BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving

Henry Kautz  
AT&T Laboratories  
kautz@research.att.com

Bart Selman  
Cornell University  
selman@cs.cornell.edu

It has often been observed that the classical AI planning problem (that is, planning with complete and certain information) is a form of logical deduction. Because early attempts to use general theorem provers to solve planning problems proved impractical, research became focused on specialized planning algorithms. Sometimes the relationship to inference was explicitly acknowledged; for example, the STRIPS system (Fikes and Nilsson 1971) was originally described as a way to make theorem-proving practical. In other work the relationship to deduction was developed after the fact. For example, Chapman's (1985) work on TWEAK clarified the logic behind one variety of non-linear planning.

The belief that planning required specialized deductive algorithms was challenged by our work on planning as propositional satisfiability testing (Kautz and Selman 1992, 1996). SATPLAN showed that a general propositional theorem prover could indeed be competitive with some of the best specialized planning systems. The success of SATPLAN can be attributed to two factors:

- The use of a logical representation that has good computational properties. Both the fact that SATPLAN uses propositional logic instead of first-order logic, and the particular conventions we suggested for representing time and actions, are significant. Differently declarative representations that are semantically equivalent can still have quite distinct computational profiles. (For this reason we believe that the search for epistemologically satisfactory representations (McCarthy and Hayes 1969) should go hand-in-hand with the study of practical reasoning algorithms, rather than being carried out as a separate activity.)

- The use of powerful new general reasoning algorithms such as Walksat (Selman, Kautz, and Cohen 1994). Many researchers in different areas of computer science are creating faster SAT engines every year. Furthermore, these researchers have settled on common representations that allow algorithms and code to be freely shared and fine-tuned. As a result, at any point in time the best general SAT engines tend to be faster (in terms of raw inferences per second) than the best specialized planning engines. In principle, of course, these same improvements could be applied to the specialized engines; but by the time that is done, there will be a new crop of general systems.

An approach that shares a number of features with the SATPLAN strategy is the Graphplan system, developed independently by Blum and Furst (1995). Graphplan broke previous records in terms of raw planning speed, and has become a popular planning framework. Comparisons to SATPLAN show that neither algorithm is strictly superior. For example, SATPLAN is faster on a complex logistics domain, they are comparable on the blocks world, and on several other domains Graphplan is faster.

Graphplan bears an important similarity to SATPLAN: both systems work in two phases, first creating a propositional structure (in Graphplan, a plan graph, in SATPLAN, a CNF wff) and then searching that structure. The propositional structure corresponds to a fixed plan length, and the search reveals whether a plan of that length exists. Furthermore, we showed in Kautz and Selman (1996) that the plan graph has a direct translation to CNF, and that the form of the resulting formula is very close to the original conventions for SATPLAN. We hypothesize that the differences in performance of the two system can be explained by the fact that Graphplan uses a better algorithm for instantiating the propositional structure, while SATPLAN uses more powerful search algorithms.

SATPLAN fully instantiates a complete problem instance before passing it to a simplifier and a solver. By contrast, Graphplan interleaves plan graph instantiation and simplification. This can often be a big
win. Furthermore, the simplification algorithm used by 
Graphplan is more powerful than the unit-propagation 
simplifier the original SATPLAN employed. By study-
ing the details of Graphplan, we determined that it 
is employing a (limited application of) negative binary 
propagation. This rule is:

\[
\text{given: } \{ \neg p \lor \neg q \}, \quad \{ p \lor r \lor s \lor \ldots \} \\
\text{infer: } \{ \neg q \lor r \lor s \lor \ldots \}
\]

These observations have led us to create a new sys-
tem that combines the best features of Graphplan and 
SATPLAN. This system, called blackbox, works in 
three phases:

1. A planning problem (specified in a standard STRIPS 
notation) is converted to a plan graph;

2. The plan graph is converted to a CNF wff;

3. The wff is solved by any of a variety of fast SAT 
engines.

(The earlier MEDIC system of Ernst, Millstein, and 
Weld (1997) also converts STRIPS notation into CNF, 
but does not use the plan graph intermediate form. 
Further, the SAT engines included in blackbox are 
more powerful than those in the original MEDIC dis-
tribution.)

Blackbox currently includes the local-search SAT 
solver Walksat and the systematic SAT solver satz (Li 
and Anbalagan 1997), as well as the original Graph-
plan engine (that searches the plan graph instead of 
the CNF form). In order to have robust coverage over 
a variety of domains, the system can employ a schedule 
of different solvers. For example, it can run Graphplan 
for 30 seconds, then Walksat for 2 minutes, and if still 
no solution is found, satz for 5 minutes.

The blackbox system actually introduces new SAT 
technology as well, namely the use of randomized com-
plete search methods. As shown in Gomes, Selman, and 
Kautz (1998), systematic solvers in combinatorial do-
mains often exhibit a “heavy tail” behavior, whereby 
they get often “stuck” on particular instances. Adding 
a small amount of randomization to the search heuris-
tic and rapidly restarting the algorithm after a fixed 
number of backtracks can dramatically decrease the 
average solution time. We applied this randomiza-
tion/restart technique to the version of satz used by 
blackbox.

The use of the Graphplan front-end and this new 
randomized/restart solver leads to a very high level of 
performance. Of particular note is the fact that it can 
solve the largest, hardest logistics problems we have 
constructed, directly from the STRIPS-style problem 
description, in about 6 minutes. One such problem 
(“logistics.d” from Kautz and Selman 1998) contains 
\(10^{16}\) states, and its solution involves 105 actions over 
14 time steps. By comparison, Graphplan alone takes 
about 40 minutes to solve a smaller problem (logis-
tics.b) that has \(10^9\) states, and cannot handle the 
larger problem.

It is important to note that the success of SATPLAN 
on the logistics domain as reported in Kautz and Sel-
am (1996) involved a different kind of propositional 
encoding, called a “state-based encoding”, that incor-
porated general domain knowledge (Kautz and Selman 
1998) which can be hard to derive from the STRIPS in-
put. The state-based encodings were created by hand. 
Our experiments with blackbox are the first to show 
that this domain can be solved when encodings are 
automatically generated from the STRIPS-style input.

Blackbox is an evolving system. The newest imple-
mentation of blackbox accepts the PDDL input lan-
guage (McDermott et al. 1998), and can be down-
loaded from http://wwwprojects.ai.mit.edu/kautz. Fu-
ture versions will include other SAT engines and both 
general and planning-specific simplification routines. 
One open research question our future work will ad-
dress is whether there are more powerful simplification 
Algorithms than negative binary propagation that are 
generally cost-efficient across all planning domains.

One other promising extension we hope to have 
available soon is to allow for search control knowledge 
slated in a generic declarative form. The need for a 
mechanism for declarative search control has long 
advocated by John McCarthy, but so far it has been hard 
to make concrete. The idea is to be able to add heuris-
tic control to a general theorem prover or search engine 
without having to modify the search control mecha-
nisms of the search procedure itself. Examples of such 
control rules are “Do not unload a package right after 
it has been loaded on a truck or a plane,” or “Do not 
move a package after it has reached its final destina-
tion.” Recent experiments with the SATPLAN sys-
tem, where both the domain operators and the control 
knowledge are encoded by hand as logical axioms, has 
show that such statements can dramatically improve 
performance (Kautz and Selman 1998). Our future re-
lease of blackbox will allow such control knowledge to 
be easily expressed in (an extension of) PDDL.

References

through planning graph analysis. Proc. IJCAI-95, 
Montreal, Canada.

TR AI-TR-802, M.I.T. AI Lab.


