Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence

Planning with SAT, Admissible Heuristics and A*

Jussi Rintanen
The Australian National University, Canberra, Australia

Abstract

We study the relationship between optimal planning algorithms, in the form of (iterative deepening) A* with (forward) state-space search, and the reduction of the problem to SAT. Our results establish a strict dominance relation between the two approaches: any iterative deepening A* search can be efficiently simulated in the SAT framework, assuming that the heuristic has been encoded in the SAT problem, but the opposite is not possible as A* and IDA* searches sometimes take exponentially longer.

1 Introduction

We investigate the relations between two main approaches to finding optimal plans: state-space search with heuristic search algorithms such as A*, and the reduction to the propositional satisfiability problem SAT. Our optimality criterion is the minimality of the number of actions in a plan.

The theoretical limitations of the leading approaches to the optimal planning problem are well understood. A* [Hart *et al.*, 1968] is the best-known optimal search algorithm, which is guaranteed to expand the smallest possible number of states of any algorithm that does search state-by-state. The performance of A* is determined by the heuristic it uses. With a perfect heuristic it expands only the states corresponding to one optimal action sequence from the initial state to a goal state. With less informed heuristics the memory consumption is typically much higher. For example, with the trivial heuristic $h(s) = 0$ it will expand all states with a distance $< k - 1$ from the initial state when k is the length of the shortest path to a goal state. For short (polynomially long) plans A* still requires an exponential amount of memory in the worst case.

Planning by reduction to SAT, as proposed by Kautz and Selman [1992], has slightly different computational limitations. The sizes of the required formulas are linearly proportional to the length of action sequences considered. These action sequences may have an exponential length with respect to the representation of the problem instance, and, similarly to A*, will in this case require an exponential amount of memory. However, for short plans (polynomial in the size of the representation), even when A* expands an exponential number of states, the corresponding SAT problem can be solved in polynomial space (simply because SAT$\in$NP and NP$\subseteq$PSPACE.) This optimal space bound is achieved by well-known algorithms for the SAT problem, including the Davis-Putnam-Logemann-Loveland procedure [Davis *et al.*, 1962]. A counterpart of SAT in the state-space search domain is the iterative deepening A* (IDA*) algorithm [Korf, 1985] which stores in the memory, at any given time, only one bounded-length path in the state-space, and not all visited states like A* does. Under mild assumptions, IDA* does not perform many more search steps than A* [Korf, 1985, Theorem 6.4]. Although the worst-case resource requirements of SAT and IDA* are the same, the results of our work show that there is a simple implementation of SAT-based planning that is guaranteed to never do worse than state-space search with IDA*, and will sometimes perform exponentially better.

Our first result shows that if a given heuristic is encoded in the SAT problem, the Davis-Putnam-Logemann-Loveland procedure [Davis *et al.*, 1962] can simulate state-space search with IDA* [Korf, 1985], the iterative deepening variant of A* [Hart *et al.*, 1968], within the same time and space bounds. Our second result shows that the opposite does not hold: we construct a planning problem that requires an exponential amount of computation by A* and IDA*, but only a short resolution proof which is constructed by the unit propagation procedure in linear time.

The structure of the paper is as follows. In Section 2 we present the background of the work in SAT. Section 3 relates the notion of admissibility to the SAT framework and shows that one of the best known admissible heuristics is implicit in it. Section 4 presents a simulation of IDA* with the Davis-Putnam-Logemann-Loveland procedure for SAT. In Section 5 we show that both IDA* and A* are sometimes exponentially worse than SAT. We conclude by discussing related work in Section 6 and future research topics in Section 7.

2 Technical Background

Let Σ be a set of propositional variables. *Formulas* (over Σ) can be recursively constructed as follows. Every $a \in \Sigma$ is a formula. If ϕ is a formula, then so is $\neg\phi$. If ϕ_1 and ϕ_2 are formulas, then so are $\phi_1 \vee \phi_2$ and $\phi_1 \wedge \phi_2$. We also use $\top$ for the constant *true* and $\bot$ for *false*. For propositional variables $x \in \Sigma$, x and $\neg x$ are *literals*. The complement $\overline{l}$ of a literal l is defined by $\overline{x} = \neg x$ and $\overline{\neg x} = x$ when $x \in \Sigma$. A finite disjunction of literals is a *clause*.

A *valuation* of Σ is a (total) function $v : \Sigma \rightarrow \{0, 1\}$. A formula ϕ is *satisfied* by v (written $v \models \phi$) if the following holds. If $\phi = x$ for some $x \in \Sigma$, then $v \models \phi$ iff $v(x) = 1$. If $\phi = \neg\phi_1$, then $v \models \phi$ iff $v\neg \models \phi_1$. If $\phi = \phi_1 \wedge \phi_2$, then $v \models \phi$ iff $v \models \phi_1$ and $v \models \phi_1$. If $\phi = \phi_1 \vee \phi_2$, then $v \models \phi$ iff $v \models \phi_1$ or $v \models \phi_2$. The logical consequence relation $\phi \models \phi'$ holds if $v \models \phi$ implies $v \models \phi'$ for all valuations v. We use these notations also for sets of clauses. A formula ϕ (over Σ) is *satisfiable* if and only if $v \models \phi$ for some valuation v of Σ.

2.1 Planning

Planning problems are expressed in terms of a set F of facts (Boolean state variables) and a set of actions. Each action has a precondition $C \subseteq F$, consisting of the facts that have to be true for the action to be possible, and a set P of facts that become true (the *add* effects) and a set N of facts that become false (the *delete* effects). We define *prec*$(\langle C, P, N\rangle) = C$ and *eff*$(\langle C, P, N\rangle) = P \cup \{\neg f | f \in N\}$. An action is possible in state s (a valuation of F) if $s \models C$. The successor state $s' = \text{succ}_a(s)$ of s with respect to $a = \langle C, P, N\rangle$ satisfies the following: $s' \models l$ for all $l \in$ *eff*(a) and $s'(f) = s(f)$ for all $f \in F\backslash(P \cup N)$.

A problem instance is $\langle F, I, A, G\rangle$, where F is a set of facts, I is a state, A is a set of actions, and G is a set of literals. The objective is to find a shortest sequence of actions such that $\text{succ}_{a_n}(\text{succ}_{a_{n-1}}(\text{succ}_{a_2}(\text{succ}_{a_1}(I)))) \models G$.

2.2 Planning in the Propositional Logic

Planning was first represented in the propositional logic by Kautz and Selman [1992]. The idea is to look at the bounded length planning problem with time points $0, \ldots, T$. A propositional formula encodes the possible plans and their executions by describing how the state can change between two consecutive time points.

In this paper, we use a simple and efficient encoding based on explanatory frame axioms. The propositional variables we use are $f@t$ for facts $f \in F$ and $t \in \{0, \ldots, T\}$, $a@t$ for actions $a \in A$ and $t \in \{0, \ldots, T-1\}$.

We translate the action $a = \langle C, P, N\rangle$ into the following propositional clauses for all $t \in \{0, \ldots, T-1\}$.

$$\neg a@t \vee f@t \text{ for all } f \in C \quad (1)$$

$$\neg a@t \vee f@(t+1) \text{ for all } f \in P \quad (2)$$

$$\neg a@t \vee \neg f@(t+1) \text{ for } f \in N \quad (3)$$

For every fact f and time $t \in \{0, \ldots, T-1\}$ we have *frame axioms* that state when facts remain unchanged.

$$f@t \vee \neg f@(t+1) \vee a_{k_1}@t \vee \cdots \vee a_{k_m}@t \quad (4)$$

$$\neg f@t \vee f@(t+1) \vee a_{n_1}@t \vee \cdots \vee a_{n_s}@t \quad (5)$$

Here $a_{k_1}, \ldots, a_{k_m}$ are all the actions that change f from false to true. and $a_{n_1}, \ldots, a_{n_s}$ are all the actions that change f from true to false.

Additionally, at most one action can take place at a time, which is expressed by the clauses

$$\neg a@t \vee \neg a'@t \text{ for all } a, a' \in A, t \in \{0, \ldots, T-1\}, \quad (6)$$

and at least one action must take place, expressed by

$$\bigvee_{a \in A} a@t \text{ for every } t \in \{0, \ldots, T-1\}. \quad (7)$$

For given sets of facts and actions and an integer T, we denote the set of all the above clauses by H_T. The action sequences can be restricted to *plans* that reach some goals from some initial state by using H_T with additional clauses.

For a given state $s : F \rightarrow \{0, 1\}$, we define a clause set representing s by *lits*$(s) = \{f \in F | s(f) = 1\} \cup \{\neg f | f \in F, s(f) = 0\}$. For a clause set S over F, we define the set $S@t$ in which every propositional variable $f \in F$ has been replaced by its time-tagged variant $f@t$.

Let I be the initial state and G a set of literals that describes the goal states. The optimal planning problem can be defined as finding a T so that *lits*$(I)@0 \cup H_T \cup G@T$ is satisfiable and *lits*$(I)@0 \cup H_{T-1} \cup G@(T-1)$ is unsatisfiable. An optimal plan can be easily constructed from a valuation that satisfies the first clause set.

2.3 Resolution Proof Systems

The resolution rule defines one of the simplest proof systems for the propositional logic. It is used for showing that a set of clauses is unsatisfiable.

Definition 1 (Resolution) *Let $c_1 = l_1 \vee \cdots \vee l_n$ and $c_2 = \overline{l_1} \vee l'_2 \vee \cdots \vee l'_m$ be two clauses. The resolution rule allows deriving the clause $c_3 = l_2 \vee \cdots \vee l_n \vee l'_2 \vee \cdots \vee l'_m$. We use the notation $c_1, c_2 \vdash c_3$ for this. As a special case, resolving the* unit clauses *l_1 and $\overline{l_1}$ results in the empty clause.*

Definition 2 (Derivation) *Let $S = \{c_1, \ldots, c_n\}$ be a set of clauses. A* resolution derivation *of the clause c from S is any clause sequence $c'_1, \ldots, c'_m$ with the following properties.*

1. *For every $i \in \{1, \ldots, m\}$, either $c'_i \in S$ or $c'_j, c'_k \vdash c'_i$ for some $1 \leq j < k < i$.*
2. *$c'_m = c$.*

Definition 3 (Refutation) *Let $S = \{c_1, \ldots, c_n\}$ be a set of clauses. A* resolution refutation *of S (which shows S to be unsatisfiable) is any resolution derivation of the empty clause from S.*

A simple special case of the resolution rule is *unit resolution*, which infers ϕ from a unit clause l and another clause $\overline{l} \vee \phi$, where ϕ is a disjunction of literals. This rule leads to a very simple and efficient, but incomplete, inference procedure, which performs all unit resolution steps with a given clause set. This procedure can be implemented in linear time in the size of the clause set, and because there is (only) a linear number of possible unit resolution steps (each clause need to be involved at most once), it can be performed exhaustively. The fact that the clause c can be derived from S by unit resolution is denoted by $S \vdash_{UP} c$. The unit propagation procedure is a component of virtually all systematic SAT algorithms, including the Davis-Putnam procedure [Davis *et al.*, 1962] or the conflict-driven clause learning (CDCL) procedure [Mitchell, 2005].

3 Admissible Heuristics and Planning as SAT

Lower bound functions (heuristics) in algorithms such as A* are used for search space pruning. We denote a given lower bound for the distance from a state s to a goal g by $h^g(s)$.

Any admissible heuristic $h^g(s)$ is implicit in a logic formalization of planning, because admissibility means that the information given by the heuristic is a logical consequence of the formula that represent the planning problem: $h^g(s)$ is a *true* lower bound for the distance for reaching the goals from state s, not just an estimate. In this abstract sense, as far as only logical consequence is considered, all admissible heuristics are redundant.

Proposition 4 *Let h be an admissible heuristic, s a state, g a formula, T and $t \leq T$ non-negative integers, and $h^g(s) = n$. Then $\mathit{lits}(s)@t \cup H_T \models \neg g@t'$ for all $t' \in \{t, \ldots, \min(T, t+n-1)\}$.*

The above is simply by the definition of admissibility and the properties of the formalization of action sequences in the propositional logic: If g cannot be reached by less than n actions, then g must be false in all time points before n.

Although logically redundant, the information in admissible heuristics may be useful for *algorithms* for SAT because of the pruning of the search space.

We propose a notion of an *implementation* of an admissible heuristic for the unit propagation procedure.

Definition 5 *Let f be a fact. A clause set χ_T (for the plan length T)* implements *the admissible heuristic $h^f(s)$ if for all $t \in \{0, \ldots, T\}$, all states s, and all $t' \in \{t, \ldots, \min(T, t + h^f(s) - 1)\}$, we have $\mathit{lits}(s)@t \cup H_T \cup \chi_T \vdash_{UP} \neg f@t'$.*

The heuristic and χ_T may depend on a given initial state, but we have not expressed this possible dependency in the above definition. We don't discuss this issue further here.

Next we will show that H_T, alone, encodes one of the best known (but by no means the strongest) admissible heuristics, with χ_T as the empty set. This is the h_{max} heuristic of Bonet and Geffner [2001]. It is implicit in the planning as SAT approach in the strong sense that it is inferred by the unit propagation procedure. The heuristic can be defined as follows.

Definition 6 **(h_{max} [Bonet and Geffner, 2001])** *The h_{max} heuristic for all $f \in F$ is defined by the equation*

$$h^f_{max}(s) = \begin{cases} 0, \text{if } s \models f \\ \min_{a \in A, f \in \mathit{eff}(a)} \left(1 + \max_{f' \in \mathit{prec}(a)} h^{f'}_{max}(s)\right) \end{cases}$$

for which a solution can be obtained as the least fixpoint of a procedure that starts with

$$h^f_{max}(s) = \begin{cases} 0 & \text{if } s \models f \\ \infty & \text{otherwise} \end{cases}$$

and repeatedly performs updates $h^f_{max}(s) := \min(h^f_{max}(s), \min_{a \in A, f \in \mathit{eff}(a)}(1 + \max_{f' \in \mathit{prec}(a)} h^{f'}_{max}(s)))$ for every $f \in F$ until no change takes place.

It is easier use the above fixpoint iteration to first identify all facts with lower bound 0, then those with lower bound 1, 2 and so on. We do this implicitly in the next proof.

Theorem 7 *The empty clause set $\chi_T = \emptyset$, for any $T \geq 0$, implements h^f_{max} for any fact f.*

Proof: We have to show that $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f@t'$ for any state s, any $T \geq 0$, any fact f, and any t and t' such that $t' \in \{t, \ldots, \min(T, t + h^f_{max}(s) - 1)\}$. The proof is by nested inductions, one with T which we leave implicit to simplify the presentation.

Base case $h^f_{max}(s) = 1$: $h^f_{max}(s) = 1$ implies $s \not\models f$. Hence $\neg f@t$ is one of the unit clauses in $\mathit{lits}(s)@t$, and we immediately have $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f@t$.

Inductive case $h^f_{max}(s) > 1$: We show by induction that $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f@(t+i)$ for all $i \in \{0, \ldots, \min(T - t, h^f_{max}(s) - 1)\}$.

> Base case $i = 0$: Proof of $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f@t$ for $i = 0$ is as in the base case.
>
> Inductive case $i \geq 1$: Since $h^f_{max}(s) > i$, for every action $a \in A$ with $f \in \mathit{eff}(a)$ we have $h^{f'}_{max}(s) > i - 1$ for at least one $f' \in \mathit{prec}(a)$. By the outer induction hypothesis we have $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f'@(t + i - 1)$. By Formula 1 we have $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg a@(t + i - 1)$. As this holds for all actions that make f true, by the frame axiom 4 and the inner induction hypothesis $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f@(t + i - 1)$ we have $\mathit{lits}(s)@t \cup H_T \vdash_{UP} \neg f@(t + i)$.

□

All admissible heuristics commonly used in planning are computable in polynomial time. The question of whether an admissible heuristic can be encoded compactly is essentially the question of *circuit complexity* of the corresponding lower bound functions [Balcázar *et al.*, 1988; Papadimitriou, 1994]. Since every polynomial time function can be represented as a polynomial circuit [Papadimitriou, 1994], all of these heuristics can indeed be expressed "compactly" as formulas. For some admissible heuristics it is obvious that their representation is so compact that it is practically usable, for example pattern databases, but for others it is less obvious.

4 Simulation of IDA* with SAT search

Next we give a simulation of a bounded horizon search with IDA* in the DPLL procedure [Davis *et al.*, 1962], showing that with a very simple and natural branching rule, DPLL is at least as powerful as state-space search with IDA*. DPLL can be viewed as a resolution proof system, and it is one of the least powerful of such complete systems [Beame *et al.*, 2004]. Hence this result shows that several other resolution proof systems, including conflict-driven clause learning [Beame *et al.*, 2004; Pipatsrisawat and Darwiche, 2009], are more powerful than state-space search with IDA*.

To obtain the result, we limit the DPLL procedure to choose only action variables as branching variables, assign

```
1:  procedure DPLL($S$)
2:  $S$ := UP($S$);
3:  if $\{x, \neg x\} \subseteq S$ for any $x$ then return false;
4:  $x$ := any variable such that $\{x, \neg x\} \cap S = \emptyset$;
5:  if DPLL($S \cup \{x\}$) then return true;
6:  return DPLL($S \cup \{\neg x\}$)
```

Figure 1: The DPLL procedure

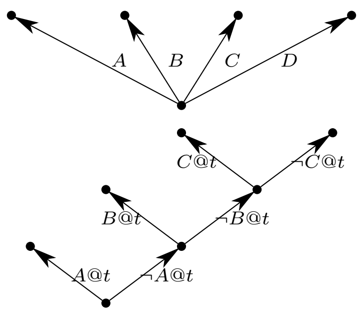

Figure 2: Branching in IDA* vs. DPLL

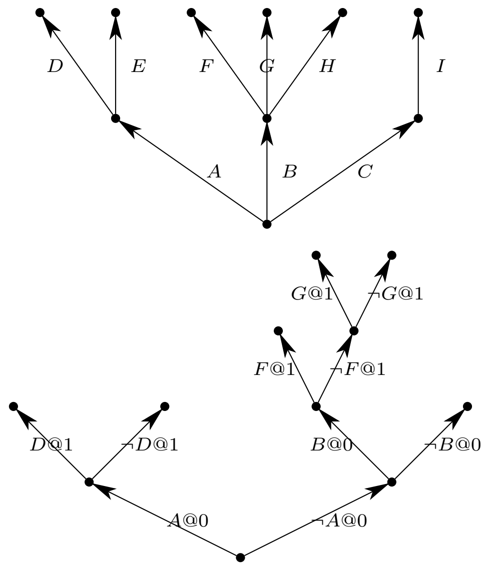

Figure 3: An IDA* search tree and a DPLL search tree

all variables for time t before proceeding with time $t+1$, and always choose the truth-value *true* first.

The next lemma shows how unit propagation computes successor states with the encoding from Section 2.2.

Lemma 8 *Let s and s' be states, $a_1, \ldots, a_k$ a sequence of actions so that $s' = succ_{a_k}(\cdots succ_{a_1}(s) \cdots)$. Let $k \leq n$. Let $L_a = \bigcup_{i=0}^{n-1} A@i$. Let $L_a^+ = \{a_1@0, \ldots, a_k@(k-1)\}$. Then lits$(s)@0 \cup H_n \cup L_a^+ \cup \{\neg a@t | a@t \in L_a \backslash L_a^+\} \vdash_{UP}$ lits$(s')@k$.*

The main result of the section shows that DPLL search trees never have more nodes than IDA* search trees.

Theorem 9 *Let N be the size of a failed IDA* search tree with heuristic h and depth-bound T. Then there is a DPLL search tree of size less than $2N$ that shows lits$(I)@0 \cup H_T \cup \chi_T \cup g@T$ unsatisfiable.*

Proof: The DPLL procedure is given in Figure 1. It uses the unit propagation procedure UP(S), which resolves every unit clause l with clauses $\bar{l} \vee \phi$ to obtain ϕ (or the empty clause when resolving l and $\bar{l}$) which is added to S. When no further clauses can be derived, the resulting clause set is returned.

We prove that the number of nodes in the DPLL search tree is at most that of the IDA* search tree, when the variable on line 4 is always chosen so that it is an action variable $a@t$ and for all actions variables $a'@t'$ such that $t' < t$, either $a'@t' \in S$ or $\neg a'@t' \in S$.

The proof is by mapping IDA* search trees to DPLL search trees of slightly different structure, and showing that DPLL traverses the trees at most as far as IDA* does. The difference between the search trees of DPLL and state-space search with IDA* and DPLL is illustrated in Figure 2. The node in the IDA* tree on top has 4 successor nodes, corresponding to the actions A, B, C and D. The DPLL tree has binary branches only, with an action and its negation as the two arc labels, as required by the DPLL procedure. The last (rightmost) successor of any node does not have its action as a branching variables. In our example, $D@t$ is not a branching variable. Instead, it is inferred from $\neg A@t, \neg B@t, \neg C@t$ and the axiom 7 from Section 2.2. A node with n successors requires a total of $2n-1$ nodes in the corresponding DPLL search tree. Hence there are less than twice as many nodes in the DPLL tree. A full example of an IDA* search tree, with each branch cut-off after two actions, and the corresponding DPLL search tree are given in Figure 3.

When IDA* has reached a node in depth n corresponding to an action sequence $a_1, \ldots, a_n$, the corresponding state is $s' = \text{succ}_{a_n}(\cdots \text{succ}_{a_1}(I) \cdots)$. The clause set in the corresponding node in the DPLL search tree includes the same actions as literals $a_1@0, \ldots, a_n@(n-1)$ and the negations of all other action variables for time points $\leq n-1$. Hence by Lemma 8 DPLL has inferred the set *lits*$(s')@(n-1)$. If $n + h(s') > T$, the node will be a cut-off node for IDA*. Because χ_T implements the heuristic h, by Definition 5 we have *lits*$(s')@(n-1) \cup H_T \cup \chi_T \vdash_{UP} \bar{l}@T$ for at least one goal literal $l \in G$. Hence DPLL will backtrack in this node, if it has not backtracked earlier (and sometimes it has, as shown by Theorem 10.) Therefore the DPLL search tree will not be expanded further than the IDA* search tree, and it is – in terms of the number of nodes – less than twice the size. □

The above theorem assumes that the IDA* search does not detect cycles. Cycles can be eliminated from the DPLL search by encoding a constraint that says that the states at any two time points are different. Eliminating the cycles, for both IDA* and DPLL, is not necessary for the correctness or the termination of the search, but may improve performance.

The above result is not specific to DPLL. A similar simulation is possible with other systematic proof methods for SAT, for example the CDCL algorithm. A close correspondence between the branches of a search tree (paths from the root to a leaf, as in Figure 3) and the clauses learned by CDCL can be easily established. To sketch the proof idea, note that every branch in the DPLL tree in Figure 3 corresponds to a clause, for example $A@0 \vee \neg B@0 \vee F@1 \vee \neg G@1$. It can be *learned* by the CDCL algorithm, similarly to the unit resolution derivation in the proof of Lemma 8. Resolving these clauses, for example $A@0 \vee \neg B@0 \vee F@1 \vee \neg G@1$ with $A@0 \vee \neg B@0 \vee F@1 \vee G@1$, yielding $A@0 \vee \neg B@0 \vee F@1$, and this clause further with $A@0 \vee \neg B@0 \vee \neg F@1$, and so on, will eventually derive the empty clause.

5 Exponential Separation of A* and SAT

We show that some problems that are trivial for SAT (in the sense that a simple unit-resolution strategy will solve them) are very difficult for state-space search with both A* and IDA* when employing the same heuristic.

Theorem 10 *State-space search with A* and a heuristic h is sometimes exponentially slower than any algorithm for SAT that uses unit resolution, if the latter implements h.*

Proof: We give an example for which unit resolution immediately proves the lack of plans of length $n-1$ when n is the shortest plan length, and finds a plan of length n, but A* with h_{max} will visit an exponential number of states.

The idea is the following (illustrated in Figure 4.) The goals can be reached with the action sequence $y_1, \ldots, y_{k+2}$ of length $k+2$. Seemingly (as far as h_{max} is concerned), the goals can also be reached with a $k+1$ step plan consisting of k actions from $\{x_1, x'_1, \ldots, x_k, x'_k\}$ followed by bad. However, bad irreversibly falsifies one of the goals. The actions $x_1, x'_1, \ldots, x_k, x'_k$ induce a very large state space with $2^{k+1}-1$ states, which will be exhaustively searched by A*, making its runtime exponential in k.

On the other hand, unit propagation will immediately infer that action bad cannot be taken and that $y_1, \ldots, y_{k+2}$ is the only possible plan. If $T < k+2$, a refutation is immediately found. If $T = k+2$, the unit resolution procedure will find a satisfying assignment.

Next we formalize the example in detail. The state variables are $p_1, \ldots, p_{k+1}$ (of which exactly one is true in any reachable state), $r_2, \ldots, r_{k+2}$ (of which at most one is true in any reachable state), $b_1, \ldots, b_k$ (which can be independently true or false), and g_1 and g_2. Only g_1 and p_1 are initially true. All other variables are false. The goal consists of g_1 and g_2. The actions are given in Table 1.

The h_{max} estimate for the state reached by action y_1 from the initial state is $k+1$ (which is also the actual distance.)

The h_{max} estimate for all states reached from the initial state by a sequence of i actions from the set $\{x_1, \ldots, x_k, x'_1, \ldots, x'_k\}$ is $k-i+1$ (which always seems better than taking the action y_1 first), although the goals cannot be reached this way: the action bad fools h_{max} to think that goals can be reached with bad as soon as p_{k+1} has been made true, but bad actually irreversibly falsifies one of the goals.

action	precon	add	del
x_1	p_1	p_2	p_1
x'_1	p_1	p_2, b_1	p_1
x_2	p_2	p_3	p_2
x'_2	p_2	p_3, b_2	p_2
$\vdots$			
x_k	p_k	p_{k+1}	p_k
x'_k	p_k	p_{k+1}, b_k	p_k
bad	p_{k+1}	g_2	g_1
y_1	p_1	r_2	p_1
y_2	r_2	r_3	r_2
$\vdots$			
y_k	r_k	r_{k+1}	r_k
y_{k+1}	r_{k+1}	r_{k+2}	r_{k+1}
y_{k+2}	r_{k+2}	g_2	r_{k+2}

Table 1: The actions in the proof of Theorem 10

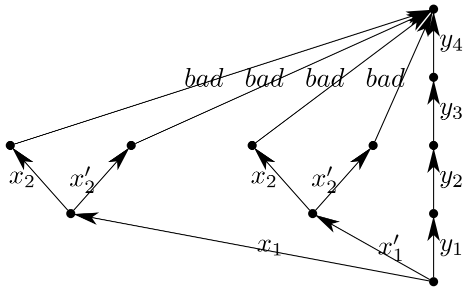

Figure 4: Illustration of the proof of Theorem 10 with $k = 2$

The A* algorithm generates 2^k states with p_{k+1} true and different values for $b_1, \ldots, b_k$, as determined by the choices between x_i and x'_i for $i \in \{1, \ldots, k\}$.

Next we show that $lits(I)@0 \cup H_{k+1} \cup \{g_1@k+1, g_2@k+1\}$ is found unsatisfiable by unit propagation.

The frame axiom for g_1 is $g_1@t \vee \neg g_1@(t+1)$ because none of the actions makes g_1 true. Hence from the goal literal $g_1@k+1$ we can infer $g_1@k, g_1@(k-1), \ldots, g_1@1$ with unit resolution, and therefore the bad action can be inferred to be never possible: we get $\neg bad@k, \ldots, \neg bad@0$.

From $\neg r_2@0$ (initial state) and $r_2@0 \vee \neg r_2@1 \vee y_2@0$ (frame axiom) and $\neg y_2@0 \vee r_2@0$ (precondition) we can infer $\neg r_2@1$ with unit resolution. Similarly we can infer $\neg r_i@1$ for all $i \geq 3$, and, more generally, $\neg r_i@j$ and $\neg y_i@j$ and $\neg g_2@(j+1)$ for all $1 < j < i \leq k+2$. Since $g_2@k+1$ belongs to the clause set, we have derived the empty clause. The number of unit resolution steps is quadratic in k and linear in the size of the clause set.

When there are $k+2$ time points, the same reasoning shows that unit propagation yields $y_1@0, \ldots, y_{k+2}@(k+1)$ and $\neg g_2@(k+1)$. As y_{k+2} is the only action that can make g_2 true, we get $y_{k+2}@(k+1)$ by unit resolution from the frame axiom. The rest of the plan is obtained analogously. □

The above proof can be adapted to other admissible heuristics that use a delete-relaxation or other simplification that makes it seem that the action bad does not delete g_2.

6 Related Work

The power of different limited inference methods for approximate reachability (e.g. planning graphs) have been investigated earlier [Brafman, 2001; Geffner, 2004; Rintanen, 2008]. All three works investigate ways of strengthening inference with SAT to derive the invariants/mutexes in planning graphs. None of them attempts to relate SAT to state-space search, and the only result that is directly related to ours is Brafman's Lemma 1 which shows that for a sequential encoding (one action at each time point) Reachable-1 – which is closely related to h_{max} – is stronger than unit propagation. This seems to conflict with our Theorem 7, which shows unit propagation to be at least as strong. This discrepancy is due to our use of explanatory frame axioms. Brafman's Lemma 3 for parallel plans uses explanatory frame axioms, and then unit propagation is at least as strong as Reachability-1.

7 Conclusions

We have shown that IDA* search for state-space reachability problems can be efficiently simulated by DPLL search. Our results also show that unit resolution is sometimes exponentially more efficient than state-space search with A* or IDA*: A* and IDA* need exponential time, but short resolution proofs are immediately found with unit resolution. The work complements recent experimental results that suggest that SAT-based methods for non-optimal planning are at least as strong as state-space search [Rintanen, 2010].

One could view Theorems 9 and 10 as suggesting a simple IDA* style state-space search algorithm extended with some (unit) resolution inferences. Such a procedure would in some cases improve IDA*, but would not fully benefit from the power of modern SAT algorithms. Much of that strength comes from the less rigid structure of the proofs, which would mostly be lost in a pure forward-chaining state-space search. The general challenge is to utilize this strength and guarantee a performance that typically exceeds A* and IDA*, also for problems that would seem to be more suitable for forward-chaining state-space search.

A more direct avenue to useful implementations is to enhance SAT-based optimal planning with encodings of more powerful admissible heuristics than h_{max}. Unlike state-space search algorithms, SAT-based methods can take advantage of lower bounds for arbitrary facts (not only goals) as additional constraints. This suggests that the more powerful lower bounding methods have much more to contribute than what is currently utilized by state-space search methods. This is an important topic for future work.

References

[Balcázar *et al.*, 1988] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I.* Springer-Verlag, Berlin, 1988.

[Beame *et al.*, 2004] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[Brafman, 2001] R.I. Brafman. On reachability, relevance, and resolution in the planning as satisfiability approach. *Journal of Artificial Intelligence Research*, 14:1–28, 2001.

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

[Geffner, 2004] Héctor Geffner. Planning graphs and knowledge compilation. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004)*, pages 662–672, 2004.

[Hart *et al.*, 1968] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum-cost paths. *IEEE Transactions on System Sciences and Cybernetics*, SSC-4(2):100–107, 1968.

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 359–363. John Wiley & Sons, 1992.

[Korf, 1985] R. E. Korf. Depth-first iterative deepening: an optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[Mitchell, 2005] David G. Mitchell. A SAT solver primer. *EATCS Bulletin*, 85:112–133, February 2005.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[Pipatsrisawat and Darwiche, 2009] K. Pipatsrisawat and A. Darwiche. On the power of clause-learning SAT solvers with restarts. In I. P. Gent, editor, *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming, CP 2009*, number 5732 in Lecture Notes in Computer Science, pages 654–668. Springer-Verlag, 2009.

[Rintanen, 2008] Jussi Rintanen. Planning graphs and propositional clause-learning. In Gerhard Brewka and Patrick Doherty, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference (KR 2008)*, pages 535–543. AAAI Press, 2008.

[Rintanen, 2010] Jussi Rintanen. Heuristics for planning with SAT. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010, 16th International Conference, CP 2010, St. Andrews, Scotland, September 2010, Proceedings.*, number 6308 in Lecture Notes in Computer Science, pages 414–428. Springer-Verlag, 2010.