Improved Integer Programming Models and Heuristic Search for AI Planning

Yannis Dimopoulos
Department of Computer Science
University of Cyprus
Nicosia, Cyprus
yannis@cs.ucy.ac.cy

Abstract

Motivated by the requirements of many real-life applications, recent research in AI planning has shown a growing interest in tackling problems that involve numeric constraints and complex optimization objectives. Applying Integer Programming (IP) to such domains seems to have a significant potential, since it can naturally accommodate their representational requirements. In this paper we explore the area of applying IP to AI planning in two different directions.

First, we improve the domain-independent IP formulation of Vossen et al., by an extended exploitation of mutual exclusion relations between the operators, and other information derivable by state of the art domain analysis tools. This information may reduce the number of variables of an IP model and tighten its constraints. Second, we link IP methods to recent work in heuristic search for planning, by introducing a variant of FF's enforced hill-climbing algorithm that uses IP models as its underlying representation. In addition to extending the delete lists heuristic to parallel planning and the more expressive language of IP, we also introduce a new heuristic based on the linear relaxation.

Introduction

Many recent successful approaches to AI planning, including planning graphs (Blum and Furst 1995), propositional satisfiability (Kautz and Selman 1999) and heuristic search (Bonet, Loerincs, and Geffner 1997), are essentially restricted to planning domains representable in propositional logic. It seems however that many practical applications are beyond this representational framework, as they require expressing features like resources, numeric constraints, costs associated with actions, and complex objectives. Integer Programming (IP), and its underlying language of linear inequalities, seem to meet many of these requirements, at least from the representation perspective.

The study of the relevance of IP techniques to AI planning has only recently started to receive some attention. The LPSAT engine (Wolfman and Weld 1999) integrates propositional satisfiability with an incremental Simplex algorithm; Lplan (Bylander 1997) uses the linear relaxation as a heuristic in a partial-order causal-link planner; ILP-PLAN (Kautz and Walser 1999) uses IP models for solving problems with resources, actions costs and complex objective functions; Bockmayr and Dimopoulos (Bockmayr and Dimopoulos 1999) show how IP models can be used to incorporate strong forms of domain knowledge and represent compactly numeric constraints; finally, Vossen et al. (Vossen et al. 1999) introduce a strong, domain-independent, method for translating STRIPS planning into IP models.

The IP models intend to enhance previous approaches to AI planning, like BLACKBOX or GRAPHPLAN , that essentially view planning as a constraint satisfaction problem. All these methods provide optimality guarantees for the solutions they generate, usually with respect to plan length. Recently, (McDermott 1996) and (Bonet, Loerincs, and Geffner 1997) introduced a new promising approach to AI planning that is based on heuristic search. Planners of this family, eg. (Bonet and Geffner 1999; Hoffmann and Nebel 2001; Refanidis and Vlahavas 1999), automatically extract heuristic functions from a planning problem specification and use it to guide the search for a solution in the state space. These planners do not provide any optimality guarantees, unless they employ admissible heuristics combined with optimal search algorithms, as eg. in (Haslum and Geffner 2000).

In this paper we extend previous work on using IP in AI planning in two different directions. First, we improve the IP formulation of (Vossen et al. 1999). Second, we link IP methods to recent work in heuristic search for planning.

In the first part of the paper we describe an improved formulation of STRIPS planning, that exploits more fully the mutual exclusion relations between both the operators and the fluents, than this is done in (Vossen et al. 1999). Mutex information can strongly influence the way a planning problem is translated into inequalities, as it can yield formulations with fewer variables and constraints. Moreover, richer forms of information that can be derived automatically by domain analysis tools (Fox and Long 1998; Gerevini and Schubert 2000), can further tighten the IP formulation of a planning domain.

In the second part of the paper we present a variant of FF's (Hoffmann and Nebel 2001) enforced hill-climbing algorithm that uses the IP models as its underlying representational language and is capable of generating parallel plans. Moreover, we introduce a new method for heuristic evalua-

tion, that is based on solving the linear relaxation of the IP models, and compare it with the IP formulation of the delete lists relaxation method of FF.
As one may expect, FF clearly outperforms the new heuristic methods in term of running speed. This can be attributed partly to the fact that FF finds totally-ordered plans, while the new algorithms generate parallel plans. Moreover, while FF and similar planners, utilize highly optimized special-purpose techniques to speed-up heuristic evaluation, we rely on general purpose algorithms like Simplex and branch and bound. Nevertheless, generality has the advantage of extended expressiveness, as our approach can handle any domain representable in the language of linear inequalities. Moreover, we reiterate that the new algorithms generate parallel plans, a feature that can increase their usability in domains that are inherently parallel. This should be contrasted with most heuristic search based planners that generate totally-ordered plans, with the exception of recent work by Haslum and Geffner (Haslum and Geffner 2000). Additionally, the IP models can easily accommodate declarative domain knowledge and exploit it in the heuristic evaluation. Finally, IP formulations, combined with the linear relaxation heuristic allow us to implement a variety of strategies that offer a trade-off between search time and solution quality.

The paper is organized as follows. Section 2 presents very briefly the IP formulation of (Vossen et al. 1999) and then introduces the improved models. In section 3 we discuss new heuristic search methods that are based on the IP models. In section 4 we present and discuss some experimental results with the new IP formulation and the heuristic search algorithms, and in section 5 we conclude.

IP Models for Planning Problems

Integer Programming is a more general representation language than propositional logic. In order to represent a general linear inequality exponentially many clauses (in the number of variables) may be needed. Integer programming combines propositional logic with arithmetic and therefore allows for more compact formulations. On the other hand, any propositional clause $x_1 \vee \ldots \vee x_k \vee \bar{x}_{k+1} \vee \ldots \vee \bar{x}_{k+l}$ can be easily represented as a linear inequality $x_1 + \ldots + x_k - x_{k+1} - \ldots - x_{k+l} \geq 1 - l$ in 0-1 variables. However, such a straightforward translation usually leads to poor performance. Problem solving with IP techniques often requires a different translation of a problem into linear inequalities. Deriving a strong model for the problem to be solved is a fundamental issue for successfully applying IP.

As it is shown in (Vossen et al. 1999) similar observations hold for AI planning. Instead of simply translating a SAT encoding into linear inequalities, (Vossen et al. 1999) presented a different, substantially stronger, domain-independent IP formulation of planning problems. A brief presentation of this approach follows (see (Vossen et al. 1999) for details).

Domain Independent Modeling

Any STRIPS planning problem can be represented by a set of variables divided into action and state change variables. For each action a in the domain, we introduce an action variable $y_{a,i}$ which assumes the value true if a is executed at period i, and false otherwise. For each fluent f we define four variables, namely $x_{f,i}^{add}, x_{f,i}^{pre-add}, x_{f,i}^{pre-del}, x_{f,i}^{maintain}$. Variable $x_{f,i}^{maintain}$ encodes 'no-op' actions, while the other variables are defined as follows (symbol / denotes set difference).

$$\sum_{a \in pre_f/del_f} y_{a,i} \geq x_{f,i}^{pre-add}$$

$$y_{a,i} \leq x_{f,i}^{pre-add} \quad \forall a \in pre_f/del_f$$

$$\sum_{a \in add_f/pre_f} y_{a,i} \geq x_{f,i}^{add}$$

$$y_{a,i} \leq x_{f,i}^{add} \quad \forall a \in add_f/pre_f$$

$$\sum_{a \in pre_f \cap del_f} y_{a,i} = x_{f,i}^{pre-del}$$

Informally, $x_{f,i}^{add} = 1$ iff an action is executed at time point i that has f as an add effect but not as a precondition. Similarly, $x_{f,i}^{pre-add} = 1$ iff an action is executed at time point i that has f as a precondition but does not delete it, and $x_{f,i}^{pre-del} = 1$ if this action has f both as a precondition and a delete effect.
The constraints below prohibit parallel execution of mutually exclusive actions.

$$x_{f,i}^{add} + x_{f,i}^{maintain} + x_{f,i}^{pre-del} \leq 1$$

$$x_{f,i}^{pre-add} + x_{f,i}^{maintain} + x_{f,i}^{pre-del} \leq 1$$

The explanatory frame axioms are encoded as

$$x_{f,i}^{pre-add} + x_{f,i}^{maintain} + x_{f,i}^{pre-del} \leq x_{f,i-1}^{add} + x_{f,i-1}^{maintain} + x_{f,i-1}^{pre-add}$$

while the initial state constraints are represented by setting $x_{f,0}^{add}$ to 1 if f is true in the initial state and 0 otherwise. Finally, if $i \in 1, .., t$, for each goal f we add the constraint $x_{f,t}^{add} + x_{f,t}^{maintain} + x_{f,t}^{pre-add} \geq 1$.

Exploiting Domain Structure

The domain independent models of planning problems described above can be substantially improved by exploiting properties of the specific planning domain at hand. In particular, by analyzing in greater detail than in (Vossen et al. 1999) the mutual exclusion relations between the operators and the fluents of a domain, we may be able to reduce the number of state-change variables and tighten the constraints of the IP model. We assume the availability of suitable domain analysis tools capable of identifying these relations, as those described eg. in (Fox and Long 1998;

Gerevini and Schubert 2000). We describe first, two improvements that aim at reducing the number of variables of the IP model.

- For each fluent f such that $\sum_{a \in add_f/pre_f} y_{a,i} \leq 1$, define $x^{add}_{f,i}$ as $x^{add}_{f,i} = \sum_{a \in add_f/pre_f} y_{a,i}$. This is a simple modification that allows us to substitute out variable $x^{add}_{f,i}$. Similar observations hold for the $x^{pre-add}_{f,i}$ variables.
- Let f be a fluent such that $y_{a_1,i} + y_{a_2,i} \leq 1$ holds for every pair of actions $a_1 \in pre_f/del_f$ and $a_2 \in add_f/pre_f$. Then merge $x^{pre-add}_{f,i}$ and $x^{maintain}_{f,i}$ substituting out variable $x^{pre-add}_{f,i}$. With this modification we can omit all constraints that refer to $x^{pre-add}_{f,i}$ but we need to include constraints of the form $y_{a,i} \leq x^{maintain}_{f,i}$ for all $a \in pre_f/del_f$ that reflect the new, extended, meaning of the $x^{maintain}_{f,i}$ variables.

The first improvement is straightforward, while the second deserves some further discussion. Since every $a_1 \in pre_f/del_f$ is mutually exclusive with every $a_2 \in add_f/pre_f$, the variables $x^{pre-add}_{f,i}$ and $x^{add}_{f,i}$, are also exclusive. Instead of adding an additional constraint, we omit variable $x^{pre-add}_{f,i}$ and "transfer" its role in the model to the corresponding variable $x^{maintain}_{f,i}$. Note that while exclusion constraints of the form $x^{pre-add}_{f,i} + x^{maintain}_{f,i} + x^{pre-del}_{f,i} \leq 1$ are dropped, the constraints $x^{add}_{f,i} + x^{maintain}_{f,i} + x^{pre-del}_{f,i} \leq 1$ remain in the formulation, marking $x^{add}_{f,i}$ and $x^{maintain}_{f,i}$ (and therefore the deleted $x^{pre-add}_{f,i}$) as mutually exclusive.

We now discuss some techniques that can further tighten the IP model of a planning domain. Our method is based on the derivation of *single-valuedness* and *XOR* constraints as described in (Gerevini and Schubert 2000). We restrict our discussion to binary fluents. A single valuedness constraint is a constraint of the form $(f(y, *z), C(y))$ stating that for every value of variable y that satisfies constraint $C(y)$ there can be only one value for (the "starred") variable $*z$. An *XOR* constraint is a constraint of the form (XOR $f_1(y, z), f_2(y, u), C(y)$) stating that for every state and for every value of y satisfying $C(y)$ either $f_1(y, z)$ or $f_2(y, u)$ must be true (for some values of z and u) but not both.

- Let $f(y, z)$ be a binary fluent for which the single-valuedness constraint $(f(y, *z), C(y))$ holds. Then, for each y that satisfies $C(y)$, replace the set of constraints $x^{add}_{f,i} + x^{maintain}_{f,i} + x^{pre-del}_{f,i} \leq 1$ that refer to each possible value of z, with a single constraint

$$\sum_z x^{add}_{f,i} + \sum_z x^{maintain}_{f,i} + \sum_z x^{pre-del}_{f,i} \leq 1$$

where $\sum_z$ denotes the sum over the domain of the second parameter of fluent f, namely z.

Going one step further we can exploit the *XOR* constraints and modify the mutual exclusion constraints as follows.

- Let $f_1(y, *z)$ and $f_2(y, *u)$ be two single-valued fluents on their second arguments, for which the constraint (XOR $f_1(y, z), f_2(y, u), C(y)$) holds. Then, replace all mutual exclusion constraints on f_1 and f_2 that refer to some specific object satisfying $C(y)$ with the constraint

$$\sum_z x^{add}_{f_1,i} + \sum_z x^{maintain}_{f_1,i} + \sum_z x^{pre-del}_{f_1,i} + \sum_u x^{add}_{f_2,i} + \sum_u x^{maintain}_{f_2,i} + \sum_u x^{pre-del}_{f_2,i} \leq 1$$

Moreover, if the model does not contain any of the variables $x^{pre-add}_{f_1,i}$ and $x^{pre-add}_{f_2,i}$ (meaning that all actions that have f_1 or f_2 as a precondition, also have it as a delete effect), we can replace $\leq$ in the above constraint with an equality.

The following simplification relates to the frame axioms.

- Let f be a fluent such that every action that adds it, has f as its sole add effect. Then, if f is true at time $i-1$ we can safely add the constraint

$$x^{add}_{f,i-1} + x^{maintain}_{f,i-1} + x^{pre-add}_{f,i-1} \leq x^{pre-add}_{f,i} + x^{maintain}_{f,i} + x^{pre-del}_{f,i}$$

which combined with the corresponding explanatory frame axiom, namely

$$x^{pre-add}_{f,i} + x^{maintain}_{f,i} + x^{pre-del}_{f,i} \leq x^{add}_{f,i-1} + x^{maintain}_{f,i-1} + x^{pre-add}_{f,i-1}$$

gives rise to an equality of the form

$$x^{pre-add}_{f,i} + x^{maintain}_{f,i} + x^{pre-del}_{f,i} = x^{add}_{f,i-1} + x^{maintain}_{f,i-1} + x^{pre-add}_{f,i-1}$$

To see that the above transformation is valid, note that if a fluent f is true at time $i-1$ (meaning that one of the $x^{add}_{f,i-1}$, $x^{maintain}_{f,i-1}$, $x^{pre-add}_{f,i-1}$ is true), and all actions that add f have no other add effects, then assigning false to $x^{add}_{f,i}$, does not have unwanted complications. In fact, if we adopt the above simplification, $x^{add}_{f,i}$ will necessarily be assigned false, if f is true at time $i-1$ and f does not have an associated variable $x^{pre-add}_{f,i}$ or this variable is assigned the value false.
We note that the above transformation of the frame axioms into equalities can be extended to more general cases, but we do not discuss this issue further.

Example: Consider the rocket domain with the usual $load$ and $unload$ operators for packages and fly for airplanes. We note that all actions that add in for packages are mutually exclusive, therefore the fluent variables $x^{add}_{in,i}$ can be substituted out and replaced by $\sum_{a \in add_{in}/pre_{in}} y_{a,i}$, where a is a $load$ action that adds the corresponding in proposition. Similar constraints hold for at for both planes and packages,

hence all corresponding $x_{f,i}^{add}$ can be omitted from the formulation. Now consider the variable $x_{at,i}^{pre-add}$ corresponding to the fluent at that refers to airplanes. Note that every action $a_i \in pre_{at}/del_{at}$ (ie. load and unload actions) is mutually exclusive with every action $a_j \in add_{at}/pre_{at}$ (ie. fly actions) and therefore we can merge $x_{at,i}^{pre-add}$ with $x_{at,i}^{maintain}$, by omitting all $x_{at,i}^{pre-add}$ and adding the constraints $y_{ld,i} \leq x_{at,i}^{maintain}$ and $y_{un,i} \leq x_{at,i}^{maintain}$ for the corresponding load (denoted as $y_{ld,i}$) and unload ($y_{un,i}$) actions.
Moreover the single-valuedness of $in(x, *y)$, where x refers to packages and y to planes, will tighten the mutual exclusion constraint $\sum_L y_{un,i} + x_{in,i}^{maintain} + \sum_L y_{ld,i} \leq 1$ into the stronger constraint

$$\sum_{Pl}\sum_{L} y_{un,i} + \sum_{Pl} x_{in,i}^{maintain} + \sum_{Pl}\sum_{L} y_{ld,i} \leq 1$$

where $\sum_{Pl}$ denotes sum over all planes and $\sum_L$, sum over all locations. A similar constraint can be derived for fluent at that refers to packages being at locations. Since fluents in and at are related with a ***XOR*** constraint (meaning that, at each time, a package must be in some plane, or at some location but not both) we can combine the two constraints and derive

$$\sum_{Pl}\sum_{L} y_{un,i} + \sum_{Pl} x_{in,i}^{maintain} + \sum_{L} x_{at,i}^{maintain} + \sum_{Pl}\sum_{L} y_{ld,i} = 1$$

Finally, since the only operator that adds at for a package and a location is $unload$, and at is the only add effect of this operator, the corresponding frame axioms can be converted into equalities. Similar observations hold for the other propositions of the domain.

In the next section, when we introduce the constraint relaxation heuristic, we will need IP models for domains with operators that do not contain delete effects. For this special case we use a straightforward translation of propositional satisfiability planning theories into linear inequalities.

Heuristic Search

The above modifications in the IP formulation of planning problems, can substantially improve performance. However, as it happens with other approaches that generate optimal plans, in many domains, IP models do not scale well. In this section, we attempt to address this issue in a flexible way, by bringing together IP modeling and heuristic search. Heuristic search methods derive a heuristic function from the problem specification and use it to guide the search in the state space. The heuristic function h for a problem P is derived by considering a relaxed problem P'.

We consider two different relaxations of a planning problem. The first, which we call the *constraint relaxation* (CR) approach, was introduced in (Bonet, Loerincs, and Geffner 1997) and modified in `FF` (Hoffmann and Nebel 2001). Here the relaxed problem is obtained from the original by ignoring the delete effects of the operators. In the second approach, which we call *linear relaxation* (LR) approach, the relaxed problem is obtained from the original problem by dropping the integrality constraint from the integer variables.

The heuristic function we use is the same as in `FF`. Let $O_1, O_2, ..., O_m$ be the sets of actions selected (action selection can be a fractional number greater than 0, if the LR approach is used) in the solution of the relaxed problem at time i. Variable m, called the *length invariant*, equals the number of time steps needed so that the relaxed problem becomes solvable (ie., the relaxation with $m-1$ time steps is infeasible). We define our heuristic function as $h(S) = \sum_{i=1,...,m} |O_i|$.

The search method we use in our approach is a variant of the enforced hill-climbing introduced in `FF`. It can be described briefly as follows.

Algorithm **MEHC**($step$,$nd\text{-}limit$,$cutoff$)
i:=0; solved:=false;
while not solved
  i:=i+1;
Solve the relaxed problem using i time steps;
if feasible **then** solved:=true;
endwhile
m:=i; /*Length invariant m used in the heuristic function*/
Set obj to the value of the objective function and current plan to empty;
S :=Initial State; $h(S) = obj$;
while $h(S) \neq 0$ **do**
  Call **EBFS**(m,$step$,$nd\text{-}limit$,$cutoff$,$h(S)$) in order to breadth-first search for a state S' with $h(S') < h(S)$;
  if (no such state is found) **then** report failure and stop;
  Add the selected actions to current plan, set $S := S'$ and $h(S) := h(S')$;
endwhile

Algorithm `MEHC` differs from `FF` in the way it performs the search for the successor state at each of its iterations. The new search method is implemented by procedure `EBFS`, which is presented below.

For a planning problem P, let P_t denote the set of constraints in the IP formulation of P over the time interval t. Moreover, let P_t^{lr} denote the set of constraints obtained if the integrality constraints of P_t are dropped. Finally, let P_t^{cr} denote the set of constraints, over the time interval t, of the IP model of the problem obtained from P by dropping the delete list of the operators. Then, procedure `EBFS` below implements the breadth-first search for a state with a better heuristic value, where P_t' stands for any of P_t^{lr} and P_t^{cr} depending on the method we employ.

procedure **EBFS**(m,$step$,$nd\text{-}limit$,$cutoff$,$h(S)$)
i=$step$;
while (i<$cutoff$+$step$)
set branch and bound node limit to $nd\text{-}limit$, and
solve min($\sum_{a \in A} \sum_{j \in [i+1,m+i]} y_{a,j}$) subject to
$P_{[0,i]} \cup P'_{[i+1,m+i]} \cup \{\sum_{a \in A} \sum_{j \in [i+1,m+i]} y_{a,j} < h(S)\}$;
if (feasible) **then** return solution **else** i:=i+1;
endwhile

Note that the set of constraints $P_{[0,i]}$ allows for parallel action execution, and therefore, the successor of a state S can be any state that can be reached from S by executing a *set* of parallel actions. Hence, the algorithm generates *parallel plans*.

Procedure `EBFS` uses branch and bound in order to perform the search for the successor state, and therefore its theoretical time complexity is determined mainly by the number of integer variables of the problem it solves. If the linear relaxation heuristic is used, each iteration of `EBFS` has time complexity which is, in the worst case, exponential in the number of variables of $P_{[0,i]}$. In the case of the constraint relaxation heuristic, this complexity is higher, as it is exponential in the number of variables of $P_{[0,i]} \cup P^{cr}_{[i+1,m+i]}$. However, our experimentation revealed that, in many domains, the set of constraints P^{lr}_t is much harder to satisfy than the corresponding set P^{cr}_t. Consequently, the constraint relaxation heuristic can outperform the linear relaxation one in terms of running speed.

Moreover, the use of branch and bound in `EBFS` has some interesting implications. For instance, in the case of the constraint relaxation approach, the algorithm can branch on any of the variables of $P_{[0,i]} \cup P^{cr}_{[i+1,m+i]}$, interleaving in this way the selection of the successor state with its evaluation. Furthermore, the objective function, which minimizes the number of actions, can provide substantial guidance in the search of a low cost successor state.

The new algorithm is parametric to the values of *step*, *nd-limit*, and *cutoff*. Parameter *nd-limit* defines the node limit of the branch and bound search algorithm. Parameter *step* defines the minimum distance (number of parallel steps) of the successor state from the current state, while *cutoff* defines the maximum such distance. Parameters *step* and *nd-limit* allow us to implement a variety of search strategies that trade-off solution quality for performance. Higher values for *nd-limit* may generate successor states with better heuristic values, while higher values for *step* usually lead to more informed choices in the selection of the successor state. Therefore, higher values for these parameters usually yield better plans, while lower values better run times.

Experimental Results

We run some initial experiments with the new IP formulation and the heuristic search method on a variety of planning domains. The models were generated by hand, using the algebraic modeling system `PLAM` (ProLog and Algebraic Modeling) (Barth and Bockmayr 1998) and following the steps described in section 2. In all the experiments `CPLEX` 6.6 was used. All variables were declared integer. The setting was the following. At each node of the branch and bound dual simplex with "steepest-edge pricing" was used. Probing was set to 1, leading to some simplifications of the models, as well as clique cuts derivation. The variable selection strategy was set to "pseudo-reduced costs", and the node selection strategy to best-bound search. All experiments were run on a Sun Ultra-250 with 512 MB RAM.

Table 1 compares the performance of the IP formulation of (Vossen et al. 1999) (column OIP) and the improved formulation discussed in section 2 (column IIP) on blocks world and rocket domain problems. The objective function in both domains was set to minimize the number of actions. In the rocket domain, some flight minimization experiments were also run, and are marked with the problem name suffix `fl-min` in Table 1. The entries under "First" refer to the run time and number of nodes explored until the first solution was found. For the blocks world problems the entries under "Optimal" refer to solving these problems to optimality (the time needed to prove optimality is included). The same entries for the, more difficult, rocket problems refer to finding a solution with cost that is provably not more than 10% higher than the cost of the optimal solution. The data of Table 1 were obtained using, in each domain and for each formulation, the cut generation strategy that seems to perform best. In the blocks world domain the default values were used for both formulations. In the rocket domain, different Clique and Gomory cut generation strategies were used for the different formulations and the different optimization objectives, but we do not discuss this issue further.

Both domains allow for parallel actions. The largest problem in the blocks world domain, `bw3`, involves 13 blocks, has plan length 9 and the optimal solution has 19 actions. The IP formulation for this domain is strong, leading to small integrality gaps and, consequently, good performance. In some moderately sized problems, the first solution was obtained by simply rounding the values of the variables obtained after solving the linear relaxation.

In the rocket domain, the largest problem, `rocket4`, involves 16 packages, 5 locations and 3 planes. The optimal parallel plan length is 7, and the optimal solution contains 41 actions. Here the IP formulation is weaker. The integrality gap is larger, and closes relatively slow, after many iterations. Nevertheless, in most cases, the new formulation is substantially stronger.

Table 2 shows some representative results from experiments with the IP based heuristic methods on the parallel rocket domain. The CR columns refer to the constraint relaxation heuristic, while the LR columns to the linear relaxation one. The `FF` column shows the number of actions in the plan generated by `FF`, which solves all problems in a few seconds.

In all problems the relaxation based algorithm was run with the step and cutoff parameters set to 1. For the smaller problems, `rocket4` to `rocket6`, the $nd\text{-}limit$ parameter of the linear relaxation based algorithm was set to infinity, ie. at each iteration the corresponding problem was solved

	First				Optimal			
	OIP		IIP		OIP		IIP	
Problem	time	nodes	time	nodes	time	nodes	time	nodes
bw1	93	5	25	0	93	5	28	11
bw2	310	0	108	0	310	0	108	0
bw3	7072	21	876	23	-	-	1277	50
rocket1	98	371	13	56	631	2554	92	287
rocket2	109	297	10	28	1521	4237	132	260
rocket3	-	-	408	459	-	-	1885	1251
rocket4	7478	2522	461	456	-	-	1886	993
rocket1-fl-min	1828	1734	19	14	3775	5417	59	422
rocket2-fl-min	591	384	80	320	3602	4754	82	360
rocket3-fl-min	-	-	236	260	-	-	1685	4791
rocket4-fl-min	3391	372	210	150	-	-	2083	4067

Table 1: Performance comparison of different IP formulations on action and flight (marked with the name suffix fl-min) minimization problems. For each problem we give run time and number of nodes in the branch and bound tree. Times in seconds. A dash denotes that no solution was found within about 2 hours (8000 sec) of CPU time.

				LR			CR			FF
Problem	pac	pl	loc	time	len	actions	time	len	actions	actions
rocket4	16	3	5	29	9	45	25	13	46	53
rocket5	16	3	5	62	9	47	36	14	45	41
rocket6	16	3	5	33	9	43	31	13	49	52
rocket7-1	21	3	6	445	11	57	27	11	56	67
rocket7-2	21	3	6	427	12					
rocket7-3	21	3	6	286	12					
rocket8	27	4	7	1598	11	75	214	13	90	80

Table 2: Performance comparison of the heuristic search algorithms. For each problem we give the number of packages (pac), planes (pl) and locations (loc), solution time in seconds (time), parallel plan length (len) and number of actions in the plan (actions).

to optimality. However, for larger problems, like rocket7 and rocket8, when the linear relaxation heuristic is used, the number of explored nodes has to be limited in order to gain acceptable efficiency. The 3 entries of Table 2 prefixed with rocket7, correspond to solutions of the same problem with different node bounds. Row rocket7-1 refers to solving the problem without limiting the number of explored nodes, while rows rocket7-2 and rocket7-3 correspond to a limit of 1000 and 500 nodes respectively. For problem rocket8 the node limit was set to 500. In the problems we considered, restricting the number of nodes affects only the first two iterations of the algorithm, as in the subsequent iterations the optimal solution is found after exploring a few nodes. More specifically, in most problems, the first two iterations make up more than 60% of the total solution time.

In most of the problems it seems that the linear relaxation heuristic offers a reasonably good trade-off between search time and solution quality. For instance, if we compare the solution time of the direct IP formulation of problem rocket4 in Table 1, with the time needed for solving the same problem with the linear relaxation heuristic algorithm in Table 2, we note a decrease from 461 secs to 29 secs. This speed-up comes with an increase of the plan length from 7 to 9.

It is interesting to compare the characteristics of the linear and constraint relaxation methods. The constraint relaxation heuristic almost always runs faster than the linear relaxation one. But as far as parallel plan length is concerned, constraint relaxation is quite unstable, and in many cases (eg. problems rocket5 and rocket8 in Table 2) generates plans that underutilize the available resources (planes) and, for this reason, are longer.

Conclusions and discussion

The results presented in (Vossen et al. 1999), suggest that careful modeling can make IP effective in solving classical STRIPS problems. This has important practical implications, since many problems can be represented as a set of STRIPS operators together with some additional complex constraints. Solving such problems effectively, requires reasonable performance on their STRIPS part.

In this paper we presented some improvements of the IP formulation of (Vossen et al. 1999) that exploit more fully the structure of the planning domains. The new translation method benefits from recent work in automated domain analysis, but also recent advances in Integer Programming. Indeed, advanced features of state-of-the-art IP solvers, such

as preprocessing, probing, and constraint derivation, most notably in the form of Gomory and Clique cuts, have positive effect on the performance of the models. Our current work focuses on further improving the IP formulation of planning problems, and combining it with strong forms of domain knowledge. More extensive experimentation, to be reported in a longer version of this paper, gives encouraging first results.

The enforced hill-climbing algorithm that we described in the second part of the paper, can be understood as an attempt towards combining IP models with heuristic search. This is done in a way different than in the `Lplan` system (Bylander 1997), which uses the linear relaxation of an IP formulation, which is different than ours, as a heuristic in a partial-order causal-link planner.

Our intention is to develop algorithms that improve efficiency at an acceptable cost in solution quality. In parallel domains, which is our main focus, the degree of parallelism of the generated plan is an integral part of solution quality. It seems that the linear relaxation heuristic performs better than the constraint relaxation in terms of solution quality, but it is slower. We currently work on improving its performance.

Obviously, the heuristic search algorithm we presented is neither complete nor optimal. Future research will focus on investigating the possibility of employing IP models in heuristic search algorithms that satisfy both properties.

References

Barth, P., and Bockmayr, A. 1998. Modelling discrete optimisation problems in constraint logic programming. *Annals of Operations Research* 81.

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*.

Bockmayr, A., and Dimopoulos, Y. 1999. Integer programs and valid inequalities for planning problems. In *Proceedings of ECP-99*, 239–251.

Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In *Proceedings of ECP-99*, 360–372.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI/IAAI-97*, 714–719.

Bylander, T. 1997. A linear programming heuristic for optimal planning. In *Proceedings of AAAI/IAAI-97*, 694–699.

Fox, M., and Long, D. 1998. The automatic inference of state invariants in tim. *J. Artif. Intell. Res. (JAIR)* 9:367–421.

Gerevini, A., and Schubert, L. 2000. Discovering state constraints in DISCOPLAN: Some new results. In *Proceedings of AAAI-00*.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proceedings of AIPS-00*, 140–149.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)* 14:253–302.

Kautz, H., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *Proceedings of IJCAI-99*.

Kautz, H., and Walser, J. 1999. State-space planning by integer optimization. In *Proceedings of AAAI-99*.

McDermott, D. V. 1996. A heuristic estimator for means-ends analysis in planning. In *Proceedings of AIPS-96*, 142–149.

Refanidis, I., and Vlahavas, I. P. 1999. GRT: A domain independent heuristic for strips worlds based on greedy regression tables. In *Proceedings of ECP-99*, 347–359.

Vossen, T.; Ball, M.; Lotem, A.; and Nau, D. 1999. On the use of integer programming models in ai planning. In *Proceedings of IJCAI-99*.

Wolfman, S., and Weld, D. 1999. The LPSAT engine and its application to resource planning. In *Proceedings of IJCAI-99*.