Constructing BDI Plans from Optimal POMDP Policies, with an Application to AgentSpeak Programming

Diego Rodrigues Pereira, Luciano Vargas Gonçalves, Graçaliz Pereira Dimuro, and Antônio Carlos da Rocha Costa

Programa de Pós-Graduação em Informática, Universidade Católica de Pelotas
Rua Felix da Cunha 412, 96010-000 Pelotas, Brazil
{dpereira,llvarga,liz,rocha}@ucpel.tche.br

Abstract. Based on the analysis of the hybrid approach BDI-MDP found in the literature, this paper introduces the algorithm **policyToBDIplan** that builds AgentSpeak plans for BDI agents that obey optimal POMDP policies, presenting an example of an application, using Jason. POMDP policies are mapped into BDI agent plans, following an intention, considering that plans extracted from a POMDP optimal policy are the ones adopted by the BDI agent that selects the plan with the greatest utility, and an optimal strategy reconsideration.

1 Introduction

The BDI (*Beliefs*, *Desires*, *Intentions*) agent architecture [1] has been used in different and complex applications. In this approach, agents have a set of beliefs about the states of the world and a set of desires, which identify those states that the agent has as goals. By a process of deliberation, the agent formulates one or more intentions (considered here as the states that the agent is committed to bringing about). The agent then builds a plan to achieve those intentions (through, e.g, some form of means-ends reasoning), and executes it.

Observe that, since this is a heuristic approach, the plan can fail, the environment can change or the intention can be no longer achieved. When this happens, the BDI agent has to reconsider its intentions, to check if they are reachable. This process usually decreases its performance, making almost impossible to reach an optimal behavior, if compared to decision theory models. This balance between the time spent deliberating and the time executing plans is very important to a good BDI agent. Also, the BDI approach does not present tools for a systematic quantitative analysis of the performance of the agents. This is the reason why BDI agents are often outperformed by the those using Markov Decision Processes (MDP) [2], when the MDP solution is tractable.

On the other hand, *Partially Observable Markov Decision Processes* (POMDP), the extension of MDPs to deal with partially observable environments [3, 4], can be regarded as an ideal approach for the implementation of intelligent agents that have to choose optimal actions in partially observable stochastic domains. There are several algorithms for finding optimal policies for POMDPs (see, e.g., the *Witness* algorithm [3], which yields policy graphs mapping actions to observations).

However, the very nature of these algorithms, and in particular the need to establish the outcome of every possible action in every belief state, means that finding policies is

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 240-249, 2008.

intractable in many practical cases, particularly in large or complex problems [5]. The BDI model, in contrast, can scale to handle problems that are beyond the scope of a POMDP solution, and can outperform approximate POMDPs for some relatively small problems [5].

Hybrid models have then been proposed in order to take advantages of both POMDPs and BDI architecture (see, e.g., [5–8]). In [5], the formal relationships existing between certain components of the BDI model and those of POMDPs were discussed, showing how BDI plans and POMDP policies could be related to one another. For example, a BDI agent with plans based on POMDP policies, or a POMDP policy built from a set of BDI plans, can be used to improve the performance of BDI agents or the tractability of PODMPs, respectively [5].

In this paper, extending the proposal presented in [9], which is directed to totally observable models, we discuss the relationship between POMDPs and the BDI approach, presenting an algorithm to build AgentSpeak plans for BDI agents, extracted from POMDPs policies, in a manner that those plans "obey" those policies.

The paper is organized as follows. The relationship between the BDI and MDP descriptions is presented in Sect. 2. A discussion about the relationship between BDI plans and optimal MDP and POMDP policies is presented in Sect 3. The algorithm for the construction of BDI plans from optimal POMDP policies is introduced in Sect. 4. Section 5 presents a case study. Section 6 is the Conclusion.

2 Relationship Between BDI and POMDP Models

In this section, we use the notation adopted in [5], which, although not standard, simplifies the BDI and MDP descriptions in order to establish relations between their components. Then, we have that:

Definition 1. *A Markov Decision Process (MDP) is defined as a tuple* (S, A, T, R, P, π)*, where:*

- S *is a finite set of states (of the agent's environment);*
- A *is a finite set of actions (on the environment);*
- $T : S \times A \rightarrow \Pi(S)$ *is the transition function, which, given an action* $a \in A$ *executed in a state* $s \in S$*, returns a probability distribution over the state set* S*, so that* $T(s, a, s')$ *denotes the probability of (the environment) reaching the state* s' *given that the current state is* s *and that the agent performs the action* a*;*
- $R : S \times A \rightarrow \mathbb{R}$ *is the reward function, that is, the reward for performing the action* $a \in A$ *in a state* $s \in S$*;*
- P *is the initial probability distribution over the state set;*
- $\pi : S \times A$ *is a policy, that is, a mapping state/action.*

An agent behaving according to the MDP description is called an MDP agent. In the case of a totally observable environment, P indicates the current state with 100% of probability, since in such case the agent always knows the current state.

Definition 2. *A Partially Observable Markov Decision Process (POMDP) is defined as a tuple* $< S, A, T, R, \Omega, O, P >$*, where:*

- S, A, T, R, P *are defined as in a MDP (see Def. 1);*
- Ω *is a finite set of observations;*
- $O : S \times A \rightarrow \Pi(\Omega)$ *is the observation function, which gives, for each action and resulting state, a probability distribution over the possible observations (on the environment) .*

In this partially observable description, we consider that the agent has a belief state, which depends on the action executed and its previous belief state, which will then update P. And finally, a BDI description is as follows:

Definition 3. *A BDI description is defined as tuple* $(S, A, T, B, D, I, Del, M)$*, where* S*,* A *and* T *are, respectively, the state space, the action set and the transition function (as in Def. 1), and*

- B *is the belief set,* D *is the desire set, and* I *is the intention set;*
- Del *is the deliberation component;*
- M *is the means-end reasoning component.*

An agent behaving according to the BDI description is called a BDI agent.

For an agent in a given environment, we consider that S, A and T are the same for both descriptions (BDI and MDP/POMDP). Besides, we consider that B and P represent the same idea: the agent's (internal) current state. With these equivalences defined, we have in one hand rewards and policies and in the other hand desires, deliberation, means-end reasoning and intentions. However, since rewards are used to determine policies, and desires are a step to determine intentions, the former components can be subsumed under the corresponding latter components, and can thus be abstracted away. So, the central relation that we must consider in detail when establishing the relation between the two models is the relation between policies and intentions. [5]

The semantics of an intention usually varies among the researchers. The most common is that of a plan that an agent takes to reach a certain state of the world. In this paper, we use intention as the state that the agent has committed to attain, and we will use the term intention-plan (or simply plan), to denote a sequence of actions built to reach a specified state, that is, to reach an intention. The plan will be built starting from the agent's current state and its intention. An agent can create different plans to accomplish the same intention, depending on its starting state. In the same way, it will create different plans to reach different intentions, from the same starting state.

In [5] there are several equivalences established between both descriptions, BDI and MDP, considering that they work in the same state space. For us, the most relevant result from such work is that given a policy, it is possible to derive from it one or more BDI plans, to solve the same problem in the same space state. Therefore, we can say that a policy incorporates a set of BDI plans.

The expected utility of a BDI plan can be obtained in the same way as for a policy in an MDP, by setting a value to each action and each state in which it is executed. The main difference is that a BDI plan focuses on a path in the state space, while a policy takes into account all possible paths.

The disadvantage of the MDP approach is, of course, that to calculate a policy one spends greater computacional cost than to create a simple plan for a BDI agent.

On the other hand, the BDI agent pays the price of building a plan while it is online, while an MDP agent pays the price of obtaining the optimal policy while it is offline. However, the BDI agent often can do with suboptimal plans, which it constantly revises for mismatches with the evolution of the environment.

3 From Policies to BDI Plans

We assume that we have a policy π that solves an MDP completely specified, and that π is optimal. However, from any π it is possible to extract utility values referring to the states that induced π, and these can be used to establish values for the BDI plans.

A BDI plan *obeys* a given policy π if, and only if, the actions prescribed by the plan are the same prescribed by the policy at each one of the BDI plan's intermediate states. Observe that we assume that BDI plans are linear, and no consideration is made respecting possible unexpected results to the actions.

Given a BDI agent with a set of plans corresponding to an optimal policy π of an MDP agent, if the BDI agent is in the state s, the BDI plan with the highest utility value for the BDI agent will be the one that obeys π, starting in s.

We assume, then, that the Del component is optimal, selecting the intention with highest utility, and the actions are completely deterministic in the environment. We also assume that π is optimal under the maximum expected utility criteria, always selecting actions that lead the agent to the best state-goal, in the best possible way. Thus, if the M component of the BDI agent chooses the intention with the highest utility, the environment is totally observable, and the actions are deterministic, then Del always selects the same intention/goal that π would do.

Conversely, if M chooses the BDI plan with the highest reward, it will choose a BDI plan that follows the same path through the space state (starting from any state that the agent is currently in) that π would do. Thus, if we assume that states with the same rewards are considered in the same order by π and M, then it is clear that BDI plans generated by M will obey π [5].

If the actions are not deterministic, the BDI plans' utility will not be clearly defined. Instead of a simple reward sum through the BDI plan's path, we would have to consider the action failures. Thus we would have to assume that the deliberation Del and the means-end reasoning M components are optimal under maximum expected utility criteria, instead of being capable of choosing the intention and the plan, respectively, with the highest reward.

We can then also assume, for non-deterministic environments, that all BDI plans will obey an optimal policy [5].

These results of [5] show why the BDI approach has difficulties in generating an optimal behavior: in classical BDI models, the deliberation component Del selects an intention and the means-end reasoning component M builds or chooses a plan to achieve such intention; so, to be able to choose a set of optimal actions, the deliberation component should be able to choose the intention that is optimal under maximum expected utility criteria before the means-end reasoning component chooses a BDI plan.

We now analyze the case in which the environment is partially observable, that is, the agent does not know in which state S it is in, and must thus rely on its observations

about the environment's current state [4]. POMDP models can handle this kind of situation by extending the concept of state. Rather than taking explicitly states from the set S (the space state describing all the states of the environment), a state in a POMDP is a probability distribution over S, denoted by S'. If we consider all possible probability distributions s'_i, we can think of policies and plans that are concerned with this new state space $S' = \bigcup_i s'_i$.

If S' is the partially observable counterpart of S and we consider that S' is the new space state where the BDI and MDP agents will operate, then both components B and P will identify some $s' \in S'$ as the agent's current state. Then, as stated in [5], we can correspondingly extend the statement that BDI plans obey a policy.

This correspondence is valid under restricted assumptions, but it assures that the generated BDI plans will reflect the optimal policy.

4 The Algorithm to Extract Plans from an Optimal POMDP Policy

In this section, we introduce the algorithm **policyToBDIplan**, that generates BDI plans in AgentSpeak language, which obey a policy that is optimal with respect to a given a POMDP.

In this work, POMDP optimal policies are obtained by applying the *Witness* [3] algorithm. Observe that, in order to reach an optimal policy, it would be necessary to enumerate all belief states, and all observation probability distributions in each belief state. Unlike other approaches, the *Witness* algorithm defines regions for the *value vector*, searching for a point where this vector is not dominant. Then it selects one action at a time and tries to find the best value for each of the actions separately. With these values, it combines the results and builds the value function V'.

The algorithm also selects one observation at a time. Then it builds a vector in a point in the belief state, based on the selection of a vector transformed from the original V for each observation. It begins in a random point in the belief state, generating the value for that belief state. Then, it adds this to the set and considers the updated set as the true V', and then tries to prove that this is really the V' vector.

During the vector building, it makes a choice for each observation; this is a selection of one vector V, representing a future strategy. The algorithm then searches for the individual choices made, observation by observation, in order to determine if a different choice has a better value. Then, it defines a region where that choice is better. If it happens that it finds a point in the belief state, where its new strategy is better than the previous one, then this belief state serves as witness to the fact that this vector choice is not the true V'.

The output of the *Witness* algorithm is a *policy table*, representing the *policy graph* of the problem (which may have nodes that are never going to be visited, and have thus to be removed from the table).

Each arc in the policy graph represents a sequencing of actions, departing from a node representing an action that is performed at some time, and going to the action that should be performed next, the sequencing being defined in accordance with the

observation that is made as the result of the performance of the action of the node that is at the origin of the arc.

Each line of the policy table represents a node of the policy graph. It has a column $Action$, to indicate the action to be performed at that node. Also, it has one column $ObsR_i$ for each of the observations that the agent can get after performing the action, each column indicating to which graph node (table line) the agent should go next, according to the observation got.

The algorithm **policyToBDIplan** (shown in Alg. 1) receives the policy graph, and then builds rules for the *AgentSpeak* [10] agent programming language, which will constitute plans for BDI agents of the *Jason* [11] agent programming environment.

The algorithm is as follows:

- The input is the (simplified) policy graph (pg) generated by the *Witness* algorithm;
- The output is a set of rules of the *AgentSpeak* language;
- For each line of the policy table, one rule $(rule)$ is extracted, to control the execution of the action $(action)$ by the agent. Also, for each observation in each line, another rule is extracted, to control the transition between table lines (in the algorithm, $ObsR1, ObsR2, ..., ObsRn$ represent the possible observations).

Algorithm 1 Algorithm to extract BDI plans from an optimal POMDP policy

```
policyToBDIplan(Policygraph pg){
Policygraphline pl
BDIplan plan
BDIrule ruleR1, ruleR2, ... ,ruleRn
set pl to firstLine(pg)
repeat
    set ruleR1 to BDIrule(head={Node(pl)}, context={True}, body={act(Action(pl)), nextGoal(Node(pl'))})
    set ruleR1 to BDIrule(head={Node(pl')}, context={ObsR1}, body={nextGoal(ObsR1(pl))})
    set ruleR2 to BDIrule(head={Node(pl')}, context={ObsR2}, body={nextGoal(ObsR2(pl))})
    ...
    set ruleRn to BDIrule(head={Node(pl')}, context={ObsRn}, body={nextGoal(ObsRn(pl))})
    addBDIrules {ruleR1, ruleR2, ... , ruleRn} to plan
    advance ln
until ln beyond lastLine(pg)
return  plan
}
```

5 Case Study: The Tiger Problem

In this section, we present an application using the classic *Tiger Problem* [3]. In this example, the agent is purely reactive, and the state space is very small, so to build a BDI agent has no advantages over a reactive one. But, nevertheless, the simplicity of the problem helps showing how the algorithm works.

The problem consists in one agent in front of two doors. Behind one of them there is a tiger, which will attack the agent if its the door is open, and behind the other door there is a great reward. The agent can listen to the doors, in order to gain some information

about the tiger location. But listening is not for free (there is a cost associated to it), and is also not reliable (so the agent can sometimes here mishear the location of the tiger).

The POMDP model for the tiger problem is given by the tuple $< S, A, T, R, \Omega, O >$ where:

- $S = \{sl, sr\}$ is the set of states, where sl represents that "the tiger is behind the left door", and sr that "the tiger is behind the right door";
- $A = \{ltn, ol, or\}$ is the set of actions of "to listen", "to open the left door" and "to open the right door";
- $T : S \times A \rightarrow \Pi(S)$ is the transition function, where the action ltn does not cause any transition in the system and ol, and or resets the problem;
- $R : S \times A \rightarrow \{-1, +10, -100\}$ is the reward function, where -1 is the cost for ltn, $+10$ is the reward for opening the door without the tiger behind, and -100 is the cost for opening the door with the tiger behind;
- $\Omega = \{tl, tr\}$ is the set of observations, where tl is for listening the "tiger on the left", and tr is for listening the "tiger on the right";
- $O : S \times A \rightarrow \Pi(\Omega)$ is the observation function, which returns the probability of making the observation o, given that action a was executed and the state s was reached.

The belief state of the agent, in the beginning of each turn, is a probability distribution over the set of states, initially 0.5 for sl (tiger on the left) and 0.5 for sr (the tiger is behind the right door). If the state is sl (the tiger is behind the left door), there is a probability of 85% that the agent listens the tiger on the left, when the agent does the action ltn (to listen), but also there is a probability of 15% that the agent listens the tiger on the wrong side (tiger on the right). The situation in the state sr (the tiger is behind the right door) is precisely the opposite.

The optimal policy for this problem, i.e., the optimal mapping observation/action, can be described by the simplified policy graph presented in Fig. 1 and in Tab. 1, obtained using the *Witness* algorithm. In this case, the policy establishes that the agent must listen that the tiger is on the same side twice before opening any door, in order to increase the certainty degree.

Table 1. Policy table for the Tiger Problem

Line	Action	Obs=tr	Obs=tl
0	ltn	1	3
1	ltn	2	0
2	ol	0	0
3	ltn	0	4
4	ol	0	0

The algorithm **policyToBDIplans** extracts BDI plans such as the one shown in Fig. 2. For example, in the line `0` (indicated in the plan by the rule head `+!State(0)`), the action ltn (to listen) should be executed, and if the observation got after the action is tr (tiger on the right), then the next line to be considered is line `1`(as indicated

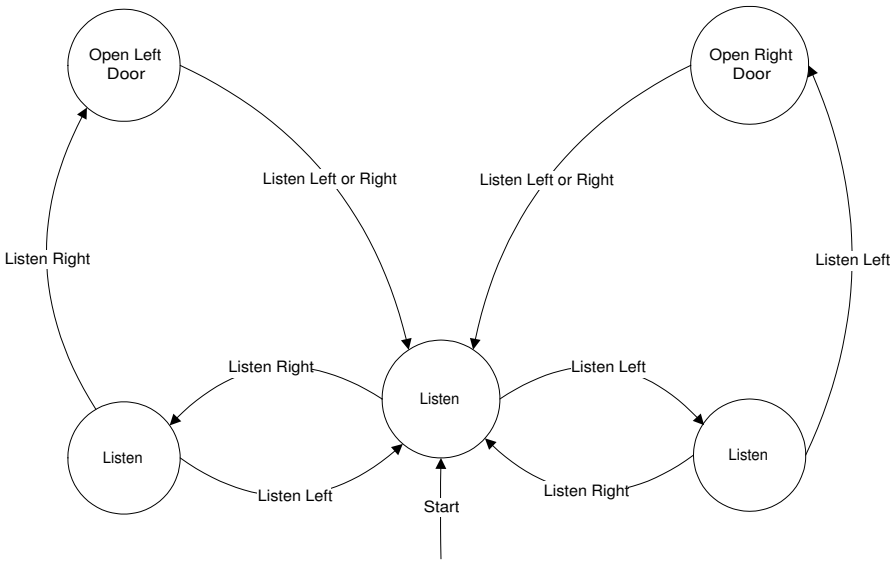

Fig. 1. Policy graph for the Tiger Problem

by `!State(1)` in the first transition control rule with head `+!State(0')`). Otherwise, if the observation is tl (tiger on the left), then the next line to be considered is line `3` (as indicated by `!State(3)` in the second transition control rule with head `+!State(0')`). In the line `1`, if tr (tiger on the right) is listened, the agent should proceed to line `2`, which tells it to open the left door (action ol).

For testing we have done 100 runs of the problem, and calculated the average behavior. In the tests, the agent chose the right door in $79,5\%$ of the cases, and missed $20,5\%$, as expected, due to the values of the observation probabilities (85% and 15%).

6 Conclusion and Final Remarks

This paper presented a discussion about the relationship between the BDI and MDP approaches introduced in [5], going further in the proposal of a hybrid BDI-POMDP approach, where POMDP policies are mapped to plans for BDI agents.

For that, we introduced the **policyToBDIplan** algorithm for building plans for BDI agents in the *AgentSpeak* language, given a POMDP policy graph obtained by using the *Witness* algorithm. To illustrate how the algorithm works, we used the Tiger Problem as an example.

An immediate application for this hybrid approach is in the problem of self-regulation of personality-based social exchanges in multiagent systems [12, 13], as proposed in [14–16]. The exchange-regulation mechanism tries to maintain the balance of social exchanges in equilibrium, without interrupting the sequence of interactions. When this mechanism is internal to the agents, the decision processes that regulate the agents'

```
+!State(0): True ->
            act(listen),
            !State(0').
+!State(0'): obs == TR ->
            !State(1).
+!State(0'): obs == TL ->
            !State(3).

+!State(1): True ->
            act(listen),
            !State(1').
+!State(1'): obs == TR ->
            !State(2).
+!State(1'): obs == TL ->
            !State(0).

+!State(2): True ->
            act(open-left),
            !State(2').
+!State(2'): obs == TR ->
            !State(0).
+!State(2'): obs == TL ->
            !State(0).

+!State(3): True ->
            act(listen),
            !State(3').
+!State(3'): obs == TR ->
            !State(0).
+!State(3'): obs == TL ->
            !State(4).

+!State(4): True ->
            act(open-right),
            !State(4').
+!State(4'): obs == TR ->
            !State(0).
+!State(4'): obs == TL ->
            !State(0).
```

Fig. 2. Example of BDI plans generated by the algorithm **policyToBDIplans**

behaviors have to operate under the condition of partial observability, since the agents do not have the access to the other agents' internal states. The hybrid BDI-POMDP approach was adopted in the social exchange simulator we are developing, which is being built using the Jason framework.

As future work, we will target an algorithm to build BDI plans for Weakly Coupled Markov Decision Process [17–20], for multiagent systems that address the problem of allocation of resources, products and processes.

Acknowledgements. This work was partially supported by Petrobrás (joint project with PENO/ COPPE/UFRJ), CAPES, FAPERGS and CNPq (Proc. 473201/2007-0).

References

1. Wooldridge, M.: Reasoning about Rational Agents. Intelligent Robots and Autonomous Agents. The MIT Press, Cambridge (2000)
2. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, New York (1994)
3. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**(1-2) (1998) 99–134
4. Lovejoy, W.S.: A survey of algorithmic methods for Partially Observable Markov Decision Processes. Annals of Operations Research **28**(1–4) (1991) 47–66
5. Simari, G.I., Parsons, S.: On the relationship between MDPs and the BDI architecture. In Nakashima, H., Wellman, M.P., Weiss, G., Stone, P., eds.: Proc. of the 5th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2006, Hakodate, Japan, May 8-12, 2006, ACM (2006) 1041–1048

6. Nair, R., Tambe, M.: Hybrid BDI-POMDP framework for multiagent teaming. Journal of Artificial Intelligence Research **23** (2005) 367–420
7. Schut, M.C., Wooldridge, M., Parsons, S.: On partially observable MDPs and BDI models. In d'Inverno, M., Luck, M., Fisher, M., Preist, C., eds.: Foundations and Applications of Multi-Agent Systems, Selected Papers of UKMAS Workshop 1996-2000. Number 2403 in LNCS. Springer, Berlin (2002) 243–260
8. Gupta, T., Varakantham, P., Rauenbusch, T.W., Tambe, M.: Demonstration of teamwork in uncertain domains using hybrid BDI-POMDP systems. In Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O., eds.: Proc. of the 6th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2007, Honolulu, May 14-18, 2007, IFAAMAS (2007) 264
9. Pereira, D.R., Dimuro, G.P.: Um algoritmo para extração de um plano BDI que obedece uma política MDP Ótima. In: Anais do Workshop-Escola de Sistemas de Agentes para Ambientes Colaborativos, Pelotas, PPGINF/UCPel (2007) (in Portuguese).
10. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In van Hoe, R., ed.: Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World. Number 1038 in LNCS. Springer, Berlin (1996) 42–55
11. Bordini, R.H., Hübner, J.F., Wooldrige, M.: Programming Multi-agent Systems in AgentSpeak Using Jason. Wiley Series in Agent Technology. John Wiley & Sons, Chichester (2007)
12. Dimuro, G.P., Costa, A.C.R.: Exchange values and self-regulation of exchanges in multi-agent systems: The provisory, centralized model. In Brueckner, S., Serugendo, G.D.M., Hales, D., Zambonelli, F., eds.: Engineering Self-Organising Systems. Number 3910 in LNCS. Springer, Berlin (2006) 75–89
13. Dimuro, G.P., Costa, A.C.R., Gonçalves, L.V., Hübner, A.: Centralized regulation of social exchanges between personality-based agents. In: Coordination, Organizations, Institutions, and Norms in Agent Systems II. Number 4386 in LNCS. Springer, Berlin (2007) 338–355
14. Pereira, D.R.: Construção de planos BDI a partir de políticas ótimas de POMDPs, com aplicação na auto-regulação de trocas sociais em sistemas multiagentes. Dissertação de mestrado, PPGINF/UCPel, Pelotas, RS (2008) (in Portuguese).
15. Gonçalves, L.V., Pereira, D.R., Dimuro, G.P.: Auto-regulação de trocas sociais baseadas em personalidades em sistemas multiagentes. In: Anais do Workshop-Escola de Sistemas de Agentes para Ambientes Colaborativos, Santa Cruz, UNISC (2008) (in Portuguese).
16. Pereira, D.R., Gonçalves, L.V., Dimuro, G.P., Costa, A.C.R.: Towards the self-regulation of personality-based social exchange processes in multiagent systems. In: Proceedings of the 19th Brazilian Symposium on Artificial Intelligence. LNAI. Springer, Berlin (2008) (to appear).
17. Meuleau, N., Hauskrecht, M., Kim, K.E., Peshkin, L., Kaelbling, L.P., Dean, T., Boutilier, C.: Solving very large weakly coupled markov decision processes. In: Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI-98) and of the 10th Conf. on Innovative Applications of Artificial Intelligence (IAAI-98), Menlo Park, AAAI Press (1998) 165–172
18. Parr, R.: Flexible decomposition algorithms for weakly coupled markov decision problems. In Cooper, G.F., Moral, S., eds.: Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, San Francisco, Morgan Kaufmann (1998) 422–430
19. Meuleau, N., Hauskrecht, M., Kim, K.E., Peshkin, L., Kaelbling, L.P., Dean, T., Boutilier, C.: Solving very large weakly coupled Markov decision processes. In: Proc. of AAAI98/IAAI98, Menlo Park, AAAI Press (1998) 165–172
20. Dolgov, D.A., Durfee, E.H.: Optimal resource allocation and policy formulation in loosely-coupled markov decision processes. In Zilberstein, S., Koehler, J., Koenig, S., eds.: Proc. of the 14th Intl. Conf. on Automated Planning and Scheduling,ICAPS 2004, Whistler, AAAI (2004) 315–324