Automated planning with goal utility dependencies within a satisfiability framework

Richard A. Russell
Richard.Russell@cl.cam.ac.uk
Computer Laboratory, University of Cambridge

1 Introduction

In classical planning the objective is represented as a conjunction of goals with the intention that each one is to be achieved. Modern real world planning agents are faced with problems where there are too many goals and only a subset of them can feasibly be achieved. This is known as oversubscription planning or partial satisfaction planning. Until recently, approaches to this problem assume that each goal is equally important or statically weighted, but my work attempts to solve problems that have a general utility function with no such assumptions. This is relevant to problem domains where the utility of achieving a collection of goals is significantly greater than the sum of the utilities of achieving each goal individually.

I have used a satisfiability (SAT) framework to investigate the problem of goal utility dependencies. Goals are mapped one-to-one with clauses and the problem becomes finding a variable assignment that makes some clauses true and others false so that the utility of those clause truth values is maximised.

Motivation

Goal utility dependencies would be beneficial in the following scenarios:

- Logistics — delivery of complementary products to retailers at the same time is likely to increase sales.
- Remote scientific agents — choosing which experiments to perform on limited resources to maximise scientific return, assuming related experiments are preferred to disjoint ones.

2 Compiling to SAT

SAT solvers have steadily progressed to the stage that they can now handle millions of clauses and thousands of variables. Kautz and Selman [2] describe an approach to planning that compiles the planning graph to a SAT formula. They then use a high-performance SAT solver to find a variable assignment that satisfies the formula. If one is found then a valid plan can be extracted from it.The following example demonstrates this using the blocks world domain.

Example

Starting with an arrangement of stacked and labelled blocks, the problem is to move blocks around so that they are placed into a desired arrangement. A block can be moved onto another only if both blocks have nothing on top of them. A block can also be moved to the table regardless of how many blocks are already on the table and only one block can be moved at a time.

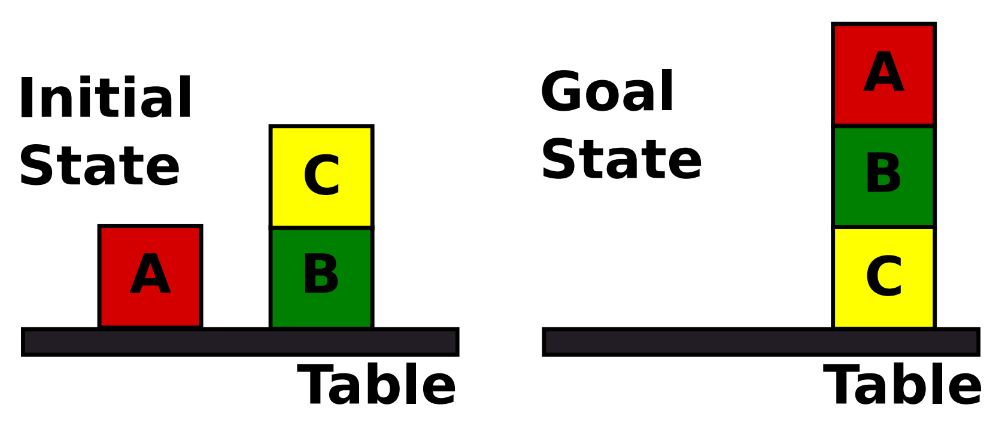

Figure 1: An example problem in the blocks world planning domain.

The blocks world problem shown in Figure 1 could be represented within the SAT framework as follows:

1. The initial state is described as:
$$\texttt{ClearA}_0 \land \texttt{ClearC}_0 \land \texttt{COnB}_0 \land \texttt{AOnT}_0 \land \texttt{BOnT}_0$$
2. The goal state is described as:
$$\texttt{AOnB}_6 \land \texttt{BOnC}_6 \land \texttt{ClearA}_6 \land \texttt{COnT}_6$$
3. Actions imply their preconditions and effects $A \Rightarrow P, E$.
$$\texttt{MoveAToC}_1 \Rightarrow$$
$$(\texttt{ClearA}_0 \land \texttt{ClearC}_0 \land \texttt{COnB}_0 \land \texttt{AOnT}_0)$$
$$\land(\texttt{AOnC}_2 \land \neg\texttt{AOnT}_2 \land \neg\texttt{ClearC}_2)$$
When put into clausal form the above becomes:
$$\{\neg\texttt{MoveAToC}_1 \lor \texttt{ClearA}_0\} \land \dots$$
$$\dots \land \{\neg\texttt{MoveAToC}_1 \lor \neg\texttt{ClearC}_2\}$$

3 Algorithm

My algorithm is a variation of Iterated Robust Tabu Search that is used in the MiniMaxSAT solver [1]. It exploits the utility function to intelligently focus its efforts on selecting clauses that will increase utility.

1. Start with a random assignment to variables and use this to determine which clauses are satisfied. Calculate the utility of this initial assignment.
2. Use the decision tree representation of the utility function to determine which clause we should try to flip to give a maximal improvement in the utility of the assignment.
3. See if we can change the variable assignment to flip that clause without changing the truth values of other clauses.
4. If we cannot do this then restore to the previous assignment. If we can then update the state of the program so that we remember this solution.
5. Loop until we reach a utility that is within some bounds of the maximum utility. Occasionally force some clauses to flip so that the search does not stagnate.
6. Return the best assignment found so far.

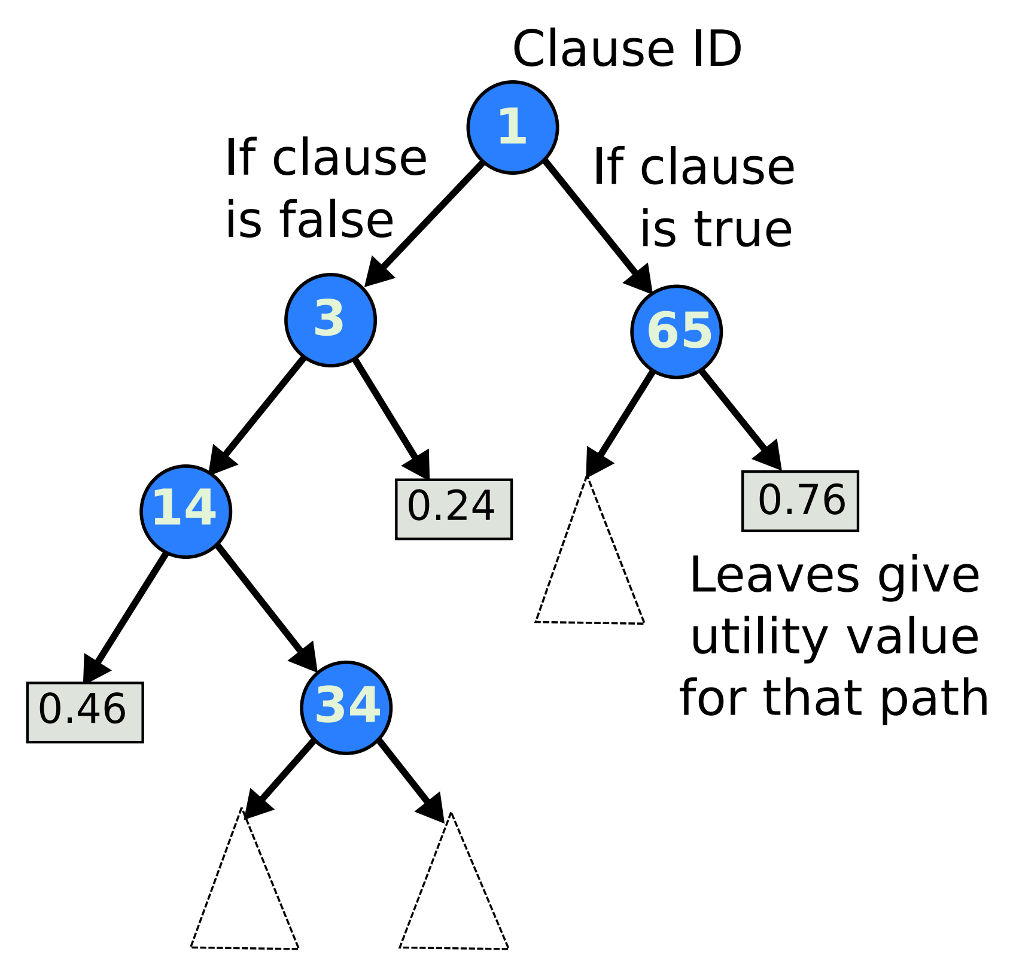

Figure 2: How the utility function is represented.

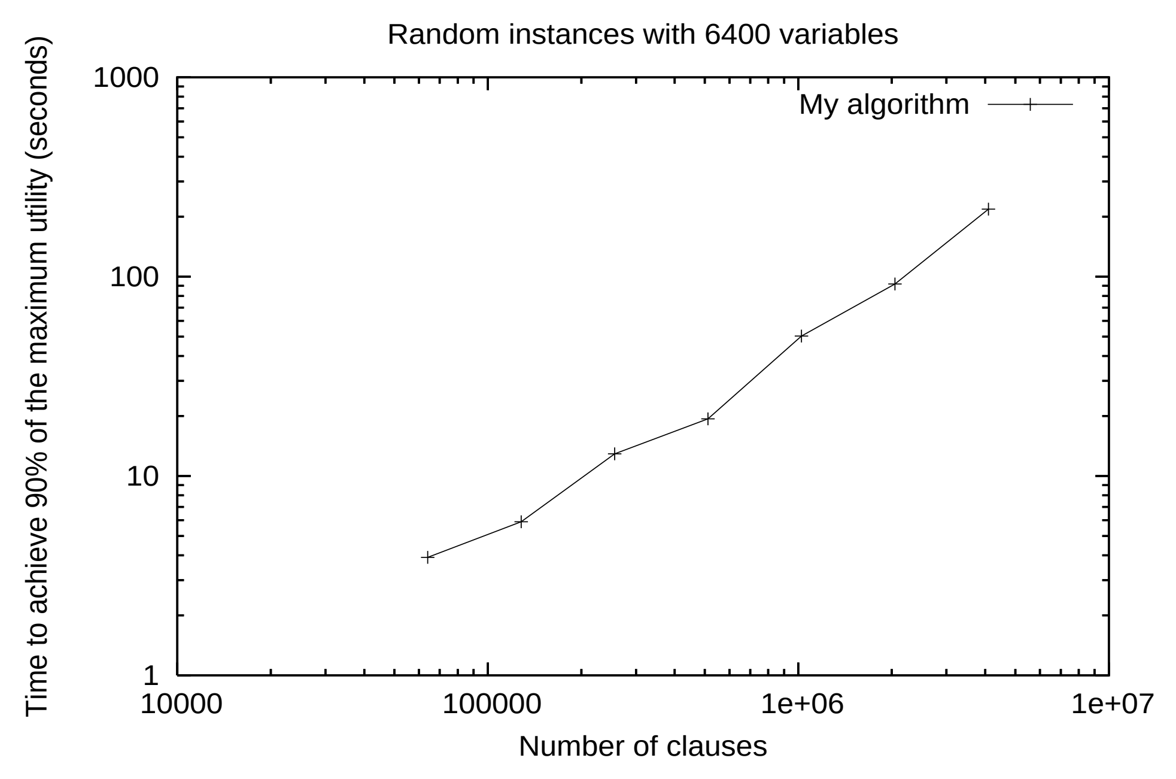

Figure 3: Results of my algorithm solving randomly generated formulae and utility functions.

4 Pros and cons

SAT compilation is a leading general purpose method for performing deterministic planning. Therefore it makes sense to try to extend this framework to handle goal utility dependencies. By using a general purpose SAT solver there is the possibility that improvements in SAT solver research carry straight through to improvements in the performance of this system.

However, we are still subject to the restrictions of working within a SAT framework. It is hard to represent actions that have durations, concurrent actions or any type of uncertainty, all of which are desirable in many real world planning problems.

5 Future work

1. Compile from planning graphs to this variation of SAT and compare with existing systems.
2. Extend planning for goal utility dependencies to more realistic models, e.g., handling actions with durations.

References

[1] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: A New Weighted Max-SAT Solver. *Proc. SAT*, 2007.

[2] H. Kautz and B. Selman. Planning as satisfiability. *Proc. ECAI*, pages 359–363, 1992.