Optiplan: Unifying IP-based and Graph-based Planning

Menkes van den Briel[1] **and Subbarao Kambhampati**[2]
[1]Department of Industrial Engineering, [2]Department of Computer Science and Engineering
Arizona State University, Tempe AZ

Abstract

The Optiplan planning system combines the ideas presented by Vossen *et al.* (1999) and Kautz and Selman (1998). It unifies integer programming with graph-based planning and computes optimal parallel length plans for STRIPS based planning problems. In addition, given a feasible parallel length, Optiplan can be used to minimize the number of actions, minimize action cost, or optimize any other objective that can be expressed as a linear function.

OptiPlan

Optiplan is a domain independent planner that, like ILP-PLAN (Kautz & Walser 1999) and the "state change model" (Vossen *et al.* 1999), uses integer programming (IP) to solve STRIPS planning problems. The architecture of Optiplan is very similar to that of Blackbox (Kautz & Selman 1999) and GP-CSP (Do & Kambhampati 2001), but instead of unifying satisfiability or CSP with graph based planning, Optiplan uses integer programming. Like Blackbox and GP-CSP, Optiplan works in two phases. In the first phase the planning graph is build and transformed into an IP, then in the second phase the IP is solved using the commercial solver CPLEX (ILO 2002). The IP formulation is based on the state change formulation (Vossen *et al.* 1999), however, a few changes have been added that "strengthen" the original formulation and make it more general at the same time.

A practical difference between the state change model and Optiplan is that the former takes as input all ground actions and fluents over all time steps, while the latter takes as input just those actions and fluents that are instantiated by Graphplan (Blum & Furst 1995). The use of a planning graph has a significant effect on the size of the final encoding, independent of which combinatorial transformation method (IP, SAT, or CSP) is used. For example, Kautz and Selman (1999) pointed out that Blackbox's success over Satplan was mainly explained by Graphplan's ability to produce better, more refined, propositional structures than Satplan. Another, although minor, practical difference between Optiplan and the state change model is that Optiplan reads in pddl files, allowing it to be directly compared to other STRIPS based planners.

In order to present the improved state change formulation that is used in Optiplan we introduce the following sets and variables: (The reader familiar with the work by Vossen *et al.* (1999) may want to skim through the formulation of the model and note that the variables $x^{predel}_{f,i}$, for all $f \in F, i \in 1, ..., t$ have been deleted and the variables $x^{del}_{f,i}$, for all $f \in F, i \in 1, ..., t$ have been added to the original formulation.):

- F, set of *fluents*, the set of all instantiated propositions;
- A, set of *actions*, the set of all instantiated operators;
- $I \subseteq F$, set of fluents that are true in the initial state;
- $G \subseteq F$, set of fluents that must be true in the goal state;
- $pre_f \subseteq A, \forall f \in F$, set of actions that have fluent f as precondition;
- $add_f \subseteq A, \forall f \in F$, set of actions that have fluent f as add effect;
- $del_f \subseteq A, \forall f \in F$, set of actions that have fluent f as delete effect;

The state change formulation defines variables for each step i in the planning graph. There are variables for the actions and there are variables for the possible state changes a fluent can make. For all $a \in A, i \in 1, ..., t$ we have the action variables

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

The "no-op" actions are not included in the y_a, i variables but are represented separately by the state change variable $x^{maintain}_{f,i}$. For all $f \in F, i \in 1, ..., t$ we have the state change variables

$$x^{maintain}_{f,i} = \begin{cases} 1 & \text{if fluent } f \text{ is propagated in period } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$x^{preadd}_{f,i} = \begin{cases} 1 & \text{if action } a \text{ is executed in period } i \\ & \text{such that } a \in pre_f \cap a \notin del_f, \\ 0 & \text{otherwise.} \end{cases}$$

$$x^{add}_{f,i} = \begin{cases} 1 & \text{if action } a \text{ is executed in period } i \\ & \text{such that } a \notin pre_f \cap a \in add_f, \\ 0 & \text{otherwise.} \end{cases}$$

$$x^{del}_{f,i} = \begin{cases} 1 & \text{if action } a \text{ is executed in period } i \\ & \text{such that } a \notin pre_f \cap a \in del_f, \\ 0 & \text{otherwise.} \end{cases}$$

In summary: $x_{f,i}^{maintain} = 1$ if the truth value of a fluent is propagated; $x_{f,i}^{preadd} = 1$ if an action is executed that requires a fluent and does not delete it; $x_{f,i}^{add} = 1$ if an action is executed that does not require a fluent and adds it; and $x_{f,i}^{del} = 1$ if an action is executed that does not require a fluent and deletes it.

There are a few differences with the original state change formulation and the formulation in Optiplan. Optiplan introduces the $x_{f,i}^{del}$ variables in order to deal with actions that delete fluents without requiring them as preconditions. Many planning domains in the International Planning Competition 2004 have such actions, making the original state change formulation ineffective. In addition, the new formulation has substituted out all $x_{f,i}^{predel}$ variables by the expression $\sum_{a \in pre_f \cup del_f} y_{a,i}$. The updated formulation is given by:

$$\min \quad \sum_{a \in A} \sum_{i \in T} y_{a,i} \tag{1}$$

$$\text{s.t.} \quad x_{f,0}^{add} = 1, \forall f \in I \tag{2}$$

$$x_{f,0}^{add} = 0, \forall f \notin I \tag{3}$$

$$x_{f,t}^{add} + x_{f,t}^{maintain} + x_{f,t}^{preadd} \geq 1 \tag{4}$$

$$\sum_{a \in add_f / pre_f} y_{a,i} \geq x_{f,i}^{add} \tag{5}$$

$$y_{a,i} \leq x_{f,i}^{add} \tag{6}$$

$$\sum_{a \in pre_f / del_f} y_{a,i} \geq x_{f,i}^{preadd} \tag{7}$$

$$y_{a,i} \leq x_{f,i}^{preadd} \tag{8}$$

$$\sum_{a \in del_f / pre_f} y_{a,i} \geq x_{f,i}^{del} \tag{9}$$

$$y_{a,i} \leq x_{f,i}^{del} \tag{10}$$

$$x_{f,i}^{add} + x_{f,i}^{maintain} + x_{f,i}^{del} + \sum_{a \in pre_f \cup del_f} y_{a,i} \leq 1 \tag{11}$$

$$x_{f,i}^{preadd} + x_{f,i}^{maintain} + x_{f,i}^{del} + \sum_{a \in pre_f \cup del_f} y_{a,i} \leq 1 \tag{12}$$

$$x_{f,i}^{preadd} + x_{f,i}^{maintain} + \sum_{a \in pre_f \cup del_f} y_{a,i} \leq x_{f,i-1}^{preadd} + x_{f,i-1}^{add} + x_{f,i-1}^{maintain} \tag{13}$$

$$x_{f,i}^{preadd}, x_{f,i}^{add}, x_{f,i}^{del}, x_{f,i}^{maintain} \in \{0,1\} \tag{14}$$

$$y_{a,i} \in \{0,1\} \tag{15}$$

Where constraints (2), and (3) represent the initial state constraints, and (4) represent the goal state constraints. For all $f \in \mathcal{F}, i \in 1, ..., t$, constraints (5) to (10) represent the logical interpretations between the action and state change variables, and for all $f \in \mathcal{F}, i \in 1, ..., t$ constraints (11) and (12) make sure that fluents can only be propagated at period i if and only if there is no action in period i that adds or deletes the fluent. For all $f \in \mathcal{F}, i \in 1, ..., t$, constraints (13) describe the backward chaining requirements. Constraints (14) and (15) are the binary constraints for the state change and action variables respectively. Since the constraints guarantee plan feasibility, no objective function is required, however, Optiplan uses an objective that minimizes the number of actions taken to guide the search.

Optiplan shows an increased performance over the original state change encoding, but it remains significantly slower than, for example, Blackbox(Chaff). Table 1 shows a comparison between the original state change formulation and Optiplan on a set of problems that we could test both encodings on. All tests were run on a Pentium 2.67 GHz with 1.00 GB of RAM and the IP encodings were solved using CPLEX 8.1. For all problems Optiplan creates smaller encodings than the original state change formulation, and in all but two instances (the two rocket problems) Optiplan's formulation is solved at least as fast as the original state change formulation.

Often times only a few nodes are explored in the branch-and-bound tree, this indicates that the LP relaxation provides a good approximation to the convex hull of integer solutions. Still, however, our IP approaches are easily outperformed by planners like Blackbox(Chaff). Possible reasons for this performance gap is that the CPLEX's integer programming solver is not specialized in solving pure 0-1 programming problems and because many "expensive" matrix operations are required when solving the LP relaxation. When these shortcomings are resolved, for example, through the use of special purpose algorithms like branch-and-cut, decomposition, or column generation, Optiplan and IP approaches in general could become competitive with other successful planners.

References

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1636–1642.

Do, M., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into csp. *Artificial Intelligence* 132(2):151–182.

ILOG Inc, Mountain View, CA. 2002. *ILOG CPLEX 8.0 User's Manual*.

Kautz, H., and Selman, B. 1999. Blackbox: Unifying sat-based and graph-based planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 318–325.

Kautz, H., and Walser, J. 1999. State-space planning by integer optimization. In *Proceedings of the 17th National Conference of the American Association for Artificial Integlligence*, 526–533.

Vossen, T.; Ball, M.; Lotem, A.; and Nau, D. 1999. On the use of integer programming models in ai planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 304–309.

	State change model				Optiplan			
Problem	#Var.	#Cons.	#Nodes	Time	#Var.	#Cons.	#Nodes	Time
bw-sussman	196	347	0	0.01	105	142	0	0.01
bw-12step	1721	3163	15	4.53	868	1040	4	1.58
bw-large-a	2729	5106	0	5.04	1800	2104	0	3.91
bw-large-b	6502	12224	25	932.26	4780	5466	9	236.45
att-log0	33	41	0	0.01	6	8	0	0.01
att-log1	151	188	0	0.01	49	71	0	0.01
att-log2	330	420	14	0.05	130	193	0	0.01
att-log3	2334	3785	0	0.26	250	455	0	0.06
att-log4	2330	3775	42	0.59	449	850	0	0.12
att-loga	3146	5091	3583	366.44	1671	3258	80	29.84
rocket-a	1615	2694	169	8.80	1127	2365	49	12.38
rocket-b	1696	2829	122	8.27	1187	2516	27	11.58
log-easy	1521	2254	32	0.86	555	1088	0	0.14
log-a	3933	6306	174	48.36	1671	3258	80	29.74
log-b	4684	7202	1797	391.75	1962	3830	41	40.67
log-c	5886	9324	1378	946.23	2691	5370	114	183.96

Table 1: Comparing the original state change formulation with Optiplan. #Var. and #Cons. give the number of variables and constraints after CPLEX's presolve. #Nodes give the number of nodes that were explored during branch-and-bound before finding the first feasible solution.