

Automatically Generating Implied Clauses for SAT

Lyndon Drake, Alan Frisch and Toby Walsh

{lyndon, frisch, tw}@cs.york.ac.uk

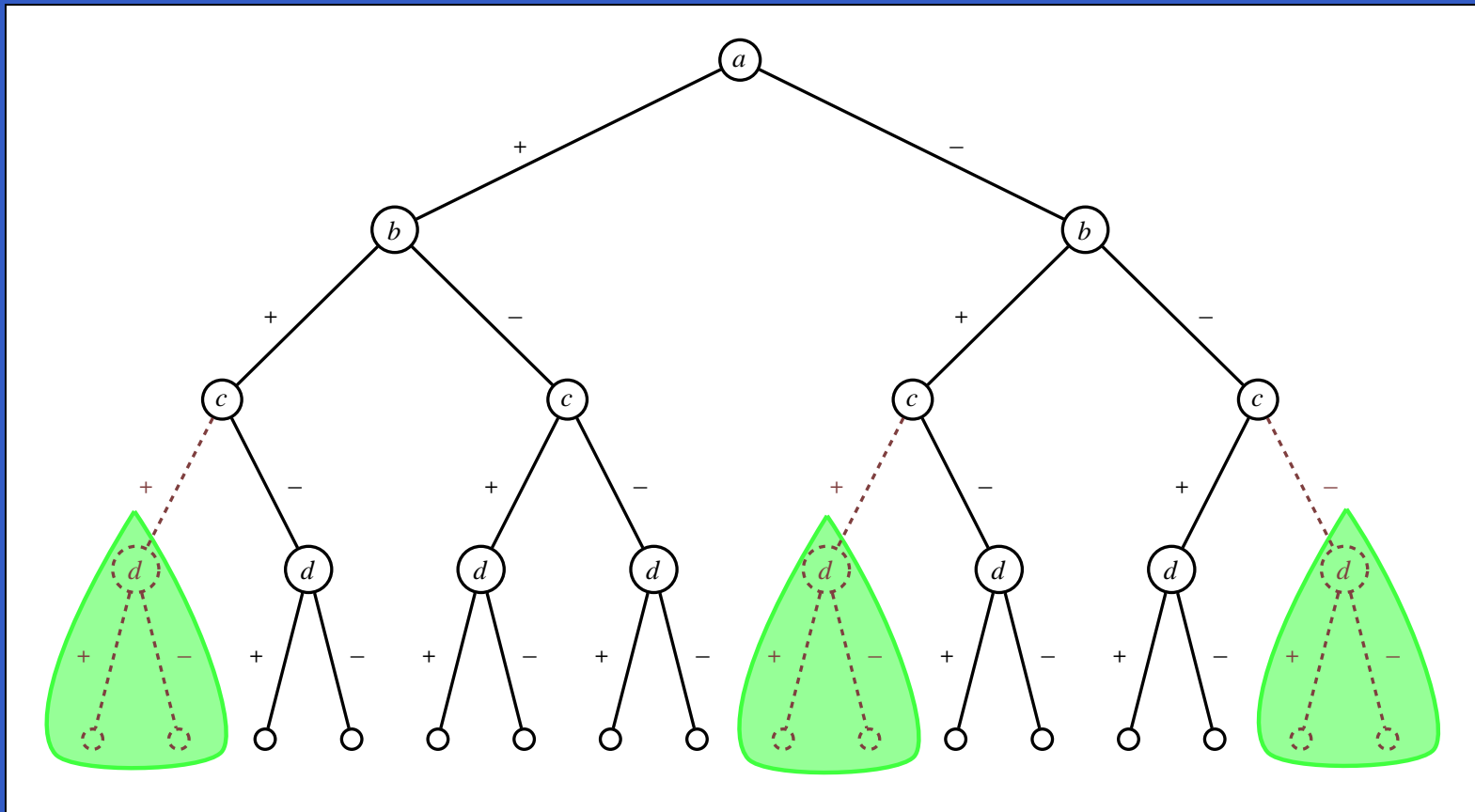
University of York, United Kingdom

Example: Unsatisfiable SAT instance

$\neg a \vee \neg b \vee \neg c$
 $a \vee \neg b \vee \neg c$
 $a \vee b \vee c$
 $\neg b \vee d$
 $\neg b \vee \neg d$
 $a \vee c \vee d$
 $\neg b \vee c \vee \neg d$

$a \vee b \vee d$
 $a \vee \neg c \vee \neg d$
 $\neg a \vee b \vee d$
 $b \vee \neg c \vee \neg d$
 $\neg a \vee c \vee d$
 $b \vee c \vee \neg d$

Example: search tree



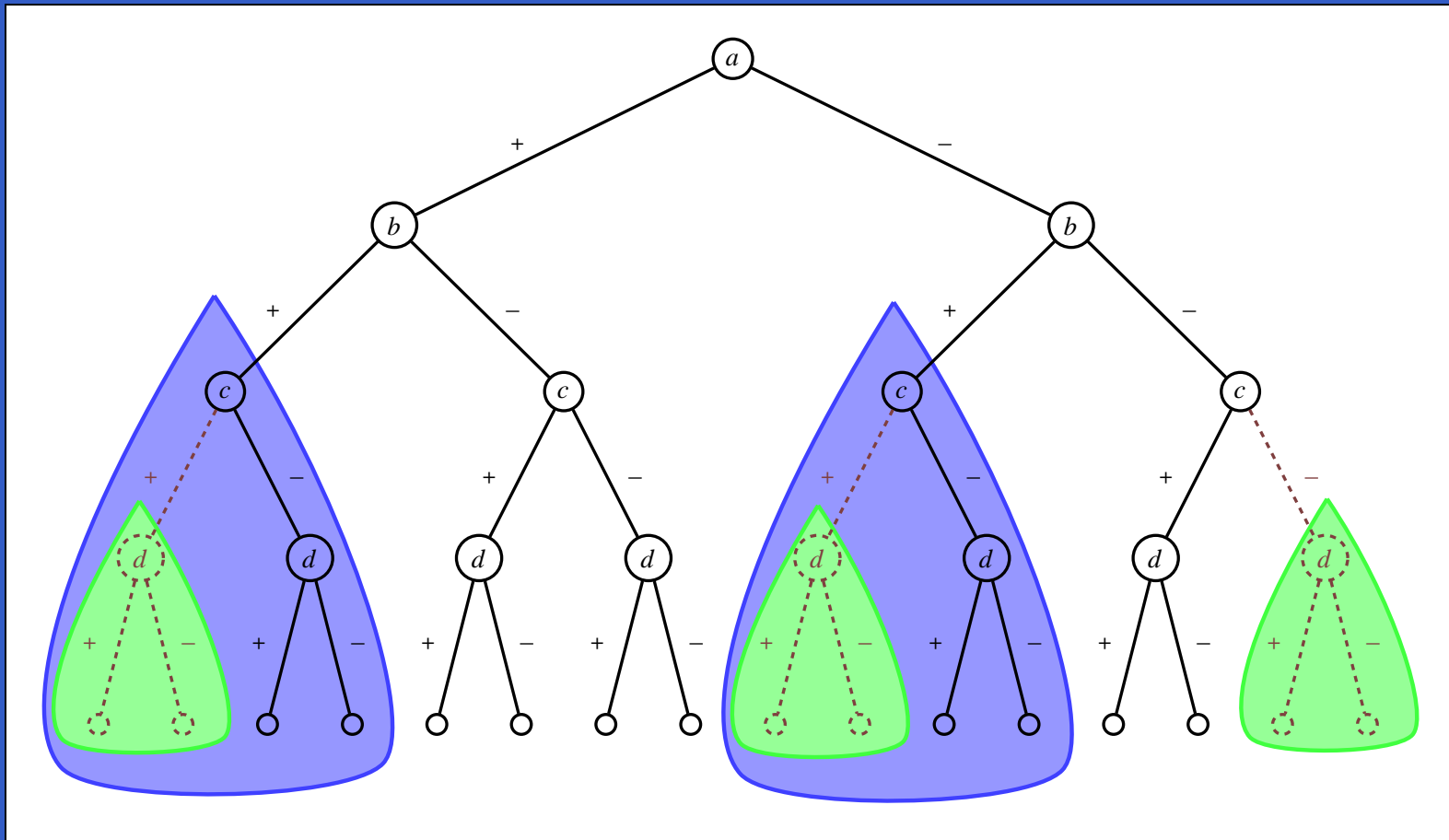
Example: neighbourhood resolution 1

$$\begin{array}{c} \neg b \vee d \\ \neg b \vee \neg d \\ \hline \neg b \end{array}$$

Example: neighbourhood resolution 2

$$\begin{array}{ccccccccc} \neg a & \vee & \neg b & \vee & \neg c & & a & \vee & b & \vee & d \\ a & \vee & \neg b & \vee & \neg c & & a & \vee & \neg c & \vee & \neg d \\ a & \vee & b & \vee & c & & \neg a & \vee & b & \vee & d \\ \implies & \neg b & & & & & b & \vee & \neg c & \vee & \neg d \\ a & \vee & c & \vee & d & & \neg a & \vee & c & \vee & d \\ \neg b & \vee & c & \vee & \neg d & & b & \vee & c & \vee & \neg d \end{array}$$

Example: pruned search tree



Motivation

- SAT is the archetypal NP-complete problem, and therefore interesting in its own right
- Many other problems can be usefully mapped to SAT, including quasigroup completion problems and model checking
- Adding inference to search can guarantee pruning, but is often too expensive

Previous work

Combining resolution and search:

- Rish and Dechter. Resolution versus search: two strategies for SAT. In SAT2000, IOS Press, 2000.
- Cha and Iwama. Adding new clauses for faster local search. In Proc AAAI-96, 1996.

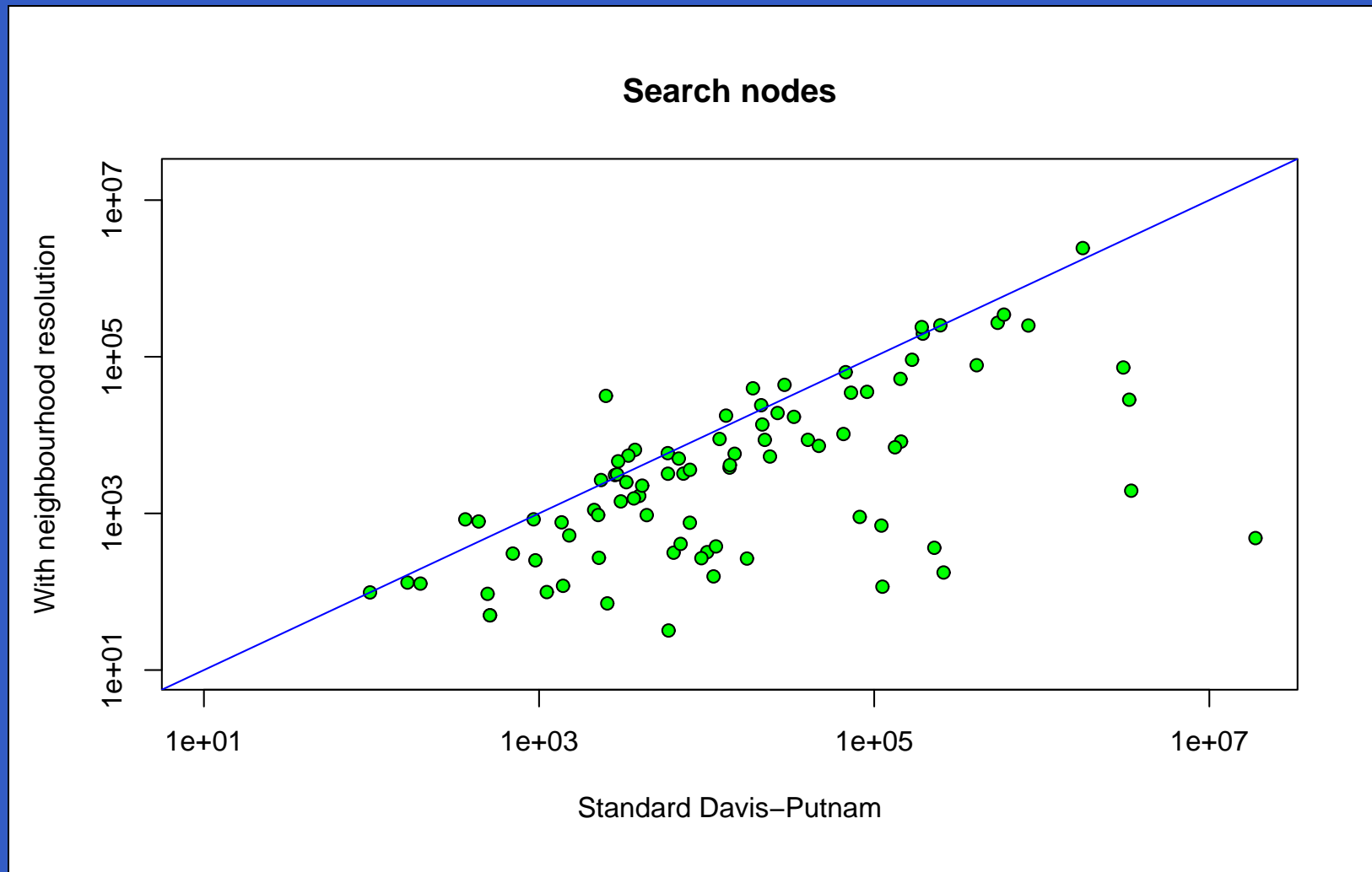
First attempt: during search

- At each branching node, we identified all current neighbours and resolved them
- Far too expensive in time for practical use, but confirmed the potential value of the technique

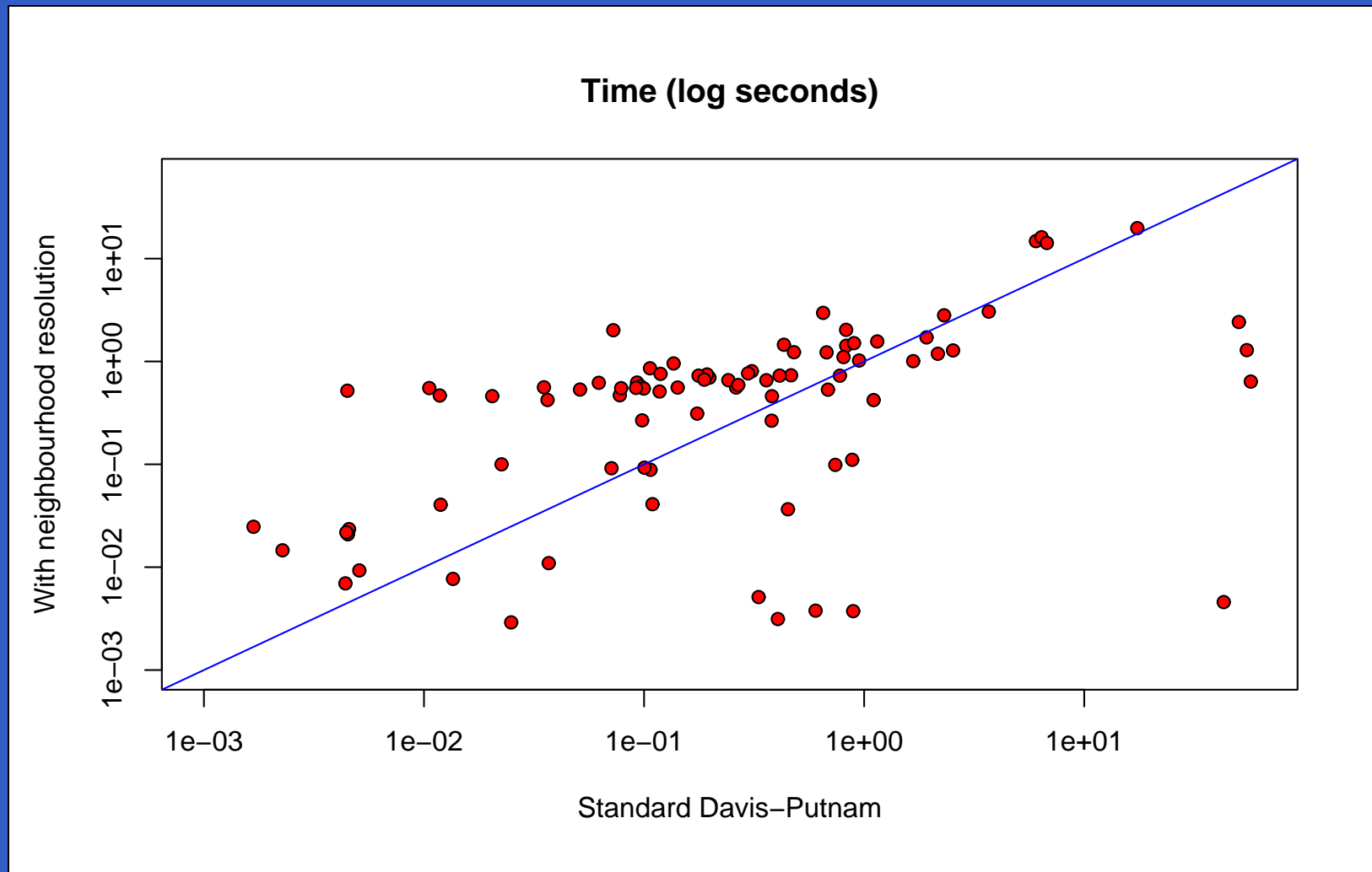
Second attempt: preprocessing

- Doing one class of neighbourhood resolutions at the root of the search tree is equivalent to doing strong neighbourhood resolutions at each branching node
- A single search for neighbours saves time, but we still potentially have a large number of new clauses

Results: the good news



Results: the bad news



Future work

- Improve the performance of neighbourhood resolution
- Investigate interactions with other techniques for generating implied clauses

Summary

- We have taken neighbourhood resolution and applied it to a complete SAT solver
- Using neighbourhood resolution as a preprocessing step is not yet viable but shows promise