Less Aggressive Backtracking of Expensive SAT Literals

BIRS Workshop

Robin Coutelier

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

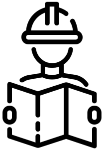
January 2026

Acknowledgements

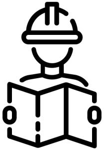
This work is a collaboration with Thomas Hader and Laura Kovács.

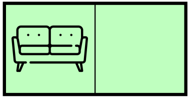
The authors acknowledge support from the ERC Consolidator Grant ARTIST 101002685; the TU Wien Doctoral College TrustACPS; the FWF SpyCoDe SFB projects F8504; the WWTF Grant ForSmart 10.47379/ICT22007

Introduction

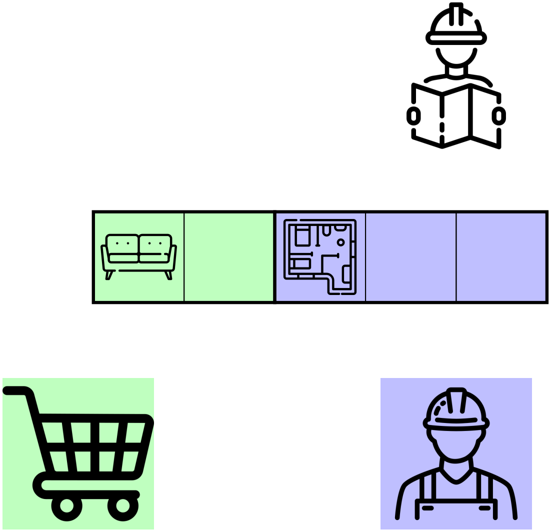


Introduction

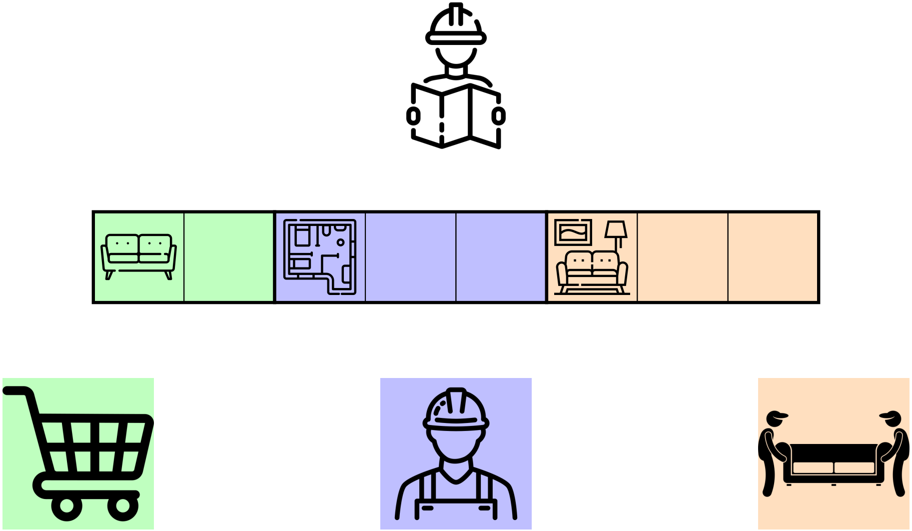




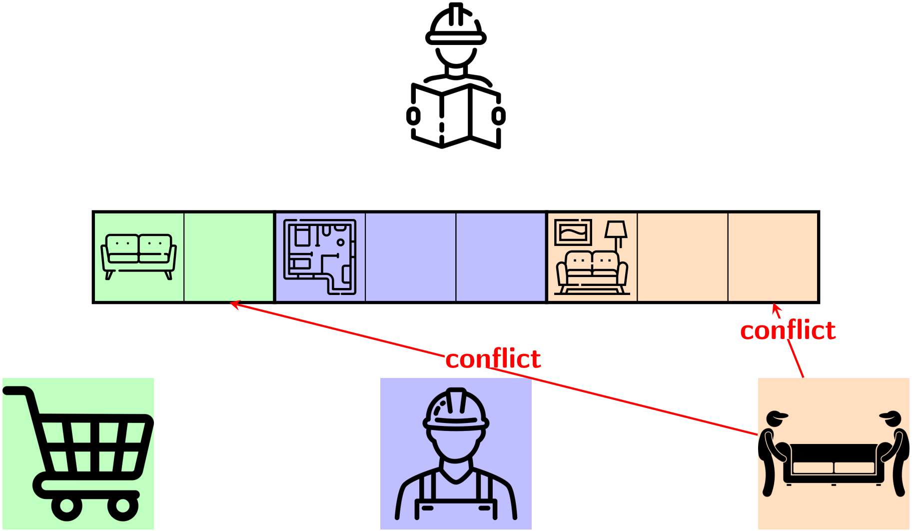
Introduction



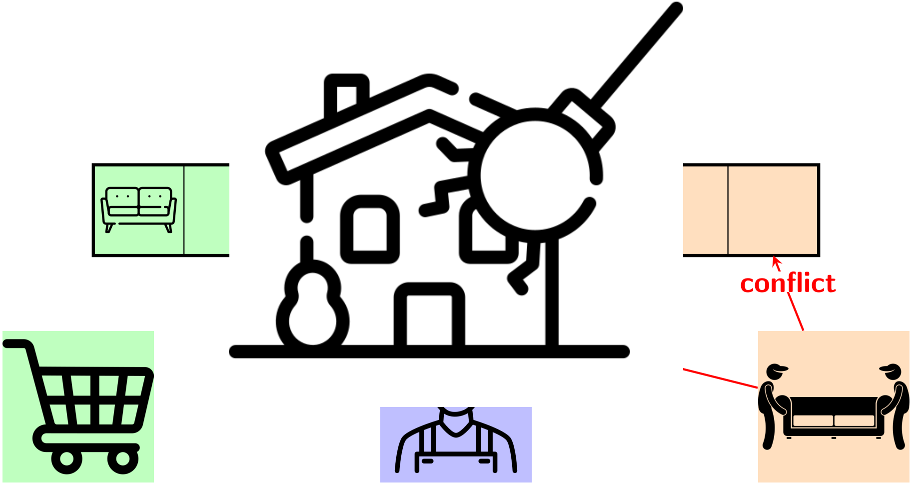
Introduction



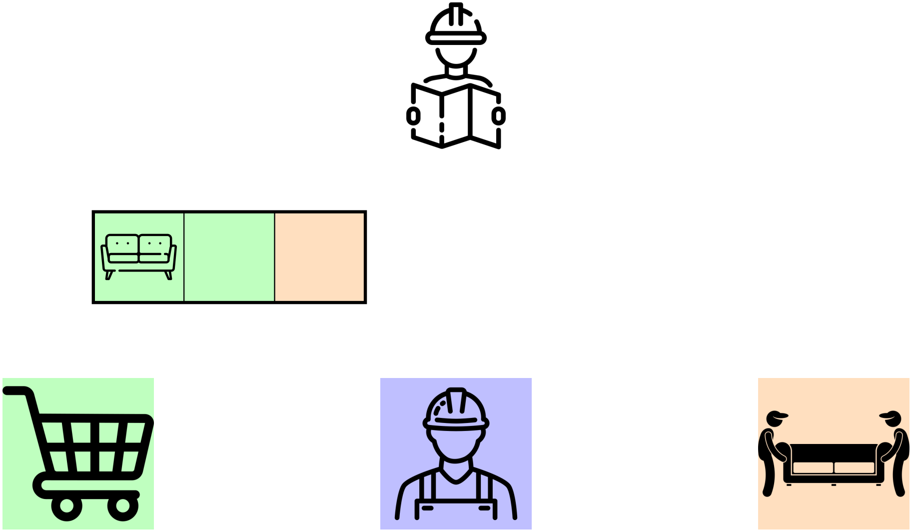
Introduction



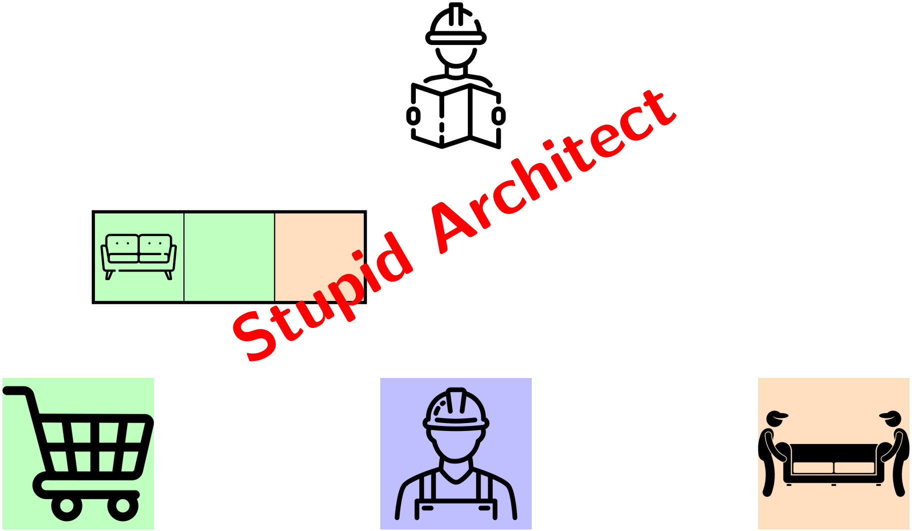
Introduction



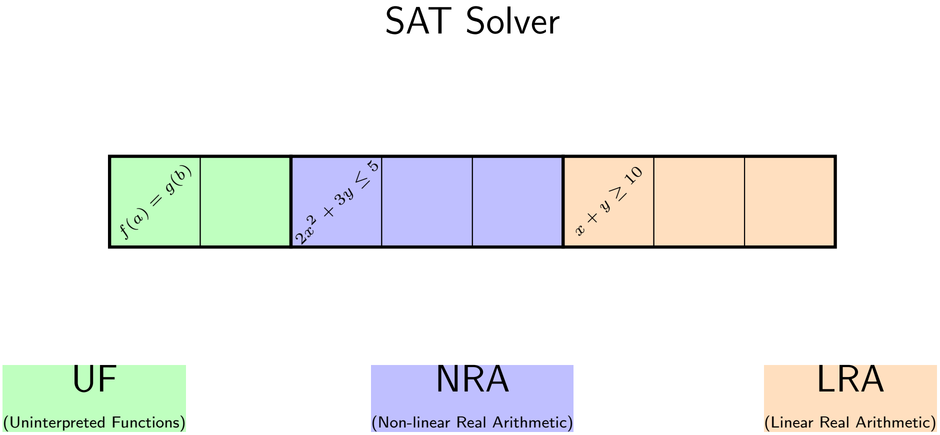
Introduction



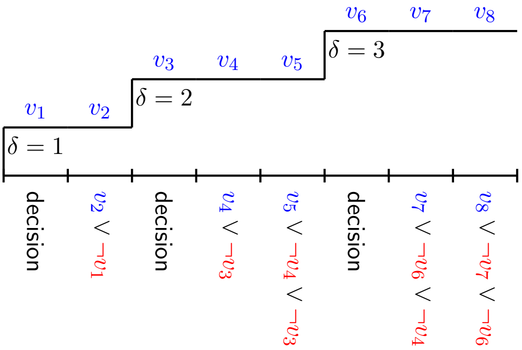
Introduction



SMT Architecture

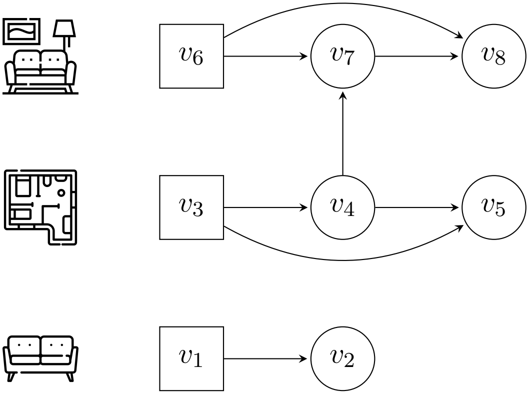


Non-Chronological Backtracking – Trail



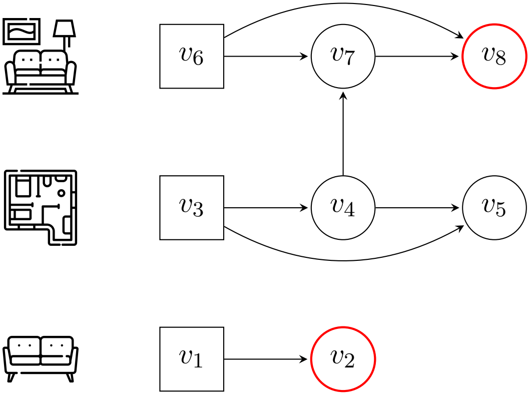
$$
\begin{aligned}
C_1 &= v_2 \vee \neg v_1 \\
C_2 &= v_4 \vee \neg v_3 \\
C_3 &= v_5 \vee \neg v_4 \vee \neg v_3 \\
C_4 &= v_7 \vee \neg v_6 \vee \neg v_4 \\
C_5 &= v_8 \vee \neg v_7 \vee \neg v_6
\end{aligned}
$$

Non-Chronological Backtracking – Implication Graph



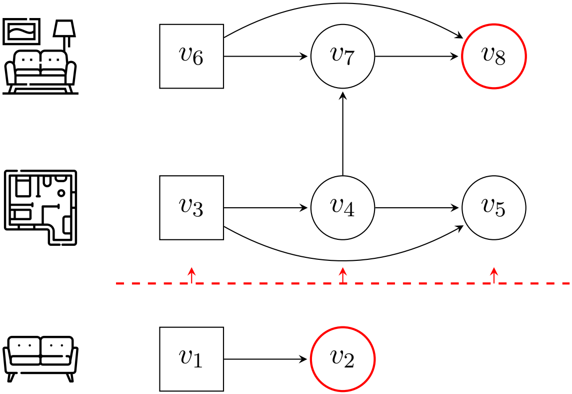
$$
\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6
\end{aligned}
$$

Non-Chronological Backtracking – Implication Graph



$$
\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6 \\
C_6 &= \neg v_8 \lor \neg v_2
\end{aligned}
$$

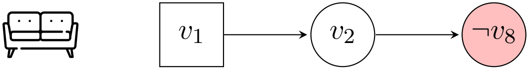
Non-Chronological Backtracking – Implication Graph



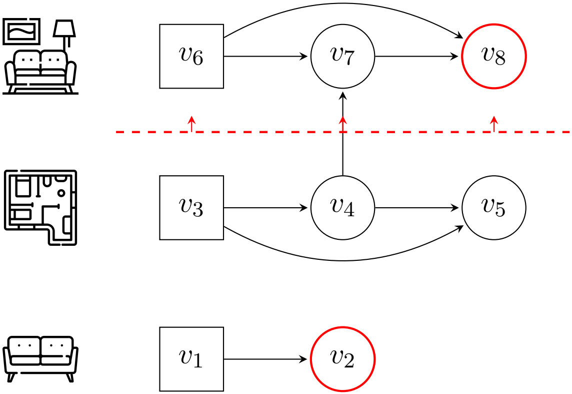
$$
\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6 \\
C_6 &= \neg v_8 \lor \neg v_2
\end{aligned}
$$

Non-Chronological Backtracking – Implication Graph

$$
\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6 \\
C_6 &= \neg v_8 \lor \neg v_2
\end{aligned}
$$

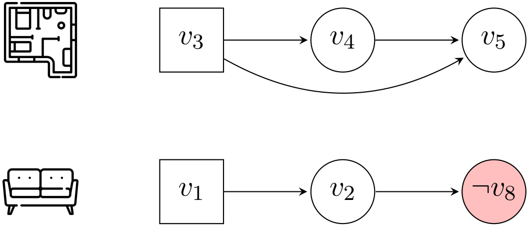


Chronological Backtracking [NR18, MB19, Nad22, CFK24]



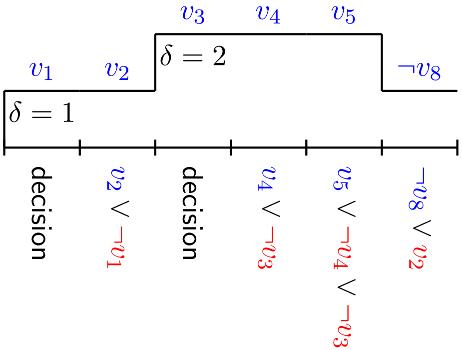
$$C_1 = v_2 \lor \neg v_1$$
$$C_2 = v_4 \lor \neg v_3$$
$$C_3 = v_5 \lor \neg v_4 \lor \neg v_3$$
$$C_4 = v_7 \lor \neg v_6 \lor \neg v_4$$
$$C_5 = v_8 \lor \neg v_7 \lor \neg v_6$$
$$C_6 = \neg v_8 \lor \neg v_2$$

Chronological Backtracking [NR18, MB19, Nad22, CFK24]



$$\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6 \\
C_6 &= \neg v_8 \lor \neg v_2
\end{aligned}$$

Chronological Backtracking – Trail



$$C_1 = v_2 \vee \neg v_1$$
$$C_2 = v_4 \vee \neg v_3$$
$$C_3 = v_5 \vee \neg v_4 \vee \neg v_3$$
$$C_4 = v_7 \vee \neg v_6 \vee \neg v_4$$
$$C_5 = v_8 \vee \neg v_7 \vee \neg v_6$$
$$C_6 = \neg v_8 \vee \neg v_2$$

Why is the Architect "Stupid"?

To be fast

- NCB maintains nice **invariants**
- Propagations are cheap
- SMT solvers assume a stack structure for the trail

Why is the Architect “Stupid”?

To be fast

- NCB maintains nice **invariants**
- Propagations are cheap
- SMT solvers assume a stack structure for the trail

To be general

- The SAT solver does not know about theories
- The cost of literals is unknown to the SAT solver

Why is the Architect “Stupid”?

To be fast

- NCB maintains nice **invariants**
- Propagations are cheap
- SMT solvers assume a stack structure for the trail

To be general

- The SAT solver does not know about theories
- The cost of literals is unknown to the SAT solver

To be smart

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

Invariants in NCB and CB

Consider the trail $\pi = \tau \cdot \omega$. For each clause $C \in F$ watched by c_1, c_2 in $\mathrm{WL}(c_1)$, we have

Invariant (NCB Watched Literals)

$$\neg c_1 \in \tau \Rightarrow c_2 \in \pi$$

Invariants in NCB and CB

Consider the trail $\pi = \tau \cdot \omega$. For each clause $C \in F$ watched by c_1, c_2 in $\mathrm{WL}(c_1)$, we have

Invariant (NCB Watched Literals)

$$\neg c_1 \in \tau \Rightarrow c_2 \in \pi$$

Invariant (CB Watched Literals)

$$\neg c_1 \in \tau \Rightarrow \big(c_2 \in \pi \wedge \delta(c_2) \leq \delta(c_1)\big)$$

Smarter Architects

Extended Interface

We allow the user to provide a **cost** $\zeta(\ell)$ for each literal ℓ.

Smarter Architects

Extended Interface

We allow the user to provide a **cost** $\zeta(\ell)$ for each literal ℓ.

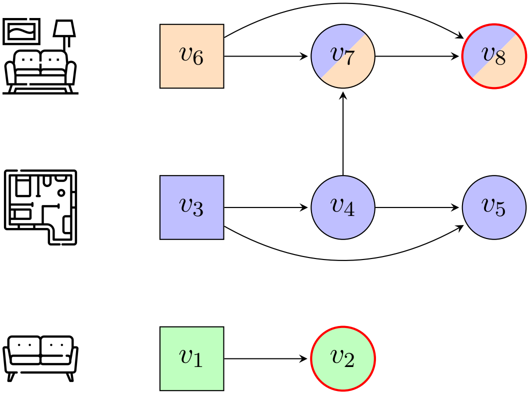
New Backtracking Scheme

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

Backtrack options

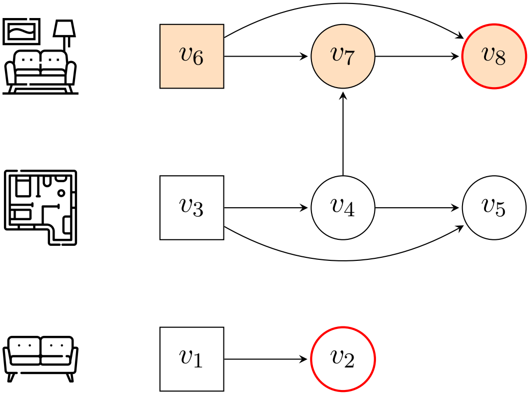
option 1:
option 2:
option 3:



Graph Backtracking

Backtrack options

option 1:
option 2:
option 3:



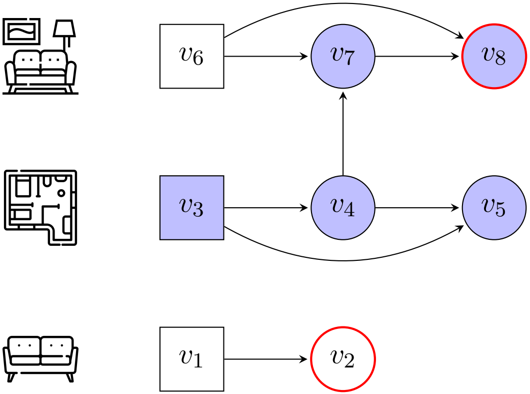
Graph Backtracking

Backtrack options

option 1:

option 2:

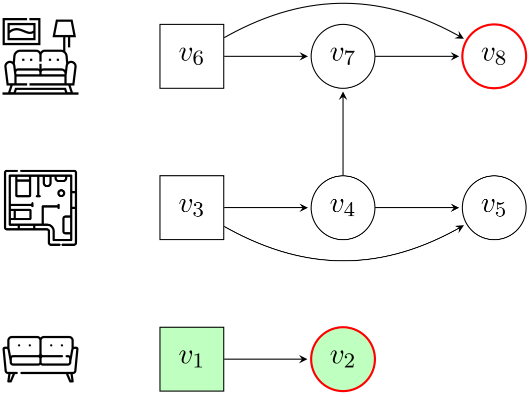
option 3:



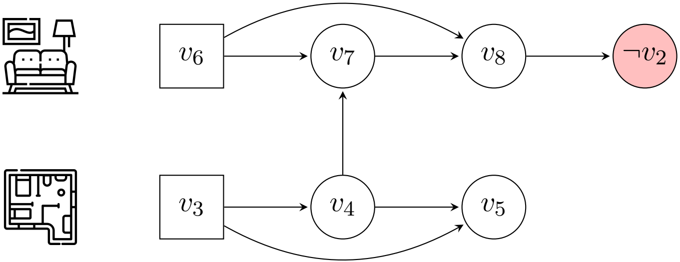
Graph Backtracking

Backtrack options

option 1:
option 2:
option 3:



Graph Backtracking



Invariants in NCB, CB and GB

Consider the trail $\pi = \tau \cdot \omega$. For each clause $C \in F$ watched by c_1, c_2 and blocked by b in $\mathrm{WL}(c_1)$, we have

Invariant (NCB Watched Literals)

$$\neg c_1 \in \tau \Rightarrow c_2 \in \pi$$

Invariant (CB Watched Literals)

$$\neg c_1 \in \tau \Rightarrow \big(c_2 \in \pi \wedge \delta(c_2) \leq \delta(c_1)\big)$$

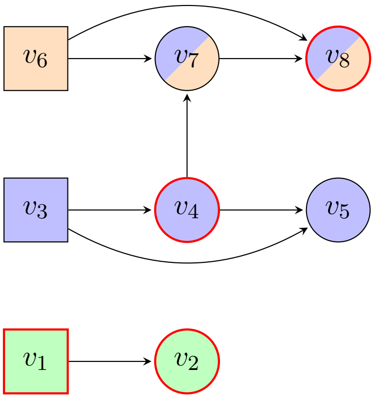
Invariant (GB Watched Literals)

$$\neg c_1 \in \tau \Rightarrow \big(c_2 \in \pi \wedge \gamma(c_2) \subseteq \eta(c_1)\big)$$

1UIP with GB

Conflicting clause: $\neg v_1 \vee \neg v_2 \vee \neg v_4 \vee \neg v_8$

Backtrack
options

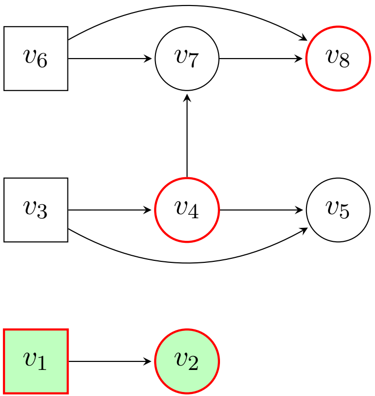


1UIP with GB

Conflicting clause: $\neg v_1 \vee \neg v_2 \vee \neg v_4 \vee \neg v_8$

Backtrack options

option 1:

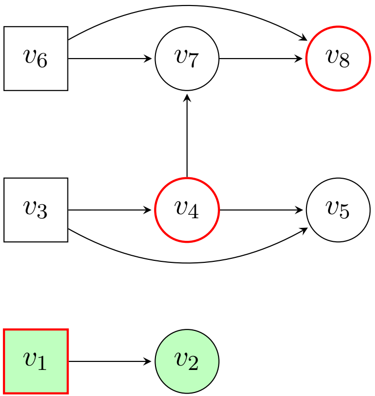


1UIP with GB

Conflicting clause: $\neg v_1 \vee \neg v_2 \vee \neg v_4 \vee \neg v_8$

Backtrack options

option 1: ◯ $\boldsymbol{\neg v_1} \vee \neg v_4 \vee \neg v_8$



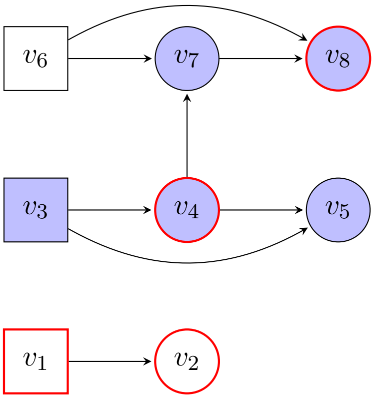
1UIP with GB

Conflicting clause: $\neg v_1 \vee \neg v_2 \vee \neg v_4 \vee \neg v_8$

Backtrack options

option 1: (green circle) $\boldsymbol{\neg v_1} \vee \neg v_4 \vee \neg v_8$

option 2: (blue circle)



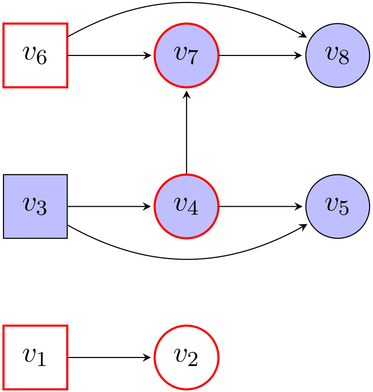
1UIP with GB

Conflicting clause: $\neg v_1 \lor \neg v_2 \lor \neg v_4 \lor \neg v_8$

Backtrack options

option 1: ◯ $\boldsymbol{\neg v_1} \lor \neg v_4 \lor \neg v_8$

option 2: ◯



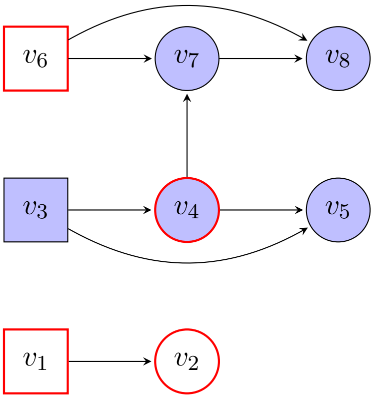
1UIP with GB

Conflicting clause: $\neg v_1 \lor \neg v_2 \lor \neg v_4 \lor \neg v_8$

Backtrack options

option 1: $\boldsymbol{\neg v_1} \lor \neg v_4 \lor \neg v_8$

option 2: $\neg v_1 \lor \neg v_2 \lor \boldsymbol{\neg v_4} \lor \neg v_6$



1UIP with GB

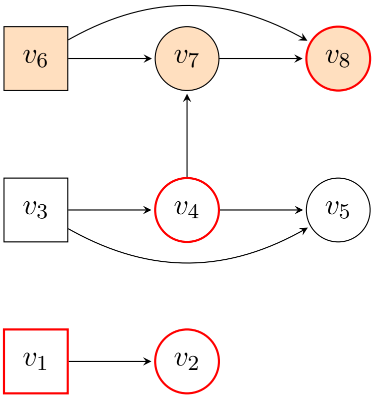
Conflicting clause: $\neg v_1 \lor \neg v_2 \lor \neg v_4 \lor \neg v_8$

Backtrack options

option 1: $\boldsymbol{\neg v_1} \lor \neg v_4 \lor \neg v_8$

option 2: $\neg v_1 \lor \neg v_2 \lor \boldsymbol{\neg v_4} \lor \neg v_6$

option 3:



1UIP with GB

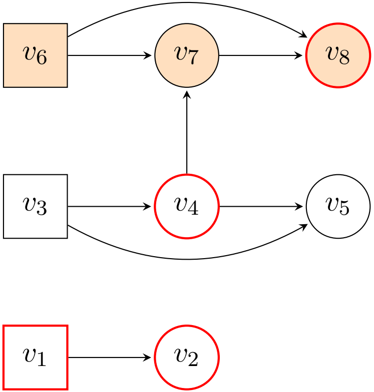
Conflicting clause: $\neg v_1 \vee \neg v_2 \vee \neg v_4 \vee \neg v_8$

Backtrack options

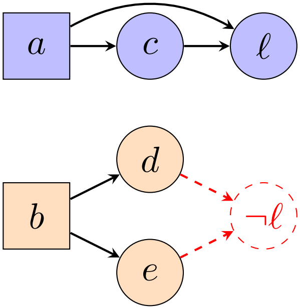
option 1: $\boldsymbol{\neg v_1} \vee \neg v_4 \vee \neg v_8$

option 2: $\neg v_1 \vee \neg v_2 \vee \boldsymbol{\neg v_4} \vee \neg v_6$

option 3: $\neg v_1 \vee \neg v_2 \vee \neg v_4 \vee \boldsymbol{\neg v_8}$

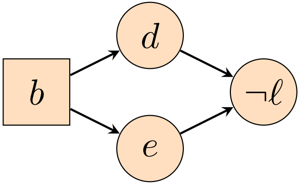


Challenge I: Termination



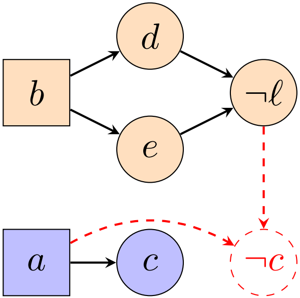
$$
\begin{aligned}
C_1 &= c \lor \neg a \\
C_2 &= \ell \lor \neg a \lor \neg c \\
C_3 &= d \lor \neg b \\
C_4 &= e \lor \neg b \\
C_5 &= \neg \ell \lor \neg d \lor \neg e
\end{aligned}
$$

Challenge I: Termination



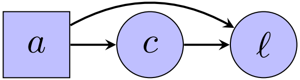
$$
\begin{aligned}
C_1 &= c \lor \neg a \\
C_2 &= \ell \lor \neg a \lor \neg c \\
C_3 &= d \lor \neg b \\
C_4 &= e \lor \neg b \\
C_5 &= \neg \ell \lor \neg d \lor \neg e
\end{aligned}
$$

Challenge I: Termination



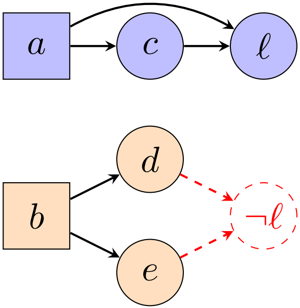
$$
\begin{aligned}
C_1 &= \quad c \lor \neg a \\
C_2 &= \quad \ell \lor \neg a \lor \neg c \\
C_3 &= \quad d \lor \neg b \\
C_4 &= \quad e \lor \neg b \\
C_5 &= \neg\ell \lor \neg d \lor \neg e
\end{aligned}
$$

Challenge I: Termination



$$
\begin{aligned}
C_1 &= c \lor \neg a \\
C_2 &= \ell \lor \neg a \lor \neg c \\
C_3 &= d \lor \neg b \\
C_4 &= e \lor \neg b \\
C_5 &= \neg \ell \lor \neg d \lor \neg e
\end{aligned}
$$

Challenge I: Termination



$$
\begin{aligned}
C_1 &= c \lor \neg a \\
C_2 &= \ell \lor \neg a \lor \neg c \\
C_3 &= d \lor \neg b \\
C_4 &= e \lor \neg b \\
C_5 &= \neg \ell \lor \neg d \lor \neg e
\end{aligned}
$$

Challenge I: Safeguards

Why this happens

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

Challenge I: Safeguards

Why this happens

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

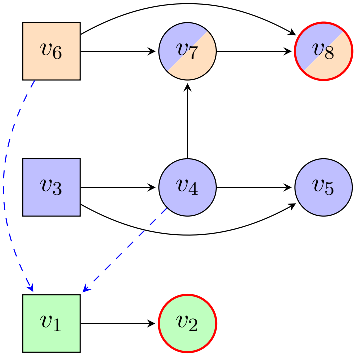
Solution

- If possible to learn a clause: choose the chunk that allows to learn the clause with minimum total cost
- Select latest decision otherwise (like in CB)

Challenge II: Redundant Implications

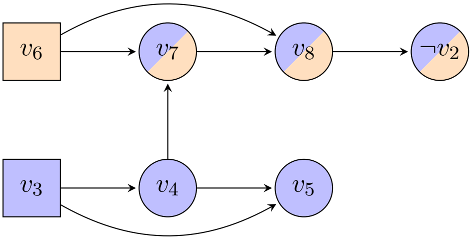
Backtrack options

option 1: (green)
option 2: (blue)
option 3: (orange)



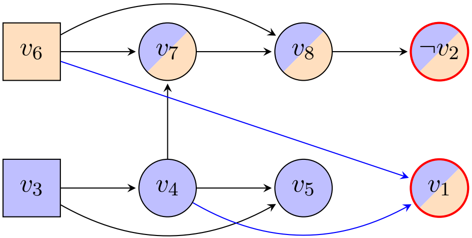
$$
\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6 \\
C_6 &= \neg v_8 \lor \neg v_2 \\
C_7 &= v_1 \lor \neg v_4 \lor \neg v_6
\end{aligned}
$$

Challenge II: Redundant Implications



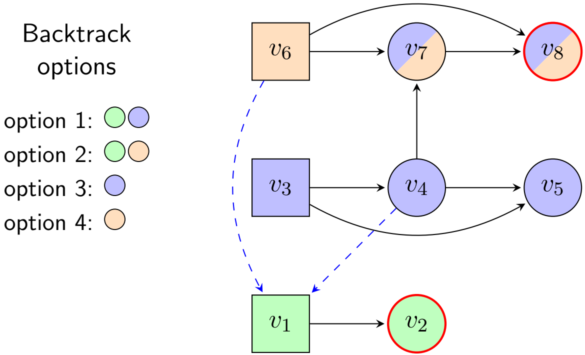
$$
\begin{aligned}
C_1 &= v_2 \lor \neg v_1 \\
C_2 &= v_4 \lor \neg v_3 \\
C_3 &= v_5 \lor \neg v_4 \lor \neg v_3 \\
C_4 &= v_7 \lor \neg v_6 \lor \neg v_4 \\
C_5 &= v_8 \lor \neg v_7 \lor \neg v_6 \\
C_6 &= \neg v_8 \lor \neg v_2 \\
C_7 &= v_1 \lor \neg v_4 \lor \neg v_6
\end{aligned}
$$

Challenge II: Redundant Implications



$$
\begin{aligned}
C_1 &= v_2 \vee \neg v_1 \\
C_2 &= v_4 \vee \neg v_3 \\
C_3 &= v_5 \vee \neg v_4 \vee \neg v_3 \\
C_4 &= v_7 \vee \neg v_6 \vee \neg v_4 \\
C_5 &= v_8 \vee \neg v_7 \vee \neg v_6 \\
C_6 &= \neg v_8 \vee \neg v_2 \\
C_7 &= v_1 \vee \neg v_4 \vee \neg v_6
\end{aligned}
$$

Challenge II: Redundant Implications

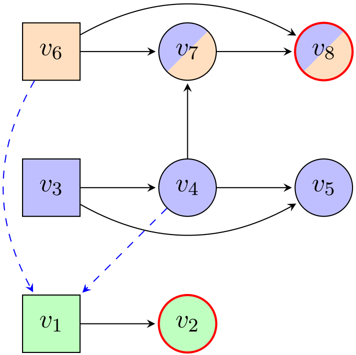


$$C_1 = v_2 \lor \neg v_1$$
$$C_2 = v_4 \lor \neg v_3$$
$$C_3 = v_5 \lor \neg v_4 \lor \neg v_3$$
$$C_4 = v_7 \lor \neg v_6 \lor \neg v_4$$
$$C_5 = v_8 \lor \neg v_7 \lor \neg v_6$$
$$C_6 = \neg v_8 \lor \neg v_2$$
$$C_7 = v_1 \lor \neg v_4 \lor \neg v_6$$

Challenge II: Redundant Implications

Backtrack options

~~option 1:~~ (green, blue)
~~option 2:~~ (green, orange)
option 3: (blue)
option 4: (orange)



$$C_1 = v_2 \vee \neg v_1$$
$$C_2 = v_4 \vee \neg v_3$$
$$C_3 = v_5 \vee \neg v_4 \vee \neg v_3$$
$$C_4 = v_7 \vee \neg v_6 \vee \neg v_4$$
$$C_5 = v_8 \vee \neg v_7 \vee \neg v_6$$
$$C_6 = \neg v_8 \vee \neg v_2$$
$$C_7 = v_1 \vee \neg v_4 \vee \neg v_6$$

Opportunities

Immediate vs. Exhaustive Conflict Repair [BF25]

When a conflict is detected, we can either

- immediately backtrack to resolve it (like NCB and CB)
- or gather all possible conflicts and resolve them in one big backtrack step

Opportunities

Immediate vs. Exhaustive Conflict Repair [BF25]

When a conflict is detected, we can either

- immediately backtrack to resolve it (like NCB and CB)
- or gather all possible conflicts and resolve them in one big backtrack step

Dynamic Cost Adjustment

The cost function $\zeta(\ell)$ can be dynamically adjusted based on cached information from the user.

Opportunities

Immediate vs. Exhaustive Conflict Repair [BF25]

When a conflict is detected, we can either

- immediately backtrack to resolve it (like NCB and CB)
- or gather all possible conflicts and resolve them in one big backtrack step

Dynamic Cost Adjustment

The cost function $\zeta(\ell)$ can be dynamically adjusted based on cached information from the user.

Multiple Clause Learning

We can attempt learning multiple clauses from the same conflict, and pick the best one according to the cost function.
(it does not work well in practice though)

Empirical Results

Option	Time (s)	Sync $\times 10^3$	Propagation $\times 10^6$
NCB	1.11 ± 3.15	567.98 ± 925.2	2.56 ± 4.29
CB	1.13 ± 5.41	508.73 ± 930.64	2.47 ± 4.58
GB	1.93 ± 5.43	$\mathbf{358.65} \pm \mathbf{571.43}$	1.84 ± 3.04
NCB+ECR	14.88 ± 68.77	511.46 ± 788.33	8.09 ± 13.05
CB+ECR	16.51 ± 73.67	487.29 ± 799.73	8.65 ± 14.65
GB+ECR	32.55 ± 131.72	$\mathbf{319.18} \pm \mathbf{510.48}$	7.38 ± 12.5

Table: Experiments on 1000 graph coloring instances `kcolor 3 gmn 400 920`. Average time, number of synchronizations and propagations. Standard deviation is shown after the $\pm$ symbol.

Conclusion

Summary

- Graph Backtracking (GB), a new backtracking scheme for SAT solvers
- GB extends the SAT API with cost functions
- GB is more gentle with an incremental user

Conclusion

Summary

- Graph Backtracking (GB), a new backtracking scheme for SAT solvers
- GB extends the SAT API with cost functions
- GB is more gentle with an incremental user

Future Work

- Implement GB in Vampire AVATAR [Vor14]
- Relax SMT stack discipline into multi stack
- 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 more gentle with an incremental user

Future Work

- Implement GB in Vampire AVATAR [Vor14]
- Relax SMT stack discipline into multi stack
- 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.

Yasmine Briefs and Mathias Fleury.

From building blocks to real SAT solvers.

First-Order Reasoning, Below and Beyond: Workshop in Honor of Christoph Weidenbach's 60th Birthday, Colocated with CADE'30, 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, 2025.

Presentation only.

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.

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.

References IV

Alexander Nadel and Vadim Ryvchin.

Chronological backtracking.

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

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.