Graph Backtracking for AVATAR

Vampire Workshop

Robin Coutelier

TU Wien, Vienna, Austria
robin.coutelier@tuwien.ac.at

Vampire Workshop 2026

Acknowledgements

This work was done in collaboration with Thomas Hader and Laura Kovács.

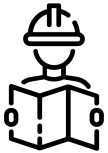
The authors acknowledge support from the ERC Consolidator Grant ARTIST 101002685; the TU Wien Doctoral College TrustACPS; the FWF SpyCoDe SFB projects F8504; the WWTF Grant ForSmart 10.47379/ICT22007

European Research Council
Established by the European Commission

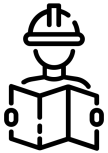
Der Wissenschaftsfonds.

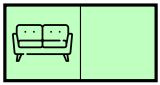
Vienna Science
and Technology Fund

Introduction

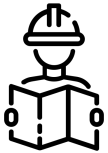


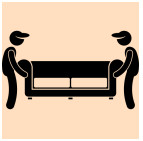
Introduction



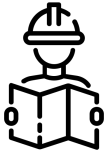


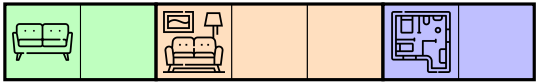
Introduction

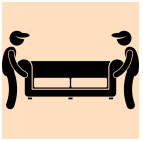




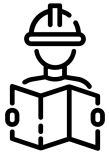
Introduction

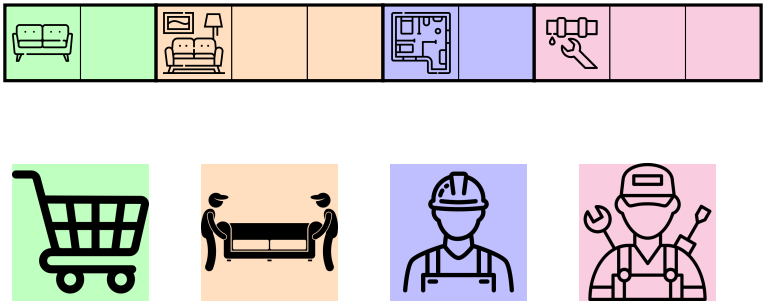




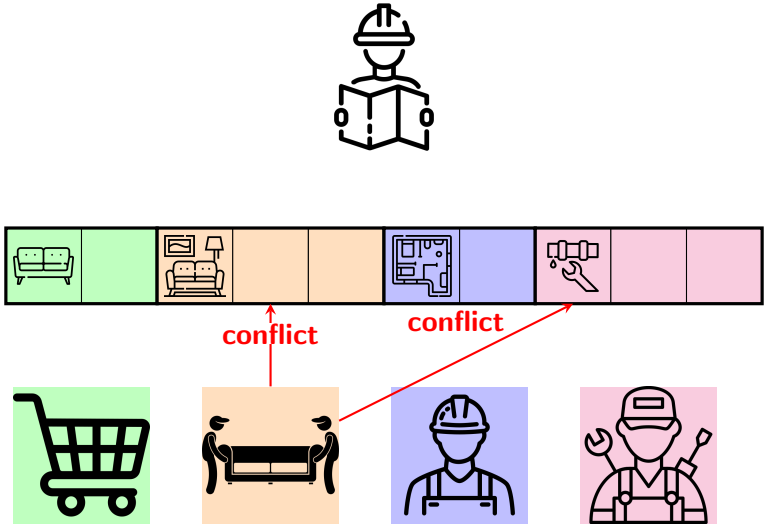


Introduction

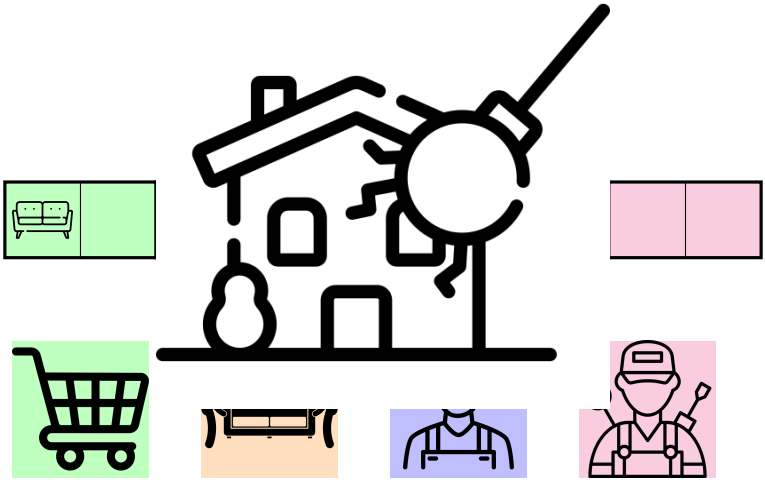




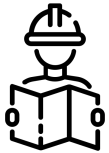
Introduction

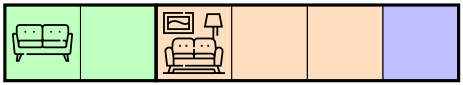


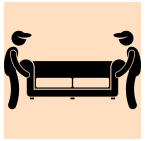
Introduction



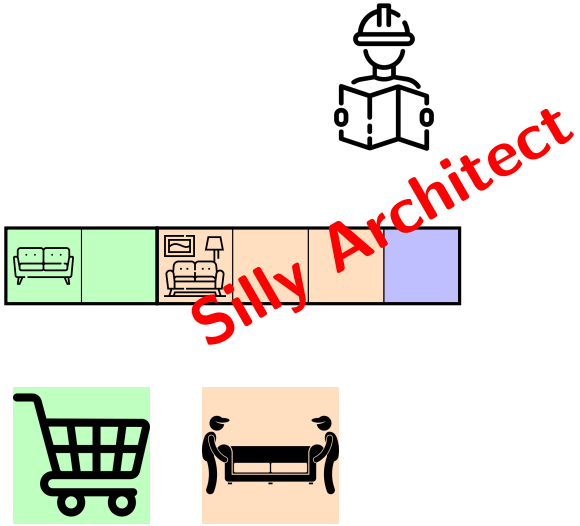
Introduction



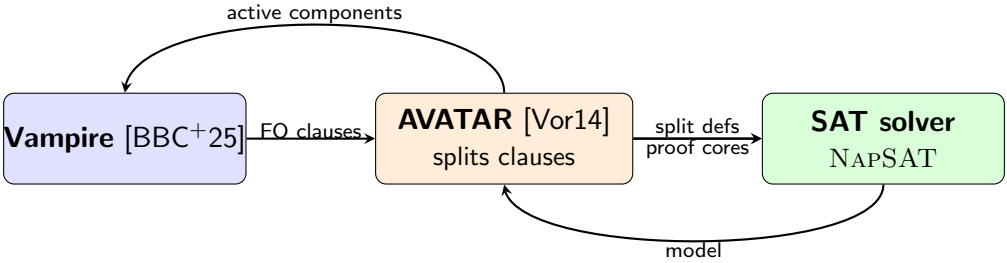




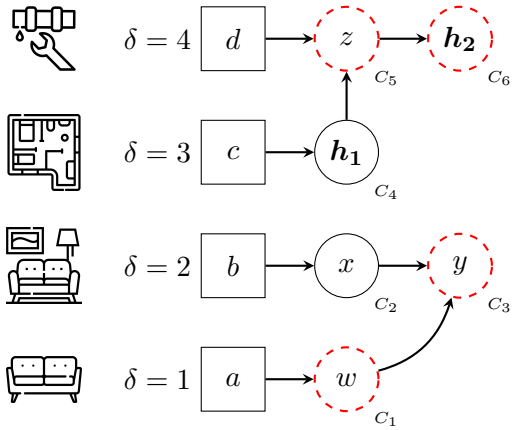
Introduction



Vampire, AVATAR, and the SAT Solver

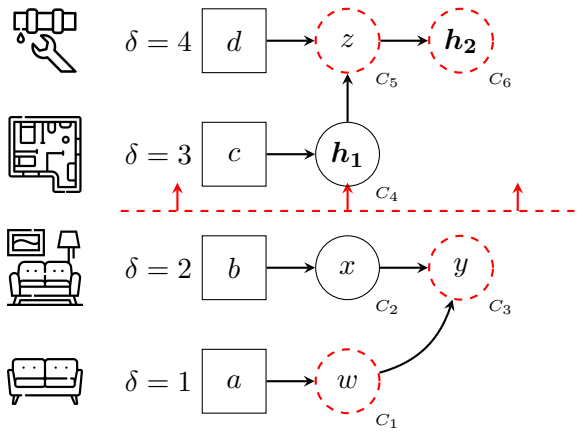


Non-Chronological Backtracking [SS99, MMZ+01]



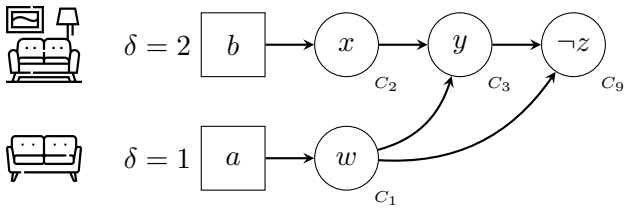
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Non-Chronological Backtracking [SS99, MMZ+01]



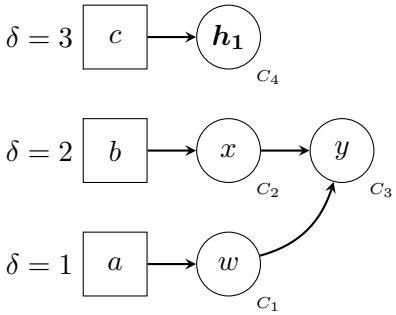
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg y$$

Non-Chronological Backtracking [SS99, MMZ+01]



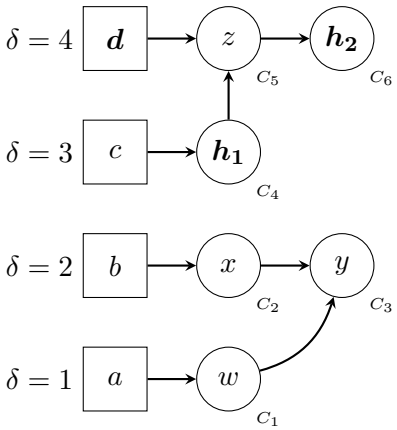
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg y$$

Incremental Solving: Adding a Clause Mid-Search



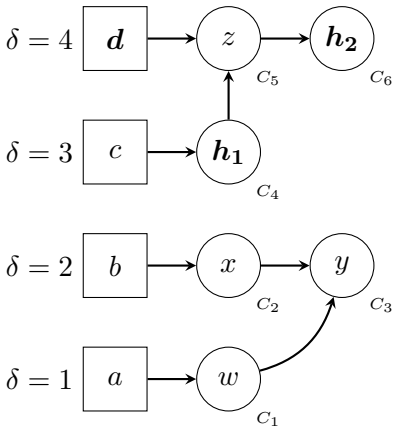
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$

Incremental Solving: Adding a Clause Mid-Search



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$

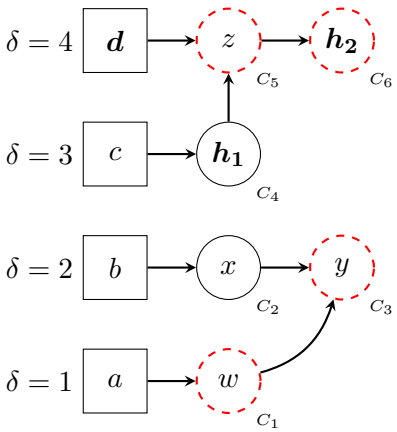
Incremental Solving: Adding a Clause Mid-Search



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = \neg a \vee \neg x \vee d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Uh oh...

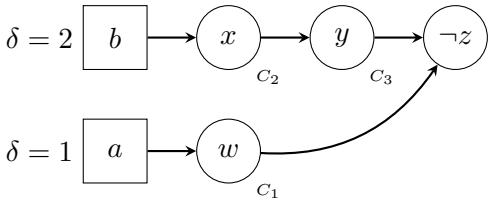
Incremental Solving: Adding a Clause Mid-Search



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = \neg a \vee \neg x \vee d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg y$$

Uh oh...

Incremental Solving: Adding a Clause Mid-Search



$C_1 = w \vee \neg a$

$C_2 = x \vee \neg b$

$C_3 = y \vee \neg w \vee \neg x$

$C_4 = h_1 \vee \neg c$

$C_5 = z \vee \neg d \vee \neg h_1$

$C_6 = h_2 \vee \neg z$

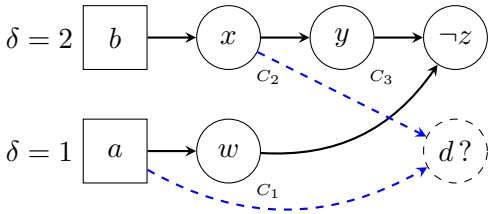
$C_7 = \neg a \vee \neg x \vee d$

$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$

$C_9 = \neg z \vee \neg w \vee \neg y$

Uh oh...

Incremental Solving: Adding a Clause Mid-Search



$C_1 = w \vee \neg a$

$C_2 = x \vee \neg b$

$C_3 = y \vee \neg w \vee \neg x$

$C_4 = h_1 \vee \neg c$

$C_5 = z \vee \neg d \vee \neg h_1$

$C_6 = h_2 \vee \neg z$

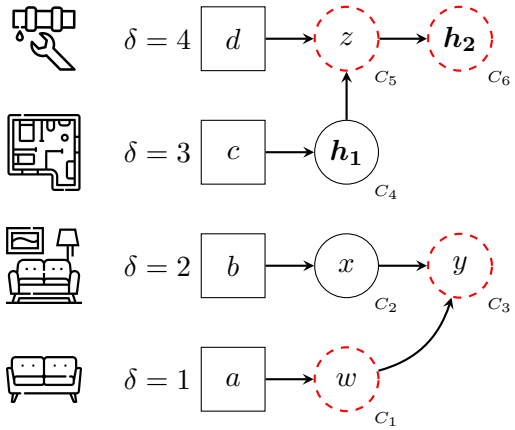
$C_7 = \neg a \vee \neg x \vee d$

$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$

$C_9 = \neg z \vee \neg w \vee \neg y$

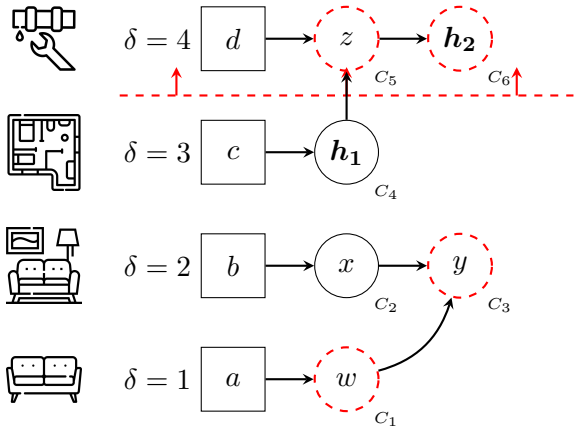
Uh oh...

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



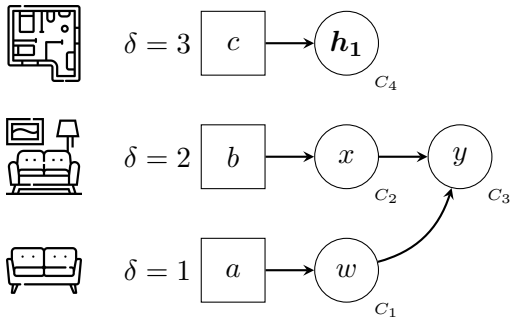
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



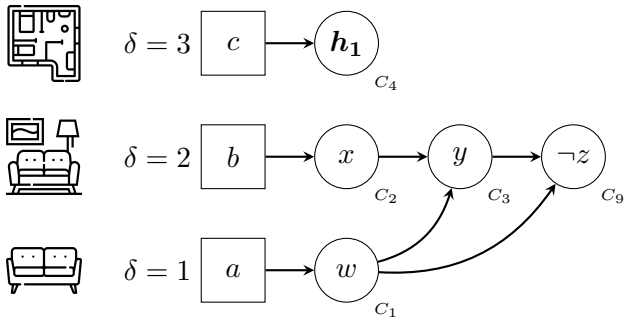
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg y$$

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg y$$

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg y$$

10617 TPTP Problems Using AVATAR - NCB vs CB

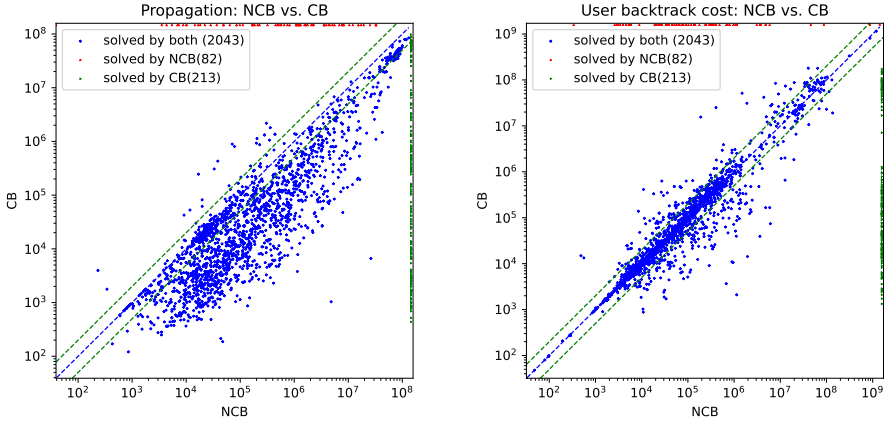
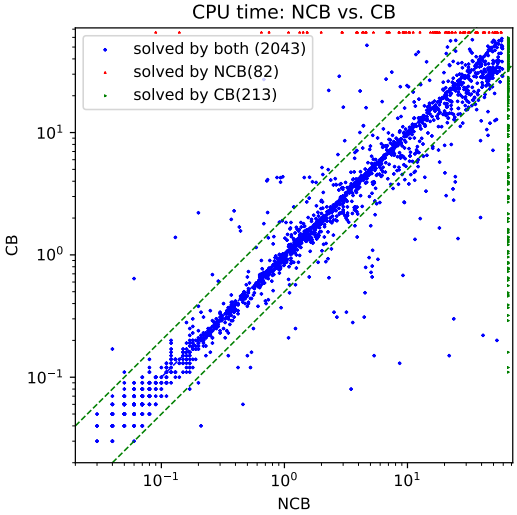


Figure: CB saves 74.3% of Boolean propagations and 3.4% of the estimated cost compared to NCB

10617 TPTP Problems Using AVATAR - NCB vs CB



What is Chronological Backtracking Good for?

Advantages

- Simple(ish) to implement.
- Resilient in incremental solving.
- Fast in practice for many problems.

What is Chronological Backtracking Good for?

Advantages

- Simple(ish) to implement.
- Resilient in incremental solving.
- Fast in practice for many problems.

Disadvantages

- User-agnostic
- Rigid: always top down

A More Tailored SAT Solver

Extended Interface

We allow the user (e.g. VAMPIRE) to provide a **cost** for each literal.

A More Tailored SAT Solver

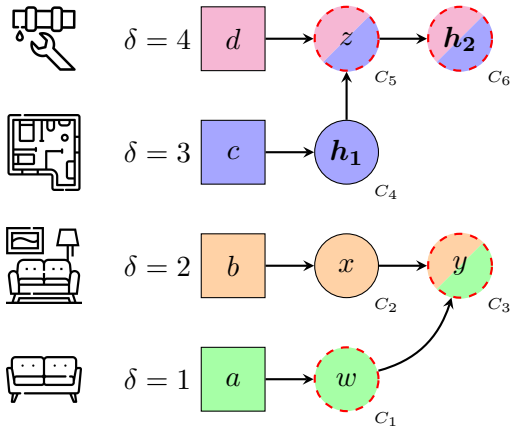
Extended Interface

We allow the user (e.g. VAMPIRE) to provide a **cost** for each literal.

New Backtracking Scheme

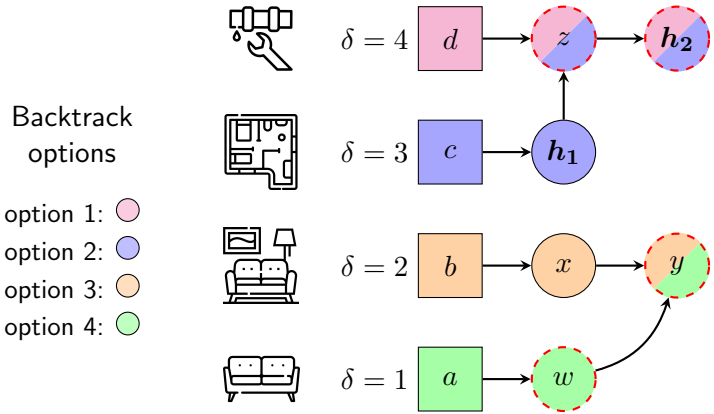
Graph Backtracking (GB) searches the cheapest set of literals to backtrack to resolve a conflict. We use the implication graph as a precise guiding structure.

Graph Backtracking (GB)

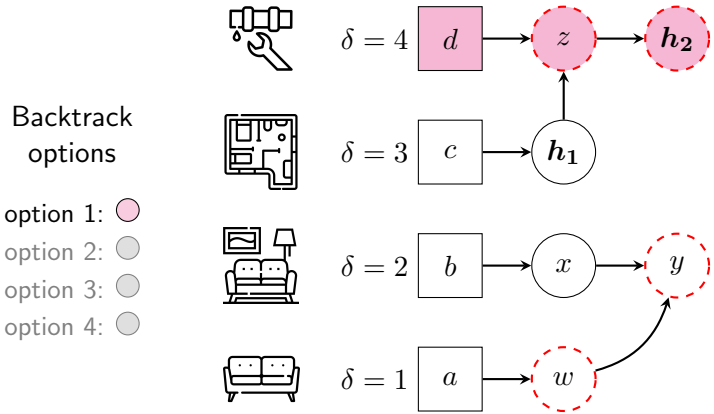


$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

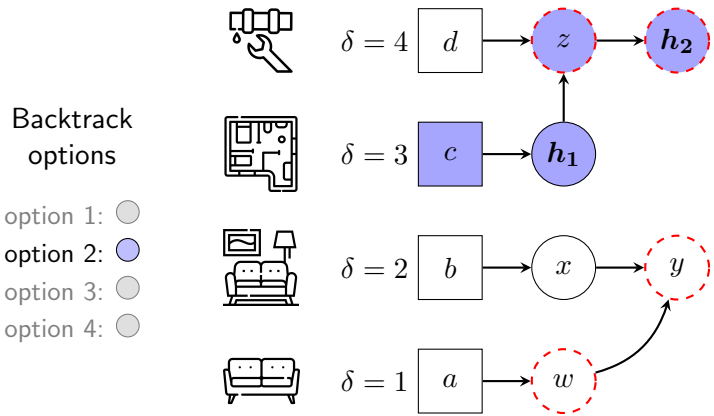
Graph Backtracking (GB)



Graph Backtracking (GB)



Graph Backtracking (GB)



Graph Backtracking (GB)

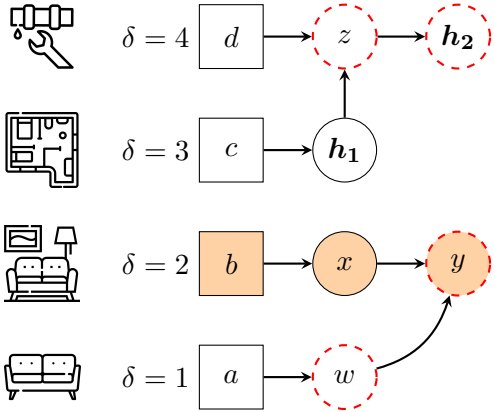
Backtrack options

option 1:

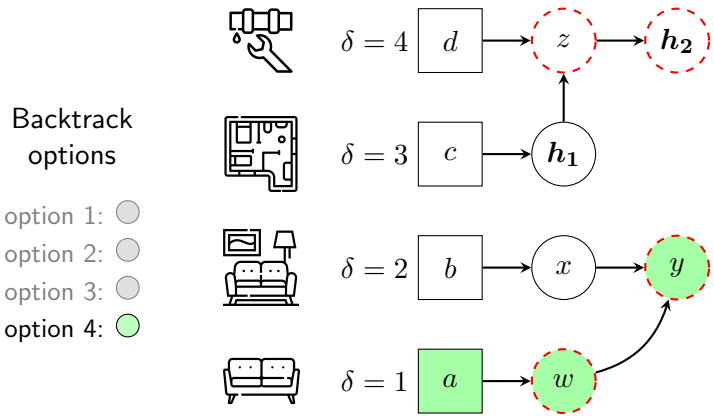
option 2:

option 3:

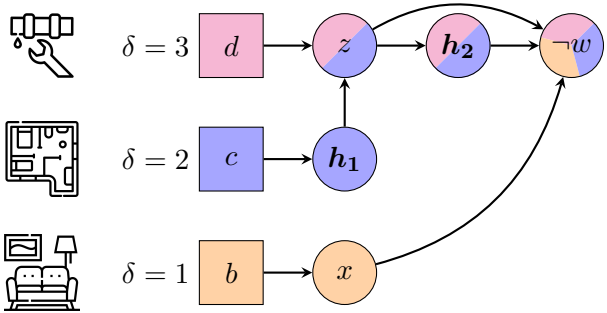
option 4:



Graph Backtracking (GB)



Graph Backtracking (GB)



Graph Backtracking: What GB Buys Us

Precision

GB unassigns exactly the literals that need to move, guided by the implication graph – not the decision level.

Graph Backtracking: What GB Buys Us

Precision

GB unassigns exactly the literals that need to move, guided by the implication graph – not the decision level.

Guarantees

- Sound, complete, and terminating (with a safeguard on chunk choice)
- Generalizes NCB and CB: both are instances of GB for specific weight functions

Graph Backtracking: What GB Buys Us

Precision

GB unassigns exactly the literals that need to move, guided by the implication graph – not the decision level.

Guarantees

- Sound, complete, and terminating (with a safeguard on chunk choice)
- Generalizes NCB and CB: both are instances of GB for specific weight functions

User Control

The user-provided cost function allows for a more flexible backtracking strategy.

10617 TPTP Problems Using AVATAR - (N)CB vs GB v1

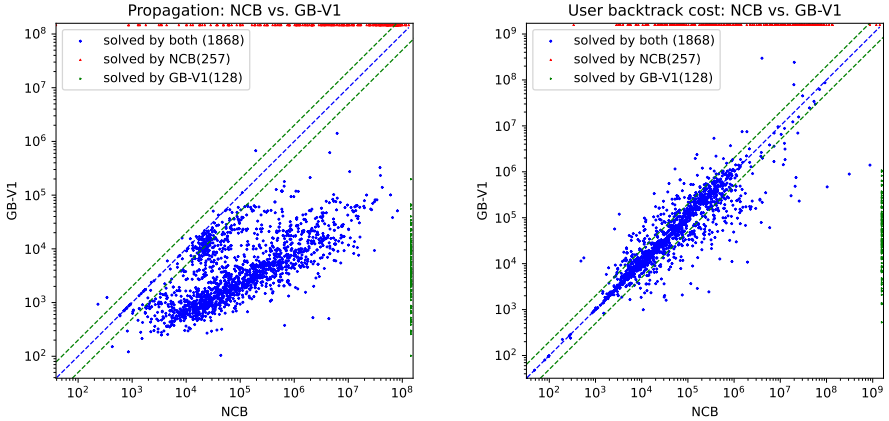


Figure: GB saves 98.7% of Boolean propagations and 70.3% of the estimated cost compared to NCB

10617 TPTP Problems Using AVATAR - (N)CB vs GB v1

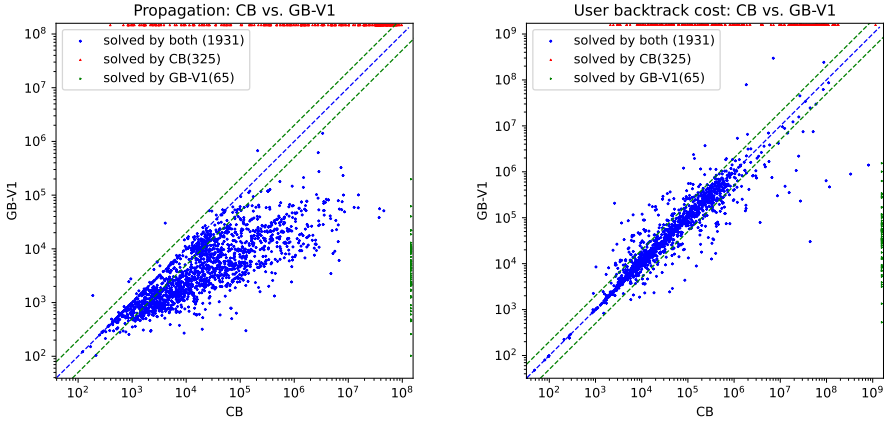
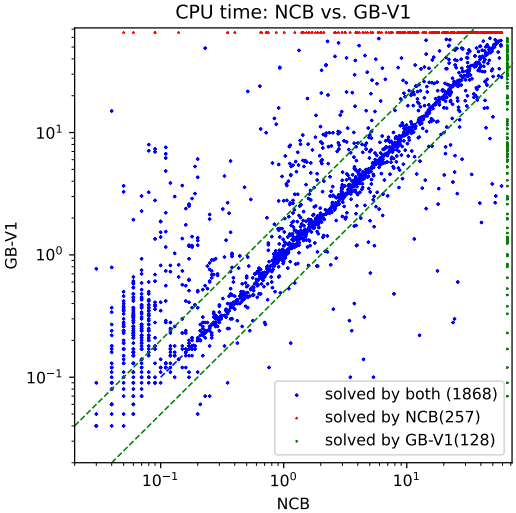
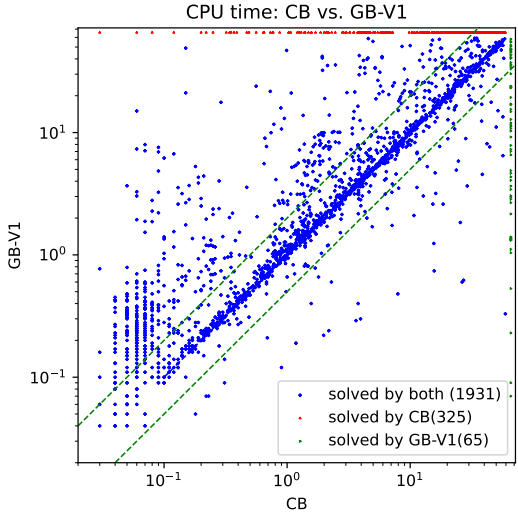


Figure: GB saves 95% of Boolean propagations and 69.3% of the estimated cost compared to CB

10617 TPTP Problems Using AVATAR - (N)CB vs GB v1





Why does it not work?

Hypothesis

1. Conflict repair too expensive?
2. Slow decision elimination?
3. Color selection too expensive?
4. Bad cost heuristic?

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Prefixes: $(\emptyset, 4)$

Backtracking options:

Many Conflicts to Solve at Once

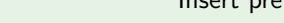
Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Backtracking options:

Prefixes: $\{\ \}$

pick prefix $\emptyset$. C_1, C_2, C_3, C_4 still needs to be solved.

Insert prefixes 

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Prefixes: $\{(\bigcirc, 1), (\bigcirc, 2), (\bigcirc, 2), (\bigcirc, 2)\}$

Backtracking options:

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Backtracking options:  ,

Prefixes: $\{(\bigcirc, 2), (\bigcirc, 2), (\bigcirc, 2)\}$
pick prefix $\{\bigcirc, 1\}$. C_3 still needs to be solved.
Commit hitting sets  ,

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Prefixes: $\{($$, 2), ($$, 2), ($$, 2)\}$

Backtracking options:  ,  ,

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Backtracking options:  ,  ,

Prefixes: $\{($  $, 2), ($$, 2)\}$
pick prefix $\{($$, 2\}$. C_2, C_4 still needs to be solved.
Subsumed 
Add prefixes $($  $, 1), ($$, 1)$

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Prefixes: $\{($$, 1), ($$, 1), ($$, 2), ($$, 2)\}$

Backtracking options:  ,  ,

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Backtracking options:  ,  ,

Prefixes: $\{($$, 1), ($$, 2), ($$, 2)\}$
pick prefix  . C_4 still needs to be solved.
Subsume .
Commit  .

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Backtracking options:  ,  ,  ,

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

Backtracking options:  ,  , 

Many Conflicts to Solve at Once

Example: four conflicts over our four colors

conflict	colors hit
C_1	
C_2	
C_3	
C_4	

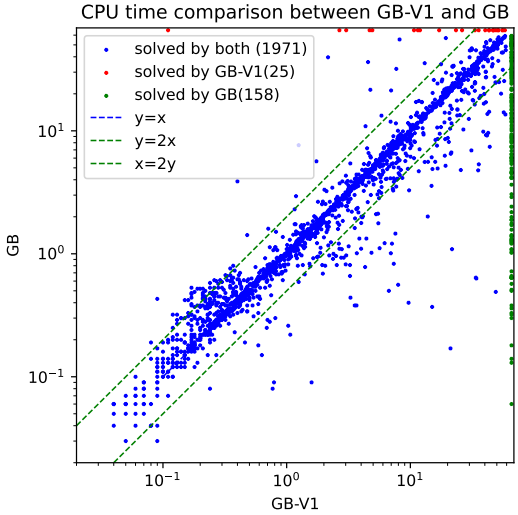
Backtracking options:  ,  , 

Observation

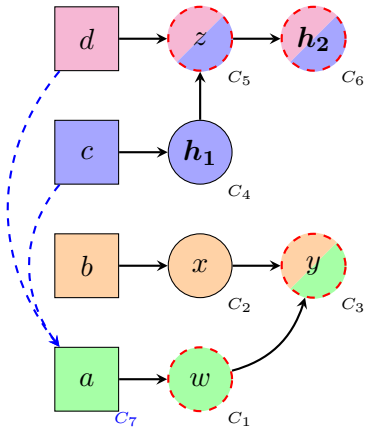
The naive approach of enumerating all hitting sets is too expensive. We need a more efficient search.

We capped the number of conflicts to 10 and the number of hitting sets to 1000.

10617 TPTP Problems Using AVATAR - GB v1 vs Better GB

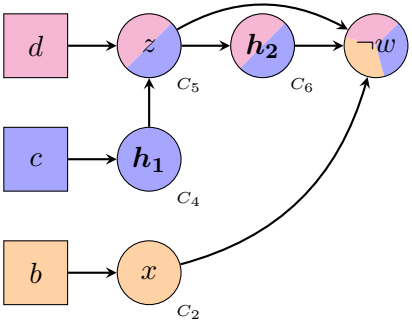


Color Merging



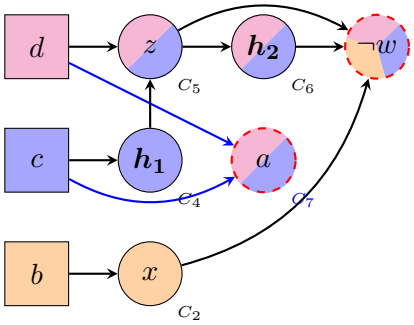
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Color Merging



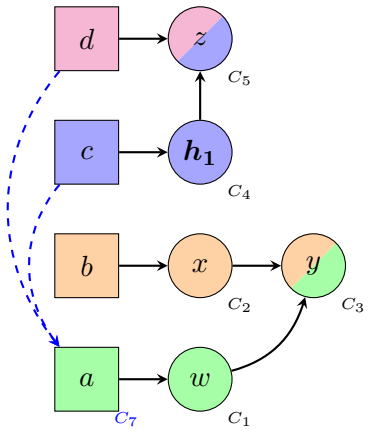
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9' = \neg w \vee \neg x \vee \neg z \vee \neg h_2$$

Color Merging



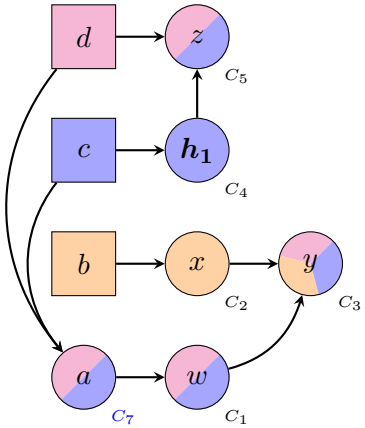
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9' = \neg w \vee \neg x \vee \neg z \vee \neg h_2$$

Eager Color Merging (ECM)



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

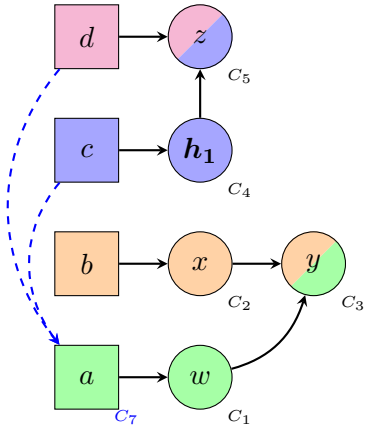
Eager Color Merging (ECM)



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Lazy Color Merging (LCM)

Backtrack options



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Lazy Color Merging (LCM)

Backtrack options

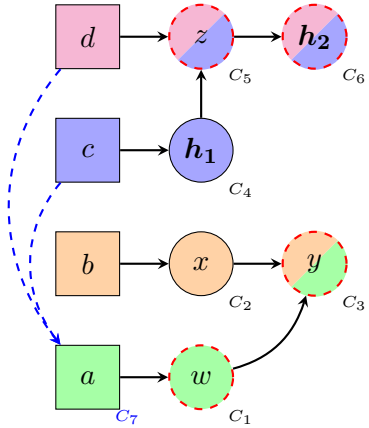
option 1:

option 2:

option 3:

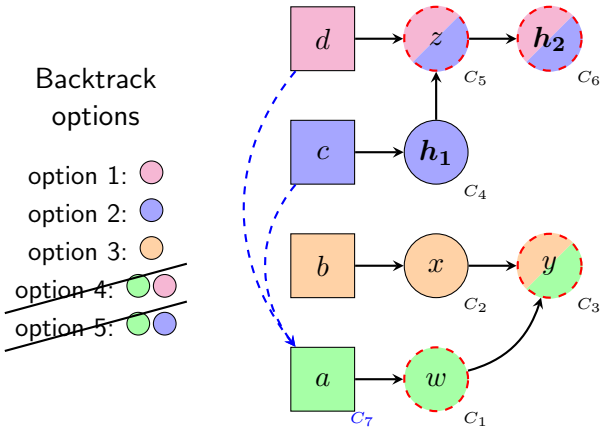
option 4:

option 5:



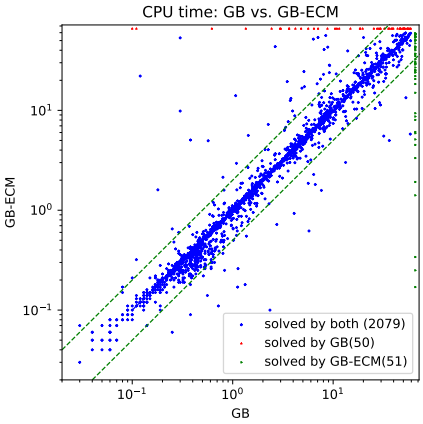
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

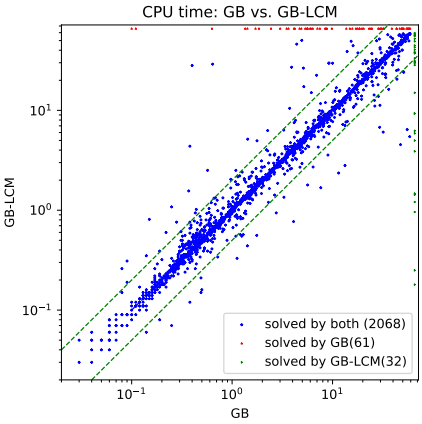
Lazy Color Merging (LCM)



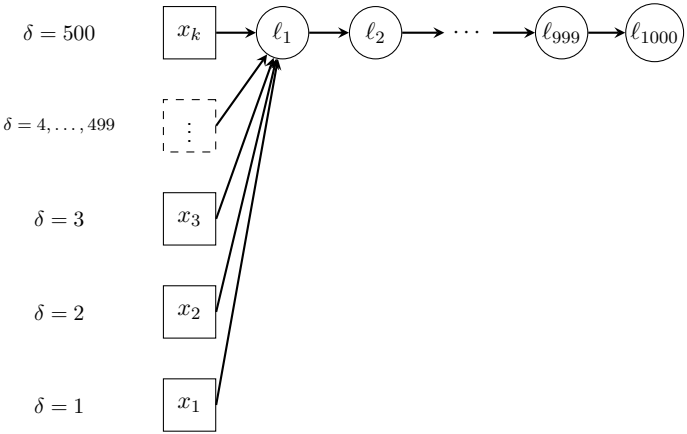
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

10617 TPTP Problems Using AVATAR - GB vs GB-Merging

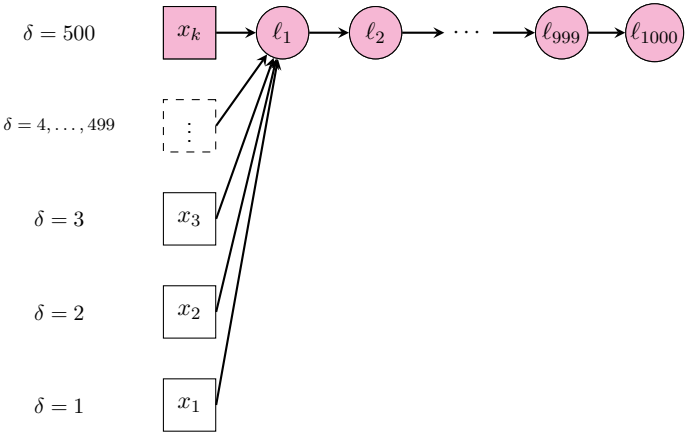




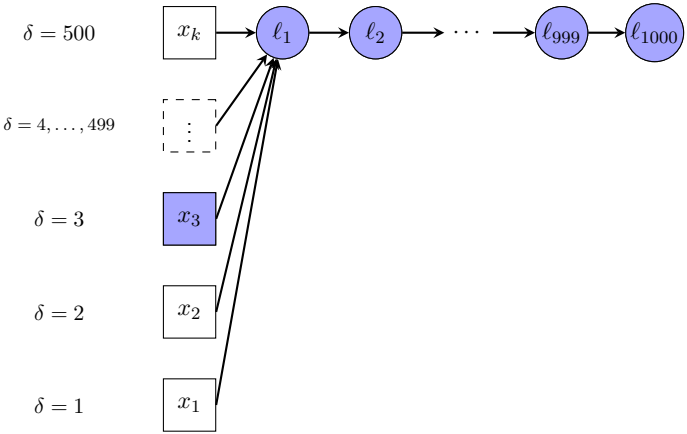
A Long Chain from Multiple Decisions



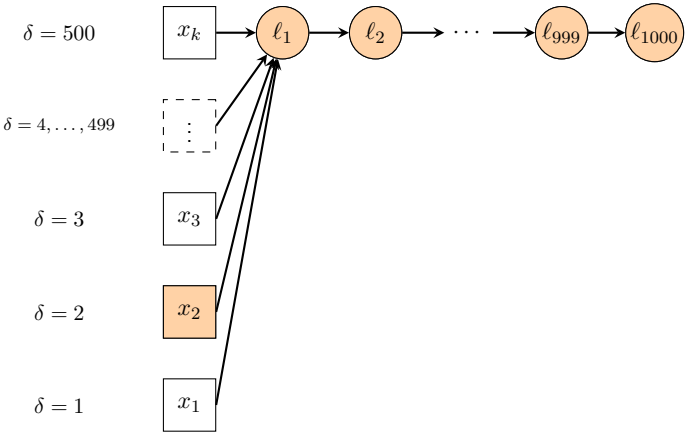
A Long Chain from Multiple Decisions



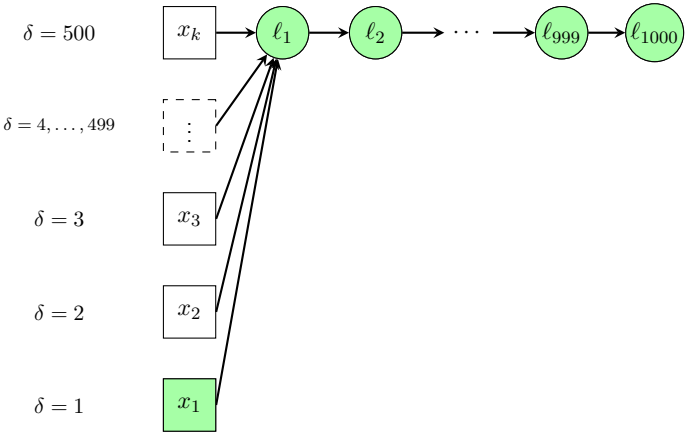
A Long Chain from Multiple Decisions



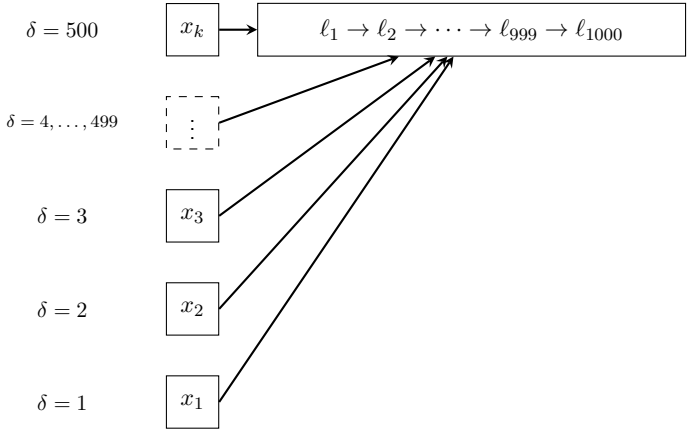
A Long Chain from Multiple Decisions



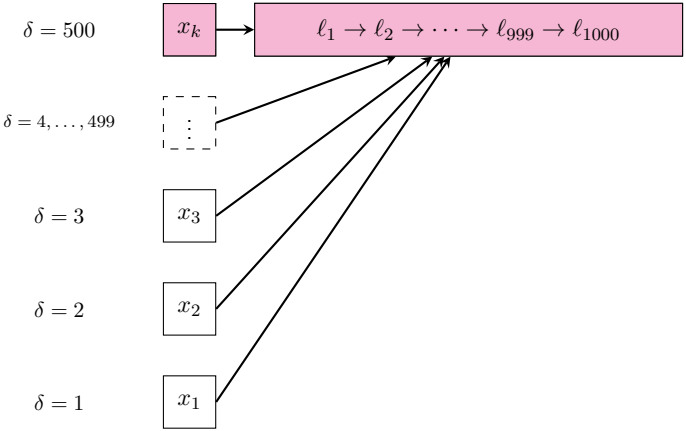
A Long Chain from Multiple Decisions



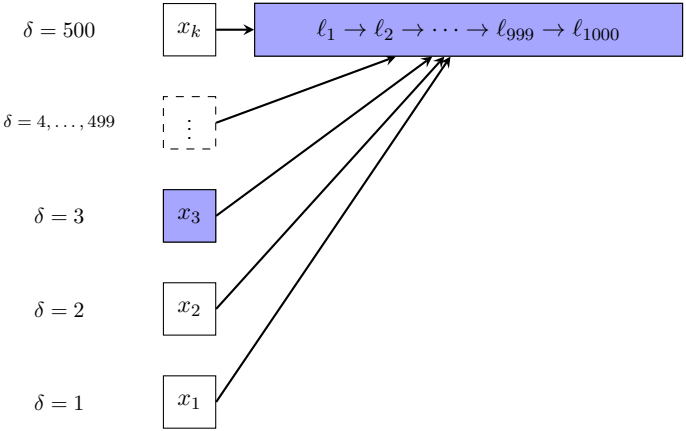
Trail Compaction



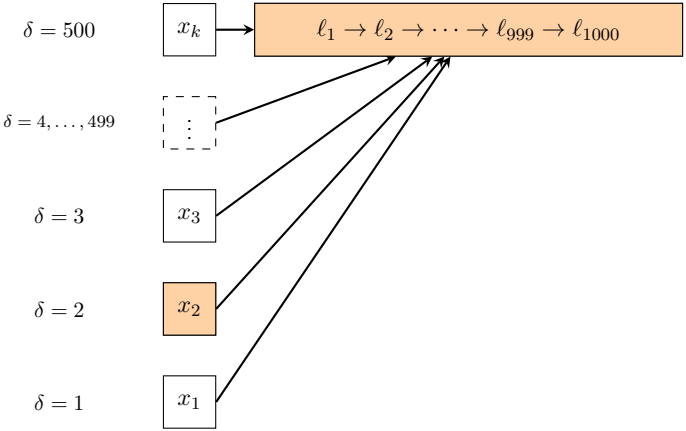
Trail Compaction



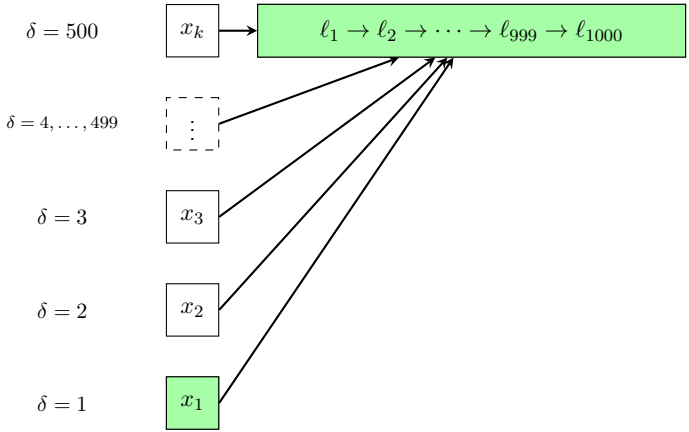
Trail Compaction



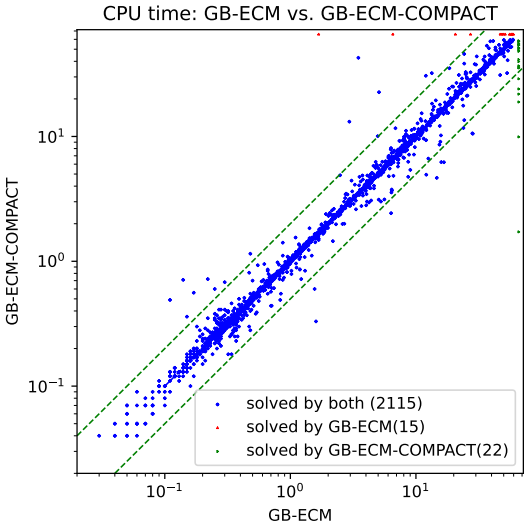
Trail Compaction



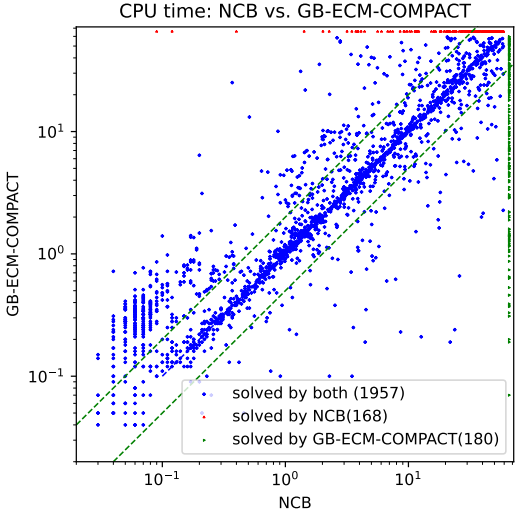
Trail Compaction

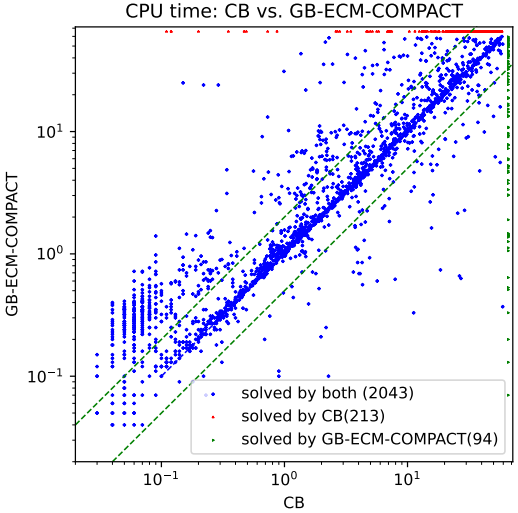


10617 TPTP Problems Using AVATAR - GB-ECM vs Compacted GB-ECM



10617 TPTP Problems Using AVATAR - (N)CB vs our best GB





The Elephant in the Room: Cost Estimation

Current Heuristic

$$\text{cost}(\ell) = \begin{cases} \#\text{activated clauses} + \#\text{deactivated clauses} & \text{if } \ell \text{ is synchronized} \\ 0 \quad \text{otherwise} \end{cases}$$

The Elephant in the Room: Cost Estimation

Current Heuristic

$$\text{cost}(\ell) = \begin{cases} \#\text{activated clauses} + \#\text{deactivated clauses} & \text{if } \ell \text{ is synchronized} \\ 0 \quad \text{otherwise} \end{cases}$$

Research Question (How to improve the cost heuristic?)

We would love your ideas.

Conclusion

Conclusion

- GB solves roughly as many problems as NCB – **competitive** in CPU time.
- GB is much more **precise**: far fewer Boolean propagations than NCB or CB.
- GB succeeds in **preserving expensive literals**, at the cost of book-keeping.

Conclusion

Conclusion

- GB solves roughly as many problems as NCB – **competitive** in CPU time.
- GB is much more **precise**: far fewer Boolean propagations than NCB or CB.
- GB succeeds in **preserving expensive literals**, at the cost of book-keeping.

Future Work

- Improve **scalability** in the number of decisions/conflicts.
- Improve the **cost heuristic**.
- Reduce the **conflict repair** overhead.

References I

Filip Bártek, Ahmed Bhayat, Robin Coutelier, Márton Hajdú, Matthias Hetzenberger, Petra Hozzová, Laura Kovács, Jakob Rath, Michael Rawson, Giles Reger, Martin Suda, Johannes Schoisswohl, and Andrei Voronkov.

The vampire diary.

In *CAV (3)*, volume 15933 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2025.

Robin Coutelier, Mathias Fleury, and Laura Kovács.

Lazy reimplication in chronological backtracking.

In *SAT*, volume 305 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

References II

Flaticon.

Free icons https://www.flaticon.com/free-icons/.

Accessed: 2026-01-10.

Sibylle Möhle and Armin Biere.

Backing backtracking.

In *SAT*, volume 11628 of *Lecture Notes in Computer Science*, pages 250–266. Springer, 2019.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.

Chaff: Engineering an efficient SAT solver.

In *DAC*, pages 530–535. ACM, 2001.

References III

Alexander Nadel.

Introducing intel(r) SAT solver.

In *SAT*, volume 236 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Alexander Nadel and Vadim Ryvchin.

Chronological backtracking.

In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, pages 111–121. Springer, 2018.

References IV

João P. Marques Silva and Karem A. Sakallah.

GRASP: A search algorithm for propositional satisfiability.

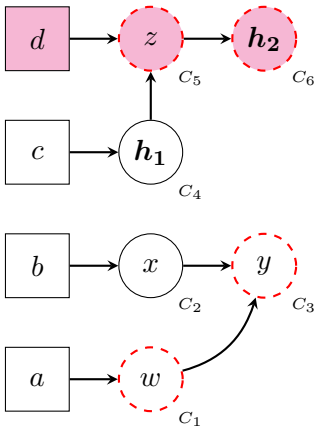
IEEE Trans. Computers, 48(5):506–521, 1999.

Andrei Voronkov.

AVATAR: the architecture for first-order theorem provers.

In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer, 2014.

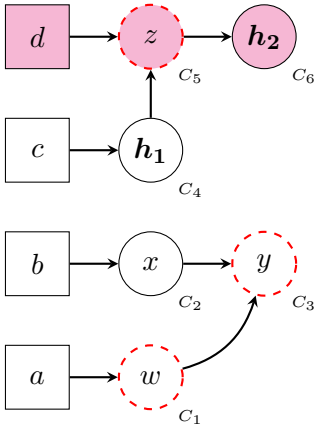
First Unique Implication Point (UIP) with GB



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB

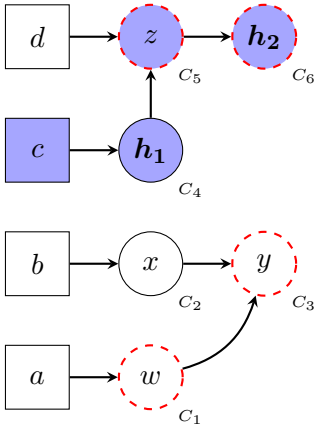
$\neg w \vee \neg y \vee \neg z$



$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB

$\neg w \vee \neg y \vee \neg z$

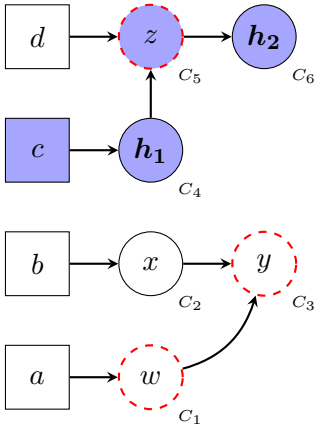


$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z$



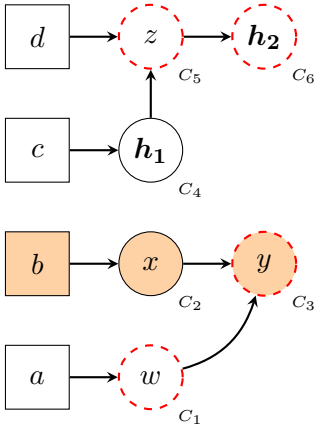
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z \vee \neg h_2$



$C_1 = w \vee \neg a$

$C_2 = x \vee \neg b$

$C_3 = y \vee \neg w \vee \neg x$

$C_4 = h_1 \vee \neg c$

$C_5 = z \vee \neg d \vee \neg h_1$

$C_6 = h_2 \vee \neg z$

$C_7 = a \vee \neg c \vee \neg d$

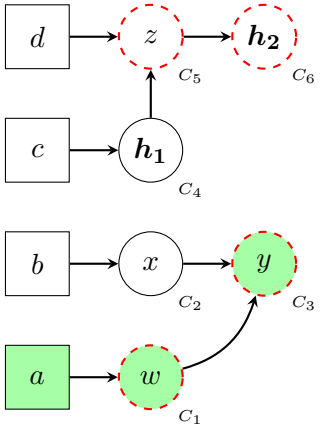
$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$

First Unique Implication Point (UIP) with GB

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z \vee \neg h_2$



$C_1 = w \vee \neg a$

$C_2 = x \vee \neg b$

$C_3 = y \vee \neg w \vee \neg x$

$C_4 = h_1 \vee \neg c$

$C_5 = z \vee \neg d \vee \neg h_1$

$C_6 = h_2 \vee \neg z$

$C_7 = a \vee \neg c \vee \neg d$

$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$

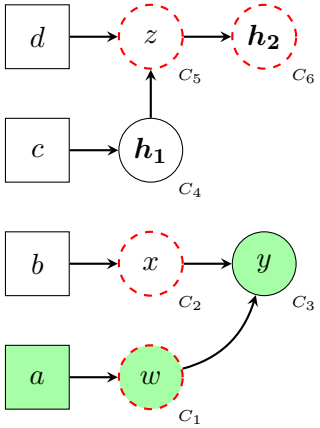
First Unique Implication Point (UIP) with GB

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z$

$\neg w \vee \neg y \vee \neg z \vee \neg h_2$

$\neg w \vee \neg x \vee \neg z \vee \neg h_2$



$C_1 = w \vee \neg a$

$C_2 = x \vee \neg b$

$C_3 = y \vee \neg w \vee \neg x$

$C_4 = h_1 \vee \neg c$

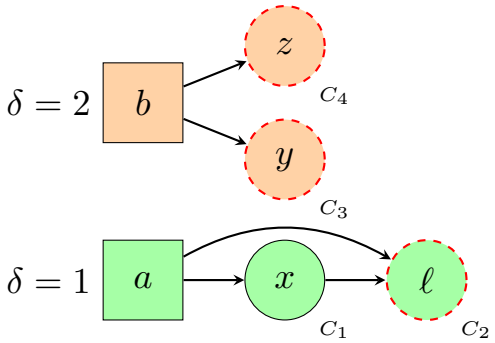
$C_5 = z \vee \neg d \vee \neg h_1$

$C_6 = h_2 \vee \neg z$

$C_7 = a \vee \neg c \vee \neg d$

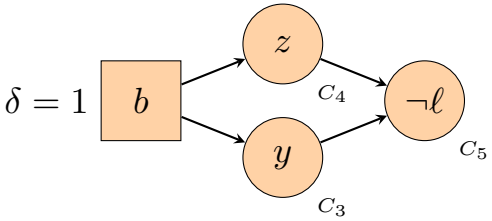
$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$

Challenge I: Termination



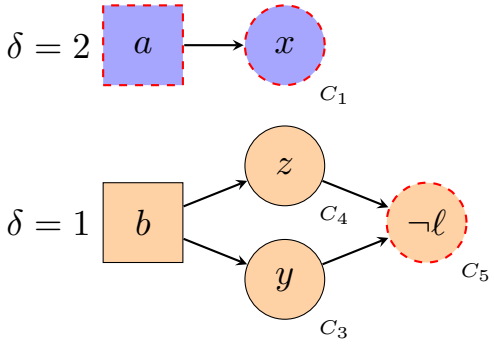
$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= z \vee \neg b \\
C_5 &= \neg \ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Termination



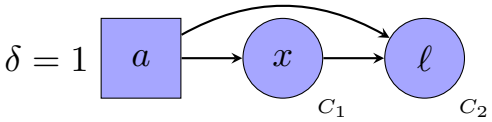
$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= z \vee \neg b \\
C_5 &= \neg\ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Termination



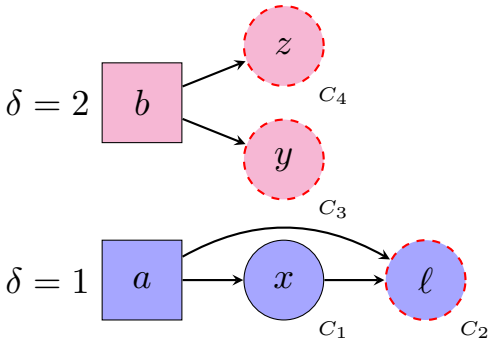
$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= y \vee \neg b \\
C_5 &= \neg\ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Termination



$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= z \vee \neg b \\
C_5 &= \neg \ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Termination



$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= z \vee \neg b \\
C_5 &= \neg \ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Safeguards

Why this happens

- Arbitrary cost function
- No guarantee to learn a new clause after backtracking (like in CB [MB19])

Challenge I: Safeguards

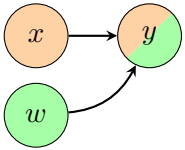
Why this happens

- Arbitrary cost function
- No guarantee to learn a new clause after backtracking (like in CB [MB19])

Solution

- If possible to learn a clause: choose the color that allows to learn the clause with minimum total cost
- Otherwise, select latest decision (like in CB)

Challenge II: Watched Literals



Let us watch y and $\neg x$ in C_3.

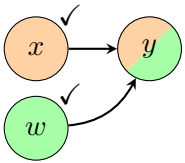
$$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$$

Invariant (Watched Literals)

For each clause $C \in F$ watched by c_1, c_2, we have

$$\textit{propagated}(\neg c_1) \Rightarrow \textit{satisfied}(c_2)$$

Challenge II: Watched Literals



$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$

Let us watch y and $\neg x$ in C_3.

$$\text{propagated}(\neg y) \Rightarrow \text{satisfied}(\neg x)$$
$$\text{propagated}(x) \Rightarrow \text{satisfied}(y)$$

is satisfied.

Invariant (Watched Literals)

For each clause $C \in F$ watched by c_1, c_2, we have

$$\textit{propagated}(\neg c_1) \Rightarrow \textit{satisfied}(c_2)$$

Challenge II: Watched Literals

Let us watch y and $\neg x$ in C_3.

$$\text{propagated}(\neg y) \Rightarrow \text{satisfied}(\neg x)$$
$$\text{propagated}(x) \Rightarrow \text{satisfied}(y)$$

is violated

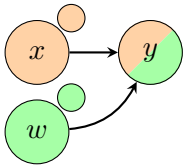
$$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$$

Invariant (Watched Literals)

For each clause $C \in F$ watched by c_1, c_2, we have

$$\textit{propagated}(\neg c_1) \Rightarrow \textit{satisfied}(c_2)$$

Challenge II: Watched Literals (Solution)



We define cross-colors(ℓ) as the set of colors that trigger the repropagation of ℓ.

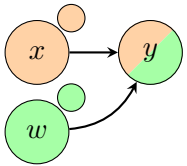
$$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$$

Invariant (Watched Literals)

For each clause $C \in F$ watched by c_1, c_2, we have

$$\mathit{propagated}(\neg c_1) \Rightarrow \left[\mathit{satisfied}(c_2) \wedge \mathit{colors}(c_2) \subseteq \mathit{cross\text{-}colors}(c_1)\right]$$

Challenge II: Watched Literals (Solution)



$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$

Let us watch y and $\neg x$ in C_3.

propagated$(\neg y) \Rightarrow$ [satisfied$(\neg x) \wedge$ ◯ $\subseteq$ ◔]

propagated$(x) \Rightarrow$ [satisfied$(y) \wedge$ ◔ $\subseteq$ ◯]

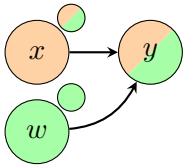
is satisfied.

Invariant (Watched Literals)

For each clause $C \in F$ watched by c_1, c_2, we have

$$\textit{propagated}(\neg c_1) \Rightarrow \left[\textit{satisfied}(c_2) \wedge \textit{colors}(c_2) \subseteq \textit{cross-colors}(c_1)\right]$$

Challenge II: Watched Literals (Solution)



$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$

Let us watch y and $\neg x$ in C_3.

propagated$(\neg y) \Rightarrow$ [satisfied$(\neg x) \wedge$  $\subseteq$ ]

propagated$(x) \Rightarrow$ [satisfied$(y) \wedge$ $\subseteq$ ]

is satisfied.

Invariant (Watched Literals)

For each clause $C \in F$ watched by c_1, c_2, we have

$$\textit{propagated}(\neg c_1) \Rightarrow \left[\textit{satisfied}(c_2) \wedge \textit{colors}(c_2) \subseteq \textit{cross-colors}(c_1)\right]$$

Backup: The Hitting-Set Search

Given

Conflicting clauses $C_1, \ldots, C_n$ discovered simultaneously, each with a candidate chunk set $\gamma(C_i)$.

Want

A cheapest Γ with $\Gamma \cap \gamma(C_i) \neq \emptyset$ for every i.

Algorithm sketch

- Seed a shortest/greedy hitting set to bound the search early.
- Best-first search over partial hitting sets, ranked by their (bounded) cost.
- Prune a branch once no completion of its prefix can beat the current best; deduplicate hitting sets discovered via different branch orders.
- Cap the number of candidates explored (Section 6.1 of the paper) to bound worst-case blow-up.

Backup: Trail Compaction, in Detail

Compacted trail

Scan the trail once; whenever a run of consecutive literals shares *exactly* the same chunk membership, replace the run by a single piece $(\text{chunks}, \text{literals})$.

Where it is used

Only for costing candidate chunk sets during conflict repair – not for propagation, where per-literal chunk information is still needed.

When it helps

Long runs of binary, single-chunk implications (deep AVATAR component chains). When chunks fragment quickly (many merges, many decisions touching the same region), the compacted trail degenerates back to one piece per literal.

References I

Filip Bártek, Ahmed Bhayat, Robin Coutelier, Márton Hajdú, Matthias Hetzenberger, Petra Hozzová, Laura Kovács, Jakob Rath, Michael Rawson, Giles Reger, Martin Suda, Johannes Schoisswohl, and Andrei Voronkov.

The vampire diary.

In *CAV (3)*, volume 15933 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2025.

Robin Coutelier, Mathias Fleury, and Laura Kovács.

Lazy reimplication in chronological backtracking.

In *SAT*, volume 305 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

References II

Flaticon.

Free icons https://www.flaticon.com/free-icons/.

Accessed: 2026-01-10.

Sibylle Möhle and Armin Biere.

Backing backtracking.

In *SAT*, volume 11628 of *Lecture Notes in Computer Science*, pages 250–266. Springer, 2019.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.

Chaff: Engineering an efficient SAT solver.

In *DAC*, pages 530–535. ACM, 2001.

References III

Alexander Nadel.

Introducing intel(r) SAT solver.

In *SAT*, volume 236 of *LIPIcs*, pages 8:1–8:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Alexander Nadel and Vadim Ryvchin.

Chronological backtracking.

In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, pages 111–121. Springer, 2018.

References IV

João P. Marques Silva and Karem A. Sakallah.

GRASP: A search algorithm for propositional satisfiability.

IEEE Trans. Computers, 48(5):506–521, 1999.

Andrei Voronkov.

AVATAR: the architecture for first-order theorem provers.

In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer, 2014.