On the Use of Integer Programming Models in AI Planning

Thomas Vossen **Michael Ball**
Robert H. Smith School of Business
and Institute for Systems Research
University of Maryland
College Park, MD 20742 USA
{tvossen, mball}@rhsmith.umd.edu

Amnon Lotem **Dana Nau**
Department of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742 USA
{lotem, nau}@cs.umd.edu

Abstract

Recent research has shown the promise of using propositional reasoning and search to solve AI planning problems. In this paper, we further explore this area by applying Integer Programming to solve AI planning problems. The application of Integer Programming to AI planning has a potentially significant advantage, as it allows quite naturally for the incorporation of numerical constraints and objectives into the planning domain. Moreover, the application of Integer Programming to AI planning addresses one of the challenges in propositional reasoning posed by Kautz and Selman, who conjectured that the principal technique used to solve Integer Programs—the linear programming (LP) relaxation—is not useful when applied to propositional search.

We discuss various IP formulations for the class of planning problems based on STRIPS-style planning operators. Our main objective is to show that a carefully chosen IP formulation significantly improves the "strength" of the LP relaxation, and that the resultant LPs *are* useful in solving the IP and the associated planning problems. Our results clearly show the importance of choosing the "right" representation, and more generally the promise of using Integer Programming techniques in the AI planning domain.

1 Introduction

Although some of the application areas addressed in the fields of Artificial Intelligence (AI) and Operations Research (OR) are very similar (e.g., planning, scheduling), the techniques that are used to solve these problems are oftentimes substantially different. Therefore, it seems only natural that recent research in the interface between AI and OR has focused on comparing the relative merits of the techniques and tools that are used in these areas. In this paper, we further explore the interface between AI and OR by applying Integer Programming (IP), which has a rich history in OR, to a classical AI problem, AI planning.

The possibility of using OR techniques in AI planning has not received much attention so far. Bylander [1997] uses Linear Programming as a heuristic for nonlinear planning; Bockmayr and Dimopoulos [1998] describe domain-dependent IP models for specific problem domains; and Kautz and Walser [1999], who use IP formulations for planning problems with resources, action costs, and complex objective functions. However, theirs is the only work we know of besides ours.

One potential advantage of using Integer Programming for AI planning is that IP formulations quite naturally allow the incorporation of numeric constraints and objectives into planning domains(for example, see Kautz and Walser [1999]). The use of numerical constraints and objectives is not addressed adequately in most existing AI planning systems, but it is critical in real-world planning [Nau *et al.*, 1998].

One difficulty indeveloping Integer Programming formulations for AI planning is that the performance of the resulting IP will depend critically on *how* AI planning problems are formulated as Integer Programs. The purpose of this paper is therefore to develop good domain-independent IP formulations for AI planning. In particular, we discuss various IP formulations for the class of action-based planning problems using STRIPS-style operators. Our main objective is to show that a carefully chosen IP formulation significantly improves the "strength" of the LP relaxation, so that it can provide useful guidance in solving the problem.

Our IP formulations are principally derived from work by Kautz and Selman [1996], which showed that planning problems can be efficiently solved by general propositional satifiablity algorithms. As such, the use of Integer Programming also addresses one of the challenges posed in the paper "Ten Challenges in Propositional Reasoning and Search," by Selman *et al.* [1997]. Specifically, the challenge that we address concerns the development of IP models and methods for propositional reasoning. Selman *et al.* mention that the basic technique used to solve integer programs—that is, the Linear Programming (LP) relaxation of the problem—does not appear to be useful for satisfiability problems, since it usually

sets all variables (modulo unit propagation) to the value $\frac{1}{2}$ and therefore does not guide the selection of variables in solving the problem.

Our results are as follows:

- In our experiments, IP formulations derived directly from SAT encodings proposed by Kautz and Selman [1996] performed rather poorly—but an alternative IP formulation that we call the "state-change formulation" was competitive with BlackBox using the systematic Satz solver [Kautz and Selman, 1998], in terms of the number of nodes expanded in the search space. Since the branching rule that is used in systematic satisfiability algorithms is an important factor in reducing the size of the search tree, this indicates that the LP relaxation *does* guide the selection of variables in solving the problem.
- Like Graphplan-based planners such as Blackbox, state-change formulation is guaranteed to find plans that have optimal values for the number of time steps in the plan. However, this formulation also takes into account the number of actions required by the plans (i.e., the plan *length*); and the number of actions obtained using the state-change IP formulation was usually much less than the number of actions in the plans obtained by BlackBox.

The organization of this paper is as follows. In Section 2, we discuss the various IP formulations of the planning problem. Next, Section 3 provides experimental results for these formulations, and a comparison with the systematic satisfiability solver. We conclude in Section 4 with a brief discussion of issues that arise in using IP techniques, and of plans for future work.

2 Integer Programming Formulations

The most effective current approach for solving general integer programs involves the use of branch and bound employing a linear programming (LP) relaxation. Thus, the key to the effectiveness of using integer programming to solve planning problems will lie in the effectiveness of the LP relaxation in improving the underlying tree search. The LP relaxation is typically solved at every node in the search tree. Search can be terminated at a node 1) if LP relaxation value indicates that further search could only uncover solutions with objective function values inferior to the best known, 2) if the LP is infeasible, which in turn implies the integer program is infeasible and 3) if the LP yields an integer solution. Since for planning problems, the objective function is only of secondary consideration, 1) will have little value. On the other hand 2) and 3) can be quite useful in improving search performance for planning problems. In particular, if the initial LP solves integer then no search is necessary. Another role the LP relaxation plays is that it provides information useful in deciding which variables to branch on.

A key issue in the performance of integer programming algorithms is the "strength" of the formulation. In general, there can be many equivalent integer programming formulations for a given problem. One formulation is stronger than another if the feasible region of the LP relaxation more closely approximates the integer program (see [Wolsey, 1998] for more details on this concept). Stronger formulations are more likely to yield integer solutions and produce objective function values closer to the values of the integer program.

In the remainder of this section we discuss two IP formulations for STRIPS-style planning problems. In order to express these formulations, we first introduce the following sets:

- $\mathcal{F}$, the set of *fluents*, that is, the set of all instantiated predicates;
- $\mathcal{A}$, the set of *actions*, that is, the set of all instantiated operators;
- $\mathcal{I} \subseteq \mathcal{F}$. $\mathcal{I}$ represents the set of fluents that hold initially;
- $\mathcal{G} \subseteq \mathcal{F}$. $\mathcal{G}$ represents the set of fluents that have to to hold in the goal state.

We assume that the number of time steps in the plan, t, is given. Furthermore, we introduce the sets

- $\text{pre}_f \subseteq \mathcal{A}$ for all $f \in \mathcal{F}$. pre_f represents the set of actions which have fluent f as a precondition;
- $\text{add}_f \subseteq \mathcal{A}$ for all $f \in \mathcal{F}$. add_f represents the set of actions which have fluent f as an add effect;
- $\text{del}_f \subseteq \mathcal{A}$ for all $f \in \mathcal{F}$. del_f represents the set of actions that delete fluent f.

2.1 SATPLAN-based IP Formulations

Initially, our IP formulations were motivated by the well known SATPLAN encodings, as discussed in [Kautz and Selman, 1996]. In SATPLAN, the problem of determining whether a plan exists, given a fixed number of time steps, is expressed as a satisfiability problem.

It is well known that satisfiability problems can be expressed as integer linear programs. (see for instance [Blair *et al.*, 1986] or [Hooker, 1988]). Usually, this is done by converting the clauses in the CNF representation of the satisfiability problem to 0-1 linear inequalities. For instance, the clause

$$x_1 \vee \neg x_2 \vee x_3$$

is equivalent to the $0-1$ inequality

$$x_1 + (1 - x_2) + x_3 \geq 1; x_1, x_2, x_3 \in \{0, 1\}.$$

Our first formulation consisted of this conversion for the SATPLAN encodings that are based on GraphPlan [Blum and Furst, 1996], i.e., we allow for parallel actions and the propagation of fluents using the "no-op" operator. The resulting formulation is summarized as follows:

Variables For all $f \in \mathcal{F}$, $i \in 1, \ldots, t+1$, we have fluent variables, which are defined as

$$x_{f,i} = \begin{cases} 1 & \text{if fluent } f \text{ is true in period } i, \\ 0 & \text{otherwise.} \end{cases}$$

For all $a \in \mathcal{A}$, $i \in 1, \ldots, t$, we have action variables, which are defined as

$$y_{a,i} = \begin{cases} 1 & \text{if action } a \text{ is carried out in period } i, \\ 0 & \text{otherwise.} \end{cases}$$

We remark that the action variables include the "no-op" maintain operators from GraphPlan for each time step and fact, which simply has that fact both as a precondition and as an add effect. "no-op" actions are necessary to propagate the fluent values.

Constraints The constraints are separated into different classes, which can be outlined as follows:

- **Initial/Goal State Constraints** These constraints set the requirements on the initial and final period, i.e.

$$x_{f,1} = \begin{cases} 1 & \text{if} f \in \mathcal{I}, \\ 0 & \text{if} f \notin \mathcal{I}. \end{cases}$$

$$x_{f,t+1} = 1 \quad \text{if} \quad f \in \mathcal{G}.$$

- **Precondition Constraints** Actions should imply their preconditions, which is expressed as follows.

$$y_{a,i} \leq x_{f,i} \qquad \forall a \in \text{pre}_f, i \in 1, \ldots, t.$$

- **Backward Chaining Constraints** Backward chaining is expressed as

$$x_{f,i+1} \leq \sum_{a \in \text{add}_f} y_{a,i} \quad \forall i \in 1, \ldots, t, f \in \mathcal{F}.$$

- **Exclusiveness Constraints** Actions conflict if one deletes a precondition or add effect of the other. The exclusiveness of conflicting actions is expressed as

$$y_{a,i} + y_{a',i} \leq 1,$$

for all $i \in 1, \ldots, t$, and all a, a' for which there exist $f \in \mathcal{F}$ such that $a \in \text{pre}_f$ and $a' \in \text{pre}_f \cup \text{add}_f$.

Objective Function The objective function was set to minimize the number of actions in the plan. It should be noted that in theory we could have chosen any objective function, since the constraints guarantee a feasible solution. In practice however, the choice of an objective function can significantly impact performance.

In addition, we made the following two modifications in the formulation. First of all, we used the notion of clique inequalities to strengthen the formulation. The basic idea behind this is that the inequalities $x_1 + x_2 \leq 1$, $x_2 + x_3 \leq 1$ and $x_1 + x_3 \leq 1$ can be replaced by a single inequality $x_1 + x_2 + x_3 \leq 1$. This leads to a formulation which is not only more compact but also stronger, in the sense that the fractional solution $x_1 = x_2 = x_3 = \frac{1}{2}$ is feasible in the first set of inequalities, but not in the second. It should be noted that the ability to detect clique inequalities is available in most of today's commercial solvers.

Secondly, we did not restrict all variables to be 0-1 integers. Specifically, the integrality of the fluent variables $x_{f,i}$ was relaxed, that is, the constraints $x_{f,i} \in \{0, 1\}$ were replaced by $0 \leq x_{f,i} \leq 1$. This is possible because the integrality of these variables is implied by the integrality of the action variables. We remark that as a consequence, none of the fluent variables will be selected in the branch and bound tree.

2.2 An Alternative Formulation

We now describe an alternative formulation of the planning problem, which we shall refer to as the "state-change formulation". The differences with respect to the formulation described in the previous section are twofold. First of all, the original fluent variables are "compiled away" and suitably defined "state change" variables are introduced instead. As we will see, this results in a stronger representation of the exclusion constraints. Secondly, we more explicitly restrict the possible propagation of fluents through "no-op"-actions, so as to reduce the number of equivalent feasible solutions.

Before giving this formulation, we again first define the variables. The action variables are the same as before, i.e.,

$$y_{a,i} = \begin{cases} 1 & \text{if action } a \text{ is executed in period } i, \\ 0 & \text{otherwise.} \end{cases}$$

for all $a \in \mathcal{A}$, $i \in 1 \ldots t$. Now however, the "no-op" actions are *not* included, but represented separately by variables $x_{f,i}^{\text{maintain}}$, for all $f \in \mathcal{F}$, $i \in 1, \ldots, t$.

In order to express the possible state changes, we introduce auxiliary variables $x_{f,i}^{\text{pre-add}}$, $x_{f,i}^{\text{pre-del}}$ and $x_{f,i}^{\text{add}}$, which are defined logically as

$$x_{f,i}^{\text{pre-add}} \equiv \bigvee_{a \in \text{pre}_f / \text{del}_f} y_{a,i},$$

$$x_{f,i}^{\text{pre-del}} \equiv \bigvee_{a \in \text{pre}_f \cap \text{del} f} y_{a,i},$$

$$x_{f,i}^{\text{add}} \equiv \bigvee_{a \in \text{add}_f / \text{pre} f} y_{a,i}.$$

Informally, $x_{f,i}^{\text{pre-add}} = 1$ if and only if an action is executed in period i that has f as a precondition but does not delete it. We note that the execution of such an action at a given time step implicitly asserts that the value fluent f is propagated. Similarly, $x_{f,t}^{\text{pre-del}} = 1$ if and only if an action is executed in period i that has f both as a precondition and a delete effect. $x_{f,i}^{\text{add}} = 1$ if and only if an action is executed in period i that has f as an add effect but not as a precondition.

The logical interpretation of these variables is repre-

sented in the IP formulation by the following constraints:

$$\sum_{a \in \text{pre}_f/\text{del}_f} y_{a,i} \geq x_{f,i}^{\text{pre-add}}$$

$$y_{a,t} \leq x_{f,i}^{\text{pre-add}} \quad \forall \quad a \in \text{pre}_f/\text{del}_f$$

$$\sum_{a \in \text{add}_f/\text{pre}_f} y_{a,t} \geq x_{f,i}^{\text{add}}$$

$$y_{a,t} \leq x_{f,i}^{\text{add}} \quad \forall \quad a \in \text{add}_f/\text{pre}_f$$

$$\sum_{a \in \text{pre}_f \cap \text{del}_f} y_{a,t} = x_{f,i}^{\text{pre-del}}$$

for all $f \in \mathcal{F}$, $i \in 1, \ldots, t$. The equality in the definition of $x_{f,i}^{\text{pre-del}}$ follows from the fact that all actions that have f both as a precondition and as an add effect are mutually exclusive. As a consequence these variables can in fact be substituted out, although for reasons of clarity we shall not do so here. The remaining exclusiveness constraints can easily be expressed in terms of the auxiliary variables, by stating that $x_{f,i}^{\text{pre-del}}$ is mutually exclusive with $x_{f,i}^{\text{add}}$, $x_{f,i}^{\text{pre-add}}$, and $x_{f,i}^{\text{maintain}}$. However, in order to strengthen the formulation we furthermore assert that $x_{f,i}^{\text{maintain}}$ is mutually exclusive with $x_{f,i}^{\text{add}}$ and $x_{f,i}^{\text{pre-add}}$. Informally, this means that a fluent can only be propagated at a time step if no action that adds it is executed. The resulting constraints are as follows.

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

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

for all $f \in \mathcal{F}$, $i \in 1, \ldots, t$.

The backward chaining requirements can also be expressed in terms of the auxiliary variables. Since, all auxiliary variable that assert the precondition of a fact f at a certain time step (i.e., $x_{f,i}^{\text{pre-add}}$, $x_{f,i}^{\text{maintain}}$, and $x_{f,i}^{\text{pre-del}}$) are mutually exclusive, we have the following constraint

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

for all $f \in \mathcal{F}$, $i \in 1, \ldots, t$.

Finally, we can express the initial/goal state constraints as

$$x_{f,t}^{\text{add}} + x_{f,t}^{\text{pre-add}} + x_{f,t}^{\text{maintain}} \geq 1$$

for all $f \in \mathcal{G}$, and

$$x_{f,0}^{\text{add}} = \begin{cases} 1 & \text{if} \quad f \in \mathcal{I}, \\ 0 & \text{otherwise.} \end{cases}$$

The objective function is again set to minimize the number of actions. Also, the integrality requirement of the auxiliary variables variables was again relaxed, as it is implied by the integrality of the action variables.

3 Experimental Results

We tested the IP formulations on a variety of planning problems from the Blackbox software distribution, and compared the results with those obtained by Blackbox using the systematic Satz solver. The integer programs were solved using Cplex 6.0, a widely used LP/IP solver. In solving the integer programs, we used all of Cplex's default settings, except the following: the initial LP optimum was obtained by solving the dual problem, and the variable selection strategy used was "pseudo-reduced cost". In addition, the solver was terminated as soon as a feasible integer solution was found. All problems were run on a Sun Ultra workstation.

The results are shown in Table 1. "Nodes" represents the number of nodes visited in the branch and bound procedure, and "iterations" the number of simplex iterations performed. All times are in seconds. It should be noted that, both for the IP formulations and BlackBox, the results shown are for the problem of finding a feasible solution given the number of time steps (i.e., t is known in advance and given).

As shown in Table 1, the state-change formulation led to a significant improvement in performance. Whereas the SATPLAN-based IP formulation solved only the smallest problems, the state-change formulations solved all, and required both fewer nodes and less computation time. While the systematic BlackBox solver usually required less time than the state-change formulation, both BlackBox and the state-change formulation explored similar numbers of nodes. Moreover, the BlackBox/Satz did not find a feasible solution to the "bw-large.b" blocks-world problem, while the state-change formulation did find a solution, using only 28 nodes.

It should be noted that the introduction of auxiliary variables can possibly introduce a large number of variables and constraints. However,we found that the size of the formulation was significantly reduced by standard IP preprocessing (similar to the use of Graphplan as a preprocessing tool in BlackBox). For example, while the initial formulation of the problem "rocket.a" had 27744 variables and 40018 constraints, preprocessing reduced this to 1573 variables and 3007 constraints. Similar reductions in size were also obtained for the other problems.

A further indication of the strength of respective formulations can be found by examining the value of the LP relaxations. Since the objective function that is used is to minimize the number of actions in the plan, the value of the LP relaxation may also viewed as a lower bound on the number of actions required in the plan. The results for the SATPLAN-based and the state-change formulation are shown in Table 2. In almost all cases, the state-change formulation has a much higher lower bound, which indicates that its formulation is indeed much stronger.

In the SATPLAN and Graphplan framework, the *parallel length* (i.e., the number of time steps) of plans is minimized. The IP formulation follows this framework,

Table 1: Experimental results: IP formulations vs. Systematic BlackBox solver.

Problem	SATPLAN IP			State-change IP			BlackBox/Satz	
	nodes	its.	time	nodes	its.	time	nodes	time
anomaly	59	1471	3.1	3	161	0.1	3	0.55
bw-12step	*	*	*	4	2037	9.7	3	2.42
bw-large.a	*	*	*	4	4261	36	38	20.8
bw-large.b	*	*	*	28	89048	2500	–	–
att-log2	491	2748	4.9	24	177	0.57	7	0.56
att-log3	99	1296	12.1	33	406	4.2	16	0.58
att-log4	1179	20778	101.4	40	961	5.7	23	0.56
rocket.a	*	*	*	213	40877	140	234	3.37
rocket.b	*	*	*	73	18492	67	630	6.38
log-easy	*	*	*	102	2505	5.8	16	0.62
logistics.a	*	*	*	40	9305	80	41	1.66
logistics.b	*	*	*	30	9532	92	46	2.37
logistics.c	*	*	*	285	89760	1400	39297	79.3

– denotes that no plan was found after 10 hours of computation time.
* denotes that the node limit of 2500 was reached without finding a feasible integer solution.

Table 2: LP relaxation values.

Problem	SATPLAN-based	State-change
anomaly	2.62	5
bw-12step	2.33	5
bw-large.a		12
bw-large.b		16
att-log2	2.19	6.75
att-log3	1.57	6.75
att-log4	2.89	10.7
rocket.a	12.73	20.6
rocket.b		20.6
log-easy	5.28	19.25
logistics.a		42.8
logistics.b		30.9
logistics.c		38.9

so it also is guaranteed to minimize the number of time steps. In addition, the IP formulation also explicitly uses minimization of the number of actions in the objective function. Since we set the solver to terminate as soon as the first feasible integer solution was found, the IP solutions were not guaranteed to minimize the possible number of actions. Still, we found that in most cases, the IP formulation found plans with a significantly smaller total number of actions than those obtained by BlackBox. This is shown in Table 3.

4 Conclusions

Although Selman *et al.* [1997] reported difficulty in making effective use of IP techniques for propositional reasoning in general, our results suggest that IP techniques may potentially work well for AI planning problems, for the following reasons.

- First, the IP formulation has the potential to do efficient planning. In our results, the number of nodes expanded in the search space was typically small, and comparable to a systematic satisfiablity solver. This indicates that the LP relaxation gave significant guidance in the selection of variables in solving planning problems.

Table 3: Plan length comparison (total number of actions in plan).

Problem	State-change IP	BlackBox/Satz
rocket.a	30	33
rocket.b	26	29
log-easy	25	25
logistics.a	60	72
logistics.b	47	68
logistics.c	66	90

- Second, IP models may provide a natural means of incorporating numeric constraints and objectives into the planning formulation. This capability would be important in many application domains, but it is not available in most existing approaches to AI planning. It should be noted, however, that the way in which numeric constraints will be represented may have a significant influence on the performance, much in the same way as we saw with the various IP formulations. Therefore, the development of strong IP representations that capture common numeric constraints that arise in the planning domain is an issue for further research.

We would like to emphasize that so far our main concern has been the development of different IP formulations, rather than improving the efficiency of the LP relaxation itself. While we believe that the state-change formulation is reasonably strong, solving the LP relation at each node is still sometimes computationally expensive. One

of the main reasons for this, we believe, is the degeneracy of the LP relaxation (a condition that can cause the LP solver to execute many non-productive iterations). Therefore, we are currently also investigating techniques to resolve this degeneracy, as well as further strengthening of the IP formulation. In particular, we want to investigate the use of constraint and column generation techniques.

References

[Blair *et al.*, 1986] Blair, C.E., Jeroslow, R.G., and J.K. Lowe. 1986. Some results and experiments in programming techniques for propositional reasoning. *Computers and Operations Research* 13:633–645.

[BockMayr and Dimopoulos, 1998] Alexander Bockmayr and Yanis Dimopoulos. 1998. Mixed Integer Programming Models for Planning Problems. *CP'98 Workshop on Constraint Problem Reformulation.*

[Bylander, 1997] T. Bylander. 1997. A Linear Programming Heuristic for Optimal Planning. *Proc. AAAI-97.*

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

[Hooker, 1988] J. N. Hooker. 1988. A quantitative approach to logical inference. *Decision Support Systems* 4:45–69.

[Kautz *et al.*, 1996] Henry Kautz, David McAllester, and Bart Selman. 1996. Encoding plans in propositional logic. *Proc. KR-96.*

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. *Proc. AAAI-96.*

[Kautz and Selman, 1998] Henry Kautz and Bart Selman. 1998. BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. *Working notes of the Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98, Pittsburgh, PA, 1998.*

[Kautz and Walser, 1999] Henry Kautz and Joachim P. Walser. 1999. State-space Planning by Integer Optimization. *Proc. AAAI-99.*

[Nau *et al.*, 1998] D. S. Nau, S. J. Smith and Kutluhan Erol. 1998. Control strategies in HTN planning: theory versus practice. In *AAAI-98/IAAI-98 Proceedings*, 1127–1133, 1998.

[Selman *et al.*, 1997] B. Selman, H. Kautz, and D. McAllester. 1997. Ten challenges in propositional reasoning and search. In *Proc. Fifteenth International Joint Conf. Artificial Intelligence (IJCAI-97)*, Nagoya, Japan.

[Wolsey, 1998] L. Wolsey, *Integer Programming*, 1998, John Wiley, New York.