Reachability, Relevance, Resolution and the Planning as Satisfiability Approach

Ronen I. Brafman
Dept. of Math and Computer Science
Ben-Gurion University
Beer-Sheva, Israel
brafman@cs.bgu.ac.il
http://www.cs.bgu.ac.il/ brafman

Abstract

We investigate the ability of two central encoding methods to propagate reachability and relevance information using resolution steps. More specifically, we compare the ability of unit-propagation and higher-order resolution steps to propagate reachability and relevance information in the context of the linear and GRAPHPLAN encoding schemes to the ability of a natural class of reachability and relevance algorithms that operate at the plan level. As a result of our observations and additional considerations, we experiment with a preprocessing step based on limited binary resolution that shows nice results.

1 Introduction

The success of the planning as satisfiability (PAS) approach [Kautz and Selman, 1992; 1996] has led to various attempts to refine the initial methods used and to improve our understanding of its performance. In particular, various methods for generating formulas from planning instances have been compared [Ernst *et al.*, 1997], and various systematic alternatives to the original stochastic method have been examined (e.g., [Li and Anbulagan, 1997; Bayardo and Schrag, 1997]). Still, many issues surrounding this approach are poorly understood. In particular, little is known about the influence of the encoding method on performance.

Concentrating on the two encoding methods proposed by [Kautz and Selman, 1996], the linear and the GRAPHPLAN-based encodings, we examine their influence on the ability to propagate reachability and relevance information via unit propagation and, more generally, k-clause resolution. We do so by comparing the pruning ability of these techniques to that of variants of existing algorithms that operate on the original problem formulation [Boutilier *et al.*, 1998].

Our work is motivated by unit resolution's central role in the Davis-Putnam algorithm [Davis *et al.*, 1962] and many of its offsprings (e.g., [Freeman, 1995; Crawford and Auton, 1993; Li and Anbulagan, 1997; Gomes *et al.*, 1998]), and its use as a preprocessing step when stochastic methods are applied. Moreover, a limited form of binary propagation is used in Crawford's COMPACT program for simplifying CNF formulas and is utilized in the BLACKBOX planner [Kautz and Selman, 1998]. Reachability and relevance analysis play a major role in recent planning algorithms, most notably in the GRAPHPLAN planner [Blum and Furst, 1995].

Finally, motivated by some of our observations and by the fact that binary clauses form a significant fraction of the clauses within sat-encoded planning problems, we show that a simple and cheap preprocessing step based on limited binary resolution can yield nice savings in running time.

The paper is organized as follows: in Section 2, we discuss the Reachable-k algorithm and its counterpart, Relevant-k. We compare each algorithm's ability to prune possible actions to that of k-clause resolution. A number of theoretical results are presented as well as an empirical comparison of the various methods for $k = 1$. In Section 3, we motivate and explain the use of binary-binary resolution as a preprocessing step and show its effect on two recent efficient systematic algorithms. The overhead of this preprocessing step is relatively small, and in some cases it yields nice savings. We conclude with a discussion of future and related work in Section 4. Throughout the paper we assume basic familiarity with the essential ideas behind the PAS framework and the GRAPHPLAN algorithm. Proofs appear in a longer version of this paper, although their central points are discussed here.

2 Reachability, Relevance, Resolution

Reachability and relevance analysis form an essential part of successful modern planning algorithms. The most notable example of reachability analysis is GRAPHPLAN's planning graph [Blum and Furst, 1995], and many recent planners employ either reachability analysis (e.g., [Bonet *et al.*, 1997]), relevance analysis (e.g., [McDermoot, 1996; Nebel *et al.*, 1997]), or both [Kambhampati *et al.*, 1997]. The importance of reachability and relevance analysis has been noted in the context of decision-theoretic planning as well. For example, [Boutilier and Dearden, 1994] employ relevance analysis to reduce the state-space, and [Boutilier *et al.*, 1998] describe a general method for reachability analysis for MDPs. Below, we discuss this method in a simplified form suitable for classical planning problems described using the STRIPS representation language [Fikes and Nilsson, 1971]. We shall also present a counterpart of this method for performing relevance analysis and relate these algorithms to k-clause resolution in the context of sat-encoded planning problems.

- S_0 = propositions that are true initially.
- $CS_0 = \{\}$.
- A_0 = actions all of whose preconditions are in S_0.
- $CA_0^L = \{\{a_i, a_j\} | a_i, a_j \in A_0,\ i \neq j,$ neither a_i nor a_j are noops or a_j is a noop whose effect is destroyed by $a_i\}$.
- $CA_0^P = \{\{a_i, a_j\} | a_i, a_j \in A_0,\ i \neq j,$ and a_i deletes a precondition or an effect of $a_j\}$.

We define S_i, A_i inductively as follows:

- S_i = literals that appear in the effects of A_{i-1}.
- A_i = actions whose preconditions appear in S_i and no subset of them appears in CS_i^*.
- CS_i^* = k-tuples of literal appearing in S_i such that some subset of any set of actions from A_{i-1} that has these literals as their effects appears in CA_{i-1}^* (where $* \in \{L, P\}$ as appropriate).
- $CA_0^L = \{\{a_l, a_j\} | a_l, a_j \in A_0, l \neq j,$ neither a_l nor a_j are noops, or a_j is a noop whose effect is destroyed by $a_l\}$.
- $CA_i^P = \{\{a_{j_1}, \ldots, a_{j_l}\} | l \leq k,\ a_{j_1}, \ldots, a_{j_l} \in A_i,\ j_m \neq j_n$ for $m \neq n$, and either (1) a_{j_m} deletes a precondition or an effect of a_{j_n} for some $m \neq n$ or (2) some subset of the preconditions of $a_{j_1}, \ldots, a_{j_l}$ appears in $CS_i^*\}$.

Figure 1: Reachable-k

2.1 Propagating Reachability Information

Reachable-k [Boutilier *et al.*, 1998] is an algorithm for estimating the states reachable from a given initial state. As formulated, it is quite general and applies to domains with nondeterministic actions and conditional effects. In Figure 1, we present a simplified version of Reachable-k that deals with deterministic, unconditional actions represented in the STRIPS representation language. An important reason for our interest in it is its similarity to the influential planning graph construction of the GRAPHPLAN planner [Blum and Furst, 1995]. In fact, it generalizes the ideas behind GRAPHPLAN's planning graph, which is equivalent to Reachable-2. We use A_i to denote the set of actions feasible i steps from the initial state, S_i to denote the corresponding set of propositions, and CS_i^* to denote constraints on these propositions, such that if $\{p_1, \ldots, p_m\} \in CS_i^*$ then these propositions cannot co-occur after i steps. CA_i^* denotes similar constraints on actions. Here, $* \in \{L, P\}$, where L is used when we restrict our attention to linear action sequences, and P is used when we allow concurrent non-conflicting actions (i.e., actions that do not destroy each others' effects or preconditions and whose preconditions are not constrained not to co-occur). Of course, for $k = 1$ the sets CS_i^* and CA_i^* are empty for all $i \geq 0$. Finally, note that in this description, the set of possible actions contains all actions of the form noop(l), where l is a literal.

When $k = 2$, S_i and A_i represent the propositional and action levels of GRAPHPLAN's planning graph, and CS_i and CA_i^P hold their respective mutual exclusion constraints. We have not stated a termination condition for this algorithm, but one can be formulated based on the content of S_i or the index i itself. In the PAS framework, where the number of time-steps is fixed, one would opt for the second alternative. Reachable-k gives us sets of actions and propositions, A_j, S_j, that can occur after the performance of j actions (or j sets of concurrently non-conflicting actions) from the initial state. It is sound in the following sense: if a set of propositions or a set of actions is excluded by it at time j, we know that these cannot occur (resp. be executed) after j steps.

Sometimes, all actions that can be executed at a particular time point in which p holds have $\neg p$ as an effect. In that case, we can ignore the noop(p) action, as it will not be part of any useful plan. However, as formulated, p will appear in Reachable-k's next level. We denote by Reachable*-k a variant in which noop(p) does not appear in such a case.

2.2 k-Clause Resolution and Reachability

k-clause resolution (or propagation) refers to the resolution of pairs of clauses one of whose length is k at most. The $k = 1$ variant, i.e., unit propagation, is an integral part of all major algorithms for generating satisfying assignments.

We wish to compare the type of reachability information derived by performing k-clause resolution on sat-encoded planning problems, with the information obtained by running the Reachable-k algorithm. By reachability information we mean constraints on the set of actions possible at a time point or constraints on world states (in the form of, e.g., sets of unreachable propositions or k-tuples of propositions). Our focus will be on the first type of constraints. We say that one algorithm generates *more* reachability information than another if it always generates a superset of the constraints on actions generated by the other algorithm, and there are instances in which this is a strict superset. We compare the two Reachable variants with two encoding methods discussed in [Kautz and Selman, 1996]:[1] the linear encoding and the more interesting GRAPHPLAN encoding.

Linear Plan Encoding

The linear plan encoding [Kautz and Selman, 1992] is a simple and natural method for translating a planning problem into a formula that is satisfiable iff there is a valid plan of length n (for some given n). The clauses in the linear plan encoding fall into the following classes:

1. an action implies its preconditions prior to its execution;
2. an action implies its effects following its execution;
3. an action does not affect any other proposition (frame axioms);
4. there is at least one action at each time point;
5. there is at most one action at each time point.

Here, noop actions are not considered. In addition, the formula contains unary clauses describing the initial and goal states. In analyzing reachability effects, we *ignore* information about the goal state (discussed later in the section on relevance).

Consider the mechanism by which resolution can yield reachability information: Given the propositions that hold at the initial state, we can derive the negation of actions whose preconditions do not hold using unit propagation on axioms of class 1. Propagating these unit clauses with the appropriate

[1] The third (state-based) encoding method cannot be generated automatically.

	$\|A\|$	Reach	Rel	R+R	U-rch(l)	U-rel(l)
log.a	4565	2922	617	3476	401	38
log.b	5941	3517	680	3905	442	20
log.c	8021	5051	2782	6214	600	32
bw.a	3888	1697	408	2105	639	300
bw.b	10890	3565	830	4395	1201	440
bw.c	44100	12818	2394	15212	3141	840
bw.d	116964	26963	5238	32201	6482	5114

Table 1: **Pruning Effects of Unary Methods.** $|A|$ is the number of possible actions. Following entries hold number of actions pruned using: Reachable-1, Relevant-1, both combined, unit propagation on linear encoding using initial state, and using the final state. Unit propagation in the GRAPHPLAN encoding using the final state yielded no pruning. Execution times for the Reach/Relevant algorithms are ≤ 0.01 seconds except for bw.c (0.03 sec.), and bw.d (0.07 sec.).

instance of axiom class 4, we will obtain a disjunction of all actions that can be executed at the first time point. So far, this is identical to what Reachable*-k provides. To propagate this information forward, we can resolve these action disjunctions with axioms of class 2 and 3. This, however, requires binary resolution. Hence, except for the unlikely case in which a single action is possible, there is no more that we can derive using unit propagation alone. Reachable*-1, on the other hand, can provide us with a list of all possible effects of these actions and possibly prune out future actions whose preconditions do not appear in this list. We conclude:

Lemma 1 *In the context of the linear encoding, Reachable*-1 yields more reachability information than unit propagation.*

Example: Consider a blocks' world domain with a single action schema MOVE(object,source,destination). Its preconditions are: ON(object,source), CLEAR(object), clear(destination) and its effects are: ON(object,destination), CLEAR(source), ¬ON(object,source), ¬CLEAR(destination) (except when the destination is the table which is always clear). If we have k stacks of blocks initially, k^2 actions can be performed at the initial state (i.e., moving a block from the top of a stack to the top of another stack or the table). This will be discovered by both algorithms. In particular, unit propagation will yield a disjunction of all these actions. We know that all blocks that are 2 or more blocks below the top cannot participate in the second MOVE action. Reachable-1 will find this out due to the fact that they are not CLEAR. Suppose that A is one such block. All initially feasible actions participate in a frame axiom of the form $Move\text{-}x \wedge \neg clear(A,0) \rightarrow \neg clear(A,1)$, which, in clausal form is $\neg Move\text{-}x \vee clear(A,0) \vee \neg clear(A,1)$. Resolving against $\neg clear(A,0)$, we have $\neg Move\text{-}x \vee \neg clear(A,1)$. If we could deduce $\neg clear(A,1)$, we could rule out all actions that have it as a precondition. But if we are restricted to unit propagation, this requires deducing $Move\text{-}x$ for some initially feasible action, and we cannot make such a deduction. Precise numbers for a number of instances appear in Table 1.

If we propagated information forward using axioms of class 2 and 3 and binary resolution (i.e., as discussed before Lemma 1), we now have a set of disjunctions of the possible effects (including frame effects) of the initially allowable actions. The number of such disjuncts is $O(e^m)$, where e is the maximal number of effects of an action and m is the number of actions that can be executed initially. In some cases, these disjunctions could contain a single literal, e.g., when all initially allowable actions leave some proposition unchanged. When one of these disjunctions contains only literals that are negations of some action's precondition, we can deduce the negation of this action by resolving with axioms of class 1.

In the example considered above we would generate a disjunction of the form $Move\text{-}x_1 \vee Move\text{-}x_2 \vee \cdots \vee Move\text{-}x_r$, containing all time 0 actions whose negations have not been deduced. As discussed above, for all such actions, we can obtain a clause of the form $\neg Move\text{-}x_i \vee \neg clear(A,1)$. Resolving these binary clauses against the clause above, we obtain a unary clause $\neg clear(A,1)$, that can be used in conjunction with class 1 axioms to deduce the negations of actions whose preconditions include $clear(A,1)$.

Using the effect disjunctions we deduce mutual exclusion constraints on actions, but these are already built into the encoding. In principle, one can resolve these effect disjunctions with each other, but any useful new resolvants already appear among them (simply based on the fact that an action cannot have a proposition and its negation as effects). Consequently, we have:

Lemma 2 *In the linear encoding, binary resolution is sufficient to conclude all possible reachability constraints.*

Reachable*-k propagates information forward in a similar manner but does not consider interactions between more than k actions or propositions. Since such interactions can occur, we have:

Consequence 1 *Reachable*-k yields less information than binary resolution.*

The GRAPHPLAN Encoding

The GRAPHPLAN encoding differs from the linear encoding by its ability to consider multiple concurrent (non-interfering) actions, allowing one to obtain shorter plans which, in turn, can reduce the search space size. It constructs the following sets of clauses:

1. An action implies its preconditions;
2. An effect implies one of the actions that has this effect;
3. There is at least one action at each time-point;
4. Two conflicting actions cannot occur together.

Besides the obvious ability to consider multiple parallel (non-interfering) actions, the important difference between the GRAPHPLAN and Linear encoding is in axiom class 2 (referred to in [Ernst *et al.*, 1997] as *explanatory* frame axioms.) Clauses in this class will contain positive occurrences of action literals and negative occurrences of state literals.

As in the linear case, using unit propagation we can infer which actions cannot be applied at the initial state. Using axioms of class 2, we can propagate this information forward, deducing the negation of all effects that cannot be produced by the initially allowable actions. This information enables us to exclude actions whose preconditions cannot be produced. This forward propagation is essentially identical to

Reachable-1. If we ignore the explicit constraints appearing in axiom class 4, we can conclude:

Lemma 3 *In the context of the* GRAPHPLAN *encoding, unit propagation and Reachable-1 yield the same reachability information.*

Notice that axioms from class 4 do not participate in unit propagation because they contain pairs of negated actions. Resolving against them requires a positive action literal which cannot be deduced using the given axioms (except for the unlikely case in which a single action is possible initially). In general, we can deduce only negated action literals, which can be resolved against axioms of class 2 to yield, at best, deduced effects, or against axioms of class 3 to yield a disjunction of possible actions. This is precisely what Reachable-1 yields.

When $k > 1$, the mechanism remains the same, but now axioms of class 4 can play a part if we apply binary propagation. First, we will have binary constraints on co-occurring actions. These will propagate forward, possibly resulting in constraints on state propositions. These constraints need not necessarily be binary. Their propagation will require, in the general case, more than binary resolution. More generally, we observe that:

Lemma 4 *In the context of the* GRAPHPLAN *encoding, for $k > 1$, k-clause resolution yields more information than Reachable-k.*

As a special case, when $k = 2$ it has been observed [Kautz and Selman, 1998] that GRAPHPLAN's propagation of mutexes (which is equivalent to Reachable-2) is equivalent to a restricted form of binary resolution.

In general, one difference between these methods lies in the ability of k-clause resolution to yield disjunctions of more than k state propositions or actions, although these constraints cannot be propagated forward unless k is larger than the maximal size of clauses of class 2. As an example of how binary resolution can yield more reachability information than GRAPHPLAN's planning graph consider a situation where p can be produced only by actions a_1, a_2; q can be produced only by a_1, a_3; and r can be produced only by a_2, a_3; but each pair of a_1, a_2, a_3 is mutually exclusive. If a_1, a_2, a_3 are the only possible actions, then we can derive the ternary constraint $\neg p \vee \neg q \vee \neg r$.

2.3 k-Clause Resolution and Relevance

We formulate an algorithm similar to Reachable-k, which we call Relevant-k, with a similar soundness property. Relevant-k prunes the search space by generating a set of propositions that could appear in states that precede the goal state by k steps in any execution of a valid plan. Actions whose preconditions are not among these propositions can be ruled out. This leads to a reduced search space. Relevant-k is described in Figure 2.[2] For $k = 1$ we ignore the sets CA_i and CS_i. Various existing algorithms use ideas similar to Relevant-1 (e.g., [McDermoot, 1996; Nebel *et al.*, 1997]).

For Relevant-k to work in practice we must make the following closure assumption: if a proposition appears in the effect of an action (possibly negated), it must also appear in its precondition (possibly negated). When propositions stand for path properties (e.g., see the TSP domain in the GRAPHPLAN distribution), one cannot enforce this condition.[3]

- S_0 = propositions that are part of the goal.
- $CS_0 = \{\}$.
- A_0 = actions, some of whose effects are in S_0.
- $CA_0 = \{\{a_i, a_j\} | a_i, a_j \in A_0,\ i \neq j,$ and a_i deletes a precondition or an effect of $a_j\}$.

We define S_i, A_i inductively as follows:

- S_i = propositions that appear in the preconditions of A_{i-1}.
- A_i = actions whose effects appear in S_i and no subset of them appears in CS_i.
- CS_i = k-tuples of propositions appearing in S_i such that some subset of any set of actions from A_{i-1} that has these propositions as their preconditions appears in CA_{i-1}.
- $CA_i = \{\{a_{j_1}, \ldots, a_{j_l}\} | l \leq k,\ a_{j_1}, \ldots, a_{j_l} \in A_i,\ j_m \neq j_n$ for $m \neq n$ and either (1) a_{j_m} deletes a precondition or an effect of a_{j_n} for some $m \neq n$ or (2) some subset of the effects of $a_{j_1}, \ldots, a_{j_l}$ appears in $CS_i\}$.

Figure 2: Relevant-k

We now compare the amount of relevance information that can be propagated backwards using k-clause resolution and the goal literals as opposed to Relevant-k. Consider unit propagation first. In the context of the linear encoding, we see that all actions that destroy some goal condition will be ruled out. However, actions that are irrelevant because they produce irrelevant effects will not be pruned. This is incomparable to Relevant-1. There, irrelevant actions will be pruned out, but a relevant action that destroys some goal proposition will not be ruled out without modification to the algorithm.

In the context of the GRAPHPLAN encoding the relationship is clearer. From the goal propositions and axioms of class 2 we can deduce disjunctions of actions that must have produced these effects. Typically, these disjunctions will not contain unit clauses, and unit propagation cannot proceed farther. Notice that we cannot deduce the kind of information obtained via the linear encoding. That is, if an action destroys some goal proposition, we cannot conclude its negation using unit propagation. For example, if a has $\neg p$ as an effect and p is part of the goal, we have $\neg p \rightarrow a \vee \cdots$ as an instance of axiom class 2. Since a appears positive in this axiom, we cannot deduce its negation by resolving against it. Rather, deducing negated actions requires explicit effect axioms of the form $p \rightarrow \neg a$. Finally, while such information is not deduced necessarily by Relevant-1, the simple relevance information deduced by Relevant-1 cannot be deduced here either. That is, we have no way of deducing $\neg a$ if all of a's effects are irrelevant to the goal. We conclude (again, ignoring the explicit mutual exclusion information contained in axiom class 4):

Lemma 5 *In the context of the* GRAPHPLAN *encoding, unit propagation yields less relevance information than Relevant-1.*

[2] We consider the parallel execution case only.

[3] With small modification, this assumption can be removed.

Some actual values appears in Table 1. In particular, in the examples we looked at, the GRAPHPLAN encoding could not prune any action. This follows from the (quite typical) fact that in these domains, each of the facts that hold at the final state can be achieved by a number of actions. Hence, unit propagation can deduce only disjunctions of possible actions, none of which are a unit clause. Since we have no way of deducing negated actions, propagation stops at this point.

The general case is similar. In the linear encoding, having obtained a disjunction of allowable actions, we can generate a disjunction of allowable preconditions. This information is propagated backwards much like the forward case. Yet, as in the $k = 1$ case, all we can expect is a form of backwards reachability analysis from the goal state, rather than true relevance analysis. In the context of the GRAPHPLAN encoding, we will generate disjunctions of relevant actions, from which disjunctions of relevant preconditions can be deduced, etc. However, irrelevant actions will not be excluded explicitly (since more than one action is allowed at each step) and we will only conclude that some relevant action must appear. Nor can we exclude actions that destroy a goal proposition. On the other hand, using k-clause resolution we can deduce constraints of order greater than k, unlike Relevant-k. Therefore, no clear winner emerges. We hypothesize that Relevant-k would perform better, but this remains to be tested.

Finally, we note that (1) the GRAPHPLAN planner does not incorporate relevance analysis, but Mea-GRAPHPLAN, a more recent variant, does [Kambhampati *et al.*, 1997]. (2) [Ernst *et al.*, 1997] discuss an enhanced version of the GRAPHPLAN encoding which contains effects axioms as well (i.e., axioms of the form action $\rightarrow$ effect). In terms of the ability to propagate reachability and relevance analysis we see here only an added ability to rule out actions that destroy needed propositions (as in the linear encoding.)

3 Binary Resolution Preprocessing

Specialized subroutines that exploit binary clauses in SAT problems have been considered in the past [Larrabee, 1992]. We believe that judicious use of binary resolution is a promising direction in the context of the PAS framework for a number of reasons: (1) GRAPHPLAN's mutexes are equivalent to binary clauses, and their propagation is equivalent to a limited form of binary resolution [Kautz and Selman, 1998]. (2) Binary resolution yields all the reachability information in the linear encoding (Lemma 2). (3) Binary clauses form a large fraction of the clauses within encoded formulas: In the linear encoding, all axioms of class 1,2,5 yield binary clauses, and in the GRAPHPLAN encoding, this is true of axioms of class 1,4. (4) 2-SAT problems can be solved in polynomial time.

Unlike unit propagation, binary propagation increases the size of the formula (although it yields clauses that are no larger than the clauses resolved). This increase can slow down the solution process considerably and the increased memory consumption can lead to thrashing. Consequently, one must either restrict the extent of binary propagation or devise fast, efficient methods for performing them.

We experimented with a simple preprocessing step which resolves pairs of binary clauses until no new clauses are derived. This method can be used by systematic and stochastic methods as a simplification step, and it can be implemented efficiently. It is not always useful, as sometimes no or few unit clauses are deduced. Yet, the overhead it incurs is relatively low, especially when we consider the more complex examples, and it seems to be a useful enhancement. In Table 2, we see a comparison of the running time of Bayardo and Schrag's REL-SAT algorithm with and without the preprocessing step. We also show the time required for binary-binary resolution and the number of unit clauses derived.[4]

	RELSAT		RELSAT + bin/bin res.			
Problem	mean	std	mean	std	res.	unit
log-dir.a	1.69	1.41	1.1	0.97	0.01	72
log-dir.b	5.83	11.3	0.97	0.67	0.01	60
log-dir.c	105.70	231.34	3.11	6.06	0.01	84
log-gp.a	1.27	0.54	1.42	0.49	0.27	0
log-gp.b	4.59	2.68	4.63	2.44	0.45	0
log-gp.c	14.63	17.08	15.65	17.21	0.81	0
log-un.a	0.34	0.08	0.53	0.09	0.18	12
log-un.b	29.11	4.10	27.01	3.77	0.31	0
log-un.c	54.17	7.8	48.77	6.73	0.57	0
bw-dir.a	0.08	0.01	0.12	0.01	0.07	92
bw-dir.b	0.76	0.42	0.71	0.19	0.15	146
bw-dir.c	35.58	15.62	28.84	9.43	0.54	272
bw-dir.d	513.5	444.4	428.6	335.7	4.4	438

Table 2: **Effect of Binary Clause Preprocessing on** REL-SAT. Avg. over 100 runs on an AMD-K6 200MHz processor running Linux. Running times for bin/bin include bin/bin resolution times (see *res.* column). For the number of unit clauses generated see *unit* column. Results for bw-dir.d are on a SUN UltraEnterprise 4000 running Solaris 2.5.1.

We also experimented with the performance of the SATZ algorithm [Li and Anbulagan, 1997] on the above instances with and without bin/bin resolution. In Table 3 we give the running times for SATZ as applied to the original and the simplified formula for those instances in which they differ. In addition, we conducted a number of experiments in which we attempted to resolve binary clauses with clauses of arbitrary size. We found the overhead of this method too large.

4 Conclusion

We have shown a connection between the scheme used to encode planning instances and the ability to propagate reachability and relevance information from the initial and final steps to other time points. We compared this ability to that of the Reachable-k and Relevant-k algorithms, the first being a generalization of GRAPHPLAN's planning graph, and the second being a natural extension into relevance analysis. We also pointed out the fact that binary clauses form a major part among all clauses in sat-encoded planning problems, and we attempted to exploit this phenomena. Our initial experiments show nice improvements in instances where unit clauses can be derived from binary resolution, and a small overhead otherwise. We are currently experimenting with various extensions

[4] See ftp://ftp.research.att.com/dist/ai/logistics.tar.Z and satplan.data.tar.Z for the instances used.

Problem	SATZ	SATZ with BBR	bin. res.	unit cl.
log-dir.a	119.32	0.37	0.01	72
log-dir.b	0.52	0.35	0.01	60
log-dir.c	3.97	0.58	0.01	84
log-un.a	0.73	0.67	0.18	12
bw-dir.a	0.28	0.16	0.07	92
bw-dir.b	1.01	0.75	0.15	146
bw-dir.c	4.71	5.25	0.54	272
bw-dir.d	1716.93	15.81	3.19	438

Table 3: **Effect of Binary Clause Preprocessing on the SATZ Algorithm.** Experiments conducted on an AMD-K6 200MHz processor running Linux. Times for SATZ with BBR do *not* include binary resolution preprocessing step. Only instances in which binary resolution yielded some unit clauses were examined.

of the Davis-Putnam procedure that perform limited amounts of binary resolution during the search process.

This work is among the first attempts to theoretically analyze different encoding schemes. We have concentrated on one particular aspect of such encodings, i.e., their ability to propagate concrete state information backwards and forwards. Naturally, this attempt is a-priori limited in its scope, as this ability is only one factor influencing the performance of various algorithms, and its influence is probably more significant in systematic methods based on the David-Putnam procedure than in methods based on stochastic local search.

Other authors have considered some of the ideas presented here, too. Kautz and Selman (1996) point to the ability to backward propagate information thanks to the encoding of effects in the GRAPHPLAN encoding. Kautz and Selman (1998) mention a relation between GRAPHPLAN's mutex constraints and binary propagation. Recent work on this planner employs Crawfords's COMPACT algorithm which uses a restricted form of binary propagation to reduce the size of the formula. Finally, [Ernst *et al.*, 1997] discuss optimizations performed on sat-encoded planning problems, among them the use of type inference and a form of dataflow analysis that seems related to Reachable-1.

Acknowledgments

I wish to thank Craig Boutilier and Chris Geib for valuable discussions on reachability analysis and the anonymous reviewers for their useful comments. This work was partially funded by the Paul Ivanier Center for Robotics Research and Production Management.

References

[Bayardo and Schrag, 1997] R. J. Bayardo and R. C. Schrag. Using csp look-back techniques to solve real-world sat instances. In *Proc. AAAI-97*, pages 203–208, 1997.

[Blum and Furst, 1995] A. Blum and M. L. Furst. Fast planning through planning graph analysis. In *Proc. Fourteenth International Joint Conference on AI*, 1995.

[Bonet *et al.*, 1997] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proc. AAAI-97*, pages 714–719, 1997.

[Boutilier and Dearden, 1994] C. Boutilier and R. Dearden. Using abstractions for decision theoretic planning with time constraints. In *Proc. of AAAI'94*, 1994.

[Boutilier *et al.*, 1998] C. Boutilier, R. I. Brafman, and C. Geib. Structured reachability analysis for markov decision processes. In *Proc. UAI'98*, 1998.

[Crawford and Auton, 1993] J. Crawford and L. D. Auton. Experimental results on the cross-over point in satisfiability problems. In *Proc. AAAI'93*, 1993.

[Davis *et al.*, 1962] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communication of the ACM*, 5(7):394–397, July 1962.

[Ernst *et al.*, 1997] M. D. Ernst, T. D. Millstein, , and D. S. Weld. Automatic SAT-compilation of planning problems. In *Proc. IJCAI'97*, 1997.

[Fikes and Nilsson, 1971] R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Art. Int.*, 2(3–4):189–208, 1971.

[Freeman, 1995] J. W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, U. Pennsylvania CIS Dept., 1995.

[Gomes *et al.*, 1998] C. P. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *Proc. of 15th Nat. Conf. AI*, pages 431–437, 1998.

[Kambhampati *et al.*, 1997] S. Kambhampati, E. Parker, and E. Lambrecht. Understanding and exending graphplan. In *Proc. 4th European Conf. on Planning*, 1997.

[Kautz and Selman, 1992] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. of the 10th European Conf. on AI*, pages 359–363, 1992.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. of the 13th National Conference on AI (AAAI'96)*, pages 1194–1201, 1996.

[Kautz and Selman, 1998] H. Kautz and B. Selman. Blackbox: A new approach to the application of theorem proving to problem solving. In *Working notes of the Workshop on Planning as Combinatorial Search*, 1998.

[Larrabee, 1992] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, pages 4–15, January 1992.

[Li and Anbulagan, 1997] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proc. IJCAI-97*, 1997.

[McDermoot, 1996] D. McDermoot. A heuristic estimator for means-ends analysis in planning. In *Proc. 3rd Int. Conf on AI Planning Systems*, pages 142–149, 1996.

[Nebel *et al.*, 1997] B. Nebel, Y. Dimopoulos, and J. Koehler. Ignoring irrelevant facts and operators in plan generation. In *Proc. 4th Euro. Conf. on Planning*, 1997.