Journal of Experimental & Theoretical Artificial Intelligence
Vol. 00, No. 00, Month-Month 200x, 1–16

Transformation of optimal planning problems

Martin C. Cooper, Marie de Roquemaurel, Pierre Régnier,

IRIT, University of Toulouse III,
118 route de Narbonne, 31062 Toulouse, France
(*Received 00 Month 200x; In final form 00 Month 200x*)

Cost-optimal planning, in which the aim is to minimize the sum of costs of actions, is a challenging problem due to its computational complexity. A linear program derived from a relaxation which ignores the constraints on the ordering of actions can be used to obtain a lower bound on the cost of a solution-plan. We show that the dual of this linear program provides a transformation of the problem into an equivalent optimal planning problem in which the cost of the goal-achieving action is exactly equal to this lower bound. This transformation is of universal utility since it can be applied as a preprocessing technique and can thus be combined with any of the diverse techniques which have been developed for optimal planning.

keywords: planning, optimization, linear programming.

1 Introduction

In many combinatorial problems, time spent in preprocessing can often greatly reduce search time. Intelligent search algorithms often employ similar techniques at every node of a search tree. In both cases, whether applied during preprocessing or during search, these techniques must be applicable in polynomial time to be useful. This is particularly true in planning, since the propositional planning problem is PSPACE-complete (Bylander, 1994).

The polynomial-time transformation of combinatorial problem instances into equivalent instances which are easier to solve is a classic approach in Artificial Intelligence. For example, the notion of arc consistency is ubiquitous in constraint satisfaction. In recent years, several different generalizations of arc consistency have been developed for weighted constraint satisfaction problems (Cooper and Schiex, 2004; Cooper *et al.*, 2010). These notions are known generically as soft arc consistency. We show in this paper that optimal planning problems can also be transformed in polynomial time into equivalent problems which are easier to solve. These techniques are largely inspired by corresponding soft arc consistency techniques.

The work which is presented here is most closely related to previous work on linear programming approaches to planning (Bylander, 1997; Benton *et al.*, 2007; van den Briel *et al.*, 2007, 2008). However, our transformation of an optimal planning problem into an equivalent problem with an explicit lower bound on the cost of a solution would appear to be completely novel.

The paper is organized as follows. Section 2 gives the necessary preliminary definitions. In Section 3 we give some lemmas concerning the number of occurrences of actions in a plan and the number of times fluents are established during execution of the plan. The resulting linear program can be seen as a relaxation of the original planning problem in which we ignore the order of actions in the plan. In Section 4 we show how an optimal planning problem can be transformed into an equivalent problem by shifting costs between those actions which affect the value of a particular fluent. The primary aim of such a local transformation is to increase the lower bound on the cost of an optimal plan. Section 5 shows that an optimal lower bound obtained by a set of simultaneous local transformations is the solution of a linear program which is the dual of the linear program established in Section 3. In Section 6 we show that this lower bound can be improved by merging coupled fluents.

*Corresponding author. Email: cooper@irit.fr

2 Definitions

Definition 2.1 A *state* S is a set of fluents (propositional literals), such that $\nexists f$ such that $(f \in S) \wedge (\neg f \in S)$. A fluent is *positive* (*negative*) if it is a non-negated (negated) propositional variable. We identify a state S with its sets of positive and negative fluents, denoted respectively S^+ and S^-. A set T of fluents is *satisfied* in a state S if $T \subseteq S$, i.e. $T^+ \subseteq S^+$ and $T^- \subseteq S^-$.

Definition 2.2 An *action* a is a triple $\langle prec(a), \mathit{eff}(a), cost(a)\rangle$, where the preconditions and effects, $prec(a)$ and $\mathit{eff}(a)$, are sets of fluents and $cost(a)$ is a real number representing the cost of applying the action a.

If T is a set of fluents, we use the notation $\overline{T} = \{\neg f : f \in T\}$ to represent the set of the negations of all the fluents in T. We use the notation $B - C = \{b \in B : b \notin C\}$ to denote set-difference.

Definition 2.3 An action a is *applicable* in a state S if $prec(a)$ is satisfied in S. The result of applying a in S is a state denoted by $S{\uparrow}a$ with $S{\uparrow}a \;=\; (S - \overline{\mathit{eff}(a)}) \cup \mathit{eff}(a)$.

Definition 2.4 A *planning problem* is a triple $\Pi = \langle A, I, G\rangle$ where I is the initial state I, A is the set of actions, and G is the set of goal fluents to be satisfied.

Definition 2.5 A *plan* is a sequence of actions, denoted $P = \langle a_1, \ldots, a_n\rangle$. Given an initial state S_0, define S_i $(i = 1, \ldots, n)$ to be the result of applying action a_i to S_{i-1} (assuming it is applicable). The plan P is *applicable* in a state S_0 if $\forall i \in \{1, \ldots, n\}$, a_i is applicable in S_{i-1}. If P is applicable in state S_0, then $S_0{\uparrow}P$ denotes the final state S_n. The plan P is a *solution-plan* for the problem $\Pi = \langle A, I, G\rangle$ if $a_1, \ldots, a_n \in A$, P is applicable in the initial state I and G is satisfied in the final state $I{\uparrow}P$.

The quality of a plan P is estimated by a function known as a plan metric. In this article we assume an additive metric which is the sum of the costs of the actions composing P. This is an essential assumption in order to apply the linear programming approaches described in Sections 3 and 5. It is generally satisfied by financial cost, for example.

Definition 2.6 Given a planning problem $\Pi = \langle A, I, G\rangle$, an *optimal plan* is a solution-plan $P = \langle a_1, \ldots, a_n\rangle$ which minimizes the metric

$$cost_\Pi(P) \;=\; \sum_{i=1}^{n} cost(a_i)$$

An *optimal planning problem* is a planning problem in which the aim is to find an optimal plan.

We do not always impose the restriction that costs be non-negative. In other words, an action may produce a profit. However, if profit-making actions are allowed, then it is possible to define optimal planning problems which have an infinite number of solution-plans of decreasing cost (or rather increasing profit) and hence have no optimal solution-plan.

In the following sections we introduce different techniques for the analysis or transformation of optimal planning problems. Beforehand we require some more notation and definitions.

Notation 2.7 *Let* eff_f *denote the set of actions* $\{a \in A : f \in \mathit{eff}(a)\}$.

Definition 2.8 A fluent f is *strict-effect* if for all actions a, $f \in \mathit{eff}(a) \;\Rightarrow\; \neg f \in prec(a)$.

Definition 2.9 An action a is *strict-delete* if for all positive fluents f, $\neg f \in \mathit{eff}(a)^- \;\Rightarrow\; f \in prec(a)$. An action a is *strict-effect* if for all (positive or negative) fluents f, $f \in \mathit{eff}(a) \;\Rightarrow\; \neg f \in prec(a)$.

Definition 2.10 We say that a planning problem is *strict-delete* or *strict-effect* if all its actions are respectively strict-delete or strict-effect.

3 Counting occurrences of actions and fluents

A classic approach in analysis of planning problems is to calculate a lower bound on the cost of a solution-plan by solving a relaxed version of the problem in which all delete effects (negative fluents in effects) of actions are ignored (McDermott, 1999; Bonet and Geffner, 2001; Hoffman and Nebel, 2001). A good lower bound is an essential ingredient of a complete search for an optimal plan. Although computing an optimal relaxed plan is known to be NP-hard (Bylander, 1994), this technique has nevertheless been successfully employed, for example, in the FF planner (Hoffman and Nebel, 2001). A disadvantage of this relaxation is that each positive fluent need only be established once and each action need only be executed once, since deletes are ignored. A better lower bound can be obtained by counting the number of occurrences of each action or fluent since this takes into account multiple occurrences (van den Briel *et al.*, 2007). This section briefly resumes the derivation of linear inequalities between these counts. Solving the linear relaxation of the resulting integer program provides a polynomial-time method for calculating a lower bound on the cost of a solution-plan. This encoding takes into account the delete effects of the actions but ignores action ordering.

Our aim is to develop a set of computational tools which can be used to analyze or transform optimal planning problems in the same way that recently developed techniques analyze or transform weighted constraint satisfaction problems (Cooper and Schiex, 2004; Cooper *et al.*, 2010). These techniques are themselves generalizations of classical consistency-enforcing techniques in CSPs (Dechter, 2003). As we show in Section 5, solutions to the dual of our IP formulation correspond to transformations of the optimal planning problem into an equivalent problem.

Consider a solution-plan P to a planning problem Π. In a state S, we consider that fluents in S^+ are true, fluents in S^- are false, and all other fluents are undefined. This is a slight generalisation of the closed-world assumption of STRIPS in which a state corresponds to an assignment of truth values to all fluents. We allow the possibility that the value of a fluent may be undefined in the initial state. Its value may later be set to either true or false by an action, but once a fluent has been assigned a value, it cannot later become undefined. In this section, unless explicitly stated, negative fluents may occur both as preconditions of actions and as goals.

Definition 3.1 A fluent f is *established* by an action a of P if f passes from false or undefined to true during the execution of a.

Let e_f be the number of times fluent f is established during the execution of P. Note that Definition 3.1 applies to negative as well as positive fluents. Thus $e_{\neg f}$ is the number of times that f passes from true or undefined to false. Let g_f (i_f) be the number of times the fluent f is true in the goal (initial state). g_f and i_f are either 0 or 1.

LEMMA 3.2 *Let f be a fluent. Then*

$$g_f \; + \; i_{\neg f} \; - \; 1 \; \leq \; e_f \; - \; e_{\neg f}$$

If $\neg f$ is strict-effect and $f \notin I$, then

$$g_f \; \leq \; e_f \; - \; e_{\neg f}$$

Proof For a fluent to be established by an action a, it must be false (or undefined) when a is executed. This implies that, during the execution of a plan, establishments of f and $\neg f$ alternate. Hence, for all fluents f, $-1 \leq (e_f - e_{\neg f})$. If $g_f = 1$ (i.e. f must be true at the end of the execution of the plan) or $i_{\neg f} = 1$ (i.e. f was false before the execution of the plan), then $0 \leq (e_f - e_{\neg f})$. Furthermore, if we have both $g_f = 1$ and $i_{\neg f} = 1$, then $1 \leq (e_f - e_{\neg f})$. It follows immediately that

$$g_f \; + \; i_{\neg f} \; - \; 1 \; \leq \; e_f \; - \; e_{\neg f}$$

When $\neg f$ is strict-effect, for all actions a, $\neg f \in \mathit{eff}(a) \Rightarrow f \in \mathit{prec}(a)$. If $f \notin I$, then we have the stronger

inequality $g_f \leq e_f - e_{\neg f}$, since, in this case, f must be established before $\neg f$. □

COROLLARY 3.3 *For all fluents f,*

$$g_f + i_{\neg f} - 1 \leq e_f - e_{\neg f} \leq 1 - g_{\neg f} - i_f$$

Proof Applying Lemma 3.2 to $\neg f$, we obtain $g_{\neg f} + i_f - 1 \leq e_{\neg f} - e_f$. Rearranging and combining with Lemma 3.2 gives $g_f + i_{\neg f} - 1 \leq e_f - e_{\neg f} \leq 1 - g_{\neg f} - i_f$. □

Let o_a be the number of times action a occurs in P.

LEMMA 3.4 *Let f be a fluent. Then*

$$e_f \leq \sum_{a \in \mathit{eff}_f} o_a$$

and furthermore, if f is strict-effect, then there is equality.

Proof The inequality follows directly from the fact that each establishment of the fluent f must be achieved by some action a for which $f \in \mathit{eff}(a)$. If f is strict-effect, then all actions in eff_f actually establish f, and hence we have equality. □

Example 3.5 Consider a planning problem with the following four actions:

$A1 = \langle \{a\}, \{b, \neg a\}, 1 \rangle$
$A2 = \langle \{c\}, \{a, \neg c\}, 1 \rangle$
$A3 = \langle \{d\}, \{b, \neg d\}, 1 \rangle$
$A4 = \langle \{b\}, \{c, d, \neg b\}, 1 \rangle$

and with initial state $I = \{c\}$ and goal $G = \{a, b\}$. For example, this could correspond to establishing a career plan for Mr X who is initially unemployed. A government-funded professional training organization only offers full-time places to people who are already in work and who wish to follow a training course in order to get a better job afterwards. Part-time places are awarded to students who have just completed one year of the course (thus giving them the possibility of returning to employment while studying). Mr X's goal is to be both in employment and studying. He wants to find the shortest plan: this corresponds to finding the cost-optimal plan when, as in this case, all actions have identical cost. The four fluents a, b, c, d have the following significations:

a = Mr X has a job
b = Mr X is a student on a training course
c = Mr X is unemployed
d = Mr X has just completed the first year of training

and the four possible actions correspond to:

$A1$: Mr X leaves his job and takes a full-time place on a course
$A2$: Mr X gets a job
$A3$: Mr X registers on a training course as a part-time student
$A4$: Mr X completes one year of the training course

From the above lemmas, using the fact that $\neg a$, $\neg b$ and $\neg d$ are strict-effect, we can deduce that

$1 \leq e_a - e_{\neg a}$,	$1 \leq e_b - e_{\neg b}$,	$0 \leq e_d - e_{\neg d}$	by Lemma 3.2
$e_{\neg a} = o_{A1}$,	$e_{\neg b} = o_{A4}$,	$e_{\neg d} = o_{A3}$	by Lemma 3.4
$e_a \leq o_{A2}$,	$e_b \leq o_{A1} + o_{A3}$,	$e_d \leq o_{A4}$	by Lemma 3.4

Hence

$$\begin{aligned} o_{A1} + o_{A2} + o_{A3} + o_{A4} &\geq e_a + e_b + e_d \\ &\geq 2 + e_{\neg a} + e_{\neg b} + e_{\neg d} \\ &= 2 + o_{A1} + o_{A4} + o_{A3} \\ &\geq 3 + e_{\neg b} + o_{A4} \\ &\geq 3 + 2o_{A4} \end{aligned}$$

Since the cost of each action is 1, we can deduce a lower bound of 3 on the cost of P.

By exhaustive search, we find that an optimal plan is $A2$, $A1$, $A4$, $A2$, $A3$, i.e. Mr X gets a job ($A2$), leaves this job to become a full-time student on a training course ($A1$), completes the first year of the course ($A4$), finds another job ($A2$) and registers as a part-time student ($A3$). The cost of this optimal plan is 5 since all actions have unit cost.

In general, we want to find a lower bound on the cost of a solution-plan P. Thus we want to solve the following problem:

$$\text{minimize} \quad \sum_{a \in A} o_a cost(a)$$

such that the inequalities derived from the above lemmas are satisfied and each of the variables o_a, e_f is a non-negative integer. If we solve the linear relaxation of the resulting integer program, we may obtain non-integer values for the variables o_a, but this will nonetheless still provide a useful lower bound on $cost(P)$. If the number of actions and fluents is not excessive we can even envisage an exhaustive search to solve the corresponding integer program.

4 Transformation of planning problems

Definition 4.1 A *cost-preserving transformation (CPT)* of an optimal planning problem Π is a mapping which transforms Π into an optimal planning problem Π' such that Π and Π' have the same set of solution-plans and such that for each solution-plan P,

$$cost_{\Pi}(P) \;=\; cost_{\Pi'}(P)$$

We say that Π and Π' are *cost-equivalent*.

It is important to note that a CPT does not, in general, preserve costs of actions or partial plans. Indeed, by Definition 4.1, all problems Π which have no solution-plans are deemed to be cost-equivalent.

The main utility of a CPT is to transform a problem Π into another Π' in which a lower bound on the cost of an optimal solution-plan is made explicit. Other applications include equalizing costs of actions so as to find a better upper bound on the number of actions in an optimal plan or, on the contrary, increasing the cost of a particular action in order to prove that it is too costly to be part of an optimal plan. A simple example will make this clearer. Imagine a transportation problem Π in which, to get from A to B, we have two possible solutions:

(i) take a taxi from A to the nearest airport (cost 20 euros); take a plane to the airport nearest to B (cost 140 euros); take a taxi from this airport to B (cost 20 euros).
(ii) take a bus from A to the nearest railway station (cost 2 euros); take a train to the station nearest to B (cost 152 euros); take a bus from this station to B (cost 2 euros).

This problem is clearly cost-equivalent to an identical problem Π' in which the cost of each of the taxi rides is 60 euros, the flight costs 60 euros, the cost of each of the bus rides is 52 euros and the train costs

52 euros. In this version, the minimum cost of actions is 52 euros (compared to 2 euros in the original problem). Once the solution-plan (bus-train-bus) of total cost 156 euros has been found, we can deduce that no better plan exists which uses more than two actions, since in Π' each action costs at least 52 euros. In another cost-equivalent version Π'' of the same problem, the taxis are free but the flight costs 180 euros, which immediately tells us that this action cannot be part of an optimal plan, since we already have a plan of total cost only 156 euros.

Π is also equivalent to a problem Π_{opt} which is identical to Π except that the cost of a taxi from A to the airport is 24 euros, all other actions have cost 0, but there is a cost of 156 euros incurred when we arrive at B (by whatever means of transport). The novelty of our approach lies in the fact that we do not just produce a lower bound, in this case 156 euros. We actually produce an equivalent problem in which this lower bound is explicit (as the cost of a unique goal-achieving action). The remaining costs in the transformed problem can help to guide search (for example by choosing 0-cost actions first or pruning actions whose cost when added to the lower bound exceeds the cost of the best plan found so far).

Before introducing a local CPT **Shift-costs** which can be applied to optimal planning problems, we first show that most planning problems of interest can be transformed into an equivalent strict-effect planning problem. This is essential since **Shift-costs** is only guaranteed to preserve costs when applied to strict-effect planning problems. The following two lemmas describe two distinct methods to render an optimal planning problem strict-effect.

Definition 4.2 The *arity of an action* a is $|\mathit{eff}(a)|$, the number of positive or negative fluents in $\mathit{eff}(a)$. The optimal planning problem $\Pi = \langle A, I, G\rangle$ is *k-local* if the arity of each action $a \in A$ is bounded above by k.

Notation 4.3 *F_A^+ denotes the set of positive fluents associated with the set of actions A, i.e. the set of non-negated literals f such that f or $\neg f$ occurs in the precondition or effect of at least one action in A. $F_A = F_A^+ \cup \overline{F_A^+}$ represents the positive and negative versions of all fluents associated with the set of actions A.*

Definition 4.4 A state S is *complete* with respect to the set of positive fluents F if for each fluent $f \in F$ either $f \in S$ or $\neg f \in S$. In the context of a planning problem $\Pi = \langle A, I, G\rangle$, we say that a state S is *complete* if it is complete with respect to the set of positive fluents F_A^+.

LEMMA 4.5 *There is a polynomial reduction from the class of k-local optimal planning problems with complete initial state, where k is a constant, to the class of strict-effect optimal planning problems.*

Proof Let $\Pi = \langle A, I, G\rangle$ be an optimal planning problem in which I is a complete state and such that, $\forall a \in A, |\mathit{eff}(a)| \leq k$. For a fluent $f \in F_A$ and an action $a \in A$ such that $\neg f \in \mathit{eff}(a)$ and $f \notin \mathit{prec}(a)$, we can replace a by two new actions a_1 and a_2 such that

$$\begin{array}{lll} \mathit{prec}(a_1) = \mathit{prec}(a) \cup \{f\} & \mathit{eff}(a_1) = \mathit{eff}(a) & \mathit{cost}(a_1) = \mathit{cost}(a) \\ \mathit{prec}(a_2) = \mathit{prec}(a) \cup \{\neg f\} & \mathit{eff}(a_2) = \mathit{eff}(a) - \{\neg f\} & \mathit{cost}(a_2) = \mathit{cost}(a) \end{array}$$

Action a_1 (a_2) is a version of a which can be applied in a state S if f is true (false) in S; a_1 deletes f whereas a_2 leaves the value of f unchanged at false. We will show that the resulting problem $\Pi_{(f,a)}$ is equivalent to Π. Let P be a solution-plan for Π containing the action a. Let S be the state that is attained during the execution of P just before the execution of action a. Since I is complete, so is S. We can therefore simply replace a by either a_1 or a_2, depending on whether $f \in S$ or $\neg f \in S$. Thus any solution-plan for Π can be converted in polynomial time into a solution-plan for $\Pi_{(f,a)}$. Conversely, if P' is a solution-plan for $\Pi_{(f,a)}$, then it is trivial to obtain a solution-plan P for Π, since it is sufficient to replace all actions a_1 or a_2 in P' by a.

We can clearly apply the above transformation in turn for each (positive or negative) fluent f and each action a such that $\neg f \in \mathit{eff}(a)$ and $f \notin \mathit{prec}(a)$. The resulting optimal planning problem Π' is strict-effect. The resulting transformation from Π to Π' introduces 2^{k_a} copies of action a where k_a is the number of fluents f for which $\neg f \in \mathit{eff}(a)$ and $f \notin \mathit{prec}(a)$. Since $k_a \leq k$ and k is a constant k, the transformation

is a polynomial reduction. □

Definition 4.6 A planning problem is *positive* if all goal fluents and all preconditions of all actions are positive.

Note that in a positive planning problem, actions may have negative effects. It is well known that any planning problem can be transformed into an equivalent positive problem of comparable size by the introduction of a new fluent $notf$ for each positive fluent f (Ghallab *et al.*, 2004).

Lemma 4.5 only provides a polynomial reduction from the class of k-local planning problems to the class of strict-effect planning problems. In the case that the original problem was positive and strict-delete, we do not need to place a bound on the arity of actions, as shown by the following lemma.

LEMMA 4.7 *There is a polynomial reduction from the class of positive strict-delete optimal planning problems to the class of strict-effect optimal planning problems.*

Proof Let $\Pi = \langle A, I, G\rangle$ be a positive strict-delete optimal planning problem. By hypothesis, all actions $a \in A$ are strict-delete. For each action $a \in A$, define action a' to be identical to action a except that it has been rendered strict-effect by the addition of negative fluents to its preconditions, i.e.

$$\begin{aligned} prec(a') &= prec(a) \cup \{\neg f | f \in \mathit{eff}(a)^+\} \\ \mathit{eff}(a') &= \mathit{eff}(a) \\ cost(a') &= cost(a) \end{aligned}$$

We also define, for each positive fluent $f \in F_A^+$, a new dummy action d_f which simply deletes the fluent f, i.e.

$$\begin{aligned} prec(d_f) &= \{f\} \\ \mathit{eff}(d_f) &= \{\neg f\} \\ cost(d_f) &= 0 \end{aligned}$$

All the actions a' (for $a \in A$) and d_f (for $f \in F_A^+$) are strict-effect.

Let $\Pi' = \langle A', I, G\rangle$, where

$$A' = \{a' | a \in A\} \cup \{d_f | f \in F_A^+\}$$

Π' is strict-effect. It remains to prove that it is equivalent to Π.

Any solution-plan P for Π can be transformed into a solution-plan for Π' by

(i) replacing each action a in P by the action a', and
(ii) adding the dummy action d_f just before an action a' whenever $\neg f$ does not hold but is required by the precondition of a'.

Conversely, any solution-plan P' for Π' can be transformed into a solution-plan for Π by deleting from P' dummy actions d_f, and by replacing each action a' in P' by the corresponding action a. Since the preconditions of the actions of Π contain no negated fluents, the resulting plan is valid. In both directions, the solution-plans have the same cost since the dummy actions each have cost 0. Furthermore, the reduction is clearly polynomial. □

We studied all problems from the 26 non-temporal STRIPS domains among the benchmark problems from the International Planning Competitions IPC-2[1] (Bacchus, 2001), IPC-3[2] (Long and Fox, 2003), IPC-

[1] http://www.cs.toronto.edu/aips2000/
[2] http://planning.cis.strath.ac.uk/competition/

Table 1. The percentage increase in the number of actions obtained by applying the transformation of Lemma 4.7 to positive strict-delete benchmark problems from different domains. The average is over the 20 (or more) problems from the same domain. For each domain, the "largest problem" column gives the percentage increase for the problem with most actions.

	percentage increase in number of actions	
domain	on average	largest problem
depots	22.14	7.04
driverlog	22.42	8.26
logistics	26.96	13.57
zenotravel	9.67	2.60
rovers	40.23	12.12
freecell	5.28	1.78
blocks	65.00	55.13
psr	68.53	6.29
elevator	17.99	5.00
openstack	20.07	5.90
parcPrinter	47.49	32.89
pegsol	57.44	54.05
scanalyzer	4.10	0.69
sokoban	62.71	39.15
transport	6.92	3.91
pathways	43.82	27.78
tpp	75.26	11.51
trucks	17.74	10.38

4[3] (Hoffmann *et al.*, 2006), IPC-5[4] (Dimopoulos *et al.*, 206) and IPC-6[5]. The 26 domains studied were: schedule, logistics, freecell, elevator, blocks, depots, driver, zeno, rovers, satellite, airport, pipesworld, promela optical telegraph, promela philosophers, psr, openstacks, pathways, storage, tpp, trucks, parcPrinter, pegsol, scanalyser, sokoban, transport and woodworking. For 19 of the 26 benchmark domains (namely schedule, logistics, freecell, elevator, blocks, depots, driver, zeno, rovers, psr, pathways, tpp, trucks, openstacks, parcPrinter, pegsol, scanalyser, sokoban and transport), all problem instances from the domain were found to be both positive and strict-delete, and hence Lemma 4.7 applies. We calculated the percentage increase in the number of actions when these benchmark problems are transformed as described in the proof of Lemma 4.7: the results are given in Table 1. We only counted actions and fluents which are accessible from the initial state (determined by means of relaxed planning graph analysis using the planner hsp_0^* (Haslum *et al.*, 2008)). The overall average increase in problem size (measured in terms of number of actions) is 34.10%. This percentage tends to fall as problem size increases and drops to 16.56% if we only consider the largest problem in each domain. We can conclude that most positive and strict-effect problems can be transformed into equivalent strict-effect problems with only a small increase in problem size.

In four of the remaining seven domains, the maximum value of k_a (for $a \in A$) was relatively small: 2 for satellite; 3 for storage; 6 for pipesworld; 7 for woodworking. Hence for these domains we can envisage applying the transformation described in the proof of Lemma 4.5. The maximum values of k_a were attained in the domains airport, promela optical telegraph and promela philosophers with values of k_a ranging from 12 (for airport) to 178 (for problem pfile7 of promela optical telegraph). We can conclude that the majority of the benchmark problems can be automatically transformed into equivalent strict-effect problems without a inordinate increase in the size of the set of actions.

Shift-costs (whose definition is given later) is only guaranteed to preserve costs when applied to fluents which have identical values before and after the execution of every solution-plan. As we show below, it is always possible to convert a planning problem into a normalized version so that this is true. Recall that we use F_A^+ to denote the set of positive fluents associated with the set of actions A, i.e. the set of non-negated literals f such that f or $\neg f$ occurs in the precondition or effect of at least one action in A.

Definition 4.8 An optimal planning problem $\Pi = \langle A, I, G \rangle$ is *normalized* if the initial state is $\{init\} \cup$

[3]http://ls5-web.cs.uni-dortmund.de/ edelkamp/ipc-4/
[4]http://zeus.ing.unibs.it/ipc-5/
[5]http://ipc.informatik.uni-freiburg.de/

$\{\neg f | f \in F_A^+ - \{init\}\}$ and the goal state is $\{goal\} \cup \{\neg f | f \in F_A^+ - \{goal\}\}$.

LEMMA 4.9 *There is a polynomial reduction from the class of strict-effect optimal planning problems with complete initial state to the class of normalized strict-effect optimal planning problems.*

Proof Let $\Pi = \langle A, I, G\rangle$ be a strict-effect optimal planning problem, with I a complete state. We introduce two new fluents *init* and *goal*, which, without loss of generality, we can assume were not present in F_A^+, and we let $F = F_A^+ \cup \{init, goal\}$. Define the initializing-action a_I and the goal-action a_G as follows

$$\begin{aligned} prec(a_I) &= \{init\} \cup \{\neg f | f \in F - \{init\}\} \\ \mathit{eff}(a_I) &= I^+ \cup \{\neg init\} \\ cost(a_I) &= 0 \end{aligned}$$

$$\begin{aligned} prec(a_G) &= G \cup \{\neg f | f \in F - G^+\} \\ \mathit{eff}(a_G) &= \{\neg f | f \in G^+\} \cup \{goal\} \\ cost(a_G) &= 0 \end{aligned}$$

The result of executing the initializing-action a_I is the complete state $I \cup \{\neg init, \neg goal\}$. The precondition of the goal-action a_G is also a complete state in which all fluents not belonging to G take the value false. In order to be able to set the non-goal fluents in F_A^+ to false, for each fluent $f \in F_A^+ - G^+$, we define the dummy action d_f^G as follows

$$\begin{aligned} prec(d_f^G) &= G \cup \{f\} \\ \mathit{eff}(d_f^G) &= \{\neg f\} \\ cost(d_f^G) &= 0 \end{aligned}$$

The actions a_I, a_G and d_f^G (for $f \in F_A^+ - G^+$) are all strict-effect. The effect of executing the action d_f^G is simply to delete f; this action therefore plays the same role as the dummy action d_f in the proof of Lemma 4.7 (except that d_f^G can only be applied after the goal G has already been achieved).

Let $\Pi' = \langle A', I', G'\rangle$, where

$$\begin{aligned} A' &= A \cup \{a_I, a_G\} \cup \{d_f^G | f \in F_A^+ - G^+\} \\ I' &= \{init\} \cup \{\neg f | f \in F - \{init\}\} \\ G' &= \{goal\} \cup \{\neg f | f \in F - \{goal\}\} \end{aligned}$$

Π' is strict-effect and normalized. It remains to prove that it is equivalent to Π.

Any solution-plan P for Π can be transformed into a solution-plan for Π' by prefixing P by the initializing-action a_I, suffixing P by the goal-action a_G and inserting the actions d_f^G just before a_G for all positive fluents $f \in F_A^+ - G^+$ that are true at the end of the execution of P. Conversely, any solution-plan P' for Π' can be transformed into a solution-plan for Π by deleting from P' all the actions a_I, a_G and d_f^G (for $f \in F_A^+ - G^+$). In both directions, the solution-plans have the same cost since the initializing-action, the goal-action and the dummy actions each have cost 0. The reduction is clearly polynomial. □

In the rest of the paper, we consider normalized strict-effect optimal planning problems. We assume that one of the transformations described in Lemma 4.5 or Lemma 4.7 has been applied, followed by the transformation described in Lemma 4.9. Note that if the original problem was positive and strict-delete, then it is unnecessary to introduce the dummy actions d_f^G since the dummy actions d_f (described in the proof of Lemma 4.7) already fulfil the same role.

Shift-costs(f, δ):
for all actions $a \in \mathit{eff}_f$ do
$\mathit{cost}(a) := \mathit{cost}(a) - \delta$;
for all actions $a \in \mathit{eff}_{\neg f}$ do
$\mathit{cost}(a) := \mathit{cost}(a) + \delta$;

Figure 1. A cost-preserving transformation of normalized strict-effect optimal planning problems.

Consider the transformation **Shift-costs**(f, δ), given in Figure 1, which modifies the costs in an optimal planning problem Π. Without loss of generality, we suppose that f is a positive fluent and $\delta \in \mathbb{R}$.

LEMMA 4.10 *Let Π be a normalized optimal planning problem in which f is a positive fluent such that $f \notin \{\mathit{init}, \mathit{goal}\}$. If both f and $\neg f$ are strict-effect, then* **Shift-costs***(f, δ) is a cost-preserving transformation.*

Proof Let P be a solution-plan for Π. Let e_f ($e_{\neg f}$) be the number of times f (respectively $\neg f$) is established during the execution of P and o_a the number of times action a occurs in P. Since Π is normalized and f is a positive fluent such that $f \notin \{\mathit{init}, \mathit{goal}\}$, $\neg f$ occurs in both the initial state and the goal state. Applying Corollary 3.3, we obtain

$$0 \leq e_f - e_{\neg f} \leq 0$$

and hence $e_f = e_{\neg f}$. Since both f and $\neg f$ are strict-effect, by Lemma 3.4, we have

$$e_f = \sum_{a \in \mathit{eff}_f} o_a \quad \text{and} \quad e_{\neg f} = \sum_{a \in \mathit{eff}_{\neg f}} o_a$$

Thus the cost of P in the transformed problem Π' is

$$\mathit{cost}_{\Pi'}(P) = \mathit{cost}_{\Pi}(P) - e_f\delta + e_{\neg f}\delta = \mathit{cost}_{\Pi}(P)$$

□

5 Optimal transformation of planning problems

Applying **Shift-costs**(f, δ) to a normalized strict-effect optimal planning problem transforms it into another cost-equivalent problem. But **Shift-costs** is just a local operation. It can be much more productive to simultaneously apply a set of such shifts. A common aim in combinatorial minimization problems is to find a lower bound on the value of an optimal solution. Such a lower bound is essential in intelligent search algorithms involving pruning of the search tree. We concentrate, therefore, on maximizing such a lower bound. If a_G is the goal-action, then we would like to maximize $\mathit{cost}(a_G)$. Of course, this is a lower bound on the cost of an optimal solution-plan only if all other costs are non-negative.

Any set of shifts is clearly equivalent to a set of calls **Shift-costs**(f, δ_f) where f varies over all positive fluents $F_A^+ - \{\mathit{init}, \mathit{goal}\}$. To simplify notation, in this section, all sums or quantifications over fluents are assumed to be over all positive fluents $F_A^+ - \{\mathit{init}, \mathit{goal}\}$.

For finite real costs, calls to **Shift-costs** commute. Note that the original costs of actions may be negative (representing a profit gained by executing the action), provided that in the transformed version Π', the cost of each action a is non-negative:

$$\mathit{cost}(a) - \sum_{f \in \mathit{eff}(a)} \delta_f + \sum_{\neg f \in \mathit{eff}(a)} \delta_f \geq 0$$

Therefore finding a best lower bound can be expressed as the following linear program:

LP1: maximize $\sum_{f \in G} \delta_f$ such that
for all actions $a \in A - \{a_G\}$,
$\sum_{f \in eff(a)} \delta_f \; - \; \sum_{\neg f \in eff(a)} \delta_f \; \leq \; cost(a)$

We do not impose the constraint that $cost(a_G)$ be non-negative in Π', since this is exactly the value we are trying to maximize. Indeed, if profit-making costs are allowed, the maximum value of $\sum_{f \in G} \delta_f$ may be negative.

Definition 5.1 A normalized strict-effect optimal planning problem with goal-action a_G is *optimally transformed* if no set of **Shift-costs** operations can increase $cost(a_G)$ while keeping the costs of all other actions non-negative.

THEOREM 5.2 *If* Π *is a normalized strict-effect optimal planning problem, then an optimally-transformed problem* Π' *which is cost-equivalent to* Π *can be found in polynomial time.*

Proof This follows immediately from the above discussion and from the fact that linear programming can be solved in polynomial time (Karmarkar, 1984; Schrijver, 1998) □

Example 5.3 Consider a planning problem with the following ten actions:

$A1 = \langle \{a, \neg b\}, \{b, \neg a\}, 1 \rangle$
$A2 = \langle \{c, \neg a\}, \{a, \neg c\}, 1 \rangle$
$A3 = \langle \{d, \neg b\}, \{b, \neg d\}, 1 \rangle$
$A4 = \langle \{b, \neg c, \neg d\}, \{c, d, \neg b\}, 1 \rangle$
$a_I = \langle \{\neg a, \neg b, \neg c, \neg d, \neg goal, init\}, \{c, \neg init\}, 0 \rangle$
$a_G = \langle \{a, b, \neg c, \neg d, \neg init, \neg goal\}, \{\neg a, \neg b, goal\}, 0 \rangle$
$d_f = \langle \{f\}, \{\neg f\}, 0 \rangle$ for each $f \in \{a, b, c, d\}$

This problem Π is equivalent to the problem of Example 3.5. It has been transformed as described in the proofs of Lemmas 4.7 and 4.9. It is strict-effect and normalized.

Applying the above approach to find the transformation of Π which maximizes the sum of the costs shifted to the goal fluents, we obtain the following linear program:

$$\begin{array}{ll} \text{maximize} & \delta_a + \delta_b \\ \text{subject to} & \delta_b - \delta_a \leq 1 \\ & \delta_a - \delta_c \leq 1 \\ & \delta_b - \delta_d \leq 1 \\ & \delta_c + \delta_d - \delta_b \leq 1 \\ & \delta_c \leq 0 \\ & -\delta_f \leq 0 \text{ for each } f \in \{a, b, c, d\} \end{array}$$

This linear program has an optimal solution $\delta_a = 1$, $\delta_b = 2$, $\delta_c = 0$, $\delta_d = 1$. This means that Π is cost-equivalent to an identical problem in which the costs of actions are

$$\begin{array}{lll} cost(\text{A1}) = 0 & cost(a_I) = 0 & cost(d_a) = 1 \\ cost(\text{A2}) = 0 & cost(a_G) = 3 & cost(d_b) = 2 \\ cost(\text{A3}) = 0 & & cost(d_c) = 0 \\ cost(\text{A4}) = 2 & & cost(d_d) = 1 \end{array}$$

in which a lower bound of 3 for all solution-plans has been made explicit since the cost of the goal-achieving action a_G is 3.

Consider a normalized strict-effect optimal planning problem Π which has the goal-action a_G as given in the proof of Lemma 4.9. By the argument in the proof of Lemma 4.10, for each positive fluent $f \notin$

Table 2. Average lower bounds (expressed as a fraction of the cost of an optimal plan) obtained by using **LP2** on benchmark problems.

domain	number of problems	lower bound/optimal
logistics	7	0.73
freecell	5	1.00
driverlog	6	0.66
zenotravel	6	0.71
tpp	5	0.76
blocks	4	0.72

$\{init, goal\}$,

$$\sum_{a \in \mathit{eff}_f} o_a \;=\; \sum_{a \in \mathit{eff}_{\neg f}} o_a$$

which we can rewrite as

$$\sum_{a \in \mathit{eff}_f - \{a_G\}} o_a \;-\; \sum_{a \in \mathit{eff}_{\neg f} - \{a_G\}} o_a \;=\; g_f$$

where $g_f = 1$ if $f \in G$ (and 0 otherwise), since the positive fluent f belongs to G iff $\neg f \in \mathit{eff}(a_G)$, there are no positive fluents $f \notin \{init, goal\}$ in $\mathit{eff}(a_G)$ and in a solution-plan we clearly have $o_{a_G} = 1$. We can therefore obtain a lower bound on the cost of a solution-plan, following the technique of Section 3, by solving the following linear program:

LP2: minimize $\sum_{a \in A - \{a_G\}} o_a cost(a)$
subject to $\forall f \in F_A^+ - \{init, goal\}$,
$\sum_{a \in \mathit{eff}_f - \{a_G\}} o_a \;-\; \sum_{a \in \mathit{eff}_{\neg f} - \{a_G\}} o_a \;=\; g_f$
and $\forall a,\; o_a \geq 0$

This is exactly the dual of **LP1**. The following result follows directly from the Fundamental Duality Theorem (Trustrum, 1971).

THEOREM 5.4 *The lower bounds obtained by solving the linear programs* **LP1** *and* **LP2** *are identical provided that at least one of the problems is feasible.*

Van den Briel *et al.* (2007) performed experimental trials in which they used **LP2** to obtain a lower bound on the cost of optimal plans in benchmark problems from the six domains listed in Table 2. In these trials all actions were assigned a cost of 1. The 'no. of problems' column gives the number of problems for which they were able to calculate the cost of an optimal plan (using Satplanner (Rintanen *et al.*, 2005)). For each domain we calculated the average of the lower bounds obtained by **LP2** when expressed as a fraction of the cost of an optimal plan. These values are given in the 'lower bound/optimal' column of Table 2. In each case, the lower bound is at least 66% of the actual cost of an optimal plan. For each of these problems the transformation found by **LP1** produces an equivalent problem Π' with the same lower bound λ rendered explicit as the cost of the goal-achieving action. Let Π'' be identical to Π' except that the cost of the goal-achieving action is 0. Any admissible heuristic calculated for Π'', derived by any other method, can then be cumulated with λ.

In order to apply the transformation (given by LP1), we need to normalize planning problems and this might result in an increase in the size of the problem. An important point is that we do not actually need to solve the normalized (and hence size-increased) version of the problem; we can use the normalized version simply to obtain a better lower bound, but then actually solve the original problem (if the lower bound is insufficient to prune this branch). We have seen in Section 4 that on most IPC benchmark problems, this increase is not enough to prevent us from using polynomial-time techniques for calculating a lower bound.

Example 5.5 We consider the transformed problem in Example 5.3 (which is a normalized and trans-

formed version of the problem in Example 3.5). We can cumulate the explicit lower bound $cost(A_G) = 3$ with any lower bound derived from the transformed version of the problem.

We say that an action $a \in A$ is *indispensable* for a planning problem $\langle A, I, G\rangle$ if it occurs in every plan-solution (Cooper *et al.*, 2009). This is equivalent to saying that the problem $\langle A - \{a\}, I, G\rangle$ has no solution. Several polynomial-time techniques exist for detecting certain indispensable actions (Cooper *et al.*, 2009). For example, for a planning problem to have a solution, the goal fluents must occur in some level of the planning graph (Blum and Furst, 1997) without any mutual exclusions between them. Thus, if the planning graph of the problem $\langle A - \{a\}, I, G\rangle$ levels off before all the goal fluents occur in the same level with no mutual exclusions between them, then a is necessarily an indispensable action for the problem $\langle A, I, G\rangle$. This is exactly what occurs in our example problem (Example 5.3) when $a = A4$, showing that $A4$ is an indispensable action. Since the cost of $A4$ in the transformed problem is 2, this immediately provides us with a combined lower bound of 5, which is, in fact, the cost of an optimal plan (as observed in Example 3.5).

Problem transformations have uses other than cumulating lower bounds. We can also clearly solve a linear program similar to **LP1** to determine, for example, whether there is a set of **Shift-costs** operations which transform Π into an equivalent problem in which the costs of all actions are bounded below by some constant C. In particular, in the case that the original problem had some actions with negative cost (i.e. profit-making actions), we can ask whether there is an equivalent problem with strictly positive costs. Similarly, we can solve a linear program to determine whether there is a locally-transformed version of Π in which $cost(a) > M$, where M is the cost of the best solution-plan found so far. If this is the case, then we can deduce that action a cannot be part of any optimal solution.

6 Coupled fluents

As discovered by van den Briel *et al.* (2007), the lower bound returned by **LP2** can be considerably improved by merging certain fluents. It is possible to replace two fluents g, h by four fluents $f[x, y]$ ($x \in \{g, \neg g\}$, $y \in \{h, \neg h\}$), where $f[x, y] \Leftrightarrow x \wedge y$. Although merging g, h in this way increases the number of fluents, it can in certain cases allow the linear program (**LP1** or **LP2**) to output an improved lower bound.

Definition 6.1 (van den Briel *et al.*, 2007) Two fluents g_0, g_1 are *coupled* if, for $i = 0, 1$, every action which modifies g_i has g_{1-i} or $\neg g_{1-i}$ as a precondition.

If g, h are coupled in a strict-effect planning problem, then merging them produces another strict-effect problem for which the linear program **LP1** has the same number of constraints but a larger number of variables. Each action which changes the state of g or h is replaced by a single action. For example, the action

$$a \quad = \quad \langle \{\neg g, h\}, \{\neg h\}, 1\rangle$$

is replaced by the action

$$a' \quad = \quad \langle \{f[\neg g, h], \neg f[\neg g, \neg h]\}, \{\neg f[\neg g, h], f[\neg g, \neg h]\}, 1\rangle$$

The fluent $f[\neg g, \neg h]$ represents the result of the action a on the fluents g, h. Its negation is present in the precondition of a' to ensure that a' is strict-effect. The fluent $f[\neg g, h]$ corresponds to the precondition of a on fluents g, h. Since it becomes false after execution of a', its negation is present in the effect of a'.

In general, each action a is replaced by an action a' which is identical to a except that all occurrences of $g, \neg g, h, \neg h$ have been replaced in its precondition and effect by fluents of the form $f[x, y]$ according to the following rules: for all $x \in \{g, \neg g\}$, $y \in \{h, \neg h\}$,

(i) if $x, y \in prec(a)$ then $f[x, y] \in prec(a')$;

(ii) if $x, y \in prec(a)$ and $(\neg x \in \mathit{eff}(a) \ \vee \ \neg y \in \mathit{eff}(a))$ then $\neg f[x,y] \in \mathit{eff}(a')$;

(iii) if $(x, y \in \mathit{eff}(a))$ or $(y \in \mathit{eff}(a) \wedge x \in prec(a) \wedge \neg x \notin \mathit{eff}(a))$ or $(x \in \mathit{eff}(a) \wedge y \in prec(a) \wedge \neg y \notin \mathit{eff}(a))$ then $f[x,y] \in \mathit{eff}(a')$ and $\neg f[x,y] \in prec(a')$.

Technically speaking, an action which does not modify g or h but has just one of $g, \neg g, h, \neg h$ as a precondition should be replaced by two new actions. An action a which has, for example, $g \in prec(a)$ but $h, \neg h \notin prec(a)$ (and hence $\mathit{eff}(a) \cap \{g, \neg g, h, \neg h\} = \emptyset$ by Definition 6.1 and since a is strict-effect) must be split into two separate actions $a1, a2$ which are identical except that the precondition g is replaced by $f[g,h]$ in $a1$ and by $f[g, \neg h]$ in $a2$. However, the actions $a1$, $a2$ have the same cost and keep the same cost after any **Shift-costs** operation (since $\mathit{eff}(a1) = \mathit{eff}(a2)$). Therefore, the number of constraints in **LP1** does not actually increase, since the constraint

$$\sum_{f \in \mathit{eff}(a)} \delta_f \ - \ \sum_{\neg f \in \mathit{eff}(a)} \delta_f \ \leq \ \mathit{cost}(a)$$

is identical for $a = a1$ and $a = a2$.

Example 6.2 Consider a simple planning problem with the following two actions:

$A1 = \langle \{a, \neg b\}, \{\neg a, b\}, 0 \rangle$
$A2 = \langle \{a, \neg b\}, \{b\}, 1 \rangle$

and with initial state $I = \{a\}$ and goal $G = \{a, b\}$. It suffices to add the following two actions to produce an equivalent normalized strict-effect planning problem:

$a_I = \langle \{\mathit{init}, \neg a, \neg b, \neg \mathit{goal}\}, \ \{\neg \mathit{init}, a\}, \ 0 \rangle$
$a_G = \langle \{a, b, \neg \mathit{init}, \neg \mathit{goal}\}, \ \{\neg a, \neg b, \mathit{goal}\}, \ 0 \rangle$.

Although the minimal cost of a solution-plan is clearly 1, **LP1** provides a lower bound of 0 since it consists in maximizing $\delta_a + \delta_b$ subject to

$-\delta_a + \delta_b \leq 0$
$\delta_b \leq 1$
$\delta_a \leq 0$.

It is easy to verify that a, b are coupled. After replacing a, b by fluents $f[a,b]$, $f[a, \neg b]$, $f[\neg a, b]$, $f[\neg a, \neg b]$ in $A1, A2$, setting $I' = \{f[a, \neg b]\}$, $G' = \{f[a,b]\}$ and re-normalizing, we obtain the new actions (which replace $A1, A2, a_I, a_G$):

$A1' = \langle \{f[a, \neg b], \neg f[\neg a, b]\}, \ \{f[\neg a, b], \neg f[a, \neg b]\}, \ 0 \rangle$
$A2' = \langle \{f[a, \neg b], \neg f[a, b\}, \ \{f[a,b], \neg f[a, \neg b]\}, \ 1 \rangle$
$a_{I'} = \langle \{\mathit{init}, \neg f[a,b], \neg f[a, \neg b], \neg f[\neg a, b], \neg f[\neg a, \neg b], \neg \mathit{goal}\},$
$\{\neg \mathit{init}, f[a, \neg b]\}, \ 0 \rangle$
$a_{G'} = \langle \{f[a,b], \neg f[a, \neg b], \neg f[\neg a, b], \neg f[\neg a, \neg b], \neg \mathit{init}, \neg \mathit{goal}\},$
$\{\neg f[a,b], \mathit{goal}\}, \ 0 \rangle$.

With this new set of actions, **LP1** consists in maximizing $\delta_{f[a,b]}$ subject to

$\delta_{f[\neg a,b]} \ - \ \delta_{f[a,\neg b]} \ \leq \ 0$
$\delta_{f[a,b]} \ - \ \delta_{f[a,\neg b]} \ \leq \ 1$
$\delta_{f[a,\neg b]} \ \leq \ 0$

which provides a lower bound of 1.

Fluents g, h which are not coupled can still be merged, but this requires splitting each action (for which Definition 6.1 does not hold) into two new actions. This produces an equivalent problem in which g, h are

coupled. In this case, the number of constraints in **LP1** actually increases. For example, the action

$$a \quad = \quad \langle \{g\}, \{\neg g\}, 1 \rangle$$

(in which g, h are clearly not coupled) can be split into two actions

$$a1 = \langle \{g, h\}, \{\neg g\}, 1 \rangle$$
$$a2 = \langle \{g, \neg h\}, \{\neg g\}, 1 \rangle$$

Replacing the fluents g, h by fluents of the form $f[x, y]$ gives

$$a1' = \langle \{f[g, h], \neg f[\neg g, h]\}, \{f[\neg g, h], \neg f[g, h]\}, 1 \rangle$$
$$a2' = \langle \{f[g, \neg h], \neg f[\neg g, \neg h]\}, \{f[\neg g, \neg h], \neg f[g, \neg h]\}, 1 \rangle$$

Since $\mathit{eff}(a1') \neq \mathit{eff}(a2')$ the number of constraints in **LP1** actually increases.

7 Conclusion

The analysis of optimal planning problems requires new tools compared to classical planning. A linear program which ignores the order of actions can be used to determine in polynomial time a lower bound on the cost of any solution-plan (van den Briel *et al.*, 2007). We have shown that the dual of this linear program provides a transformation of the problem into an equivalent problem in which the cost of the goal-achieving action is exactly equal to this lower bound. This transformation is a universal technique in the sense that it can be combined with any other technique for solving optimal planning problems. This transformation can be improved by first detecting coupled fluents.

The transformation of a weighted CSP (constraint satisfaction problem) obtained by solving a linear program to determine the optimal simultaneous shifting of costs between unary and higher-order cost functions is known as optimal soft arc consistency (OSAC) (Cooper *et al.*, 2007). The transformation of an optimal planning problem (described in Section 5) by shifting costs as dictated by the solution to a linear program is clearly analogous to OSAC. Several weighted-CSP techniques have also been developed which provide fast approximations to OSAC: they avoid solving a linear program by searching for local inconsistencies in the CSP obtained by only retaining zero-cost values and tuples (de Givry *et al.*, 2005; Cooper *et al.*, 2008). Further research is required to determine whether similar approximations can be applied to an optimal planning problem when the problem consisting of only zero-cost actions is found to have no solution. If so, then this would provide (as is already the case for weighted CSPs) a whole range of problem transformation techniques allowing trade-offs between computation time and the quality of the resulting lower bound.

An interesting theoretical question is whether there is a tractable class of optimal planning problems for which the lower bound obtained from the linear program is always equal to the actual cost of an optimal solution-plan.

References

F. Bacchus, "AIPS 2000 Planning Competition: The Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems", *AI Magazine* 22 (3) (2001) pp. 47–56.

J. Benton, M.H.L. van den Briel and S. Kambhampati, "A hybrid linear programming and relaxed plan heuristic for partial satisfaction planning problems", *Proc. ICAPS'07*, pp. 34–41.

A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis", *Artificial Intelligence* 90(1-2) (1997), 281–300.

B. Bonet and H. Geffner, "Planning as heuristic search", *Artificial Intelligence* 129(1) (2001) pp. 5–33.

M.H.L. van den Briel, J. Benton, S. Kambhampati and T. Vossen, "An LP-Based Heuristic for Optimal Planning", *Proc. International Conference on Principles and Practice of Constraint Programming (CP)* (2007) pp. 651–665.

M.H.L. van den Briel, T. Vossen and S. Kambhampati, "Loosely Coupled Formulations for Automated Planning: An Integer Programming Perspective", *JAIR* 31 (2008) pp. 217–257.

T. Bylander, "The Computational Complexity of Propositional STRIPS Planning", *Artificial Intelligence* 69 (1-2) (1994) pp. 165–204.

T. Bylander, "A Linear Programming Heuristic for Optimal Planning", *Proc. AAAI'97*, pp. 694–699.

M.C. Cooper, S. de Givry and T. Schiex, "Optimal soft arc consistency", *Proc. IJCAI'07*, pp. 68–73.

M.C. Cooper, S. de Givry, M. Sanchez, T. Schiex and M. Zytnicki, "Virtual arc consistency for valued CSP", *Proc. AAAI'08*, pp. 253–258.

M.C. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki and T. Werner, "Soft arc consistency revisited", *Artificial Intelligence*, 174 (2010) pp. 449–478.

M.C. Cooper, M. de Roquemaurel and P. Régnier, "A weighted CSP approach to cost-optimal planning", *IRIT Internal Report* RR-2009-28-FR (2009) (accepted for publication in *AI Communications*).

M.C. Cooper and T. Schiex, "Arc consistency for soft constraints", *Artificial Intelligence* 154 (1–2) (2004) pp. 199–227.

de Givry S., Heras, F., Zytnicki, M. and Larrosa, J., "Existential arc consistency: getting closer to full arc consistency in weighted CSPs", *Proc. IJCAI-05*, Edinburg, Scotland (2005) pp. 84–89.

R. Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.

Y. Dimopoulos, A. Gerevini, P. Haslum and A. Saetti, "The Benchmark Domains of the Deterministic Part of IPC-5", *Abstract Booklet of the competing planners of the Fifth International Planning Competition - Satellite Event of ICAPS'06*, pp. 14–19.

M. Ghallab, D. Nau and P. Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004.

P. Haslum, "Additive and Reversed Relaxed Reachability Heuristics Revisited", *Proceedings of 6th International Planning Competition (IPC'2008)*.

J. Hoffmann, S. Edelkamp, S. Thiebaux, R. Englert, F. Liporace and S. Trüg, "Engineering Benchmarks for Planning: the Domains Used in the Deterministic Part of IPC-4", *JAIR* 26 (2006), pp. 453–541.

J. Hoffman and B. Nebel, "The FF Planning System: Fast Plan Generation Through Heuristic Search", *JAIR* 14 (2001) pp. 253–302.

N. Karmarkar, "A new polynomial time algorithm for linear programming", *Combinatorica* 4 (4) (1984) pp. 373–395.

D. Long and M. Fox, "The 3rd International Planning Competition: Results and Analysis", *JAIR* 20 (2003) pp. 1–59.

D. McDermott, "Using regression-match graphs to control search in planning", *Artificial Intelligence* 109 (1999) pp. 111–159.

J. Rintanen, K. Heljanko and I. Nielmelä, "Planning as satisfiability: parallel plans and algorithms for plan search", Albert-Ludwigs-Universitt Freiburg, Institut für Informatik, Technical Report 216 (2005).

A. Schrijver, *The Theory of Linear and Integer Programming*, John Wiley and Sons (1998).

K. Trustrum, *Linear Programming*, Routledge and Kegan Paul, London (1971).