Properties of Planning with Non-Markovian Rewards *

Charles Gretton
charlesg@csl.anu.edu.au

Abstract

We examine technologies designed to solve decision processes with non-Markovian rewards (NMRDPs). More specifically, target decision processes exhibit Markovian dynamics, called *grounded dynamics*, and desirable behaviours are modelled as state trajectories specified in a temporal logic. Each technology operates by automatically translating NMRDPs into corresponding equivalent MDPs amenable to classical MDP solution methods. They do however differ in their representations of grounded dynamics, the MDP and in the class, structured or non-structured, of MDP solution methods to which they are suited. Therefore two temporal logics and numerous translation procedures have been adopted. This presentation is based on an integrated system for solving NMRDPs which implements these methods and several variants under a common interface; we use it to compare the various approaches and identify some problem features favouring one over the other.

Introduction

Currently the de facto model for decision-theoretic planning (DTP) is the Markov decision process (MDP) (Boutilier *et al.* 1999). Recently efforts have been made to address weaknesses in this formalisms' provision of a framework for developing plans in a decision-theoretic setting. For many application domains the Markov assumption is a hindrance as desirable control strategies require that reward is allocated to behaviours or sequences of states. The model which has been developed to address this is the non-Markovian reward decision process (NMRDP) (Bacchus *et al.* 1996). Corresponding frameworks provide for system dynamics given by a grounded MDP and desirable behaviour specified using a linear temporal logic (Bacchus *et al.* 1996; Thiébaux *et al.* 2002).

A number of solution methods for NMRDPs have been proposed in the literature (Bacchus *et al.* 1996; Bacchus *et al.* 1997; Thiébaux *et al.* 2002). These methods translate NMRDPs into a corresponding equivalent MDP which incorporates temporal variables capturing sufficient history to make the reward Markovian. These methods differ in both the types of MDP representations and hence solution methods to which they are tied, and in the temporal logic that is adopted for reward specification. For instance, methods which expressly enumerate the MDP states, expanded states, use classic dynamic programming algorithms when solving the expanded decision problem. In these cases two logics are adopted, \$FLTL, a future looking logic, for domains where heuristic search (Hansen and Zilberstein 2001) should prove favourable and PLTL, a temporal logic of the past, for situations where dynamic programming over the entire state space is preferred. In contrast to their state based counterparts *structured* methods avoid state based enumeration. These methods also adopt PLTL as the language for expressing non-Markovian reward.

To date, these approaches do not appear to have been fully implemented, and none of the three cited papers report any experimental results. All, however, agree that the most important item for future work is the implementation and experimental comparison of the respective approaches, with a view to identifying the features that favour one over the other.

This abstract summarises our effort so far in this direction. We begin with a review of NMRDPs and the three cited solution methods. We then describe NMRDPP an integrated system which implements, under a single interface, a family of NMRDP solution methods based on the three approaches, and reports a range of statistics about their performance. Using this system we compare how the methods fare under the influence of various factors such as the class of rewards and the syntax used to describe them, reachability, and relevance of rewards to the optimal policy.

NMRDP Solution Methods

Extending MDPs

We start with some notation and definitions. Given a finite set S of states, we write S^* for the set of finite sequences of states over S, and S^ω for the set of possibly infinite state sequences. Where 'Γ' stands for a possibly infinite state sequence in S^ω and i is a natural number, by 'Γ_i' we mean the state of index i in Γ, and by '$\Gamma(i)$' we mean the prefix $\langle \Gamma_0, \ldots, \Gamma_i \rangle \in S^*$ of Γ.

DTP problems are typically modelled such that domain states S are characterised by propositions P, numeric reward is allocated to propositions/states according to their

*An extended version of this paper is to be presented at ICAPS'03 Workshop on Planning under Uncertainty and Incomplete Information.

associated desirability and the dynamics of the system is given by stochastic actions A. We typically write $A(s)$ to denote actions applicable at state s. A solution algorithm, provided with a start state $s_0 \in S$, 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 state s_0, $V(\pi)$, is given by Equation 1 where β is a discount factor usually close to 1.

$$V(\pi) = \lim_{n \rightarrow \infty} \mathsf{E}\left[\sum_{i=0}^{n} \beta^i R(\Gamma_i) \mid \pi, \Gamma_0 = s_0\right] \quad (1)$$

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. $R : S^* \rightarrow \Re$. As before, the value of π, which we seek to maximise, is the expectation of the discounted cumulative reward over an infinite horizon:

$$V(\pi) = \lim_{n \rightarrow \infty} \mathsf{E}\left[\sum_{i=0}^{n} \beta^i R(\Gamma(i)) \mid \pi, \Gamma_0 = s_0\right] \quad (2)$$

As introduced, NMRDP solution methods facilitate generation of an optimal policy by first expanding the NMRDP into an equivalent MDP, and then applying either traditional or structured MDP solution algorithms to the resulting construct. Before we summarise the three solution algorithms we formally define what it is for an MDP to be an extension of a corresponding NMRDP. MDP $D'{=}\langle S', s'_0, A', \Pr', R'\rangle$ is an expansion of NMRDP $D = \langle S, s_0, A, \Pr, R\rangle$ if there exists a mapping $\tau : S' \mapsto S$ such that:

1. $\tau(s'_0) = s_0$.
2. For all $s' \in S'$, $A'(s') = A(\tau(s'))$.
3. For all $s_1, s_2 \in S$, if there is $a \in A(s_1)$ such that $\Pr(s_1, a, s_2) > 0$, then for all $s'_1 \in S'$ such that $\tau(s'_1){=}s_1$, there exists a unique $s'_2{\in}S'$, $\tau(s'_2){=}s_2$, such that for all $a \in A'(s'_1)$, $\Pr'(s'_1, a, s'_2){=}\Pr(s_1, a, s_2)$
4. For any feasible[1] state sequence Γ for D and any feasible state sequence Γ' for D' such that $\Gamma_0 = s_0$ and $\forall$i $\tau(\Gamma'_i) = \Gamma_i$, we have: $\forall$i $R'(\Gamma'_i) = R(\Gamma(i))$.

Items 1–3 ensure that there is a bijection between feasible state sequences in the NMRDP and feasible expanded state sequences in the MDP. Therefore, a stationary policy for the MDP can be reinterpreted as a non-stationary policy for the NMRDP. Furthermore, item 4 ensures that the two policies have identical values, and that consequently, solving an NMRDP optimally reduces to producing an equivalent MDP and solving it optimally (Bacchus *et al.* 1996).

[1]All transitions along the sequence have non-zero probability.

PLTLMIN and FLTL

We use PLTLMIN to refer to methods which were presented in (Bacchus *et al.* 1996) while FLTL is the name given to that in (Thiébaux *et al.* 2002). Each of these methods are tied to, and responsible for, an explicit enumeration of the expanded state space, yet while the former have adopted a linear logic of the past PLTL (see (Bacchus *et al.* 1996) for a formal semantics) for expressing desirable behaviours, the latter choose \$FLTL (see (Thiébaux *et al.* 2002) for a formal semantics) a logic which, unlike PLTL, is suited to on-line translation. For PLTLMIN calculation of an optimal policy for the resulting MDP is left to classic dynamic programming techniques (Howard 1960) while for FLTL this is left to heuristic search methods such as LAO* (Hansen and Zilberstein 2001) or labelled RTDP (Bonet and Geffner 2003). The logic PLTL includes the modalities $\ominus$ (previously), S (since), $\diamondsuit f \equiv \top \mathsf{S} f$ (previously) 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)[2].

Here 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 such state reachable from s_0. Indeed, methods are characterised by the properties of the equivalent MDP which they generate. PLTLMIN attempts to generate the minimal MDP required to allocate reward given specified behaviours and FLTL produces a blind minimal MDP which is as small as possible given that the entire MDP may never be generated. Intuitively, a blind minimal MDP is the smallest MDP achievable by any on-line translation suited to heuristic search.

PLTLSTR and PLTLSTR(A)

The approach in (Bacchus *et al.* 1997), which we call PLTLSTR, targets structured MDP representations: the transition model, policies, reward and value functions are represented in a compact form, e.g. as trees or algebraic decision diagrams (ADDs)(Boutilier *et al.* 1995; Hoey *et al.* 1999). The reward specification language adopted by this approach, as with PLTLMIN, is PLTL. Here reward formulae and regressions of these are called *temporal* variables[3]. The structured approaches have advantages, transition information is often compactly represented in terms of effects on variables. The primary intuition behind this approach stems from the idea that factoring the state space, both grounded and expanded, into abstract/aggregate states reduces the number

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

[3]The truth of temporal variables at any state are given by a boolean function over temporal variables and state characterising propositions in any predecessor.

of expected values that are calculated during the Bellman backup.

Essentially in this case translation amounts to augmenting the compact representation of the grounded domain with new temporal variables together with the compact representation of (1) their dynamics, e.g. as a tree over the previous values of the relevant variables, and (2) of the non-Markovian reward function in terms of the variables' current values. After translation *structured policy iteration* or the SPUDD algorithm (Hoey *et al.* 1999) is applied to the resulting factored MDP. PLTLSTR(A) is identical to PLTLSTR except it applies constraints to action dynamics which ensure that states which are not reachable from s_0 are not considered.

PLTLSTR and PLTLSTR(A) do not pay any attention to minimality during translation. However the structured solution algorithms that are called upon after this phase have the ability to automatically detect the irrelevance of variables during policy construction. Hence structured methods over time are able to dynamically detect the irrelevance of some states.

The NMRDP Planner

The first step towards a comparison of the different approaches is to have an integrated implementation of them all. We developed the non-Markovian reward decision process planner, NMRDPP which is such a system. NMRDPP is controlled by a command language, which is parsed either from a file or interactively. Commands support different phases of the algorithms, inspection of the resulting policy and value functions, e.g. with rendering via DOT (AT&T Research Labs), as well as timing and memory usage. The input language for specifying domains is compatible with available systems with a similar purpose. Most notably, the format for the action specification is essentially the same as in the SPUDD system (Hoey *et al.* 1999).

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 support for ADDs. The non-structured algorithms make use of the MTL—Matrix Template Library for matrix operations. We believe that our implementations of MDP solution methods are comparable with the state of the art. For instance, we found that our implementation of SPUDD is comparable in performance (within a factor of 2) to the reference implementation (Hoey *et al.* 1999).[4]

Experimental Observations

Altogether, we are faced with three substantially different approaches which are not easy to compare, as their performance will depend on domain features as varied as the type, syntax, and length of the temporal reward formula, the availability of good heuristics and control-knowledge, etc, and on the interactions between these factors. In this section we summarise key observations made in our investigation into the influence of some of these factors. All results were obtained using a Pentium4 2.6GHz GNU/Linux 2.4.20 machine with 500MB of ram.

Influence of Reward Types

The types of reward significantly affect the size of the MDP: certain rewards only make the size of the minimal MDP increase by a constant number of states or a constant factor, while others make it increase by a factor exponential in the length of the formula. Table 1 illustrates this. The third column reports the size of the minimal MDP induced by the formulae on the left hand side.[5]

A legitimate question is whether there is a direct correlation between size increase and (in)appropriateness of the different methods. For instance, we might expect the state-based methods to do particularly well given reward types inducing a small MDP and otherwise badly in comparison with structured methods. Interestingly, this is not always the case, as is demonstrated in Table 1 whose last two columns report the fastest and slowest methods over the range of domains where $||S|| = n^2$ and $1 \leq n \leq 12$ and whose first row contradicts such an expectation. Moreover, although PLTLSTR prevails in the last row, for larger values of n (not represented in the table), it aborts through lack of memory, unlike the other methods.

Before we continue we remark that PLTLSTR typically scales to larger state spaces, inevitably leading it to outperform state-based methods. However, this effect is not uniform: structured solution methods sometimes impose excessive memory requirements which renders them uncompetitive in certain cases, for example where $\ominus^n \phi$, for large n, features as a reward formula.

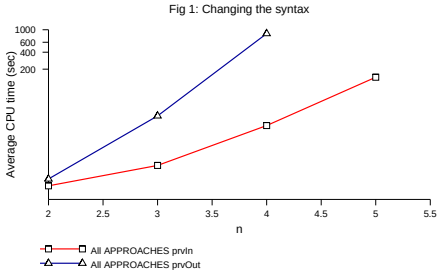
The most obvious result arising out of these experiments is that PLTLSTR is nearly always the fastest—until it runs out of memory. More interesting results are those in the second row, which exposes the inability of methods based on PLTL to deal with rewards specified as long sequences of events. In converting the reward formula to a set of subformulae (*temporal variables*), they lose information about the order of events, which then has to be recovered at great expense during the dynamic programming phase. $FLTL progression in contrast takes the events one at a time, preserving the relevant structure at each step. Further experimentation led us to observe that all PLTL based algorithms perform poorly where reward is specified using formulae of the form $\ominus^k \phi$, $\vee_{i=1}^{k} \ominus^i \phi$, and $\wedge_{i=1}^{k} \ominus^i \phi$ (ϕ has been true k steps ago, within the last k steps, or at all the last k steps).

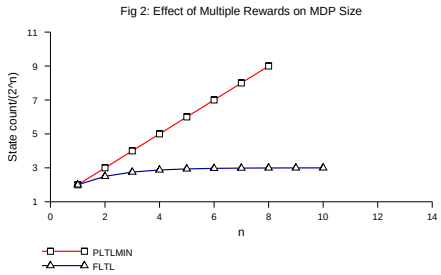
Influence of Syntax

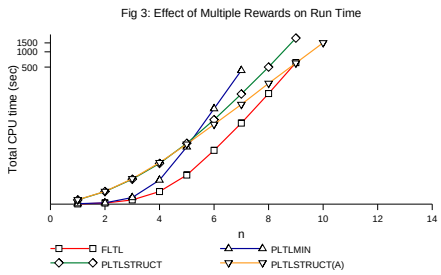
Not surprisingly, for all solution methods we find that the syntax used to express rewards can determine execution time. A typical example of this effect is captured in Figure 1. This graph demonstrates how re-expressing *prvOut* $\equiv \ominus^n(\wedge_{i=1}^{n} p_i)$ as *prvIn* $\equiv \wedge_{i=1}^{n} \ominus^n p_i$, thereby creating n times more temporal subformulae in PLTL, alters the running time

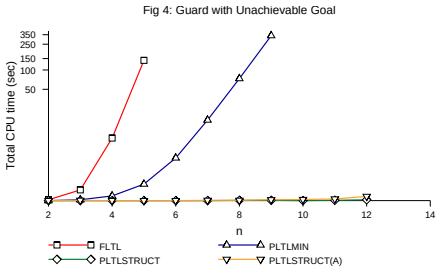
[4] The small difference in performance may be due to our use of high-level C++ CUDD bindings.

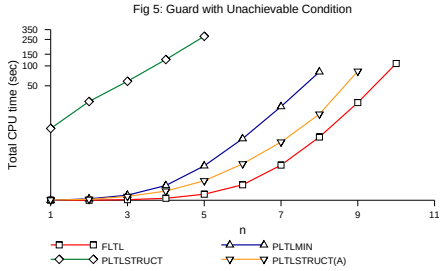
[5] The reported increase in size is not necessarily valid for non-completely connected reflexive NMRDPs, however we found that altering the grounded dynamics did not usually change the reported hierarchy.

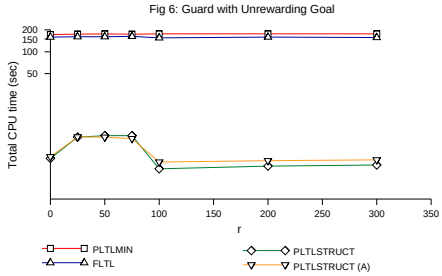


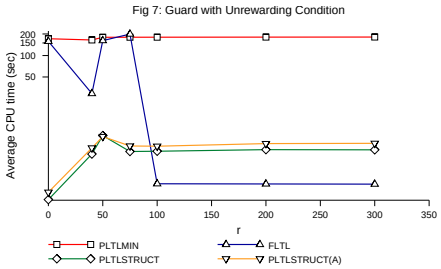












type	formula	size	fastest	slowest
first time all p_is	$(\wedge_{i=1}^{n} p_i) \wedge (\neg \ominus \Diamond \wedge_{i=1}^{n} p_i)$	$\mathcal{O}(1)\|\|S\|\|$	PLTLSTR(A)	PLTLMIN
p_i sequence from s_0	$(\wedge_{i=1}^{n} \ominus^i p_i) \wedge \ominus^n \neg \ominus \top$	$\mathcal{O}(n)\|\|S\|\|$	FLTL	PLTLSTR
two consecutive p_is	$\vee_{i=1}^{n-1} (\ominus p_i \wedge p_{i+1})$	$\mathcal{O}(n^k)\|\|S\|\|$	PLTLSTR	FLTL
all p_is n times ago	$\ominus^n \wedge_{i=1}^{n} p_i$	$\mathcal{O}(2^n)\|\|S\|\|$	PLTLSTR	PLTLMIN

Table 1: Influence of reward type on MDP size and method performance

of all methods. Figure 1 plots the average running time over all the methods for both cases, for a fully connected reflexive grounded dynamics.

Our most serious concern in relation to the PLTL approaches is their handling of reward specifications containing multiple reward elements. Most notably we found that PLTLMIN does not necessarily produce the minimal MDP in this situation. To demonstrate, we consider the set of reward formulae $\{\phi_1, \phi_2, \ldots, \phi_n\}$, each associated with the same numeric reward r. Given this, PLTL approaches will distinguish unnecessarily between past behaviours which lead to identical future rewards. This may occur when the reward at an expanded state is determined by the truth value of $\phi_1 \vee \phi_2$. This formula does not necessarily require that an expanded space distinguish between the cases in which $\{\phi_1 \equiv \top, \phi_2 \equiv \bot\}$ and $\{\phi_1 \equiv \bot, \phi_2 \equiv \top\}$ hold; however, given the above specification, PLTLMIN shall make this distinction. For example, taking $\phi_i = \ominus p_i$, Figure 2 shows that FLTL leads to an MDP whose size is at most 3 times that of the NMRDP. In contrast, the relative size of the MDP produced by PLTLMIN is linear in n, the number of rewards and propositions. These results are typical of all domains where such distinction is not required. Figure 3 shows the run-times as a function of n. FLTL dominates and is only overtaken by PLTLSTR(A) for large values of n, when the MDP becomes too large for explicit exploration to be practical. To obtain the minimal MDP using PLTLMIN, a bloated reward specification of the form $\{\ominus \vee_{i=1}^{n} (p_i \wedge_{j=1, j\neq i}^{n} \neg p_j) : r, \ldots, \ominus \wedge_{i=1}^{n} p_i : n * r\}$ is necessary, which, by virtue of its exponential length, is not an adequate solution.

Influence of Reachability

All approaches, claim to have some ability to ignore variables which are irrelevant because the condition they track is unreachable: PLTLMIN detects them through preprocessing, PLTLSTR exploits structured solution methods dynamic detection of variable relevance, and FLTL ignores them when progression never exposes them. However, given that the mechanisms for avoiding irrelevance are so different, we expect corresponding differences in their effects. We found that the differences in performance are best illustrated by looking at *guard formulae*, which assert that if a trigger condition c is reached then a reward will be received upon achievement of the goal g in, resp. within, k steps. In PLTL, this is written $g \wedge \ominus^k c$, resp. $g \wedge \vee_{i=1}^{k} \ominus^i c$, and in \$FLTL, $\Box(c \rightarrow \bigcirc^k(g \rightarrow \$))$, resp. $\Box(c \rightarrow \wedge_{i=1}^{k} \bigcirc^i(g \rightarrow \$))$.

Where the *goal* g is unreachable, PLTL approaches perform well as g does not lead to behavioural distinctions. On the other hand, while constructing the MDP, FLTL considers the successive progressions of $\bigcirc^k g$ without being able to detect that it is unreachable until it actually fails to happen. This is exactly what the blindness of blind minimality amounts to. Figure 4 illustrates the difference in performance as a function of the number n of propositions involved in a hightly structured grounded domain, when the reward is of the form $g \wedge \ominus^n c$, with g unreachable.

FLTL shines when the *trigger* c is unreachable: $\Box(c \rightarrow \bigcirc^k(g \rightarrow \$))$ will always progress to itself, and the goal, however complicated, is never tracked in the MDP. In this

situation PLTL approaches still consider $\ominus^k c$ and its subformulae, only to discover, after expensive preprocessing for PLTLMIN, after reachability analysis for PLTLSTR(A), and never for PLTLSTR which accommodates every possible starting state, that these are irrelevant. This is illustrated in Figure 5, where the grounded domain is identical to that of Figure 4 and reward is of the form $g \wedge \ominus^n c$, with c unreachable.

Dynamic Irrelevance

(Bacchus *et al.* 1997; Thiébaux *et al.* 2002) claim that one advantage of PLTLSTR and FLTL over PLTLMIN is that the former perform a dynamic analysis of rewards capable of detecting irrelevance of variables to particular policies, e.g. to the optimal policy. Our experiments confirm this claim. However, as for reachability, whether the goal or the triggering condition in a guard formula becomes irrelevant plays an important role in determining whether a PLTLSTR or FLTL approach should be taken: PLTLSTR is able to dynamically ignore the goal, while FLTL is able to dynamically, via heuristic search, ignore the trigger.

This is illustrated in Figures 6 and 7. In both figures, the domain considered is a slightly stoachastic domain suited to structured representation in which $||S|| = 36$, the guard formula is $g \wedge \ominus^n c$ as before, here with both g and c achievable. This guard formula is assigned a fixed reward. To study the effect of dynamic irrelevance of the goal, in Figure 6, achievement of $\neg g$ is rewarded by the value r. In Figure 7, on the other hand, we study the effect of dynamic irrelevance of the trigger and achievement of $\neg c$ is rewarded by the value r. Both figures show the runtime of the methods as r increases.

Achieving the goal, resp. the trigger, is made less attractive as r increases up to the point where the guard formula becomes irrelevant under the optimal policy. When this happens, the run-time of PLTLSTR resp. FLTL, exhibits an abrupt but durable improvement. The figures show that FLTL is able to pick up irrelevance of the trigger, while PLTLSTR is able to exploit irrelevance of the goal. As expected, PLTLMIN whose analysis is static does not pick up either and performs consistently badly. Note that in both figures, PLTLSTR progressively takes longer to compute as r increases because dynamic programming requires additional iterations to converge.

Conclusion and Future Work

NMRDPP proved a useful tool in the experimental analysis of approaches for decision processes with Non-Markovian rewards. Both the system and the analysis are the first of their kind. We were able to identify a number of general trends in the behaviours of the methods and to indicate which are best suited to certain circumstances.

We found PLTLSTR and FLTL preferable to the state-based PLTL approach in most cases. In all cases, attention should be paid to the syntax of the reward formulae and in particular to minimising its length. FLTL is the technique of choice when the reward requires tracking a long sequence of events or when the desired behaviour is composed of many elements with identical rewards. For guard formulae, we advise the use of PLTLSTR if the probability of reaching the goal is low or achieving the goal is very risky, and conversely, of FLTL if the probability of reaching the triggering condition is low or if reaching it is very risky.

This abstract has focused on current technologies. Future work should examine an adaption of these technologies to situations where reward allocation as well as dynamics is stochastic. Also of importance for the future, the importing of FLTL to the structured paradigm as well as the specification of heuristics for planning problems with temporally extended goals.

References

AT&T Research Labs. Graphviz. http://www.research.att.com/sw/tools/graphviz/.

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.

B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proc. ICAPS-03*, 2003.

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

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. In *Journal of Artificial Intelligence Research*, volume 11, pages 1–94, 1999.

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.

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.