then the information the agent has about its environment may not necessarily be up to date when it carries out an action. (In the experiments in this paper we assume the environment is fully observable, i.e., the agent can update its information at every cycle of its own clock.) The *planning cost* p represents the time cost of planning, i.e., the number of time-steps required to form a plan. The *effectiveness* ϵ of an agent is the ratio of the actual score achieved by the agent to the score that could in principle have been achieved.

We conducted a similar series of experiments with the bdi agent, based on the conceptual bdi architecture as explained in Section 2. The implemented bdi architecture is described in [12]. This bdi agent adopts single intentions to visit a particular hole, constructs a plan, consisting of move actions, to achieve that intention – a path to the hole – and sequentially executes actions of the adopted plan. An intention value corresponds to the reward received by the agent for reaching a hole, and an intention cost is the distance between the current location of the agent and the location that the agent intends to reach. The meta-level control function determines the *stability* of an adopted plan. The stability is computed based on a discrete deliberation scheduling method [10]. This method determines the efficient trade off between continuing to execute the current plan

or to spend computational resources on adopting a new plan[7]. Deciding this trade off is based on knowledge of the probability distributions controlling when holes appear and disappear. The results of the series of experiments with a bdi agent are shown in Figure 3 (for planning cost $p = 0, 1, 2, 4$) in comparison with an mdp agent.

In Figure 4 the results of the bdi experiments are shown in comparison with a cautious and bold agent. A *cautious* agent reconsiders its intentions at every possible opportunity whereas a *bold* agent does not reconsider until it has fully executed its current plan. We have investigated the relationship between the reconsideration rate and various properties of an agent's environment in [11]. The results of this investigation led us to undertake further research on the problem of adaptive reconsideration, hence the bdi agent based on discrete deliberation scheduling.

We conclude this Section with a short analysis of the demonstrated results. A more in depth analysis of the bdi experiments are described in [12]. Firstly, we observe that the effectiveness of both agents decreases as the dynamism of the world increases and the bdi agent's effectiveness decreases as the cost of planning cost increases (the cost of the online part of the computation). The planning cost is a time cost, since it denotes the number of time-steps required to construct a plan.

Secondly, the most important observation we make from comparing the graphs in Figure 3 to each other is that the bdi effectiveness curve clearly approximates the mdp effectiveness curve, assuming that the planning cost is small enough. We base this conclusion on matching the mdp agent's effectiveness curve to the bdi agent's effectiveness curve for $p = 0$. This suggests that the bdi approach might be viewed as an approximation to the mdp approach, and one which is tractable, but shifts the computational burden from offline to online. As this burden increases (p gets larger), the quality of the approximation decreases.

Thirdly, the bdi approach can handle Tileworld examples which are beyond the scope of the mdp approach, e.g., a 40×40 Tileworld. As the size of the Tileworld increases, Table 1 shows that mdp computation times increase rapidly. One may safely assume that from some point onwards, computing an optimal mdp policy becomes unfeasible and even impossible. As mentioned above, it is necessary to investigate where this point is and how the bdi approach performs from that point onwards. For this, we need to know how well bdi methods scale and this issue is currently under investigation.

Finally, we comment on the performance of the mdp agent in this real-time domain. We observe that although every action is chosen optimally by the mdp agent, the overall effectiveness of the agent is not a maximum. For example, in the Tileworld domain, we observe that for environments with a dynamism that is more than 6 ($\gamma > 6$), the effectiveness of the mdp agent is less than 1 ($\epsilon < 1$). The reason for this is that the frequency with which holes appear and disappear is too high for the agent to get to one of those that appear even when choosing the decision-theoretically optimal actions. Additionally, the fact that an agent cannot precisely *anticipate* future events – the appearance and disappearance of holes – lowers the effectiveness of the agent. One example of this is the appearance of an hole exactly at the same time that the agent moves away from that location. With the benefit of hindsight, the agent would have done better to have

[7] In bdi terminology this decision making function is better known as an *intention reconsideration* function.

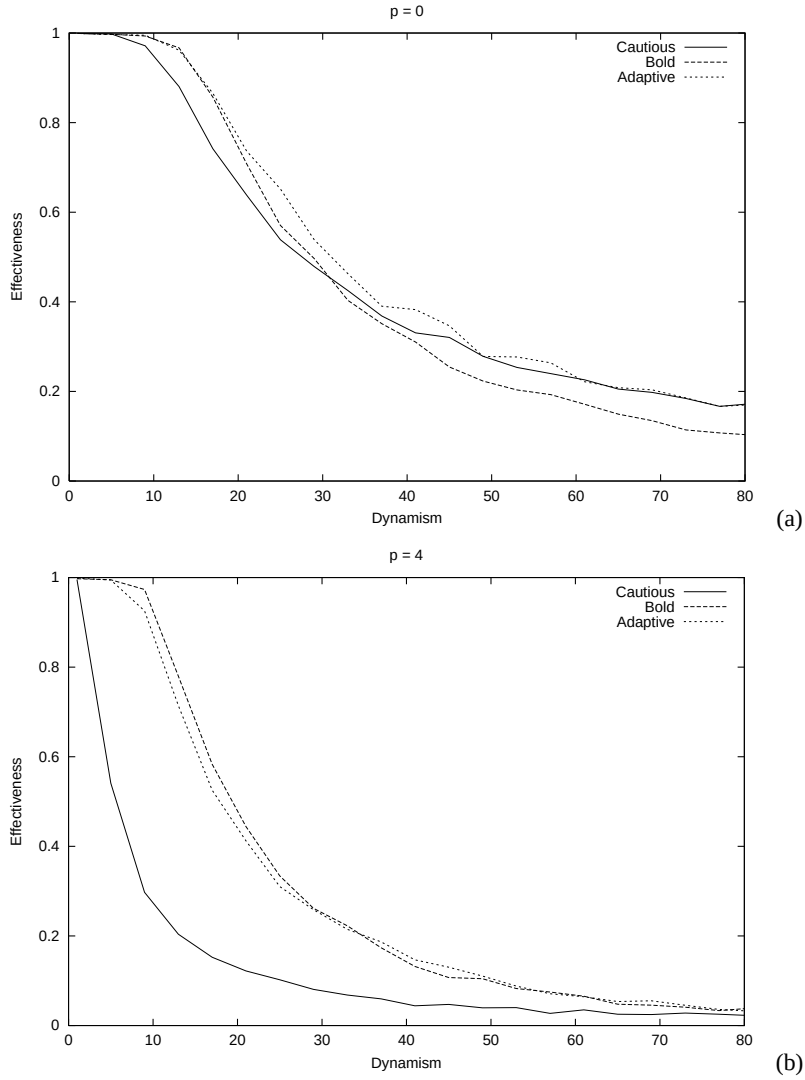

Fig. 4. Performance of a cautious, bold and adaptive agent. Effectiveness is measured as a result of a varying degree of dynamism of the world at planning costs (denoted by p) p = 0 in (a) and p = 4 in (b). The adaptive agent is a bdi agent based on discrete deliberation scheduling. (From [12])

stayed there instead of moving away. However, from the viewpoint of the agent, these are merely unlucky situations from which it is hard to escape in realistic domains. These two issues illustrate that choosing optimal actions individually does not guarantee overall optimal performance.

6 Related Work

The research described in this paper relates to a number of different research areas that we briefly describe in this Section. Firstly, we describe research by Kinny and Georgeff on which the experimental methodology in this paper is based. Secondly, we briefly discuss the issue of computational intractability in solving pomdps. Finally, we discuss work by Boutilier, which focuses on the relationship between agents and Markovian planning.

The experimental methodology as used in our investigation is based on the work of Kinny and Georgeff [6]. Their work includes an experimental program, based on Pollack's Tileworld, that aims to investigate how commitment to goals contributes to the effective behaviour of situated agents. This research is part of a more general investigation into the reactive meta-level control of deliberation for resource-bounded agents situated in dynamic domains. Kinny and Georgeff show that in dynamic environments different meta-level control strategies achieve a different effectiveness. The empirical results as obtained by Kinny and Georgeff emphasise the importance of meta-level control, but closely relate to the Tileworld domain.

We have extended the investigation of Kinny and Georgeff in two ways[8] as described in [11, 12]. Firstly, we considered partially observable and non-deterministic domains for the investigation of effectiveness of situated agents. This enabled us to clearly identify a relationship between the environment (in terms of dynamism, observability and determinism) and the deliberation control strategy as used by the agent. Secondly, we aimed to develop domain independent deliberation control strategies to be applied in a more general context then only the Tileworld testbed. As to the latter, we developed the adaptive bdi agent as described above, based on the decision-theoretic concept of deliberation scheduling. In this type of agent, the control strategy (or: reconsideration policy) determines an efficient trade off between acting and deliberating. We developed an additional decision-theoretic agent, in which the control strategy is an mdp policy that lets the agent either act or deliberate at any moment in time[9] [13]. This work illustrates the close relationship between the bdi agent architecture and the pomdp framework, which we have further worked out in this paper.

The problem of computational intractability of solution algorithms for pomdps has received much research attention (summarised in [2]). The main focus of many of these investigations has been on factorisation, abstraction and aggregation techniques to reduce the state space or action space. Probabilistic strips operators and influence diagrams are such techniques that can be used to factor the state space of Markov problems. Although these methods have been developed without the pomdp framework directly in mind, they have proven successful in factoring pomdp state spaces and consequently rendering computation times feasible. As such, our proposal for using the bdi agent architecture to solve pomdps can be considered a similar effort. However, more than solely reducing the state space or action space, the bdi architecture includes techniques to direct reasoning while solving a pomdp. This is a potential important benefit over other methods.

[8] This work has been presented at the UKMAS workshops 2000 and 2001, respectively.

[9] Since this agent suffers from the same intractability of solving pomdps, as mentioned above, we decided to compare this paper's mdp agent with the deliberation scheduling agent.

The pomdp planning framework has for long been brought into relation with agent based architectures. As mentioned in the paragraph above, several techniques from planning under uncertainty have been successfully applied in a pomdp setting as well as in the agent based research. Recently, the pomdp planning framework is being applied in multi-agent settings[10], in which either a pomdp problem is distributed among several agents or every agent is represented as a pomdp. This new development brings, as any, novel problems with it, but the multi-agent research area can contribute much to better solve existing pomdp problems. Work in this area is relevant to the research described in this paper, since it illustrates the importance and suggested benefit of combining pomdp planning and agent-based systems.

Finally, we point out research by Boutilier et al. [3] which integrates Markov decision processes with Golog, a high level programming language with a situation calculus semantics. Our model distinguishes from this work in the way that our method views the programming and planning approaches as distinctive alternatives for each other, whereas the work in [3] views them as complementary processes. Golog can be understood as an agent specification language and as such can replace the bdi part of our approach. This replacement is an interesting further extension in order to investigate the behaviour of our model with respect to other correspondence specifications.

7 Discussion

In this paper we presented a preliminary analysis of the correspondence between the theory of Markov decision processes for planning in partially observable domains and the belief-desire-intention agent architecture. The main contributions of integrating these two models are as follows: it would explain the existence of a correspondence between the pomdp and bdi models, it would demonstrate how intentions contribute to efficiently solving pomdps, and it would provide an intuitive method to specify pomdps by using bdi models. We have not addressed all these issues in this paper, and, as described below, leave further elaboration of non-addressed issues to future work.

Our research is centered around the hypothesis that bdi can still be used when mdp is intractable. The results in this paper give reasonable support to suppose that this hypothesis is true. Further support must be gathered through more rigorous theoretical and empirical investigation as initiated above. Supposing the hypothesis is correct, one concrete issue to address is to find the point at which it becomes impossible to compute mdp solutions, but where bdi models still give reasonable performance.

The main contributions of this paper are to point out the correspondence between the bdi and pomdp model and to demonstrate empirically that the performance of a bdi model approximates the effectiveness of a pomdp model. Exactly how good this approximation is depends on the time cost of planning in the bdi model, as we have shown in this paper. Although the analysis and formalisation of our approach in this paper are preliminary, the results of our experimental validation are promising as such that further research is necessary to explore our findings in more detail.

[10] The application of the pomdp framework in multi-agent systems was addressed by Boutilier in the keynote talk of UKMAS 2000.

The conclusions we derived from the Tileworld experiments are as follows. Firstly, our findings confirm results as obtained earlier in similar experiments. Secondly, the bdi model approximates the mdp model in terms of effectiveness. Thirdly, we claim that on the basis of our results, bdi can deal with problems that are beyond the mdp approach. Finally, we remark that the optimality of mdp solutions is only relevant with respect to individual actions, not necessarily regarding overall optimal performance. To this extent, the bdi approach might approximate the pomdp approach, where the computational burden has been shifted from offline to online. In the testbed used in this paper, the bdi approach can handle problems which are beyond the scope of an mdp approach. We propose future research to investigate the behaviour of performance with respect to balancing offline and online computation.

Our method is to be used for the design of autonomous agents that will operate in uncertain environments. We express this uncertainty by measurements of: *dynamism*, the rate of change of the environment, independent of the activities of the agent; *observability*, the extent to which the agent has access to the current state of the environment; and *determinism*, the degree of predictability of the system behaviour for identical system have inputs.

Exploration of future research paths from here is interesting from a number of different viewpoints. Firstly, as mentioned above, we intend to conduct further investigation of the formal analysis of our approach. Such research will give more insight into the computational efficiency of our method compared to traditional pomdp solution algorithms. The issue of balancing offline and online computation is a serious consideration for design. Through research on how these two different types of computation contribute to the computational cost of our model and under which circumstances, we hope to eventually automate balancing offline and online computation.

Secondly, we have undertaken preliminary research into the potential benefit of using the notion of intentions in solving pomdps. Previously, pomdp researchers have combined single actions into plans (called options or macro-actions) as a type of action abstraction. We have used intentions to cover this notion of plans. The added benefit of intentions over options is that, by definition, intentions direct and constrain future reasoning. As such, intentions are a very natural way for abstracting the action space.

Finally, our experimental validation can be extended in different ways. We are currently working on the implementation of our model in a more realistic type of testbed, robot navigation, to demonstrate wider model applicability. Besides this, we are investigating the implementation of observability as means of resource-bounded information gathering, i.e., acquiring value of information, in the Tileworld. For this, it is necessary to have solution algorithms for infinite horizon pomdps, and these algorithms are currently, to our best knowledge, not available.

References

1. R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
2. C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research*, pages 1–94, 1999.

3. C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00)*, pages 355–362, Menlo Park, CA, 2000.
4. M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.
5. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
6. D. Kinny and M. Georgeff. Commitment and effectiveness of situated agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 82–88, Sydney, Australia, 1991.
7. J. Lang, L. v. d. Torre, and E. Weydert. Utilitarian desires. *Journal of Autonomous Agents and Multi-Agent Systems*, 2002. To appear.
8. M. E. Pollack and M. Ringuette. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 183–189, Boston, MA, 1990.
9. A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 439–449, 1992.
10. S. Russell and E. Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1-3):361–395, 1991.
11. M. C. Schut and M. Wooldridge. Intention reconsideration in complex environments. In M. Gini and J. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 209–216, Barcelona, Spain, 2000.
12. M. C. Schut and M. Wooldridge. Principles of intention reconsideration. In E. Andre and S. Sen, editors, *Proceedings of the Fifth International Conference on Autonomous Agents (Agents 2001)*, Montreal, Canada, 2001.
13. M. C. Schut, M. Wooldridge, and S. Parsons. Reasoning about intentions in uncertain domains. In D. Dubois and H. Prade, editors, *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Toulouse, France, 2001.
14. M. Wooldridge and S. D. Parsons. Intention reconsideration reconsidered. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V (LNAI Volume 1555)*, pages 63–80. Springer-Verlag: Berlin, Germany, 1999.