Computing Explanations in Problem Solving

A Review of Formal Approaches

Barry O'Sullivan[1] Ulrich Junker[2]

[1]**Cork Constraint Computation Centre**
Department of Computer Science, University College Cork, Ireland
b.osullivan@cs.ucc.ie

[2]**ILOG, An IBM Company**
Sophie Antipolis, France
uli.junker@free.fr

IJCAI 2009 Tutorial Programme

Where can you get the slides?

Tutorial web-site

`http://www.cs.ucc.ie/~osullb/ijcai-tutorial-2009/`

Acknowledgements

Science Foundation Ireland
Grant 05/IN/I886.

COST Action IC0602 on
Algorithmic Decision Theory.

We would also like to thank our colleagues

Industrial Collaborator: David Lesaint (British Telecom)

Colleagues at 4C: Tarik Hadzic, Deepak Mehta, Alexandre Papadopoulos, Luis Quesada, and Nic Wilson.

Outline

Outline

What is this tutorial about?

Example

In November 2003, a client had the problem that constraint propagation in their configurator was failing for a system described by $300,000$ constraints.

How do we debug this?

There are $2^{300,000}$ possible causes, but in our example, only 8 of the constraints were sufficient to produce the failure, but there are still $> 10^{39}$ combinations of possibilities.

After this tutorial you will how to . . .

Identify these 8 constraints after only 270 consistency checks!

What is this tutorial about?

Example

In November 2003, a client had the problem that constraint propagation in their configurator was failing for a system described by $300,000$ constraints.

How do we debug this?

There are $2^{300,000}$ possible causes, but in our example, only 8 of the constraints were sufficient to produce the failure, but there are still $> 10^{39}$ combinations of possibilities.

After this tutorial you will how to . . .

Identify these 8 constraints after only 270 consistency checks!

What is this tutorial about?

Example

In November 2003, a client had the problem that constraint propagation in their configurator was failing for a system described by $300,000$ constraints.

How do we debug this?

There are $2^{300,000}$ possible causes, but in our example, only 8 of the constraints were sufficient to produce the failure, but there are still $> 10^{39}$ combinations of possibilities.

After this tutorial you will how to . . .

Identify these 8 constraints after only 270 consistency checks!

Where can I apply what I learn?

1. Product Configuration
2. Test Generation
3. Recommender Systems
4. Case-based Reasoning Systems
5. Knowledge-based Systems
6. Software Product Lines
7. Debugging
8. Can you think of any others?

What are Explanations?

Two forms of explanation in artificial intelligence [13]

1. Explanations as part of the reasoning process – used in the search for a diagnostic result in order to support a particular hypothesis.
2. Explanations that attempt to make the reasoning process, its results, or the usage of the result understandable to the user.

We will focus on the latter form of explanation.

Let's consider an example

Pediatrics

1. A doctor needs to explain why a 12 week old baby needs to be kept in hospital for observation because of an infection.
2. The doctor has a premature baby and a 14 week old baby with a similar infection.

Challenge

How can the doctor give a convincing explanation for why the 12 week old needs to be kept in hospital?

Let's consider an example

Pediatrics

1. A doctor needs to explain why a 12 week old baby needs to be kept in hospital for observation because of an infection.
2. The doctor has a premature baby and a 14 week old baby with a similar infection.

Challenge

How can the doctor give a convincing explanation for why the 12 week old needs to be kept in hospital?

Let's consider an example

Is this a good explanation?

"We have a premature baby with a similar condition, therefore we think your 12 week old baby should also stay in hospital."

Let's consider an example

Is this a good explanation?

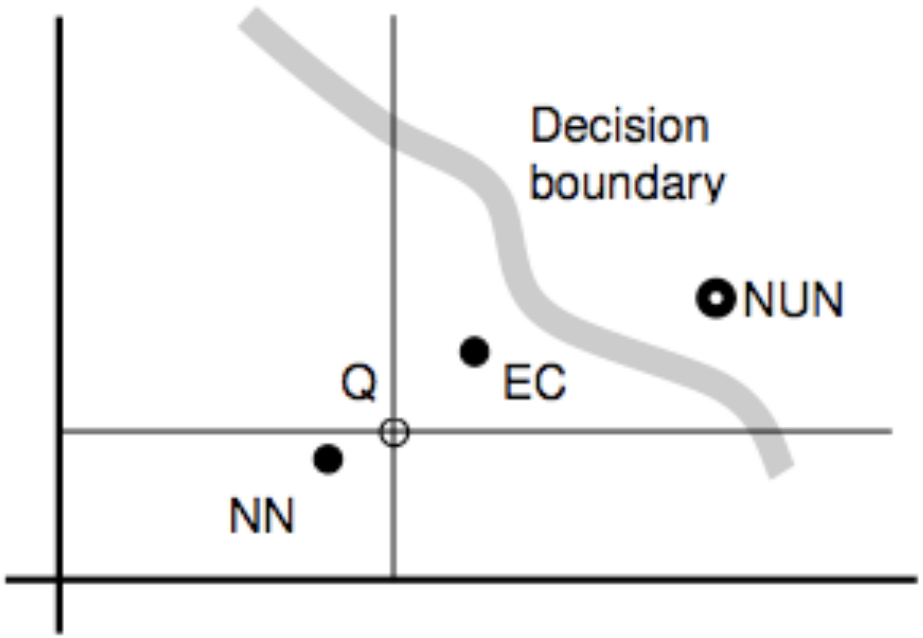
"Your 12 week old baby needs to be kept in hospital because there is a 14 week old baby (older and stronger) who has a similar condition and is being kept in hospital."

What is the principle at play here?

Explanation in decision making

- We are explaining a decision.
- In AI, a decision problem has a reasonably well defined decision boundary.
- Choosing a explanatory case that is closer to the decision boundary that the case we have at hand makes intuitive sense.

Formalising this Example



From Doyle et al. ECCBR-04 [4]:

- $Q =$ query case
- $NN =$ nearest neighbour
- $EC =$ explanation case
- $NUC =$ nearest unlike neighbour

Similarity between experiences

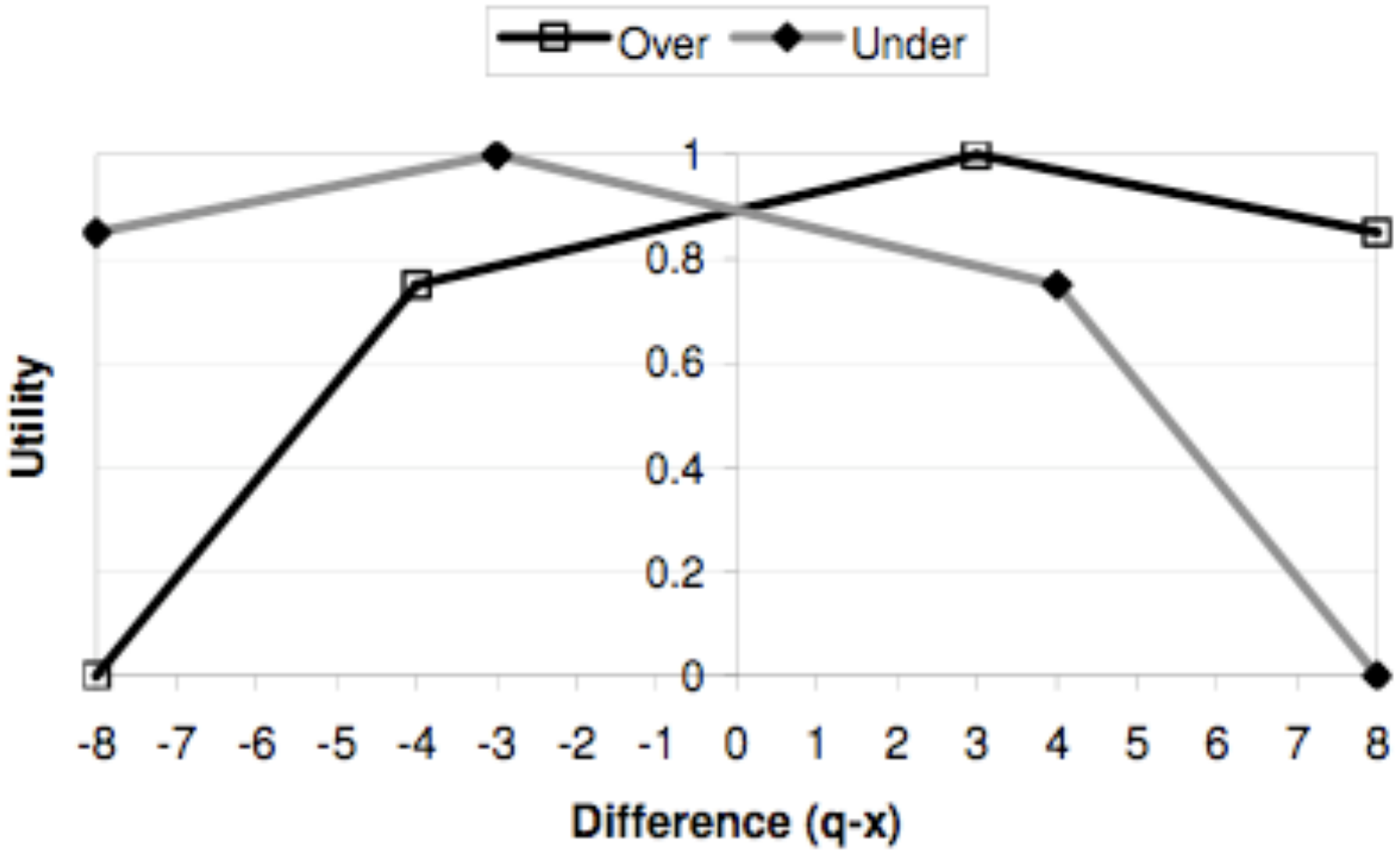
- Let's represent an 'experience' as a vector of features F.
- We can define the similarity between experience x and y as follows:

$$sim(x,y) =_{def} \sum_{f \in F} w_f \times \sigma_f(x[f], y[f])$$

where w_f is the weight of feature f, and σ_f is measure of the similarity between values for feature f.

- Example: 10 is more similar to 20 than 100

Explanation Utility [4]



Explanations - Blood Alcohol [4]

	Target Case (Q_l)	*Nearest Neighbour* (NN_l)	*Explanation Case* (EC_l)
Weight (Kgs)	82	82	73
Duration (mins)	60	60	60
Gender	Male	Male	Male
Meal	Full	Full	Full
Units Consumed	2.9	2.6	5.2
BAC	Under	Under	Under

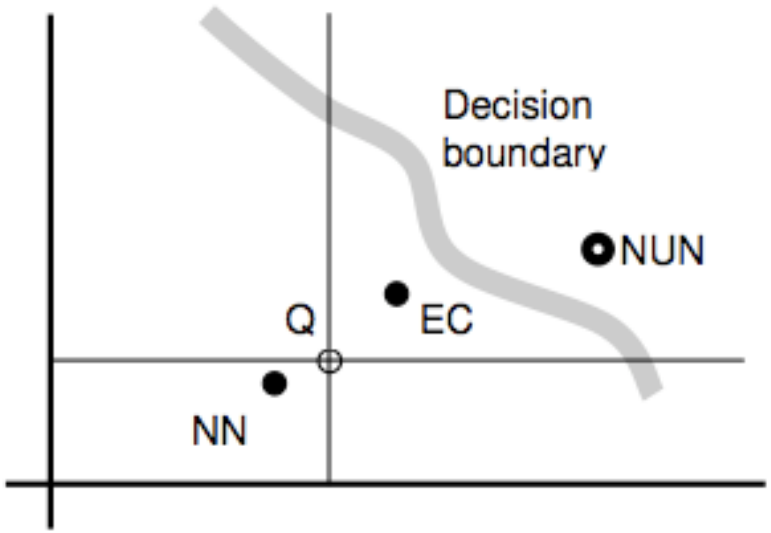
Explanation Case: Select k nearest neighbours and sort by explanation utility.

Explanations - Blood Alcohol [4]

	Target Case (Q_2)	*Nearest Neighbour* (NN_2)	*Explanation Case* (EC_2)
Weight (Kgs)	73	76	79
Duration (mins)	240	240	240
Gender	Male	Male	Male
Meal	Full	Full	Full
Units Consumed	12.0	12.4	9.6
BAC	Over	Over	Over

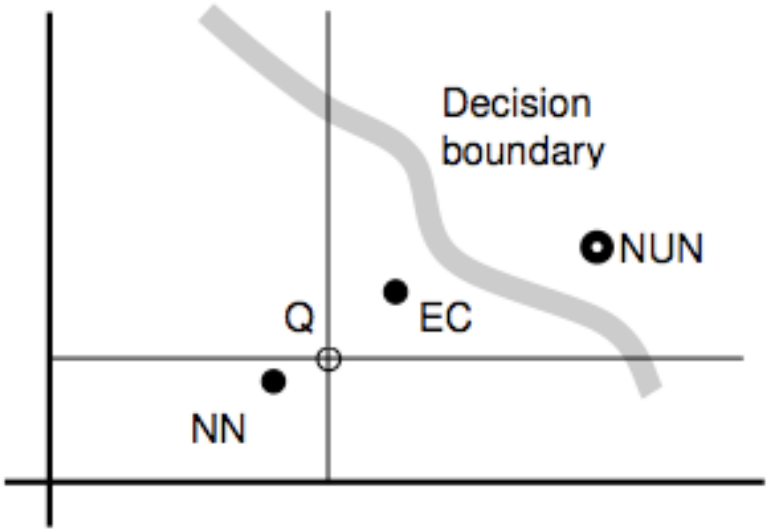
Explanation Case: Select k nearest neighbours and sort by explanation utility.

Explanations and Relaxation



- We need to reason about the 'direction' to a decision boundary.
- In the example, we say how this could be encoded using ad-hoc explanation utility functions.
- But we are simply assuming a well defined relaxation space for each feature of our decision.

Explanations and Relaxation



- ...but when we have constraints, a natural decision boundary is the unsatisfiable-satisfiable boundary, and constraint relaxation defines the relaxation space.

Let's focus on configuration [12]

A Running Example Domain

To make our examples and methods concrete, we will focus on configuration as a domain for the rest of the tutorial.

Why configuration?

- Almost every AI technique we know has been applied to product configuration.
- Richness: NP-Complete class, preferences, interactivity, scalability.
- Familiarity.

Let's focus on configuration [12]

A Running Example Domain

To make our examples and methods concrete, we will focus on configuration as a domain for the rest of the tutorial.

Why configuration?

- Almost every AI technique we know has been applied to product configuration.
- Richness: NP-Complete class, preferences, interactivity, scalability.
- Familiarity.

What is Configuration?

Definition

Configuration is the task of composing a customised system out of generic components from a catalogue, satisfying user constraints and preferences.

Examples

- Computer systems
- Automobiles
- Kitchens
- Holidays
- Software systems (a detailed example later)

What is Configuration?

Configuration Problem

- A catalogue which describes the generic components in terms of their functional and technical properties and the relationship between both.
- User requirements and user preferences about the functional characteristics of the desired configuration.

What is Configuration?

My Preferences

Frame Type

- City Bike
- Grandma Bike
- Mountain Bike
- Racer Bike

Frame Type

- Female
- Male

My Height

Color

Yellow

Parts

Frame	Colibri Street Bike Plus
Size	
Gear	
Speeds	
Rims	
Width	
Tires	
Profile	
Pedals	PD M545

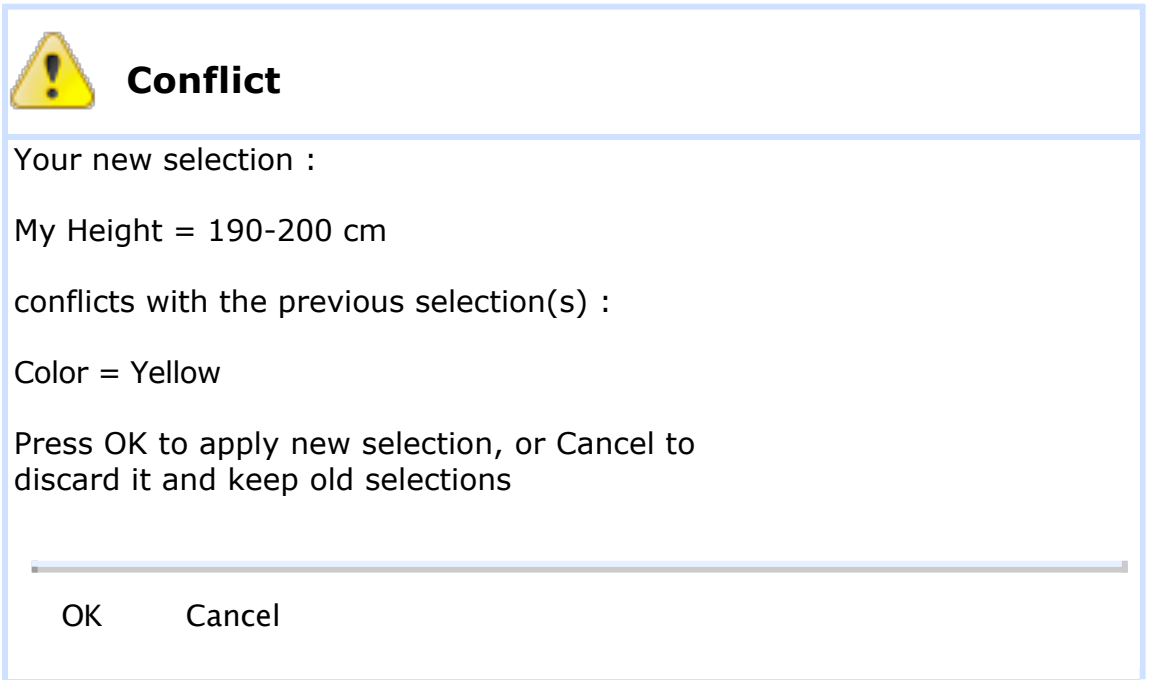
From Configit, Web `http://www.configit.com`.

What is Configuration?

Configuration Task

- One or more configurations that satisfy all requirements and that optimize the preferences if those requirements are consistent.
- An explanation of failure otherwise.

What is Configuration?



From Configit, Web `http://www.configit.com`.

Configuration Techniques

1. Rule-based reasoning
2. Model-based reasoning
   - Description Logics
   - Constraint Satisfaction
   - Resource Models
   - Multi-valued Decision Diagrams
   - Satisfiability – Negation Normal Form representations
3. Ontologies

Gartner

Model-based, and in particular constraint-based approaches are most successful in practice: declarative, maintainable, etc.

Configuration Techniques

1. Rule-based reasoning
2. Model-based reasoning
   - Description Logics
   - Constraint Satisfaction
   - Resource Models
   - Multi-valued Decision Diagrams
   - Satisfiability – Negation Normal Form representations
3. Ontologies

Gartner

Model-based, and in particular constraint-based approaches are most successful in practice: declarative, maintainable, etc.

Configuration Techniques

Key Configurator Capabilities

1. Generation of components to carry out the functional requirements.
2. Reasoning about the interactions of multiple components.
3. Detection and explanation of cases where the desired functionality cannot be implemented.

Modelling Challenges for Configuration

Product Catalogue Integration & Integrity

Maintaining integration between product catalogues and constraint-based configuration models.

Knowledge Representation

Constraint-based approaches need to be able to handle taxonomic inheritance properly. How do we properly handled unbounded configuration spaces with resource restrictions.

Preference Models

Users have preferences, unfortunately.

Modelling Challenges for Configuration

Product Catalogue Integration & Integrity

Maintaining integration between product catalogues and constraint-based configuration models.

Knowledge Representation

Constraint-based approaches need to be able to handle taxonomic inheritance properly. How do we properly handled unbounded configuration spaces with resource restrictions.

Preference Models

Users have preferences, unfortunately.

Modelling Challenges for Configuration

Product Catalogue Integration & Integrity

Maintaining integration between product catalogues and constraint-based configuration models.

Knowledge Representation

Constraint-based approaches need to be able to handle taxonomic inheritance properly. How do we properly handled unbounded configuration spaces with resource restrictions.

Preference Models

Users have preferences, unfortunately.

Reasoning Challenges for Configuration

Top-down Refinement

Solver strategy is restricted somewhat: configure components before subcomponents.

Component Generation

Number of components can be unbounded, there can be sharing of function, isomorphisms, etc.

Scalability and Explanability

Solve many similar problems, response times, etc. We need to generate explanations of conflicts. Preferences are also important.

Reasoning Challenges for Configuration

Top-down Refinement

Solver strategy is restricted somewhat: configure components before subcomponents.

Component Generation

Number of components can be unbounded, there can be sharing of function, isomorphisms, etc.

Scalability and Explanability

Solve many similar problems, response times, etc. We need to generate explanations of conflicts. Preferences are also important.

Reasoning Challenges for Configuration

Top-down Refinement

Solver strategy is restricted somewhat: configure components before subcomponents.

Component Generation

Number of components can be unbounded, there can be sharing of function, isomorphisms, etc.

Scalability and Explanability

Solve many similar problems, response times, etc. We need to generate explanations of conflicts. Preferences are also important.

Knowledge Representation and Reasoning

Our Working Assumptions

We will assume a constraint-based representation of our decision problem, over which we assume an efficient propagation method to reason about our decision boundary.

Constraint Satisfaction Problem

A CSP is a defined by a triple (X, D, C):

- $X = \{x_1, \ldots, x_n\}$ variables
- $D = \{D_1, \ldots, D_n\}$ domains
- $C = \{c_1, \ldots, c_m\}$ constraints
- Sol solution space, all valid configurations

Knowledge Representation and Reasoning

Our Working Assumptions

We will assume a constraint-based representation of our decision problem, over which we assume an efficient propagation method to reason about our decision boundary.

Constraint Satisfaction Problem

A CSP is a defined by a triple (X, D, C):

- $X = \{x_1, \ldots, x_n\}$ variables
- $D = \{D_1, \ldots, D_n\}$ domains
- $C = \{c_1, \ldots, c_m\}$ constraints
- Sol solution space, all valid configurations

Efficient Reasoning through Compilation

Compiling Sol

Compiled representations of Sol useful since they guarantee the efficient execution of queries that support explanation generation in domains such as product configuration.

Compilation Target

A class of DAGs often used: *deterministic finite-state automata* (DFAs), *multi-valued decision diagrams* (MDDs), and *binary decision diagrams* (BDDs)

Efficient Reasoning through Compilation

Compiling Sol

Compiled representations of Sol useful since they guarantee the efficient execution of queries that support explanation generation in domains such as product configuration.

Compilation Target

A class of DAGs often used: *deterministic finite-state automata* (DFAs), *multi-valued decision diagrams* (MDDs), and *binary decision diagrams* (BDDs)

T-Shirt MDD

Example

- Variables: $X = \{x_1, x_2, x_3\}$ for *color, size, print*
- Domains:
  - $D_1 = \{black, white, red, blue\}$,
  - $D_2 = \{small, medium, large\}$,
  - $D_3 = \{MIB, STW\}$.
- Constraints:
  - $f_1 : (x_3 = MIB) \Rightarrow (x_1 = black)$
  - $f_2 : (x_3 = STW) \Rightarrow (x_2 \neq small)$

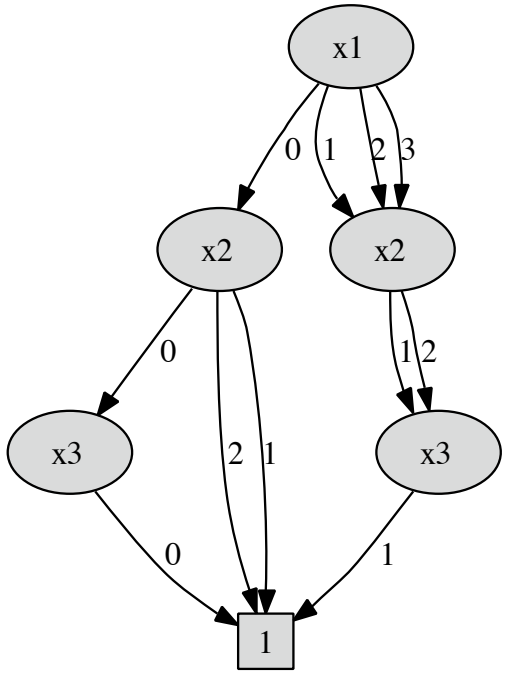


Figure: (Reduced Ordered) MDD

Decision Diagrams

Definition (Decision Diagram)

A *decision diagram* is a rooted directed acyclic graph $G = (V, E)$ where every node u is labeled with a variable x_i and every edge e, originating from a node labeled x_i, is labeled with a value $a_i \in D_i$. No node may have more than one outgoing edge with the same label. The decision diagram contains a special *terminal* node **1**, that has no outgoing edges. The terminal node has to be reachable by every other node in V.

If all domains D_i are binary, i.e. $D_1 = \ldots = D_n = \{0, 1\}$, then we have a *binary decision diagram* (BDD), otherwise we have a *multi-valued decision diagram* (MDD).

Exploiting Isomorphism

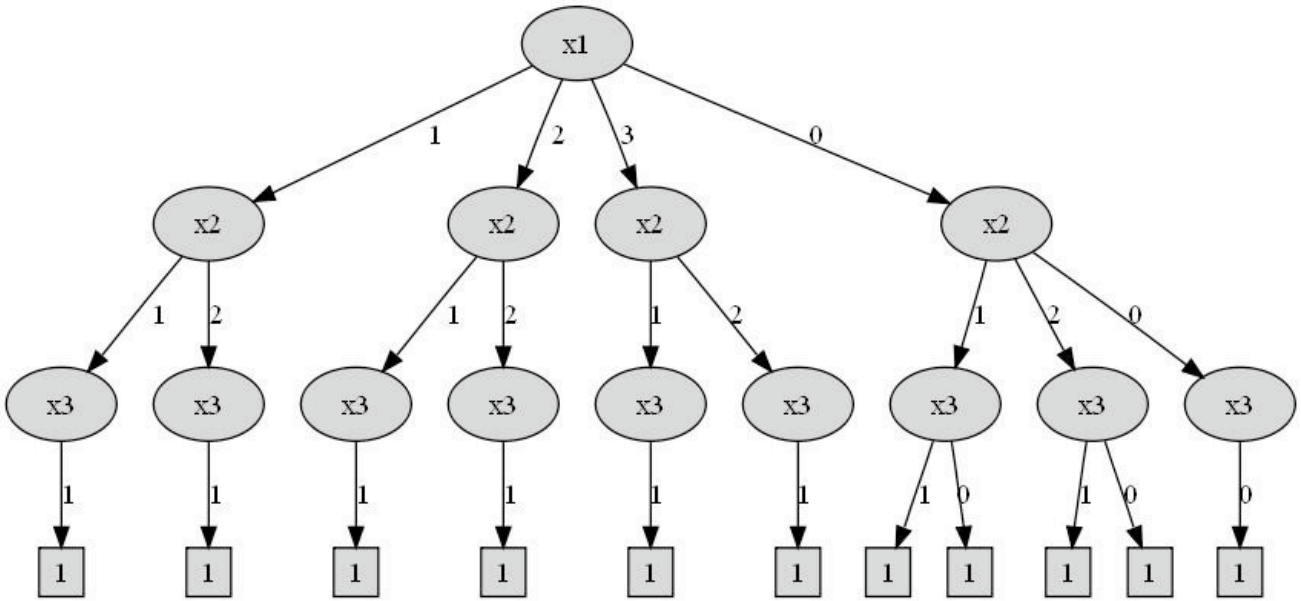


Figure: T-Shirt Decision Tree

Isomorphism

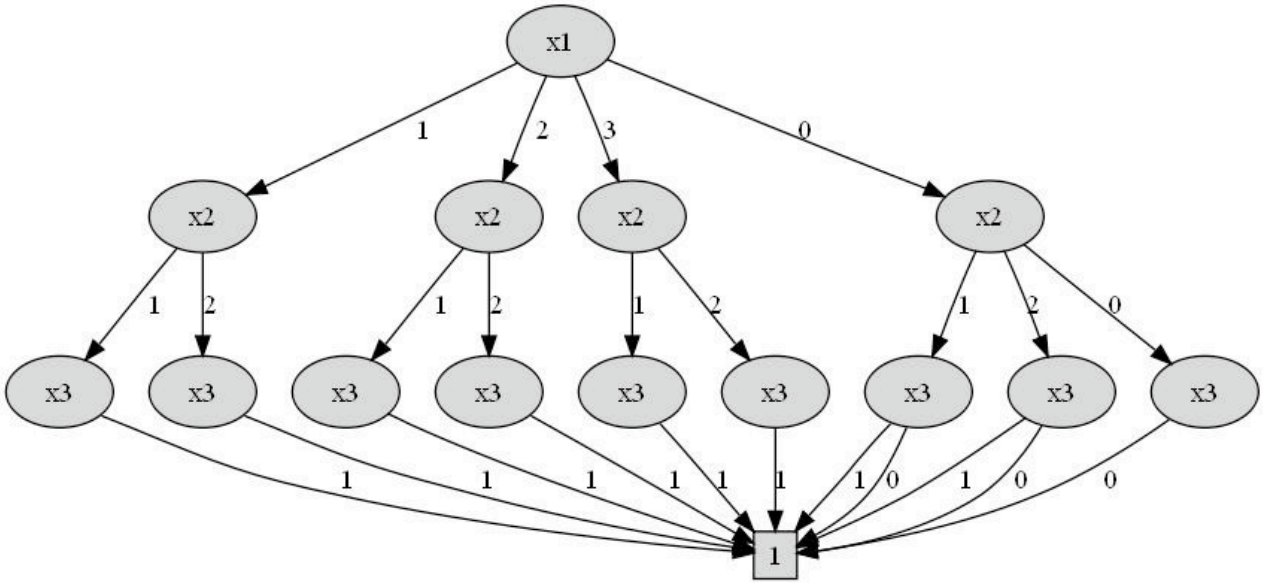


Figure: Merging Isomorphic Nodes

Isomorphism

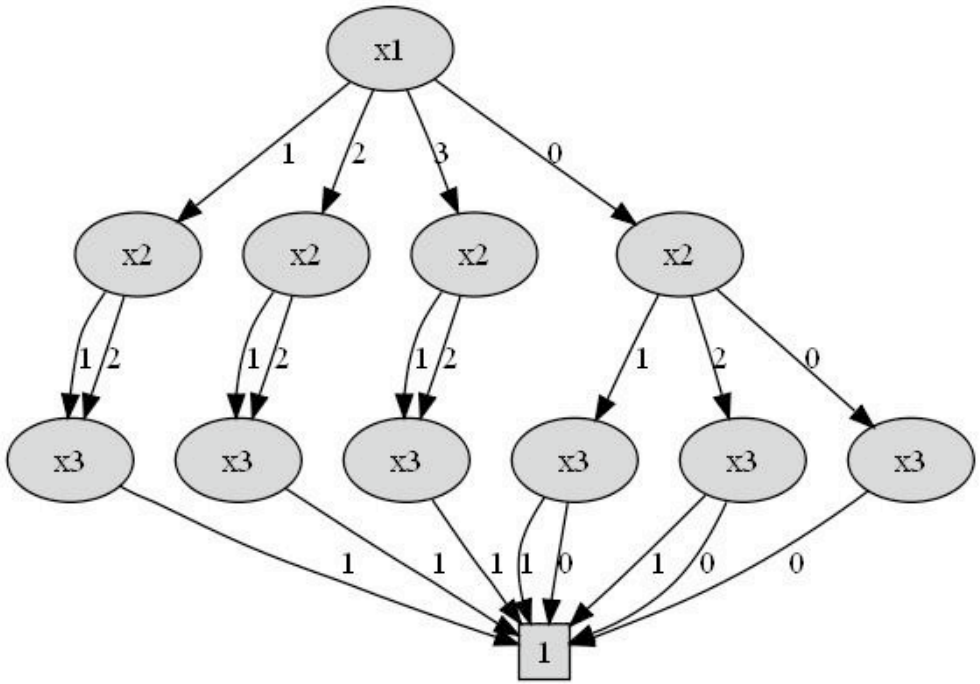


Figure: Merging Isomorphic Nodes

Isomorphism

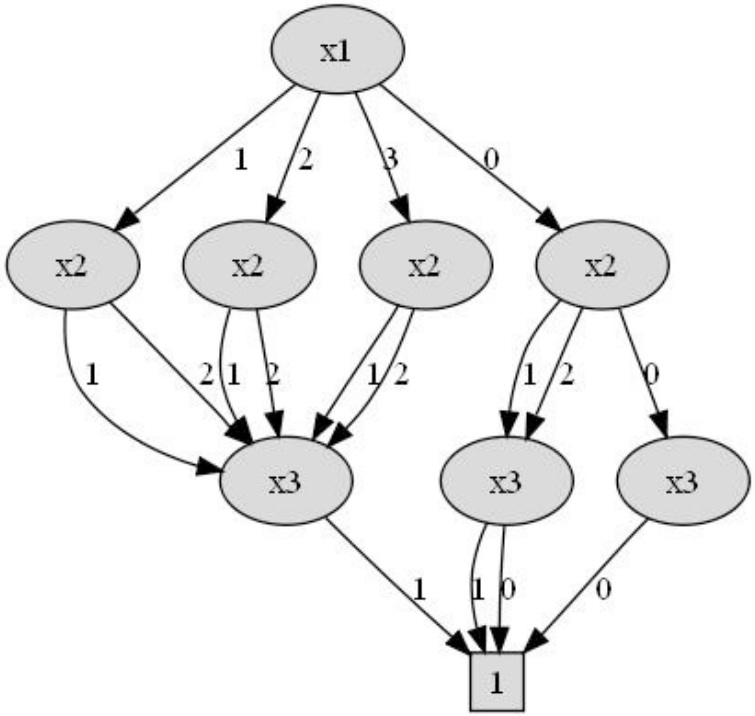


Figure: Merging Isomorphic Nodes

Isomorphism

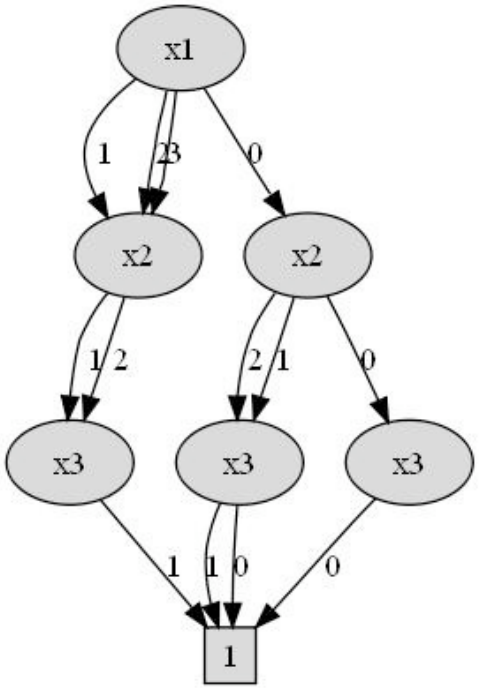


Figure: Merging Isomorphic Nodes

Merged MDD

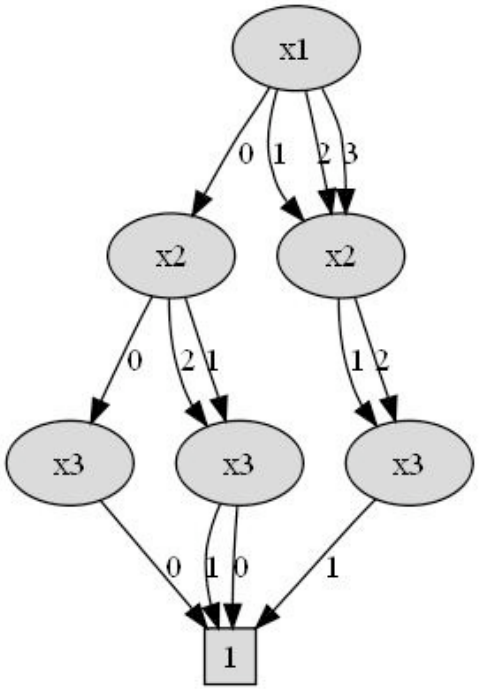


Figure: Merged MDD

Reduced MDD

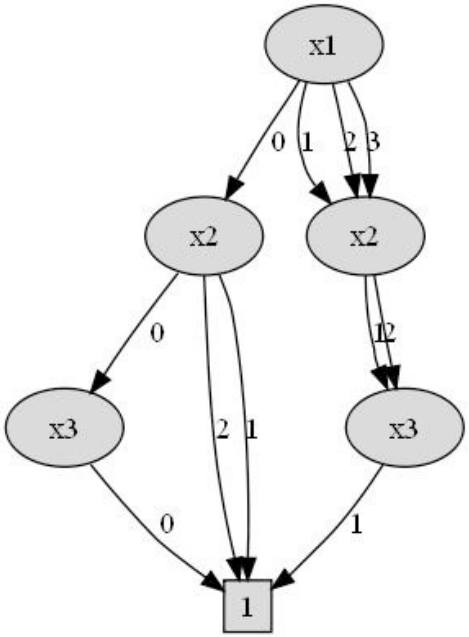


Figure: Reduced MDD

Exploiting a Boolean Encoding

- For a CSP (X, D, F)
- For each $x_i \in X$
  - $X_b^i = \{x_j^i \mid j = 1, \ldots, k_i\}$
  - x_j^i Boolean variables
  - $enc_i(a) = (a_1, \ldots, a_{k_i}) \in \{0,1\}^{k_i}$, injective
- The *log encoding*:
  - $k_i = \lceil log|D_i| \rceil$
  - $x_i = a \Leftrightarrow x_j^i = a_j$: $a = \sum_{j=1}^{k_i} 2^{j-1} a_j$
- The *direct encoding*:
  - $k_i = |D_i|$
  - $x_i = a \Leftrightarrow x_j^i = a_j$: $x_j^i = 1$ for $j = a$ and $x_j^i = 0$ for $j \neq a$.

T-Shirt log-BDD

Example

- Variables: $X = \{x_1, x_2, x_3\}$ for *color, size, print*
- Domains:
  - $D_1 = \{black, white, red, blue\}$,
  - $D_2 = \{small, medium, large\}$,
  - $D_3 = \{MIB, STW\}$.
- Constraints:
  - $f_1 : (x_3 = MIB) \Rightarrow (x_1 = black)$
  - $f_2 : (x_3 = STW) \Rightarrow (x_2 \neq small)$

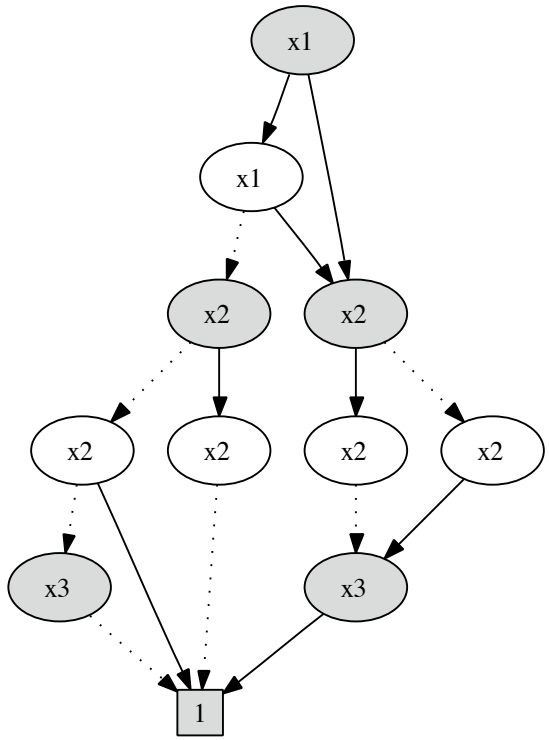


Figure: BDD with log encoding

T-Shirt direct-BDD

Example

- Variables: $X = \{x_1, x_2, x_3\}$ for *color, size, print*
- Domains:
  - $D_1 = \{black, white, red, blue\}$,
  - $D_2 = \{small, medium, large\}$,
  - $D_3 = \{MIB, STW\}$.
- Constraints:
  - $f_1 : (x_3 = MIB) \Rightarrow (x_1 = black)$
  - $f_2 : (x_3 = STW) \Rightarrow (x_2 \neq small)$

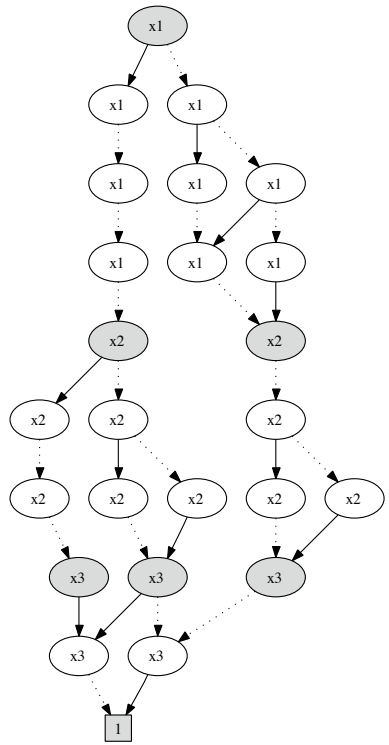


Figure: BDD with direct encoding

Scalability from Compilation in Configuration

Benchmark	Virtual Table			Avg. RT (sec)		Wst. RT (sec)	
	Time(sec)	Size(KB)	#Sol	Configit	ILOG	Configit	ILOG
Renault	460.00	1292	$2.8x10^{12}$	**0.1273**	489.29*	**0.240**	489.29*
Bike	0.45	22	$1.3x10^{8}$	**0.0005**	1.855	**0.010**	882.68
PC	0.89	24	$1.1x10^{6}$	**0.0007**	1.302	**0.010**	2.12
PSR	0.38	37	$7.7x10^{9}$	**0.0014**	2.398	**0.010**	486.12
Parity32_13	30.00	1219	$2.0x10^{8}$	0.0960	**0.061**	0.416	**0.24**
Big-PC	14.82	76	$6.2x10^{19}$	0.0012		0.010	
v1	5.67†	253	$8.2x10^{123}$	0.1620		0.320	
w1	56.52†	1347	$1.0x10^{89}$	0.0680		0.160	
ESVS	0.25	6.7	$3.5x10^{9}$	**0.0004**	0.059	**0.010**	0.14
FS	0.25	5.8	$2.4x10^{7}$	**0.0003**	0.036	**0.010**	0.21
FX	0.22	5.3	$1.2x10^{6}$	**0.0003**	0.029	**0.010**	0.10
Machine	0.14	6.7	$4.7x10^{8}$	**0.0004**	0.009	**0.010**	0.03
C169_FV	2.30 (144)	287	$3.2x10^{15}$	**0.0134**	0.195	**0.010**	28.77
C211_FS	6.93 (957)	370	$1.4x10^{67}$	**0.0219**	0.314	**0.020**	67.09
C250_FW	3.22 (111)	308	$1.2x10^{37}$	**0.0148**	0.203	**0.010**	38.98
C638_FVK	16.53 (1980)	534	$8.8x10^{123}$	**0.0385**	0.608	**0.050**	72.62

*For finding one solution only (i.e., not complete).
†The variable order file has been provided by Configit Software.

Outline

Classic Setting

Two Categories of Constraints

- *background constraints* expressing the connections between the components of the "product", that cannot be removed
- *user constraints* interactively stated by the user when deciding on options (= a query)

Consistency

- A set of constraints is *consistent* if it admits a solution.
- The background constraints are assumed to be consistent.
- The "solubility" of a set of constraints refers to the number of solutions it is consistent with.

Classic Setting

Two Categories of Constraints

- *background constraints* expressing the connections between the components of the "product", that cannot be removed
- *user constraints* interactively stated by the user when deciding on options (= a query)

Consistency

- A set of constraints is *consistent* if it admits a solution.
- The background constraints are assumed to be consistent.
- The "solubility" of a set of constraints refers to the number of solutions it is consistent with.

Terminology

Explanations

- **Conflict**: an inconsistent subset of U: show one cause of inconsistency.
- **Relaxation**: a consistent subset of U: show one possible way of recovering from it

Optimality – sort of

- A relaxation is **maximal** when *no constraint can added* while remaining consistent.
- A conflict is **minimal** when *no constraint can be removed* while remaining inconsistent.

Terminology

Explanations

- **Conflict**: an inconsistent subset of U: show one cause of inconsistency.
- **Relaxation**: a consistent subset of U: show one possible way of recovering from it

Optimality – sort of

- A relaxation is **maximal** when *no constraint can added* while remaining consistent.
- A conflict is **minimal** when *no constraint can be removed* while remaining inconsistent.

Example explanation tasks

Configuration as a CSP

- A “product” is fully specified by some constraints
- Several options are available to the user
- The user expresses his preferences as constraints

Explanations

When preferences conflict:

Conflict show a set of conflicting preferences

Relaxation show a set of feasible preferences

Example explanation tasks

Configuration as a CSP

- A "product" is fully specified by some constraints
- Several options are available to the user
- The user expresses his preferences as constraints

Explanations

When preferences conflict:

Conflict	show a set of conflicting preferences
Relaxation	show a set of feasible preferences

Example explanation tasks

Debugging a Constraint Programme

- A model represents a reality using some constraints
- The programmer "proposes" a model

Explanations

When the model/reality conflict:

Conflict show a set of conflicts between the model and reality

Relaxation show a set of feasible constraints

Example explanation tasks

Debugging a Constraint Programme

- A model represents a reality using some constraints
- The programmer "proposes" a model

Explanations

When the model/reality conflict:

Conflict show a set of conflicts between the model and reality

Relaxation show a set of feasible constraints

Conflicts, Arguments, and Counterarguments (I)

Assumption

The *propagation capability* of a constraints solver can be described by operator Π mapping a set of given constraints to a set of deduced constraints. (e.g. arc consistency deduces constraints of form $x \neq v$)

Conflicts, Arguments, and Counter-arguments (II)

Conflict

For given set of constraints $\mathcal{X}$ + background $\mathcal{B}$:

- Π**-conflict:** subset X of $\mathcal{X}$ such that $\Pi(\mathcal{B} \cup X)$ contains an inconsistency.
- **minimal** Π**-conflict:** no proper subset is a conflict
- **preferred** Π**-conflict:** culprits are chosen according to a total order
- **global conflict:** Π is complete (i.e. achieves global consistency)

Arguments and Counter-Arguments

(counter-)argument for ϕ: add $\neg\phi$ (ϕ) to $\mathcal{B}$ + find conflict

Conflicts, Arguments, and Counter-arguments (II)

Conflict

For given set of constraints $\mathcal{X}$ + background $\mathcal{B}$:

- Π**-conflict:** subset X of $\mathcal{X}$ such that $\Pi(\mathcal{B} \cup X)$ contains an inconsistency.
- **minimal** Π**-conflict:** no proper subset is a conflict
- **preferred** Π**-conflict:** culprits are chosen according to a total order
- **global conflict:** Π is complete (i.e. achieves global consistency)

Arguments and Counter-Arguments

(counter-)argument for ϕ: add $\neg\phi$ (ϕ) to $\mathcal{B}$ + find conflict

Which Explanations?

Example

A customer wants station-wagon with options:

1. requirement r_1: roof racks ($\$500$)
2. requirement r_2: CD-player ($\$500$)
3. requirement r_3: extra seat ($\$800$)
4. requirement r_4: metal color ($\$500$)
5. requirement r_5: luxury version ($\$2600$)

Total budget for options is $\$3000$

User requirements cannot be satisfied

Which requirements are in conflict?

Which Explanations?

Example

A customer wants station-wagon with options:

1. requirement r_1: roof racks ($\$500$)
2. requirement r_2: CD-player ($\$500$)
3. requirement r_3: extra seat ($\$800$)
4. requirement r_4: metal color ($\$500$)
5. requirement r_5: luxury version ($\$2600$)

Total budget for options is $\$3000$

User requirements cannot be satisfied

Which requirements are in conflict?

An Arbitary Explanation

Maintain explanations during propagation

r_1	roof racks	$c \geq 500$	$\{r_1\}$
r_2	CD-player	$c \geq 1000$	$\{r_1, r_2\}$
r_3	extra seat	$c \geq 1800$	$\{r_1, r_2, r_3\}$
r_4	metal color	$c \geq 2300$	$\{r_1, r_2, r_3, r_4\}$
r_5	luxury version	$c \geq 4900$	$\{r_1, r_2, r_3, r_4, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_1, r_2, r_3, r_4, r_5, b\}$

explanation: $\{r_1, r_2, r_3, r_4, r_5, b\}$

This explanation is not minimal (irreducible)!

The user may retract constraints unnecessarily.

An Arbitary Explanation

Maintain explanations during propagation

r_1	roof racks	$c \geq 500$	$\{r_1\}$
r_2	CD-player	$c \geq 1000$	$\{r_1, r_2\}$
r_3	extra seat	$c \geq 1800$	$\{r_1, r_2, r_3\}$
r_4	metal color	$c \geq 2300$	$\{r_1, r_2, r_3, r_4\}$
r_5	luxury version	$c \geq 4900$	$\{r_1, r_2, r_3, r_4, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_1, r_2, r_3, r_4, r_5, b\}$

explanation: $\{r_1, r_2, r_3, r_4, r_5, b\}$

This explanation is not minimal (irreducible)!

The user may retract constraints unnecessarily.

An Arbitary Explanation

Maintain explanations during propagation

r_1	roof racks	$c \geq 500$	$\{r_1\}$
r_2	CD-player	$c \geq 1000$	$\{r_1, r_2\}$
r_3	extra seat	$c \geq 1800$	$\{r_1, r_2, r_3\}$
r_4	metal color	$c \geq 2300$	$\{r_1, r_2, r_3, r_4\}$
r_5	luxury version	$c \geq 4900$	$\{r_1, r_2, r_3, r_4, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_1, r_2, r_3, r_4, r_5, b\}$

explanation: $\{r_1, r_2, r_3, r_4, r_5, b\}$

This explanation is not minimal (irreducible)!

The user may retract constraints unnecessarily.

Minimal Explanation

Some other propagation order

r_4	metal color	$c \geq 500$	$\{r_4\}$
r_5	luxury version	$c \geq 3100$	$\{r_4, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_4, r_5, b\}$

explanation: $\{r_4, r_5, b\}$

Minimal - Good!

The **explanation is minimal**, since any proper subset is consistent.

Minimal Explanation

Some other propagation order

r_4	metal color	$c \geq 500$	$\{r_4\}$
r_5	luxury version	$c \geq 3100$	$\{r_4, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_4, r_5, b\}$

explanation: $\{r_4, r_5, b\}$

Minimal - Good!

The **explanation is minimal**, since any proper subset is consistent.

Minimal Explanation

Some other propagation order

r_4	metal color	$c \geq 500$	$\{r_4\}$
r_5	luxury version	$c \geq 3100$	$\{r_4, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_4, r_5, b\}$

explanation: $\{r_4, r_5, b\}$

Minimal - Good!

The **explanation is minimal**, since any proper subset is consistent.

Finding a Minimal Conflict

Example

Step	Activated constraints	Result	Partial conflict
1.	ρ_1	no fail	$\{\}$
2.	ρ_1 ρ_2	no fail	$\{\}$
3.	ρ_1 ρ_2 ρ_3	no fail	$\{\}$
4.	ρ_1 ρ_2 ρ_3 ρ_4	no fail	$\{\}$
5.	ρ_1 ρ_2 ρ_3 ρ_4 ρ_5	fail	$\{\rho_5\}$
6.	ρ_5	no fail	$\{\rho_5\}$
7.	ρ_5 ρ_1	fail	$\{\rho_1, \rho_5\}$

rePlayXplain: Detect culprit and replay

Modified example

Requested options 1,2,3,4,7 cost $100\$$ each; requested options 5,6,8 cost $800\$$ each; budget is 2200.

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.	21.	22.
R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8	R_8
	R_2	R_2	R_2	R_2	R_2	R_2	R_2		R_1	R_1	R_1	R_1	R_1	R_1	R_6	R_6	R_6	R_6	R_6	R_6	R_6
		R_3	R_3	R_3	R_3	R_3	R_3			R_2	R_2	R_2	R_2	R_2		R_1	R_1	R_1	R_1	R_1	R_5
			R_4	R_4	R_4	R_4	R_4				R_3	R_3	R_3	R_3			R_2	R_2	R_2	R_2	
				R_5	R_5	R_5	R_5					R_4	R_4	R_4				R_3	R_3	R_3	
					R_6	R_6	R_6						R_5	R_5					R_4	R_4	
						R_7	R_7							R_6						R_5	
							R_8														
							F							**F**						**F**	**F**
							R_8							R_6						R_5	

*Add available constraints to CP Solver one after the other; when failure (**F**) occurs new culprit is detected; backtrack to initial state + add culprit there*

QuickXplain: Detect culprit and divide

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_8	R_8	R_8
	R_2	R_2	R_2	R_2	R_2	R_2	R_2	R_2	R_2	R_2	R_2	R_2		R_6	R_6
		R_3	R_3	R_3	R_3	R_3	R_3	R_3	R_3	R_3	R_3	R_3			R_5
			$R_.$	$R_.$	R_4	R_4	R_4	R_4	R_4	R_4	R_4	R_4			
				R_5	R_5	R_5	R_5	R_8	R_8	R_8	R_8	R_8			
					R_6	R_6	R_6		R_5	R_5	R_6	R_6			
						R_7	R_7			R_6		R_5			
							R_8								
							F			**F**		**F**			**F**
							R_8			R_6		R_5			

Go To Next Page

Divide conflict detection problem into 2 subproblems when culprit is detected:

1. keep all constraint of first subproblem when solving second subproblem;
2. add culprits of second subproblem when solving first subproblem.

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5 + b$
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5 + b$
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5$ + b
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5 + b$
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5 + b$
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5 + b$
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Unnecessary Retractions

Use explanation for finding a solution

1. user submits requirements $r_1, \ldots, r_5 + b$
2. response: failure due to $\{r_4, r_5, b\}$
3. user prefers luxury (r_5) to metal color (r_4), so removes r_4
4. response: failure due to $\{r_3, r_5, b\}$
5. user prefers extra seats (r_3) to luxury (r_5), so removes r_5
6. response: **success**

The retraction of r_4 is no longer justified.

Can we avoid unnecessary retractions?

Preferred Explanation

Again another propagation order

r_3	metal color	$c \geq 800$	$\{r_3\}$
r_5	luxury version	$c \geq 3300$	$\{r_3, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_3, r_5, b\}$

explanation: $\{r_3, r_5, b\}$

Explanation is preferred

Its worst element r_5 can safely be retracted

Preferred Explanation

Again another propagation order

r_3	metal color	$c \geq 800$	$\{r_3\}$
r_5	luxury version	$c \geq 3300$	$\{r_3, r_5\}$
b	total budget	$c \leq 3000$	$\{b\}$
		failure	$\{r_3, r_5, b\}$

explanation: $\{r_3, r_5, b\}$

Explanation is preferred

Its worst element r_5 can safely be retracted

Preferences between Constraints [7]

Intuitive statements with simple semantics

- preferences between constraints

```
prefer(luxury version, metal color)
prefer(extra seat, luxury version)
```

- groups of constraints
  - `equipment` contains requirements for roof racks, extra seat
  - `look` contains requirements for metal color, seat material
- preferences between groups

```
prefer(equipment, look)
```

The Tasks

Overconstrained problem with preferences

- background B
- constraints $C := \{c_1, \ldots, c_n\}$
- preferences P between the c_i's

such that $B \cup C$ is inconsistent

The tasks

- preferred relaxations
- preferred explanations

The Tasks

Overconstrained problem with preferences

- background B
- constraints $C := \{c_1, \ldots, c_n\}$
- preferences P between the c_i's

such that $B \cup C$ is inconsistent

The tasks

- preferred relaxations
- preferred explanations

Intuition behind the Approach

Preferred Conflicts

We use a preference-guided algorithm that successively adds most preferred constraints until they fail. It then backtracks and removes the least preferred constraints if this preserves the failure.

Preferred Relaxations

We remove the least preferred constraints from an inconsistent set until it is consistent.

Duality

Preferred conflicts explain why best elements cannot be added to preferred relaxations.

Intuition behind the Approach

Preferred Conflicts

We use a preference-guided algorithm that successively adds most preferred constraints until they fail. It then backtracks and removes the least preferred constraints if this preserves the failure.

Preferred Relaxations

We remove the least preferred constraints from an inconsistent set until it is consistent.

Duality

Preferred conflicts explain why best elements cannot be added to preferred relaxations.

Intuition behind the Approach

Preferred Conflicts

We use a preference-guided algorithm that successively adds most preferred constraints until they fail. It then backtracks and removes the least preferred constraints if this preserves the failure.

Preferred Relaxations

We remove the least preferred constraints from an inconsistent set until it is consistent.

Duality

Preferred conflicts explain why best elements cannot be added to preferred relaxations.

Preferred Relaxations [5]

Simple lexicographic semantics

- relaxation: subset of C that is consistent w.r.t. B
- relaxation of ranking π: $LexRelax(c_{\pi_1}, \ldots, c_{\pi_n})(B)$ is best relaxation w.r.t lexicographical order $<_{lex}$ that maximizes selection of more important constraints c_{π_i} (those with smaller indices i)
- preferred relaxation: relaxation of a ranking π that respects the preferences

Preferred Conflicts [5]

Do it in a similar way...

- conflict: subset of C that is inconsistent w.r.t. B
- conflict of ranking π:
  $LexXplain(c_{\pi_1}, \ldots, c_{\pi_n})(B)$ is best conflict w.r.t lexicographical order $<_{antilex}$ that minimizes selection of less important constraints c_{π_i} (those with larger indices i)
- preferred conflict: conflict of a ranking π that respects the preferences

Algorithm QUICKXPLAIN [5]

Recursive decomposition à la QUICKSORT

1. If B is inconsistent then: $LexXplain(c_{\pi_1}, \ldots, c_{\pi_n})(B) = \emptyset$
2. If B is consistent and C is a singleton then: $LexXplain(c_{\pi_1}, \ldots, c_{\pi_n})(B) = C$
3. If B is consistent and C has more than one element then split at k
   1. let $C_k := \{c_{\pi_1}, \ldots, c_{\pi_k}\}$
   2. let E_2 be $LexXplain(c_{\pi_{k+1}}, \ldots, c_{\pi_n})(B \cup C_k)$
   3. let E_1 be $LexXplain(c_{\pi_1}, \ldots, c_{\pi_k})(B \cup E_2)$
   4. $LexXplain(c_{\pi_1}, \ldots, c_{\pi_n})(B) = E_1 \cup E_2$

Where to Split?

Effect

If a subproblem does not contain an element of the conflict then it can be solved by a single consistency check, namely $B \cup C_k$ or $B \cup E_2$

Strategy

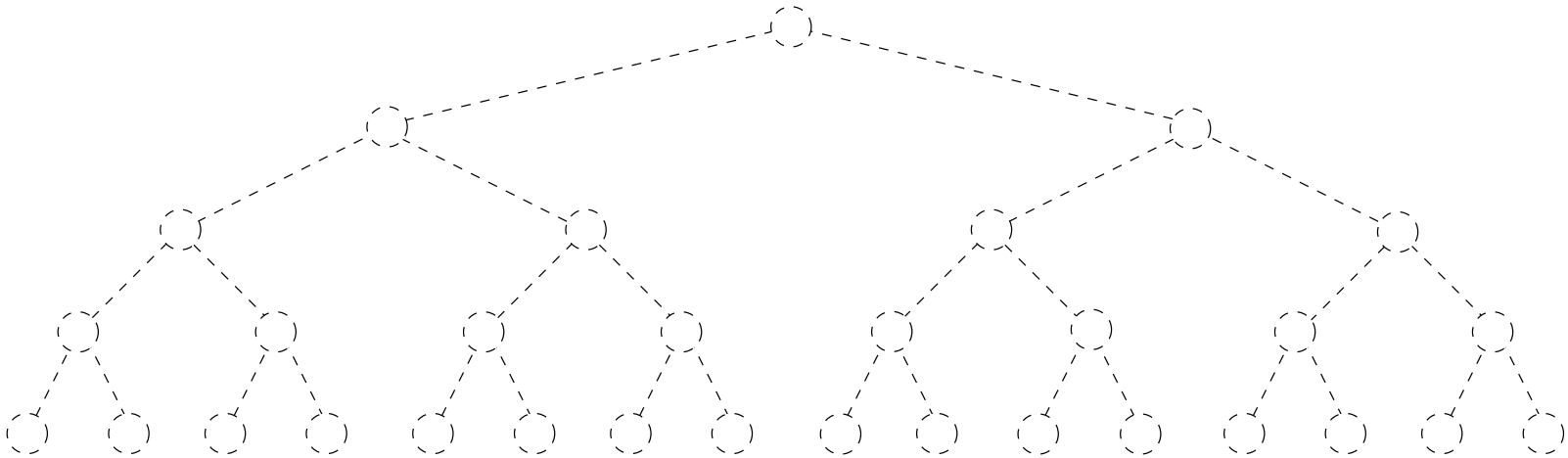
Choose subproblems of same size to exploit this effect in a best way

#Consistency Checks

Between $\log_2 \frac{n}{k} + 2k$ and $2k \cdot \log_2 \frac{n}{k} + 2k$ (for conflicts of size k)

Call Graph for QUICKXPLAIN

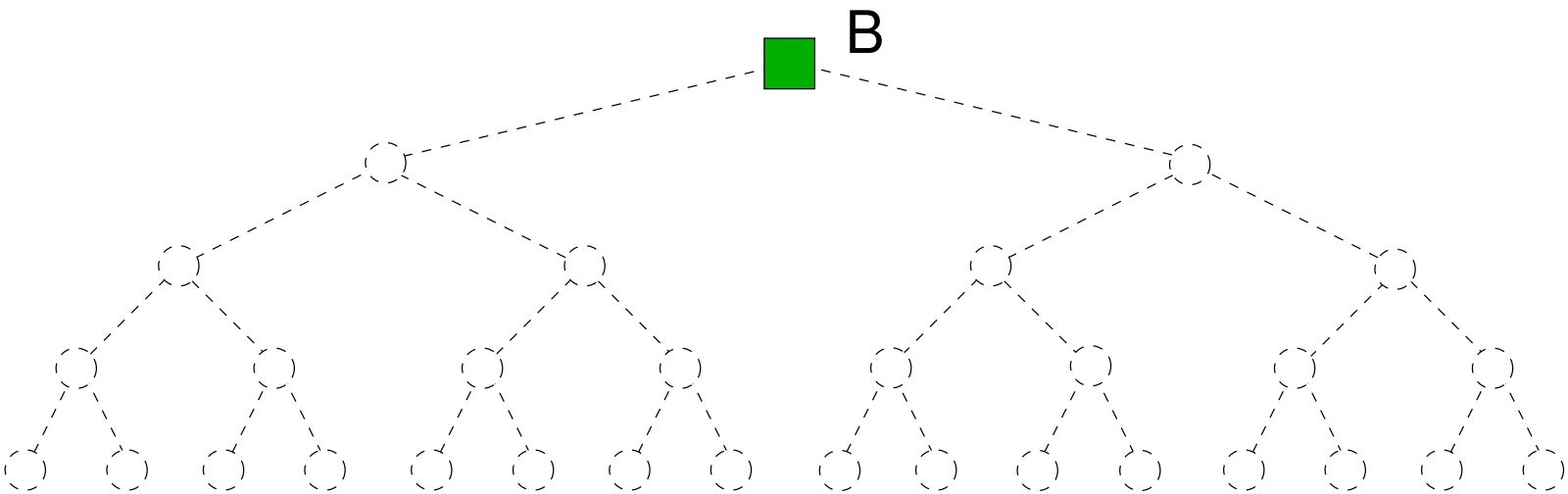
constraints $c_1, \ldots, c_{16}$ + background B.



we compute $LexXplain(c_1, \ldots, c_{16})(B)$

Call Graph for QUICKXPLAIN

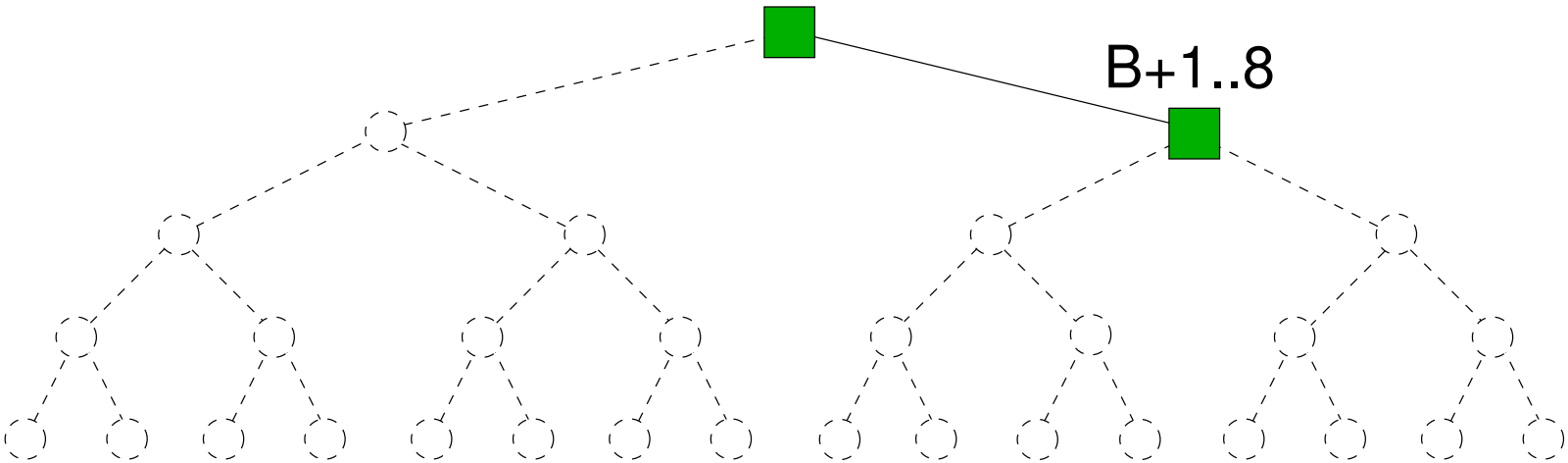
checking B



success $\Rightarrow$ **some of of 1..16 needed to fail**

Call Graph for QUICKXPLAIN

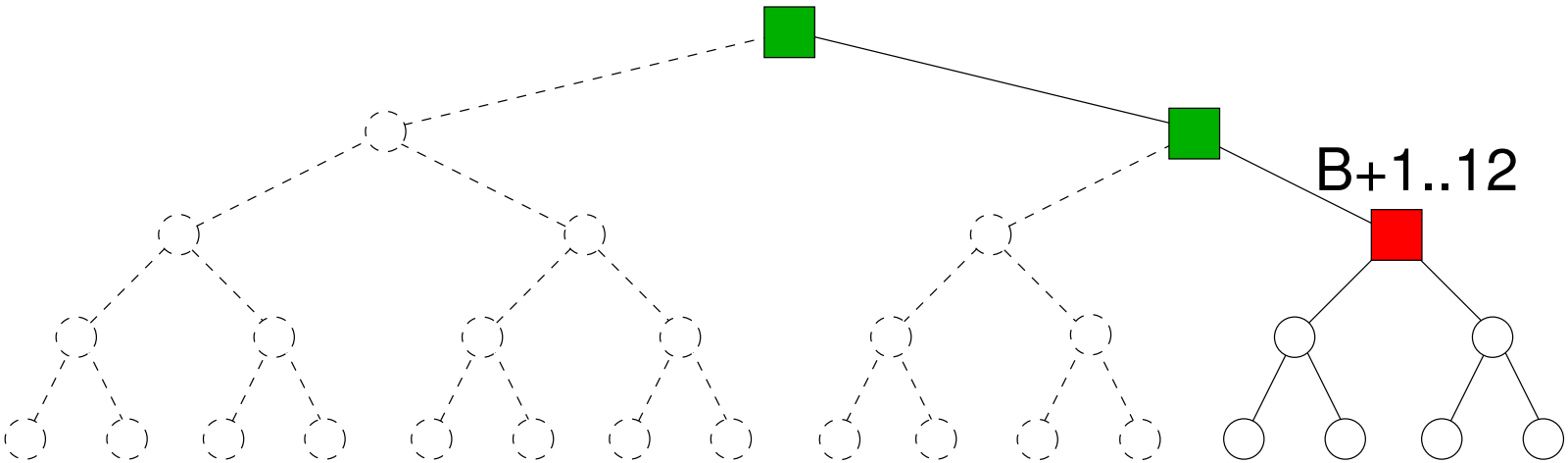
checking $B + 1..8$



success $\Rightarrow$ **some of of 9..16 needed to fail**

Call Graph for QUICKXPLAIN

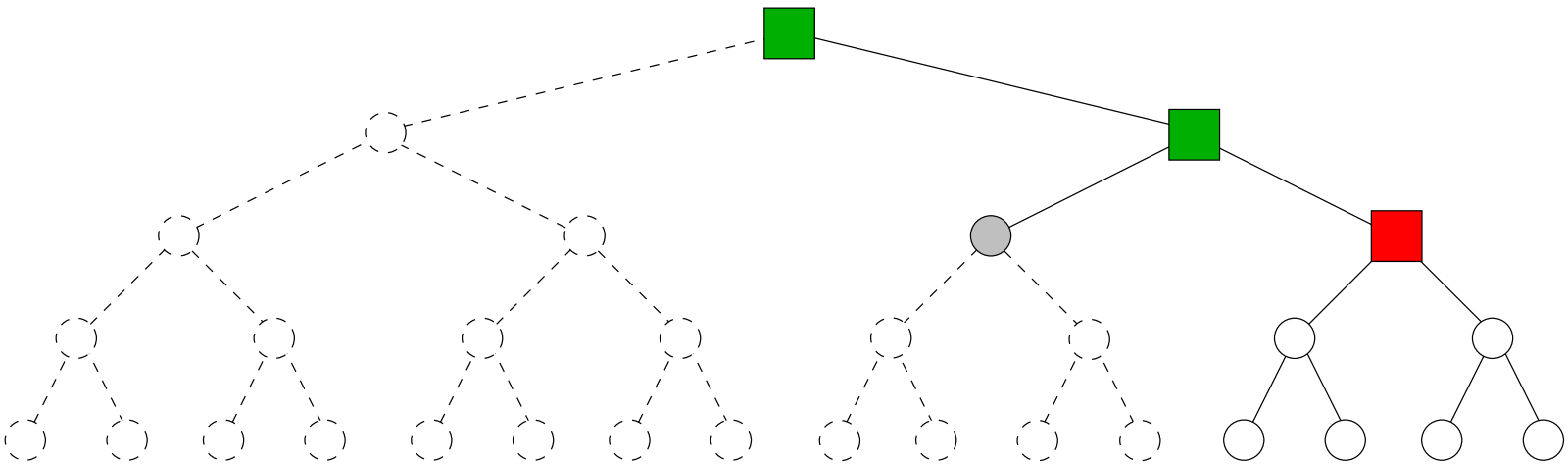
checking $B + 1, \ldots, 12$



failure $\Rightarrow$ **none of 13..16 needed to fail**

Call Graph for QUICKXPLAIN

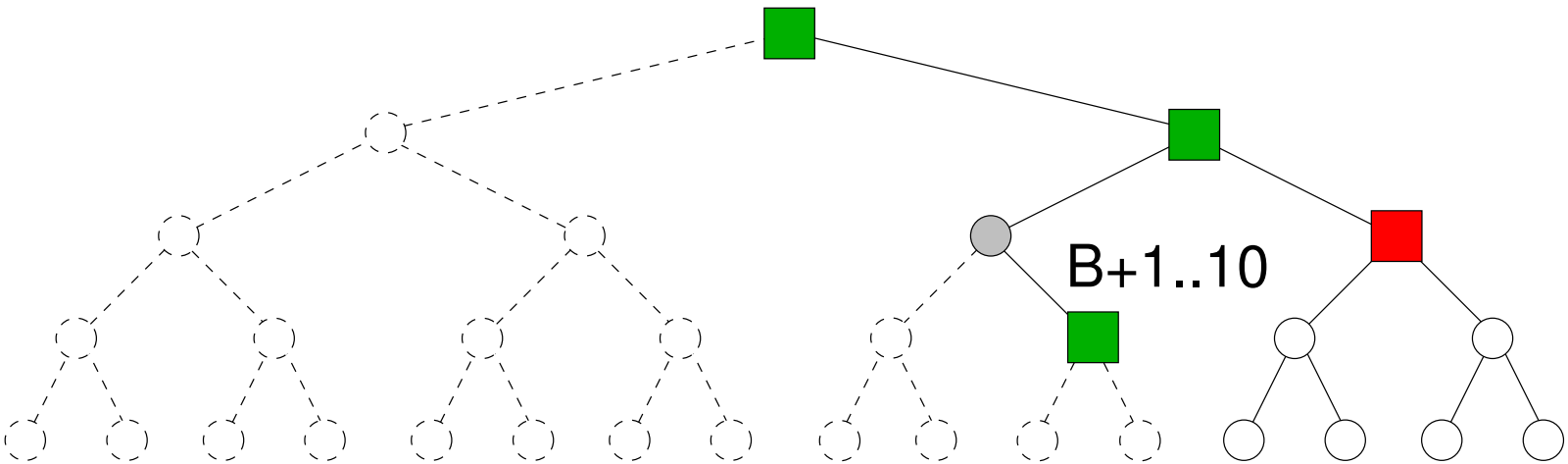
checking $B + 1..8$?



none of 13..16 added to background $\Rightarrow$ **already checked**

Call Graph for QUICKXPLAIN

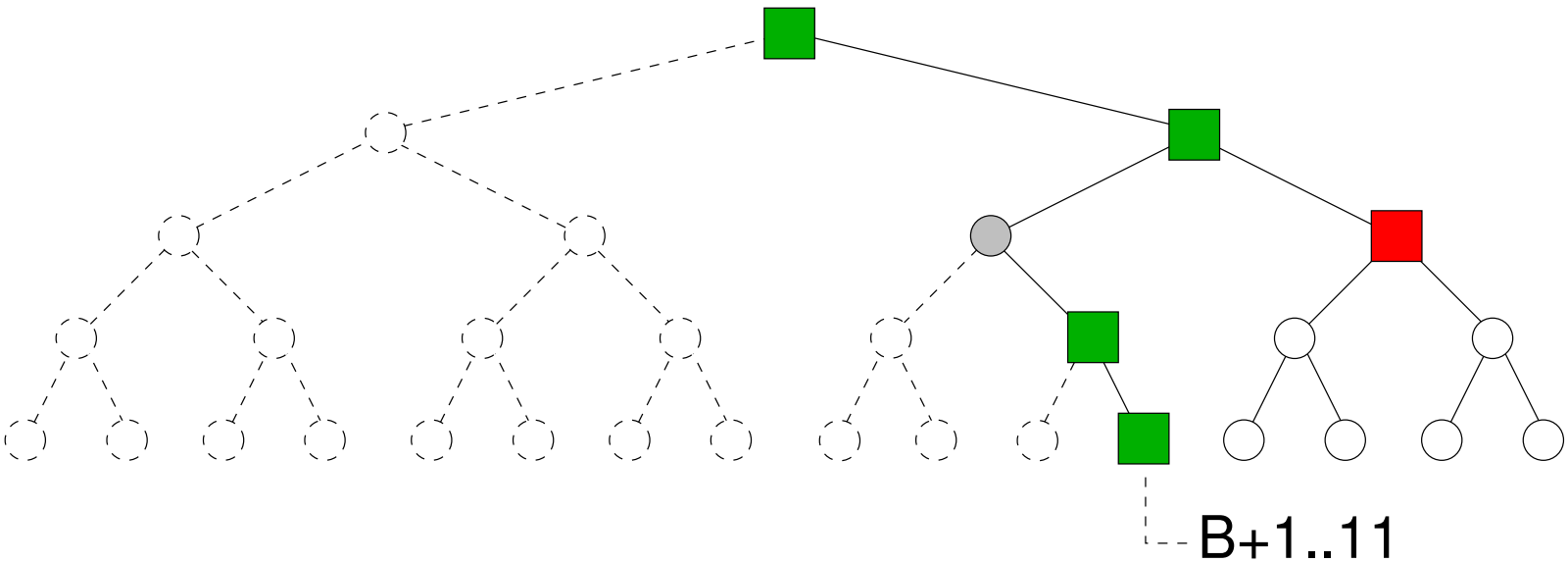
checking $B + 1..10$



success $\Rightarrow$ **some of 11..12 needed to fail**

Call Graph for QUICKXPLAIN

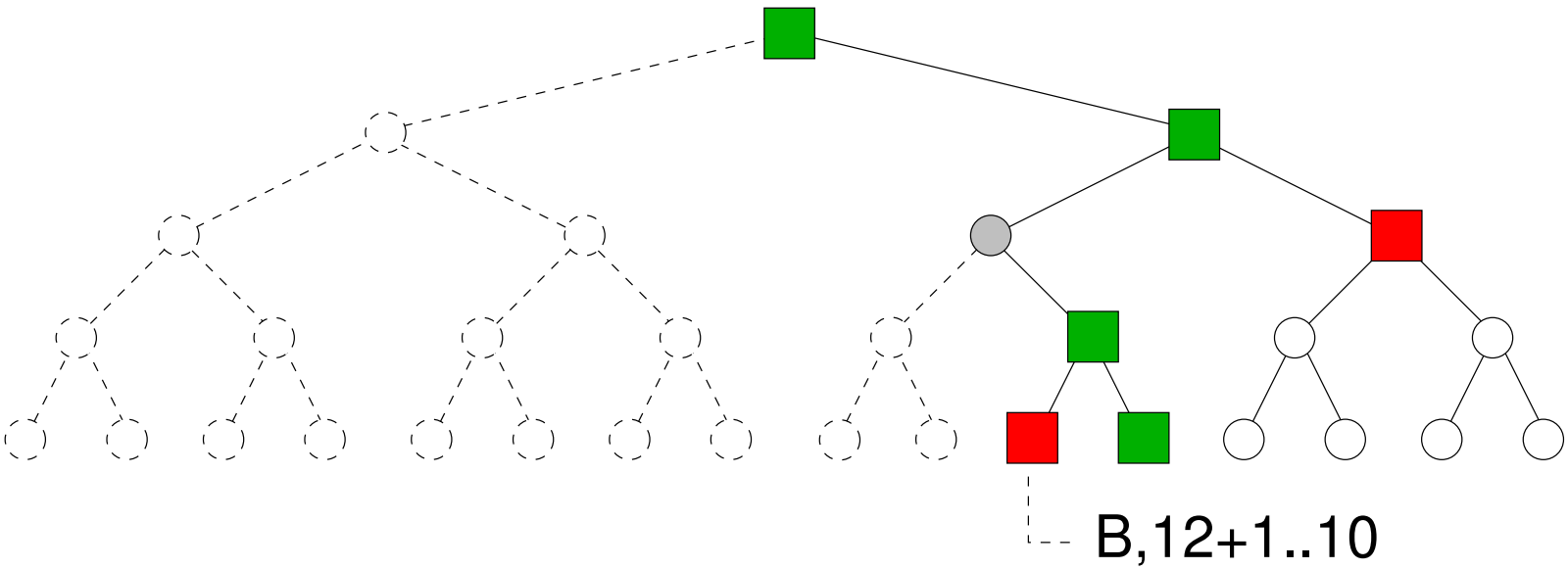
checking $B + 1..11$



success $\Rightarrow 12$ **is needed to fail**

Call Graph for QUICKXPLAIN

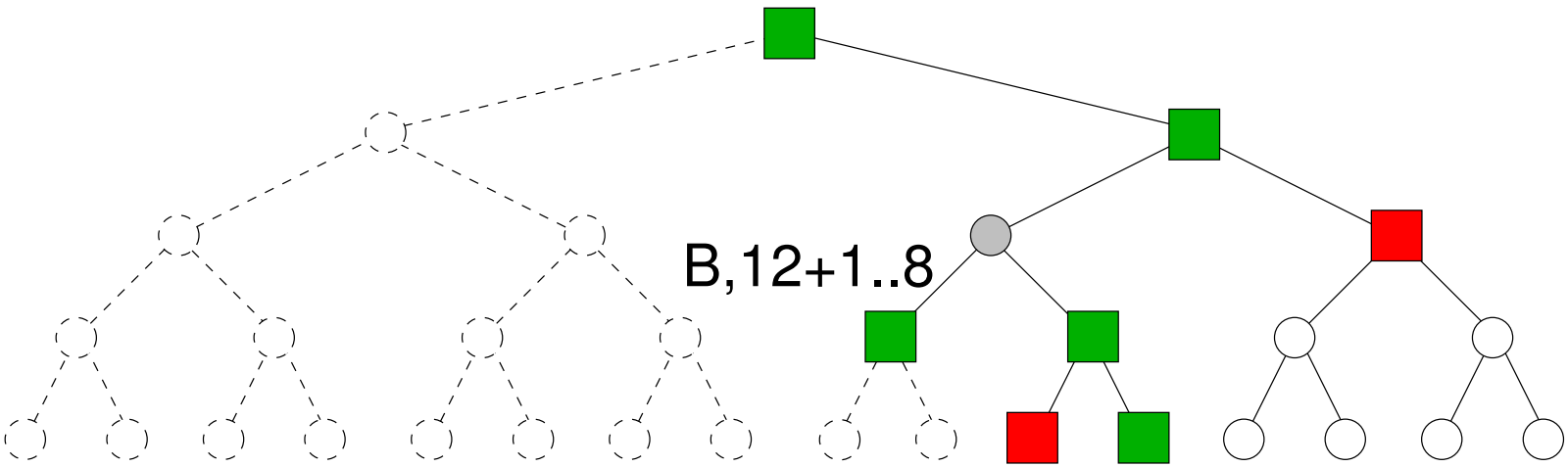
checking $B, 12 + 1..10$



failure $\Rightarrow 11$ **is not needed to fail**

Call Graph for QUICKXPLAIN

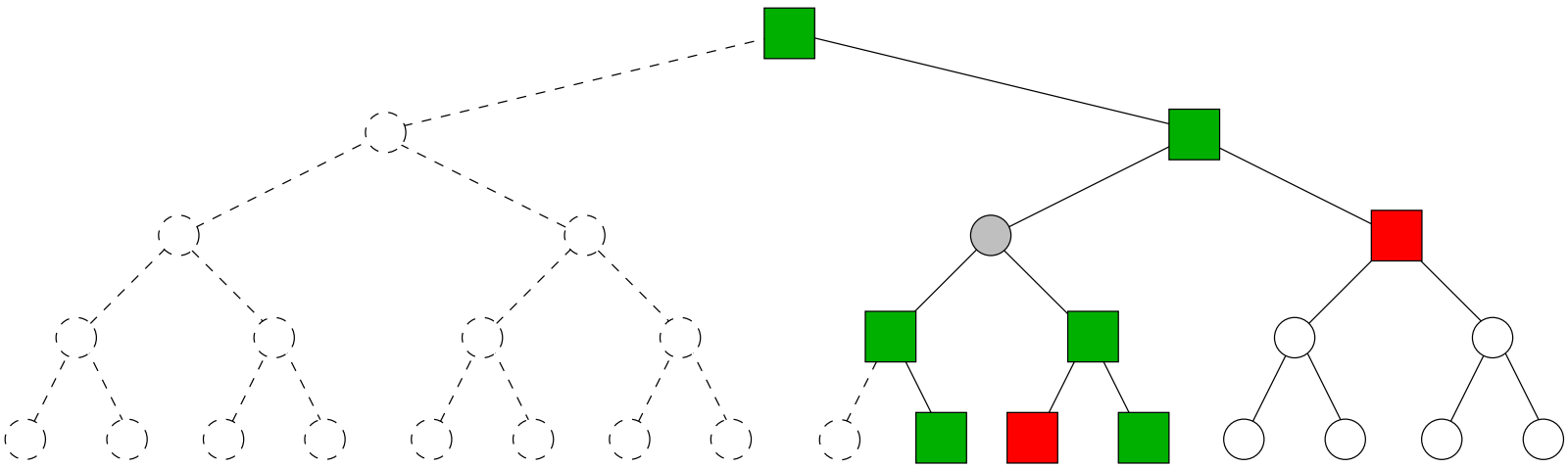
checking $B, 12 + 1..8$



success $\Rightarrow$ **some of 9..10 needed to fail**

Call Graph for QUICKXPLAIN

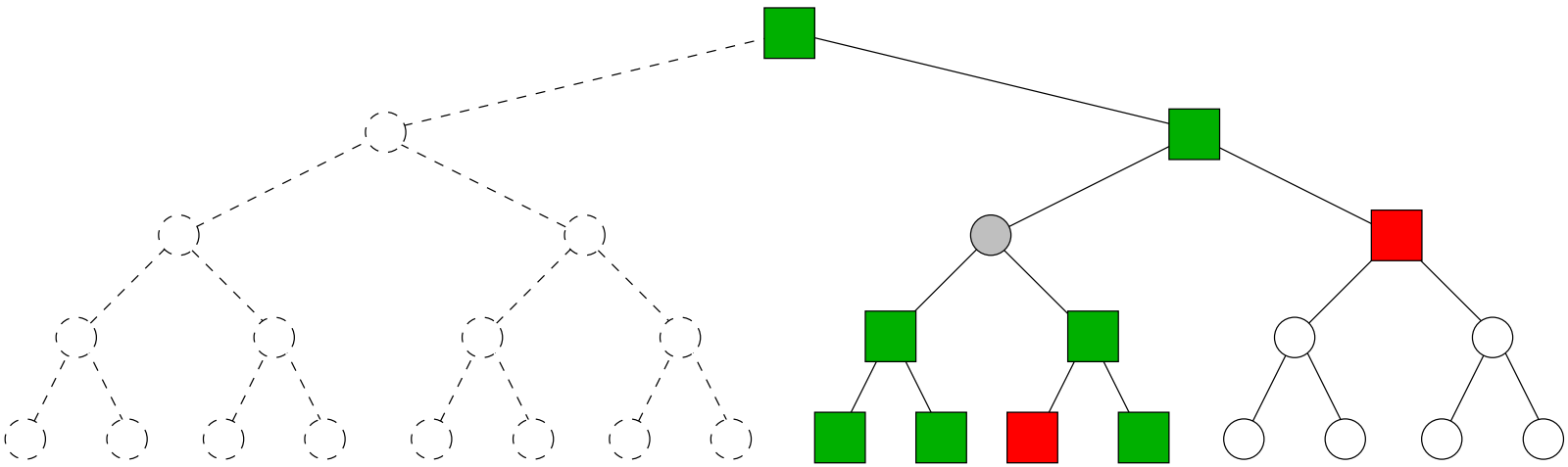
checking $B, 12 + 1..9$



success $\Rightarrow 10$ **is needed to fail**

Call Graph for QUICKXPLAIN

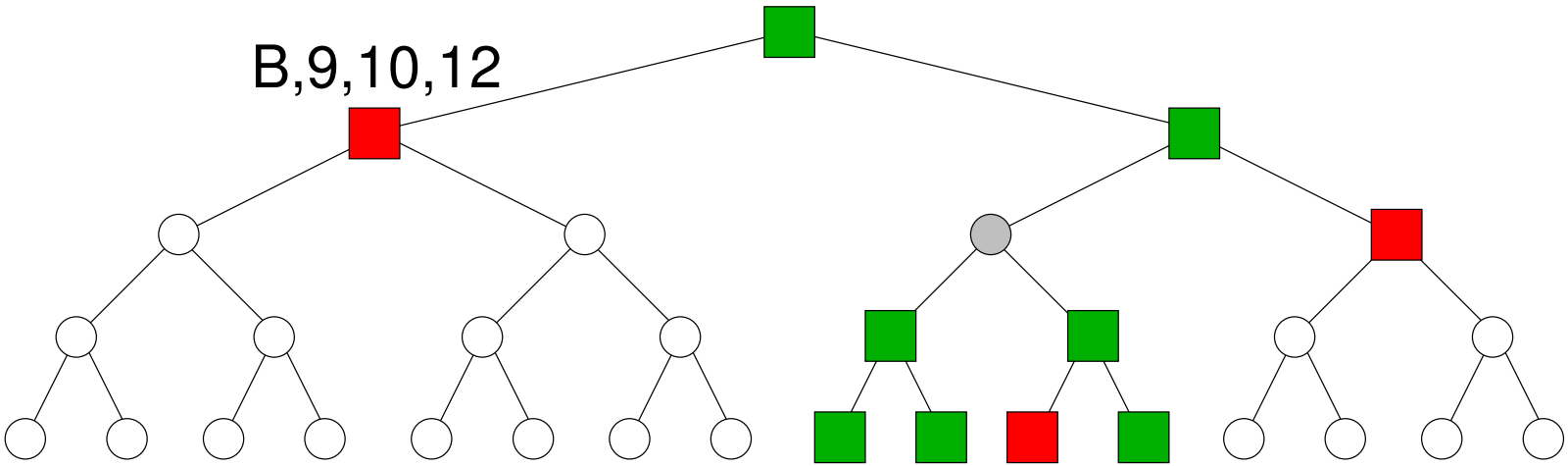
checking $B, 11, 12 + 1..8$



success $\Rightarrow 9$ **is needed to fail**

Call Graph for QUICKXPLAIN

checking $B, 9, 10, 12$



failure $\Rightarrow$ **none of** $1..8$ **needed to fail**

preferred explanation: $9, 10, 12$

Consistency Checking

The cost of consistency checking

QUICKXPLAIN does multiple consistency checks that are NP-hard in general, but

- complexity is polynomial for tree-like CSPs
- approximations possible: trade time and optimality
- keep witnesses for success (= solution) and try them when adding constraints
- keep witnesses for failure (= critical search decisions) and try them when removing constraints

Compilation helps in practice

Most problems in practice give small compiled forms.

Consistency Checking

The cost of consistency checking

QUICKXPLAIN does multiple consistency checks that are NP-hard in general, but

- complexity is polynomial for tree-like CSPs
- approximations possible: trade time and optimality
- keep witnesses for success (= solution) and try them when adding constraints
- keep witnesses for failure (= critical search decisions) and try them when removing constraints

Compilation helps in practice

Most problems in practice give small compiled forms.

How to use QuickXplain

- **Background:** effort is reduced by putting as many constraints as possible in the initial background
- **Preference order**: order of constraint uniquely characterizes the conflict found
- **Consistency checker**: time can be traded against minimality by an incomplete consistency checker, giving "anytime" behaviour

How to use QuickXplain

- **Background:** effort is reduced by putting as many constraints as possible in the initial background
- **Preference order**: order of constraint uniquely characterizes the conflict found
- **Consistency checker**: time can be traded against minimality by an incomplete consistency checker, giving "anytime" behaviour

How to use QuickXplain

- **Background:** effort is reduced by putting as many constraints as possible in the initial background
- **Preference order**: order of constraint uniquely characterizes the conflict found
- **Consistency checker**: time can be traded against minimality by an incomplete consistency checker, giving "anytime" behaviour

Applications of QuickXplain

- Configuration: B2B, B2C find conflicts between user requests.
- Constraint model debugging isolate failing parts of the constraint model.
- Rule verification find tests that make a rule never applicable.
- Benders decomposition.
- Diagnosis of ontologies.

Applications of QuickXplain

- Configuration: B2B, B2C find conflicts between user requests.
- Constraint model debugging isolate failing parts of the constraint model.
- Rule verification find tests that make a rule never applicable.
- Benders decomposition.
- Diagnosis of ontologies.

Applications of QuickXplain

- Configuration: B2B, B2C find conflicts between user requests.
- Constraint model debugging isolate failing parts of the constraint model.
- Rule verification find tests that make a rule never applicable.
- Benders decomposition.
- Diagnosis of ontologies.

Applications of QuickXplain

- Configuration: B2B, B2C find conflicts between user requests.
- Constraint model debugging isolate failing parts of the constraint model.
- Rule verification find tests that make a rule never applicable.
- Benders decomposition.
- Diagnosis of ontologies.

Applications of QuickXplain

- Configuration: B2B, B2C find conflicts between user requests.
- Constraint model debugging isolate failing parts of the constraint model.
- Rule verification find tests that make a rule never applicable.
- Benders decomposition.
- Diagnosis of ontologies.

Enumerating all Maximal Relaxations

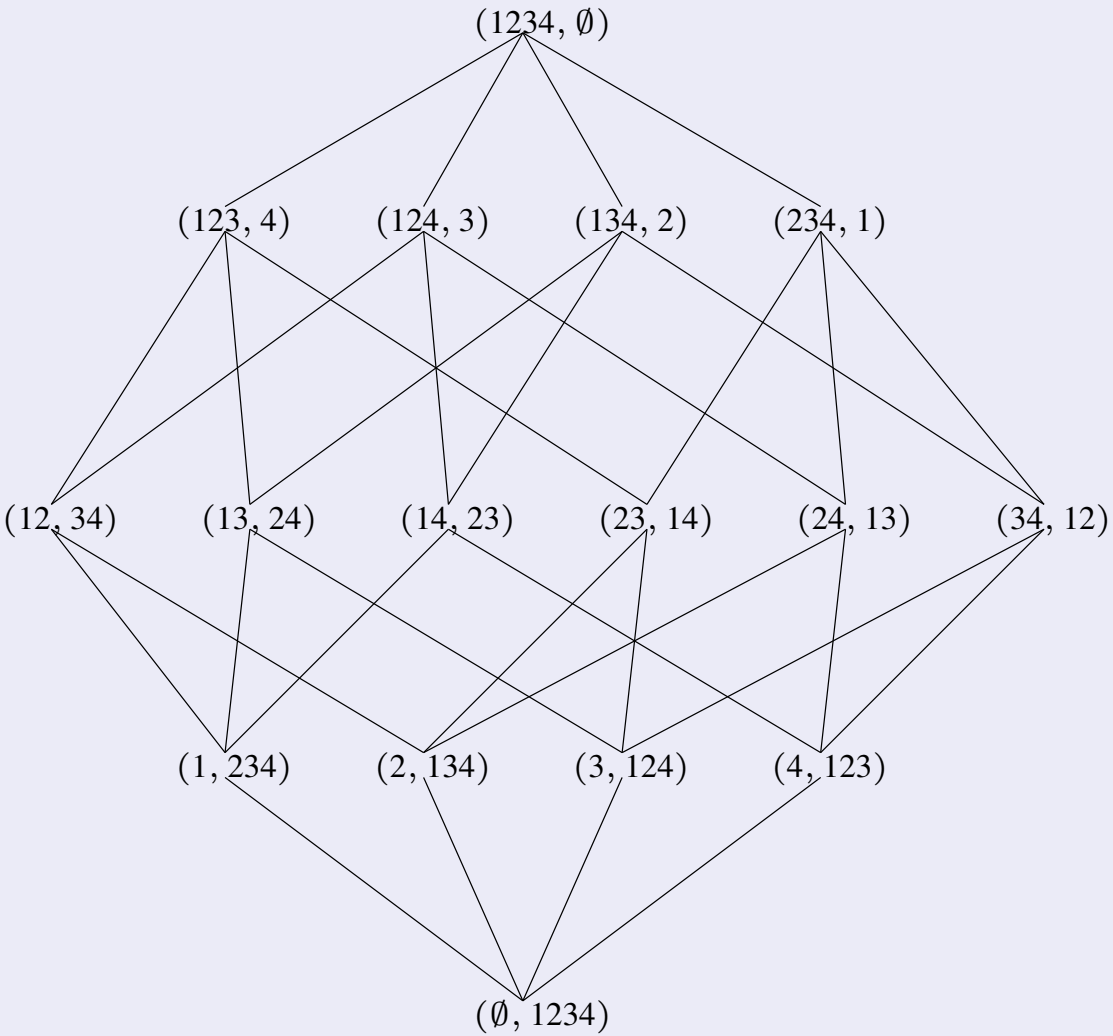
Problem

As an example, we consider a problem with the following explanations and conflicts. Here we will regard an explanation as a pair comprising a (maximal) relaxation and its complement, which we refer to as an exclusion set

$$\begin{array}{rl} \text{Explanations:} & (12, 34)\ (13, 24)\ (4, 123) \\ \text{Conflicts:} & 14, 23, 24, 34 \end{array}$$

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



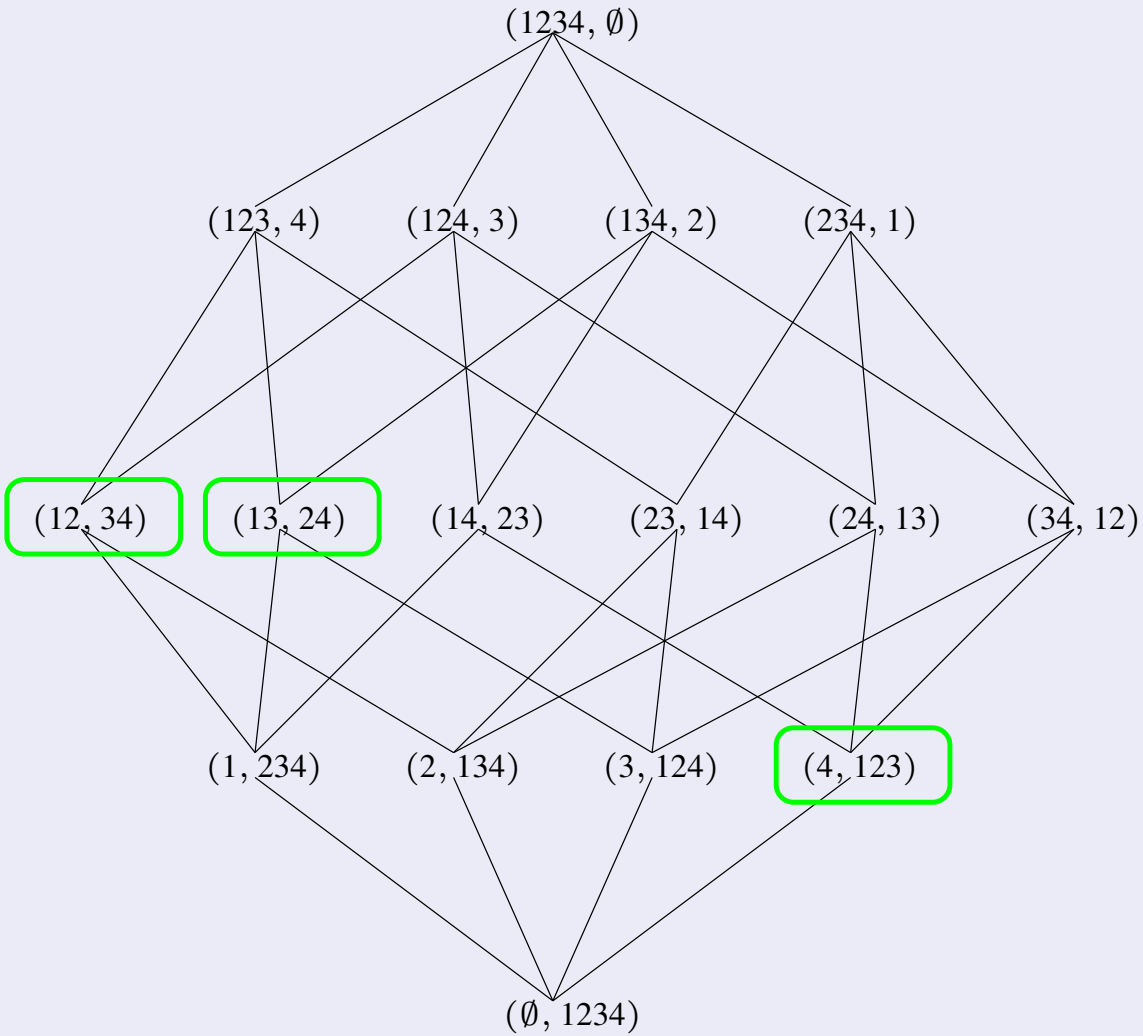
Conflicts

14, 23, 24, 34

Steps

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



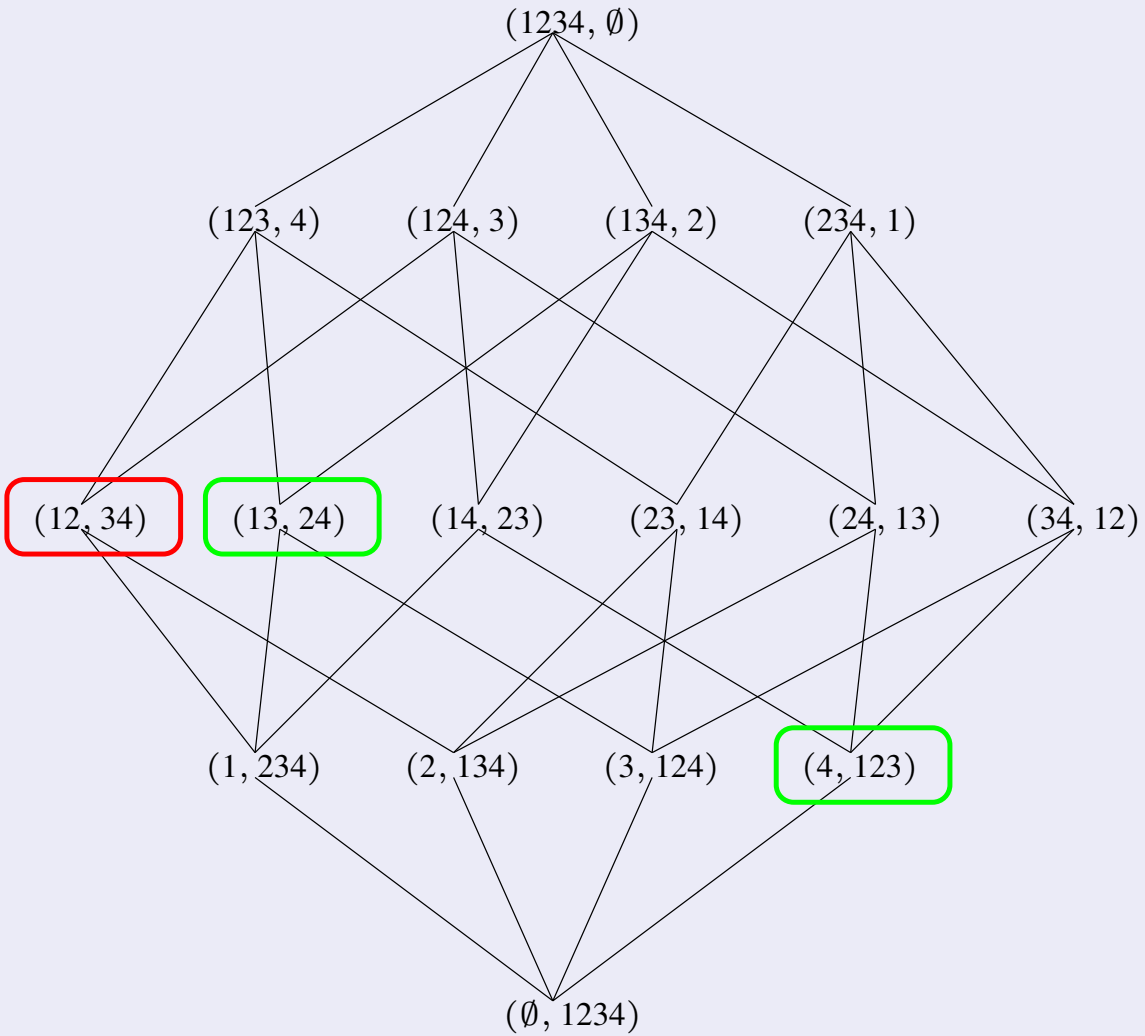
Conflicts

14, 23, 24, 34

Steps

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

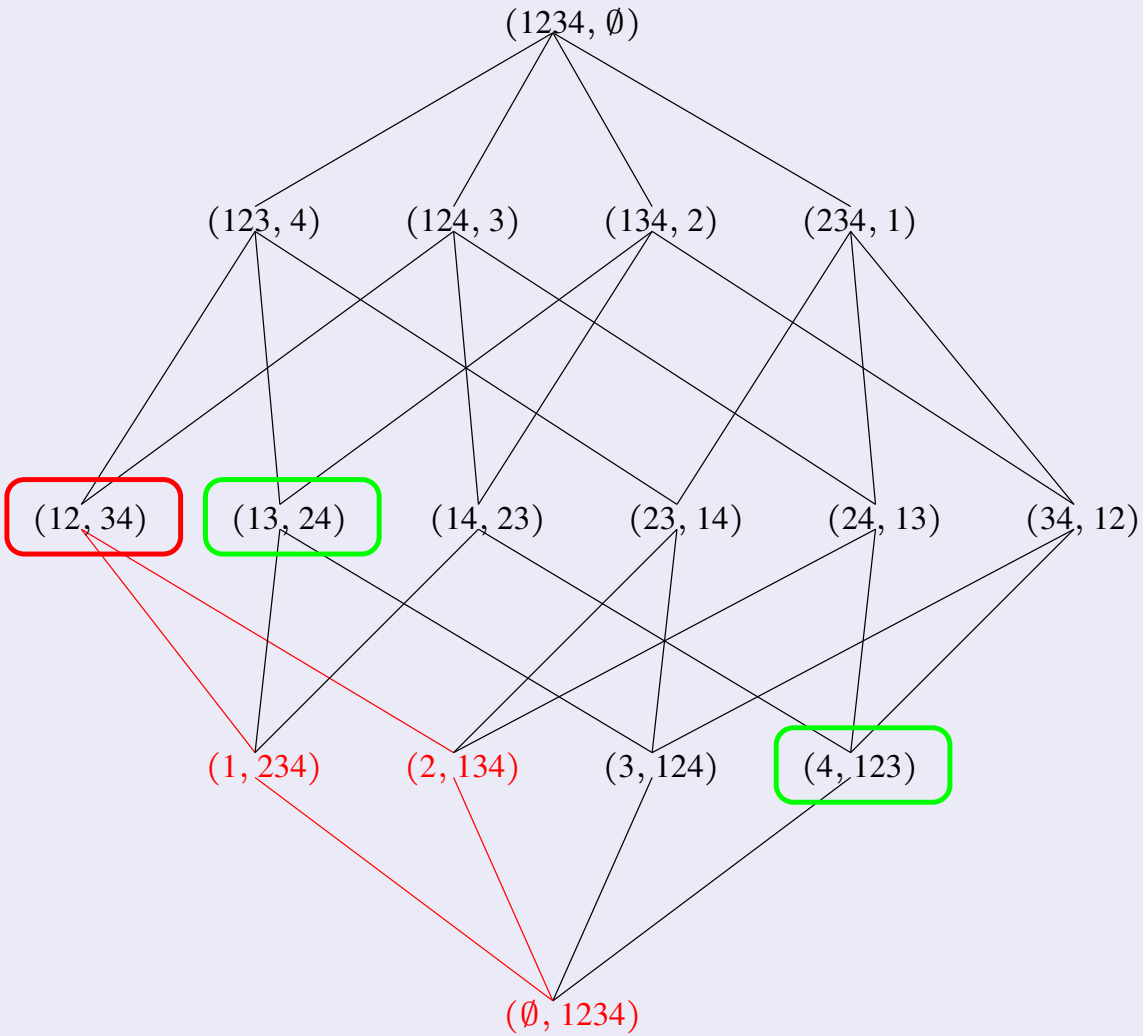
$14, 23, 24, 34$

Steps

Find one maximal relaxation.

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

$14, 23, 24, 34$

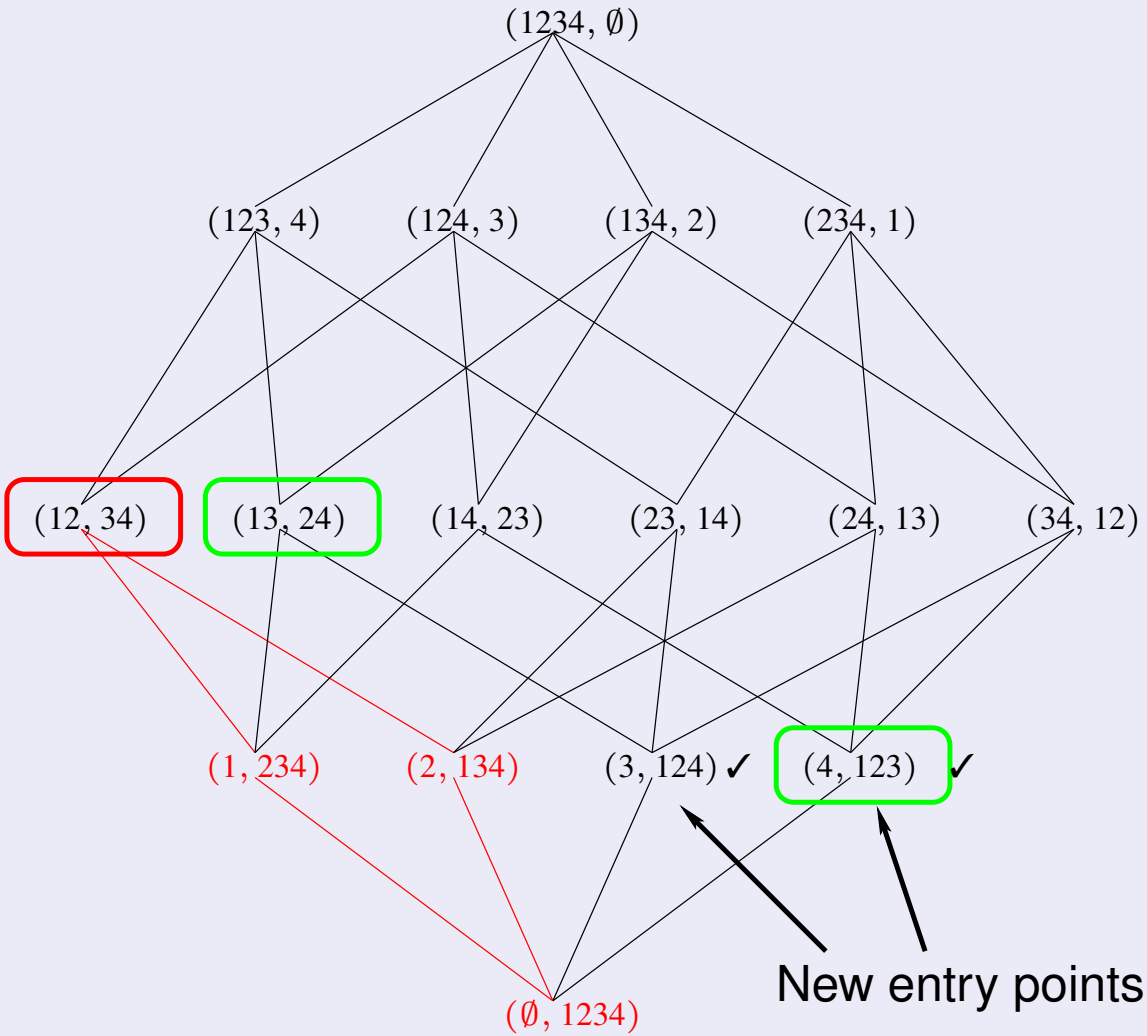
Steps

Find one maximal relaxation.

➔ All subsets of it are "forbidden".

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

$14, 23, 24, 34$

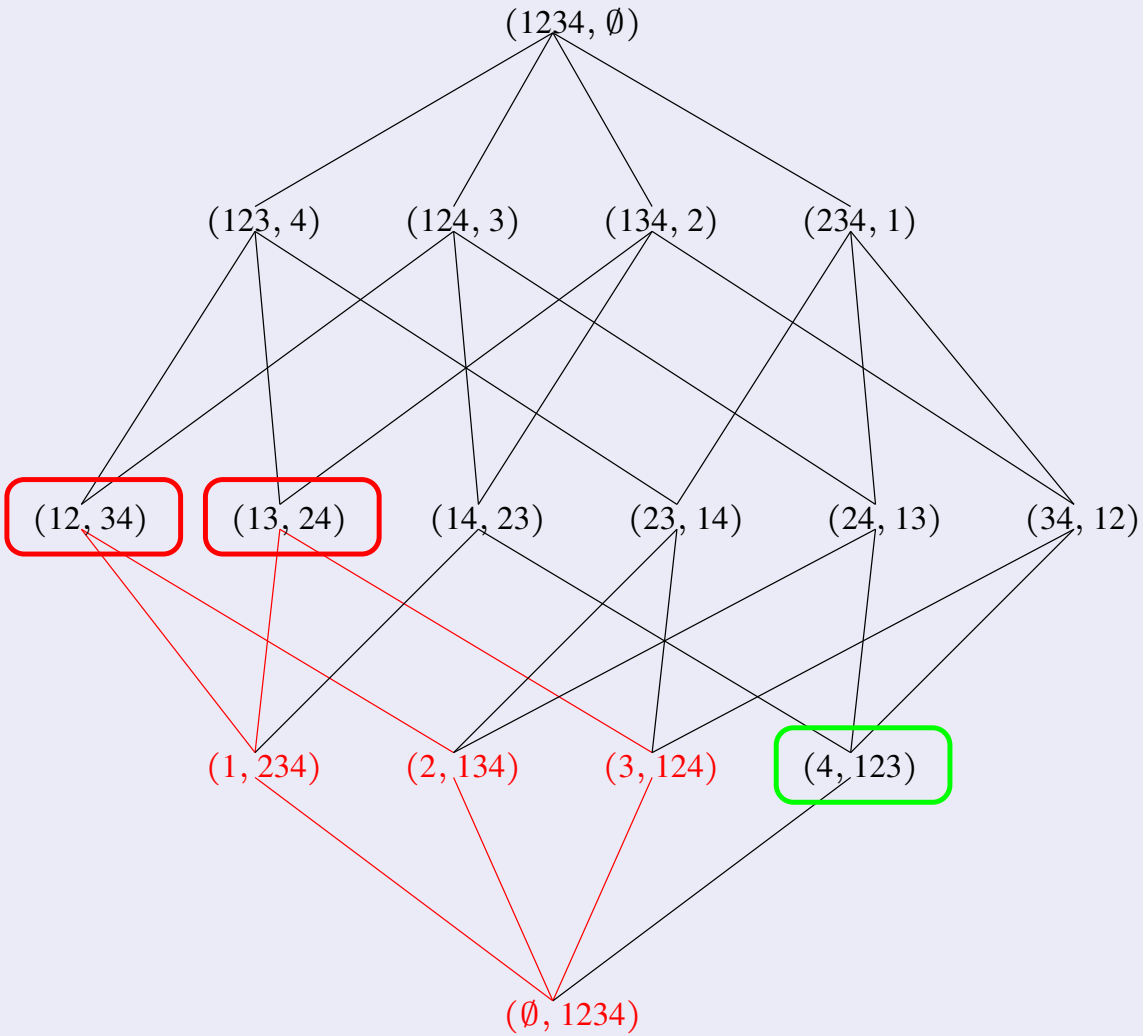
Steps

Compute all the *minimal hitting sets* of the exclusion sets found so far.

➔ They are minimal sets *incomparable* with any relaxation found so far.

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

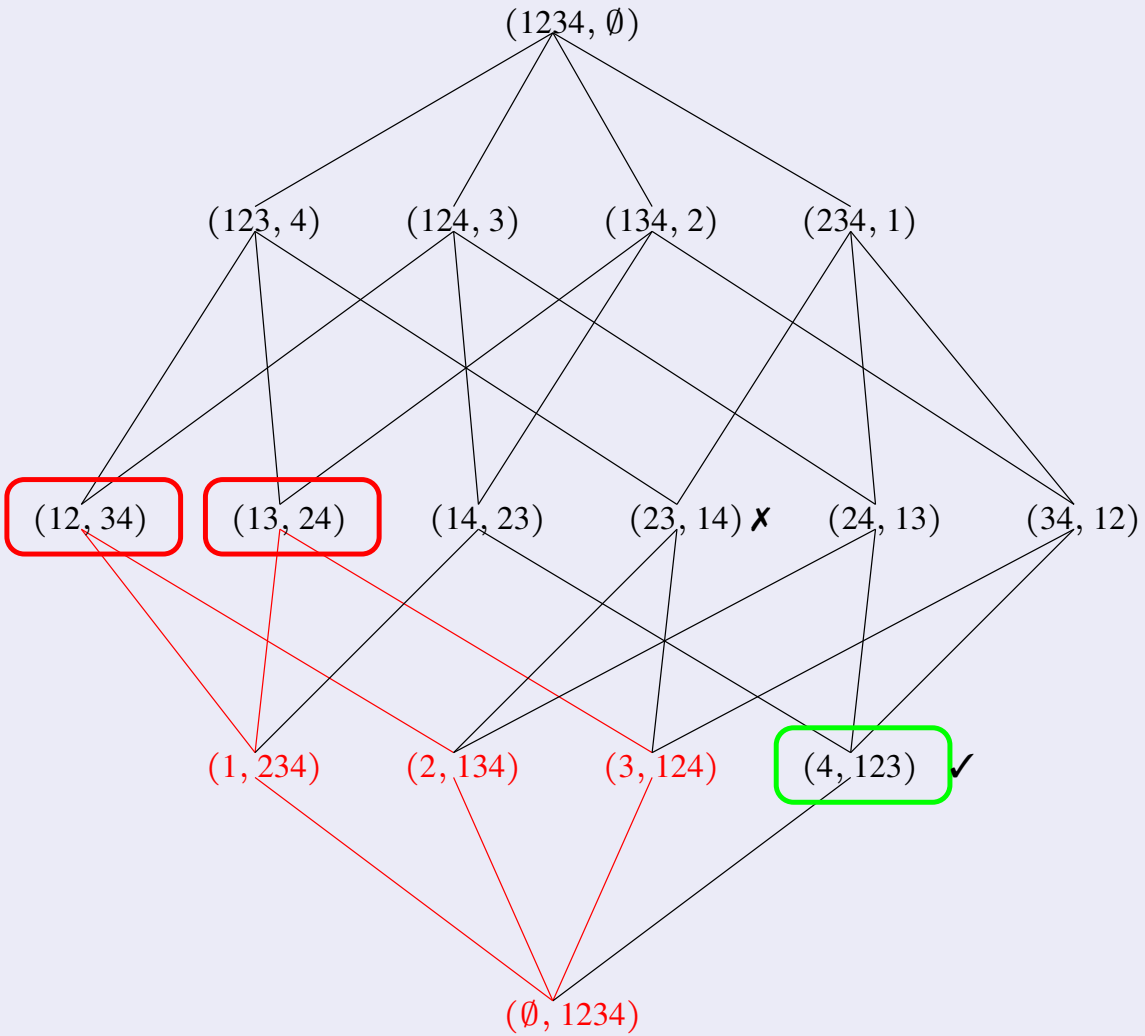
$14, 23, 24, 34$

Steps

Pick a *consistent* one, and extend it to a maximal relaxation.
➔ It will be different from any maximal relaxation found so far.

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

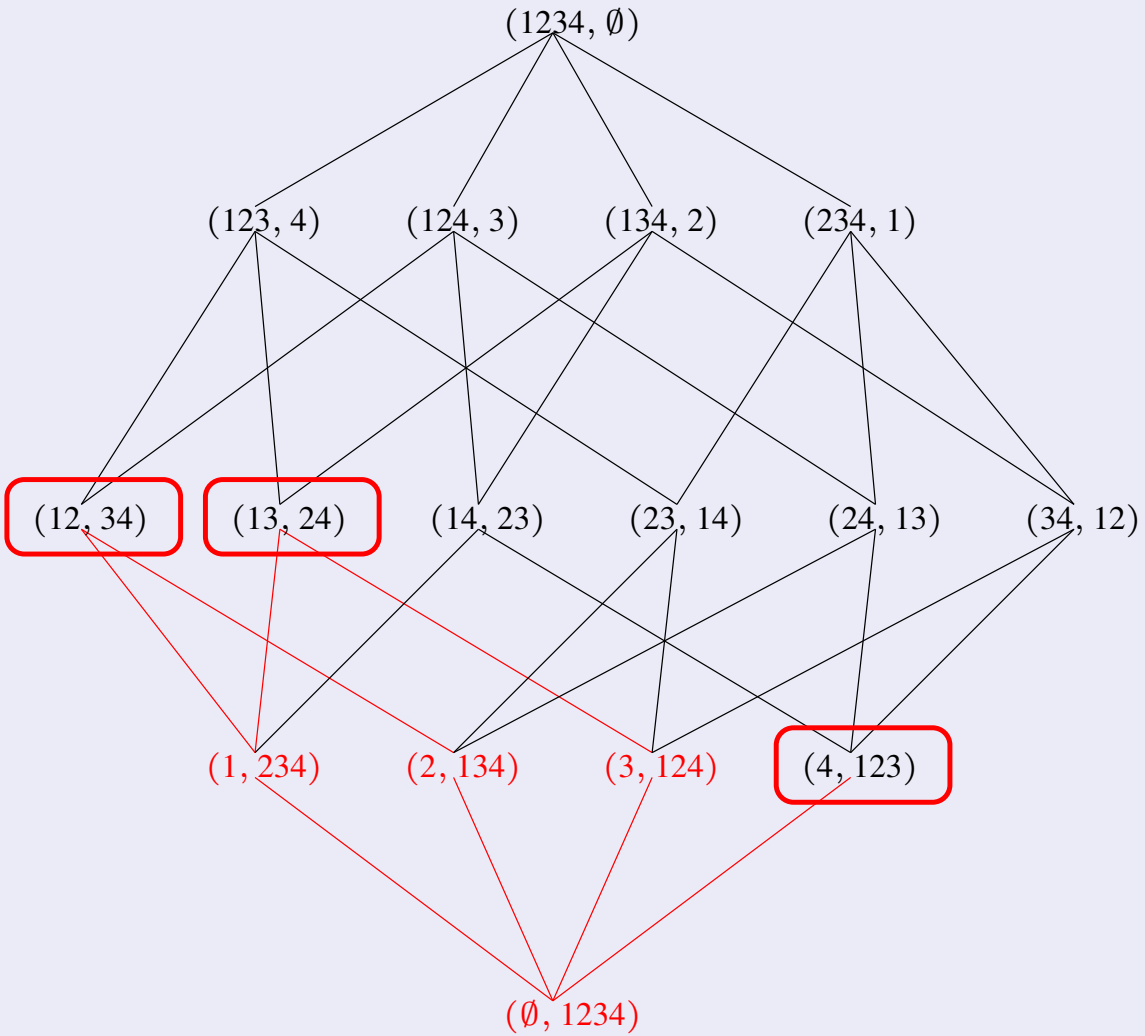
14, 23, 24, 34

Steps

23 is not consistent: it is a *minimal conflict*.
➔ Pick 4 (the only consistent minimal hitting set).

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

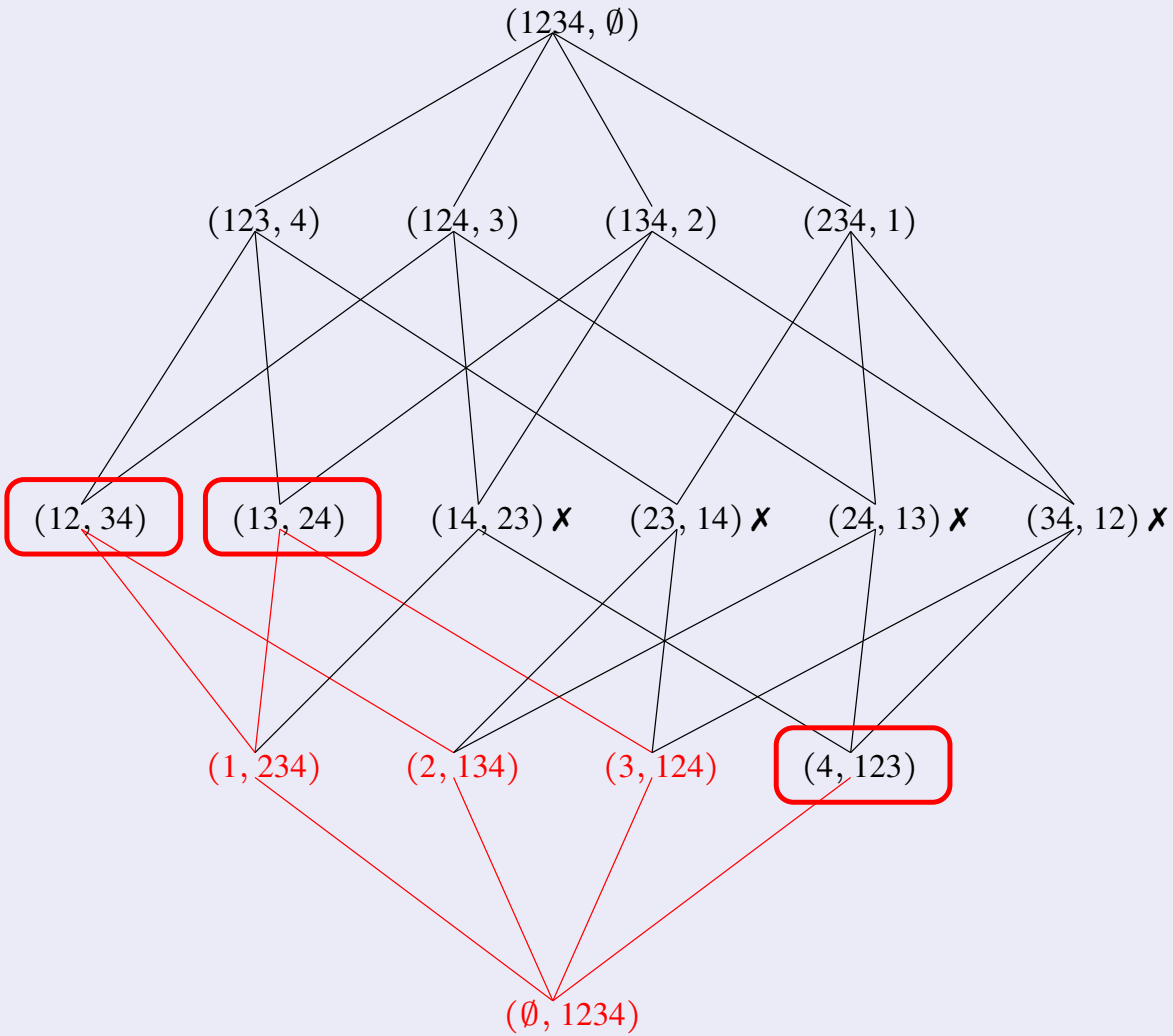
$14, 23, 24, 34$

Steps

It is already a maximal relaxation.

Dualize and Advance (Bailey and Stuckey, 2005) [2]

Explanations



Conflicts

14, 23, 24, 34

Steps

All the minimal hitting sets are inconsistent: they're all the minimal conflicts.

➔ The algorithm ends.

Representative Explanations [10]

Observations

1. **Conflict**: doesn't guide the user to solving the problem
2. **Single relaxation**: may not satisfy the user desires
3. **All relaxations**: can theoretically be too large

An Alternative Approach

- show a set of relaxations
- that must be *representative* of all possible relaxations

as a trade-off between compactness and comprehensiveness

Representative Explanations [10]

Observations

1. **Conflict**: doesn't guide the user to solving the problem
2. **Single relaxation**: may not satisfy the user desires
3. **All relaxations**: can theoretically be too large

An Alternative Approach

- show a set of relaxations
- that must be *representative* of all possible relaxations

as a trade-off between compactness and comprehensiveness

Representative Explanations [10]

Observations

1. **Conflict**: doesn't guide the user to solving the problem
2. **Single relaxation**: may not satisfy the user desires
3. **All relaxations**: can theoretically be too large

An Alternative Approach

- show a set of relaxations
- that must be *representative* of all possible relaxations

as a trade-off between compactness and comprehensiveness

Representative Explanations [10]

Observations

1. **Conflict**: doesn't guide the user to solving the problem
2. **Single relaxation**: may not satisfy the user desires
3. **All relaxations**: can theoretically be too large

An Alternative Approach

- show a set of relaxations
- that must be *representative* of all possible relaxations

as a trade-off between compactness and comprehensiveness

Example

Car configuration

Option	Cost
Roof rack	500
Convertible	500
CD Player	500
Leather Seats	2600

➠ Convertible cars cannot have roof racks.

User constraints

c_1	Total cost ≤ 3000
c_2	Roof rack
c_3	Convertible
c_4	CD Player
c_5	Leather Seats

Relaxations: $\{c_1c_2\}$, $\{c_1c_5\}$ are consistent
Maximality: $\{c_1c_2c_4\}$ is still consistent, but no more constraint can be added to $\{c_1c_5\}$.

Example

Car configuration

Option	Cost
Roof rack	500
Convertible	500
CD Player	500
Leather Seats	2600

➠ Convertible cars cannot have roof racks.

User constraints

c_1	Total cost ≤ 3000
c_2	Roof rack
c_3	Convertible
c_4	CD Player
c_5	Leather Seats

Relaxations: $\{c_1c_2\}$, $\{c_1c_5\}$ are consistent
Maximality: $\{c_1c_2c_4\}$ is still consistent, but no more constraint can be added to $\{c_1c_5\}$.

Showing many Explanations

Representative set of explanations

c_1	c_2	c_3	c_4	c_5
✗	✗	✓	✓	✓
✗	✓	✗	✓	✓
✓	✗	✓	✓	✗
✓	✓	✗	✓	✗
✓	✗	✗	✗	✓

- Every constraint that can be kept is kept at least once
- Every constraint that can be relaxed is relaxed at least once
- Minimal (setwise) representative set of explanations

Complexity

Decision problems

- Does a maximal relaxation contain a given constraint?
  ➥ Polynomial (in terms of number of calls to the consistency checker)
- Does a minimal exclusion set contain a given constraint?
  ➥ NP-Complete (with an oracle for the consistency checker)

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Generating Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Empirical Analysis

Random problems

- 15 variables,
- One background table constraint, with varying tightness
- Random assignments on the variables

Renault

- Real-world problem
- 99 variables
- 2.8×10^{12} solutions
- 30 variables randomly assigned

Empirical Analysis

Random problems

- 15 variables,
- One background table constraint, with varying tightness
- Random assignments on the variables

Renault

- Real-world problem
- 99 variables
- 2.8×10^{12} solutions
- 30 variables randomly assigned

Behaviour

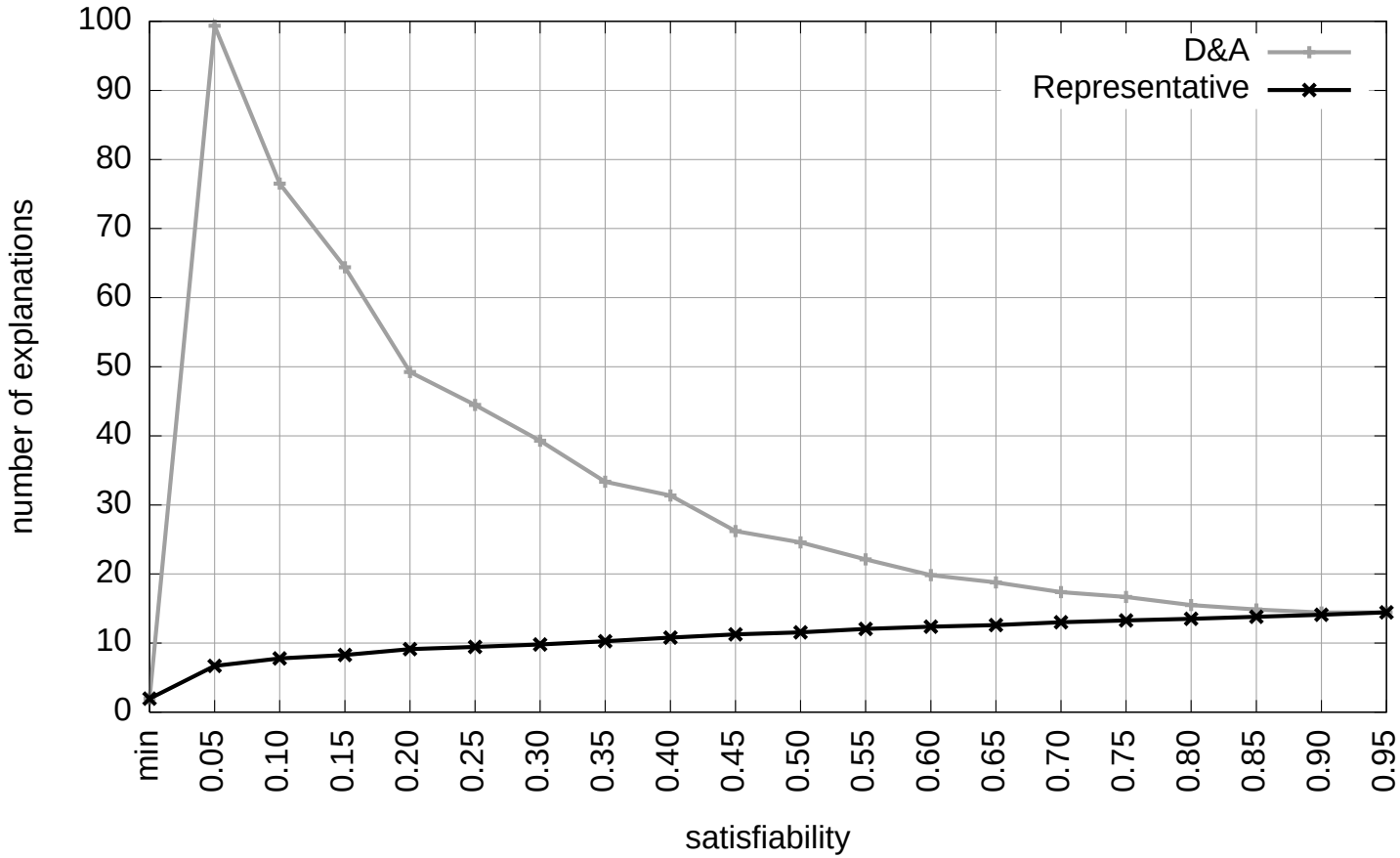


Figure: Number of explanations

Behaviour

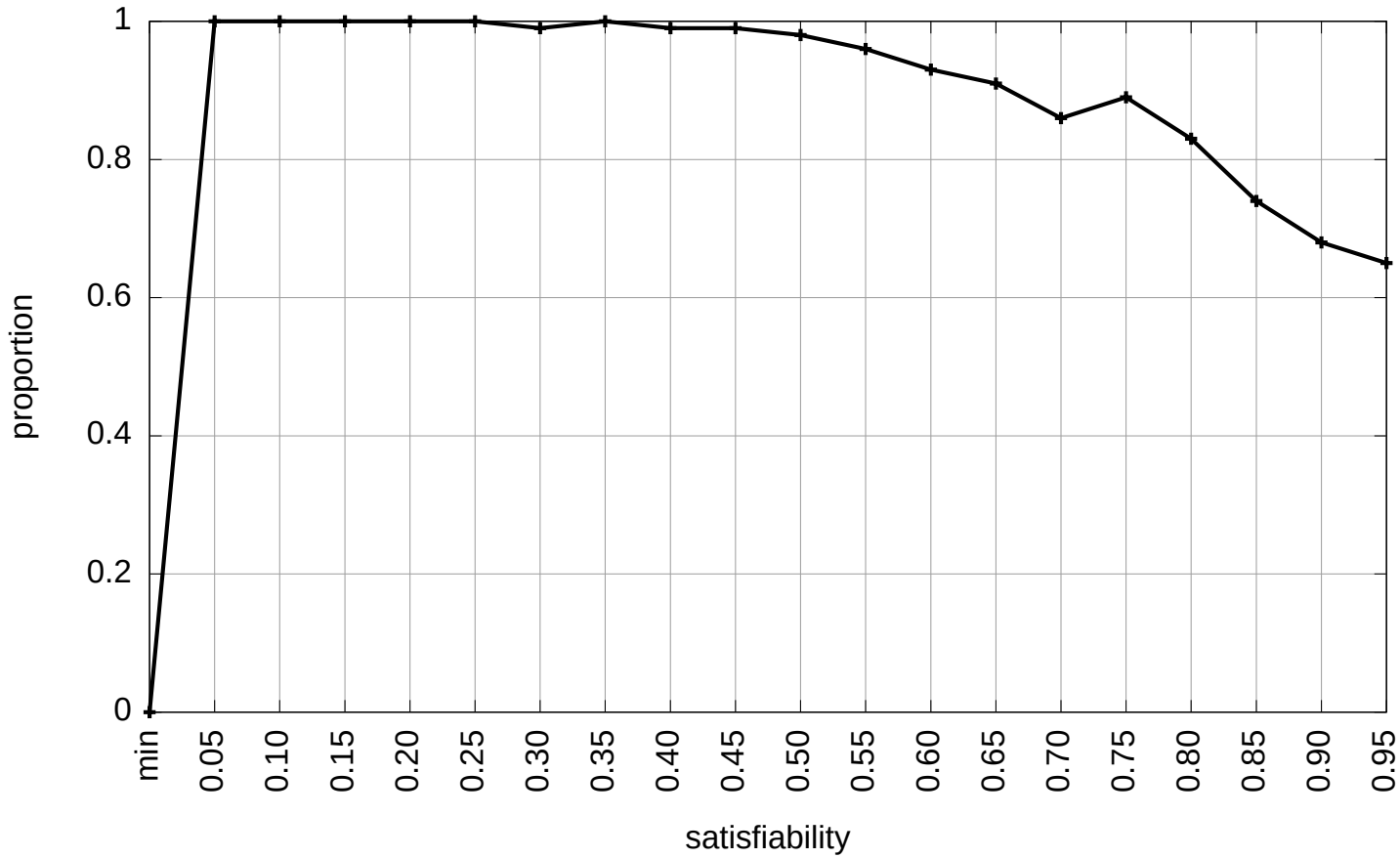


Figure: Proportion of “true instances”

Empirical Analysis

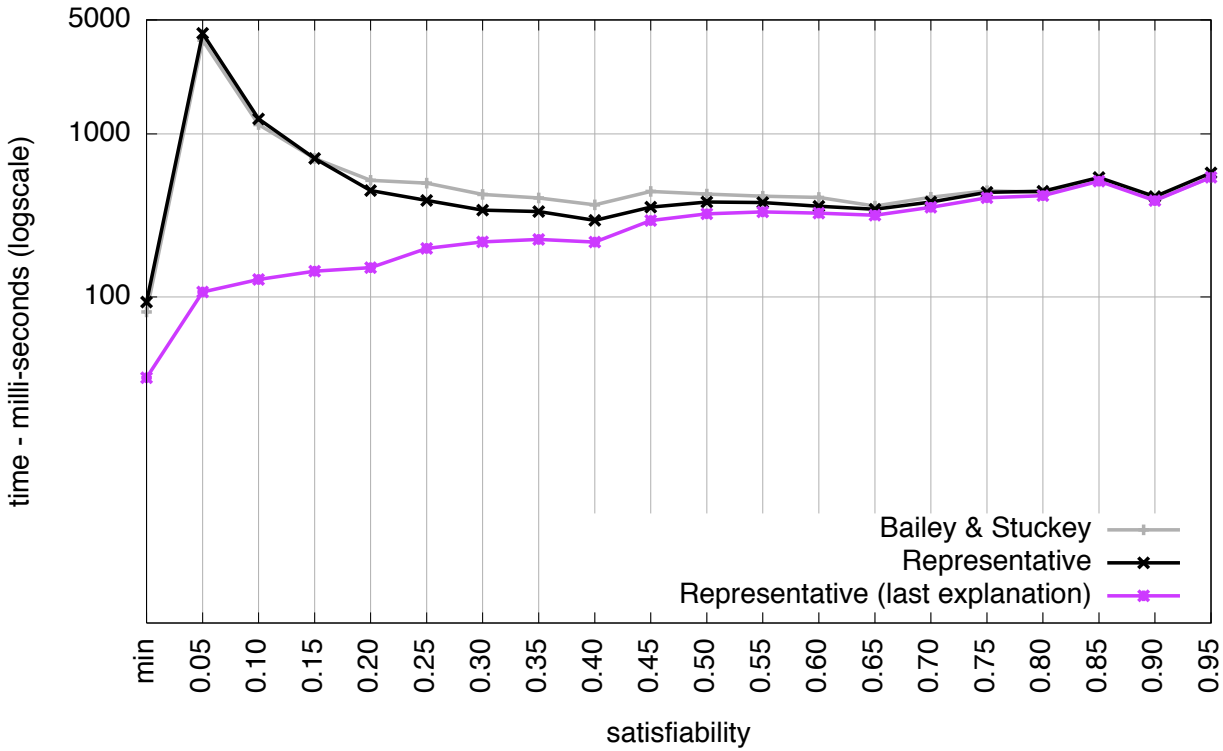


Figure: Running times

Empirical Analysis

Renault instance

Instance	Baseline time	Baseline #exps	RepresentativeXplain time last	RepresentativeXplain time all	RepresentativeXplain #exps
renault 10^6	474.76	17	318.87	618.76	3
renault 10^7	263.95	11	125.51	324.71	3
renault 10^8	205.82	8	97.98	232.32	3
renault 10^9	293.00	12	139.67	350.51	3

Table: Running times for the Renault instances

Explanations and Solubility [11]

Principle – Automaton representation

Common approach [1]:

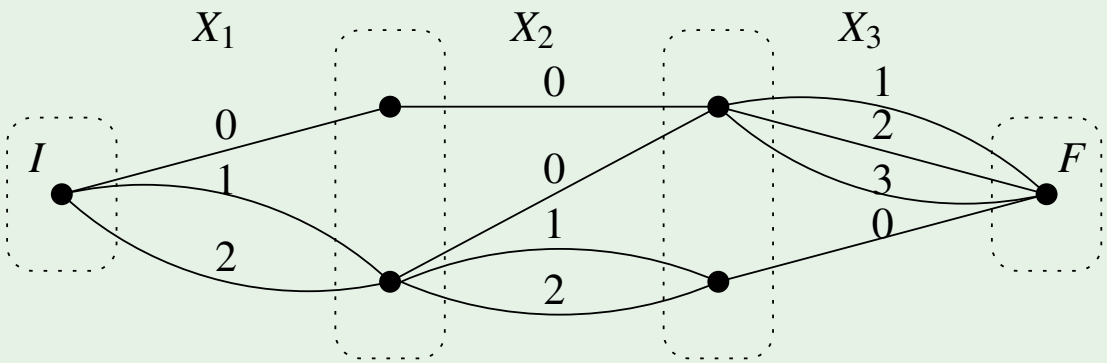
- *compile* constraint satisfaction problems
- allowing more operations to be tractable in practice.

Representation:

- Only the background constraints are compiled (they do not change)
- The user constraints are implicit (they change), information is associated with states and transitions

Automaton representation

Example (problem representation)

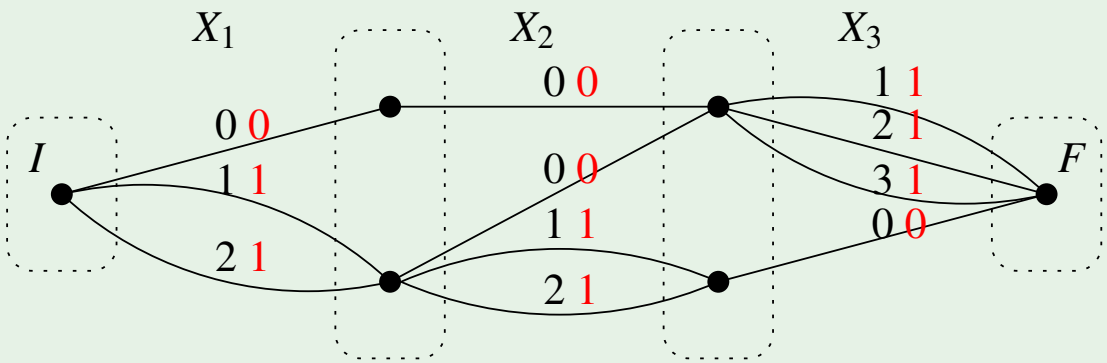


This represents a problem that has:

- three variables X_1, X_2, X_3 on $\{a, b, c\}$
- 13 solutions, including 001, 002, 103, etc.

Automaton representation

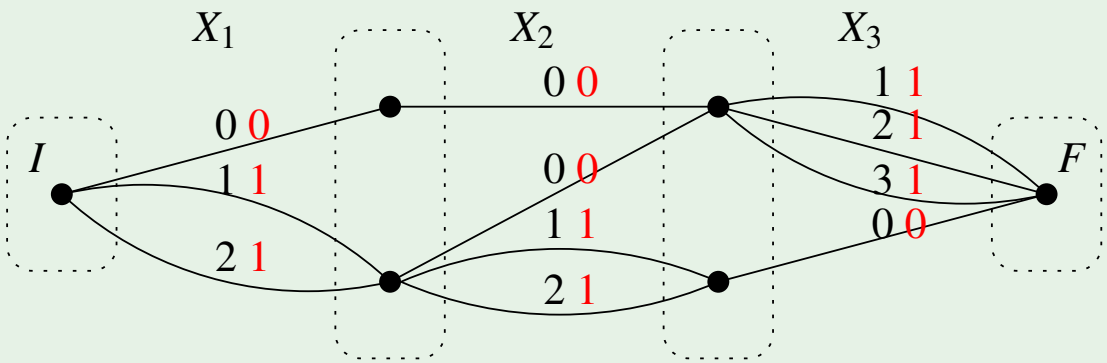
Example (user constraints)



- User constraints: for every i, $X_i = 0$
- For a transition t, we associate a cost $c(t)$.
- $c(t) = 0$ on valid transitions, $c(t) > 0$ on invalid transitions.

Automaton representation

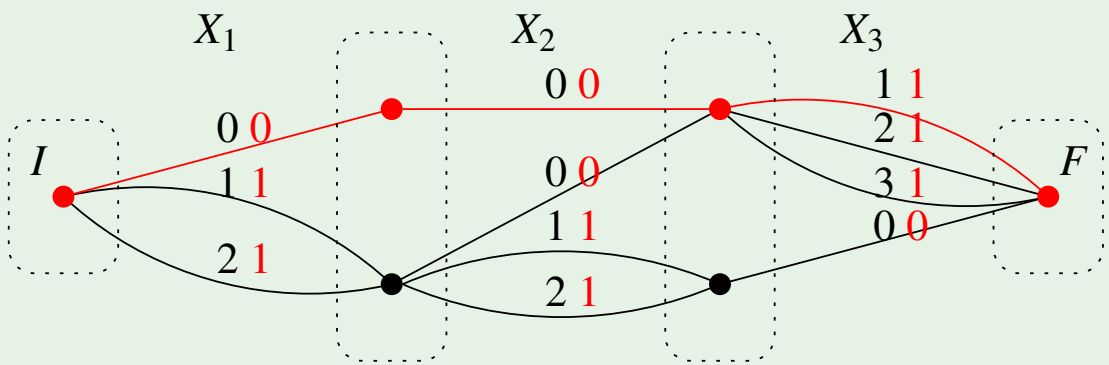
Example (user constraints)



- User constraints: for every i, $X_i = 0$
- For a transition t, we associate a cost $c(t)$.
- $c(t) = 0$ on valid transitions, $c(t) > 0$ on invalid transitions.

Automaton representation

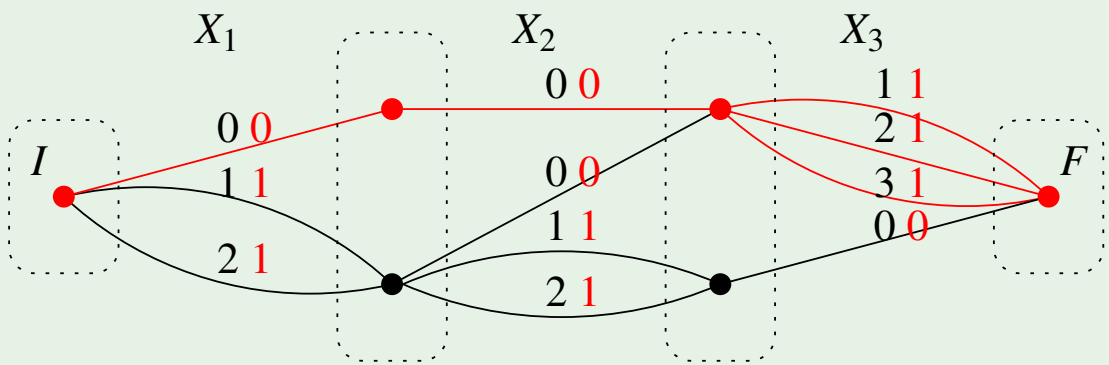
Example (relaxations)



- A path is associated with a relaxation (e.g. $\{1, 2\}$).
- Several paths per relaxation (e.g. 3 paths recognise $\{1, 2\}$)
- Their cost corresponds to the cardinality of the corresponding exclusion set (here 1).
- A shortest path corresponds to a max cardinality relaxation.

Automaton representation

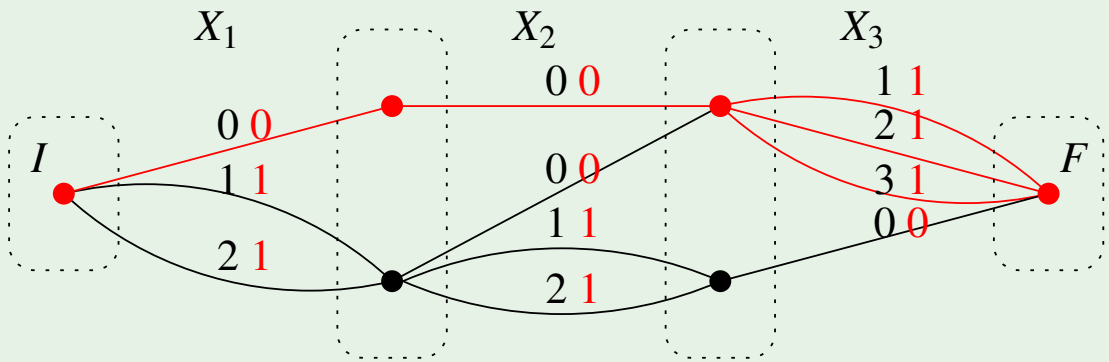
Example (relaxations)



- A path is associated with a relaxation (e.g. $\{1, 2\}$).
- Several paths per relaxation (e.g. 3 paths recognise $\{1, 2\}$)
- Their cost corresponds to the cardinality of the corresponding exclusion set (here 1).
- A shortest path corresponds to a max cardinality relaxation.

Automaton representation

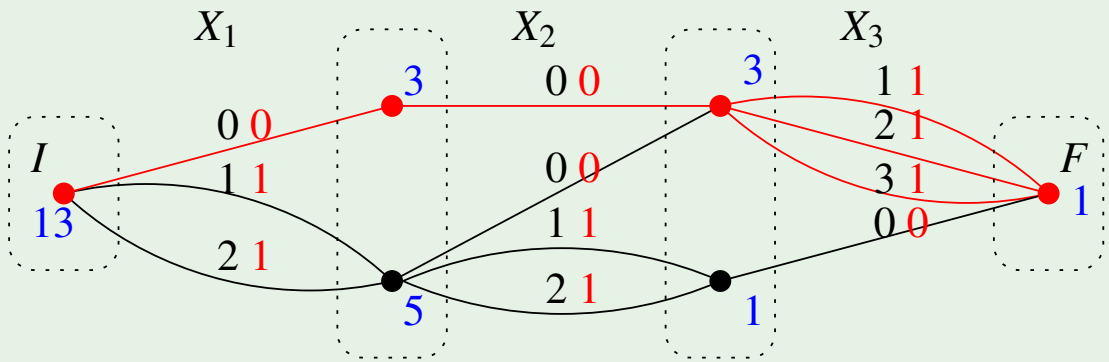
Example (relaxations)



- A path is associated with a relaxation (e.g. $\{1, 2\}$).
- Several paths per relaxation (e.g. 3 paths recognise $\{1, 2\}$)
- Their cost corresponds to the cardinality of the corresponding exclusion set (here 1).
- A shortest path corresponds to a max cardinality relaxation.

Automaton representation

Example (number of solutions)



- The number of solutions of a state is the sum of the number of solutions of its successors.
- The number of solutions of a relaxation is the number of paths that recognise it (e.g. $\{1, 2\}$ is compatible with 3 solutions).

Automaton-based Explanation Algorithms

Two exact algorithms

- Find the most/least soluble longest relaxation
  ➥ linear in the size of the automaton.
- Find the most/least soluble maximal relaxation
  ➥ linear in the size of the automaton $\times$ the number of maximal relaxations.

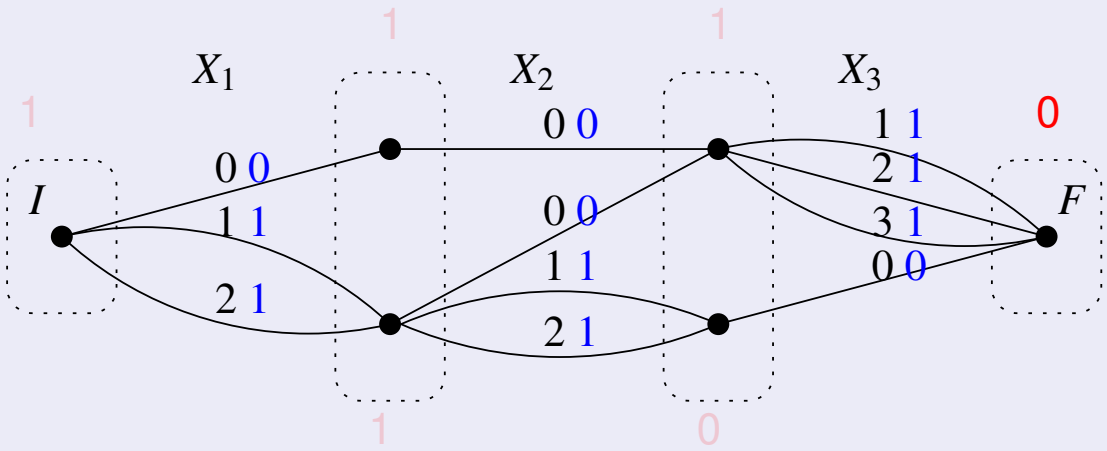
Automaton-based Explanation Algorithms

Two exact algorithms

- Find the most/least soluble longest relaxation
  ➥ linear in the size of the automaton.
- Find the most/least soluble maximal relaxation
  ➥ linear in the size of the automaton $\times$ the number of maximal relaxations.

Shortest path

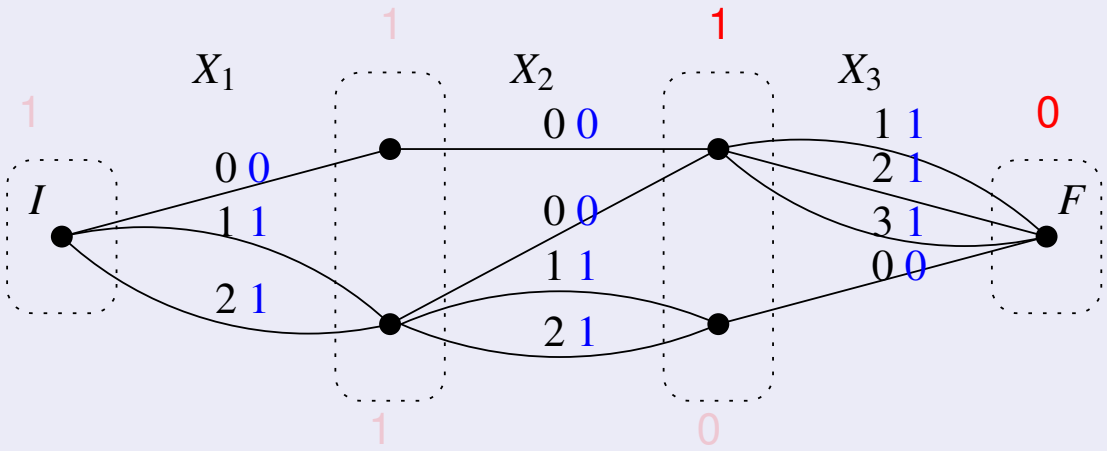
Example



- For a state q, let $c(q)$ be the size of a shortest path from q to F.
- $c(q) = \min(c(t) + c(q'))$

Shortest path

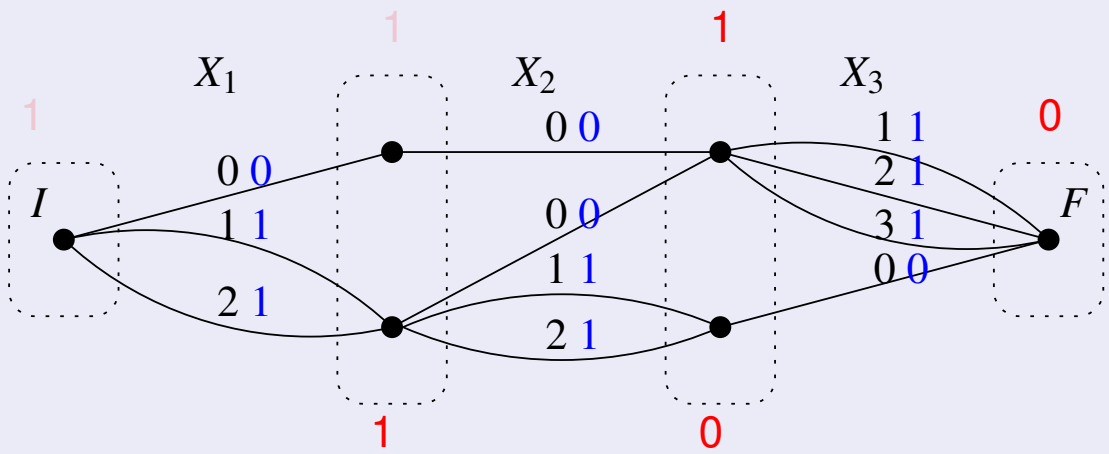
Example



- For a state q, let $c(q)$ be the size of a shortest path from q to F.
- $c(q) = \min(c(t) + c(q'))$

Shortest path

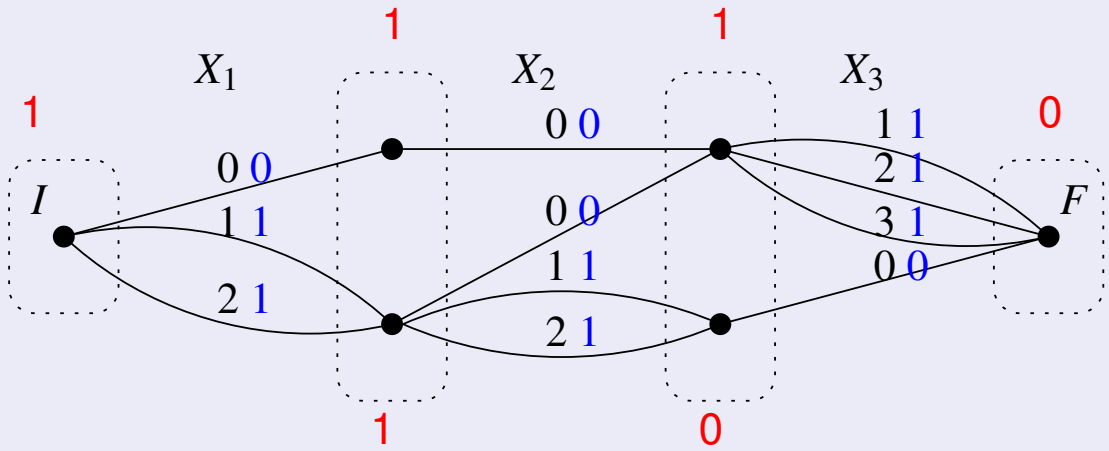
Example



- For a state q, let $c(q)$ be the size of a shortest path from q to F.
- $c(q) = \min(c(t) + c(q'))$

Shortest path

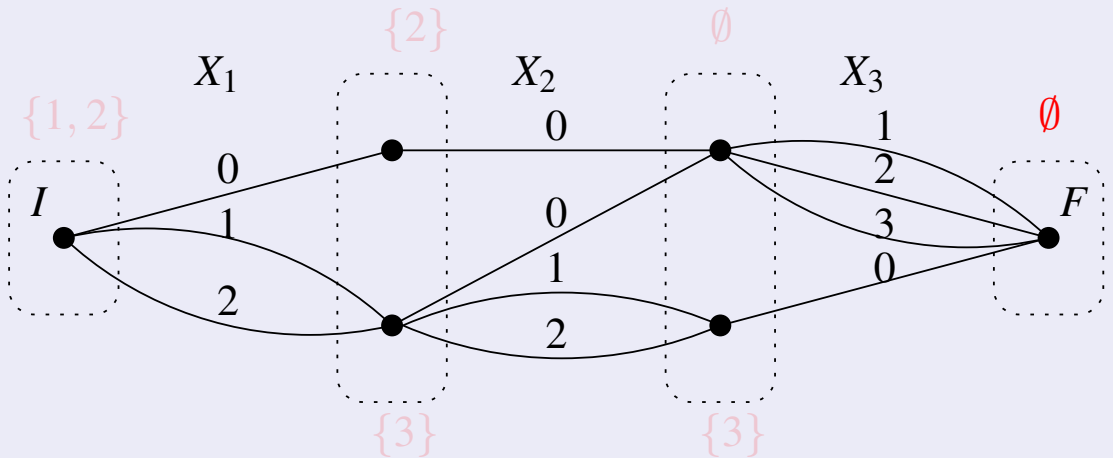
Example



- For a state q, let $c(q)$ be the size of a shortest path from q to F.
- $c(q) = \min(c(t) + c(q'))$

Longest relaxation

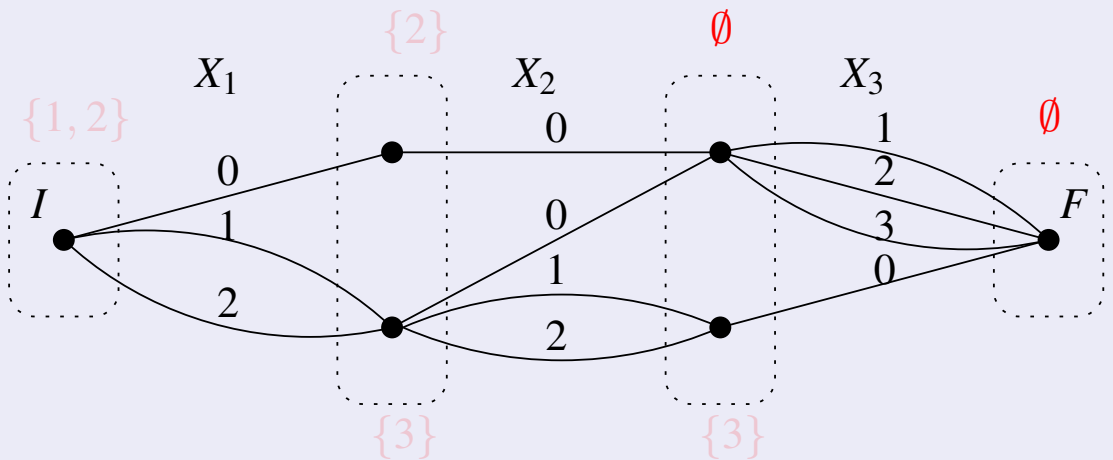
Example



- For a state q, we associate instead the relaxation corresponding to *a* shortest path.
- Several maximal cardinality relaxations can correspond to a state (choose arbitrarily)
- The relaxation associated with I is a maximal cardinality relaxation.

Longest relaxation

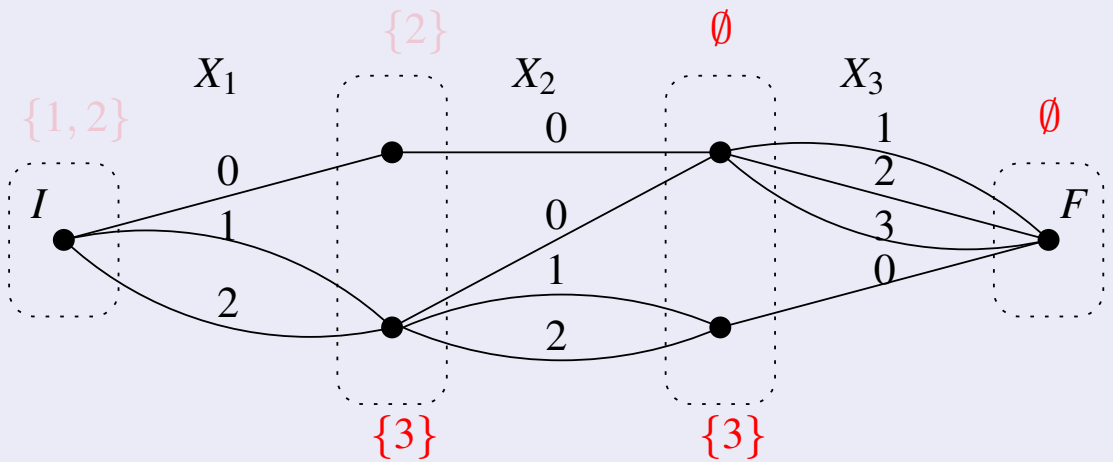
Example



- For a state q, we associate instead the relaxation corresponding to *a* shortest path.
- Several maximal cardinality relaxations can correspond to a state (choose arbitrarily)
- The relaxation associated with I is a maximal cardinality relaxation.

Longest relaxation

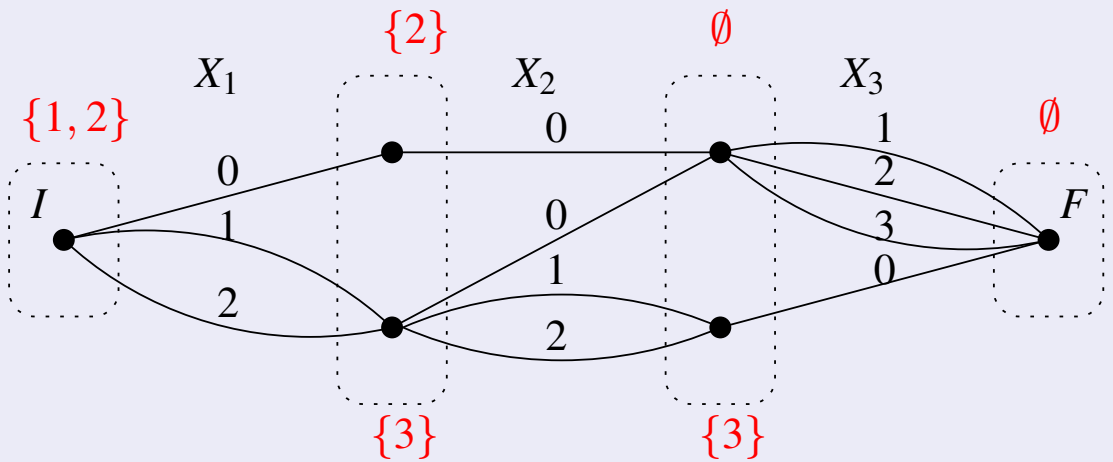
Example



- For a state q, we associate instead the relaxation corresponding to *a* shortest path.
- Several maximal cardinality relaxations can correspond to a state (choose arbitrarily)
- The relaxation associated with I is a maximal cardinality relaxation.

Longest relaxation

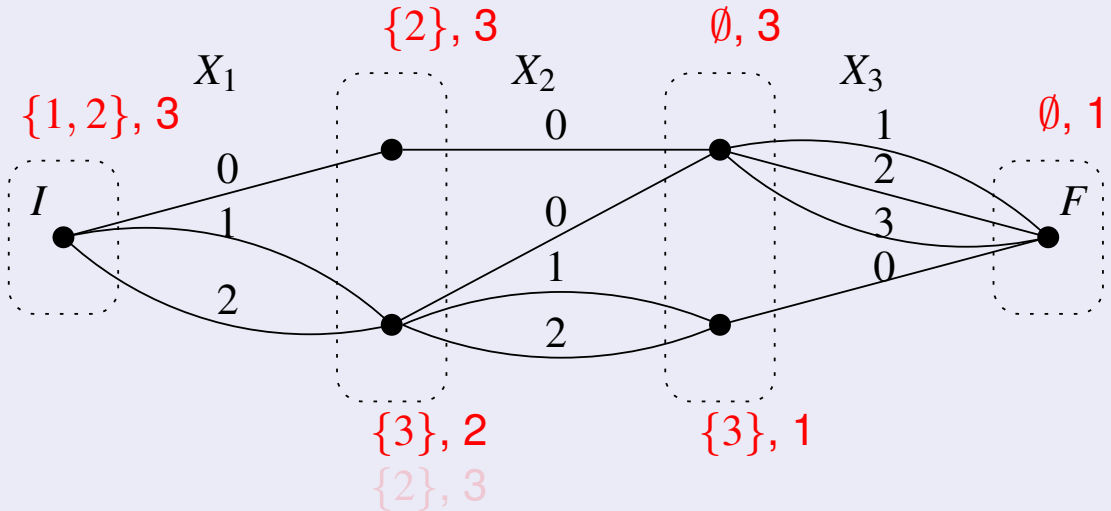
Example



- For a state q, we associate instead the relaxation corresponding to *a* shortest path.
- Several maximal cardinality relaxations can correspond to a state (choose arbitrarily)
- The relaxation associated with I is a maximal cardinality relaxation.

Most soluble longest relaxation

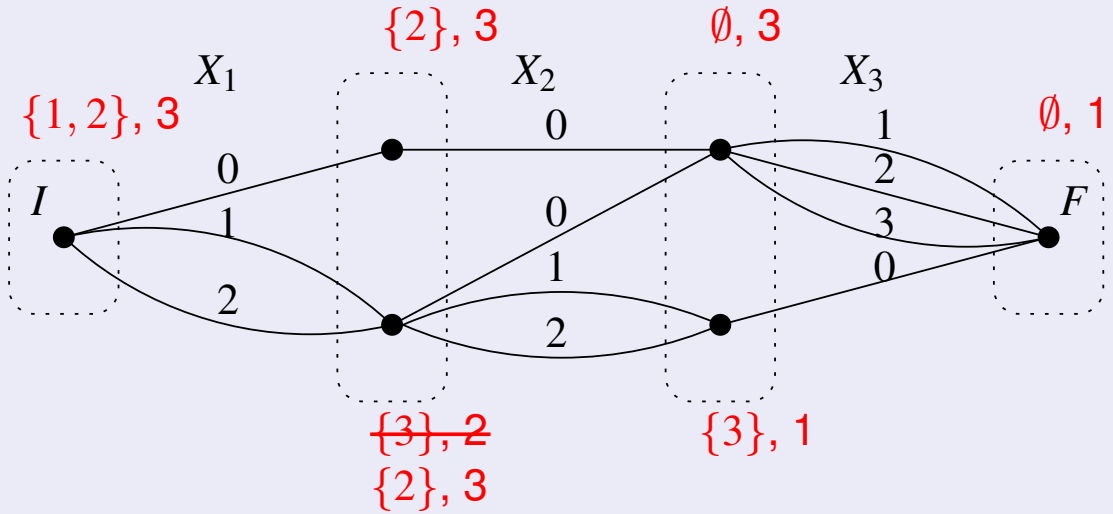
Example



- We can also keep track of the number of paths supporting the relaxation of each state.
- Choose maximal cardinality relaxation with the highest number of solutions
- The relaxation $\{1, 2\}$ is compatible with 3 solutions.

Most soluble longest relaxation

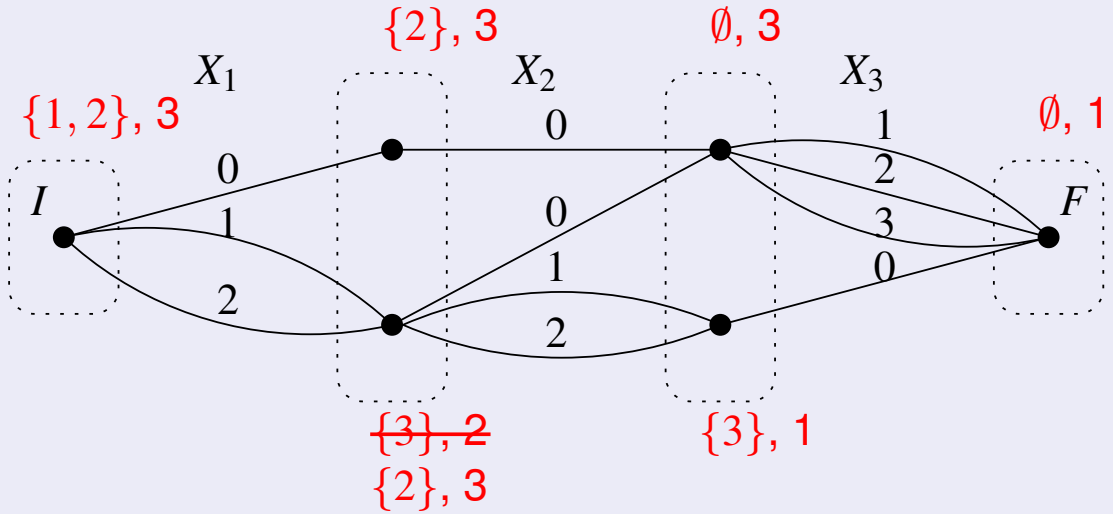
Example



- We can also keep track of the number of paths supporting the relaxation of each state.
- Choose maximal cardinality relaxation with the highest number of solutions
- The relaxation $\{1, 2\}$ is compatible with 3 solutions.

Most soluble longest relaxation

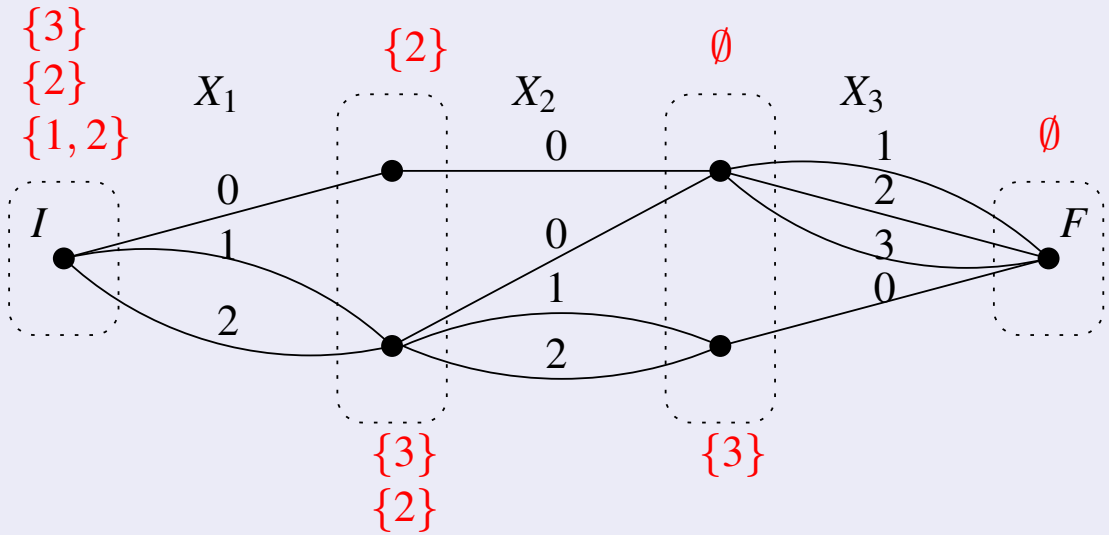
Example



- We can also keep track of the number of paths supporting the relaxation of each state.
- Choose maximal cardinality relaxation with the highest number of solutions
- The relaxation $\{1, 2\}$ is compatible with 3 solutions.

Enumerating maximal relaxations

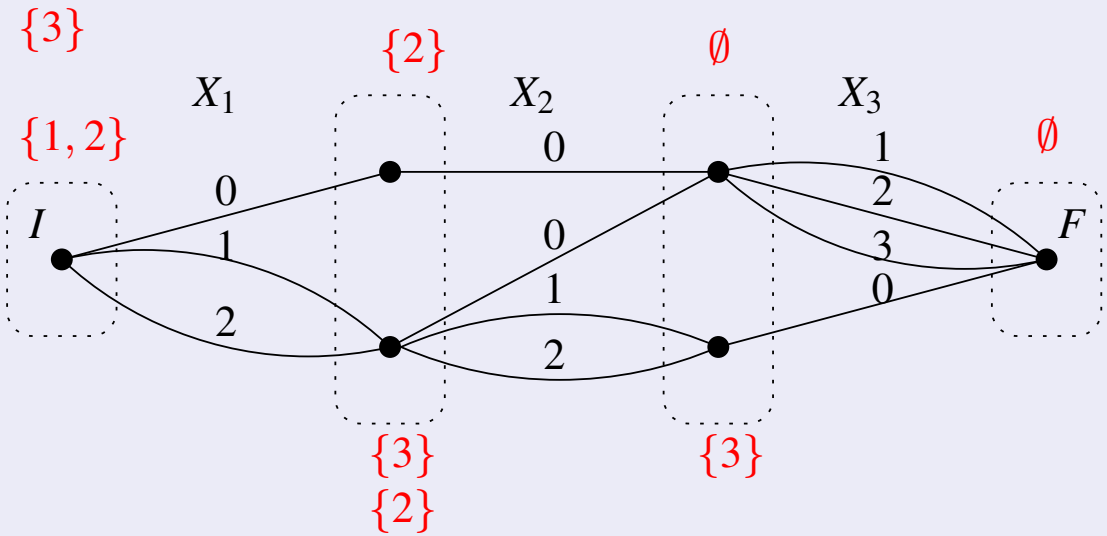
Example



- Keep track at each state of all the relaxations.
- Filter out subsumed sets.

Enumerating maximal relaxations

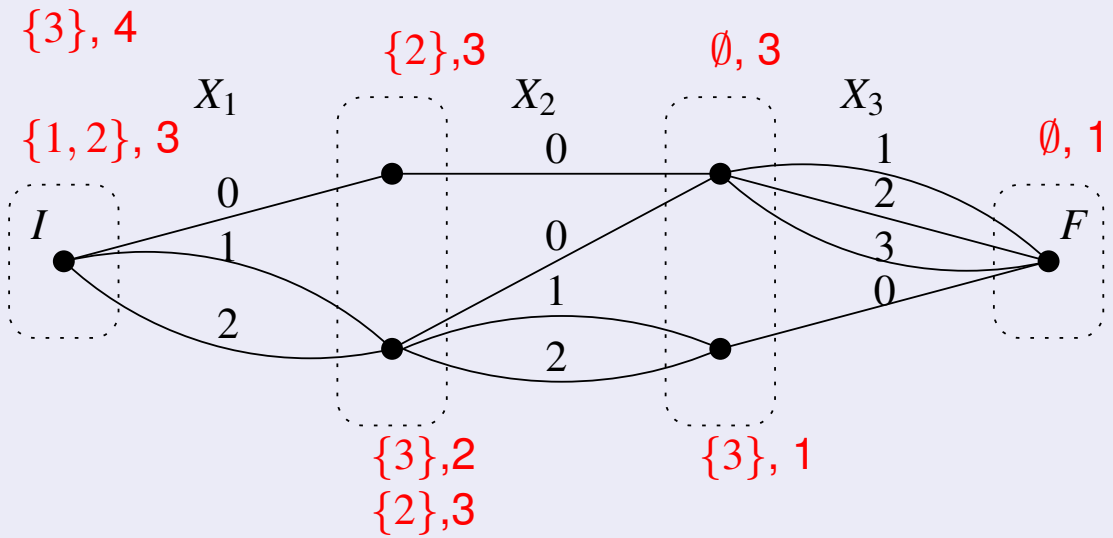
Example



- Keep track at each state of all the relaxations.
- Filter out subsumed sets.

Enumerating most soluble maximal relaxations

Example



- Add in the corresponding number of solutions
- No relaxation can be filtered based on its number of solutions
- $\{3\}$ is the most soluble maximal relaxation.

Outline

The Importance of Preferences [6]

- Explaining the decision in rational decision making
- Explanations are crucial for interactive decision making: *"Accept or critique"*
- Different decision making problems lead to different explanation problems

Decision Making with Preferences

Problem

- An agent can choose one of several actions.
- Each action leads to an outcome.
- The agent prefers certain outcomes to others.

Example

Choose the ADT-2009 location:

- Actions: ask Jose Figueira, ask Francesca Rossi, ask Alexis Tsoukias, ask Barry O'Sullivan.
- Outcomes: Coimbra, Venice, Paris, Cork.
- Preferences: please tell me.

Assumptions of the Basic Model

- There is a single agent.
- The set of actions is known.
- The outcome of an action is certain and known.
- The preferences between outcomes are partially known.

Preference Modeling

Concept

- The agent prefers some outcomes as least as much as other outcomes.
- This is modeled by a binary relation over outcomes.
- This relation is reflexive and transitive, i.e. it is a preorder.
- The relation may be complete, but this need not to be so.

Example

- Weak preferences:
  $Venice \succsim Coimbra$
  $Venice \succsim Cork$
  $Coimbra \succsim Cork$
  $Cork \succsim Coimbra$
  $x \succsim x$ for all x
- Incomparable outcomes:
  Paris and *Venice*
  Paris and *Cork*
  Paris and *Coimbra*

Strict Preferences and Indifference

Concept

- An agent is indifferent between A and B iff she prefers A at least as much as B and B as least as much as A.
- An agent strictly prefers A to B iff she prefers A at least as much as B, but does not prefer B as least as much as A.
- We split the preorder $\succsim$ into strict preferences $\succ$ and indifference $\sim$.

Example

- Weak preferences:
  $Venice \succsim Coimbra$
  $Venice \succsim Cork$
  $Coimbra \succsim Cork$
  $Cork \succsim Coimbra$
- Indifference:
  $Cork \sim Coimbra$
- Strict preferences:
  $Venice \succ Coimbra$
  $Venice \succ Cork$

Rational Decision Making

Problem

Given

- a set of actions A,
- a set of outcomes Ω,
- an outcome of actions $z : A \rightarrow \Omega$,
- a preference preorder $\succsim$ on Ω,

make a decision by choosing an action from A.

Rational Decision

- An outcome ω^* is optimal (most preferred) iff no other outcome is strictly preferred to ω^*.
- Rational decisions are the actions that have an optimal outcome.
- There may be multiple optimal outcomes and multiple rational decisions.

Decision Making under Constraints

Problem

- An agent needs to make a decision in different situations.
- The agent's preferences usually do not change.
- However, the actions may differ from one situation to the other.
- Constraints describe which actions are possible (feasible) in a situation.

Example

- Choose the ADT location in several years.
- Constraint: Alexis Tsoukias does not want to organize ADT in 2009
- Your preferences should not change in different years.

Rational Decision Making under Constraints

Problem

Given

- a set of actions A,
- a subset of feasible actions $F \subseteq A$,
- a set of outcomes Ω,
- an outcome of actions $z : A \rightarrow \Omega$,
- a preference preorder $\succsim$ on Ω,

make a decision.

Rational Decision

- An outcome ω^* is feasible iff some feasible action in F leads to ω^*.
- A feasible outcome ω^* is optimal iff no other feasible outcome is strictly preferred to ω^*.
- Rational decisions are the actions with optimal outcomes.
- There may be multiple optimal outcomes and rational decisions.

Does the agent respect her preferences?

Rational Decision Making

- The decisions made by the agent are consistent w.r.t. her preferences:
  - If the agent prefers A to B and A is possible (feasible), then she will not choose B.
  - If the agent prefers A to B, but chooses B then A is not possible (feasible).
- Hence the decision can be explained in terms of the preferences and the infeasibility of actions.

Non-rational Decision Making

- The agent's decision is not the best possible and cannot be explained in terms of her preferences.

How to explain a decision?

Why this decision?

- Why not an action with a better outcome?
- Why not an action with another optimal outcome?

Different answers for different cases:

1. Complete preference orders: unique optimal outcome.
2. Partial preference orders: multiple optimal outcome.
   - Use artificial constraints to eliminate rational decisions with other outcomes.
   - Use artificial preferences to prefer the chosen outcome to other optimal outcomes.

Explanations for Complete Preferences

Why not an action with a better outcome?

- An action α dominates an action β iff its outcome is strictly preferred to the outcome of β.
- If the decision α^* has been made, then all dominating actions are infeasible.
- We therefore use the infeasibility of the set of dominators $Dom(\alpha^*)$ as explanation

Why not an action with another optimal outcome?

- As the preorder is complete all outcomes are comparable.
- The outcome ω^* of α^* is the unique optimal outcome.

Explanations with Artificial Constraints

Why not an action with a better outcome?

- The set of dominators $Dom(\alpha^*)$ is infeasible.

Why not an action with another optimal outcome?

- If the decision α^* has been made, then it need to be justified that none of the other rational decisions has been chosen.
- We may add artificial constraints that make exactly he other rational decisions infeasible.
- We use these artificial constraint as explanation.

Explanations with Artificial Preferences

Why not an action with a better outcome?

- The set of dominators $Dom(\alpha^*)$ is infeasible.

Why not an action with another optimal outcome?

- If the decision α^* has been made and its outcome is ω^*, then all incomparable outcomes should be less preferred.
- We define an extension $\succ^*$ of the strict preference order $\succ$
  - if ω is incomparable to ω^* then $\omega^* \succ^* \omega$
  - if $\omega_1 \succ \omega_2$ then $\omega_1 \succ^* \omega_2$
- We use this extension of the preferences as explanation.

Explanation of Optimality: Summary

Why this decision?

- All better decisions are infeasible.
- All other rational decisions are eliminated by artificial constraints or by artificial preferences.

Which is the crucial information in this explanation?

- **Optimal outcome:** the outcome of the rational decision.
- **Preferences:** the extended preference order.
- **Constraints:** the infeasibility of the strictly better decisions and the artificial constraints.

Benefits of Explanations

Acceptance of decisions

- The explanation unveils the preferences and constraints that make that the decision is the best one.
- If a rational stakeholder accepts these preferences then she will accept the decision.

Critique of decision

- Critique of artificial preferences: add more preferences.
- Critique of the feasibility of the decision: add more constraints.
- Critique of the exclusiveness of the decision: add more actions.

Decision Making as Combinatorial Optimization

Combinatorial action space

- Actions are combinations of multiple local options.
- Each local option is chosen from a domain.
- Background constraints describe the legal combinations.
- Feasibility constraints describe which actions are feasible in a given situation.
- (Additive) criterion maps combinatorial actions to numerical outcomes s.t. greater values are preferred.

Question

Which of the constraints are making a decision rational?

Example: Resource Allocation and Scheduling

Variables for local options

- Assign tasks to workers.
- Assign starting times to tasks.

Constraints for feasibility

- A worker cannot do two tasks at the same time.
- If a task precedes another one the second task can only start after the first one has finished.

Expression for criterion

- For example the sum of lateness of all tasks.

Combinatorial Problem

Problem space: $X_1 \times \ldots \times X_n$

For example, for each task $i = 1, \ldots, t$ we introduce

- set X_i of project members who can do task i;
- set X_{t+i} of time periods for performing task i;

Combinatorial Decision Space

Constraints: $C \subseteq X_{i_1} \times \ldots \times X_{i_k}$

Local constraints of small scope $\{i_1, \ldots, i_k\}$, e.g.

- precedence constraint between tasks i, j, $x_{t+i} < x_{t+j}$.
- resource constraint for each project member:

$$\text{if } x_i = x_j \text{ then } x_{t+i} < x_{t+j} \vee x_{t+j} < x_{t+i}$$

where $x \in X_1 \times \ldots \times X_n$

Decision space: $D \subseteq X_1 \times \ldots \times X_n$

...such that $x \in D$ iff $(x_{i_1}, \ldots, x_{i_k}) \in C$ for all constraints with scope $\{i_1, \ldots, i_k\}$.

Combinatorial Outcome Space

Outcome space: $\Omega_1 \times \ldots \times \Omega_m$

Cartesian product of all outcomes.

Criteria: $z_j : X_{j_1} \times \ldots \times X_{j_{k_j}} \rightarrow \Omega_j$

Global criteria of large scope $\{j_1, \ldots, j_{k_j}\}$:

- delivery time
- extra hours

Local criteria of small scope $\{j_1, \ldots, j_{k_j}\}$:

- task of project member l in period p for each l, p

Incomplete & Local Preferences

Preferences are viewpoint specific

Each viewpoint is defined by one or more criteria

- Marketing: prefer earlier delivery dates all else ignored
- Administration: prefer less extra-hours all else ignored
- Project member i: prefer task A over B all else ignored

rationality principles are restricted to viewpoints!

Preferences may be incomplete

Project member i prefers task A over B, but has no opinion about C.

Incomplete & Local Preferences

Preferences are viewpoint specific

Each viewpoint is defined by one or more criteria

- Marketing: prefer earlier delivery dates all else ignored
- Administration: prefer less extra-hours all else ignored
- Project member i: prefer task A over B all else ignored

rationality principles are restricted to viewpoints!

Preferences may be incomplete

Project member i prefers task A over B, but has no opinion about C.

Questions about Preferences

Modelling

How to aggregate viewpoint-specific preferences?

Which preference models can do this?

Solving

How to solve combinatorial problems under those preferences?

Explaining

How to explain the results while allowing user critics?

Can we use the original user preferences for this?

Why is there no action with a better outcome?

Why are all the dominators of the decision infeasible?

- If α^* is the decision and ω^* its outcome then all actions with strictly greater outcome are infeasible.
- We want to describe this set of actions in a most general form by using the original constraints.
- We want to find a minimal subset of the original constraints.

Characterizing the dominators

- β is a dominator of α^* iff its outcome strictly preferred to ω^*
- The dominators are exactly the solutions of $z(\vec{x}) > \omega^*$
- As the dominators are infeasible, no solution of the constraints satisfies $z(\vec{x}) > \omega^*$

Atomic Optimization Problems (I)

Preference model:

- single criterion $z : \mathcal{X} \rightarrow \Omega$
- total order $\geq$ on Ω

Problem

$$Max_{z,>}(D) := \{x \in D \mid \nexists x^* \in D : z(x^*) > z(x)\}$$

Classic combinatorial optimization

- represent order by utility u s.t. $\omega_1 \geq \omega_2$ iff $u(\omega_1) \geq u(\omega 2)$
- solve $max\{u(z(x)) \mid x \in D\}$ and let x^* be a solution
- can be solved by existing optimizers

Atomic Optimization Problems (I)

Preference model:

- single criterion $z : \mathcal{X} \rightarrow \Omega$
- total order $\geq$ on Ω

Problem

$$Max_{z,>}(D) := \{x \in D \mid \nexists x^* \in D : z(x^*) > z(x)\}$$

Classic combinatorial optimization

- represent order by utility u s.t. $\omega_1 \geq \omega_2$ iff $u(\omega_1) \geq u(\omega 2)$
- solve $max\{u(z(x)) \mid x \in D\}$ and let x^* be a solution
- can be solved by existing optimizers

Atomic Optimization Problems (I)

Preference model:

- single criterion $z : \mathcal{X} \rightarrow \Omega$
- total order $\geq$ on Ω

Problem

$$Max_{z,>}(D) := \{x \in D \mid \nexists x^* \in D : z(x^*) > z(x)\}$$

Classic combinatorial optimization

- represent order by utility u s.t. $\omega_1 \geq \omega_2$ iff $u(\omega_1) \geq u(\omega 2)$
- solve $max\{u(z(x)) \mid x \in D\}$ and let x^* be a solution
- can be solved by existing optimizers

Atomic Optimization Problems (II)

Solved form

- let ω^* be the value $z(x^*)$ of z in solution x^*
- then $Max_{z,>}(D) = \{x \mid x \in D \wedge z(x) = \omega^*\}$

Explanation of Optimality

Optimization proble:m

Let ω^* be the optimal value of z under constraints $\mathcal{C}$.

Explanation questions

- Why is ω^* optimal?
- Why isn't ω chosen instead?

Explanation of optimality: $(>, \omega^*, E)$

where E is a simplest subproblem (minimal subset) of $\mathcal{C}$ s.t. ω^* is the optimal value of z under E

- ω^* is optimal as E defeats all better values
- ω is not chosen since $\omega^* > \omega$ or ω is defeated by E

Explanation of Optimality

Optimization proble:m

Let ω^* be the optimal value of z under constraints $\mathcal{C}$.

Explanation questions

- Why is ω^* optimal?
- Why isn't ω chosen instead?

Explanation of optimality: $(>, \omega^*, E)$

where E is a simplest subproblem (minimal subset) of $\mathcal{C}$ s.t. ω^* is the optimal value of z under E

- ω^* is optimal as E defeats all better values
- ω is not chosen since $\omega^* > \omega$ or ω is defeated by E

Explanation of Optimality

Optimization proble:m

Let ω^* be the optimal value of z under constraints $\mathcal{C}$.

Explanation questions

- Why is ω^* optimal?
- Why isn't ω chosen instead?

Explanation of optimality: $(>, \omega^*, E)$

where E is a simplest subproblem (minimal subset) of $\mathcal{C}$ s.t. ω^* is the optimal value of z under E

- ω^* is optimal as E defeats all better values
- ω is not chosen since $\omega^* > \omega$ or ω is defeated by E

How to compute explanations?

Reduce to conflicts

- find a minimal unsatisfiable subset E' ("conflict") of $\mathcal{C} \cup \{z > \omega^*\}$
- $(>, \omega^*, E' \setminus \{z > \omega^*\})$ is an explanation of optimality

How to compute conflicts?

Perform a sequence of satisfiability checks – QUICKXPLAIN

How to compute explanations?

Reduce to conflicts

- find a minimal unsatisfiable subset E' ("conflict") of $\mathcal{C} \cup \{z > \omega^*\}$
- $(>, \omega^*, E' \setminus \{z > \omega^*\})$ is an explanation of optimality

How to compute conflicts?

Perform a sequence of satisfiability checks – QUICKXPLAIN

Explaining the Optimality of Solutions

Explanation of optimality

- A minimal subset X of the original constraints such that no solution of X satisfies $z(\vec{x}) > \omega^*$.

Reduction to Conflict between Constraints

- An explanation of optimality is a minimal subset X of the original constraints C such that there is no solution of the constraints X and $z(\vec{x}) > \omega^*$.
- Hence, explanations of optimality are obtained as conflicts if the original constraints are moved to the foreground and the dominator constraint is moved into the background.

Optimize and Explain

Optimize

- Constraints: C
- Objective: maximize $z(\vec{x})$
- Method: some optimizer
- Result: optimum ω^*

Explain

- Background: $z(\vec{x}) > \omega^*$
- Foreground: C
- Method: e.g. QuickXplain
- Result: conflict X

Explanation of optimality

The optimum ω^*, the preference order $>$, and the conflict X.

Incomplete Preference Orders

Assumption.

- the decision maker has given only some preferences
- hence, the complete preference relation is a superset of the given preferences
- the given preferences define a space of possible complete preference relations

Preference model

- Single criterion $z : \mathcal{X} \rightarrow \Omega$
- space of complete orders on Ω that are supersets of a given (Partial) preorder $\succsim$ on Ω

Incomplete Preference Orders

Assumption.

- the decision maker has given only some preferences
- hence, the complete preference relation is a superset of the given preferences
- the given preferences define a space of possible complete preference relations

Preference model

- Single criterion $z : \mathcal{X} \rightarrow \Omega$
- space of complete orders on Ω that are supersets of a given (Partial) preorder $\succsim$ on Ω

Alternative Optimizations

Problem

- let $\tau(\succ)$ the set of complete extensions of $\succ$
- $Max_{z,\succ}(D) := \bigcup_{>\in\tau(\succ)} Max_{z,>}(D)$

Optimization under Partial Orders

$$Max_{z,\succ}(D) = \{x \in D \mid \nexists x^* \in D : z(x^*) \succ z(x)\}$$

Solved form

- let Ω^* be the optima to be found by the optimizer
- $Max_{z,\succ}(D) = \{x \mid x \in D \wedge \bigvee_{\omega^*\in\Omega^*} z(x) = \omega^*\}$

Explanations under Partial Orders

Approach

- chooses a linear extension $>$ of $\succ$ and
- generates the optima $\omega_1^*, \ldots, \omega_k^*$ in decreasing $>$-order

Explanations under Partial Orders

Explanation with dominance constraints

- each ω_i^* has an explanation of optimality $(>, \omega_i^*, E_i)$
- but E_i contains dominance constraints $z(x) \not\preceq \omega_j^*$

Explanation without dominance constraints

- define extension $\succ_i$ of $\succ$ s.t. $\omega_i^* \succ_i \omega_j^*$ for $j \neq i$
- choose a linear extension $>_i$ of $\succ_i$
- find new explanation $(>_i, \omega_i, E_i')$, namely for the optimality of ω_i^* w.r.t. to $>_i$

Decision Making as Sequential Optimization

Combinatorial outcome space

- An action can be evaluated via multiple viewpoints.
- Each viewpoint determines an outcome for the action.
- Each viewpoint defines preferences on its outcome.
- There is a strict importance ordering among the viewpoints.
- Global outcomes are obtained as combinations of viewpoint-specific outcomes.

Question

Which of the viewpoint-specific constraints/preferences are making a decision rational?

Lexicographic Optimization

Preference Structure

- Multiple criteria z_i map the actions to outcomes from Ω_i.
- A preference order $\succsim_i$ for each Ω_i.
- The criteria $z_1, \ldots, z_m$ are ordered in increasing importance.

Aggregation

- We define the lexicographical preference order $\succ_{lex}$ on $\Omega_1 \times \ldots \times \Omega_m$.
- A vector $(\omega_1^*, \ldots, \omega_m^*)$ is lexicographically preferred to $(\omega_1, \ldots, \omega_m)$ iff ω_k^* is preferred to ω_k for the smallest index k for which the two vectors differ.

Lexicographic Optimize and Explain

Optimize

1. Compute optimum ω_1^* of $z(\vec{x_1})$ under $\succ_1$ and C.
2. Compute optimum ω_2^* of $z(\vec{x_2})$ under $\succ_2$ and $C, z(\vec{x_1}) = \omega_1^*$.
3. Compute optimum ω_3^* of $z(\vec{x_3})$ under $\succ_3$ and $C, z(\vec{x_1}) = \omega_1^*, z(\vec{x_2}) = \omega_2^*$ and so on.

Explain

- Sequence $(\xi_1, \ldots, \xi_n)$ of explanations $\xi_i = (>_i, \omega_i^*, E_i)$.
- E_i may contain constraints $z_j(x) = \omega_j^*$ for $j < i$.
- This indicates that the optimal values of more important criteria defeated the values of z_i that are better than ω_i^*.

Explanation Networks

Graphical representation of lexicographical explanations

- Each $\xi_i = (>_i, \omega_i^*, E_i)$ is represented by a node.
- An edge is drawn from ξ_i to ξ_j if the defeaters E_j of z_j contain a constraint of the form $z_i(\vec{x}) = \omega_i$.

Example

see Preference-Based Problem Solving for Constraint Programming in Recent Advances in Constraints, LNCS 5129 Springer 2008.

Unconstrained Importance Orders

Optimize

- Choose a permutation π of the indices $1, \ldots, m$.
- Compute lexicographical optimum $(\omega^*_{\pi_1} \ldots, \omega^*_{\pi_m})$ for $z_{\pi_1}, \ldots, z_{\pi_m}$.

Explain

- Take an explanation $(\xi_{\pi_1} \ldots, \xi_{\pi_m})$ of the lex-optimality of $(\omega^*_{\pi_1} \ldots, \omega^*_{\pi_m})$.
- It lists the criteria in the chosen order of importance.

Decision Making as Multi-Objective Optimization

Combinatorial outcome space

- An action can be evaluated via multiple viewpoints.
- Each viewpoint determines an outcome for the action.
- Each viewpoint defines a preferences on its outcome.
- Certain viewpoints may have the same importance.
- Global outcomes are obtained as combinations of viewpoint-specific outcomes.

Question

Which are the view-point specific constraints and preferences are making a decision rational?

Pareto-Optimization

Preference Structure

- Multiple criteria z_i map the actions to outcomes from Ω_i.
- A preference order $\succsim_i$ for each Ω_i.
- The criteria $z_1, \ldots, z_m$ have all the same importance.

Aggregation

- We define the weak Pareto-dominance order $\succsim_{Pareto}$ on $\Omega_1 \times \ldots \times \Omega_m$.
- A vector $(\omega_1^*, \ldots, \omega_m^*)$ weakly dominates $(\omega_1, \ldots, \omega_m)$ iff ω_i^* is weakly preferred to ω_i for all i.

Pareto-Optimize and Explain

Optimize

- Outer branching uses the lexicographic order as linear extension of the strict Pareto-dominance.
- Dominance constraints $z(\vec{x}) \not\preceq_{pareto} \vec{\omega}$ have the form $\bigvee_{j=1}^{n} z_j(\vec{x}) >_j \omega_j$.

Explanations with Artificial Constraints

- Artificial preferences: lexicographical order as before.
- Dominance constraints reduce to $z_j(\vec{x}) >_j \omega_j$.
- These artificial constraints can be viewed as penalization limits, which gives a clearer explanation than an arbitrary artificial preference order over a combinatorial space.

Lessons learned

- Combinatorial structure impacts the form of explanations.

Open questions

- Does uncertainty impact the form of explanations?
- There is a relationship between reasoning about uncertainty and quantified constraint satisfaction [3]
- Which kind of explanations do we obtain for CP-nets?

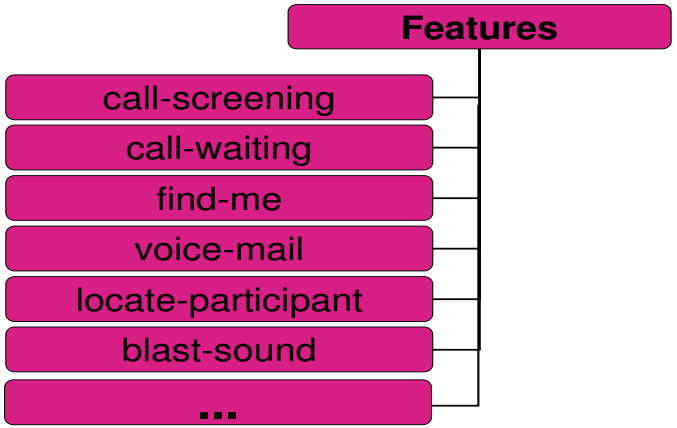
Uncertainty and Explanation

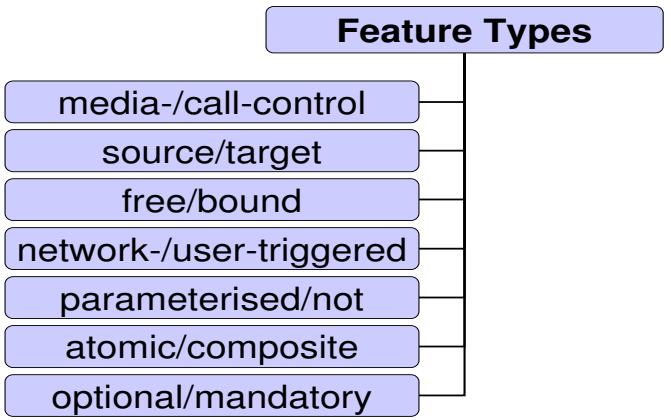
- An action may have uncertain outcomes modeled by a probability distribution.
- Actions are then compared by their expected utility.
- Preferences on outcomes can thus be transformed into preferences on actions.
- We can produce explanations in terms of action preferences.
- Question: should explanations unveil also the probabilities and the outcome preferences?

Outline

Call Control Features [8, 9]

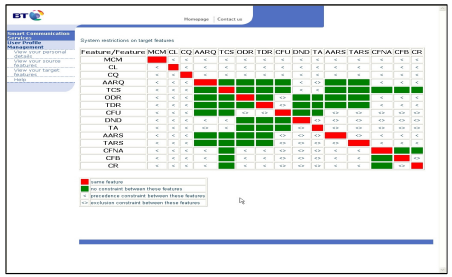
- **Communication services are pervasive and disruptive**
  - personalisation solutions to control and enrich services are a must
- **Call control features**
  - increments of the basic call service
  - primitive service configuration options for subscribers
  - 10s of features available





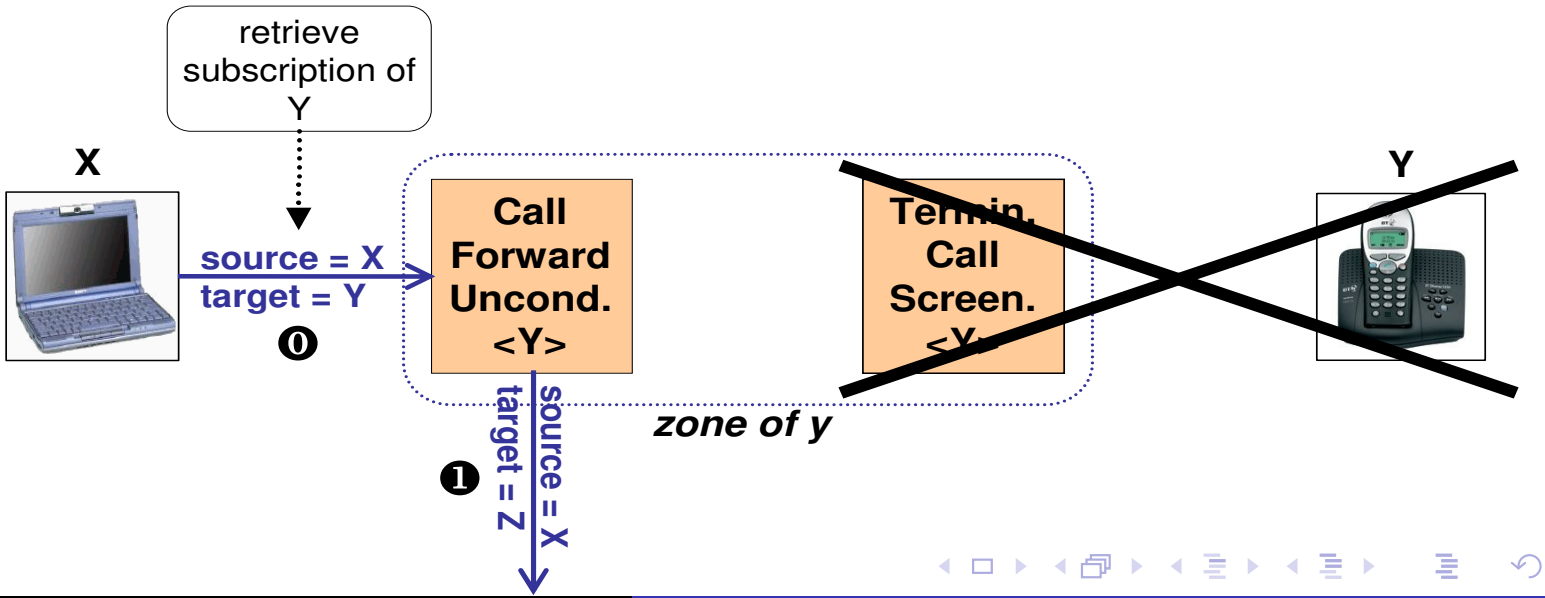
Feature-Based Personalisation

- **Service providers** implement and expose features in catalogues
- **Service subscribers** pick n features from catalogues to configure their subscriptions
- **Communication sessions** are set up based on the subscriptions of participants



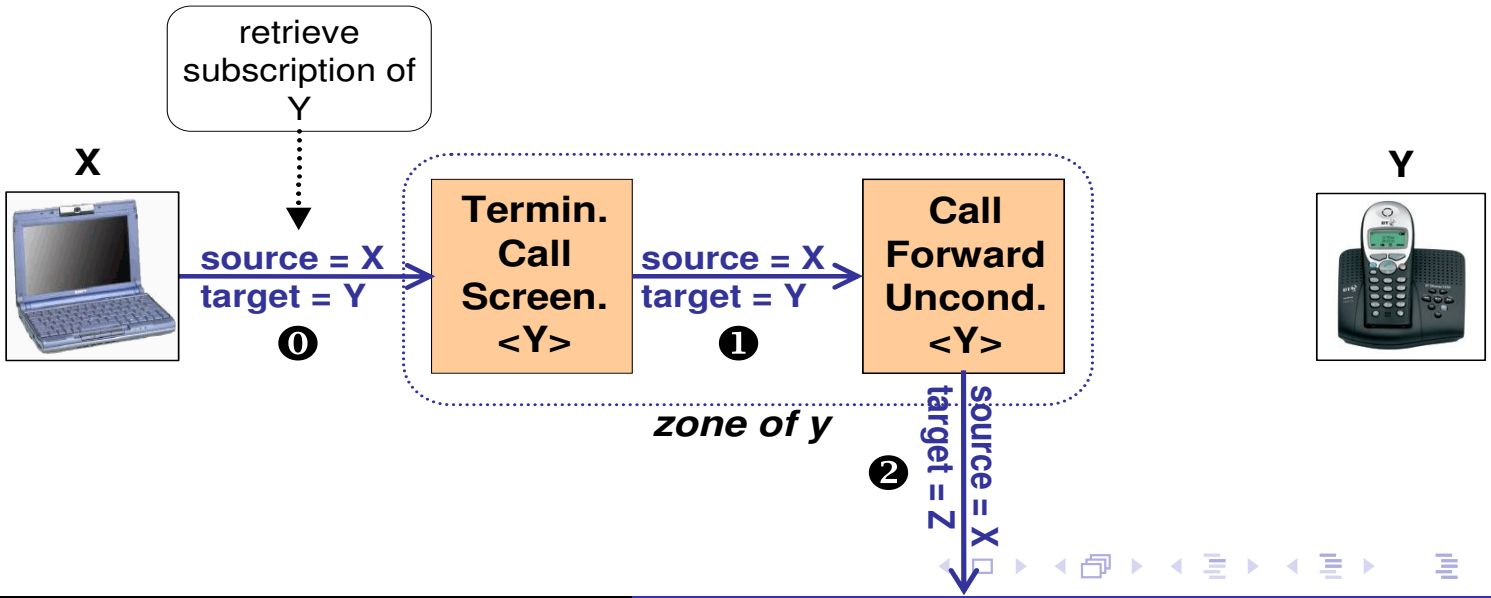
Feature Interactions

- "Some way in which a feature modifies or influences the behaviour of another feature in generating the system′s overall behaviour"
- Undesirable interactions must be detected and avoided when users configure their subscriptions



Feature Interactions

- "Some way in which a feature modifies or influences the behaviour of another feature in generating the system′s overall behaviour"
- Undesirable interactions must be detected and avoided when users configure their subscriptions

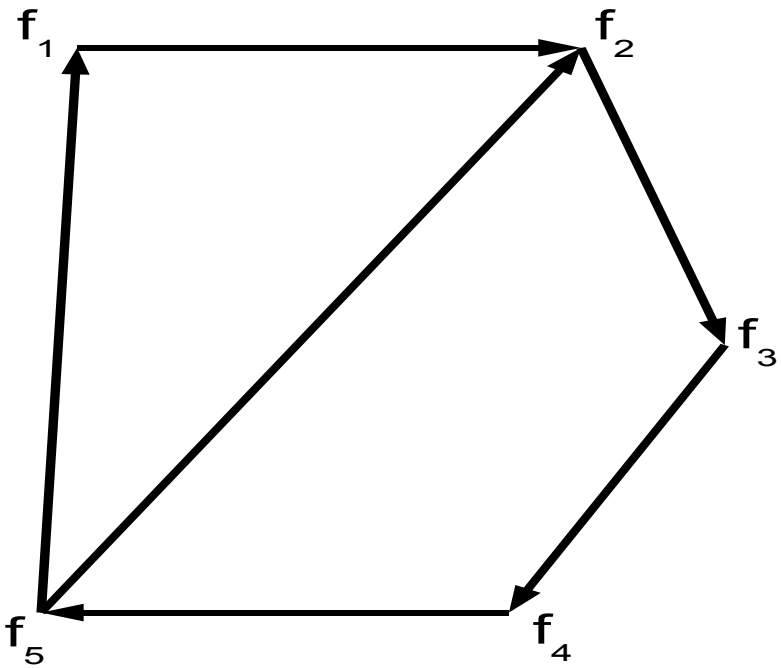


Configuring Subscriptions

- **Subscribers** select features, and specify precedence constraints between features
- **Configuration engine**
  - **Verification.** Checking the consistency of a subscription is linear with respect to the number of features and precedence constraints.
  - **Partial Completion.** Computing transitive closure is cubic with respect to the number of features.
  - **Completion.** Ordering a subscription is linear with respect to the number of features and precedence constraints.
  - **Filtering.** Finding incompatible features is cubic with respect to the number of features.
  - **Revision.** Finding an optimal relaxation is NP-hard since it is a generalization of the feedback vertex problem.

Catalogue Graph

- A **catalogue** is a pair $\langle \mathcal{F}, \mathcal{H} \rangle$ of features and precedence constraints. It can be seen as a *directed graph*.
- Example, $\mathcal{F} = \{f_1, \ldots, f_5\}$, $\mathcal{H} = \{\langle f_1, f_2 \rangle, \langle f_2, f_3 \rangle, \langle f_3, f_4 \rangle, \langle f_4, f_5 \rangle, \langle f_5, f_1 \rangle, \langle f_5, f_2 \rangle\}$.

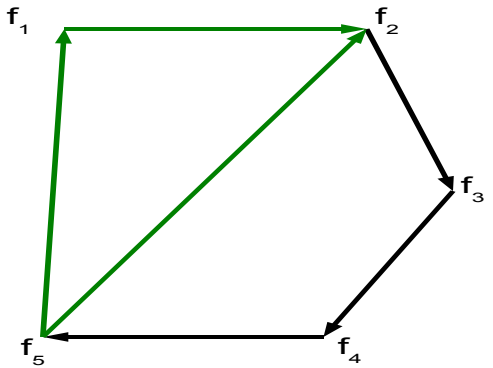


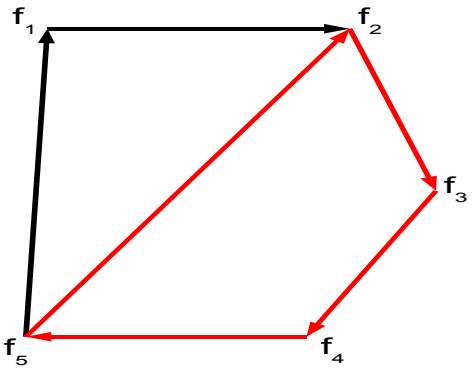
Subscription

- A feature subscription S of catalogue $\langle \mathcal{F}, \mathcal{H} \rangle$ is a tuple $\langle F, H, P, W_F, W_P \rangle$, where
  - $F \subseteq \mathcal{F}$,
  - H is the projection of $\mathcal{H}$ on F,
  - P is a set of (user defined) precedence constraints on F,
  - $W_F : F \rightarrow \mathbb{N}$ is a function that assigns weights to features and
  - $W_P : P \rightarrow \mathbb{N}$ is a function that assigns weights to user precedence constraints.
- The value of S is defined by $\text{Value}(S) = \sum_{f \in F} W_F(f) + \sum_{\prec_{ij} \in P} W_P(\prec_{ij})$.

Subscription

- A feature subscription $\langle F, H, P, W_F, W_P \rangle$ is consistent if and only if the directed graph $\langle F, H \cup P \rangle$ is acyclic.
- $F = \{f_1, f_2, f_5\}$ and $P = \emptyset$ induces a consistent subscription.
- $F = \{f_2, f_3, f_4, f_5\}$ and $P = \emptyset$ induces an inconsistent subscription.

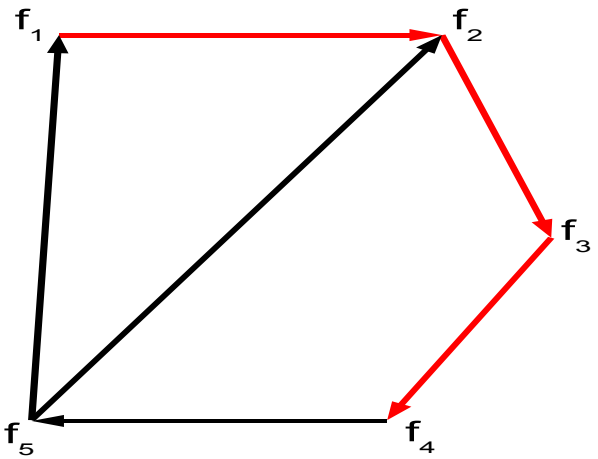




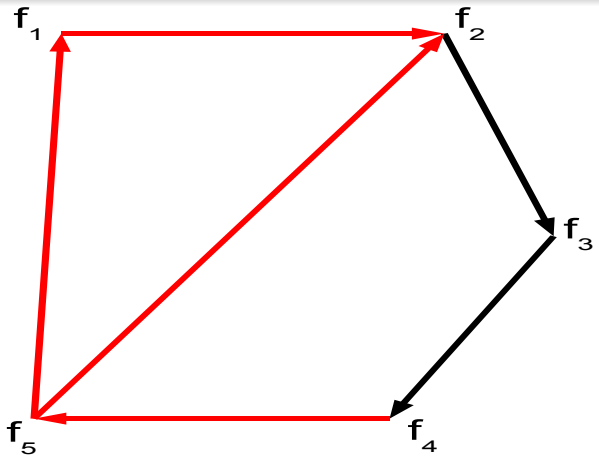
Relaxation

- A **relaxation** of a subscription $\langle F, H, P, W_F, W_P \rangle$ is a subscription $\langle F', H', P', W'_F, W'_P \rangle$ such that
  - $F' \subseteq F$,
  - $H' = H{\downarrow_{F'}}$,
  - $P' \subseteq P{\downarrow_{F'}}$.
  - $W_{F'}$ is W_F restricted to F', and
  - $W_{P'}$ is W_P restricted to P'.
- Let S be an inconsistent feature subscription and R_S be the set of all consistent relaxations of a feature subscription S. We say that $S_i \in R_S$ is an **optimal relaxation** of S if it has maximum value among all relaxations.

Relaxation



- A maximal relaxation S_5 induced by $F \setminus \{f_5\}$
- $\text{Value}(S_5) = 1+2+3+4 = 10$

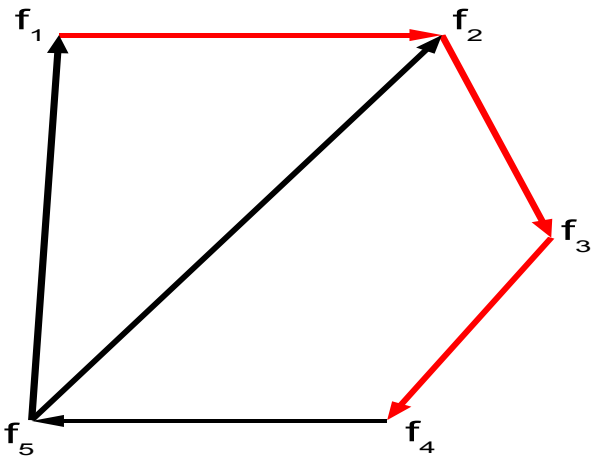


- A maximal relaxation S_3 induced by $F \setminus \{f_3\}$
- $\text{Value}(S_3) = 1+2+4+5 = 12$

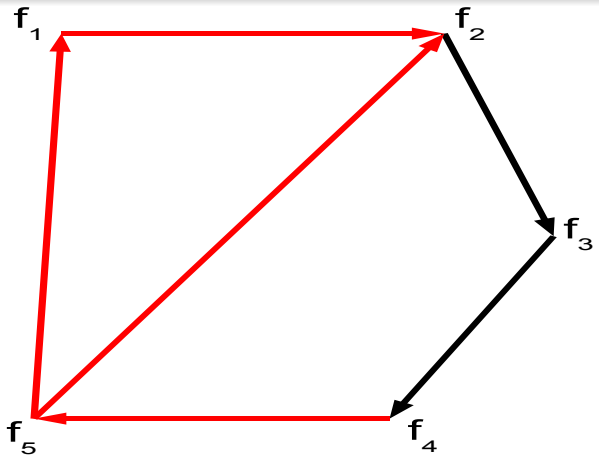
Complexity

Finding an optimal relaxation is NP-hard!

Relaxation



- A maximal relaxation S_5 induced by $F \setminus \{f_5\}$
- $\text{Value}(S_5) = 1+2+3+4 = 10$

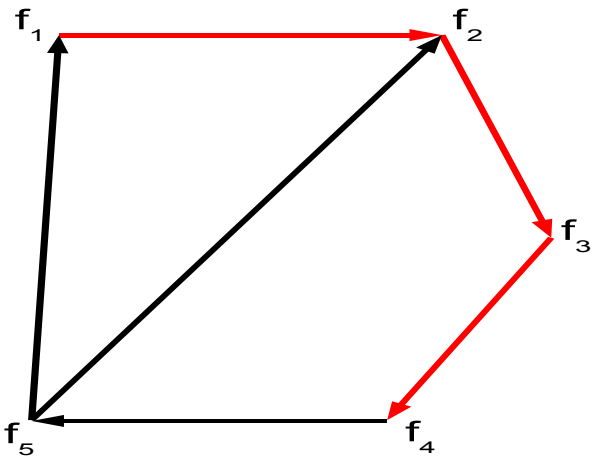


- A maximal relaxation S_3 induced by $F \setminus \{f_3\}$
- $\text{Value}(S_3) = 1+2+4+5 = 12$

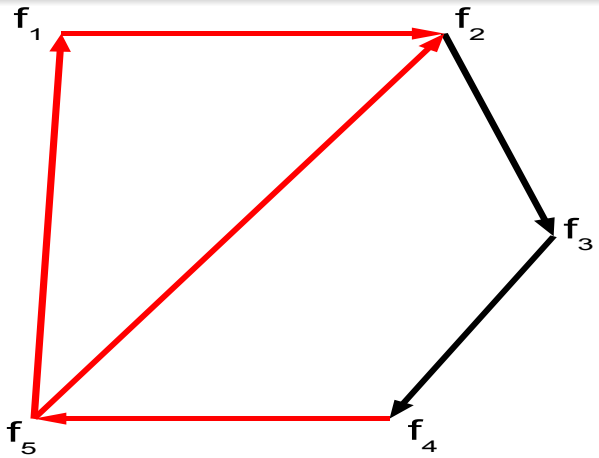
Complexity

Finding an optimal relaxation is NP-hard!

Relaxation



- A maximal relaxation S_5 induced by $F \setminus \{f_5\}$
- $\text{Value}(S_5) = 1+2+3+4 = 10$



- A maximal relaxation S_3 induced by $F \setminus \{f_3\}$
- $\text{Value}(S_3) = 1+2+4+5 = 12$

Complexity

Finding an optimal relaxation is NP-hard!

Different Approaches

- Constraint Programming
- Partial Weighted Maximum Satisfiability
- Integer Linear Programming

Modeling the problem as a COP(I)

Table: Variables and Domains

Variable	Type	Domain	Purpose
s_{f_i}	Boolean	0/1	inclusion/exclusion of a feature $f_i \in F$
p_{f_i}	Integer	$1 \ldots \lvert F \rvert$	position of a feature $f_i \in F$
$s_{p_{ij}}$	Boolean	0/1	inclusion/exclusion of a user precedence $p_{ij} \in P$

Modeling the problem as a COP(II)

- **Constraints**
  - Precedence constraints in catalogue

$$s_{f_i} \wedge s_{f_j} \rightarrow (p_{f_i} < p_{f_j})$$

  - Precedence constraints defined by the user (Preference)

$$s_{p_{ij}} \rightarrow (s_{f_i} \wedge s_{f_j} \wedge (p_{f_i} < p_{f_j}))$$

- **Objective Function.** The objective is to maximize:

$$\sum_{f_i \in F} s_{f_i} \times wf_i + \sum_{p_{ij} \in P} s_{p_{ij}} \times wp_{ij}$$

Consistency Techniques

- **Arc Consistency** (AC)
- **Mixed Consistency:** different levels of consistency on different sets of variables of a given problem.
  - *Singleton Arc Consistency* (SAC_b) on the Boolean variables and Arc Consistency on the remaining variables.
  - *Restricted Singleton Arc Consistency* (RSAC_b) on the Boolean variables and Arc Consistency on the remaining variables.

Partial Weighted MaxSAT

Boolean satisfiability (SAT) is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE.

$$\begin{array}{lllll} (p & \vee & \neg q & \vee & r) \quad \wedge \\ (q & \vee & w & \vee & s) \quad \wedge \\ (r & \vee & t & \vee & \neg q) \end{array}$$

Partial Weighted Maximum Boolean Satisfiability extends SAT by including the notions of hard and soft clauses. The goal is to find an assignment that maximizes the value.

$$\begin{array}{llllll} \langle \top, & (p & \vee & \neg q & \vee & r)\rangle \quad \wedge \\ \langle w_i, & (q & \vee & w & \vee & s)\rangle \quad \wedge \\ \langle w_j, & (r & \vee & t & \vee & \neg q)\rangle \end{array}$$

Partial Weighted MaxSAT: Model (I)

- Precedence constraints in the catalogue:

$$\frac{p_{ij} \in H}{\langle \top, (\neg s_{f_i} \vee \neg s_{f_j} \vee s_{p_{ij}}) \rangle \in SatInst}$$

- The precedence relation is transitive:

$$\frac{\{p_{ij}, p_{jk}\} \subseteq H \cup P}{\langle \top, (\neg s_{p_{ij}} \vee \neg s_{p_{jk}} \vee s_{p_{ik}}) \rangle \in SatInst}$$

- The precedence relation is antisymmetric:

$$\frac{p_{ij} \in H \cup P}{\langle \top, (\neg s_{p_{ij}} \vee \neg s_{p_{ji}}) \rangle \in SatInst}$$

Partial Weighted MaxSAT: Model (II)

- Each feature is associated with its weight:

$$\frac{f_i \in F}{\langle w_{f_i}, (s_{f_i}) \rangle \in SatInst}$$

- Each user precedence relation is associated with its weight:

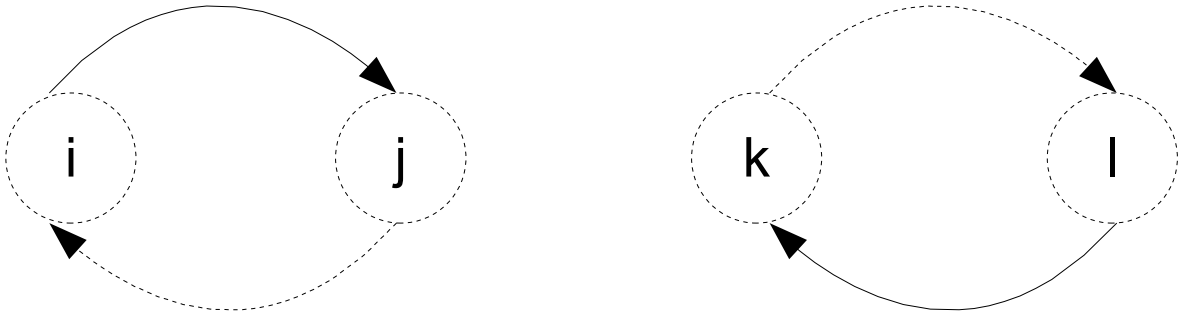
$$\frac{p_{ij} \in P}{\langle w_{p_{ij}}, (s_{p_{ij}}) \rangle \in SatInst}$$

- A user precedence relation is only satisfied if its features are included:

$$\frac{p_{ij} \in P}{\langle \top, (s_{f_i} \vee \neg s_{p_{ij}}) \rangle \in SatInst} \quad \frac{p_{ij} \in P}{\langle \top, (s_{f_j} \vee \neg s_{p_{ij}}) \rangle \in SatInst}$$

Reducing the number of variables and clauses

- The scope of the precedence constraint variables can be restricted to the transitive closures of $H \cup P$ since two features that are unrelated under $H \cup P$ can appear in any order.



The order between j and k is irrelevant to any optimal relaxation of this inconsistent subscription

- The clause whose consequence is already enforced by the catalogue, i.e., $\{p_{ji}, p_{kj}, p_{ik}\} \cap H \neq \emptyset$, can be avoided.

Integer Linear Programming: Formulation

- Maximize

$$\sum_{f_i \in F} w_{f_i} s_{f_i} + \sum_{p_{ij} \in P} w_{p_{ij}} s_{p_{ij}}$$

- Catalogue Precedence Constraint

$$p_{f_i} - p_{f_j} + n * s_{f_i} + n * s_{f_j} \leq 2n - 1$$

- User Precedence Constraint

$$p_{f_i} - p_{f_j} + n * s_{p_{ij}} \leq n - 1$$

$$s_{p_{ij}} - s_{f_i} \leq 0$$

$$s_{p_{ij}} - s_{f_j} \leq 0$$

Comparison

	AC		$RSAC_b$		SAC_b	
$\langle f, p \rangle$	time	#nodes	time	#nodes	time	#nodes
$\langle 15, 20 \rangle$	92	726	34	41	42	41
$\langle 20, 10 \rangle$	203	1,694	39	47	50	46
$\langle 25, 40 \rangle$	14,985	88,407	595	187	678	169
$\langle 30, 20 \rangle$	6,073	29,211	653	184	768	161
$\langle 35, 35 \rangle$	124,220	481,364	7,431	1,279	8,379	1,093
$\langle 40, 40 \rangle$	1,644,624	5,311,838	67,798	9,838	76,667	8,475

Table: Average results of maintaining AC, $RSAC_b$ and SAC_b.

Comparison

Table: Catalogue $\langle 50, 250, \{<, >\}\rangle$.

	PWMSAT			CPLEX			CP	
$\langle f, p\rangle$	#nodes	time	#us	#nodes	time	#us	#nodes	time
$\langle 15, 20\rangle$	721	1,039	0	51	61	0	41	34
$\langle 20, 10\rangle$	1,295	1,619	0	50	47	0	47	39
$\langle 25, 40\rangle$	5,039	4,391	0	3,482	1,945	0	187	595
$\langle 30, 20\rangle$	5,415	6,397	0	1,901	1,025	0	184	653
$\langle 35, 35\rangle$	30,135	23,955	0	35,247	22,763	0	1,279	7,431
$\langle 40, 40\rangle$	186,913	282,760	0	299,829	247,140	0	9,838	67,798
$\langle 45, 90\rangle$	6,291,957	12,638,251	8	5,280,594	7,690,899	2	104,729	1,115,515

Review of the Case-Study

- We presented, and evaluated, three optimisation-based approaches to finding optimal reconfigurations of call-control features when the user's preferences violate the technical constraints.
- Our results also suggest that the CP approach, when applied with stronger consistency, is able to scale well compared to the other approaches.
- Finding an optimal reconfiguration of a subscription of reasonable size is feasible using CP.
- It's good, in fact crucial, to focus on real-world applications!

Outline

Wrap-up

Where can you get the slides?

Tutorial web-site

`http://www.cs.ucc.ie/~osullb/ijcai-tutorial-2009/`


Jérôme Amilhastre, Hélène Fargier, and Pierre Marquis.
Consistency restoration and explanations in dynamic csps application to configuration.
Artif. Intell., 135(1-2):199–234, 2002.


James Bailey and Peter J. Stuckey.
Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization.
In Manuel V. Hermenegildo and Daniel Cabeza, editors, *PADL*, volume 3350 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2005.


Alex Ferguson and Barry O'Sullivan.
Quantified constraint satisfaction problems: From relaxations to explanations.
In Manuela M. Veloso, editor, *IJCAI*, pages 74–79, 2007.


Peter Funk and Pedro A. González-Calero, editors.
Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004, Proceedings, volume 3155 of *Lecture Notes in Computer Science*. Springer, 2004.


Ulrich Junker.
Quickxplain: Preferred explanations and relaxations for over-constrained problems.
In *AAAI*, pages 167–172, 2004.


Ulrich Junker.
Preference-based problem solving for constraint programming.

In François Fages, Francesca Rossi, and Sylvain Soliman, editors, *CSCLP*, volume 5129 of *Lecture Notes in Computer Science*, pages 109–126. Springer, 2007.

Ulrich Junker and Daniel Mailharro.
Preference programming: Advanced problem solving for configuration.
AI EDAM, 17(1):13–29, 2003.

David Lesaint, Deepak Mehta, Barry O'Sullivan, Luis Quesada, and Nic Wilson.
Personalisation of telecommunications services as combinatorial optimisation.
In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 1693–1698. AAAI Press, 2008.


David Lesaint, Deepak Mehta, Barry O'Sullivan, Luis Quesada, and Nic Wilson.
Solving a telecommunications feature subscription configuration problem.
In Stuckey [14], pages 67–81.


Barry O'Sullivan, Alexandre Papadopoulos, Boi Faltings, and Pearl Pu.
Representative explanations for over-constrained problems.

In *AAAI*, pages 323–328. AAAI Press, 2007.


Alexandre Papadopoulos and Barry O'Sullivan.
Relaxations for compiled over-constrained problems.
In Stuckey [14], pages 433–447.


Carsten Sinz, Albert Haag, Nina Narodytska, Toby Walsh, Esther Gelle, Mihaela Sabin, Ulrich Junker, Barry O'Sullivan, Rick Rabiser, Deepak Dhungana, Paul Grünbacher, Klaus Lehner, Christian Federspiel, and Daniel Naus.
Configuration.
IEEE Intelligent Systems, 22(1):78–90, 2007.


Frode Sormo, Jörg Cassens, and Agnar Aamodt.
Explanation in case-based reasoning-perspectives and goals.
Artif. Intell. Rev., 24(2):109–143, 2005.


Peter J. Stuckey, editor.
Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia,

September 14-18, 2008. Proceedings, volume 5202 of *Lecture Notes in Computer Science*. Springer, 2008.

Computing Explanations in Problem Solving

A Review of Formal Approaches

Barry O'Sullivan[1] Ulrich Junker[2]

[1]**Cork Constraint Computation Centre**
Department of Computer Science, University College Cork, Ireland
b.osullivan@cs.ucc.ie

[2]**ILOG, An IBM Company**
Sophie Antipolis, France
uli.junker@free.fr

IJCAI 2009 Tutorial Programme