This paper appears in the *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98), Pittsburgh, PA, 1998.*

The Role of Domain-Specific Knowledge in the Planning as Satisfiability Framework

Henry Kautz
AT&T Laboratories
180 Park Avenue
Florham Park, NJ 07932
kautz@research.att.com

Bart Selman
Department of Computer Science
Cornell University
Ithaca, NY 14853
selman@cs.cornell.edu

Abstract

SATPLAN is currently one of the fastest planning systems for domain-independent planning. In nearly all practical applications, however, there exists an abundance of domain-specific knowledge that can be used to improve the performance of a planning system. This knowledge is traditionally encoded as procedures or rules that are tied to the details of the planning engine. We present a way to encode domain knowledge in a purely declarative, algorithm independent manner. We demonstrate that the same heuristic knowledge can be used by completely different search engines, one systematic, the other using greedy local search. This approach greatly enhances the power of SATPLAN: solution times for some problems are reduced from days to seconds.

Introduction

Planning is a notoriously hard combinatorial search problem. In Kautz and Selman (1996) we considered domain-independent, state-space planning as a computational challenge. This style of planning is a core problem in AI; similar computational issues arise in many other models, such as reactive planning, planning with uncertainty and utilities, planning with metric time, and so on. We showed how planning problems could be converted into large propositional satisfiability problems, and then solved using highly-optimized search procedures. In particular, we showed that Walksat (Selman *et al.* 1994) could be used to solve certain hard planning problems that were beyond the capabilities of previous specialized planning systems.

The SATPLAN system operates by converting a planning problem into a set of axiom schemas. These schemas are instantiated for plans of a fixed given length. The instantiated formulas are solved by a SAT engine, and the plan can be read off of the satisfying model. If no model is found, the length parameter is incremented and the problem is re-instantiated, until it can be proven to be consistent.

The benchmark planning testbed used in both our early paper and this one is described in Table 1. There are a series of problems from a logistics domain, that involve moving packages between locations in various cities by truck and by airplane. Non-conflicting actions in this domain may occur in parallel, and each action takes one time step. For example, the best known solution to the problem logistics.a contains 54 actions, but they are parallelized in such a manner that they can be executed over a span of 11 time steps. The largest logistics problem solved in our 1996 paper involved 10^{10} possible configurations of packages and vehicles. In this paper we have solved a much harder instance involving 10^{16} configurations. (This corresponds to a logistics problem with 9 packages, 5 trucks, 2 airplanes, and 15 locations.)

problem	soln length	soln size	configurations
log.a	11	54	10^{10}
log.b	13	47	10^{8}
log.c	13	63	10^{10}
log.d	14	74	10^{16}
bw.a	6	12	10^{6}
bw.b	9	18	10^{9}
bw.c	14	28	10^{13}
bw.d	18	36	10^{19}

Table 1: Planning benchmark testbed. For logistics problems, the solution length is optimal, and the number of actions per solution is the smallest that has been found using any search procedure so far. For the blocks world (one arm), the solution length and sizes are optimal (each time step includes a pickup/putdown pair).

Note that the state-space of the SAT encoding of these instances is much larger than the number of configurations. For example, the largest problems in this paper contain about 2,000 Boolean variables, for a state-space of size 2^{2000}. It has been interesting to discover that we can efficiently search this "exploded" state-space. We also worked on a series of traditional blocks world problems involving up to 19 blocks and 10^{19} configurations.

Figure 1 summarizes the main results from that paper. It compares the performance of the highly-efficient Graphplan system of Blum and Furst (1995) with the

---

Copyright (c) 1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

SATPLAN approach. The problems "rocket.a" and "rocket.b" are simpler versions of our logistics problems. The graph refers to the time required to search an instantiated data structure for a solution: in the case of Graphplan, a graph, and in the case of SATPLAN, a CNF formula. Times to instantiate and pre-process the problems for SATPLAN ranged from 42 seconds (log.a) to 271 seconds (log.d). Because Graphplan does not separate out the instantiation and search times, we subtracted the SATPLAN instantiation times from the Graphplan numbers. (The Graphplan implementation in fact included a more efficient instantiation routine than did SATPLAN.)

The chart compares Graphplan with four variations of SATPLAN. The "ntab" variations employ an implementation (Crawford and Auton 1993) of the classic Davis-Putnam-Loveland systematic search procedure, while the "Walksat" variations use stochastic local search. (Our specific version of local search is based on the GSAT / Walksat algorithm (Selman *et al.* 1992; 1994).) The "STRIPS" variations used SAT encodings automatically generated from a STRIPS-style problem specification: the code to do this was based on a modified version of the Graphplan implementation. The "STATE" variations use a hand-crafted "state-based" encoding, as described in Kautz and Selman (1996).

We see that in this domain the SATPLAN approaches dominate. Best performance is obtained with stochastic local search and state-based encodings. Two caveats are in order. First, in other domains Graphplan or other planning systems can outpeform SATPLAN; indeed, Graphplan is comparable with SATPLAN on the blocks world instances from the testbed. Second, the state-based encodings are quite different from STRIPS, and are not currently automatically generated from STRIPS operators. However, we have argued that state-based axioms are a natural and convenient way to specify many planning domains: there is no need to insist that STRIPS-notation is "primary". Furthermore, this paper will show that the general approach of specifying a problem domain by a set of axiomatic constraints on changes between states makes it straightforward to incorporate many different kinds of heuristic knowledge.

Given the computational difficulty of the planning problem, one might wonder why it has ever been possible to deploy traditional planning systems in real applications. The reason is that practical systems minimize search by using techniques such as domain-specific control rules (Bacchus and Kabanza 1995, Carbonell *et al.* 1992, Penberthy 1992, Sacerdoti 1977, Slaney and Thiebaux 1996, Veloso 1992, and Weld 1994), "compiled" reactive plans (Agre and Chapman 1987, Williams and Nayak 1997), and heuristics based on temporal and resource constraints (Vere 1985, Muscettola 1994). However, scaling remains problematic in many interesting domains.

All these techniques involve incorporating more domain specific knowledge into the planner. The natural question arises: can the power of knowledge also be incorporated into the planning as satisfiability framework, without sacrificing its elegance or generality?

Domain dependent planning

SATPLAN's use of efficient propositional representations and search engines dramatically increased the size of problems that could be solved. One path to further improvement is to apply newer, faster satisfiability testing procedures, and indeed we will briefly mention some promising preliminary experiments with the *satz* system of Li and Anbulagan (1997). The other approach, which is the focus of this paper, is to encode more knowledge in the problem representations.

There several kinds of knowledge that could be added to a problem representation. First, there is *knowledge about the domain itself.* For instance, a fact about the logistics domain is that "a truck is only in one location". Second, there is *knowledge about good plans*, for example, "do not return a package to a location from which it has been removed". Any plan violating this constraint is obviously suboptimal. Third, one could encode explicit control knowledge about search, such as "plan air routes before land routes".

Such information is traditionally incorporated in the planning algorithm itself or in a special programming language interpreted by the planner. Instead, we propose expressing domain knowledge as additional declarative axioms. Thus, a problem instance will be represented as the union of three sets of axioms: those for operators, those for the initial and goal states, and those for additional domain knowledge, which we will call "heuristic axioms". Domain information simply functions as a constraint on the search and solution spaces, and is independent of any search engine strategy. It is important to note that domain knowledge has exactly the same status as any of the other problem constraints. In particular, the domain knowledge is *not* a "meta-constraint" on the search algorithm, as is, for instance, the "cut" operator in logic programming. (Related to our approach is that of Bacchus and Kabanza (1995), who also declaratively specify heuristic axioms for a planning system. It differs in that they used a detailed modal temporal logic specification to tightly control the PRODIGY planning system, whereas we use propositional axioms to assist, but not replace, search.)

This paper builds on the observations of Ernst, Millstein, and Weld (1997) concerning the effect of domain-specific axioms on our original blocks-world testbed. They noted that including state-invariant axioms made the problems larger before simplification (as described below), but smaller and easier to solve after simplification.

The logical status of heuristics

The straightforward, logical representation of heuristic knowledge we have adopted allows us to determine the logical relationship between that knowledge and the original problem statement. It allows us to distinguish

Invariant: A truck is at only one location

$$at(truck, loc1, i) \wedge loc1 \neq loc2 \supset \neg at(truck, loc2, i)$$

Optimality: Do not return a package to a location

$$at(pkg, loc, i) \wedge \neg at(pkg, loc, i+1) \wedge i < j \supset \neg at(pkg, loc, j)$$

Simplifying: Once a truck is loaded, it should immediately move

$$\neg in(pkg, truck, i) \wedge in(pkg, truck, i+1) \wedge at(truck, loc, i+1) \supset \neg at(truck, loc, i+2)$$

Table 2: Example axiomatic form of logistics heuristics.

between heuristics that add entirely new constraints, and those that simply make *explicit* information that is implicit in other parts of the problem. In fact, the categories defined by the logical relationship of the heuristics to the rest of the problem instance creates a natural classification of kinds of heuristics. Consider the following four classes:

1. Heuristics entailed by the operator axioms alone are those for *conflicts and derived effects*. For example, any two simultaneous *fly* actions for the same airplane conflict. For "pure" STRIPS operators, conflicts can be easily derived, by simply comparing the precondition and delete lists of operators. When derived predicates and auxiliary axioms are included, however, determining action conflicts can become NP-complete. In such cases it can be helpful to add heuristic axioms that make the conflicts explicit.
2. Heuristics entailed by the operator and initial state axioms together include *state invariants*. For example, the typical STRIPS formulation of the operator for moving a vehicle from one location to another, $drive(truck, origin, dest)$, does not entail that a truck is always at a single location. However, if in the initial state every truck is at exactly one location, the operator will propagate that invariant to all future states.
3. Heuristics entailed by the operators and initial state axioms, together with the plan length n are a kind of *optimality condition*. Assuming that the given plan length is the minimum needed to solve the problem, then the fact that the plan contains no redundant or unnecessary steps logically follows. Optimality conditions can be hard to infer: as hard, in fact, as solving the problem instance, since the full set of optimality conditions would rule out all actions not in a solution. However, although difficult to *automatically* derive, at least some of the "obvious" optimality conditions are easy for a person to encode. The heuristic we noted earlier, "do not return a package to a location from which it has been removed", is such an obvious condition.
4. Finally, new constraints on problem instance that are not entailed by any of the previous axioms are *simplifying assumptions*. Since they are not entailed, they must actually rule out some of the valid solutions to the problem; thus, they introduce incompleteness into the heart of the planning system. However, computational considerations already make virtually all planners incomplete *in practice*, and in many cases one can prove the "lost" solutions never transform a solvable instance into an inconsistent one. In the logistics domain, the heuristic "once a truck is loaded, it should immediately move" is such a "safe" simplifying assumption, because any solution that violates it can be made into one that satisfies it by delaying *load* actions.

Examples of the axiomatic forms of some the heuristics used in the experiments described below appear in Table 2.

How can one can go about developing heuristics for a problem domain? The traditional approach, and the one followed in this paper, is to simply employ introspection to capture both "obvious" inferences that are hard to deduce mechanically (McAllester 1991) and simplifying assumptions that follow from abstracting the essence of the domain. It is clearly a matter of debate whether the introspective approach is feasible in the long run. The common complaint is that manually-created heuristics are hard to maintain as a complex system evolves over time. However, in previous work such heuristics are encoded either as procedures or as statements in a meta-language that directly controls the operation of a planning algorithm. By contrast, in our approach the heuristic axioms refer only to constraints on states and plans — not *how* plans are found. Thus, they may prove to be no harder to write or maintain than any other part of the domain specification.

A different (complementary) approach is to automatically generate heuristics. There has been a great deal of work on the problem of heuristic generation, including that on explanation-based learning (Minton 1988, Kambhampati *et al.* 1996), static analysis of operator schemas (Smith 1989, Etzioni 1993), problem abstraction (Knoblock 1994), and operator-graph analysis (Smith and Peot 1996). All of this work has aimed at producing explicit search control rules: for example, rules that would state when to prefer one operator application over another, or when to cut off a branch of recursive search. In our framework, however, we instead wish to generate constraints on admissible plans and sequences of events, in order to reduce and simplify the search space. It would be interesting to investigate how to modify techniques such as EBL so that the output is a set of such constraints.

There is also a relationship between heuristic generation and what has been called "theory compilation" (Selman and Kautz 1996). For example, one could fix the operators and the goal state, generate a number of consequences, and add them back to the original theory. In the worse case, the compiled theory is exponentially large, but the general approach can be efficient in practice (Williams and Nayak 1997). Another area that investigates potentially useful techniques is that of simplification of search problems by the discovery of "symmetries". For example, Joslin and Roy (1997) add "symmetry breaking predicates" to a theory based on the discovery of symmetries in operator schema; in our vocabulary, this would be a case of a "safe simplifying assumption".

Finally, any system that automatically generates heuristics must amortize the cost of that generation over some set of problem instances. Heuristics such as type (1) above could be generated by analysis of the operators alone, and thus amortized over all problem instances in that domain. Heuristics based on both the operators and some information about the specific problem instance, such as types (2) and (3), could only be amortized over a subset of the domain.

Experiments

We designed a series of experiments to test the feasibility of specifying declarative domain knowledge. It is not obvious that it would work: the heuristics might simply blow up the size of the problem with redundant axioms, and simply make the problem harder to solve.

The first series of experiments were based on the blocks world, where there is a single arm and no parallel actions. The number of "move" actions (a move is equivalent to a pickup and putdown pair) in the optimal solutions ranged from 6 to 19. We created three versions of each problem instance: The first included axioms for the move operator only. These instances are labeled bw.a, bw.b, bw.c, and bw.d. The second added axioms for state invariants of the predicate "on"; for example, that at most one block could be on another. These examples are labeled "bw.a.state", etc. The third version also included optimality heuristics, namely: never more a block twice in a row; once a block is moved onto a block (not the table) it stays there; and never move a block more than twice. These examples are labeled "bw.a.opt", etc. Because there is only a single predicate "on" and no parallel actions, there is no need for the class of heuristics for conflicts and derived effects. The blocks world problems that appear in Kautz and Selman (1996) are the "state" versions of this paper. These experiments expand on the ones performed in Ernst *et al.* (1997), which considered only the "move" and "state" versions.

Each version of each problem was instantiated, simplified by unit propagation, and then solved by the "ntab" implementation of the Davis-Putnam-Loveland procedure (Crawford and Auton 1993), and by Walksat. The number of time steps was set to the smallest (optimal) value. Because Walksat is a randomized procedure, we took the the average timings for 100 runs with different random seeds. (Simplification by unit propagation can be done in linear time, and for problems of this size takes less than one second with a good C implementation; for convenience we used a simple shell-script that required about 20 seconds.) All times are on a 200 Mhz SGI Challenge.

The results are summarized in Table 3. First, let us consider the size of the problems. We see that before the formulas were simplified ("orig. no. vars/clauses"), adding the heuristic axioms increased the number of variables by about 10%, and increased the number of clauses by 300% — 600%. For example, bw.d contained 62,734 clauses, but bw.d.opt contained 388,755 clauses. However, simplification by unit propagation made the heuristic instances significantly smaller in the number of distinct variables. For the simplified versions of bw.d, the number of variables dropped from 7,132 (no heuristics) to 6,325 (state), and dramatically to 2,174 (opt). The final number of clauses was *highest* for the "state" versions, and smallest for the "opt" versions.

Turning to the timings, we see that our stochastic algorithm Walksat was fastest on the "state" version, except for bw.b, where the "opt" version was slightly easier to solve. On the other hand, the "opt" versions were significantly easier to solve for the systematic algorithm ntab, except for bw.c, where "state" was slightly better, and bw.d, where for Walksat only "state" was comparable. Problems bw.c and bw.d could not be solved in less than 48 hours by either routine without the addition of the heuristic axioms.

These results demonstrated that the general approach was sound, and could extend the SATPLAN approach to planning to classes of larger and harder problems. They also suggested that decreasing the number of variables after simplification was more important than decreasing the number of clauses, and that the point of diminishing returns from the addition of axioms would be sooner reached for stochastic search than for systematic search.

The second series of experiments were based on the logistics domain mentioned earlier. The first three instances also appeared in Kautz and Selman (1996), but for this paper we also created a much harder instance, logistics.d. As described in our earlier paper, our axiomatization is "state based" in that it represents invariants on states and between pairs of adjacent states, but does not explicitly represent the action operators. The actions in a plan can be derived in linear time from a satisfying model of the problem instance. Because there are no operators represented, there can be no additional axioms for conflict heuristics. Similarly, there can be no additional state invariant heuristics, because such invariants are already part of the representation. Therefore we studied the effect of optimality heuristics and simplifying assumptions in this domain.

There were a number of questions we wished to investigate:

- Are simplifying assumptions and optimality conditions useful heuristics for stochastic search? In the blocks world, the optimality heuristics helped sys-

wff	orig no. vars	orig no. clauses	final no. vars	final no. clauses	walksat time	ntab time
bw.a	804	5118	534	3060	1.84	0.26
bw.a.state	973	14582	459	4710	0.03	0.23
bw.a.opt	946	16391	294	2448	0.55	0.12
bw.b	1635	11091	1235	7457	13.08	5.13
bw.b.state	1865	35971	1087	13845	2.22	0.66
bw.b.opt	1876	42230	564	5759	0.75	0.38
bw.c	4258	30986	3526	22535		
bw.c.state	4723	126866	3016	50621	3.84	16.5
bw.c.opt	4738	155741	1225	17082	77	25.7
bw.d	8282	62734	7132	47098		
bw.d.state	9023	311862	6325	132257	688	
bw.d.opt	9042	388755	2174	40384	643	2651

Table 3: Effect of heuristic axioms on blocks world problems. Times in seconds. No entry means procedure failed to solve problem after more than 48 hours.

tematic search, but not stochastic search. Furthermore, because stochastic search is particularly good on under-constrained problems, simplifying assumptions might hurt Walksat rather than help.

- Would the overall positive impact of the additional axioms be greater on systematic or stochastic methods?
- Is problem size in terms of number of variables the more important factor, as in the blocks world? Would the optimal level of heuristics again differ for the two algorithms?

The question of the impact of the heuristics on systematic versus stochastic search was particularly intriguing, because there are intuitive grounds for either answer. One could argue that because the additional axioms enable more unit propagations during search, and systematic methods handle unit propagation more efficiently, the heuristics would be more beneficial to systematic search. Alternatively, one could argue that the additional axioms would *shorten* chains of unit propagation, and therefore be more helpful to stochastic algorithms, which handle long chains more slowly.

We began this set of experiments by writing down four sets of heuristics. We then performed a few preliminary experiments to see which set of heuristics, taken alone, usually provided the greatest improvement in solution time. We then ordered the heuristics from most useful to least useful. Finally, we made each set properly include all the previous (more useful) heuristics. The result gives five versions of each problem instance, labeled "none" (no extra axioms), h1, h2, h3, and h4. Each version is a strict superset of the clauses in the previous version.

The axioms that first appear in each set are:

h1 (optimality condition): Once a package leaves a location, it never returns to that location.

h2 (simplifying assumption): A package is never in any city other than its origin or destination cities. (This is not an optimality condition because it rules out solutions where packages are transferred between airplanes in an intermediate city.)

h3 (simplifying assumption): Once a vehicle is loaded, it should immediately move.

h4 (optimality condition): A package never leaves its destination city.

Again the problems were instantiated, simplified, and solved by both Walksat and ntab. Figures 2–5 summarize the results of the experiments. Copies of the data and the formulas themselves are available from the authors.

We first consider the size of the formulas. Figures 2 and 3 plot the number of variables and the number of clauses after simplification against the heuristic version for each of the problems log.a, log.b, and log.c. The data for log.d follows the same general curve, but is not plotted because it would too greatly compress the scale of the Y-axis. The values for log.d are each about twice that of the corresponding values for log.c.

Only the simplifying assumption h2 decreased the number of variables. This is because it rules out all propositions that represent a package being at a location that is not in the origin or destination cities. Considering the number of clauses after simplification, we see that h1 increased the size of the formula by about 30%, but when h2 is added, the formula is about 30% *smaller* than the original one. Increasing the heuristic set to h3 and h4 caused a gradual increase in the number of clauses. Thus, h2 represents a point at which both the number of variables and the number of clauses is minimized.

Figure 4 is based on the timings from Walksat. The Y-axis is normalized to represent multiples of the minimal solution time for the problem instance across variants. For example, log.c.none required 6 times as much time to solve as log.c.h2. For log.a, log.c, and log.d the h2 variants were fastest to solve, and for log.b the h3 variant was fastest. We see that the greatest decrease in solution time came from the addition of h1; recall that h1 *increased* the number of clauses and did *not* decrease the number of variables. There was a further slight decrease at h2, where the problem size *was* minimized. However, the decrease at h1 over "none" indicates that problem size was not the only factor involved in making the problems easier to solve. Set h2 was the point of diminishing returns for Walksat. There was a slight increase at h3, and either a slight increase or decrease at h4.

The timings for systematic search in Figure 5 are more dramatic. Here the normalized solution times are plotted on a log scale. None of the problems could be solved without the additional axioms, nor could any version of log.d. For the smallest problem, log.a, the fastest time was realized at h4, but log.b and log.c were solved most quickly at h2. Again the most important heuristic was the optimality condition h1, because it made unsolvable problems solvable. However, the simplifying assumption h2 was also important, particularly for log.b, where it reduced the solution time by a factor of 38,000. (In absolute terms, from about 39 hours to about 3 seconds.)

Let us now consider the questions we raised earlier. First, it is clear that simplifying assumptions as well as optimality conditions can enhance the power of stochastic search as well as systematic search. We saw a 6-fold improvement for Walksat on the largest, hardest problem instances. At least on this benchmark set, the worry that reducing the number of solutions would make the problems harder for local search was unfounded.

Second, the greatest impact of the additional axioms was again for the systematic search engine. We saw that for ntab problems that could not be solved in two days could then be solved in a few seconds. As described below, our future work will examine why this is the case. As we have mentioned, our current hypothesis is the additional axioms greatly increase the number of unit propagations at each branch-point.

Third, both algorithms were usually optimal on the variations of the problems that minimized both the number of variables and the number of clauses. However, problem size was not the only or most important factor in determining the difficulty of solution. This is notable in the h1 versions, which were strictly larger than the versions without any heuristic axioms, but were much easier to solve.

For both algorithms the point of diminishing returns was reached at h2. This may have been because few optimally-short plans would violate h3, and so it had little impact. It also appears that h4 follows with a fairly short proof from h1 and h2. It is an open question whether there are other natural heuristics that are more powerful than h2.

Conclusions and Future Work

This paper is a preliminary report on a promising new method of enhancing the planning as satisfiability framework with domain-dependent knowledge. We have shown that it is easy to encode such knowledge in a purely declarative manner that is completely independent of the kind of algorithm used to search the space. The approach appears to be the key to the next order of magnitude scaling of state-space planning algorithms. Problems that could not be solved in days could be solved in seconds with the additional axiomatic knowledge.

Our framework makes clear the logical status of different kinds of heuristics, and in particular distinguishes invariants, optimality conditions, and simplifying assumptions. The role of the optimality conditions deserves further discussion. We observed that setting the length of the plan to the minimum value that keeps it consistent places a *global* optimality condition on the problem. This global optimality condition logically entails many *local* rules about the construction of the solution: for example, that the solution not contain such locally inefficient actions such as moving a block twice in a row. However, it can be quite hard to deduce all the relevant local optimality conditions for a particular problem. The optimality heuristics we have described simply make *explicit* some of these local rules. Thus, less needs to be deduced by combinatorial search. In other words, it can be easier to satisfy a global constraint with the "advice" given by a subset of the local *consequences* of that constraint.

Finally, we have seen that on the hardest problems we examined local search continues to outperform systematic search, just as in Kautz and Selman (1996). However, the use of heuristic axioms does make systematic methods more competitive. Figure 6 presents the overall best solution times for any version of each of the logistics problems for three search engines: Walksat, ntab, and satz, a new systematic search engine by Li and Anbulagan (1997). Satz is the first systematic engine we have found that can solve log.d, although it is 100,000 times slower than Walksat on that problem.

There are many avenues for future exploration. One task is to see exactly how the shape of the search space is changed by different kinds of heuristics. Frank, Cheeseman, and Stutz (to appear) present a detailed picture of the search space for random 3SAT. We would like to create such an analysis of the space generated by a range of planning domains under different sets of heuristics.

Another task is experiment with different search engines in our broadened planning as satisfiability model. We are beginning experiments with a new SAT engine combining features of stochastic and systematic search that is particularly good at solving planning problems. Third, we will be comparing the manually-created heuristic axioms used here with those that can be automatically generated, either by exact compilation (Williams and Nayak 1997) or approximate compilation

(Selman and Kautz 1996).

Finally, we are beginning to experiment with SATPLAN for richer planning models, such as ones involving causal plans (Kautz, McAllester, and Selman 1996), metric and qualitative temporal constraints, and soft constraints. The basic principle behind all our work is to view planning as *efficient combinatorial search.* We believe that many of the lessons we have learned about state-space planning apply to more general models.

Acknowledgements

We thank Jim Hendler, Martha Pollack, and Dan Weld for their valuable comments on an earlier draft of this paper.

References

Adorf, H.M. Johnston, M.D. (1990). A discrete stochastic neural network algorithm for constraint satisfaction problems. *Proc. of the Int. Joint Conf. on Neural Networks*, San Diego, CA, 1990.

Agre, P. and Chapman, D. (1987). Pengi: an implementation of a theory of activity. *Proc. AAAI-87*, Seattle, WA.

Bacchus, F. and Kabanza, F. (1995). Using temporal logic to control search in a forward-chaining planner. Technical report, University of Waterloo.

Blum, A. and Furst, M.L. (1995). Fast planning through planning graph analysis. *Proc. IJCAI-95*, Montreal, Canada.

Bonet, B., Loerincs, G., and Geffner, H. (1990). A robust and fast action selection mechanism. *Proc. AAAI-97*, Providence, RI, 714-719.

Carbonell, J. , Blythe J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M., Wang, X (1992). Prodigy 4.0: the manual and tutorial. CMU, CS Tech. Report CMU-CS-92-150.

Crawford, J.M. and Auton, L.D. (1993). Experimental results on the cross-over point in satisfiability problems. *Proc. AAAI-93*, Washington, DC, 21–27.

Ernst, M.D., Millstein, T.D., and Weld, D.S. (1997). Automatic SAT-compilation of planning problems. *Proc. IJCAI-97*, Nagoya, Japan, 1169–1176.

Erol, K., Nau, D.S., and Subrahmanian, V.S. (1992). On the complexity of domain-independent planning. *Proceedings AAAI92*, 381–386.

Etzioni, Oren (1993). Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62(2), 255–302.

Frank, J., Cheeseman, P., and Stutz, J. (to appear). When gravity fails: local search topology.

Gupta and Nau (1992). On the complexity of blocks-world planning. *Artificial Intelligence*, 56, 139–403.

Joslin, D. and Pollack, M. (1995). Passive and Active Decision Postponement in Plan Generation. In the *European Workshop on Planning (EWSP)*, Assisi, Italy, Sept. 1995.

Joslin D. and Roy, A. (1997). Exploiting symmetries in lifted CSPs. *Proc. AAAI-97*, Providence, RI.

Kambhampati, S., Katukam, S., and Qu, Y. (1996). Failure driven dynamic search control for partial order planners: an explanation based approach. *Artificial Intelligence* 88(1–2), 253-315.

Kautz, H., McAllester, D., and Selman, B. (1996). Encoding plans in propositional logic. *Proc. KR-96*, Cambridge, MA.

Kautz, H. and Selman, B. (1996). Pushing the envelope: planning, propositional logic, and stochastic search. *Proc. AAAI-1996*, Portand, OR, 1996.

Knoblock, C. (1994). Automatically generating abstractions for planning. *Artificial Intelligence* 68(2).

Li, Chu Min and Anbulagan (1997). Heuristics based on unit propagation for satisfiability problems. *Proc. IJCAI-97*, Kyoto, Japan, 1997.

McAllester, David (1991). Observations on cognitive judgements. *Proc. AAAI-91*, Anaheim, CA, 910-915.

Minton, S. (1988). Quantitative results concerning the utility of explanation-based learning. *Proc. AAAI-88*, St. Paul, MN, 564–569.

Muscettola, N. (1994). On the utility of bottleneck reasoning for scheduling. *Proc. AAAI-94*, Seattle, WA, 1105-1110.

Penberthy, J. and Weld, D. (1992). UCPOP: A sound, complete, partial order planner for ADL. In the *Proc. of the Third Int. Conf. on Knowledge Representation and Reasoning*, Boston, MA, 103–114.

Sacerdoti, E. D. (1977). A Structure for Plans and Behavior. Elsevier, NY.

Selman, B. and Kautz, H. (1996). Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193-224, 1996.

Selman, B., Kautz, H., and Cohen, B. (1994). Noise Strategies for Local Search. *Proc. AAAI-94*, Seattle, WA, 1994, 337–343.

Selman, B., Levesque, H.J., and Mitchell, D. (1992) A new method for solving hard satisfiability problems. *Proc. AAAI-92*, San Jose, CA (1992) 440–446.

Slaney, J. and Thiebaux, S. (1996). Linear time near-optimal planning in the blocks world. *Proc. AAAI-96*, Portland, OR, 1996.

Smith, D. (1989). Controlling backward inference. *Artificial Intelligence*, 39, 145–208.

Smith, D. and Parra, E. Transfornational Approach to Transportation Scheduling, *Proc. of the 8th Knowledge-Based Software Eng. Conf.*, 1993, Chicago, IL.

Smith, D. and Peot, M. (1996). Suspending recurison in causal link planning. *Proc. AIPS-96*, Edinburgh, UK, 1996.

Stone, P., Veloso, M., and Blythe, J. (1994). The need for different domain-independent heuristics. In *AIPS94*, pages 164-169, Chicago, 1994.

Veloso, M. (1992). Learning by analogical reasoning in general problem solving. Ph.D. Thesis, CMU, CS Techn. Report CMU-CS-92-174.

Vere, S. (1985). Temporal scope of assertions and window cutoff. *Proc. IJCAI-85*, Los Angeles, CA.

Williams, B. and Nayak, P. (1997). A reactive planner for a model-based executive. *Proc. IJCAI-97*, Kyoto, Japan.

Weld, D. (1994). An introduction to least commitment planning. *AI Magazine*, Summer/Fall 1994.

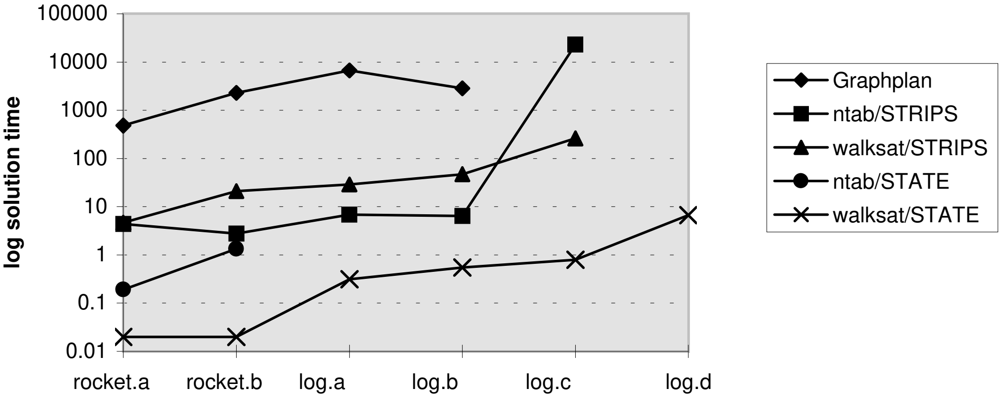

Fig. 1. Comparison of search times on original logistics problems.

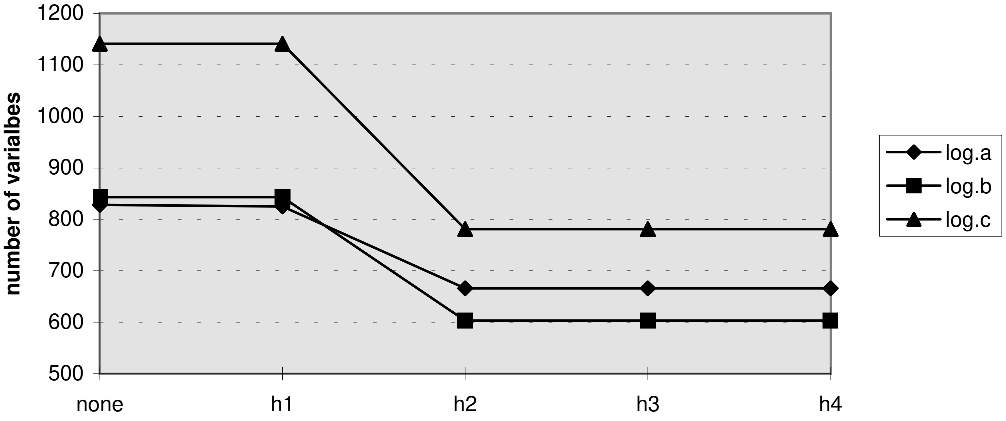

Fig 2. Number of variables in logistics formulas after simplification.

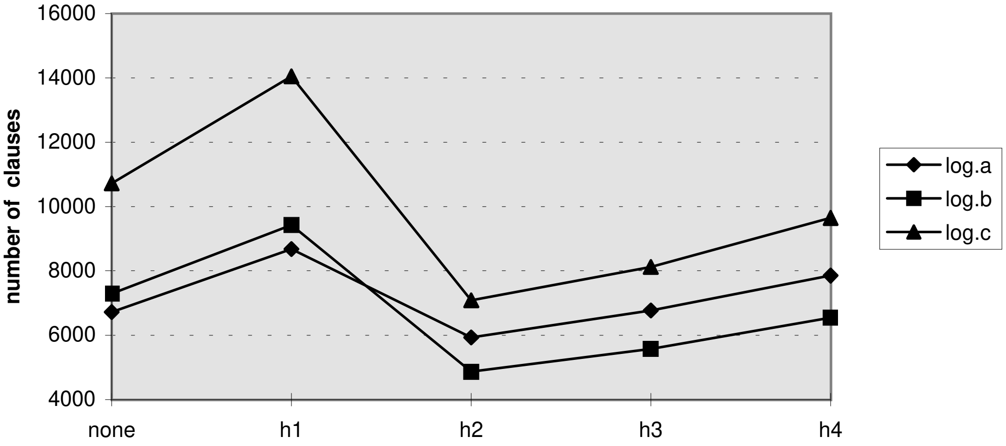

Fig. 3. Number of clauses in logistics formulas after simplification.

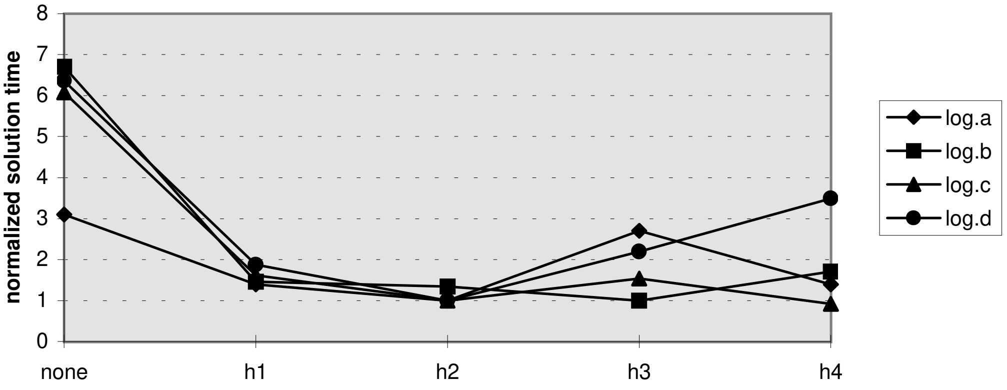

Fig. 4: Solution times for walksat on logistics with different sets of heuristics. Times are normalized as multiples of the minimum solution time for each problem.

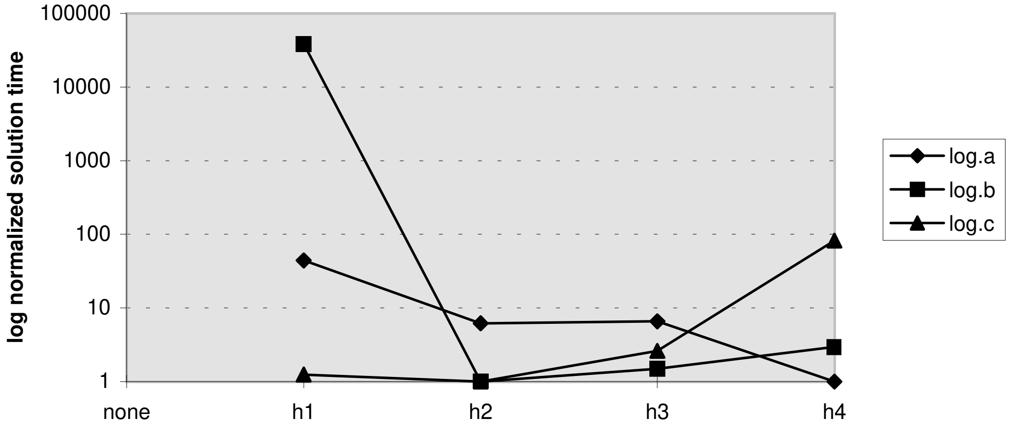

Fig. 5: Normalized solution time for ntab on logistics wffs. Times are normalized as multiples of the minimum solution time for each problem (a, b, c, d), and plotted on a log scale.

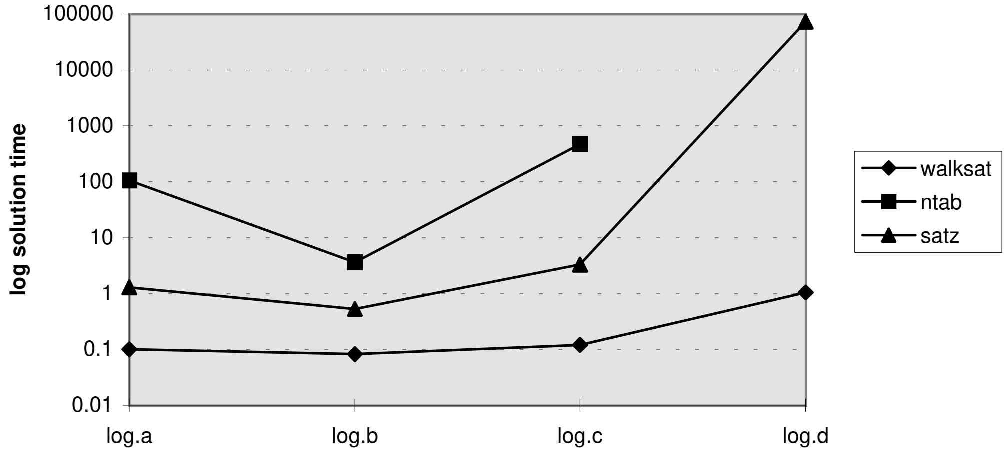

Fig 6: Best solution times logistics wffs, any solver.