NMRDPP: A System for Decision-Theoretic Planning with Non-Markovian Rewards*

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 present the NMRDPP system, which implements under a common interface a range of methods for decision-theoretic planning with Non-Markovian rewards. We have used NMRDPP to compare the various approaches and identify certain problem features favouring one over the other.

Motivation

A decision process in which rewards depend on the sequence of states passed through rather than merely on the current state is called a decision process with *non-Markovian rewards* (NMRDP). Non-Markovian rewards are to decision-theoretic planning what temporally extended goals are to non-deterministic planning. They are particularly useful since many desirable behaviours are more naturally expressed as properties of execution sequences rather than as properties of states.

The more tractable solution methods developed for Markov decision processes (MDPs) do not directly apply to NMRDPs. However, a number of solution methods for NMRDPs have been proposed in the literature (Bacchus *et al.* 1996; 1997; Thiébaux *et al.* 2002). These all start with a temporal logic specification of the non-Markovian reward function, which they exploit to automatically translate the NMRDP into an equivalent MDP whose states include extra information capturing enough history to make the reward Markovian. This MDP can then be solved using efficient MDP solution methods.

The above approaches differ in mainly 3 respects. Firstly, they consider different representation and solution methods for the equivalent MDP. Specifically PLTLSIM and PLTLMIN, the approaches in (Bacchus *et al.* 1996), target *state-based* representations and classical solution methods such as value or policy iteration. FLTL, the approach in (Thiébaux *et al.* 2002), also considers state-based representation but targets heuristic search methods such as LAO* or labelled RTDP. Finally, PLTLSTR, the approach in (Bacchus *et al.* 1997) considers *structured* representations and solution methods such as structured policy iteration or SPUDD. Secondly, these different targets lead the approaches to adopt very different types of translation, as appropriate. For instance, since state-based solution methods are very sensitive to the size of the MDP, state-based approaches such as PLTLMIN and FLTL require a sophisticated translation producing an equivalent MDP as small as possible, without irrelevant history distinctions. On the other hand, for a structured approach like PLTLSTR, a very crude translation is enough because structured representation and solution methods have their own ability to ignore irrelevant information. Thirdly, the approaches consider different temporal logics to express non-Markovian rewards, as suited to the type of translation chosen. For instance, PLTLSIM, PLTLMIN and PLTLSTR use *linear temporal logic* with *past* operators (PLTL), as this yields a straightforward semantics of non-Markovian rewards. FLTL on the other hand, relies on a more complex extension of LTL with *future* operators ($FLTL), as it naturally leads to a style of translation suited to the needs of heuristic search methods.

The literature lacks any report of implementation or comparison of these approaches. Our goal in developing NMRDPP (NMRDP planner) was therefore to provide a first implementation of all of them, and to do this in a common framework, within a single system and with a common input language, so as to facilitate their experimental comparison.

System Description

NMRDPP's input language enables specification of actions, initial states, rewards, and control-knowledge. The format for the action specification is essentially the same as in the SPUDD system.[1] When the input is parsed, the action specification trees are converted into algebraic decision diagrams (ADDs) by the CUDD package.[2] The reward specification is one or more formulae, each associated with a real. These formulae are in either PLTL or $FLTL and are stored as trees by the system. Control knowledge is given in the same language as that chosen for the reward. Control knowledge formulae will have to be verified by any sequence of states feasible under the generated policies. Initial states are simply specified as part of the control knowledge or as explicit assignments to propositions.

For instance, consider a simple example consisting of a

*This demonstration supports the technical paper (Gretton *et. al* 2003) too be presented at the ICAPS-03 workshop on Planning under Uncertainty and Incomplete Information.

[1] http://www.cs.ubc.ca/spider/staubin/Spudd/.

[2] http://vlsi.colorado.edu/ fabio/CUDD/cuddIntro.html

```
action flip
   heads (0.5)
endaction
action tilt
   heads (heads (0.9) (0.1))
endaction
heads = ff
[first, 5.0]? heads and ~prv (pdi heads)
[seq, 1.0]? (prv^2 heads) and (prv heads) and ~heads
```

Figure 1: Input for the coin example

coin showing either heads or tails (¬heads). There are two actions that can be performed. The flip action changes the coin to show heads or tails with a 50% probability. The tilt action changes it with 10% probability, otherwise leaving it as it is. The initial state is tails. We get a reward of 5.0 for the very first head (this is written heads $\wedge \neg \ominus \Diamond$heads in PLTL) and a reward of 1.0 each time we achieve the sequence heads, heads, tails ($\ominus^2$heads $\wedge \ominus$heads $\wedge \neg$heads in PLTL). In our input language, this NMRDP is described as shown in Figure 1.

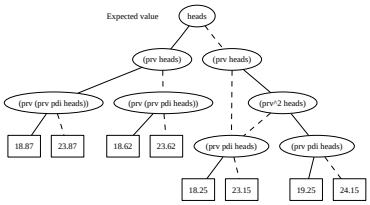
The common framework underlying NMRDPP takes advantage of the fact that NMRDP solution methods can, in general, be divided into the distinct phases of preprocessing, expansion, and solving. The first two are optional. For instance, for PLTLSTR, *preprocessing* involves computing a set of temporal variables to be included in the states of the equivalent MDP, as well as the ADDs describing (1) their dynamics and (2) the rewards. *Expansion* is the optional generation of the entire equivalent MDP prior to solving. Whether or not off-line expansion is sensible depends on the MDP solution method used. If state-based value or policy iteration is used, as in PLTLMIN, then the MDP needs to be expanded anyway. If, on the other hand, a heuristic search algorithm or structured method is used, as in FLTL or PLTLSTR, it is definitely a bad idea. In our experiments, we often used expansion solely for the purpose of measuring the size of the equivalent MDP. *Solving* the equivalent MDP can be done with a number a methods. Currently, NMRDPP provides implementations of classical dynamic programming methods, namely state-based value and policy iteration, of heuristic search methods: state-based LAO* using either value or policy iteration as a subroutine, and of structured methods based on SPUDD. Altogether, the various types of preprocessing, the choice of whether to expand, and the MDP solution methods, give rise to quite a number of NMRDP approaches, including, but not limited to those previously mentioned.

NMRDPP is controlled by a command language, which is read either from a file or interactively. The command language provides commands for different phases of the different algorithms, commands to inspect the resulting policy and value functions, e.g. with rendering via DOT[3] as well as supporting commands for timing and memory usage. A sample session[4] may be as in Figure 2.

[3]http://www.research.att.com/sw/tools/graphviz/.

[4]Oval ADD nodes represent a dependence on a particular variable. Solid lines leaving an oval represent the corresponding variable being true, dashed lines represent it being false. The rectangular boxes at the leaves represent the value of the path followed (or, for a policy, the action associated to it).

```
> loadWorld('coin.world')          load coin NMRDP
> preprocess('sPLTL')              PLTLSTR preprocessing
> startCPUtimer
> spudd(0.99, 0.0001)              solve MDP with SPUDD(β, ε)
converged after 1277 iterations with delta= 5.00e-07
> stopCPUtimer
> readCPUtimer                     report solving time
1.22000
> displayDot(spuddValueToDot)      display ADD of value function
```




```
> displayDot(spuddPolicyToDot)     display policy
```

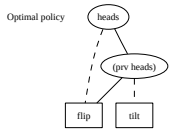


Figure 2: Sample session

NMRDPP is implemented in C++, and makes use of a number of supporting libraries. In particular, the structured algorithms rely heavily on the CUDD library for representing ADDs. The non-structured algorithms make use of the MTL—Matrix Template Library for matrix operations. MTL takes advantage of modern processor features such as MMX and SSE and provides efficient sparse matrix operations. We believe that our implementations of MDP solution methods are comparable with the state of the art.

Results

NMRDPP proved a useful tool in the experimental analysis of approaches for decision processes with Non-Markovian rewards. In (Gretton *et al.* 2003), we use it to compare their behaviours under the influence of various factors such as the structure and degree of uncertainty in the dynamics, the class of rewards and the syntax used to describe them, reachability, and relevance of rewards to the optimal policy. We were able to identify a number of general trends in the behaviours of the methods and to provide advice as to which are the best suited to certain circumstances. In the future, we will use NMRDPP to see what form these results take in the context of domains of more practical interest.

References

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

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

Gretton, C.; Price, D.; and Thiébaux, S. 2003. NMRDPP: A System for Decision-Theoretic Planning with Non-Markovian Rewards. In *Proc. ICAPS-03 Workshop on Planning under Uncertainty and Incomplete Information.*

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