COST-OPTIMAL STRONG PLANNING IN NON-DETERMINISTIC DOMAINS

Giuseppe Della Penna, Fabio Mercorio
Dept. of Computer Science, University of L'Aquila, L'Aquila, Italy

Benedetto Intrigila
Dept. of Mathematics, University of Rome Tor Vergata, Rome, Italy

Daniele Magazzeni
Dept. of Sciences, University of Chieti-Pescara, Chieti and Pescara, Italy

Enrico Tronci
Dept. of Computer Science, University of Rome La Sapienza, Rome, Italy

Keywords: Non-deterministic systems, Strong planning, Optimal planning.

Abstract: Many real world domains present a non-deterministic behaviour, mostly due to unpredictable environmental conditions. In this context, strong planning, i.e., finding a plan which is guaranteed to achieve the goal regardless of non-determinism, is a significant research challenge for both the planning and the control communities. In particular, the problem of cost-optimal strong planning has not been addressed so far. In this paper we provide a formal description of the cost-optimal strong planning problem in non-deterministic finite state systems, present an algorithm to solve it with good complexity bounds and formally prove the correctness and completeness of the proposed algorithm. Furthermore, we present experimental results showing the effectiveness of the proposed approach on a meaningful case study.

1 INTRODUCTION

In recent years, a mutual interest between control theory and AI planning communities has emerged, showing that planning and control are closely related areas. The use of sophisticated controllers as well as intelligent planning strategies has become very common in robotics, manufacturing processes, critical systems and, in general, in hardware/software embedded systems (see, e.g., (Chesi and Hung, 2007)).

In particular, efforts made to deal with planning in non-deterministic domains could be very helpful to solve control problems for real-world appliances. Indeed, many processes take place in an environment that may have variable and unpredictable influences on the action outcomes, which need to be taken into account to design a correct and efficient control system. In this context, strong planning, that is finding automatically a plan which is guaranteed to achieve the goal regardless of non-determinism, is a very interesting research challenge.

To this aim, planning based on Markov Decision Processes (MDP) has been proved very effective (see, e.g., (Boutilier et al., 1999; Bonet and Geffner, 2005; Yoon et al., 2002)), and, more recently, a variety of techniques have been proposed to solve continuous MDPs (see, e.g., (Meuleau et al., 2009; Mausam and Weld, 2008; Mausam et al., 2007)). However, MDP-based approaches deal with probabilistic distributions taking into account the stochastic outcomes of actions. Therefore, whether a solution provided by MDP planning algorithms is strong depends on probability and cost distribution.

On the other hand, planning under partial observability works on a setting where only a subset of variables are observable, and looks for conditional plans (see, e.g., (Bertoli et al., 2001; Bertoli et al., 2006; Huang et al., 2007)), whilst in conformant planning (Bonet and Geffner, 2000; Albore et al., 2010) no observation is available.

In this paper we focus on non-deterministic domains with full observability. In our setting, as typically in the case of dynamic systems, the size of the graph defining the dynamics of the system is exponen-

Della Penna G., Mercorio F., Intrigila B., Magazzeni D. and Tronci E..
COST-OPTIMAL STRONG PLANNING IN NON-DETERMINISTIC DOMAINS.
DOI: 10.5220/0003448200560066
In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2011), pages 56-66
ISBN: 978-989-8425-74-4

tial (*state explosion*) in the size of the input (i.e., the code which implicitly describes the graph by means of a *next state* function). As a result, classical algorithms for explicit graphs cannot be used, and instead suitable symbolic (e.g., (Burch et al., 1992)) or explicit (e.g., (Della Penna et al., 2004)) algorithms are used to counteract state explosion. This is also the typical situation for model checking problems.

Indeed, a key contribution to this field is described in (Cimatti et al., 1998), where the authors present an algorithm to find strong plans which has been implemented in MBP, a planner based on symbolic model checking. MBP produces a universal plan (Schoppers, 1987) which provides optimal solutions with respect to the plan length (i.e., the worst execution among the possible non-deterministic plan executions is of minimum length). Moreover, the use of Ordered Binary Decision Diagrams (OBDDs) together with symbolic model checking techniques allows a compact encoding of the state space and very efficient set theoretical operations.

Our work is strictly based on (Cimatti et al., 1998), however, there are the following main differences: (i) we consider a *cost function* and present a novel technique to look for *cost-optimal* strong plans while preserving a good complexity bound, and (ii) we use an *explicit approach* rather than a symbolic one, so extending the class of problems on which strong planning can be applied to hybrid and/or nonlinear continuous domains, which are actually very common in the practice. In fact, since representing addition and comparison with OBDDs requires opposite variable orderings and since this kind of problems involve both such operations, OBDDs in our context tend to have size exponential in the input size. On the other hand, using an explicit approach allows us to expand "on demand" the transition relation, generating and representing only the reachable states.

Note that the algorithm in (Cimatti et al., 1998) could be adapted to support costs and devise cost-optimal solutions only by "unary encoding" the weights, i.e., by replacing a transition of weight k with k contiguous deterministic transitions. In this case, however, its complexity, in the worst case, would be exponentially higher than the one of the algorithm presented in this paper.

Our synthesis problem could also be cast as a strategy synthesis problem for a multistage game with two players moving simultaneously (see, e.g., (Fudenberg and Tirole, 1991)), where the first player is the controller, the second is the disturbance (causing the non-deterministic behaviours), and the game rules are given by the plant dynamics. In this setting, our control strategy could be seen as a *minmax* strategy for the controller player. That is, in each game state, the controller chooses the action that minimises the maximum cost (to reach the goal) that the disturbance, with its (simultaneous) choice, may inflict to it. Such a game theoretic casting, however, would be of little help from a computational point of view, since in our setting the normal form of the game would be intractable even for small systems. Indeed, if the game has $|S|$ states and, in each state, the controller and the disturbance have at most $|\mathcal{A}|$ and $|D|$ actions available, respectively, then the game would be represented by a graph with $|S|$ nodes, each having $|\mathcal{A}||D|$ outgoing edges. Thus, even considering simple plants, we would have very large graphs: for example, in the case study presented in Section 4, we have that $|S| = 5 \cdot 10^7$, $|\mathcal{A}| = 25$ and $|D| = 17$, thus representing explicitly such a problem would yield a graph with $5 \cdot 10^7$ nodes and more than $2 \cdot 10^{10}$ edges. To the best of our knowledge, no tool based on game theory techniques can handle games of such size.

The situation is exactly analogous to that for model checking based analysis of Markov chains (e.g, see (Kwiatkowska et al., 2004), (Della Penna et al., 2006)). Of course, in principle, stationary distributions for Markov chains can be computed using classical numerical techniques (e.g., see (Behrends, 2000)) for Markov chains analysis. However, for dynamic systems, our setting here, the number of states (easily beyond 10^{10}) of the Markov chains to be analysed rules out matrix based methods.

Finally, casting our problem as a Mixed Integer Linear Programming (MILP) problem would be possible but, again, it would generate a MILP of size exponential in the input. Thus, to the best of our knowledge, this is the first approach to cost-optimal strong planning and no better solutions for this problem have been devised so far, even in other computer science fields.

In this paper we make the following contributions: we provide a formal description of cost-optimal strong planning in non-deterministic finite state systems (NDFSS), then we present an algorithm for cost-optimal strong planning on NDFSS, proving its correctness and completeness, and finally we show some meaningful experimental results.

The paper is structured as follows. Section 2 introduces the cost-optimal strong planning problem together with all the required background notions. Then, Section 3 describes an algorithm to solve this problem and proves its correctness. Section 4 presents a non-deterministic planning problem and shows the solution obtained through our algorithm. Finally, Section 5 outlines some concluding remarks.

2 STATEMENT OF THE PROBLEM

In this section we first introduce some background notions about non-deterministic systems and non-deterministic planning, then we define the concept of cost-optimal strong plan and the corresponding planning problem.

2.1 Non-deterministic Finite State Systems

We now introduce the formal definition of the non-deterministic systems we are interested in.

Definition 1. *A* Non-Deterministic Finite State System *(*NDFSS*) $\mathcal{S}$ is a 4-tuple (S,s_0,$\mathcal{A}$,F), where: S is a finite set of* states, $s_0 \in S$ *is the* initial state, $\mathcal{A}$ *is a finite set of* actions *and* $F : S \times \mathcal{A} \to 2^S$ *is the* non-deterministic transition function, *that is* $F(s,a)$ *returns the set of states that can be reached from state s via action a.*

It is worth noting that we are restricting our attention to NDFSS having a single initial state s_0 only for the sake of simplicity. Indeed, if we are given a NDFSS $\mathcal{S}'$ with a set of initial states $I \subseteq S$, we may simply turn it into an equivalent NDFSS by adding a dummy initial state connected to all the states in I by a deterministic transition with fixed cost. The algorithm given in Section 3 could then be trivially adapted to return the set of cost-optimal strong plans for I.

The non-deterministic transition function implicitly defines a set of transitions between states which, in turn, give raise to a set of trajectories as specified in the following definitions.

Definition 2. *Let $\mathcal{S} = \{S, s_0, \mathcal{A}, F\}$ be an* NDFSS. *A* non-deterministic transition τ *is a triple of the form $(s,a,F(s,a))$ where $s \in S$ and $a \in \mathcal{A}$. A* deterministic transition *(or simply a* transition*)* τ *is a triple of the form (s,a,s') where $s,s' \in S$, $a \in \mathcal{A}$ and $s' \in F(s,a)$. We say that $\tau = (s,a,s')$* is in *$(s,a,F(s,a))$ if $s' \in F(s,a)$. We denote with $\mathcal{S}_\tau$ the set of all the transitions in $\mathcal{S}$.*

Definition 3. *A* trajectory π *from a state s to a state s' is a sequence of transitions $\tau_0,\ldots,\tau_n$ such that τ_0 has the form (s,a,s_1), for some s_1 and some a, τ_n has the form (s_n,a',s'), for some s_n and some a' and for every $i = 0,\ldots,n-1$ if $\tau_i = (s_i,a_i,s_{i+1})$, for some s_i,a_i,s_{i+1}, then $\tau_{i+1} = (s_{i+1},a_{i+1},s_{i+2})$, for some s_{i+2},a_{i+1}. We denote with $|\pi|$ the length of π, given by the number of transitions in the trajectory.*

As usual we stipulate that *the empty set of transitions* is a trajectory from any state to itself.

We now need to specify several notions concerning non-deterministic transitions and trajectories.

Definition 4. *Let $\mathcal{S} = \{S, s_0, \mathcal{A}, F\}$ be an* NDFSS *and Π be a set of non-deterministic transitions. We say that a transition $\tau = (s,a,s')$ is* extracted from Π *if $(s,a,F(s,a)) \in \Pi$ and (s,a,s') is in $(s,a,F(s,a))$. Similarly, we say that a trajectory $\pi = \tau_0,\ldots,\tau_n$ is* extracted from Π *if $\forall i = 0\ldots n, \tau_i = (s_i,a_i,s_i')$ is in $(s_i,a_i,F(s_i,a_i))$ and $(s_i,a_i,F(s_i,a_i)) \in \Pi$. Finally, we say that a state s* is in Π *if there exists a transition $\tau = (s,a,s')$ extracted from Π.*

Since we are interested in cost-optimal solutions, we extend our setting with a cost function.

Definition 5. *Let $\mathcal{S} = \{S, s_0, \mathcal{A}, F\}$ be an* NDFSS. *A* cost function *(also called* weight function*) for $\mathcal{S}$ is a function $\mathcal{W} : \mathcal{S}_\tau \to \mathbf{R}_+$ that assigns a cost to each transition in $\mathcal{S}$. Using the cost function for transitions, we define the cost of the non-deterministic transition $(s,a,F(s,a))$, denoted by $\mathcal{W}(s,a)$, as follows: $\mathcal{W}(s,a) = \max_{s' \in F(s,a)} \mathcal{W}((s,a,s'))$.*

It is worth noting that, for the sake of generality, the definition of the cost function above allows the transition cost to depend on both the corresponding action and the source state. However, usually the transition costs are bound to the corresponding action only.

2.2 Cost-optimal Strong Plan

Now let $\mathcal{S}$ be a given *NDFSS*. In order to define the cost-optimal strong planning problem for such a kind of system, we assume that a non-empty set of *goal states* $G \subset S$ has been specified.

Definition 6. *Let $\mathcal{S} = \{S, s_0, \mathcal{A}, F\}$ be an* NDFSS. *Then a* Cost-Optimal Strong Planning Problem (COSPP) *is a triple $\mathcal{P} = (\mathcal{S}, \mathcal{W}, G)$ where G is the set of the goal states and $\mathcal{W} : \mathcal{S}_\tau \to \mathbf{R}_+$ is the cost function associated to $\mathcal{S}$.*

In this setting, we aim to find a *strong plan* from the initial state s_0 to G, that is a sequence of actions that, starting from s_0, leads the system to the goal states, regardless of the non-deterministic outcome of each action. Before formally describe such a solution, we need the following definitions.

Definition 7. *Let $\mathcal{P} = \{\{S, s_0, \mathcal{A}, F\}, \mathcal{W}, G\}$ be a* COSPP *and $s \in S$. A* deterministic plan *p from s to a goal $g \in G$ is a trajectory π such that:*

- *either $s \in G$ and $|\pi| = 0$;*
- *or $\pi = \tau_0,\tau_1,\ldots,\tau_n$, with $\tau_0 = (s,a,s_1)$ and $\pi' = \tau_1,\ldots,\tau_n$ is a* deterministic plan *from s_1 to g.*

Now we can formally define the cost-optimal strong solutions we are interested in finding.

Definition 8. *Let $\mathcal{P} = \{\{S, s_0, \mathcal{A}, F\}, \mathcal{W}, G\}$ be a* COSPP. *Let s be a state in S. A* strong plan *from s to G is a set P of non-deterministic transitions such that either $s \in G$ and $P = \emptyset$ or $s \notin G$ and P satisfies the following conditions:*

1. *there exists a natural number n_0 such that every trajectory π that can be extracted from P has length $|\pi| \leq n_0$;*
2. *every trajectory π starting from s which can be extracted from P, can be extended to a deterministic plan π' from s to a goal $s_g \in G$ such that π' is extracted from P;*
3. *for every state s' such that $s' \notin G$ and s' is in P there exists a trajectory π, extracted from P, starting from s and ending in s';*
4. *for every state s' such that $s' \notin G$ and s' is in P, there exists exactly one non-deterministic transition in P of the form $(s', a, F(s', a))$, for some $a \in \mathcal{A}$. We denote with $P(s')$ such non-deterministic transition.*

We have the following characterisation of plans.

Proposition 1. *Let $\mathcal{P} = \{\mathcal{S}, \mathcal{W}, G\}$ be a* COSPP. *P is a strong plan from s to G iff P is a set of non-deterministic transitions such that either $s \in G$ and $P = \emptyset$ or $s \notin G$ and there exists a unique non-deterministic transition in P of the form $\tau = (s, a, F(s,a))$, for some $a \in \mathcal{A}$, such that:*

- *either $F(s,a) \subseteq G$;*
- *or $P \setminus \{(s, a, F(s,a))\}$ is the union of strong plans P_i from every state s_i in $F(s,a)$ to G.*

Proof 1. *Assume first that P is a strong plan from s to G. If P is not empty, then there exists a unique non-deterministic transition $\tau = (s, a, F(s,a)) \in P$, for some a. Let s_i be an element of $F(s,a)$. We define P_i as the set of non-deterministic transitions in P such that P_i contains some transitions of a deterministic plan from s_i.*

Now observe that any sequence starting from s_i can be completed in P to a deterministic plan without using the node s. Indeed, no deterministic plan extracted from P can return to the node s, since otherwise there would be a cycle, contradicting the requirement that every deterministic sequence in P is bounded. It follows that P_i is a subset of $P \setminus \{\tau\}$ and is a strong plan from s_i.

Moreover, let s' be any node in $P \setminus \{\tau\}$. Then there exists a trajectory π from s to s'. By the uniqueness of τ, the first transition of π is in τ and therefore has the form (s, a, s_i) for some s_i, it follows that s' is in P_i, and that $P \setminus \{\tau\} = \bigcup_{s_i \in F(s,a)} P_i$.

The other direction is easy and left to the reader.

By Proposition 1 we can define the cost of a plan as follows:

Definition 9. *The cost of a strong plan P from s to G, denoted by $\mathcal{W}(P)$, is defined by recursion as follows:*

- *if P is empty then $\mathcal{W}(P) = 0$;*
- *if P is composed only of the non-deterministic transition $(s, a, F(s,a))$, for some a, then $\mathcal{W}(P) = \mathcal{W}(s,a)$;*
- *if P is composed of the non-deterministic transition $(s, a, F(s,a))$, for some a, and of plans P_i from every node s_i in $F(s,a)$ then $\mathcal{W}(P) = \max_{s_i \in F(s,a)} (\mathcal{W}((s, a, s_i)) + \mathcal{W}(P_i))$.*

It is easy to see that the cost of a plan P is the maximum cost of a deterministic plan extracted from P.

Definition 10. *Let $\mathcal{P} = \{\mathcal{S}, \mathcal{W}, G\}$ be a* COSPP, *with $\mathcal{S} = \{S, s_0, \mathcal{A}, F\}$. Then a* cost-optimal strong solution *of the* COSPP *$\mathcal{P} = \{\mathcal{S}, \mathcal{W}, G\}$ is a strong plan P from s_0 to G such that the cost of P is minimal among the strong plans from s_0 to G.*

2.3 An Example of Cost-optimal Strong Planning Problem

As an example of COSPP, let us consider the *hurried passenger* problem. A passenger wants to arrive to San Francisco airport (SFO) departing from one of the Rome airports (CIA or FCO) and according to the flight scheduling shown in Table 1. Moreover, there is a bus on every hour that allow the passenger to go from home to one of the Rome airports above in one hour.

Table 1: Flight scheduling.

From	To	Flight #	Depart	Arrive
Rome-FCO	Paris-CDG	A	08.00	09.00
Rome-FCO	Berlin-BER	E	08.00	10.00
Rome-CIA	Amsterdam-AMS	D	05.00	08.00
Paris-CDG	San Francisco-SFO	B	10.00	12.00 (GMT-7)
Paris-CDG	San Francisco-SFO	C	19.00	21.00 (GMT-7)
Berlin-BER	San Francisco-SFO	F	11.00	14.00 (GMT-7)
Berlin-BER	Amsterdam-AMS	I	12.00	13.00
Berlin-BER	San Francisco-SFO	G	12.00	15.00 (GMT-7)
Amsterdam-AMS	San Francisco-SFO	H	15.00	20.00 (GMT-7)

The goal is to arrive to San Francisco as soon as possible, and, however, no later than 21.00 local time. We require the passenger to arrive at the airport at least one hour before a flight departure. Moreover, we assume that each flight may arrive at destination later than the expected arrival time. The objective is

to generate a strong plan (if any) that guarantees the passenger to reach the San Francisco airport before 21.00 local time regardless of possible flight delays.

Table 2: COSPP for the *hurried passenger* problem.

S	home = s_0, AMS = s_1, AMS$_d$ = s_2, CDG = s_3, CDG$_d$ = s_4, CIA = s_5, FCO = s_6, BER = s_7, BER$_d$ = s_8. SFO$_a$ = s_9, SFO$_m$ = s_{10}, SFO$_n$ = s_{11}.
$\mathcal{A}$	A, B, C, D, E, F, G, H, I, P, Q
F	$F(s_0,Q)=\{s_6\}$, $F(s_0,P)=\{s_5\}$ $F(s_1,H)=\{s_9\}$, $F(s_2,H)=\{s_9\}$ $F(s_3,B)=\{s_9,s_{10}\}$, $F(s_4,C)=\{s_9,s_{11}\}$ $F(s_5,D)=\{s_1,s_2\}$ $F(s_6,A)=\{s_3,s_4\}$, $F(s_6,E)=\{s_7,s_8\}$ $F(s_7,F)=\{s_9\}$, $F(s_8,G)=\{s_9\}$, $F(s_8,I)=\{s_1,s_2\}$
$\mathcal{W}$	$\mathcal{W}(s_0,Q,s_6)=1, \mathcal{W}(s_0,P,s_5)=1$; $\mathcal{W}(s_1,H,s_9)=12, \mathcal{W}(s_1,H,s_9)=13$; $\mathcal{W}(s_2,H,s_9)=11, \mathcal{W}(s_2,H,s_9)=12$; $\mathcal{W}(s_3,B,s_{10})=10, \mathcal{W}(s_3,B,s_9)=11$; $\mathcal{W}(s_4,C,s_9)=18, \mathcal{W}(s_4,C,s_{11})=19$; $\mathcal{W}(s_5,D,s_1)=9, \mathcal{W}(s_5,D,s_2)=10$; $\mathcal{W}(s_6,A,s_3)=2, \mathcal{W}(s_6,A,s_4)=3$, $\mathcal{W}(s_6,E,s_7)=3, \mathcal{W}(s_6,E,s_8)=4$; $\mathcal{W}(s_7,F,s_9)=11, \mathcal{W}(s_7,F,s_9)=12$; $\mathcal{W}(s_8,G,s_9)=11, \mathcal{W}(s_8,G,s_9)=12$, $\mathcal{W}(s_8,I,s_2)=3, \mathcal{W}(s_8,I,s_1)=2$;
G	s_9, s_{10}
P	$P(s_0)=Q(17); P(s_5)=D(22); P(s_6)=E(16); P(s_7)=F(12)$; $P(s_8)=G(12); P(s_2)=H(12); P(s_1)=H(13), P(s_3)=B(11)$

The corresponding COSPP (according to Definitions 1,5 and 6) is reported in Table 2. Here the actions correspond to the flights and the nondeterminism is given by the possible delay which, for the sake of simplicity, we assume to be limited to one hour for each flight. Note that the delay probability distribution is uniform, i.e., each flight has equal probability to arrive on time or with a 1-hour delay. The cost of each transition (d,f,a) is $\mathcal{W}(d,f,a) = (t(d)+t(f)+t(a))$ where $t(d)$ is the time spent at airport d waiting for the flight departure, $t(f)$ is the duration of the flight and $t(a)$ is the time spent at airport a waiting for the next flight (which could be zero). Moreover, the special actions P and Q represent the bus journey from home (state s_0) to Rome-CIA and Rome-FCO, respectively: for the sake of simplicity, we do not consider delays on these actions, so the corresponding transitions are deterministic and have cost 1 (i.e., the bus journey takes an hour).

A graphical description of the problem is given in Figure 1 where tagged nodes represent the arrival time at the corresponding airport, while the edges are labelled with the flight code.

The cost-optimal solution consists in flying from Rome-FCO to Berlin-BER and then to San Francisco-SFO. The total cost of the solution (considering all the possible delays) is 17. Note that another strong solution would be flying from Rome-CIA to Amsterdam-AMS and then to San Francisco-SFO, but its cost is 23. Finally, flying from Rome-FCO to Paris-CDG is not a strong solution since in case of delay of flight A it would be impossible to reach San Francisco on time.

3 THE COST-OPTIMAL STRONG PLANNING ALGORITHM

In this section we describe a procedure that looks for a cost-optimal strong solution to a given COSPP. The main algorithm (Procedure 4) consists of two subroutines described in the following. All the procedures make use of some auxiliary functions and sets : *Cost*, *Cand*, *ExtGoals*, *OldExtGoals* and Δ .

The *cost function* $Cost(s)$ returns the minimum cost of a strong plan from s to the goals calculated so far. The algorithm updates this function every time a better strong plan is found for s. Initially all the goal states have a cost equal to zero, while the cost of the other states is set to ∞.

The set of *candidates* (*Cand*) contains the pairs (s,a) corresponding to all the states s which, at any step, are recognised to have a plan starting with action a, possibly of *non* minimum cost. The elements in the set *Cand* can be partially ordered with respect to the cost function *Cost*. Initially the set *Cand* is empty.

The set of *extended goals* (*ExtGoals*) contains all the states s which, at any step, are recognised to have a plan P of *minimum cost*. Initially the set *ExtGoals* contains all the goal states in G. On the other hand, the set of *old extended goals* (*OldExtGoals*) contains, at any step, the extended goals collected up to the previous step: that is, the expression *ExtGoals* $\setminus$ *OldExtGoals* represents the states that have been just added to the extended goals.

Finally, each set $\Delta(s,a)$ in initialised with the states reachable from s via action a, i.e., $F(s,a)$, which are consumed during the algorithm iterations.

In the following, we assume that all procedures take as input the COSPP $\mathcal{P} = ((S,s_0,\mathcal{A},F),\mathcal{W},G)$ as well as the auxiliary sets and functions. The output is a strong plan *SP*.

Note that, in the algorithms, some arithmetic operations (i.e., min, max and sum) may involve infinity. In this case, we assume the usual semantics, e.g., $\max(x,\infty)=\infty$ or $x+\infty=\infty$.

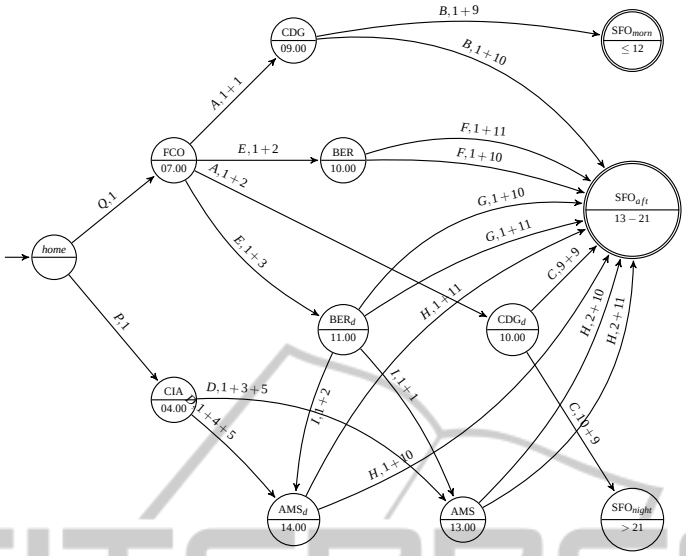

Figure 1: Graphical description of the COSPP for the *hurried passenger* problem.

3.1 The CandidateExtension Routine

The CANDIDATEEXTENSION routine (Procedure 1) extends the set *Cand* of candidates. The function $Pre(s)$ returns all the transitions leading to s and is applied to the extended goals found in the previous iteration of the main algorithm. At any step, the set $\Delta(s,a)$ contains only the states reachable from s via action a which have not been moved to the extended goals yet. Thus, once $\Delta(s,a)$ is empty, s is guaranteed to have a strong plan through action a, since all the transitions in $(s,a,F(s,a))$ lead to an extended goal. The pair (s,a) is then added to the set of candidates if it improves the cost currently associated to s.

Procedure 1: CANDIDATEEXTENSION.

1: **for all** $s' \in (ExtGoals \setminus OldExtGoals)$ **do**
2: $Pre(s') \leftarrow \{(s,a) \in S \times \mathcal{A} \,|\, s' \in F(s,a)\}$;
3: **for all** $(s,a) \in Pre(s')$ **do**
4: $\Delta(s,a) \leftarrow \Delta(s,a) \setminus \{s'\}$;
5: **if** $\Delta(s,a) = \emptyset$ **then**
6: $c' = \max_{\bar{s} \in F(s,a)}(\mathcal{W}(s,a,\bar{s}) + Cost(\bar{s}))$;
7: **if** $c' < Cost(s)$ **then**
8: $Cand \leftarrow Cand \cup (s,a)$;
9: $Cost(s) = c'$;
10: **end if**
11: **end if**
12: **end for**
13: **end for**

3.2 The PlanExtension Routine

The effect of the PLANEXTENSION routine (Procedure 2) is twofold. First, it selects the states in the candidates set of minimum cost and moves them to the set of extended goals. Indeed, the current solution for such states cannot be improved, since there are no actions which provide a strong solution with a lower cost (see Proposition 2). Second, it inserts the new extended goals together with the associated action (i.e., the corresponding non-deterministic transition) in the strong plan SP.

Procedure 2: PLANEXTENSION.

1: $\alpha \leftarrow \min_{(s,a) \in Cand} Cost(s)$;
2: **for all** $(s,a) \in Cand | Cost(s) = \alpha$ **do**
3: $ExtGoals \leftarrow ExtGoals \cup \{s\}$;
4: $Cand \leftarrow Cand \setminus \{(s,a)\}$;
5: $SP \leftarrow SP \cup (s,a,F(s,a))$;
6: **end for**

Note that the extraction of the candidates with the lowest cost (first two lines of Procedure 2) can be accomplished with a small complexity if we suppose to have a structure *costvector* where each element $costvector[c]$ holds a list of references to the states with cost c. Insertion in this structure is constant time, whereas updates can be also accomplished in constant time by re-inserting the state with updated cost without removing the previous instance (i.e., creating a duplicate with different cost). Indeed, the states with minimum cost can be extracted from this structure as shown by the MINCOSTCAND routine (Procedure 3).

The procedure takes as input the cost of the last

Procedure 3: MINCOSTCAND.

Input: *lastc*, the cost of the last states returned
1: $c \leftarrow lastc$
2: **loop**
3: $c \leftarrow c+1$
4: $AllCand_c \leftarrow costvector[c]$
5: **if** $AllCand_c \neq \emptyset$ **then**
6: $Cand_c \leftarrow \emptyset$
7: **for all** $s \in AllCand_c$ **do**
8: **if** $s \notin ExtGoals$ **then**
9: $Cand_c \leftarrow Cand_c \cup \{s\}$
10: **end if**
11: **end for**
12: **if** $Cand_c \neq \emptyset$ **then**
13: $lastc \leftarrow c$
14: **return** $Cand_c$
15: **end if**
16: **end if**
17: **end loop**

states returned, and scans the *costvector* starting from the element corresponding to the next (higher) cost c (for the sake of simplicity, in the pseudocode we suppose it to be $lastc+1$, but in general it depends on the approximation of the cost function). If $costvector[c]$ contains some states that are not yet in the extended goals, the procedure returns them, otherwise it increases c and loops. Thus, even if updates may create duplicates of the same state in different elements of *costvector*, since the algorithm always extracts *first* the minimum cost instance of a state, and inserts it in *ExtGoals*, all its further instances (with higher cost) in *costvector* will be simply ignored.

3.3 The CostOptimalStrongPlan Routine

Finally, the COSTOPTIMALSTRONGPLAN routine (Procedure 4) initialises the cost value of each state and the sets Δ, *Cand*, *ExtGoals* and *OldExtGoals*, then iterates applying the subroutines described above. In particular, the procedure loops until either the initial state s_0 is included in the extended goals (that is a strong solution has been found) or a fix point is reached, since there are no new extended goals (in this case there is no strong solution for s_0). Note that, as a collateral effect, the algorithm also finds all the strong plans for the states in S having minimal cost less or equal to the cost of s_0. Thus, if s_0 does not reach the goal (i.e., its cost is ∞), or if we explicitly remove the guard that stops the algorithm in this case, the COSTOPTIMALSTRONGPLAN would actually calculate a cost-optimal strong *universal* plan.

Procedure 4: COSTOPTIMALSTRONGPLAN.

Input: a COSPP $\mathcal{P} = ((S, s_0, \mathcal{A}, F), \mathcal{W}, G)$
Output: a cost-optimal strong plan *SP*
1: **for all** $(s, a, s') \in \mathcal{S}_\tau$ **do**
2: **if** $s \in G$ **then**
3: $Cost(s) = 0$;
4: **else**
5: $Cost(s) = \infty$;
6: $\Delta(s, a) = F(s, a)$;
7: **end if**
8: **end for**
9: $Cand \leftarrow \emptyset$;
10: $SP \leftarrow \emptyset$;
11: $OldExtGoals \leftarrow \emptyset$;
12: $ExtGoals \leftarrow G$;
13: **while** $(ExtGoals \neq OldExtGoals)$ **do**
14: **if** $s_0 \in ExtGoals$ **then**
15: **return** *SP*;
16: **end if**
17: CANDIDATEEXTENSION();
18: $OldExtGoals \leftarrow ExtGoals$;
19: PLANEXTENSION();
20: **end while**
21: **return** Fail;

3.4 Complexity of the Algorithm

Let m' be the number of transitions visited by the algorithm, which is less or equal than the total number of transitions in the graph, which mainly depends on the degree of non-determinism of the system. Moreover, let c' be the number of different costs that a state can be assigned to.

The CANDIDATEEXTENSION procedure removes at least one state from a set $\Delta(s, a)$ in each iteration. In other words, it consumes at least a transition of the transition graph for each call. The cost of each execution is constant, thus the complexity of the procedure is $O(m')$, since it is executed once for each visited transition.

On the other hand, the MINCOSTCAND algorithm, which is the core of the PLANEXTENSION procedure, scans the *costvector* exactly once, thus $O(c')$. If a particular element $costvector[c]$ does not have states assigned, the algorithm does not perform any other operation, otherwise it looks at each state with cost c to see if it has been already used. Since the cost of a state can be updated by CANDIDATEEXTENSION (and thus the state itself duplicated in *costvector*) at most once for each possible transition, the algorithm performs this check $O(m')$ times. It follows that the complexity of PLANEXTENSION is $O(m' + c')$

The overall complexity of COSTOPTIMALSTRONGPLAN is therefore $O(m' + c')$. It is worth noting that, if c' becomes too large, the *costvector* could not fit in RAM, and the algorithm complexity

measure could be unacceptable. In this case, we may employ a suitable sparse structure for *costvector*, e.g., a Fibonacci heap: indeed, with this kind of structure, it is easy to see that the complexity would become $O(m' \log m')$, which does not depend on c' any more.

Note that in CANDIDATEEXTENSION we suppose to use a precalculated *Pre* function. Otherwise, we may apply an explicit state space exploration algorithm to build it in $O(|\mathcal{S}_\tau|)$.

3.5 Correctness and Completeness of the Algorithm

The algorithm given in Procedure 4 essentially iterates the two procedures CANDIDATEEXTENSION and PLANEXTENSION until the desired state has a plan or the fix point is reached. Let us indicate with $ExtGoals_k$ and $Cand_k$ the contents of the *ExtGoals* and *Cand* sets, respectively, at the k-th step of the algorithm. Moreover, let us call $Goals_k$ the union $G \cup ExtGoals_k$.

Proposition 2. *Let u_k be the maximum cost of a state in $Goals_k$, that is $u_k = \max_{s \in Goals_k} Cost(s)$. Then, in any step $k \geq 1$ of the* COSTOPTIMALSTRONGPLAN *algorithm, all the states with a plan of minimum cost no greater than u_k are in $Goals_k$. That is $\forall s \in S$, $Cost(s) \leq u_k \Rightarrow s \in Goals_k$*

Proof 2. *At the first iteration of the algorithm ($k = 1$), $Goals_k = ExtGoals_k = G$ contains, by definition, all the states with a plan having cost zero (i.e., the goals).*

Now, let us assume by induction that the property holds at step k. We shall prove that it still holds at step $k+1$, i.e., the new elements inserted in $Goals_{k+1}$ do not falsify it.

To this aim, let α_{k+1} be the minimum cost of a candidate in $Cand_{k+1}$, that is $\alpha_{k+1} = \min_{s \in Cand_{k+1}} Cost(s)$. We can simply prove that $u_k < \alpha_{k+1}$. Indeed, assume that $u_k \geq \alpha_{k+1}$: then there exists a state $s \in Cand_{k+1}$ s.t. $Cost(s) \leq u_k$. However, a state in $Cand_{k+1}$ cannot be in $Goals_k$ (since the algorithm moves to the $ExtGoals$ only states that are already in Cand), and this contradicts the induction hypothesis.

Note that the fact above implies that, at the end of step $k+1$, i.e., after the execution of PLANEXTENSION*, we have that $u_{k+1} = \alpha_{k+1}$, since the algorithm moves in $Goals_{k+1}$ all the candidates with cost α_{k+1}, which is greater than the previous maximum cost u_k.*

Now assume that the property to be proved is falsified at step $k+1$. This implies that there exist one or more states s s.t. $Cost(s) \leq u_{k+1}$ but $s \notin Goals_{k+1}$. Let us choose among these states the one with minimum cost. Since we know that $u_{k+1} = \alpha_{k+1}$, we can also write that $Cost(s) \leq \alpha_{k+1}$.

By induction hypothesis, since a state which is not in $Goals_{k+1}$ could not also be in $Goals_k$, we have that $u_k < Cost(s)$. Let us consider a cost-optimal strong plan for s. Such plan must contain at least one state $s' \notin Goals_k$. Indeed, if all the states of such plan were in $Goals_k$, then s should be in $Goals_{k+1}$. Let us choose among these states the one with minimum cost.

We have two cases: if $s' = s$, then we have that, for some suitable action a, $F(s,a) \subseteq Goals_k$. This would imply that $s \in Cand_{k+1}$ and, since $Cost(s) \leq \alpha_{k+1}$, we would have that $Cost(s) = \alpha_{k+1}$ (recall that α_{k+1} is the minimum cost of a candidate in $Cand_{k+1}$). But in this case the algorithm would move s in $Goals_{k+1}$, contradicting the hypothesis.

Finally, if $s \neq s'$, then $Cost(s') < Cost(s)$ (by definition of cost of a plan). Again, since $Cost(s) \leq \alpha_{k+1}$, we have that $Cost(s') < \alpha_{k+1}$, so $s' \notin Cand_{k+1}$. Thus we also have that $s' \notin Goals_{k+1}$, and this contradicts the hypothesis since s would not be the state with minimum cost s.t. $Cost(s) \leq \alpha_{k+1}$ and $s \notin Goals_{k+1}$.

Thus, if a state enters in the extended goals (and is therefore included in the strong optimal plan), then its cost, i.e., the cost of the corresponding strong plan, cannot be improved. This shows the algorithm correctness.

The algorithm completeness can be easily derived from Proposition 2, too. To this aim, we can use the following proposition.

Proposition 3. *Let $s \in S$. If s has a cost-optimal strong plan P, whose cost is not greater than $Cost(s_0)$, then there exists $k > 0$ s.t. $s \in ExtGoals_k$ and $(s, a, F(s,a))$ is added to SP.*

Proof 3. *The proof follows from Proposition 2. Indeed, we have that the minimum cost of a candidate $\alpha_{k'}$ is strictly increasing in each step of the algorithm (otherwise, $u_{k'} < \alpha_{k'+1}$ would not hold). Thus, the process will eventually end with one of the following conditions:*

- *the initial state s_0 is in $ExtGoals_k$ (if a strong plan exists for such state): in this case, at step k all the states whose cost is not greater than $Cost(s_0)$, including s, are guaranteed to be in $ExtGoals_k$, too.*
- *there are no more candidate states that can be reached from the (extended) goals: in this case, since by hypothesis s has a strong plan, thus it can reach the goal, it would be included in the last set $ExtGoals_k$.*

Finally, the algorithm termination is guaranteed by the arguments used in Proposition 3. Indeed, since

the minimum cost of a candidate is strictly increasing, the algorithm will eventually build the cost-optimal strong plans for the states with highest cost: at this point, no new candidates will be available, and the process will terminate.

4 EXPERIMENTAL RESULTS

In this section we present a case study for the cost-optimal strong planning algorithm, the *inverted pendulum on a cart*, depicted in Figure 2.

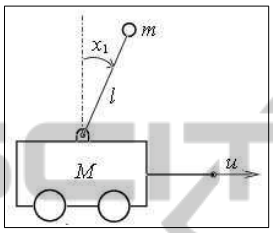

Figure 2: Inverted pendulum on a cart.

Here the goal is to bring the pendulum to equilibrium using a minimum amount of force applied to the cart. The non-determinism is given by possible disturbances on the actuator that may result in a variation of the force actually applied. Note that this apparently simple case study is instead an important issue in the controller design for many real-world systems. Indeed, many control problems, e.g., in engineering (i.e., the regulation of a steering antenna (Ilcev, 2009)) or robotics (Yokoi et al., 2003) can be reduced to an inverted pendulum problem.

The pendulum state is described by two real variables:

- x_1 is the pendulum angle (w.r.t. the vertical axis) with $x_1 \in [-1.5, 1.5]$ rad;
- x_2 is the angular velocity with $x_2 \in [-8, 8]$ rad/sec.

The continuous dynamics is described by a system of differential equations:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \frac{g_e \sin(x_1) - [\frac{\cos(x_1)}{m_p + m_c}][m_p l x_2^2 \sin(x_1) + u]}{4l/3 - [m_p/(m_p + m_c)] l \cos^2(x_1)} \end{cases}$$

where g_e is the gravitational constant, $m_p = 0.1Kg$ is the mass of the pole, $m_c = 0.9Kg$ is the mass of the cart, $l = 0.5m$ is the half-length of the pole and $u \in [-50, -46, \ldots, 46, 50]$N is the force applied to the cart. Note that, due to disturbances, x_2 can non-deterministically assume, with uniform probability distribution, a value that differs from the expected one by a small $\lambda \in [-\Lambda, \Lambda]$ with steps of 0.01 rad/sec.

The actions that can be applied in each state of the system correspond to the force applied to the cart, i.e., $\mathcal{A} = [-50, -46, \ldots, 46, 50]$. In this setting, the cost of a transition is given by the absolute value of the applied force, i.e., for any s, s', $\mathcal{W}(s, a, s') = |a|$. Therefore, a plan of minimum cost minimises the energy consumption.

In order to solve this continuous non-deterministic problem, we first had to approximate its continuous behaviour, i.e., to find a suitable *discretisation*. To this aim we exploited the discretise and validate approach implemented in the UPMurphi planner (Della Penna et al., 2009b), applied to the deterministic version of the problem, to devise such a discretisation. Then we applied the same discretisation to the non-deterministic version of the problem and ran the strong algorithm on the resulting NDFSS.

Table 3: Initial states for the inverted pendulum on a cart problem.

#	x_1	x_2	#	x_1	x_2	#	x_1	x_2
1	0.039	1.92	8	0.038	1.85	15	0.037	1.79
2	0.037	1.72	9	0.036	1.66	16	0.035	1.60
3	0.035	1.53	10	0.034	1.47	17	0.033	1.40
4	0.033	1.34	11	0.032	1.27	18	0.032	1.21
5	0.031	1.14	12	0.030	1.08	19	0.030	1.01
6	0.026	6.3	13	0.025	5.7	20	0.024	5.0
7	0.024	4.4	14	0.023	3.7	21	0.022	3.1

Table 4: Experimental results for the inverted pendulum on a cart problem.

Instance		1	2	3	4
Λ		0.02	0.04	0.07	0.08
G	$\lvert S\rvert$	$5 \cdot 10^7$			
	$\lvert S_\tau\rvert$	$6.25 \cdot 10^9$	$1.125 \cdot 10^{10}$	$1.875 \cdot 10^{10}$	$2.125 \cdot 10^{10}$
	Reach	11,439	11,889	12,509	12,649
	$Reach_\tau$	342,272	643,370	1,114,484	1,279,010
	$\lvert\mathcal{A}\rvert$	25	25	25	25
	$\max(\delta(s))$	77	140	244	284
	$\text{avg}(\delta(s))$	29.9	51.14	89.09	101.12
SP	size	1,146	1,198	698	—
	$\max(\mathcal{C}(s))$	$75N$	$304N$	$180N$	—
	time	10	18	13	7

In particular, we considered different instances of the problem, taking into account disturbances of increasing size, i.e., with $\Lambda \in \{0.02, 0.04, 0.07, 0.08\}$. For each instance, we applied the strong algorithm on the sample set of initial states shown in Table 3, which produced the results summarised in Table 4. Here, for each problem instance, we report some statistics about the corresponding graph G, i.e., total number of states ($|S|$), the total number of edges ($|S_\tau|$), the number of reachable states and edges (*Reach* and $Reach_\tau$, respectively), the number of actions ($|\mathcal{A}|$) and

the average and maximum out degree ($avg(\delta(s))$ and $\max(\delta(s))$, respectively) of the states. Then, we summarise the corresponding cost-optimal strong plan *SP*, as devised by the algorithm, giving its size (i.e., the number of plans that can be extracted from *SP*), the maximum cost ($\max(C(s))$ of a cost-optimal strong plan that can be extracted from *SP* and the total plan synthesis time (in minutes). The complete data set is available online at (Della Penna et al., 2010).

Note that, as discussed in Section 1, the size of the explicit graph representation of even a simple problem is huge: $|S| = 5 \cdot 10^7$ nodes and $|S_\tau| = 6.25 \cdot 10^9$ edges in our best case. By visiting only the reachable states, our algorithm succeeds in effectively counteracting state explosion.

We may note that, as expected, the greater the size of disturbances, the bigger the number of transitions, the smaller the number of states for which a strong plan is found. In particular, for the fourth instance, no strong plan exists. Moreover, the size of the resulting strong plan could be effectively compressed making use of Ordered Binary Decision Diagrams, as described in (Della Penna et al., 2009a).

5 CONCLUSIONS AND FUTURE WORK

In this paper we described an algorithm to solve the cost-optimal strong planning problem in non-deterministic finite state systems.

The presented approach extends the strong planning methodology given in (Cimatti et al., 1998) by introducing the concept of cost, and thus generating *cost-optimal* strong plans, and by exploiting explicit algorithms to extend the class of solvable problems.

The devised algorithm has been formally proved as correct and complete, and its complexity, if the number of transitions in the system or the range of possible transition costs are reasonable (to say, a billion of transitions or different costs), is dominated by the number of (visited) transitions in the system graph, which is a good bound for such kind of problem.

Finally, the proposed methodology has been illustrated through a case study based on the well known inverted pendulum on a cart problem. Future work will include an extensive experimentation on different case studies. However, the first results are very promising and show how the algorithm is effective and scalable.

REFERENCES

Albore, A., Palacios, H., and Geffner, H. (2010). Compiling uncertainty away in non-deterministic conformant planning. In *ECAI*, pages 465–470.

Behrends, E. (2000). *Introduction to Markov Chains*. Vieweg.

Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P. (2001). Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proc. 17th IJCAI*, pages 473–478. Morgan Kaufmann.

Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P. (2006). Strong planning under partial observability. *Artif. Intell.*, 170:337–384.

Bonet, B. and Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In Chien, S., Kambhampati, S., and Knoblock, C., editors, *Proc. 6th ICAPS*, pages 52–61. AAAI Press.

Bonet, B. and Geffner, H. (2005). mGPT: A probabilistic planner based on heuristic search. *JAIR*, 24:933–944.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94.

Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., and Hwang, L. J. (1992). Symbolic model checking: 10^{20} states and beyond. *Inf. Comput.*, 98(2):142–170.

Chesi, G. and Hung, Y. (2007). Global path-planning for constrained and optimal visual servoing. *IEEE Trans. on Robotics*, 23(5):1050–1060.

Cimatti, A., Roveri, M., and Traverso, P. (1998). Strong planning in non-deterministic domains via model checking. In *AIPS*, pages 36–43.

Della Penna, G., Intrigila, B., Lauri, N., and Magazzeni, D. (2009a). Fast and compact encoding of numerical controllers using obdds. In *Informatics in Control, Automation and Robotics: Selected Papers from ICINCO 2008*, pages 75–87. Springer.

Della Penna, G., Intrigila, B., Magazzeni, D., and Mercorio, F. (2009b). UPMurphi: a tool for universal planning on PDDL+ problems. In *Proc. ICAPS 2009*, pages 106–113. AAAI Press.

Della Penna, G., Intrigila, B., Magazzeni, D., and Mercorio, F. (2010). Non-deterministic inverted pendulum on a cart data set.

Della Penna, G., Intrigila, B., Melatti, I., Tronci, E., and Venturini Zilli, M. (2004). Exploiting transition locality in automatic verification of finite state concurrent systems. *STTT*, 6(4):320–341.

Della Penna, G., Intrigila, B., Melatti, I., Tronci, E., and Zilli, M. V. (2006). Finite horizon analysis of markov chains with the murphi verifier. *STTT*, 8(4-5):397–409.

Fudenberg, D. and Tirole, J. (1991). *Game theory*. MIT Press.

Huang, W., Wen, Z., Jiang, Y., and Wu, L. (2007). Observation reduction for strong plans. In *Proc. 20th IJCAI*, pages 1930–1935. Morgan Kaufmann.

Ilcev, S. D. (2009). Antenna systems for mobile satellite applications. In *Microwave Telecommunication Technology, 2009. CriMiCo 2009. 19th International Crimean Conference*, pages 393 –398.

Kwiatkowska, M. Z., Norman, G., and Parker, D. (2004). Probabilistic symbolic model checking with prism: a hybrid approach. *STTT*, 6(2):128–142.

Mausam, M., Bertoli, P., and Weld, D. S. (2007). A hybridized planner for stochastic domains. In *Proc. 20th IJCAI*, pages 1972–1978. Morgan Kaufmann.

Mausam, M. and Weld, D. S. (2008). Planning with durative actions in stochastic domains. *JAIR*, 31:33–82.

Meuleau, N., Benazera, E., Brafman, R. I., Hansen, E. A., and Mausam, M. (2009). A heuristic search approach to planning with continuous resources in stochastic domains. *J. Artif. Int. Res.*, 34:27–59.

Schoppers, M. (1987). Universal plans of reactive robots in unpredictable environments. In *Proc. IJCAI 1987*.

Yokoi, K., Kanehiro, F., Kaneko, K., Fujiwara, K., Kajita, S., and Hirukawa, H. (2003). Experimental study of biped locomotion of humanoid robot hrp-1s. In *Experimental Robotics VIII*, volume 5 of *Springer Tracts in Advanced Robotics*, pages 75–84. Springer.

Yoon, S. W., Fern, A., and Givan, R. (2002). Inductive policy selection for first-order MDPs. In *UAI*, pages 568–576.