Association Rule-Based Markov Decision Processes

Ma.de Guadalupe García-Hernández[1], José Ruiz-Pinales[1], Alberto Reyes-Ballesteros[2], Eva Onaindía[3], J. Gabriel Aviña-Cervantes[1], Sergio Ledesma[1], Donato Hernández[1]

[1] Universidad de Guanajuato, Comunidad de Palo Blanco s/n, Salamanca, Guanajuato, México {garciag,pinales,avina,selo,donato}@salamanca.ugto.mx
[2] Instituto de Investigaciones Eléctricas, Reforma 113, 62490, Temixco, Morelos, México, areyes@iie.org.mx
[3] Universidad Politécnica de Valencia, DSIC, Camino de Vera s/n, 46022, Valencia, España, onaindia@dsic.upv.es

Abstract. In this paper we present a novel approach for the fast solution of Markov decision processes based on the concept of association rules. These processes have successfully solved many probabilistic problems such as: process control, decision analysis and economy. But for problems with continuous or high dimensionality domains, high computational complexity arises, because the search space grows exponentially with the number of variables. In order to reduce the complexity in these processes, we propose a new approach to represent, learn and apply the actions that really operate on the current state as a small set of association rules and a new value iteration algorithm based on association rules. Experimental results on a robot path planning task indicate that the solution time and therefore the complexity of the proposed approach are considerably reduced, because they increase linearly when the number of the states increases.

Keywords: Markov decision processes, association rules.

1 Introduction and Motivation

In artificial intelligence, planning under uncertainty can find more realistic policies through decision-theoretic planning [7], considering that actions can have different effects in the world, pondering the weight of alternative plans for achieving the problem goals, and considering their costs and rewards. In a process control problem, many variables change dynamically because of the operation of devices (valves, equipment's switches, etc.) or the occurrence of exogenous events (uncontrollable events). If the control system does not consider the possibility of fault, then it will surely not make intelligent actions in the event of a fault occurrence. This problem is very complex and uncertainty plays an important role during the search for solutions. In this way, decision-theoretic planning allows the evaluation of the strengths and weaknesses of alternative plans. Since the addition of new capabilities to a planner, heuristic search has shown limitations for the case of non integer data and additive graphs in the solution of real world problems [6], so that it has been chosen to solve them by using Bayesian representation and inference [9] and Markov decision processes (MDPs) [16]. The latter have successfully solved decision problems, e.g. process control, decision analysis and economy. However, the computational complexity of MDPs is a significant one for the case of continuous or high dimensionality domains, resulting in an intractable solution time for very large problems [19]. There have been many efforts for developing different representation forms and computational methods for decreasing the search space. Boutilier *et al.* [8] grouped similar states, but they did not achieve automation because they made some steps by hand. In this way, Reyes *et al.* have proposed a simulator for planning of robotic movements (SPRM) [18]; they partitioned the search space based on a reward function, but the scalability was a great problem. Hauskrecht *et al.* [11] tested the hierarchy with macro-actions, but appropriate coupling was not accomplished and additional complexity was obtained because of the presence of macro-actions.

In order to reduce complexity, we propose a novel approach for the solution of MDPs in which relational actions are represented, learned and applied as a set of association rules (they imply certain relationships between objects in a database). We present experimental results indicating the high viability of the new approach. In Section 2 the MDPs are presented shortly and the problem faced by MDP solution methods is explained. In Section 3 the approach based on relational actions is described too. In Section 4 our approach based on the representation, learning and application of relational actions through the use of association rules on MDPs is discussed. In Section 5 our experimental results are presented. In Section 6 we present the conclusions.

2 Markov Decision Processes

MDPs are techniques for planning under uncertainty, named in honor of Andrei A. Markov. They were introduced originally by Bellman [2] and they have been studied extensively [16]. MDPs are a mathematical formalism for modeling a sequential decision problem, whose main goal is to find a reactive strategy or action policy, by maximizing an expected reward [16]. Markov's work supposes that an autonomous agent always knows the state where it is placed before executing any action, and the transition probability from a state depends only on the current state, not on its whole history. Bellman's equation is the base for the resolution of MDPs, and one of its characteristics is the careful balance between reward and risk. For each possible state one equation is formulated, the unknown quantity is the utility of the reached state. Therefore, it results in a non linear system of equations that is solved by means of the value iteration algorithm [4, 16], an efficient dynamic-programming algorithm, where the optimal utility is found by successive approximations of the form:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s') \qquad (1)$$

where $U(s)$ is the expected utility of the current state s, $U(s')$ is the expected utility of the reached state s', $R(s)$ is the immediate reward of the current state s,

$T\left(s,a,s'\right)$ is the transition model that gives the probability of reaching state s' when action a in state s is applied, and finally γ is the discount factor ($0 < \gamma < 1$). The state transition model can be represented by a dynamic Bayesian network [10]. Otherwise, when the future reward is insignificant, then $\gamma = 0$. In contrast, when the current reward is identical to the expected discounted reward then $\gamma = 1$. Therefore, equation (1) gives the utility of an agent in a specific state, and that utility is the immediate reward of this state plus the discounted utility from the reached state; this state results from applying an action over the previous one. The main idea consists in calculating the expected utility of each state and to use these utilities for the selection of the optimal action at each state. But in complex problems (with continuous domains or with high dimensionality), the main challenge during the resolution of MDPs is due to its high computational complexity, because in the transition model $T\left(s,a,s'\right)$ the search space grows exponentially with the number of variables.

3 The Relational Action Approach

It is clear that if we apply all the domain actions over the whole state space, then it results a great amount of information, and this one will be very difficult to process because it will require a lot of computational effort. In this way, the necessity of an action hierarchy for planning systems has been proposed [3], but it was tested on simple problems. Other researchers have used a constrained action set (*r*-actions) for each state during the inference process through STRIPS probabilistic operators [13]. Relational actions (*r*-actions) can take the form:

```
if <state(i)> then <r-action(k)>.
```

In order to obtain *r*-actions, it is necessary to evaluate the *k*-th *r*-action as a function of the *i*-th state. This type of actions only depends on the current state and the planner does not have to compute on the whole action domain.

If we consider the simple example represented in Figure 1, we can observe that the model contains three actions (`a1, a2, a3`) and four states (`s1, s2, s3, s4`) for an autonomous agent. It is easy to observe that when a1 operates on `s1` (initial state), the agent has 90% of probability of reaching `s2` and it has 10% of probability of staying in the same state. Otherwise, when `a3` is executed on `s3`, then the agent can reach `s4` (final state) with 50% of probability and it can stay in the same state with another 50%.

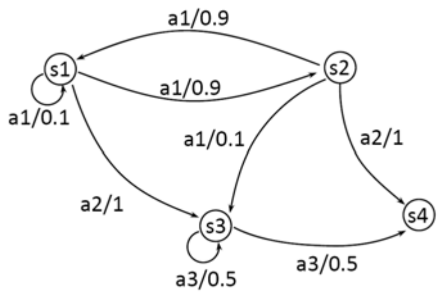

Figure 1. A Markov model.

From this diagram, it is easy to obtain the *r*-action set for each state (see Table 1).

Table 1. Sets of *r*-actions for each state.

state	*r*-action set
s1	a1,a2
s2	a1,a2
s3	a3

In this way, the use of *r*-actions promises computational effort savings during the solution of MDPs. However, the scalability problem still remains as a great challenge of these processes, because the solution time has been intractable [19].

4 Our Approach: Association Rules in the solution of MDPs

In our approach, we propose to represent, learn and apply *r*-actions through association rules (ARs) in the solution of MDPs. These rules can be derived from a decision tree, in which the leaves are the applicable actions (*r*-actions) on each state (node or attribute set) and the branches are instances of attributes. The decision tree can be constructed by using techniques such as C4.5 [15], Apriori [1], etc. For instance, the C4.5 algorithm generates a decision tree by using data recursive partitions, by means of a depth-first strategy, considering all the possible tests for dividing datasets, and selecting the best information gain test. For each attribute is considered a test with "n" possible values. For each node, the system decides which test is more convenient for dividing the data. Figure 2 shows a decision tree derived from the problem "Play Golf", based on the weather outlook [15].

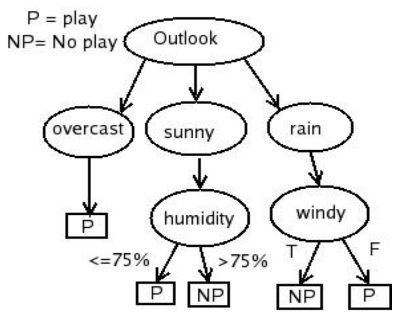

Figure 2. A decision tree.

We can observe that if raining and windy attributes have value T (True), then the decision is <NP> (NoPlay).

Summarizing, AR’s can be formulated from a decision tree. In our case, we obtain ARs by means of the Apriori algorithm, because this one yields ARs in the format required by the value iteration algorithm [16]. This algorithm calculates and selects the attribute set having at least the default support and confidence (percentage of examples covered by a rule). Afterwards, Apriori gives the best rules found, with maximum support and confidence. This means that this algorithm is capable of obtaining the set of *r*-actions required by our method.

Following with the example shown in Figure 2, the Apriori algorithm found the next best four ARs:

```
outlook=overcast 4 ==> play=yes 4    confidence:(1)
humidity=normal windy=FALSE 4 ==> play=yes 4 confidence:(1)
outlook=sunny humidity=high 3 ==> play=no 3    confidence:(1)
outlook=rainy windy=FALSE 3 ==> play=yes 3    confidence:(1)
```

In those rules, the numbers 4 and 3 indicate the support of the rule and the number 1 indicates the 100% of confidence of the rule. The first rule can be written as:

if <overcast> **then** <play>.

Then, for the reformulation of the value iteration algorithm in terms of ARs, we defined ARs which contain 3 attributes: initial state, final state and *r*-action.
Let $L = \{L_k \mid L_k = (s_k, s_k^{'}, a_k)\}$ be the set of ARs, $R = [R_1, R_2, ..., R_n]$ be the state rewards, $T = [T_1, T_2, ..., T_n]$ be the transition probabilities (such that $\forall k \; T_k > 0$) of each rule where n is the number of states. The modified value iteration algorithm [16] for the use of AR is the following:

Algorithm 1. RulesMDP.

```
function RulesMDP$(R, L, T, \gamma, \varepsilon, NumIt)$
$U_i^0 = R_i$ for $i = 1, 2, ..., n$
$t = 1$
$J(s, a) = 0$ for $s = 1, 2, ..., n$ and $a = 1, 2, ..., m$
do
    for $k = 1$ to $|L|$
        $a = action(L_k)$
        $s = initialState(L_k)$
        $s' = finalState(L_k)$
        $J(s, a) = J(s, a) + T_k U_{s'}^{t-1}$
    end
    for $s = 1$ to $n$
        $U_s^t = R_s + \gamma \max_a J(s, a)$
        $\pi_s = \arg\max_a J(s, a)$
    end
    $t = t + 1$
while $t \le NumIt$ and $\left[\left(\frac{1}{n}\right)\Sigma\left(U_s^t - U_s^{t-1}\right)^2\right]^{\frac{1}{2}} > \varepsilon$
return $\pi$
```

We can observe that the modified value iteration algorithm first calculates the expected utility $J(s,a)$ for each rule L_k and its corresponding transition probability T_k. Then, it calculates the maximum expected utility and the optimal policy from $J(s,a)$. Finally, after several iterations, it returns the optimal policy that gives maximum utility (present and future) in the current state, π.

For a given number of states n_s, number of actions n_a, and number of iterations n_{it}, the number of rules is bounded by $|L| \le n_a n_s$. Thus the complexity of the algorithm is $T(n_s, n_a, n_{it}) \in O(n_s n_a n_{it})$. An upper bound of the required number of iterations [12] is given by:

$$n_{it} \le \frac{B + \log(\frac{1}{\varepsilon}) + \log(\frac{1}{1-\gamma}) + 1}{1 - \gamma} \tag{2}$$

where B is the number of bits used to encode instantaneous costs and state transition probabilities, ε is the threshold of the Bellman residual and γ is the discount factor. For large n_s, the number of iterations is smaller than the number of states, which means that the complexity is a linear function of the number of states. In this way, our algorithm can solve quickly to the MDPs. In order to apply our algorithm, we first create a file with instances generated by means of a random walk over the whole state space and the whole action domain. Next, the instances file is used to generate the set of ARs by using the Apriori algorithm. Finally, the state transition probabilities are estimated by using dynamic Bayesian networks [10]. All rules for which the transition probability is cero are eliminated.

5 Experimental Results

We tested on a robot path planning task, and for each experiment we set up a bi-dimensional grid-world where an autonomous agent is moving with 5 possible actions. For all the experiments, we set $\gamma = 0.9$, $\varepsilon = 1\times10^{-6}$, $B = 32$, and $n_{it} = 500$ (given by equation (2)). The grid contained from 25 to 400 states, and they were associated with different number of rewards (see Table 2). We repeated 120 times every grid setup of combinations of the parameters. Our algorithm was implemented in *Java* language inside the simulator for planning of robotic movements, SPRM [18] and we compared the performance of both algorithms, our algorithm (RulesMDP) and the previous value iteration algorithm (PREV) in the SPRM environment. All the experiments were performed on a 2.66 GHz Pentium D computer with 1 GB RAM. The combinations are presented in Table 2. The policies achieved with our algorithm were very similar (with some different visited states, but with the same goal state and with the same number of steps in the obtained policies) to the ones obtained with the previous value iteration algorithm. The time spent in calculating the rules was smaller than the time taken for a single iteration of the proposed algorithm.

Table 2. Results obtained in SPRM: PREV vs. RulesMDP.

No. of rewards	No. of states	Time (ms) PREV	No. of operations PREV	Time (ms) RulesMDP	No. of operations RulesMDP
2	25	47	375000	12	14625
4	25	47	384375	12	14750
10	100	94	6000000	12	60000
26	100	94	6200000	15	60500
6	225	389	30261000	21	128625
10	225	396	30832000	21	128900
6	400	1195	96000000	31	224000
10	400	1266	102400000	31	254000
12	400	2969	242187500	266	1512500

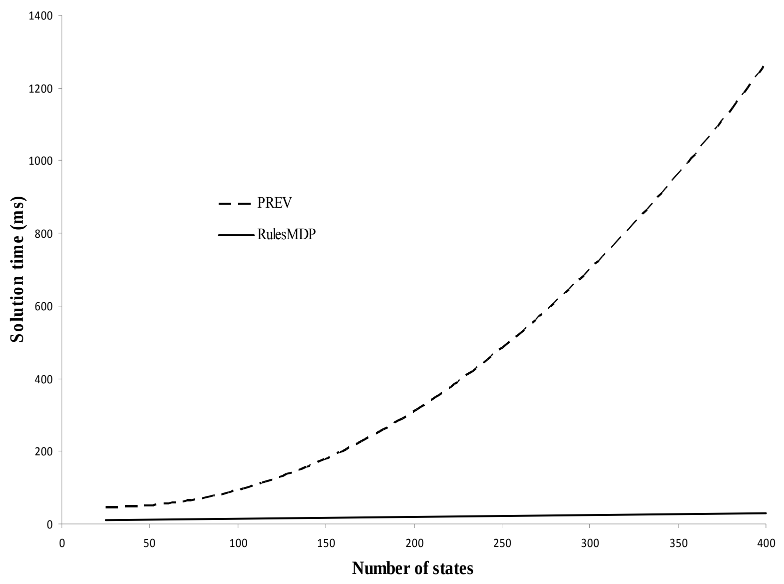

Figure 3. Solution time for previous algorithm (PREV) vs. our algorithm (RulesMDP).

In Table 2, we can see that when the number of states is 25, the solution time required by the previous algorithm is 47 ms, while it is 12 ms for our algorithm. Otherwise, when the number of states is 400, the solution time by the previous algorithm is 1266 ms, while it is 31 ms for our algorithm (it includes the calculation of the corresponding ARs). In both cases, our RulesMDP algorithm is faster than the previous one, whereas in the same table the solution time increases softly when the number of rewards grows, for each number of states. Results obtained in Table 2 are presented in Figure 3 for each method. This figure shows how the solution time required by the previous algorithm increases quadratically with the number of states, whereas the solution time required by our algorithm increases linearly. Otherwise, in the same table is shown that an increase on the number of rewards has not significant effect when the number of states grows. Summarizing, at least in our experiments, we obtained a considerable reduction in the solution time of MDPs when we applied ARs.

6 Conclusions and related works

Markov decision processes [16] have successfully solved problems of process control, but for problems with continuous or high dimensionality domains, high computational complexity arises, because the search space grows exponentially with the number of variables. In order to reduce computational complexity in these processes, we have proposed a new value iteration algorithm based on ARs for solving MDPs. The proposed technique uses the Apriori algorithm for generating rules to build the *r*-action set. At least in our experimental results we found that the solution time was linear in the number of states and our algorithm was faster than the previous value iteration algorithm [16]. Policies achieved with our method had the same goal state and the same number of steps in the resulting policies as with the previous value iteration algorithm. In the literature there are two related works: the grid based discretization and the parametric approximations. Unfortunately, in the classic grid algorithms the search space grows exponentially with the number of actions [5], and the parametric approximation based on hierarchical clustering techniques [14] has been used with little success.

References

1. Agrawal, R., Srikant, R., Fast Algorithms for Mining Association Rules, Proceedings of the 20th VLDB Conference, IBM Almaden Research Center (1994)
2. Bellman, R.E., Dynamic Programming, Princeton United Press, Princeton, USA (1957)
3. Benson, S.S., Learning action models for reactive autonomous agents, PhD Thesis, University of Stanford (1996)
4. Bertsekas, D.P., Dynamic Programming, Prentice Hall, Eaglewood Cliffs, MA, USA (1987)
5. Bonet, B., Pearl, J., Qualitative MDP and POMDP: An order-of-magnitude approach, Proceedings of the 18th Conference on Uncertainty in AI, Edmonton, Canada, pages 61-68 (2002)
6. Bonet, B., Geffner, H., Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings and its application to MDP, Proceedings of ICAPS (2006)
7. Boutilier, Craig, Dean, T., Hanks, S., Decision-theoretic planning: structural assumptions and computational leverage, Journal of AI Research, 11, pages 1-94 (1999)
8. Boutilier, C., Dearden, R., Goldszmidt, M., Stochastic Dynamic Programming with factored representations, Artificial Intelligence, 121(1-2) pages 49-107 (2000)
9. Darwiche, A., Goldszmidt M., Action networks: A framework for reasoning about actions and change under understanding, Proceedings of the 10th Conference on Uncertainty in AI, UAI-94, pages 136-144, Seattle, USA (1994)
10. Dean, T., Givan, R., Model minimization in Markov Decision Processes, Proceedings of the 14th National Conference on AI, pages 106-111 (1997)
11. Hauskrecht, M., Kveton, B., Linear program approximations for factored continuous- states Markov Decision Processes, Proceedings of the NIPS (2003).
12. Littman, M. L., Dean, T. L. and Kaelbling, L. P., On the Complexity of Solving Markov Decision Problems, Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence, pages 394-402 (1995)
13. Morales, E., Scaling up reinforcement learning with a Relational Representation, Proceedings of the Workshop on Adaptability in MultiAgent Systems, AORC, Sydney, Australia (2003)
14. Pineau, J., Gordon, G., Thrun, S., Policy-contingent abstraction for robust control, Proceedings of the 19th Conference on Uncertainty in AI, pages 477-484 (2003)
15. Quinlan, J.R., C4.5: Programs for machine learning, Morgan Kaufmann, San Francisco, CA, USA (1993)
16. Puterman, M.L., Markov Decision Processes, Wiley Editors, New York, USA (1994)

17. Quinlan, J.R., C4.5: Programs for machine learning, Morgan Kaufmann, San Francisco, CA, USA (1993)
18. Reyes, A., Ibarguengoytia, P., Sucar, L.E., Morales, E., Abstraction and Refinement for Solving Continuous Markov Decision Processes, 3rd European Workshop on Probabilistic Graphical Models, Czech Republic, pages 263-270 (2006)
19. Van Otterlo, M., A Survey of Reinforcement Learning in Relational Domains, Technical Report Series CTIT-05-31, ISSN 1381-3625, July (2005)