NMRDPP: Decision-Theoretic Planning with Control Knowledge.

Charles Gretton, David Price, and Sylvie Thiébaux
Computer Sciences Laboratory
The Australian National University
Canberra, ACT, Australia
{charlesg,davidp,thiebaux}@csl.anu.edu.au

Abstract

We discuss NMRDPP, a system for solving decision processes with non-Markovian reward. More specifically, target decision processes exhibit Markovian dynamics and rewarding behaviours are modelled as state trajectories specified in a linear temporal logic. In addition to implementing structured, tabular and online MDP solution algorithms, NMRDPP can exploit domain specific control knowledge. State trajectories which violate the users knowledge/intuition regarding useful dynamics can be pruned from consideration by the MDP solution algorithm. Thus, in addition to facilitating concise specification of complex reward structures, NMRDPP can be used to greatly speed up policy computation for propositional MDPs. To our knowledge, NMRDPP is the only implementation of solution algorithms designed to solve decision processes with non-Markovian rewards.

Introduction

NMRDPP (Gretton *et al.* 2003) (non–Markovian Reward Decision Process Planner), is a general purpose planner for non-Markovian reward[1] (and hence also Markovian) propositional decision processes. Target decision processes are usually stochastic, exhibiting Markovian dynamics. The reward is modelled as a set of state trajectories, called behaviours, specified in a linear temporal logic. NMRDPP was originally developed in order to carry out an experimental evaluation of approaches for solving decision processes with non-Markovian reward. Implemented in C++, NMRDPP supports a range of experimental algorithms and frameworks for solving NMRDPs. It is suited to participation in IPPC'04 as it facilitates planning in completely observable stochastic domains. NMRDPP is the first of its kind; previously no approaches to solving NMRDP had been fully implemented, and there was no work presenting any experimental results.

There have been two proposals regarding languages suitable for expressing rewarding behaviours. These include PLTL (Bacchus *et al.* 1996) a linear temporal logic of the past and \$FLTL (Thiébaux *et al.* 2002) a linear temporal logic of the future with reward. In either case, NMRDPP translates NMRDPs into corresponding equivalent MDPs (XMDPs) which incorporate temporal variables capturing sufficient history to make the reward of the expanded process Markovian[2]. Available translation procedures are unique and not particularly straightforward (Bacchus *et al.* 1996; Bacchus *et al.* 1997; Thiébaux *et al.* 2002). NMRDP solution algorithms differ in their representations of domain dynamics, the XMDP and in the class, structured or non-structured, of MDP solution methods to which they are tied. NMRDPP can solve target decision problems online (during translation) using LAO* heuristic search techniques (Hansen and Zilberstein 2001). Alternatively, the complete XMDP can be generated and passed to classical structured or tabular policy computation algorithms such as SPUDD (Boutilier *et al.* 1995; Hoey *et al.* 1999) or policy/value iteration (Howard 1960) respectively.

Using the same mechanisms devised for non-Markovian reward, state trajectories which violate the users knowledge/intuition regarding useful dynamics can be pruned from consideration by the MDP solution algorithm. The specification of a set of such state sequences is called *control knowledge*, and has been used to great effect by the deterministic planning community (Bacchus and Kabanza 2000). Thus, although there is no advantage to be gleaned from concise specification of complex non-Markovian reward during the competition, NMRDPP can exploit control knowledge to greatly speed up policy computation given propositional MDPs. By pruning states which violate specific behaviours, we can mitigate the effect of Bellman's so called curse of dimensionality.

In the remainder of this document, we shall present an overview of MDPs and NMRDPs and discuss their differences. We shall briefly discuss the logics that have been adopted to model reward and control knowledge, focusing in particular on \$FLTL. We shall provide some examples of using \$FLTL to specify control knowledge for a stochastic blocks-world domain. We shall conclude by summarising how we intend to compete using NMRDPP in the IPPC'04.

MDPs and NMRDPs

Problem domains which participants shall consider during the *main* and *learning* IPPC'04 tracks, although specified in PPDDL1.0 (Younes and Littman 2004), can be modelled using the MDP formalism. Indeed, decision theoretic

[1] For our purposes reward can be negative, thus we don't distinguish between reward and cost.

[2] There is a mapping from XMDP states to the reals.

planning problems are typically modelled as propositional MDPs such that domain states S are characterised by propositions, numeric reward is allocated to propositions/states according to their associated desirability and the dynamics of the system is given by actions A. We typically write $A(s)$ to denote actions applicable at state s. A solution algorithm, provided with start states $S_0 \subseteq S$, generates a stationary policy $\pi : S \rightarrow A$ (mapping from states to actions) which adherence to during system execution results in optimal behaviour over a discounted infinite horizon.

The standard MDP formulation is state based, comprising a *finite* set of states S and actions A. Actions induce stochastic state transitions, where $s, t \in S$, $a \in A$ and $Pr(s, a, t)$ gives the probability of a transition from state s to t given action a is executed at state s. Also present is a real-valued reward function $R : S \rightarrow \Re$. The value of a stationary policy π at a state $s_0 \in S_0$, $V(\pi)$, is given by Equation 1.

$$V(\pi) = \lim_{n \rightarrow \infty} \mathsf{E}\left[\sum_{i=0}^{n} \beta^i R(\Gamma_i) \mid \pi, \Gamma_0 \in S_0\right] \quad (1)$$

Here β is a discount factor usually close to 1 and $\Gamma \in S^*$ is a finite sequence of states where Γ_i is the i'th state in Γ. We consider a policy π^* optimal if, for all π, we have that $V(\pi^*) \geq V(\pi)$.

The formulation for NMRDPs is identical up to the reward function whose domain is extended to S^*, e.g. $R : S^* \rightarrow \Re$. Here, $\Gamma(i)$ is the i length subsequence of Γ starting at Γ_0. As before, the value of π, which we seek to maximise, is the expectation of the discounted cumulative reward over an infinite horizon:

$$V(\pi) = \lim_{n \rightarrow \infty} \mathsf{E}\left[\sum_{i=0}^{n} \beta^i R(\Gamma(i)) \mid \pi, \Gamma_0 \in S_0\right] \quad (2)$$

As introduced, NMRDP solution methods facilitate generation of an optimal policy by first expanding the NMRDP into an XMDP, and then applying either traditional or structured MDP solution algorithms to the resulting construct.

Reward Specification and Control Knowledge

NMRDPP supports the use of two linear temporal logics in the specification of rewarding state trajectories and control knowledge. These are PLTL (Bacchus *et al.* 1996) and \$FLTL (Thiébaux *et al.* 2002). The logic PLTL includes the modalities $\ominus$ (previously), S (since), $\diamondsuit f \equiv \top \mathsf{S} f$ (once) and $\boxminus f \equiv \neg\diamondsuit\neg f$ (always in the past) while \$FLTL includes $\bigcirc\phi$ (next), U (weak until), $\Box\phi \equiv \phi \mathsf{U} \bot$ (always), and a propositional constant \$ (receive reward now)[3].

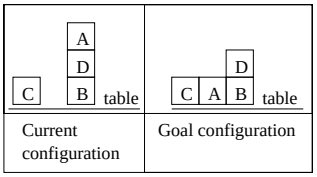
A translation from an NMRDP into a corresponding MDP is based on the fact that a PLTL, resp. \$FLTL, wff ϕ can be regressed (Bacchus and Kabanza 2000), resp. progressed, to a formula which identifies what must hold in the past, resp. future, for ϕ to hold in a current state. Given this progression/regression operator, methods annotate grounded states to form expanded states with formulae (temporal variables) which are sufficient to determine the reward allocation at any state reachable from S_0. Methods are characterised by the properties of the XMDP which they generate. Given a PLTL reward specification NMRDPP can attempt to generate the minimal MDP required to allocate reward given specified behaviours. Using the language \$FLTL, NMRDPP is able to produce a blind minimal XMDP online. Intuitively, a blind minimal XMDP is the smallest MDP achievable by online translation.

As a derivative of FLTL, \$FLTL is particularly suitable for expressing domain specific control knowledge (Bacchus and Kabanza 2000) which is useful in the context of online solution algorithms. That is, a decision process can be modified by excluding from it sequences of states which violate a control hypotheses expressed in \$FLTL.

Blocks-World with \$FLTL Control Knowledge

We have that a stochastic version of blocks-world will feature in the learning track at IPPC'04. Using the language of PPDDL1.0, BW has `block` and `table` types, predicates `holding` : `block` and `On` : (`block` $\times \top$), and two action symbols `pick-up` : (`block` $\times \top$) and `put-down` : (`block` $\times \top$). A BW state comprises two ground sets of predicate symbols, those which characterise the current and goal states. Pictorially, a BW state appears as follows:

"$\forall b.\neg$`holding`(b)"
Goal:`On`(D, B), `On`$(B,$ `table`$)$, $\ldots$, `On`$(C,$ `table`$)$
Conf:`On`(A, D), `On`(D, B), $\ldots$, `On`$(C,$ `table`$)$



We have that the `pick-up`(a, b) action is only executable if `table` $= a \vee \nexists$`block`(c).`On`(c, a) (i.e. a is clear). Similarly `put-down`(a, b) is only available if `holding`(a) and b is clear. Assuming precondition satisfaction, `pick-up`(a) either causes `block` a to be held, or possibly fall on the `table`. The action `put-down`(a, b) either drops a on the table or places it on the second argument object b.

This stochastic BW isn't particularly different from its deterministic relatives. Thus, we can appeal to near optimal planning strategies such as US and GN1 (Slaney and Thiébaux 2001) in developing control knowledge. For the purposes of this presentation we introduce the predicate `InPosition`(a, b) for a and b of type `block`. `InPosition`(a, b) is false if `On`(a, b) is not an element of the goal configuration, and otherwise true when a and b are in their goal position. Notice that this is a derived predicate, i.e. given our example state, `InPosition`$(D, B) \equiv$ `On`$(D, B) \wedge$ `On`$(B,$ `table`$)$. Thus, the following control knowledge is expressible without change to the competition specification.

The first piece of control knowledge that we consider prevents NMRDPP from disturbing towers of blocks which satisfy the goal. For each `On`(b_i, b_j), a control sentence of the following form is sufficient:

[3] See the respective papers for a comprehensive summary of the two logics.

$$\Box(\quad (\texttt{On}(b_i, b_j) \wedge \texttt{InPosition}(b_i, b_j)) \\ \rightarrow (\bigcirc\neg\texttt{holding}(b_i)))$$

By pruning from a BW domain states which violate the above sentence, NMRDPP will not consider policies which disturb blocks that are in their goal position. We can further prune the range of policies which NMRDPP shall consider by noticing that if $\texttt{On}(b_i, b_j)$ is false where b_j is a `block`, and after two action invocations $\texttt{On}(b_i, b_j)$, then this is only valuable where $\texttt{InPosition}(b_i, b_j)$. This knowledge is expressed in $FLTL as follows.

$$\Box(\quad (\neg\texttt{On}(b_i, b_j) \wedge \bigcirc\bigcirc(\texttt{On}(b_i, b_j))) \\ \rightarrow (\bigcirc\bigcirc\texttt{InPosition}(b_i, b_j)))$$

Annotating BW problems with the above control knowledge greatly increases the situations in which NMRDPP is competitive. Because progression of $FLTL formulae is linear time in the formula length, it is important that we avoid crippling NMRDPP by providing too much, mostly redundant or too complex knowledge. Furthermore, we must ensure that knowledge generation for competition domains is practical.

Participation

NMRDPP, provides an implementation of several solution approaches in a common framework, within a single system, and with a common input language. The framework includes a highly interactive command line interface which allows the user to exert fine control over the planning process. The input language enables specification of actions, initial states, rewards, and control-knowledge. Initial states are specified as part of the control knowledge or as explicit assignments to propositions. Of interest to us here is the format for the action specification, which is essentially the same as in the SPUDD system (Hoey *et al.* 1999). In particular, the precondition (BDD), reward and probabilistic effects for each action are specified by a collection of decision trees, including one for each domain proposition which the action effects. When the input is parsed, the action specification trees are converted into ADDs by the CUDD package (Somenzi 2001).

The input language in which competition domains are specified is PPDDL1.0. We will be able to accommodate this in one of two ways. 1) Because both NMRDPP and the competition software code are implemented in C++, we can directly take advantage of competition code which facilitates exploration of the explicit propositionalised state space. In this case we restrict NMRDPP to state-based solution algorithms. 2) We can translate PPDDL1.0 problem specifications into the NMRDPP input language. This is an enticing option because the competition code contains functionality implemented by Håkan L. S. Younes which encodes grounded problem actions as ADDs. There is insufficient space for us to include the details here, however such grounded action ADDs can be converted into action descriptions in the NMRDPP input format. In essence, such a process extracts diachronic and synchronic dependencies between propositions in an action's conditional probability model in order to construct decision trees/ADDs encoding action preconditions, reward and effects on individual propositions. Where we generate NMRDPP input from PPDDL1.0, the action specifications are not concise[4] as information regarding pre/post-action-variable dependencies is lost in translation. The advantage however, is that we do not restrict ourselves to a subset of solution algorithms supported by NMRDPP.

We intend to enter NMRDPP in both the *main* and *learning* tracks of IPPC'04. In the *main* track we do not expect much from NMRDPP as it is ladened with some overhead due to support for non-Markovian reward. For the *Learning Track*, we shall develop "hand coded" control knowledge specific to each competition domain in order to make NMRDPP competitive. Although this does not position us well to compete with first-order learners which are not restricted to a propositional domain model, we hope to be competitive in small to medium domain instances. At the time of writing, it was not clear which of the structured, tabular and online algorithms NMRDPP supports, we shall use in the competition.

References

F. Bacchus and F. Kabanza. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2), 2000.

F. Bacchus, C. Boutilier, and A. Grove. Rewarding behaviors. In *Proc. AAAI-96*, pages 1160–1167, 1996.

F. Bacchus, C. Boutilier, and A. Grove. Structured solution methods for non-markovian decision processes. In *Proc. AAAI-97*, pages 112–117, 1997.

C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proc. IJCAI-95*, pages 1104–1111, 1995.

Charles Gretton, David Price, and Sylvie Thiébaux. Implementation and comparison of solution methods for decision processes with non-markovian rewards. In *Proc. UAI-03*, 2003.

E. Hansen and S. Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129:35–62, 2001.

J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: stochastic planning using decision diagrams. In *Proc. UAI-99*, 1999. SPUDD is available from http://www.cs.ubc.ca/spider/staubin/Spudd/.

R.A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

J. Slaney and S. Thiébaux. Blocks world revisited. *Artificial Intelligence*, 125:119–153, 2001.

F. Somenzi. CUDD: CU Decision Diagram Package. Available from ftp://vlsi.colorado.edu/pub/, 2001.

S. Thiébaux, F. Kabanza, and J. Slaney. Anytime state-based solution methods for decision processes with non-markovian rewards. In *Proc. UAI-02*, pages 501–510, 2002.

H. Younes and M. Littman. PPDDL1.0: An extension to PDDL for Expressing Planning Domains with Probabilistic Effects, 2004. *http://www.cs.cmu.edu/ lorens/papers/ppddl.pdf*.

[4] We find that for large domains, i.e. BW with $10 >$ blocks for example, a translation into NMRDPP input format is impractical.