

Constraint Satisfaction Problems

Look-Back

Malte Helmert and Stefan Wöfl

Albert-Ludwigs-Universität Freiburg

June 5, 2007

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Literature

Look-Back Techniques

- **Look-ahead** techniques reduce the size of the searched part of the state space by excluding partial assignments from consideration if they provably lead to inconsistencies.
- This is a form of **forward analysis**: We avoid assignments which must lead to dead ends **in the future**.
- **Look-back techniques** use a complementary approach: We avoid assignments which led to dead ends **in the past**.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Literature

Types of Look-Back Techniques

We will consider two classes of look-back techniques:

- **Backjumping:** Upon encountering a dead end, do not always return to the parent in the search tree, but possibly to an earlier ancestor.
- **No-good learning:** Upon encountering a dead end, record a new constraint to detect this type of dead end earlier in the future.

No-good learning is commonly used when solving propositional logic satisfiability problems for CNF formulae. In this context, it is known as **clause learning**.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Literature

- Throughout the chapter, we assume a **fixed variable ordering** v_1, \dots, v_n .
- **Partial assignments** $a = \{v_1 \mapsto a_1, \dots, v_i \mapsto a_i\}$ for $i \in \{0, \dots, n\}$ are abbreviated as tuples: (a_1, \dots, a_i) .

Dead Ends

Recall:

Definition (dead end)

A **dead end** of a state space is a state which is not a goal state and in which no operator is applicable.

In the context of look-back methods, we use the following terminology:

Definition (leaf dead end)

A **leaf dead end** is a partial solution (a_1, \dots, a_i) such that (a_1, \dots, a_{i+1}) is inconsistent for all possible values of v_{i+1} . Variable v_{i+1} is called the **leaf dead-end variable** for the leaf dead end.

Conflict Sets

Definition (conflict set)

Let a be a partial solution (on an arbitrary set of variables), and let v_j be a variable for which a is not defined.

We say that a is a **conflict set** of v_j , (or: a is **in conflict with** v_j) if no assignment of the form $a \cup \{v_j \mapsto a_j\}$ is consistent.

If moreover a contains no subtuple which is in conflict with v_j , it is a **minimal conflict set** of v_j .

↔ A leaf dead end is a conflict set of the leaf dead-end variable, but not every conflict set is a leaf dead end.

No-Goods and Internal Dead Ends

Definition (no-good)

A partial solution that cannot be extended to a solution of the network is called a **no-good**.

A no-good is **minimal** if it contains no no-good subassignments.

A no-good is called an **internal dead end** iff it is defined on the first i variables, i.e., on $\{v_1, \dots, v_i\}$ and it is not a leaf dead end. In that case, v_{i+1} is called the **internal dead-end variable**.

Conflict sets are no-goods, but not all no-goods are conflict sets.

Constraint
Satisfaction
Problems

S. Wöflf,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Literature

Leaf Dead Ends, Conflict Sets, No-Goods: Example

Constraint Satisfaction Problems

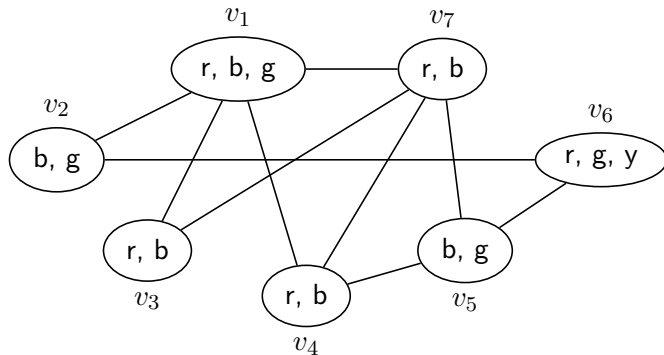
S. Wöfl, M. Helmert

Conflict Sets

Backjumping

No-Good Learning

Literature



Leaf Dead End Example

Constraint
Satisfaction
Problems

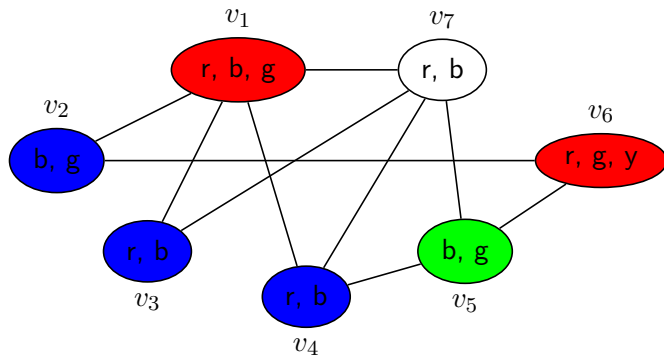
S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Literature



\rightsquigarrow a leaf dead end with leaf dead-end variable v_7

Conflict Set Example

Constraint
Satisfaction
Problems

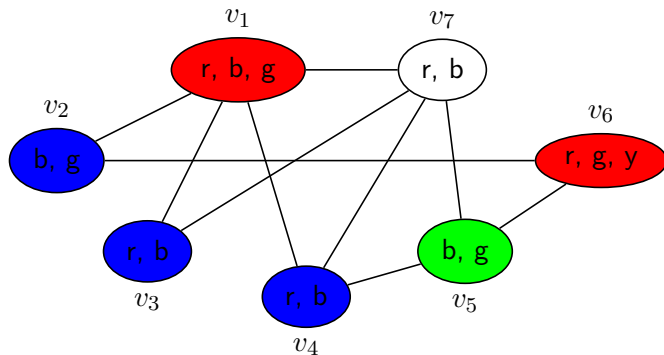
S. Wölf,
M. Helmert

Conflict Sets

Backjumping

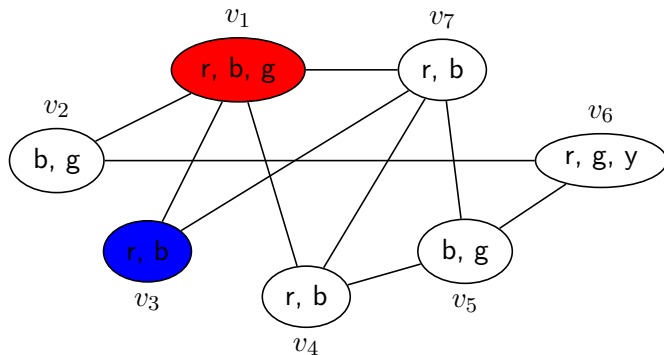
No-Good
Learning

Literature



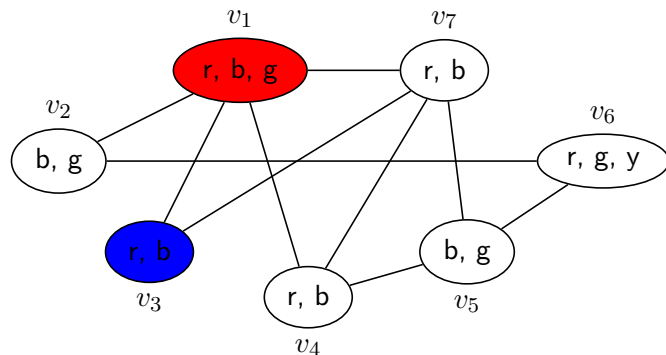
\rightsquigarrow a conflict set of v_7 , but not minimal

Conflict Set Example



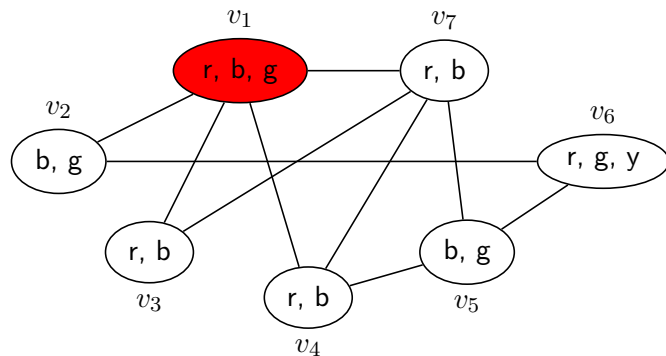
\rightsquigarrow a minimal conflict set of v_7

No-Good Example



↪ a no-good, but not a minimal one

No-Good Example



\rightsquigarrow a minimal no-good (also an internal dead end)

Definition (safe jump)

Let $a = (a_1, \dots, a_i)$ be a (leaf or internal) dead end.

We say that v_j with $j \in \{1, \dots, i\}$ is **safe** (or: a **safe jump**) relative to a if (a_1, \dots, a_j) is a no-good.

\rightsquigarrow If v_j is safe for $j < i$, we can backtrack several times and assign a new value to v_j next.

Backjumping

A **backjumping** algorithm is a modification of **backtracking** that may back up several layers in the search tree upon detecting an assignment that cannot be extended to a solution.

We study three variations:

- **Gaschnig's backjumping**
- **Graph-based backjumping**
- **Conflict-directed backjumping**

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

Literature

Gaschnig's Backjumping

We first introduce **Gaschnig's backjumping** which is one of the simplest backjumping algorithms.

It only backs up multiple layers at leaf dead ends.

Definition (culprit variable)

Let $a = (a_1, \dots, a_i)$ be a leaf dead end.

The **culprit index** relative to a is

$$\text{culp}(a) := \min\{ j \in \mathbb{N}_1 \mid (a_1, \dots, a_j) \text{ conflicts with } v_{i+1} \}$$

Gaschnig's backjumping

When detecting the leaf dead end a , jump back to $v_{\text{culp}(a)}$.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Gaschnig's Backjumping

We first introduce **Gaschnig's backjumping** which is one of the simplest backjumping algorithms.

It only backs up multiple layers at leaf dead ends.

Definition (culprit variable)

Let $a = (a_1, \dots, a_i)$ be a leaf dead end.

The **culprit index** relative to a is

$$\text{culp}(a) := \min\{ j \in \mathbb{N}_1 \mid (a_1, \dots, a_j) \text{ conflicts with } v_{i+1} \}$$

Gaschnig's backjumping

When detecting the leaf dead end a , jump back to $v_{\text{culp}(a)}$.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

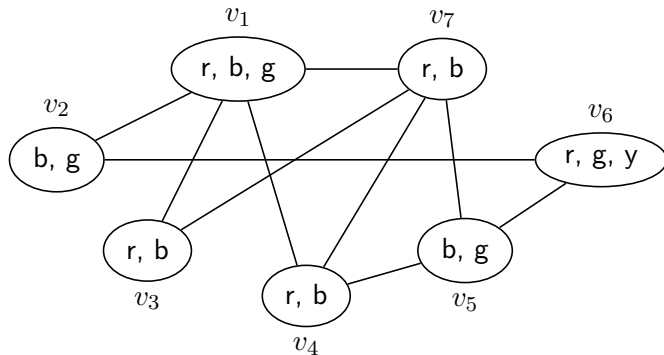
Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Gaschnig's Backjumping: Example



Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

Remarks on Gaschnig's Backjumping

- Gaschnig's backjumping was historically one of the first backjumping techniques.
- It clearly performs only **safe** jumps.
- It also performs **maximal** jumps in the sense that backing up further than Gaschnig's backjumping at leaf dead ends can lead to missing (potentially all) solutions.
- The algorithm is attractive because it is easy to implement efficiently (we do not discuss this in detail).
- However, it is not very powerful: It expands strictly more states than look-ahead search with forward checking
↪ **exercises**.
- One serious limitation is that it only jumps at leaf dead ends. The next backjumping technique will remedy this.

Constraint
Satisfaction
Problems

S. Wölf,
M. Helmert

Conflict Sets

Backjumping

**Gaschnig's
Backjumping**

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Remarks on Gaschnig's Backjumping

- Gaschnig's backjumping was historically one of the first backjumping techniques.
- It clearly performs only **safe** jumps.
- It also performs **maximal** jumps in the sense that backing up further than Gaschnig's backjumping at leaf dead ends can lead to missing (potentially all) solutions.
- The algorithm is attractive because it is easy to implement efficiently (we do not discuss this in detail).
- However, it is not very powerful: It expands strictly more states than look-ahead search with forward checking
↪ **exercises**.
- One serious limitation is that it only jumps at leaf dead ends. The next backjumping technique will remedy this.

Constraint
Satisfaction
Problems

S. Wölf,
M. Helmert

Conflict Sets

Backjumping

**Gaschnig's
Backjumping**

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Remarks on Gaschnig's Backjumping

- Gaschnig's backjumping was historically one of the first backjumping techniques.
- It clearly performs only **safe** jumps.
- It also performs **maximal** jumps in the sense that backing up further than Gaschnig's backjumping at leaf dead ends can lead to missing (potentially all) solutions.
- The algorithm is attractive because it is easy to implement efficiently (we do not discuss this in detail).
- However, it is not very powerful: It expands strictly more states than look-ahead search with forward checking
↪ **exercises**.
- One serious limitation is that it only jumps at leaf dead ends. The next backjumping technique will remedy this.

Constraint
Satisfaction
Problems

S. Wölf,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Remarks on Gaschnig's Backjumping

- Gaschnig's backjumping was historically one of the first backjumping techniques.
- It clearly performs only **safe** jumps.
- It also performs **maximal** jumps in the sense that backing up further than Gaschnig's backjumping at leaf dead ends can lead to missing (potentially all) solutions.
- The algorithm is attractive because it is easy to implement efficiently (we do not discuss this in detail).
- However, it is not very powerful: It expands strictly more states than look-ahead search with forward checking
↪ **exercises**.
- One serious limitation is that it only jumps at leaf dead ends. The next backjumping technique will remedy this.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Remarks on Gaschnig's Backjumping

- Gaschnig's backjumping was historically one of the first backjumping techniques.
- It clearly performs only **safe** jumps.
- It also performs **maximal** jumps in the sense that backing up further than Gaschnig's backjumping at leaf dead ends can lead to missing (potentially all) solutions.
- The algorithm is attractive because it is easy to implement efficiently (we do not discuss this in detail).
- However, it is not very powerful: It expands strictly more states than look-ahead search with forward checking
↪ **exercises**.
- One serious limitation is that it only jumps at leaf dead ends. The next backjumping technique will remedy this.

Constraint
Satisfaction
Problems

S. Wölf,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Remarks on Gaschnig's Backjumping

- Gaschnig's backjumping was historically one of the first backjumping techniques.
- It clearly performs only **safe** jumps.
- It also performs **maximal** jumps in the sense that backing up further than Gaschnig's backjumping at leaf dead ends can lead to missing (potentially all) solutions.
- The algorithm is attractive because it is easy to implement efficiently (we do not discuss this in detail).
- However, it is not very powerful: It expands strictly more states than look-ahead search with forward checking
↪ **exercises**.
- One serious limitation is that it only jumps at leaf dead ends. The next backjumping technique will remedy this.

Constraint
Satisfaction
Problems

S. Wölf,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Graph-Based Backjumping

- Graph-based backjumping can also jump back at **internal dead ends**.
- Unlike Gaschnig's backjumping, it does **not** use information about the values assigned to the variables in the current state when backing up.
- Instead, it only uses information about the **variables** themselves, derived from the constraint graph.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Parents

Reminder:

Definition (parents)

The **parents** of v_i are those variables v_j with $j < i$ for which the edge $\{v_i, v_j\}$ occurs in the primal constraint graph.

Definition (parents)

Let v_i be a variable with at least one parent.

The **latest parent** of v_i , in symbols $par(v_i)$, is the parent v_j for which j is maximal.

Basic idea: Jump back to the latest parent.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

Jumping back to the latest parent

Theorem

*Let a be a leaf dead end with dead-end variable v_i .
Then $\text{par}(v_i)$ is a safe jump for a .*

Proof.

Because a is a leaf dead end, (a_1, \dots, a_{i-1}) is consistent, but any extension to v_i is inconsistent. Thus (a_1, \dots, a_{i-1}) is a conflict set for v_i .

Then $(a_1, \dots, a_{\text{par}(v_i)})$ is already a conflict set for v_i , because there are no constraints between v_i and any variables v' with $\text{par}(v_i) \prec v' \prec v_i$. □

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Comparison to Gaschnig's Backjumping

- Jumping back to the latest parent of a leaf dead end is **strictly worse** than Gaschnig's Backjumping: it never jumps further, and it sometimes jumps less far.
- However, the idea can be extended to jumping from **internal dead ends**.

First idea: When encountering an internal dead end, jump back to the latest parent of the internal dead-end variable.

Unfortunately, this is **not safe**.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

Comparison to Gaschnig's Backjumping

- Jumping back to the latest parent of a leaf dead end is **strictly worse** than Gaschnig's Backjumping: it never jumps further, and it sometimes jumps less far.
- However, the idea can be extended to jumping from **internal dead ends**.

First idea: When encountering an internal dead end, jump back to the latest parent of the internal dead-end variable.

Unfortunately, this is **not safe**.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

Literature

Comparison to Gaschnig's Backjumping

- Jumping back to the latest parent of a leaf dead end is **strictly worse** than Gaschnig's Backjumping: it never jumps further, and it sometimes jumps less far.
- However, the idea can be extended to jumping from **internal dead ends**.

First idea: When encountering an internal dead end, jump back to the latest parent of the internal dead-end variable.

Unfortunately, this is **not safe**.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

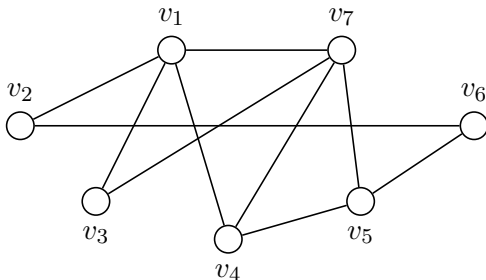
Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

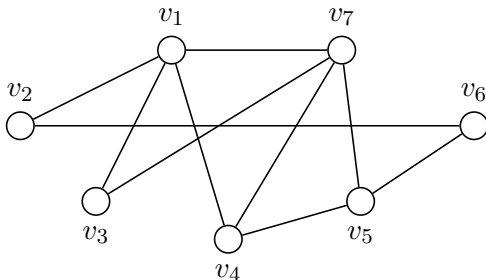
Literature

Backjumping at Internal Dead Ends: Example



- **Scenario 1:** Enter v_4 and encounter a leaf dead end with variable v_5 . Jumping back to v_4 , there are no further values for v_4 . It is then safe to backtrack to v_1 .
- **Scenario 2:** Now encounter a leaf dead end with variable v_7 . Jump back to v_5 and then to v_4 . Is it still safe to jump back to v_1 if there are no further values for v_4 ?

Backjumping at Internal Dead Ends: Example



- **Scenario 1:** Enter v_4 and encounter a leaf dead end with variable v_5 . Jumping back to v_4 , there are no further values for v_4 . It is then safe to backtrack to v_1 .
- **Scenario 2:** Now encounter a leaf dead end with variable v_7 . Jump back to v_5 and then to v_4 . Is it still safe to jump back to v_1 if there are no further values for v_4 ?

Definition (invisit, session)

We say that the backtracking algorithm **invisits** variable v_i when it attempts to extend the assignment $a = (a_1, \dots, a_{i-1})$ to v_i .

The current **session** of v_i starts when v_i is invisited and ends after all possible assignments to v_i have been tried, i. e., when the backtracking algorithm backs up to variable v_{i-1} or earlier.

Note: A session of v_i corresponds to a recursive invocation of the backtracking procedure where values are assigned to v_i .

Relevant Dead Ends

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Definition (relevant dead ends)

The **relevant dead ends** of the current session of v_i , in symbols $rel(v_i)$, are computed as follows:

- When v_i is revisited, set $rel(v_i) := \{v_i\}$.
- When v_i is reached by backing up from a later variable v_j , set $rel(v_i) := rel(v_i) \cup rel(v_j)$.

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

Graph-Based Backjumping: Algorithm

Graph-based backjumping

When detecting the (leaf or internal) dead end a with dead-end variable v_i , jump back to the **latest parent** of **any** variable in $rel(v_i)$ which is earlier than v_i .

Theorem (Soundness)

Graph-based backjumping only performs safe jumps.

Proof.

⇒ exercises



Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Graph-Based Backjumping: Algorithm

Graph-based backjumping

When detecting the (leaf or internal) dead end a with dead-end variable v_i , jump back to the **latest parent** of **any** variable in $rel(v_i)$ which is earlier than v_i .

Theorem (Soundness)

Graph-based backjumping only performs safe jumps.

Proof.

↪ exercises



Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

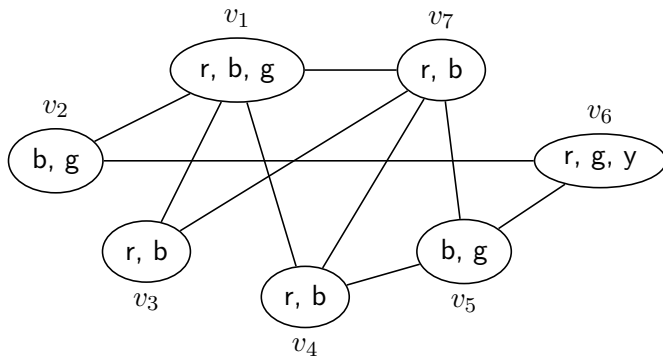
Backjumping

No-Good

Learning

Literature

Graph-Based Backjumping: Example



Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

Conflict-Directed Backjumping

- Gaschnig's backjumping exploits the information about a particular **minimal prefix conflict set** to jump further from leaf dead ends.
- Graph-based backjumping collects and integrates information from all dead ends in the current session to also jump back at internal dead ends.
- These two ideas can be combined to obtain the **conflict-directed backjumping** algorithm, which is better (avoids more states) than either of the two previous backjumping styles.

Constraint
Satisfaction
Problems

S. Wölfel,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

Literature

Constraint Ordering

Definition (earlier constraint)

Let v_1, \dots, v_n be a variable ordering, and let Q and R be two constraints. We say that Q is **earlier** than R according to the ordering, in symbols $Q \prec R$ if

- $scope(Q) \subset scope(R)$, or
- $scope(Q) \not\subseteq scope(R)$ and $scope(R) \not\subseteq scope(Q)$ and the latest variable in $scope(Q) \setminus scope(R)$ precedes the latest variable in $scope(R) \setminus scope(Q)$.

If we assume that any two constraints have different scopes, this defines a total order on constraints.

Greedy Conflict Sets

Definition (greedy conflict set)

Let a be a (leaf or internal) dead end with dead-end variable v . For all $x \in \text{dom}(v)$, define V_x as follows:

- If $a \cup \{v \mapsto x\}$ is inconsistent, let V_x be the scope of the earliest constraint which is not satisfied by $a \cup \{v \mapsto x\}$.
- Otherwise, $V_x := \emptyset$.

The **greedy conflict variable set** of a , in symbols $gcv(a)$, is defined as $gcv(a) := \bigcup_{x \in \text{dom}(v)} (V_x \setminus \{v\})$.

The **greedy conflict set** of a , in symbols $gc(a)$, is defined as $gc(a) := \{v \mapsto a(v) \mid v \in gcv(a)\}$.

In other words, $gc(a)$ is a restricted to the greedy conflict variable set.

Greedy Conflict Sets are Conflict Sets

Theorem

*Let a be a leaf dead end with dead-end variable v .
Then $gc(a)$ is a conflict set of v .*

Proof.

Since a is a leaf dead end, it is a partial solution. Moreover, $gc(a)$ is a sub-assignment of a , so it is not defined for v . We show that no assignment $gc(a) \cup \{v \mapsto x\}$ is consistent. Consider an arbitrary value $x \in \text{dom}(v)$. In a leaf dead-end, there must be a constraint R_x with scope V_x which is not satisfied by $a \cup \{v \mapsto x\}$. Then $gcv(a)$ includes all variables in $V_x \setminus \{v\}$, and thus $gc(a)$ is defined and equal to a on these variables. As $a \cup \{v \mapsto x\}$ does not satisfy R_x , $gc(a) \cup \{v \mapsto x\}$ does not satisfy R_x either. Thus, $gc(a)$ cannot be consistently extended to v and hence is a conflict set for v . \square

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

Literature

Greedy Conflict Sets are Conflict Sets

Theorem

*Let a be a leaf dead end with dead-end variable v .
Then $gc(a)$ is a conflict set of v .*

Proof.

Since a is a leaf dead end, it is a partial solution. Moreover, $gc(a)$ is a sub-assignment of a , so it is not defined for v . We show that no assignment $gc(a) \cup \{v \mapsto x\}$ is consistent. Consider an arbitrary value $x \in \text{dom}(v)$. In a leaf dead-end, there must be a constraint R_x with scope V_x which is not satisfied by $a \cup \{v \mapsto x\}$. Then $gcv(a)$ includes all variables in $V_x \setminus \{v\}$, and thus $gc(a)$ is defined and equal to a on these variables. As $a \cup \{v \mapsto x\}$ does not satisfy R_x , $gc(a) \cup \{v \mapsto x\}$ does not satisfy R_x either. Thus, $gc(a)$ cannot be consistently extended to v and hence is a conflict set for v . \square

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

Literature

Greedy Conflict Sets are Conflict Sets

Theorem

*Let a be a leaf dead end with dead-end variable v .
Then $gc(a)$ is a conflict set of v .*

Proof.

Since a is a leaf dead end, it is a partial solution. Moreover, $gc(a)$ is a sub-assignment of a , so it is not defined for v . We show that no assignment $gc(a) \cup \{v \mapsto x\}$ is consistent. Consider an arbitrary value $x \in \text{dom}(v)$. In a leaf dead-end, there must be a constraint R_x with scope V_x which is not satisfied by $a \cup \{v \mapsto x\}$. Then $gcv(a)$ includes all variables in $V_x \setminus \{v\}$, and thus $gc(a)$ is defined and equal to a on these variables. As $a \cup \{v \mapsto x\}$ does not satisfy R_x , $gc(a) \cup \{v \mapsto x\}$ does not satisfy R_x either. Thus, $gc(a)$ cannot be consistently extended to v and hence is a conflict set for v . \square

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping
Graph-Based
Backjumping
Conflict-Directed
Backjumping

No-Good
Learning

Literature

Minimality of Greedy Conflict Sets

- Dechter calls $gc(a)$ the **earliest minimal conflict set** of a .
- However, it is not always a minimal conflict set and not always the earliest conflict set that is a subassignment of a , so we avoid this terminology.

Note: The greedy conflict set is only a conflict set for **leaf dead ends!**

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Greedy Conflict Sets vs. Gaschnig's Backjumping

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Reminder:

- Gaschnig's backjumping jumps back to $v_{culp(a)}$, where $culp(a) := \min\{j \in \mathbb{N}_1 \mid (a_1, \dots, a_j) \text{ conflicts with } v\}$

Observations:

- For the greedy variable set, the latest variable in $gcv(a)$ always equals $culp(a)$.
- Thus, jumping from leaf dead ends to the latest variable in $gcv(a)$ is the same as Gaschnig's backjumping.

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Greedy Conflict Sets vs. Graph-Based Backjumping

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Observations:

- All variables in $gcv(a)$ are parents of the leaf dead end variable of a .

Idea:

- Instead of considering **all parents** of relevant dead-end variables (as in graph-based backjumping), consider **all greedy conflict sets** of relevant dead ends.
- Using this scheme, jumping from internal dead ends jumps at least as far as graph-based backjumping.

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

Jump-Back Sets

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Definition (jump-back set)

The **jump-back set** of a dead end a , in symbols J_a , is defined as follows:

- If a is a leaf dead end, $J_a := gcv(a)$.
- If a is an internal dead end, $J_a := gcv(a) \cup \bigcup_{a' \in succ(a)} J_{a'}$, where $succ(a)$ is the set of successor states of a .

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed
Backjumping

No-Good

Learning

Literature

Conflict-Directed Backjumping: Algorithm

Conflict-directed backjumping

When detecting the (leaf or internal) dead end a with dead-end variable v_i , jump back to the latest variable in J_a that is earlier than v_i .

Theorem (Soundness)

Conflict-directed backjumping only performs safe jumps.

Proof idea.

Combine the proofs for Gaschnig's backjumping and graph-based backjumping. □

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Conflict-Directed Backjumping: Algorithm

Conflict-directed backjumping

When detecting the (leaf or internal) dead end a with dead-end variable v_i , jump back to the latest variable in J_a that is earlier than v_i .

Theorem (Soundness)

Conflict-directed backjumping only performs safe jumps.

Proof idea.

Combine the proofs for Gaschnig's backjumping and graph-based backjumping. □

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's
Backjumping

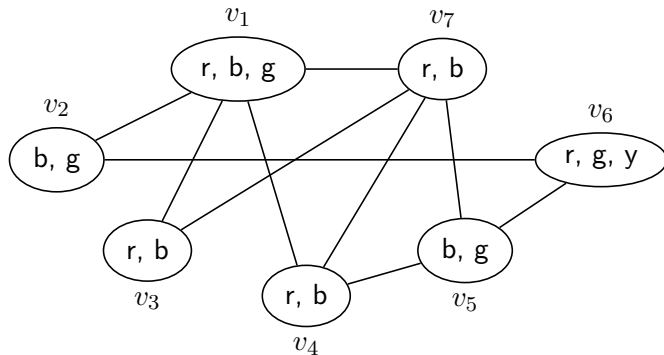
Graph-Based
Backjumping

Conflict-Directed
Backjumping

No-Good
Learning

Literature

Conflict-Directed Backjumping: Example



Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

Gaschnig's

Backjumping

Graph-Based

Backjumping

Conflict-Directed

Backjumping

No-Good

Learning

Literature

No-Good Learning

- Backjumping can significantly reduce the search effort by skipping over irrelevant choice points.
- However, **thrashing** is still possible: essentially the same no-good can be “rediscovered” over and over in different parts of the search tree.
- To alleviate this problem, we can make use of **no-good learning** or **constraint recording** techniques.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

Adding No-Good Learning

Adding no-good learning to an existing (backtracking, look-ahead, backjumping, ...) algorithm is simple:

no-good learning

When the algorithm backtracks (or jumps back), determine a conflict set and **add a constraint** to the network that **rules out this conflict set**.

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

Variations of No-Good Learning

There are many variations:

- How to determine the no-good?
 - Determine one which is **easy to generate**, but not necessarily minimal \rightsquigarrow **shallow learning**.
 - Determine one which is **minimal**, or even **all minimal ones** derivable from the current dead end \rightsquigarrow **deep learning**
- Which no-goods to store?
 - Store all constraints.
 - Store only **small** no-goods (constraints with arity $\leq c$)
 \rightsquigarrow **bounded learning**
- How long to store no-goods?
 - Store forever.
 - Discard once they differ from the current state in more than c variables
 \rightsquigarrow **relevance-bounded learning**

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

Variations of No-Good Learning

There are many variations:

- How to determine the no-good?
 - Determine one which is **easy to generate**, but not necessarily minimal \rightsquigarrow **shallow learning**.
 - Determine one which is **minimal**, or even **all minimal ones** derivable from the current dead end \rightsquigarrow **deep learning**
- Which no-goods to store?
 - Store all constraints.
 - Store only **small** no-goods (constraints with arity $\leq c$) \rightsquigarrow **bounded learning**
- How long to store no-goods?
 - Store forever.
 - Discard once they differ from the current state in more than c variables \rightsquigarrow **relevance-bounded learning**

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

Variations of No-Good Learning

There are many variations:

- How to determine the no-good?
 - Determine one which is **easy to generate**, but not necessarily minimal \rightsquigarrow **shallow learning**.
 - Determine one which is **minimal**, or even **all minimal ones** derivable from the current dead end \rightsquigarrow **deep learning**
- Which no-goods to store?
 - Store all constraints.
 - Store only **small** no-goods (constraints with arity $\leq c$)
 \rightsquigarrow **bounded learning**
- How long to store no-goods?
 - Store forever.
 - Discard once they differ from the current state in more than c variables
 \rightsquigarrow **relevance-bounded learning**

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

No-Good Learning: Issues

When performing no-good learning, there is a need to strike a good compromise between:

- **pruning power:**
more constraints lead to fewer explored states
- **constraint processing overhead:**
learning many constraints increases the satisfaction tests for every search node
- **learning overhead:**
expensive computations of no-goods may outweigh pruning benefits
- **space overhead:**
storing all no-goods eliminates the space efficiency of backtracking-style algorithms

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning
Concepts
Algorithms

Literature

Graph-Based Learning

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Graph-based learning

Augment **graph-based backjumping** by applying the following learning rule when jumping back from an internal or leaf dead-end a with dead-end variable v_i :

- Let $V(a)$ be the set of parents of some variable in the relevant dead-end variable set $rel(v_i)$.
- Learn the no-good $\{(v, a(v)) \mid v \in V(a) \text{ and } v \prec v_i\}$.

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

Conflict-Directed Backjump Learning

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Concepts
Algorithms

Literature

Conflict-directed backjump learning

Augment **conflict-directed backjumping** by applying the following learning rule when jumping back from an internal or leaf dead-end a with dead-end variable v_i :

- Learn the no-good $\{(v, a(v)) \mid v \in gcv(a) \text{ and } v \prec v_i\}$.

Nonsystematic Randomized Backtrack Learning

- Learning algorithms are not limited to minor variations of the common systematic backtracking algorithms.
- One example of a very different algorithm is **nonsystematic randomized backtrack learning**:
 - Use backtracking with random variable and value orders.
 - At each dead end, learn a new conflict set.
 - After a certain number of dead ends, restart (remembering the newly learned constraints).
 - Terminate upon solution or when \emptyset becomes a dead end.

Completeness:

- Each newly learned constraint reduces the number of states in the state space by at least 1.
- Thus, eventually either the empty assignment will be a dead end, or the search space will become backtrack-free.

Nonsystematic Randomized Backtrack Learning

- Learning algorithms are not limited to minor variations of the common systematic backtracking algorithms.
- One example of a very different algorithm is **nonsystematic randomized backtrack learning**:
 - Use backtracking with random variable and value orders.
 - At each dead end, learn a new conflict set.
 - After a certain number of dead ends, restart (remembering the newly learned constraints).
 - Terminate upon solution or when \emptyset becomes a dead end.

Completeness:

- Each newly learned constraint reduces the number of states in the state space by at least 1.
- Thus, eventually either the empty assignment will be a dead end, or the search space will become backtrack-free.

Nonsystematic Randomized Backtrack Learning

- Learning algorithms are not limited to minor variations of the common systematic backtracking algorithms.
- One example of a very different algorithm is **nonsystematic randomized backtrack learning**:
 - Use backtracking with random variable and value orders.
 - At each dead end, learn a new conflict set.
 - After a certain number of dead ends, restart (remembering the newly learned constraints).
 - Terminate upon solution or when \emptyset becomes a dead end.

Completeness:

- Each newly learned constraint reduces the number of states in the state space by at least 1.
- Thus, eventually either the empty assignment will be a dead end, or the search space will become backtrack-free.



Rina Dechter.
Constraint Processing,
Chapter 6, Morgan Kaufmann, 2003

Constraint
Satisfaction
Problems

S. Wöfl,
M. Helmert

Conflict Sets

Backjumping

No-Good
Learning

Literature