Generalizing CDCL with Graph Backtracking

SAT 2026, Lisbon, Portugal

Robin Coutelier, Thomas Hader, and Laura Kovács

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

July 21st 2026





Acknowledgements

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



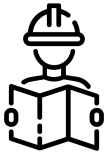




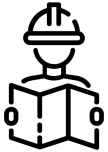


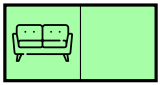


Introduction



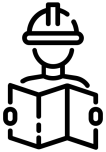
Introduction







Introduction

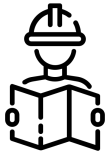


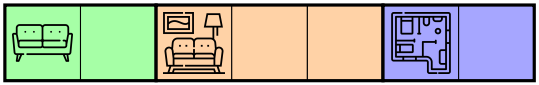




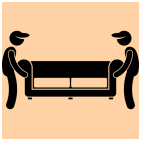


Introduction



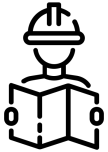


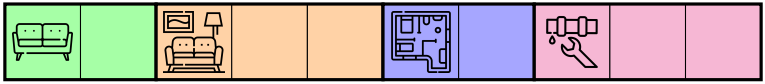




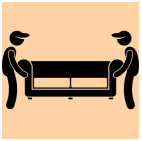


Introduction

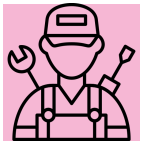




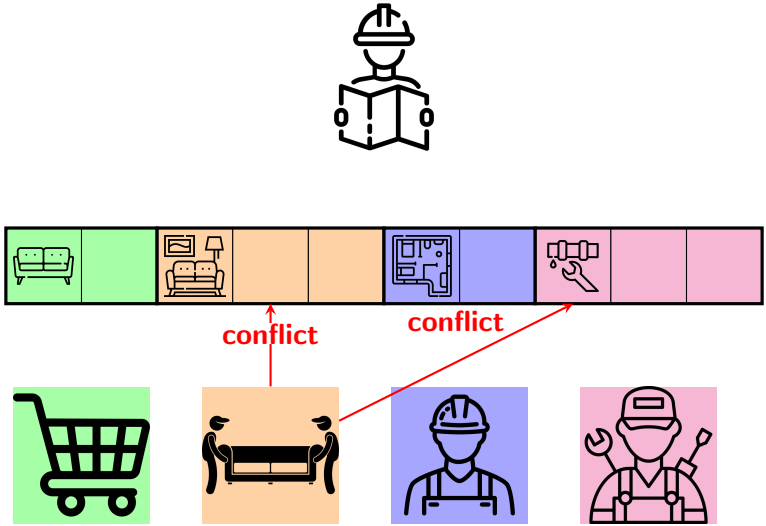




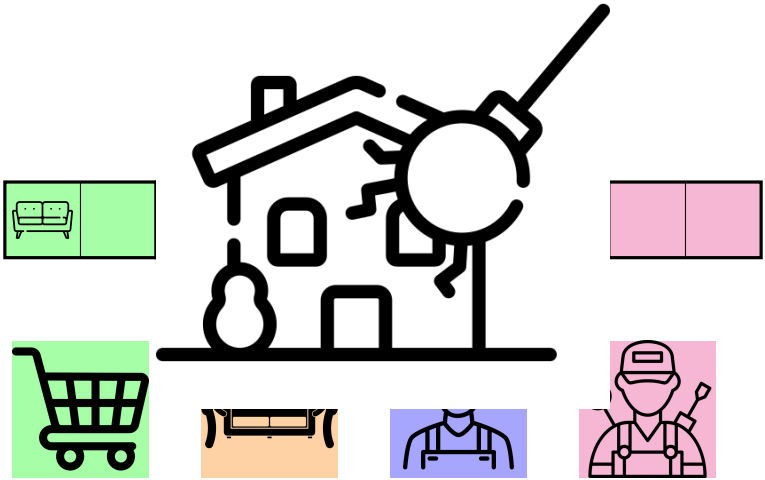




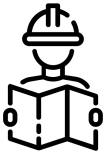
Introduction

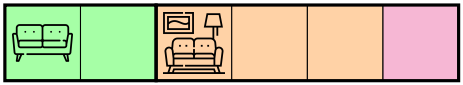


Introduction



Introduction

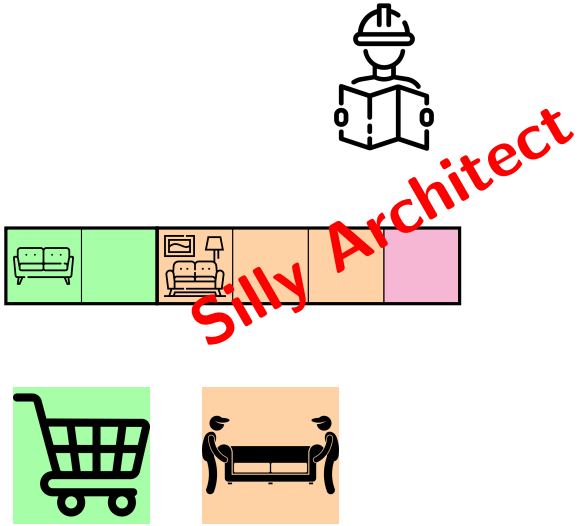






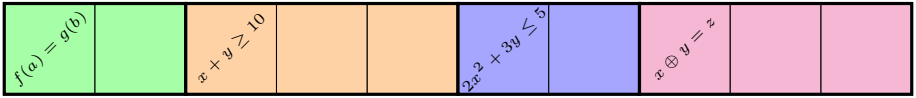


Introduction



SMT Architecture

SAT Solver



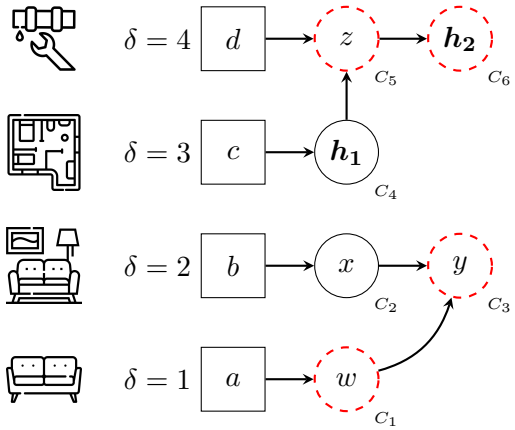
UF
(Uninterpreted Functions)

LRA
(Linear Real Arithmetic)

NRA
(Non-linear Real Arithmetic)

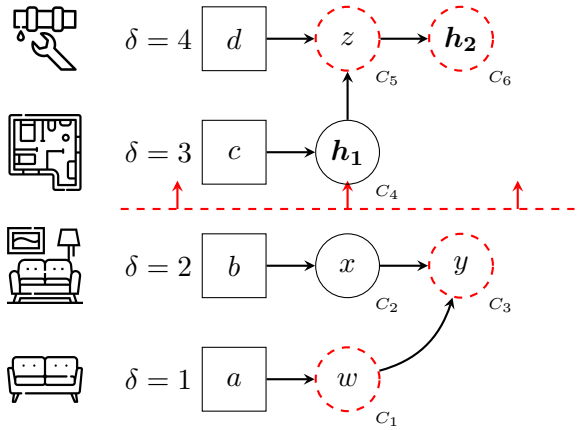
BV
(Bit Vectors)

Non-Chronological Backtracking [SS99, MMZ+01]



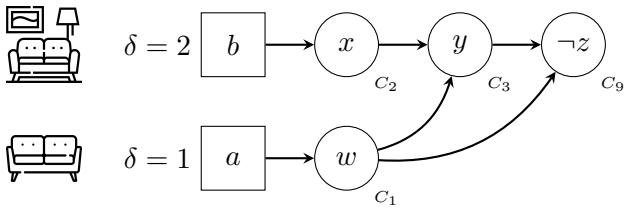
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Non-Chronological Backtracking [SS99, MMZ+01]



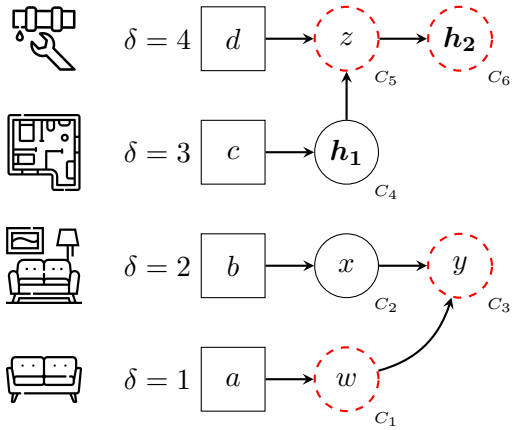
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg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

Non-Chronological Backtracking [SS99, MMZ+01]



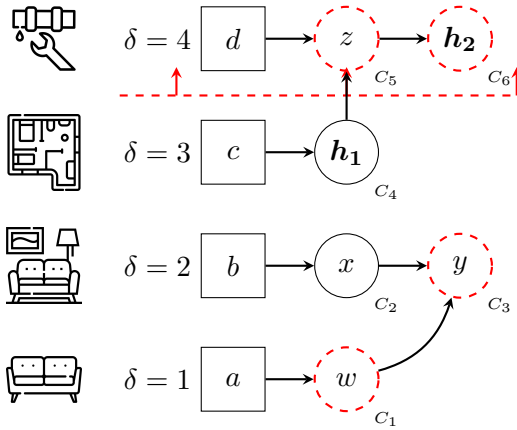
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg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

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



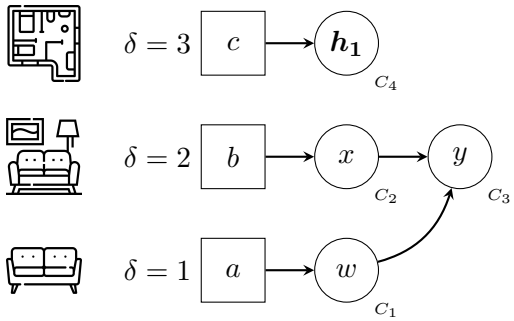
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



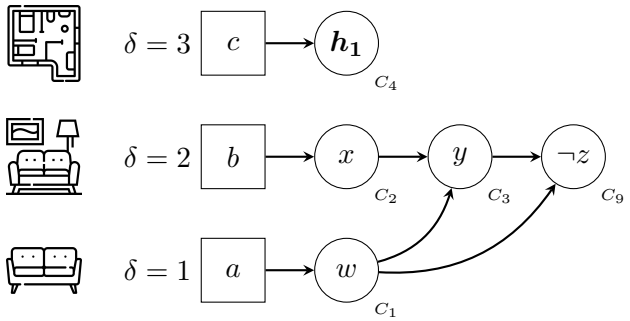
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9 = \neg z \vee \neg w \vee \neg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

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]



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

Why is the Architect a Brute?

To be fast

- NCB maintains nice **invariants**
- Propagations are cheap
- Incremental solvers often assume push-pop semantics

Why is the Architect a Brute?

To be fast

- NCB maintains nice **invariants**
- Propagations are cheap
- Incremental solvers often assume push-pop semantics

To be general

- The SAT solver does not know about literal semantics
- The cost of literals are unknown to the SAT solver

Why is the Architect a Brute?

To be fast

- NCB maintains nice **invariants**
- Propagations are cheap
- Incremental solvers often assume push-pop semantics

To be general

- The SAT solver does not know about literal semantics
- The cost of literals are unknown to the SAT solver

To be smart

- Aggressive backjumping can undo a lot of bad work
- Maybe learn fewer clauses

A More Surgical Architect

Extended Interface

We allow the user to provide a **cost** for each literal.

A More Surgical Architect

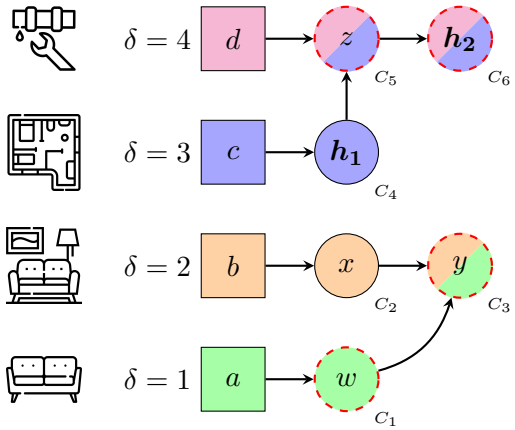
Extended Interface

We allow the user to provide a **cost** for each literal.

New Backtracking Scheme

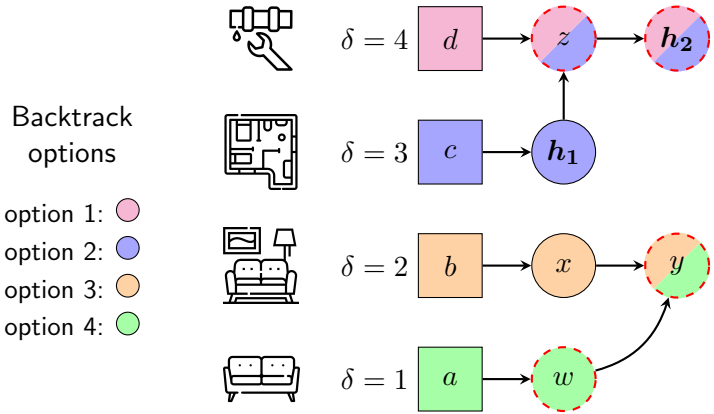
Graph Backtracking (GB) searches the cheapest set of literals to backtrack to resolve a conflict. We use the implication graph to find such a set.

Graph Backtracking (GB)

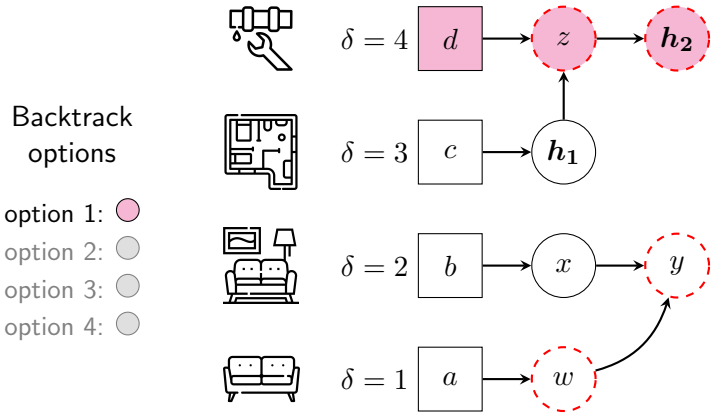


$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

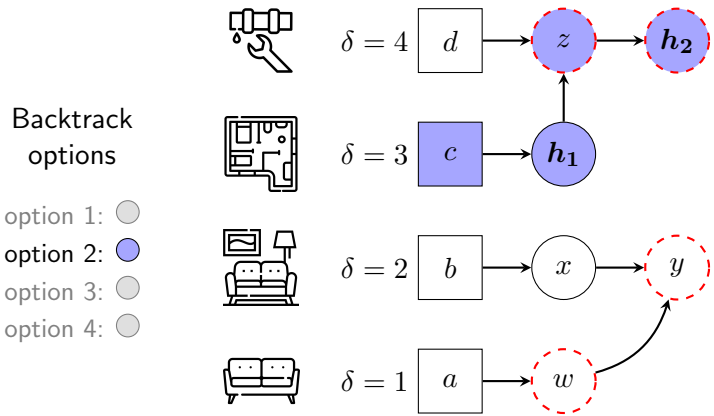
Graph Backtracking (GB)



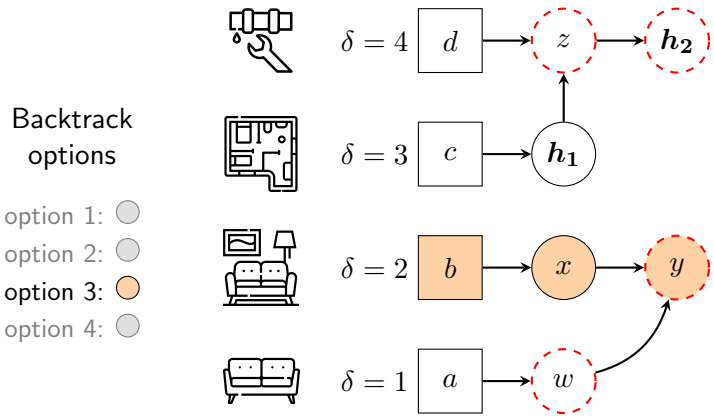
Graph Backtracking (GB)



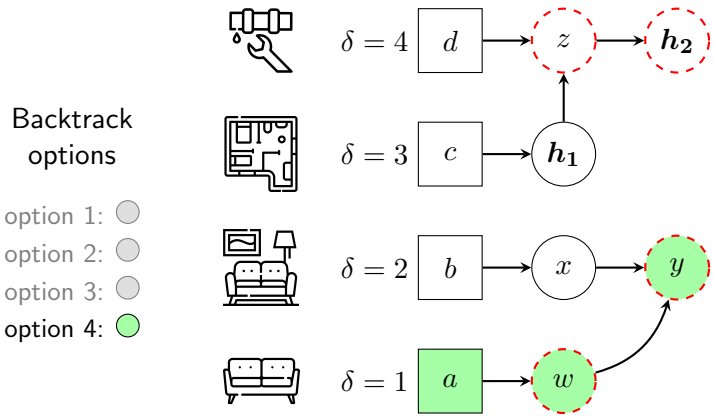
Graph Backtracking (GB)



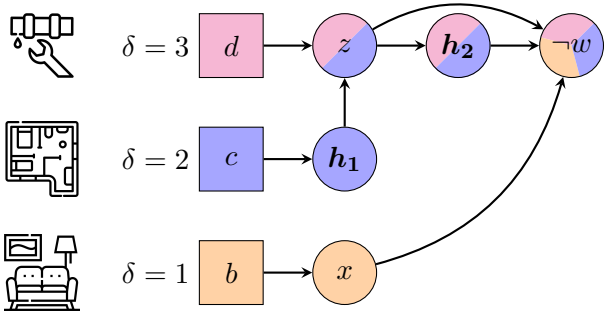
Graph Backtracking (GB)



Graph Backtracking (GB)



Graph Backtracking (GB)



What Graph Backtracking Buys Us

Precision

GB unassigns exactly the literals related to the conflict.

What Graph Backtracking Buys Us

Precision

GB unassigns exactly the literals related to the conflict.

Guarantees

- Sound, complete, and terminating
- Generalizes NCB and CB: both are instances of GB for specific weight functions

What Graph Backtracking Buys Us

Precision

GB unassigns exactly the literals related to the conflict.

Guarantees

- Sound, complete, and terminating
- 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.

How is it done?

Simple enough...

Why is this not mainstream?

How is it done?

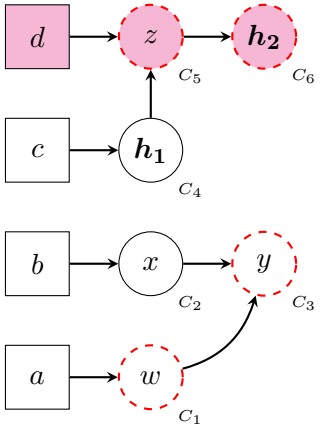
Simple enough...

Why is this not mainstream?

Core differences with (N)CB

- Decision dependency tracking
- New Unique Implication Point (UIP) scheme
- Termination safeguards
- Adjusted watch literals

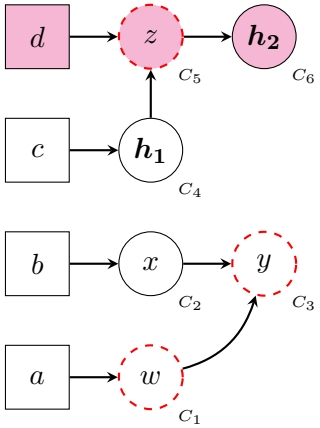
First Unique Implication Point (UIP) with GB



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

First Unique Implication Point (UIP) with GB

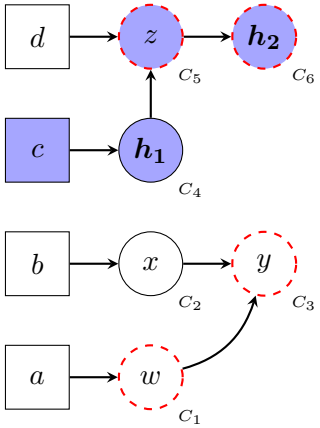
$\neg w \vee \neg y \vee \neg z$



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

First Unique Implication Point (UIP) with GB

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

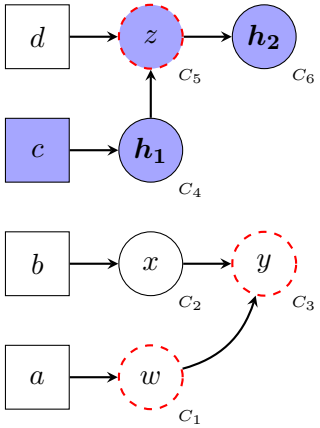


$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB

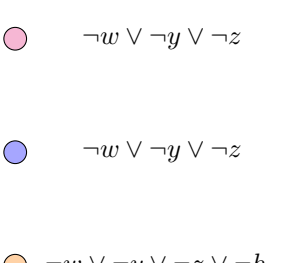
$\neg w \vee \neg y \vee \neg z$

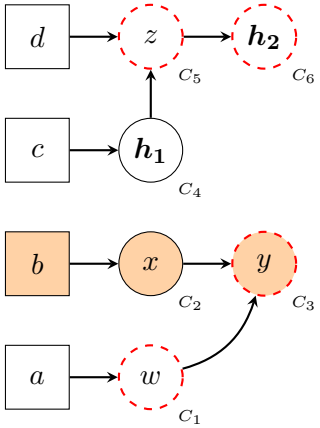
$\neg w \vee \neg y \vee \neg z$



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

First Unique Implication Point (UIP) with GB





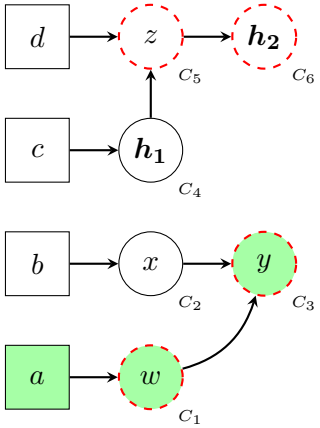
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB

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

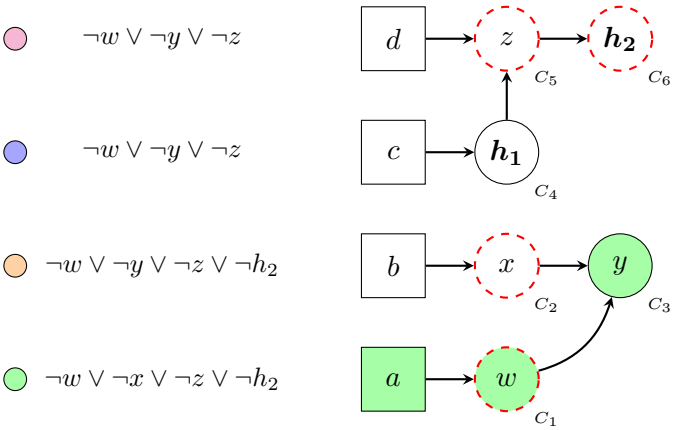
$\neg w \vee \neg y \vee \neg z$

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



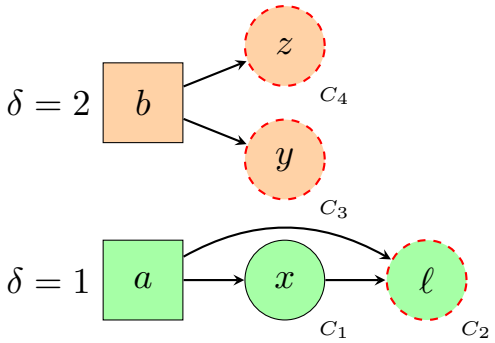
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

First Unique Implication Point (UIP) with GB



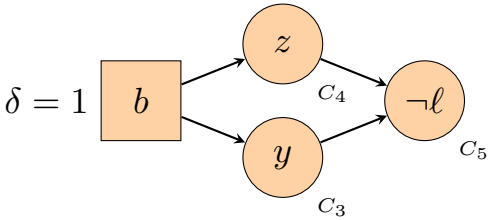
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Challenge I: Termination



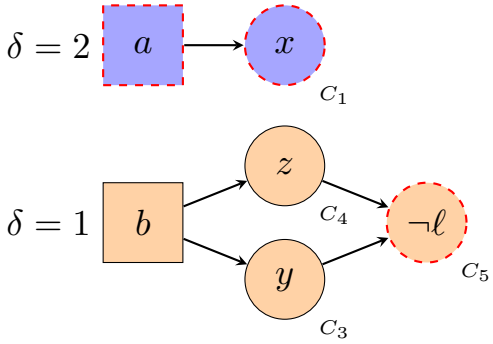
$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= z \vee \neg b \\
C_5 &= \neg \ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Termination



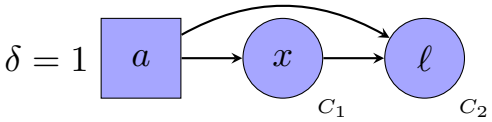
$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= z \vee \neg b \\
C_5 &= \neg\ell \vee \neg y \vee \neg z
\end{aligned}
$$

Challenge I: Termination



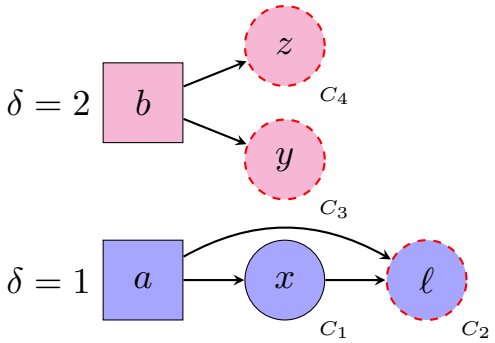
$$
\begin{aligned}
C_1 &= x \vee \neg a \\
C_2 &= \ell \vee \neg a \vee \neg x \\
C_3 &= y \vee \neg b \\
C_4 &= y \vee \neg b \\
C_5 &= \neg\ell \vee \neg y \vee \neg z
\end{aligned}
$$

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

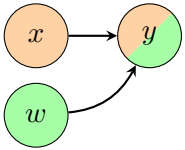
Why this happens

- 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.

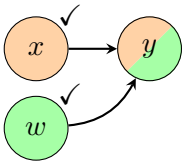
$$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$$

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

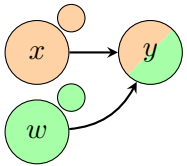
$$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$$

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.

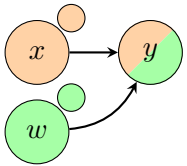
$$C_3 = \underline{y} \vee \underline{\neg x} \vee \neg w$$

Invariant (Watched Literals)

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

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

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$(y) \wedge$ ◔ $\subseteq$ ◯]

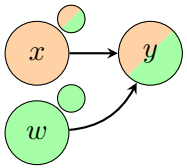
is satisfied.

Invariant (Watched Literals)

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

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

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$(y) \wedge$ ◯ $\subseteq$ ◯]

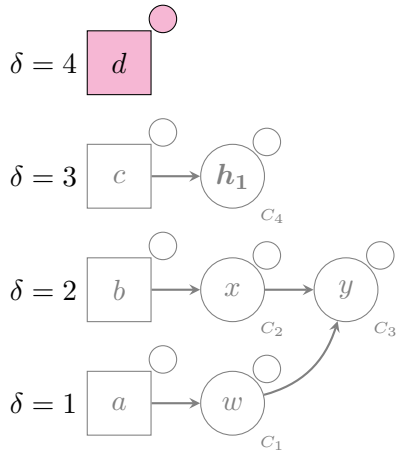
is satisfied.

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: Cross-Coloring Example



$C_1 = \underline{w} \vee \underline{\neg a}$

$C_2 = \underline{x} \vee \underline{\neg b}$

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

$C_4 = \underline{h_1} \vee \underline{\neg c}$

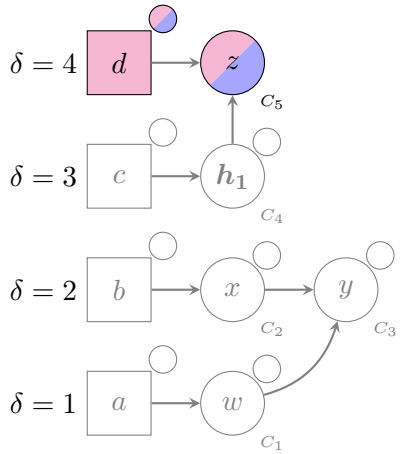
$C_5 = \underline{z} \vee \underline{\neg d} \vee \neg h_1$

$C_6 = \underline{h_2} \vee \underline{\neg z}$

$C_7 = \underline{a} \vee \neg c \vee \underline{\neg d}$

$C_8 = \neg w \vee \neg y \vee \underline{\neg z} \vee \underline{\neg h_2}$

Challenge II: Cross-Coloring Example



$C_1 = \underline{w} \vee \underline{\neg a}$

$C_2 = \underline{x} \vee \underline{\neg b}$

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

$C_4 = \underline{h_1} \vee \underline{\neg c}$

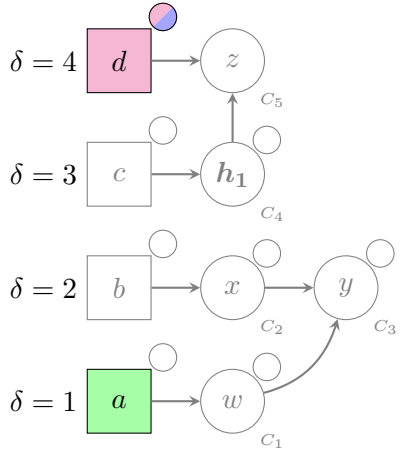
$C_5 = \underline{z} \vee \underline{\neg d} \vee \neg h_1$

$C_6 = \underline{h_2} \vee \underline{\neg z}$

$C_7 = \underline{a} \vee \neg c \vee \underline{\neg d}$

$C_8 = \neg w \vee \neg y \vee \underline{\neg z} \vee \underline{\neg h_2}$

Challenge II: Cross-Coloring Example



$C_1 = \underline{w} \vee \underline{\neg a}$

$C_2 = \underline{x} \vee \underline{\neg b}$

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

$C_4 = \underline{h_1} \vee \underline{\neg c}$

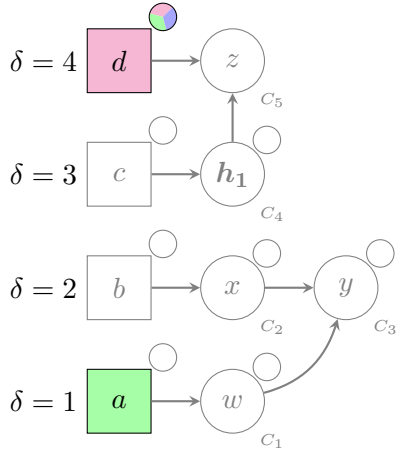
$C_5 = \underline{z} \vee \underline{\neg d} \vee \neg h_1$

$C_6 = \underline{h_2} \vee \underline{\neg z}$

$C_7 = \underline{a} \vee \neg c \vee \underline{\neg d}$

$C_8 = \neg w \vee \neg y \vee \underline{\neg z} \vee \underline{\neg h_2}$

Challenge II: Cross-Coloring Example



$C_1 = \underline{w} \vee \underline{\neg a}$

$C_2 = \underline{x} \vee \underline{\neg b}$

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

$C_4 = \underline{h_1} \vee \underline{\neg c}$

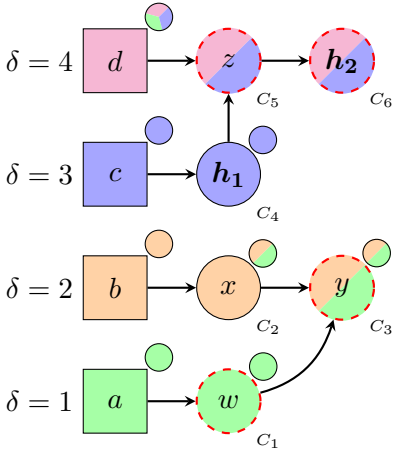
$C_5 = \underline{z} \vee \underline{\neg d} \vee \neg h_1$

$C_6 = \underline{h_2} \vee \underline{\neg z}$

$C_7 = \underline{a} \vee \neg c \vee \underline{\neg d}$

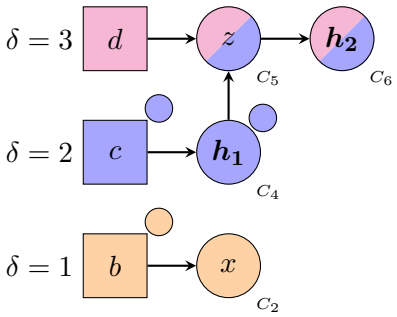
$C_8 = \neg w \vee \neg y \vee \underline{\neg z} \vee \underline{\neg h_2}$

Challenge II: Cross-Coloring Example



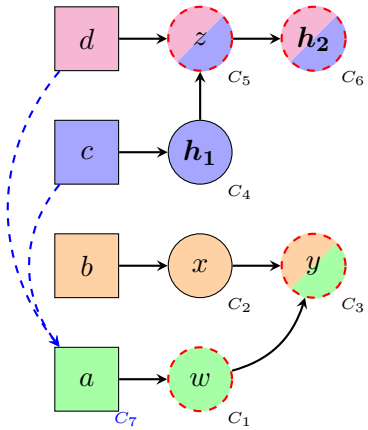
$$C_1 = \underline{w} \vee \underline{\neg a}$$
$$C_2 = \underline{x} \vee \underline{\neg b}$$
$$C_3 = \underline{y} \vee \neg w \vee \underline{\neg x}$$
$$C_4 = \underline{h_1} \vee \underline{\neg c}$$
$$C_5 = \underline{z} \vee \underline{\neg d} \vee \neg h_1$$
$$C_6 = \underline{h_2} \vee \underline{\neg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

Challenge II: Cross-Coloring Example



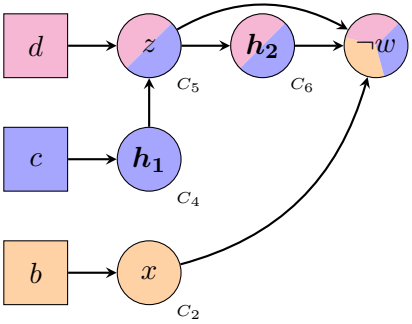
$$C_1 = \underline{w} \vee \underline{\neg a}$$
$$C_2 = \underline{x} \vee \underline{\neg b}$$
$$C_3 = \underline{y} \vee \neg w \vee \underline{\neg x}$$
$$C_4 = \underline{h_1} \vee \underline{\neg c}$$
$$C_5 = \underline{z} \vee \underline{\neg d} \vee \neg h_1$$
$$C_6 = \underline{h_2} \vee \underline{\neg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

Opportunities: Color Merging



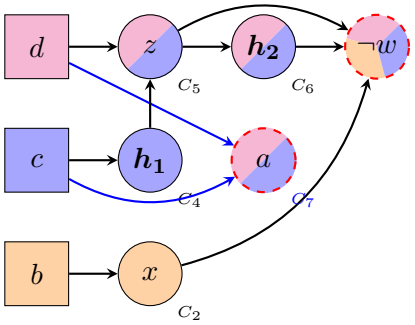
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$

Opportunities: Color Merging



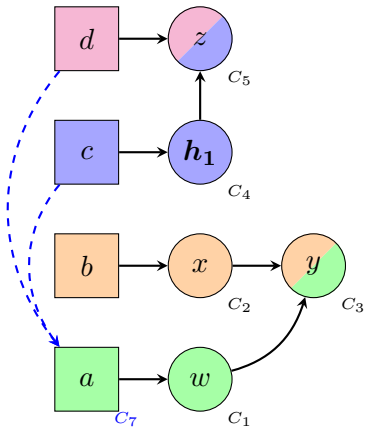
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9' = \neg w \vee \neg x \vee \neg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

Opportunities: Color Merging



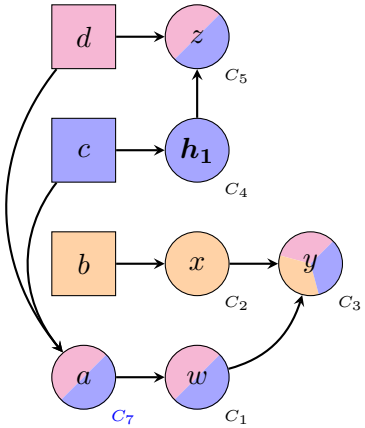
$$C_1 = w \vee \neg a$$
$$C_2 = x \vee \neg b$$
$$C_3 = y \vee \neg w \vee \neg x$$
$$C_4 = h_1 \vee \neg c$$
$$C_5 = z \vee \neg d \vee \neg h_1$$
$$C_6 = h_2 \vee \neg z$$
$$C_7 = a \vee \neg c \vee \neg d$$
$$C_8 = \neg w \vee \neg y \vee \neg z \vee \neg h_2$$
$$C_9' = \neg w \vee \neg x \vee \neg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

Refinement I: Eager Color Merging (ECM)



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

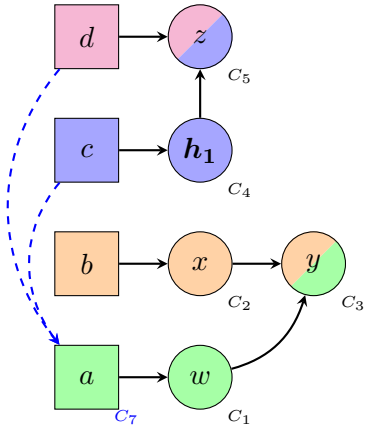
Refinement I: Eager Color Merging (ECM)



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

Refinement II: Lazy Color Merging (LCM)

Backtrack options



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

Refinement II: Lazy Color Merging (LCM)

Backtrack options

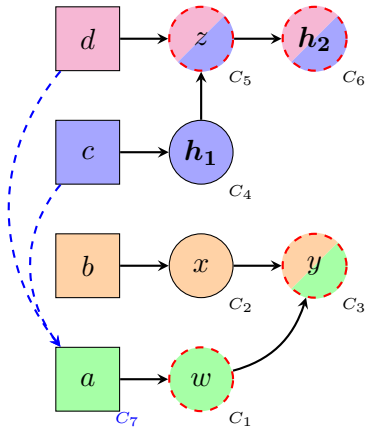
option 1:

option 2:

option 3:

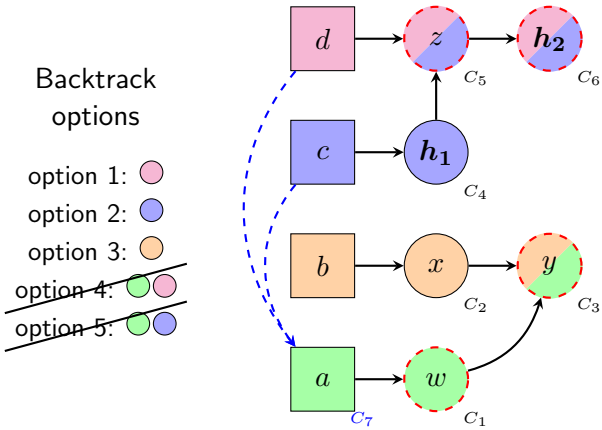
option 4:

option 5:



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

Refinement II: Lazy Color Merging (LCM)



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

Preliminary Empirical Results in NapSAT

Option	# Timeout	PAR2	Time (s)		Propagations $\times 10^6$	
NCB	16	380.7	106 ± 428	-	23.7 ± 46.4	-
CB	11	333.1	103 ± 414	97.33%	22.9 ± 42.8	96.30%
GB	17	378.1	103 ± 402	96.91%	14.2 ± 25.3	60.05%
GB+ECM	11	299.7	72.4 ± 310	67.89%	12.6 ± 23.8	52.96%
GB+LCM	14	376.8	124 ± 485	117.1%	15.8 ± 29.9	66.39%

Table: Empirical results on 1000 satisfiable random 3-coloring problems with 650 nodes and 1469 edges with 2-hour timeout.

Reminder

- (N)CB: (Non-)Chronological Backtracking
- GB: Graph Backtracking
- ECM: Eager Color Merging
- LCM: Lazy Color Merging

Results in Vampire [BBC+25] with AVATAR [Vor14]

We ran Vampire's AVATAR with NapSAT on 10617 TPTP problems requiring AVATAR.

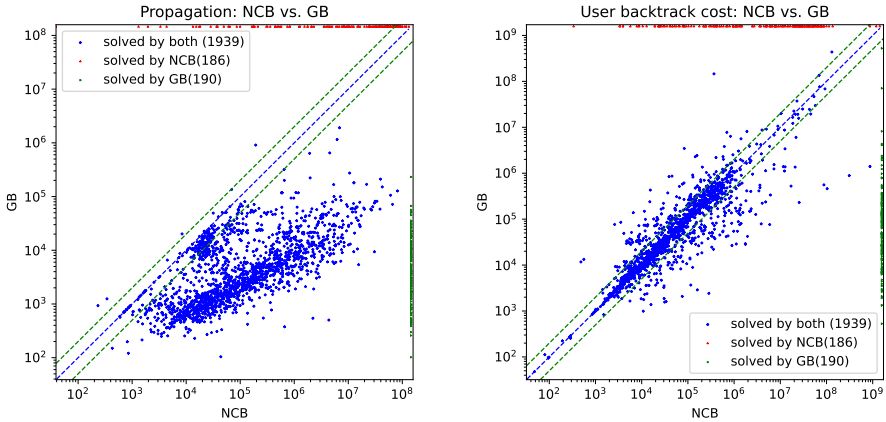


Figure: GB saves 98.7% of Boolean propagations and 70.3% of the estimated cost compared to NCB

Results in Vampire [BBC+25] with AVATAR [Vor14]

We ran Vampire's AVATAR with NapSAT on 10617 TPTP problems requiring AVATAR.

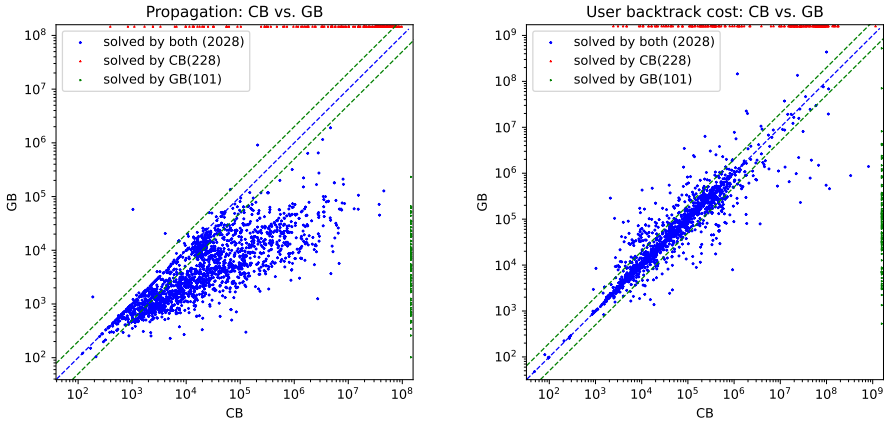
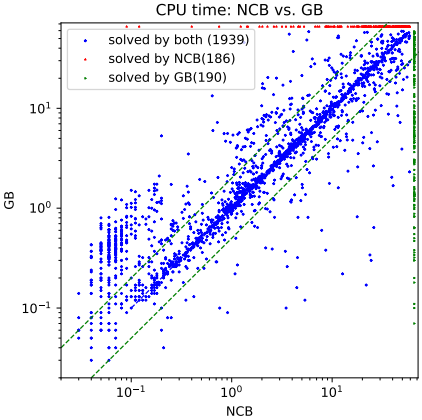
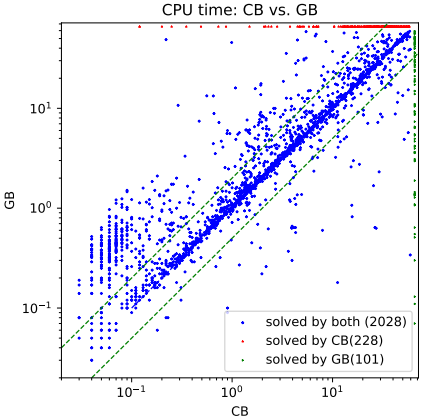


Figure: GB saves 95% of Boolean propagations and 69.3% of the estimated cost compared to CB

Results in Vampire [BBC+25] with AVATAR [Vor14]

We ran Vampire's AVATAR with NapSAT on 10617 TPTP problems requiring AVATAR.





Conclusion

Summary

- Graph Backtracking (GB), a new backtracking scheme for SAT solvers
- GB extends the SAT API with cost functions
- GB is implemented in NapSAT and integrated in Vampire's AVATAR
- GB performs significantly fewer Boolean propagations than (N)CB.
- GB achieves the goal of reducing the estimated cost of backtracking in AVATAR.

Conclusion

Summary

- Graph Backtracking (GB), a new backtracking scheme for SAT solvers
- GB extends the SAT API with cost functions
- GB is implemented in NapSAT and integrated in Vampire's AVATAR
- GB performs significantly fewer Boolean propagations than (N)CB.
- GB achieves the goal of reducing the estimated cost of backtracking in AVATAR.

Future Work

- Tune Vampire AVATAR's cost heuristic to take advantage of GB.
- Improve scalability with respect to the number of decisions.
- Relax SMT stack discipline.
- Implement GB in cvc5 [BBB+22].

Conclusion

Summary

- Graph Backtracking (GB), a new backtracking scheme for SAT solvers
- GB extends the SAT API with cost functions
- GB is implemented in NapSAT and integrated in Vampire's AVATAR
- GB performs significantly fewer Boolean propagations than (N)CB.
- GB achieves the goal of reducing the estimated cost of backtracking in AVATAR.

Future Work

- Tune Vampire AVATAR's cost heuristic to take advantage of GB.
- Improve scalability with respect to the number of decisions.
- Relax SMT stack discipline.
- Implement GB in cvc5 [BBB+22].

Thank you for your attention!

References I

Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al.

cvc5: A versatile and industrial-strength SMT solver.

In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442. Springer, 2022.

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.

References II

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.

Flaticon.

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

Accessed: 2026-01-10.

References III



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 IV



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 V



João P. Marques Silva and Karem A. Sakallah.

GRASP: A search algorithm for propositional satisfiability.

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

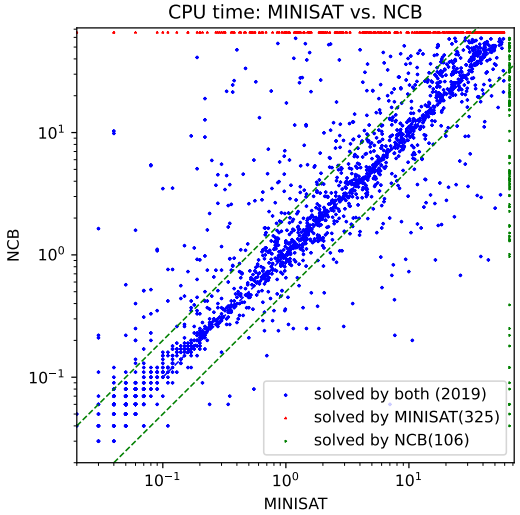


Andrei Voronkov.

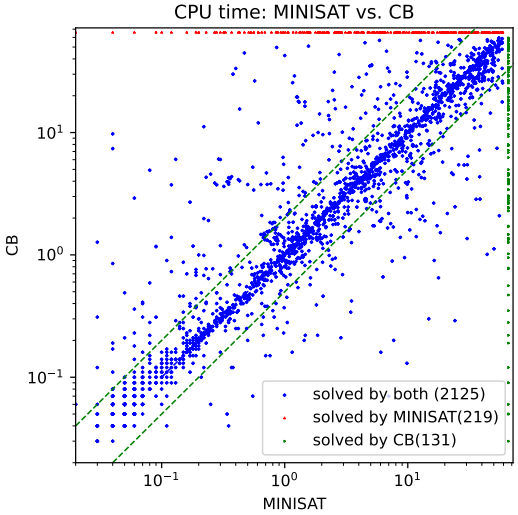
AVATAR: the architecture for first-order theorem provers.

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

MiniSAT vs NapSAT in AVATAR



MiniSAT vs NapSAT in AVATAR



MiniSAT vs NapSAT in AVATAR

