On the relationship between MDPs and the BDI architecture

Gerardo I. Simari
Department of Computer Science
University of Maryland College Park
College Park, MD 20742
gisimari@cs.umd.edu

Simon Parsons
Dept of Computer & Information Science
Brooklyn College, City University of New York
Brooklyn, NY 11210 USA
parsons@sci.brooklyn.cuny.edu

ABSTRACT

In this paper we describe the initial results of an investigation into the relationship between Markov Decision Processes (MDPs) and Belief-Desire-Intention (BDI) architectures. While these approaches look rather different, and have at times been seen as alternatives, we show that they can be related to one another quite easily. In particular, we show how to map intentions in the BDI architecture to policies in an MDP and vice-versa. In both cases, we derive both theoretical and related algorithmic mappings. While the mappings that we obtain are of theoretical rather than practical value, we describe how they can be extended to provide mappings that are useful in practice.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*.

General Terms

Theory, Design

Keywords

Markov Decision Process, Policy, Intention.

1. INTRODUCTION

Markov decision processes (MDPs) can be regarded as an ideal approach to the implementation of intelligent agents provided that one can either assign utilities to states and probabilities to transitions between states, or that it is feasible to learn such values. Once these values are known, algorithms such as value iteration [10] will yield an MEU-optimal *policy*, that is a mapping from every state into the best action to execute in that state. Such a policy is a complete specification of what the agent should do in *every* state, based on the probable outcomes of *every* action possible in that state.

However, the very nature of these algorithms, and in particular the need to establish the outcome of every possible action in every state, means that finding policies is intractable in many practical cases. This has led to the search for approximate solutions to MDPs [1], but even state of the art approaches like those in [8] are not able to handle particularly large or complex problems.

A contrasting approach to building agents is the use of the BDI architecture [3, 12, 23]. Agents constructed in this way have a set of *belief*s about the state of the world and a set of *desire*s which, broadly speaking, identify those states of the world that the agent has as goals. From its beliefs and desires, and via a process of *deliberation*, and agent formulates one or more *intention*s. The precise semantics of an intention vary across the literature, but intentions can be broadly considered to be states that the agent has committed to bringing about. The agent then constructs a plan to achieve its intentions, typically through some form of *means-ends reasoning*, and executes it.

This is a heuristic approach, and naturally enough agents built using it are outperformed by the those programmed using MDPs when the full MDP solution is tractable [19]. However, the BDI model can scale to handle problems that are beyond the scope of a full MDP solution, and can outperform approximate MDP models [19] for some relatively small problems (such as a simplified Tileworld [9]).

Given this trade-off, we are interested in the formal relationship between the two approaches, and that is our subject in this paper. In particular we investigate the following questions:

1. Given a policy that is a solution to an MDP, how can we extract a BDI description that can be used to approximate the solution to the MDP?
2. Given a complete BDI description, how can we obtain a *policy* that an MDP-based agent can use to control its actions?

We start by briefly summarizing how actions, states, and transition functions for BDI and MDP descriptions can be related. Drawn from [16] this summary considers the formal descriptions provided by the BDI and MDP frameworks. Both descriptions consists of a state space S, a set of actions A, and a a state transition function T which depends on the current state and the action performed. In addition, an MDP description includes:

- a reward function R,
- a probability distribution P over the set of states, and
- a set of policies Π, each member of which identifies the best action to take in each state.

We will denote an MDP description as a tuple

$$\langle S, A, T, R, P, \Pi \rangle$$

An agent that uses an MDP description to decide how to act will be called an MDP agent.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

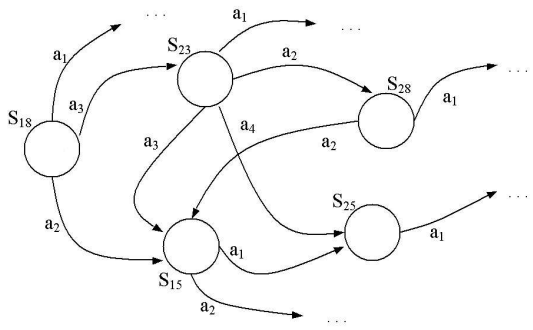

Figure 1: Part of a state-space

A BDI description consists, in addition to the state space, actions, and transition function, of:

- A set current beliefs B, desires D, and intentions I;
- a deliberation component Del; and
- a means-ends reasoning component M

We will denote a BDI description as a tuple:

$$\langle S, A, T, B, D, I, Del, M \rangle$$

An agent that uses a BDI description to decide how to act will be called a BDI agent.

For a given agent in a given environment, it is clear that the S, A and T will be the same for both the BDI and MDP descriptions. Furthermore, we will consider that P, the probability distribution in the MDP description and B in the BDI description do exactly the same thing — they identify which state in S the agent is currently in. While we deal with fully observable worlds, we can imagine both pick a single state (and we will revisit this when we talk about partially observable worlds).

With these equivalences in place, we can see that answering the two questions posed above comes down to relating rewards and policies, on the one hand, to desires, deliberation, means ends reasoning and intentions on the other hand. In fact, since rewards are merely a means of determining policies, and desires a step on the way to determine intentions, these components can be effectively ignored. Indeed, the relation we have to consider in detail is that between policies and intentions.

2. INTENTIONS, PLANS AND POLICIES

As mentioned above, the precise semantics for an intention varies across the literature. An intention is often taken to be "the agent's current focus", that a state that the agents is committed to bring about (for example by [23]). An intention can also be taken (as in PRS [5]), to be a linear plan that an agent has adopted to reach a state that the agent is committed to bring about. Here we will use the term *intention* to denote a state that an agent has committed to bring about, and use the term *intention-plan* (i-plan) to denote a sequence of actions built to reach a specific state, or, in other words, to achieve a specific intention. Since the precise sequence of actions selected will vary depending on what state the agent was in when it deliberated, an i-plan will depend on both state and intention — an agent will, in general, create different i-plans to achieve the same intention from different states, and will, in general, create different i-plans to achieve different intentions from the same state.

The reason that we are interested in i-plans is that they are key in the development of BDI agents. In practical implementations like PRS, once an agent has adopted an intention, it rifles through a pre-compiled set of plans (confusingly also called intentions, and hence i-plans in our terminology) to determine how to achieve the intention it has adopted. Building an efficient BDI system hinges on having a good plan library, and such a library will contain a set of i-plans tailored for the environment in which an agent operates. Thus, in order to answer our motivating questions, we need to relate i-plans to policies. We are interested in both establishing i-plans by solving MDPs, and using a set of i-plans to solve MDPs (hence the two questions we posed in the previous section).

We will adopt the following notation. An i-plan will be denoted by a lowercase Greek letter such as ψ, indexed $\psi^{i,s}$, where necessary, to denote the state s and intention i it relates to. I-plans are sequences of actions and ψ_i denotes the i-th action in ψ, while s_i^ψ denotes the i-th state that the agent plans to visit while executing ψ. Thus, for an i-plan ψ of length p, the agent starts at state s_0^ψ and plans on subsequently visiting states $s_1^\psi, s_2^\psi, ...s_p^\psi$. Of course, the agent may deviate from this sequence of states thanks to nondeterministic actions or changes in the environment. For example, in the case of the state-space in Figure 1, an agent in s_{18} might adopt an i-plan $\psi^{s_{18}}$:

$$s_{18} \xrightarrow{a_3} s_{23} \xrightarrow{a_4} s_{25} \xrightarrow{a_1} s_{79}$$

but find that executing a_4 in s_{23} takes it to s_{28}, where executing a_1 will take it back to s_{18} rather than on to s_{79}.

When such a deviation occurs, then execution of an i-plan will not have exactly its desired effect, though it may still help to bring the agent towards its goal. When the agent realizes that a deviation from the sequence of states laid down in the i-plan has occurred, then it needs to decide if it needs a new intention (which would typically require a new i-plan). This process of *intention reconsideration* [14, 15, 24] identifies whether or not the agent should keep its current intention, or whether it should deliberate to determine a new intention. Clearly if a new intention is adopted, the agent will need to create a new i-plan, and it may need a new i-plan even if it doesn't change intention.

Deliberation, reconsideration and the generation of i-plans can be done decision theoretically. Such approaches to intention reconsideration are explored in [17, 18]. The expected utility of an i-plan can be obtained exactly as one obtains the expected utility of a policy when solving MDP models — by establishing the expected value of executing each action in the state in which it is executed. The difference, however, is that in evaluating an i-plan in this way we only consider a single trajectory through the state-space rather than, as for policy evaluation, consider executing an action in *every* state. Typically, though, this is not the way that deliberation is done. Indeed, the BDI model was developed largely because it was felt [3] that decision-theoretic approaches were intractable (though see [11] for an example of a decision-theoretic approach to deliberation).

Whether reconsideration and deliberation are carried out decision theoretically or not, a BDI agent will have the same approach to traversing a state-space such as that in Figure 1. It will select an intention, identify an i-plan to achieve that intention, and execute that i-plan until it either it realizes that the current i-plan will no longer achieve its intention or until it realizes that the current intention is no longer achievable (or no longer the best possible intention). At such a point the agent will generate a new i-plan, or choose a new intention and generate an i-plan to achieve it, and the process will repeat.

We can contrast this process to that of an MDP agent in the same

state-space, which will always know, from its policy, what is the best action to take. The policy cannot fail to provide a suitable action — it is a universal plan [13] — and will eventually take an agent to its goal. The price paid by the MDP agent is that it has to compute the policy to begin with, a computation that is expensive compared with establishing a simple linear plan. The price paid by the BDI agent is that it has to compute what to do online (while the MDP agent can compute what to do offline) and has to suffer the sub-optimality of likely deviation from its chosen path through the state-space (though the loss of utility can be reduced by effective approaches to intention consideration [18]).

Now, despite the differences between the BDI and MDP models, we can establish a broad equivalence between them by considering how they work on the same state-space. Informally this can be done as follows. Given a policy, we have an action selection for every state, and so can establish one or more i-plans that summaries trajectories through the state-space from a given state to the goal. Thus such a policy incorporates a set of i-plans. Conversely, consider a set of i-plans, each of which captures a trajectory through a set of states. This set of i-plans identifies what action to take in all the states on the trajectory, and if one adds to that the null action for all states not on a trajectory, then one has a policy (albeit not a very good one).

The main contribution of this paper is to make this informal relationship between policies and i-plans more precise. In particular, Section 4 describes how i-plans can be used to create policies, and Section 3 details how policies can be used to derive i-plans.

3. FROM POLICIES TO I-PLANS

Assume we have a policy π that is the solution to a fully specified MDP. For now we assume that π is optimal — the results we obtain depend on this assumption — but we will see that this assumption can be dropped. From any π, it is possible to extract utility values for states that will induce π, and these can be used to establish i-plans.

For the purposes of this section, we shall consider the policy π to be the result obtained by the convergence of an algorithm such as Q-learning [20, 22] which gives a value for every state/action pair. BDI agents can also map states and actions into values. These values can be computed by assigning a value to each i-plan. Let ψ be an i-plan of length p, and ψ_i the i-th action involved in ψ. One way of assigning a value to ψ is:

$$V(\psi) = \sum_{i=1}^{p} \frac{R(s^{\psi}_{i-1}, \psi_i)}{i}$$

where $V(\psi)$ is the value of i-plan ψ, s^{ψ}_{o} is the initial state of ψ, s^{ψ}_{i+1} is the state to which the agent *expects* to arrive after executing action ψ_i, and $R(s^{\psi}_{i-1}, \psi_i)$ is the reward received for taking action ψ_i in state s^{ψ}_{i-1}. Therefore, the value assigned to an i-plan is the sum of rewards that will be achieved if all of the actions have the desired effect; the division of the reward by i in the above formula captures time discounting — rewards gained early are more valuable than the ones gained in the future.

It is important to note that this is only one of many possible ways of assigning a value to an i-plan. In general, we only require that for an i-plan ψ, every action that is added to the plan has a non-negative cost[1], and that these values depend on the rewards of the states that the agent plans to visit. For example, for a simple environment, the reward function could be defined by:

$$R(s^{\psi}_{i-1}, \psi_i) = \begin{cases} 1 & \text{if } \psi_i \text{ leads directly to a goal} \\ 0 & \text{otherwise.} \end{cases}$$

and $R(s, a) = 0$ for every other state/action pair.

We will now formally define the concepts of *i-plan* and *i-plan length*, and what it means for an i-plan to *obey* a given policy.

DEFINITION 1. *A sequence of actions $\psi = \psi_0, \psi_1, ..., \psi_p$ is called an* i-plan *if the ψ_i's ($0 \leq i \leq p$) were selected with the objective of executing them in turn in order to reach a given goal.*

DEFINITION 2. *Given an i-plan $\psi = \psi_0, \psi_1, ..., \psi_p$, we say that p is the* length *of ψ.*

DEFINITION 3. *An i-plan ψ of length p* obeys *a policy π if, and only if, $\forall i, 1 \leq i \leq p$, $\pi(s^{\psi}_{i-1}) = \psi_i$, where s^{ψ}_{i} is the state to which the agent is planning on arriving after executing action ψ_{i-1}, and s^{ψ}_{0} is the initial state.*

Definition 3 simply states that an i-plan obeys a policy if, and only if, the actions prescribed by the i-plan are the same as those prescribed by the policy through the i-plan's intermediate states. Remember that we are assuming that i-plans are *linear* plans, that is no considerations are made for unexpected outcomes of actions (which is typical for BDI implementations).

The dual of the notion of obedience is the notion of conformance:

DEFINITION 4. *A policy π* conforms *to an i-plan ψ of length p if, and only if, $\forall i, 1 \leq i \leq p$, $\pi(s^{\psi}_{i-1}) = \psi_i$ where s^{ψ}_{i} is the state that results from executing action ψ_{i-1} in state s^{ψ}_{i-1}, and s^{ψ}_{0} is the state in which the first action is executed.*

Since an i-plan is indexed by the intention that it will achieve when executed, we can extend the notions of obedience and conformance to intentions. A policy π conforms to an intention i if for all i-plans $\psi^{i,s}$, π conforms to $\psi^{i,s}$, and an intention i obeys a policy π if all i-plans $\psi^{i,s}$ obey π.

With the concepts that we have now introduced, we can state the following claim:

CLAIM 1. *Given a* BDI *agent and an* MDP *agent with an optimal policy π, if the* BDI *agent is in state s_i, then the i-plan ψ with the highest utility value will be such that ψ obeys π, starting at s_i.*

The remainder of this section explores this claim.

In general, the claim will only hold if states with the same reward are considered in the same order by both MDP and BDI approaches. Otherwise, even though the utilities are equivalent, the actions might not be exactly the same because the order in which states with the same reward are considered can affect the selection of actions. The proof of the claim can be made with respect to progressively more complex scenarios.

Our initial result relating policies to intentions and i-plans is established for the deterministic, fully accessible case (in other words the simplest):

PROPOSITION 1. *Let $\langle S, A, T, R, P, \Pi \rangle$ be an* MDP *agent, and let $\pi \in \Pi$ be a policy that is optimal under the maximum expected utility criterion. Let $\langle S, A, T, B, D, I, Del, M \rangle$ be a* BDI *agent. Let Del always select the intention with the highest reward and M select the i-plan with the highest reward. If the environment is observable and deterministic, so $\forall s \in S, \forall a \in A, |T(s, a)| = 1$, then $\forall i \in I$, $\forall s \in S$, it holds that $\psi^{i,s}$ obeys π.*

[1] This restriction, basically the same restriction that is required of the path cost function to ensure the optimality of A* search, means that the agent can follow a value gradient to the goal state.

```
Intention policyToIplan(Policy π, MDP m){
    Iplan i;
    s = getCurrentState(m);
    g = getGoalState(s,π,m);
    p = obtainPath(s,g,m);
    while not empty(p) do {
       i.addAction(p);
       p.deleteAction();
    }
    return i;
}
```

Figure 2: Pseudocode for mapping policies into intentions

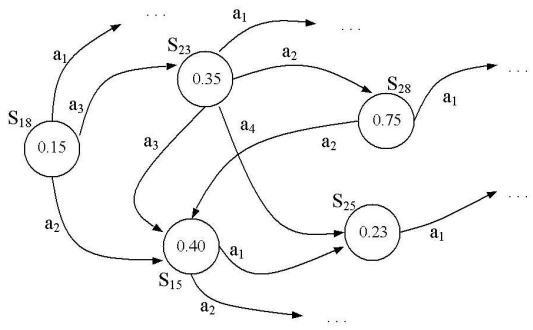

Figure 3: Part of the state space of an MDP

PROOF. *The result follows directly from the fact that we are assuming that D is optimal and that actions are completely deterministic in the environment. Because π is* MEU*-optimal, it will always select actions that take the agent to the best goal state in the best possible way. Because D picks the intention with the highest reward, and actions are deterministic, it will select the same intention/goal that π takes the agent to. Similarly since M picks the i-plan with the highest rewards it will pick an i-plan that traces the same path through the state space (from whichever state the agent is in) as π. Therefore, if we assume that states with equal rewards are considered in the same order by π and M, it is clear that the i-plans generated by M will obey π.* □

If actions are not deterministic, the utility of i-plans is not so clearly defined. Instead of a simple summation of rewards along the path of the plan, the *failure* of actions must be considered. Therefore, we must now assume that the deliberation and means-ends reasoning components are MEU-optimal as opposed to being able to pick the intention and i-plan (respectively) with the highest rewards.

PROPOSITION 2. *Let $\langle S, A, T, R, P, \Pi\rangle$ be an* MDP *agent, and let $\pi \in \Pi$ be a policy that is optimal under the maximum expected utility criterion. Let $\langle S, A, T, B, D, I, Del, M\rangle$ be a* BDI *agent where M and Del are* MEU*-optimal. If the environment is observable and non-deterministic, so $|T(s,a)| \geq 1$, then, then $\forall i \in I$ $\forall s \in S$, it holds that $\psi^{i,s}$ obeys π.*

PROOF. *Because we are now assuming that D is picking* MEU*-optimal intentions and that M is building* MEU*-optimal i-plans, the same argument as in Proposition 1 tell us that every i-plan will obey π even though actions are now non-deterministic.* □

This result also encapsulates the reason why the BDI approach struggles to generate optimal behavior. If we stick with the classic BDI model in which deliberation selects an intention and means-ends analysis then builds a plan to achieve it, in order to pick an optimal set of actions (which is what Proposition 2 is all about), the deliberation component has to be able to pick an intention that is MEU-optimal *before* means-ends analysis picks out a suitable i-plan (and thus before the agent has considered the cost and likelihood of achieving the intention it is selecting).

Now we turn to the case where the environment is not fully observable — in other words the agent doesn't know which state in S it is in, and must rely on its estimates of the current state of the environment [7]. MDP models (technically POMDP models under these conditions) handle this kind of situation by extending the notion of state. Rather than dealing (as we have until now) with a state that is some $s \in S$, the state space describing all the states of the environment, a state becomes a probability distribution across all the s. If we imagine enumerating all these possible distributions s'_i, then we can think of the policies and i-plans as being concerned with a new state space $S' = \cup_i s'_i$. If we call S' the *partially observable counterpart* to S and consider this new state space to be the space in which our BDI and MDP agents operate — so that both B and P identify some $s' \in S'$ as the current state that the agent is in — we can carry forward Proposition 2 as:

PROPOSITION 3. *Let $\langle S, A, T, R, P, \Pi\rangle$ be an* MDP *agent, and let $\pi \in \Pi$ be a policy that is optimal under the maximum expected utility criterion. Let $\langle S, A, T, B, D, I, Del, M\rangle$ be a a* BDI *agent where M and Del are* MEU*-optimal. If the environment is partially-observable, with S' the partially-observable counterpart of S, and non-deterministic, so that $|T(s,a)| \geq 1$, then $\forall i \in I$ $\forall s' \in S'$, it holds that $\psi^{i,s'}$ obeys π.*

PROOF. *This result generalizes the previous one because we are now considering a partially observable environment. However, because we have just expanded S into S' on both the* BDI *and* MDP *sides, the result follows directly from Proposition 2.* □

Thus it turns out that there is a simple formal correspondence between policies and i-plans, similar to the one informally discussed in Section 2. This correspondence holds under rather restrictive assumptions, in particular the optimality requirements, but ensures that the i-plans generated reflect an optimal policy. If we are willing to relax the requirement for a set of i-plans to correspond to an optimal policy, we can create i-plans under fewer restrictions.

Now, these formal results, though they tell us that we *can* extract i-plans that obey a policy don't tell us *how* we might do so. We consider how to do this next. We can obtain an i-plan from any policy in practice by means of a simple search through the state space, following policy π until a local maximum is reached — this state is then selected as the intention. In order to select a unique intention using this process, we assume that the agent's actions always have the most likely outcome; otherwise, a tree would result instead of a simple path. Figure 2 outlines an algorithm `policyToIplan` that can do this in a Java-like pseudocode.

By following policy π in this manner, we are able to obtain as many i-plans as desired: after reaching a intention state, simply continue following the policy from the state that results after achieving the previous intention. For example, in Figure 3 we illustrate this using a fragment of the same state space we considered before. Assume the agent is currently in state S_{18}, $\pi(S_{18}) = a_3$, and $\pi(S_{23}) = a_2$, and that state S_{28} is an intention for the agent. The i-plan that can be extracted in this situation is the linear plan $\langle a_3, a_2\rangle$. In this case, the i-plan that arises is obtained as the result of only one iteration of the process. The same is true in the case of the simplified TileWorld considered in [19], where agents only build plans for reaching one hole, not multiple hole tours.

The process that we have outlined here will construct a set of i-plans that obey an *arbitrary* policy. Such a policy will not necessarily be optimal, and so nothing can be inferred about the optimality or otherwise of the set of i-plans that are established, but the members of the set of i-plans will obey the policy.

With the procedure `policyToIplan`, and the fact that we can determine intentions from i-plans — intentions are, by definition, the states that i-plans lead to — we have answered the first of the questions that we set ourselves at the start of the paper.

4. FROM I-PLANS TO POLICIES

In this section, we will look to answer the second question. We consider how to use a set of i-plans to assign rewards to states in order to provide a policy for the underlying MDP that will mimic the behavior of an agent with the given i-plans. Such an approach makes it possible to use domain knowledge to solve problems that are intractable for existing MDP solution techniques (at the cost of providing sub-optimal solutions) — we use domain knowledge to construct i-plans and then use these i-plans to obtain a policy. Indeed, we can use intentions to construct i-plans and then i-plans to construct policies.

4.1 A single i-plan

We will start off by showing how rewards can be assigned in the state space in order to map *one* i-plan into a policy. Later on, we will see how the process is generalized to an arbitrary set of i-plans.

Assume the agent is currently in state s_a, and has adopted some i-plan ψ of length p. Then, we can assign a value to each state-action pair according to the following formula:

$$val(s^{\psi}_{i-1}, \psi_i) \stackrel{def}{=} i \cdot U(\psi)$$

$\forall i, 1 \leq i \leq p$, s^{ψ}_i is the i-th state involved in ψ, and ψ_i is the i-th action in ψ. In any other case, we have

$$val(s_j, a_k) \stackrel{def}{=} 0$$

$\forall s_j \neq s^{\psi}_i$, for any s^{ψ}_i such that $1 \leq i \leq p$, $\forall a_k$.

These values can be used induce a policy by selecting the action with the highest value given the current state. This process identifies a path through the state space assuming that nothing will go wrong. Such a policy only considers what to do in states that are involved in ψ, and therefore allows for no deviation from the path.

This mapping of states into values is then used as the reward function, which will clearly "mark the path" that the agent must follow in order to reach the goal, and executing the actions along this path will constitute a policy that conforms to the i-plan ψ and hence achieves the intention that ψ was constructed to achieve. In Figure 4, we show this assignment of values to states for the fragment of state space from Figure 3. The agent is initially in state S_{18}, and has formed the intention of reaching state S_{25} by executing actions a_2 and a_1. Therefore, if we assume that the intention has unit value, state S_{18} is given the value 1.0, S_{15} is given 2.0, and the goal is given 3.0; the rest of the states are assigned the value 0.

What we have so far is a mapping that turns an i-plan into its "equivalent" in terms of policies. That is, we have only obtained a correspondence from states to actions similar to the one described in Section 2, which doesn't consider what to do in cases in which the agent *drifts from the path* of its initial i-plan because of the failure of some action to reach the state that was planned for. To obtain a full policy — one that considers *every* possible state — we can extend the partial policy obtained from the i-plan using value iteration to establish a value for every state from the values that have been assigned to states that were part of the intention. The

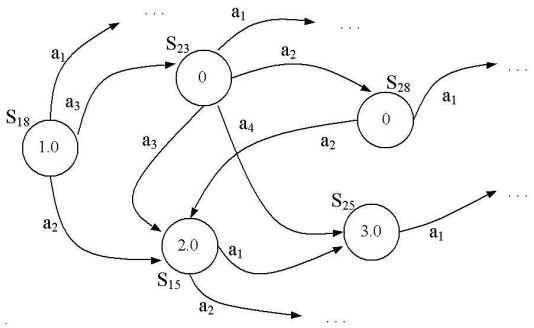

Figure 4: Part of a state space, with values assigned to the states leading to a goal.

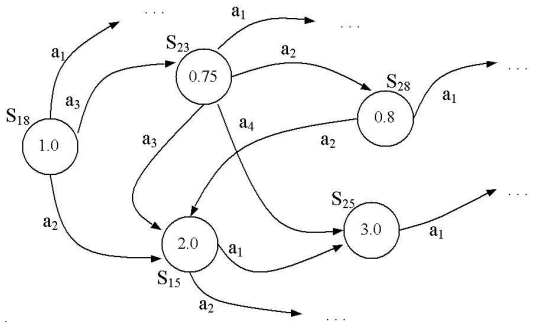

Figure 5: The state space from Figure 4, with values assigned to every state.

example in Figure 5 illustrates how doing this will assign the rest of the states an associated value, with value iteration working in exactly its normal fashion [10], assigning a state a value by suitably discounting the value of states to which it is connected by an action (with the discount determined by the expected cost of the action relating the two).

4.2 Sets of i-plans

So far we have only considered how to build a policy from a single i-plan. In order to use the same technique when an agent has more than one i-plan, only a minor change needs to be made. We must now take into account that the i-plan being considered is not necessarily the first. Therefore, we must keep count of the number of actions involved in the i-plans that have already been processed, which will be represented by the letter κ. When the process starts, we set $\kappa = 1$, indicating that the action to be considered is the first. The function val is now defined as:

$$val(s^{\psi}_{i-1}, \psi_i) \stackrel{def}{=} \kappa \cdot U(\psi)$$

$\forall i, 1 \leq i \leq p$, s_i is the i-th state involved in ψ, and ψ_i is the i-th action in ψ. After each action is considered, the value of κ is increased by 1. After all intentions have been processed, we have

$$val(s_j, a_k) \stackrel{def}{=} 0$$

$\forall s_j \neq s_i$, for any s_i involved in some i-plan, $\forall a_k$. Suppose, for example, that the agent has two i-plans ψ and φ, of lengths p and q, respectively. After processing ψ, κ will have the value $p + 1$ and, after processing φ, its value will be $p + q + 1$.

The process just described is captured by the algorithm `iplanToPolicy` outlined in the form of JAVA-like pseudocode in Fig-

```
Policy iplanToPolicy(IplanSet I, BDI A){
    MDP m; Policy π;
    ValueFunction val; int κ, j = 0;

    val.initialize(0); m.initialize(A);
    orderedIplan = I.obtainOrdering();
    for each i in orderedIplan do {
       j = 0;
       for each action a in i do {
          s = i.obtainInvolvedState(j++);
          val.setState(s,a) = κ*i.getUtil();
          κ++;
       }
    }
    π = valueIteration(m,val);
    return π;
}
```

Figure 6: Pseudocode for mapping i-plans into policies

ure 6. The following proposition formalizes the relationship between a set of i-plan and a policy as established by this algorithm:

PROPOSITION 4. *The algorithm* `iplanToPolicy` *obtains a policy which conforms to the given set of i-plans.*

PROOF. *The algorithm considers each i-plan in the set in turn, assigning a value to each state-action pair that is involved in the i-plan. The value that is assigned is the product of the current i-plan's utility and a monotonically increasing succession of integers. Therefore, each state-action pair will receive a monotonically increasing value with respect to the one before it. Up to this point we have an assignment of monotonically increasing values to the state-action pairs that the given set of intentions dictates, which allows us to obtain a straightforward policy using these values. If we feed the value iteration algorithm with these value assignments, the rest of the state-action pairs (not involved in the set of intentions) will receive values according to this assignment. The value iteration algorithm ensures us that the resulting policy will be* MEU *with respect to the input values.* □

`iplanToPolicy` generalizes the process described above for single i-plans, and to do so requires the i-plans to be ordered — the order in which the i-plans are processed. As we have seen above, i-plans can be assigned utilities in relation with how their execution will reward the agent by reaching certain goals, and this provides a suitable ordering.

Using `iplanToPolicy` to assign rewards to states, we can use value iteration as an *anytime* algorithm to construct a policy. With each iteration, more and more states will receive non-zero values, making the plan more universal. The example in Figure 7 shows how the policy evolves through three iterations of the algorithm on a specific state space.

Of course value iteration always works like this — with values backing-up across the state space from states with rewards — but it is not usually considered an anytime algorithm. That is because in early iterations most states have zero values, and so no useful policy can be extracted. However, in this case, an initial useful policy is provided by the values established by an i-plan. That policy can be used "as is". The worst that will happen is that whenever an agent strays off the "path" marked by states with non-zero values, it will try immediately to get "back on" that path by the shortest route rather than taking the optimal route that would be visible if all states had values. As more states have values filled in by value

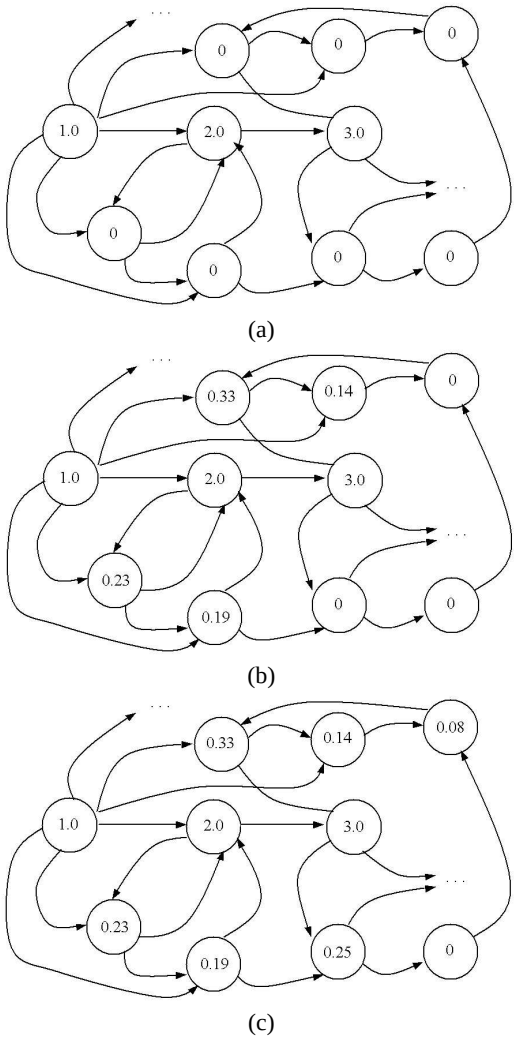

Figure 7: State values after the (a) first, (b) second, and (c) third iterations

iteration, the less likely this is to happen, and thus the better the policy that is computed.

4.3 Practical reasoning and policies

Up to now, we have only been considering the problem of how to map a single i-plan or a group of i-plans into a policy. Even though we have established a relationship between the most important components of the BDI and MDP models, the mapping is not yet complete because a policy obtained using the methods developed in the previous section corresponds to a "snapshot" of the BDI agent's state. If the agent switches intention (for example, due to reconsideration performed in order to profit from a change in the environment), the policy might no longer be valid.

It is possible to map the *complete* set of possible i-plans (all the i-plans for all its possible intentions) into a single policy by considering what intention, and hence what i-plans, the BDI agent would adopt in every possible state. If deliberation is performed in *every* state, followed by means-ends reasoning, a policy can be built by assigning actions to states corresponding to what the BDI planner dictates. It is clear that this process need not be initiated in every state in practice — if a state already has an action assigned to it

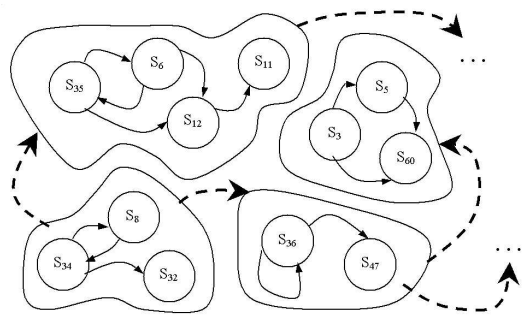

Figure 8: Grouping of states into "envelopes".

(because a previously built plan runs through it), then there is no sense in performing the same computations again.

The fact that we can do this, and that we can establish a set of i-plans from a set of intentions, gives us an answer to the second question we posed at the start of the paper.

It is interesting to point out that such a policy can be built in this way without considering rewards. Rewards played a fundamental role in previous sections because they were the basis for the construction of *contingencies* — once the main plans were laid, the actions performed in the rest of the states were based on these rewards. However, because we are now considering how the BDI agent behaves in every state, such an assignment of rewards is no longer necessary.

We should also point out that this mapping is initially of theoretical interest only, and its sole purpose is to show that a relationship exists between the universal behavior of a BDI agent and a policy for an MDP agent. The computational costs of constructing such a policy would be just as high as (or higher than) the actual process of solving the MDP directly.

5. TOWARDS HIERARCHICAL POLICIES

The analysis in the previous section established a relationship between BDI and MDP models based on the existence in the BDI agent's plan library of i-plans that lay out a precise sequence of actions that the agent has to carry out. It turns out that the i-plans that a BDI agent are typically equipped with aren't quite so detailed.

What happens in a typical BDI implementation is that the deliberation process decides *what* is to be achieved — what intention to adopt. Once a decision is made, the means-ends reasoning process then decides *how* this will be achieved — what i-plan will be used — picking from a set of relatively abstract i-plans and instantiating it to fit the specific situation. The result is to effectively groups states into "envelopes" which contain sets of states that are equivalent as far as the current intention of the agent is concerned but which might differ in detail. For example, considering Figure 8, imagine that each state represents an agent location. The agent is currently at s_{34} and needs to get to s_{60}, and its environment is split into a series of rooms that group states together — s_{34} with s_8 and s_{32}, all in the same room but at different locations within the room, say — and so on. Now the important thing as far as the agent is concerned, while navigating around, is which room it is in, so it can construct an i-plan at the "room level", figuring that it needs to get to the $\langle s_{36}, s_{47} \rangle$ room first, then to the $\langle s_3, s_5, s_{60} \rangle$ room.

The idea of this kind of state aggregation is not a novel one. In fact, much work has been dedicated to the grouping of states in MDPs and POMDPs, as can be seen in [2, 4, 6, 21], with the aim of improving the tractability of solving problems with ever larger state-spaces. However, it does tie in very neatly with the BDI model, and we can easily see that it will be possible to establish similar results to those we have obtained above, but where the correspondence is not between whole-environment policies and i-plans at the level of individual states, but between i-plans and policies at the state-aggregation level.

Now, being able to construct policies at the state-aggregation level is a useful thing to do. Open questions in work on state-aggregation are how to aggregate states, what level states should be aggregated at, and which states should be grouped with which states. The methods we have developed here potentially provide answers — i-plans identify what states to group together, and these can then be turned into high-level policies for a partial MDP solution that can then be refined, filling in policies that tell the agent how to act within grouped states (which is a smaller, and hence more tractable problem than determining a whole-world policy).

6. DISCUSSION

The BDI model was established at least in part [3] because it was felt that decision-theoretic models were too computationally intractable to use in practice. This has led to a perception that the two models are somehow cast in opposition with one another, that one adopts the BDI model only if one believes that decision theoretic models are somehow wrong or impractical, and that to use the BDI is to somehow settle for less than perfect performance (it is, after all, a heuristic approach). Our aim in this work is to steer a middle course.

We have previously shown [19] that on a particular task there are cases where using an MDP model provides the best solution, while in other cases an BDI approach works best — as the task grows in size beyond the range of problems that can be optimally solved using an MDP, the BDI approach outperforms the best one can do with an MDP. This suggests that some aspects of the BDI approach may be worth considering in more detail. Here we provide some of that detail.

The results from Section 3 show that the BDI model is not inherently sub-optimal. If we can, as the results there show we can in theory, construct a set of intentions that obey an optimal policy, then the BDI model will give us optimal performance. That it doesn't in practice, is because intentions are constructed not from optimal policies but from domain knowledge, and it is because of the mismatch of the two that sub-optimality creeps in.

The results in Section 3 don't help us in practice since there is no need to build intentions from an optimal policy — if we have an optimal policy we can just execute it directly. However, the results in section Section 4 can help us in practice. They show us how to construct policies from i-plans, and we can (as in [19]) create i-plans in situations where the state-space is so large that it isn't possible to generate anything like an optimal policy from first principles.

Finally, the discussion in Section 5 suggests that we can use the relationships we have developed to construct policies at a sufficiently abstract level that we can gain some computational advantage over directly solving an MDP. Of course, so far we have only demonstrated that it is possible to do this in theory. Some practical demonstration will be necessary to be completely convincing, and providing empirical results that do demonstrate this, using the same Tileworld testbed as we used in [19], is what we are currently working towards.

7. SUMMARY

In this paper, we have presented a series of relationships that exist between certain components of the BDI model and those of

MDPs. We have extended the work of [16] by exploring how *i-plans* on the BDI side and *policies* on the MDP side can be related to one another. To our knowledge, this is the first time that such a relationship has been proposed.

We have mapped from policies to i-plans by proving that the i-plans derived from an optimal policy are those adopted by a BDI agent which selects i-plans with the highest utility, and an optimal reconsideration strategy; furthermore, we presented a computational version of this mapping, by means of a search algorithm based on following the given policy through the state space. The other part of the mapping, obtaining policies from i-plans and thus intentions, was also characterized both theoretically and algorithmically. First, we established the mapping declaratively, by assigning rewards to states in such a way as to induce the selection of actions in the same way as they are selected by the intentions. This allows the derivation of a policy by means of any of the MDP solution algorithms. A simple algorithmic version of this mapping was also presented, indicating that one possible solution is a policy that simply follows the actions as dictated by the i-plan, and says nothing about the rest of the possible states. The use of value iteration as an anytime algorithm which refines and completes the policy with each i-plan was also proposed as an alternative way of deriving a policy.

This work constitutes the first steps in a line of research that aims to unify the two approaches. In the future we will test these findings empirically in pursue of the development of agent design methodologies that make use of the best of both the BDI and MDP models.

8. ACKNOWLEDGMENTS

The work described in this paper was partially supported by a grant from the City University of New York PSC-CUNY Research Awards Program and partially supported by the Air Force Office of Scientific Research under Grant Nr. FA95500510298, by Army Research Office grant number DAAD190310202, by NSF grants IIS0329851 and 0205489, and by the Joint Institute for Knowledge Discovery

9. 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] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1–2):49–107, 2000.

[3] M. E. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. In R. Cummins and J. L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

[4] T. Dean, R. Givan, and S. Leach. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 124–131, 1997.

[5] M. P. Georgeff and F. F. Ingrand. Decision-making in embedded reasoning systems. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 972–978, 1989.

[6] J. Goldsmith and R. H. Sloan. The complexity of model aggregation. In *Artificial Intelligence Planning Systems*, pages 122–129, 2000.

[7] W. S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1–4):47–66, Apr. 1991.

[8] 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.

[9] M. Pollack and M. Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 183–189, 1990.

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

[11] A. S. Rao and M. P. Georgeff. Deliberation and its role in the formation of intentions. In *Proceedings of the 7th Annual Conference on Uncertainty in Artificial Intelligence*, pages 300–307, 1991.

[12] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, 1995.

[13] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1039–1046, 1987.

[14] M. Schut and M. Wooldridge. Intention reconsideration in complex environments. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 209–216, 2000.

[15] M. Schut and M. Wooldridge. Principles of intention reconsideration. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 340–347, 2001.

[16] M. Schut, M. Wooldridge, and S. Parsons. On partially observable MDPs and BDI models. *Lecture Notes in Computer Science*, 2403:243–??, 2002.

[17] M. Schut, M. Wooldridge, and S. Parsons. The theory and practice of intention reconsideration. *Journal of Theoretical and Experimental AI*, 16(4):261–293, 2004.

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

[19] G. I. Simari and S. Parsons. On approximating the best decision for and autonomous agent. In *Proceedings of the Sixth Workshop on Game Theoretic and Decision Theoretic Agents*, pages 91–100, 2004.

[20] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

[21] J. Tsitsiklis and B. van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1/2/3):59–94, 1996.

[22] C. J. C. H. Watkins. *Learning with delayed rewards*. PhD thesis, Cambridge University, 1989.

[23] 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.

[24] M. Wooldridge and S. Parsons. Intention reconsideration reconsidered. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Agent Theories, Architectures, and Languages V*, pages 63–80. Springer-Verlag: Heidelberg, Germany, July 1999.