Problèmes de Satisfaction de Contraintes

Cyril Terrioux

Laboratoire des Sciences de l'Information et des Systèmes

LSIS - UMR CNRS 6168

Plan

1. Introduction
2. Le formalisme CSP
3. Méthodes de résolution énumératives

Plan

1. Introduction
2. Le formalisme CSP
3. Méthodes de résolution énumératives

Comment modéliser un problème ?

- analyser le problème :
  - identifier les différents objets composant le problème
  - identifier les propriétés de ces objets
  - identifier les relations entre ces objets
- représenter ces informations dans un formalisme donné

Comment modéliser un problème ?

- analyser le problème :
  - identifier les différents objets composant le problème
  - identifier les propriétés de ces objets
  - identifier les relations entre ces objets
- représenter ces informations dans un formalisme donné

Formalisme CSP :

- Objets $\rightarrow$ variables

Comment modéliser un problème ?

- analyser le problème :
  - identifier les différents objets composant le problème
  - identifier les propriétés de ces objets
  - identifier les relations entre ces objets
- représenter ces informations dans un formalisme donné

Formalisme CSP :

- Objets $\rightarrow$ variables
- Propriétés et relations $\rightarrow$ contraintes

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)
- Des contraintes de types très variés

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)
- Des contraintes de types très variés

Exemples de problèmes :

- coloration de graphes,

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)
- Des contraintes de types très variés

Exemples de problèmes :

- coloration de graphes,
- logique propositionnelle,

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)
- Des contraintes de types très variés

Exemples de problèmes :

- coloration de graphes,
- logique propositionnelle,
- vérification de circuits,

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)
- Des contraintes de types très variés

Exemples de problèmes :

- coloration de graphes,
- logique propositionnelle,
- vérification de circuits,
- systèmes d'équations ou d'inéquations,

Un formalisme très puissant

- Domaines quelconques ($\mathbb{N}$, $\mathbb{R}$, ...)
- Des contraintes de types très variés

Exemples de problèmes :

- coloration de graphes,
- logique propositionnelle,
- vérification de circuits,
- systèmes d'équations ou d'inéquations,
- ...

Plan

1. Introduction
2. Le formalisme CSP
3. Méthodes de résolution énumératives

Problème de satisfaction de contraintes (CSP)

Instance $\mathcal{P} = (X, D, C, R)$ avec :

- X un ensemble de n variables,
- D un ensemble de domaines finis de taille au plus d,
- C un ensemble de m contraintes,
- R un ensemble de relations de compatibilité.

Contraintes

Contrainte = un sous-ensemble de variables

Contraintes

Contrainte = un sous-ensemble de variables

Arité = nombre de variables soumises à la contrainte

Contraintes

Contrainte = un sous-ensemble de variables

Arité = nombre de variables soumises à la contrainte

Contrainte unaire = une contrainte portant sur une seule variable

Contraintes

Contrainte = un sous-ensemble de variables

Arité = nombre de variables soumises à la contrainte

Contrainte unaire = une contrainte portant sur une seule variable

Contrainte binaire = une contrainte portant sur deux variables

Contraintes

Contrainte = un sous-ensemble de variables

Arité = nombre de variables soumises à la contrainte

Contrainte unaire = une contrainte portant sur une seule variable

Contrainte binaire = une contrainte portant sur deux variables

Contrainte n-aire = une contrainte portant sur plus de deux variables

Contraintes

CSP binaire = CSP dont toutes les contraintes sont unaires ou binaires

Contraintes

CSP binaire = CSP dont toutes les contraintes sont unaires ou binaires

CSP n-aire = CSP dont au moins une contrainte est n-aire

Contraintes

CSP binaire = CSP dont toutes les contraintes sont unaires ou binaires

CSP n-aire = CSP dont au moins une contrainte est n-aire

Il existe une transformation polynomiale qui permet de passer d'un CSP n-aire à CSP binaire

Contraintes

CSP binaire = CSP dont toutes les contraintes sont unaires ou binaires

CSP n-aire = CSP dont au moins une contrainte est n-aire

Il existe une transformation polynomiale qui permet de passer d'un CSP n-aire à CSP binaire

$\Rightarrow$ on va se focaliser sur les CSP binaires

Contraintes

CSP binaire = CSP dont toutes les contraintes sont unaires ou binaires

CSP n-aire = CSP dont au moins une contrainte est n-aire

Il existe une transformation polynomiale qui permet de passer d'un CSP n-aire à CSP binaire

$\Rightarrow$ on va se focaliser sur les CSP binaires

Attention, le problème reste aussi difficile !

Graphe de contraintes

Une représentation de la structure du problème

Graphe de contraintes

Une représentation de la structure du problème

Graphe de contraintes (X, C) :

- sommets = les variables
- arêtes = les contraintes

Relations

Relation r_i = ensemble des tuples autorisés par la contrainte c_i

Relations

Relation r_i = ensemble des tuples autorisés par la contrainte c_i

Relation r_i = sous-ensemble du produit cartésien $\prod_{x \in c_i} d_x$

Relations

Relation r_i = ensemble des tuples autorisés par la contrainte c_i

Relation r_i = sous-ensemble du produit cartésien $\prod_{x \in c_i} d_x$

Relation en extension : liste de tous les tuples interdits ou autorisés

Relations

Relation r_i = ensemble des tuples autorisés par la contrainte c_i

Relation r_i = sous-ensemble du produit cartésien $\prod_{x \in c_i} d_x$

Relation en extension : liste de tous les tuples interdits ou autorisés

Relation en intention : équations, inéquations, . . .

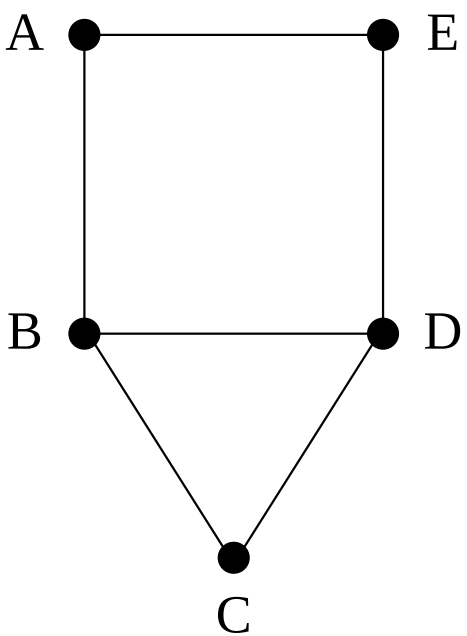
Le pouvoir d'expression des contraintes

Diversité de la nature des contraintes :

- contraintes impliquant un nombre quelconque de variables,
- énumérations des affectations compatibles ou interdites,
- équations, inéquations,
- prédicats,
- . . .

Exemple 1

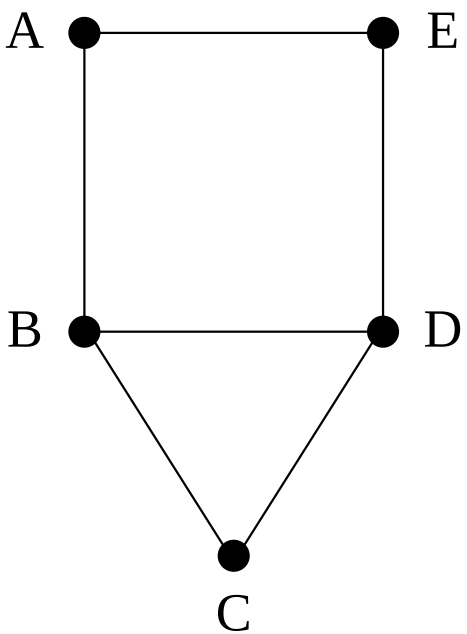
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



Exemple 1

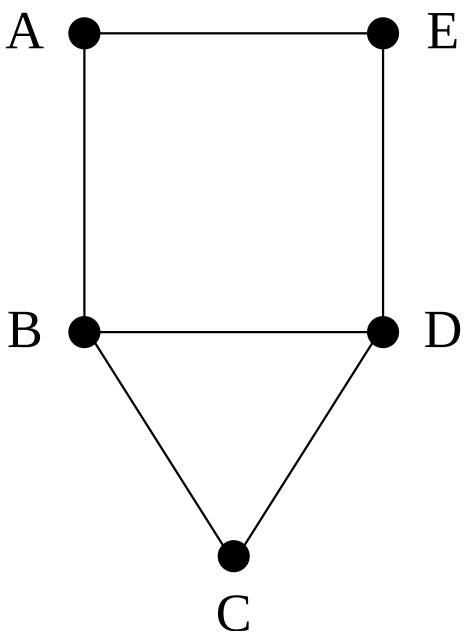
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)

- $X = \{A, B, C, D, E\}$



Exemple 1

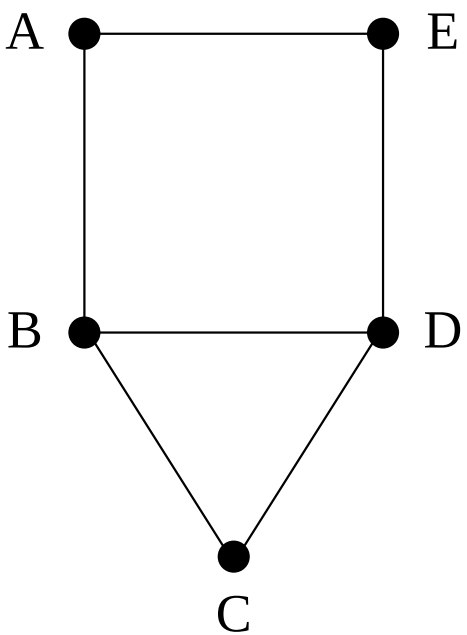
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



- $X = \{A, B, C, D, E\}$
- $D = \{d_A, d_B, d_C, d_D, d_E\}$ avec
  $d_A = d_B = d_C = d_D = d_E = \{v, j, r\}$

Exemple 1

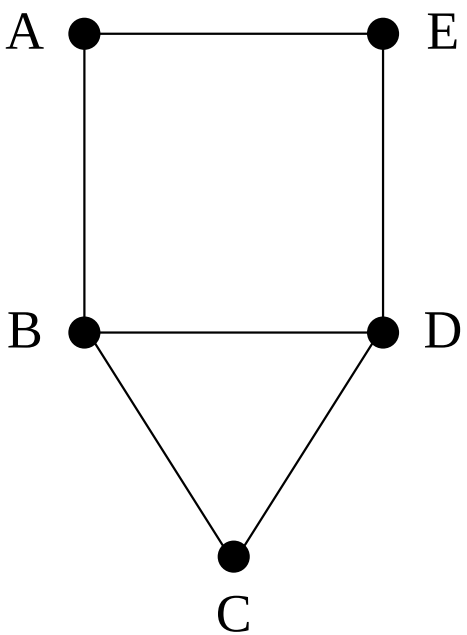
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



- $X = \{A, B, C, D, E\}$
- $D = \{d_A, d_B, d_C, d_D, d_E\}$ avec
  $d_A = d_B = d_C = d_D = d_E = \{v, j, r\}$
- $C = \{c_{AB}, c_{BC}, c_{BD}, c_{CD}, c_{DE}, c_{AE}\}$
  avec $c_{AB} = \{A, B\}$, $c_{BC} = \{B, C\}$, ...

Exemple 1

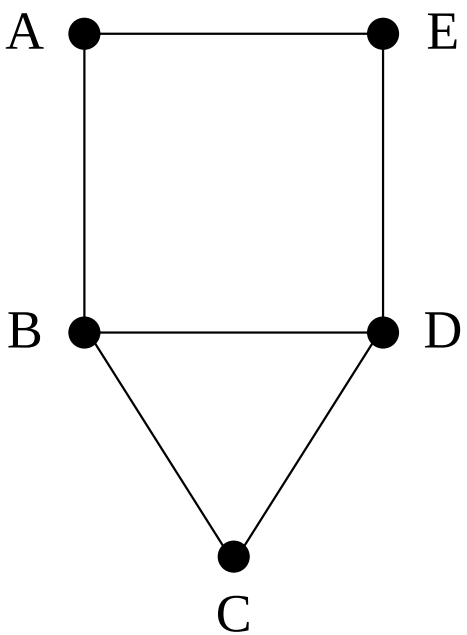
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



- $X = \{A, B, C, D, E\}$
- $D = \{d_A, d_B, d_C, d_D, d_E\}$ avec
  $d_A = d_B = d_C = d_D = d_E = \{v, j, r\}$
- $C = \{c_{AB}, c_{BC}, c_{BD}, c_{CD}, c_{DE}, c_{AE}\}$
  avec $c_{AB} = \{A, B\}$, $c_{BC} = \{B, C\}$, ...
- $R = \{r_{AB}, r_{BC}, r_{BD}, r_{CD}, r_{DE}, r_{AE}\}$
  avec $r_{AB} = \{(a, b) | a \in d_A, b \in d_B, a \neq b\}$,

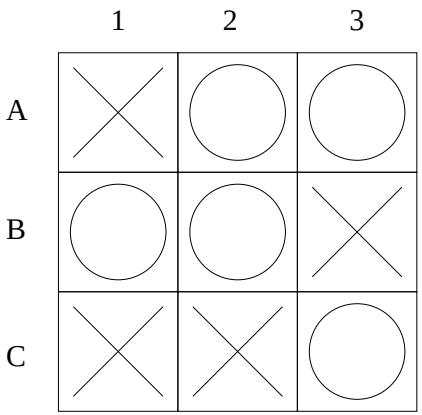
Exemple 1

Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



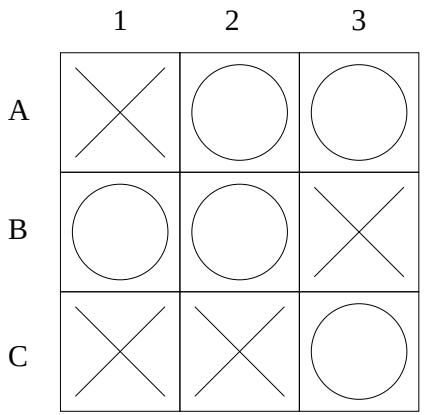
- $X = \{A, B, C, D, E\}$
- $D = \{d_A, d_B, d_C, d_D, d_E\}$ avec
  $d_A = d_B = d_C = d_D = d_E = \{v, j, r\}$
- $C = \{c_{AB}, c_{BC}, c_{BD}, c_{CD}, c_{DE}, c_{AE}\}$
  avec $c_{AB} = \{A, B\}$, $c_{BC} = \{B, C\}$, ...
- $R = \{r_{AB}, r_{BC}, r_{BD}, r_{CD}, r_{DE}, r_{AE}\}$
  avec $r_{AB} = \{(a, b) | a \in d_A, b \in d_B, a \neq b\}$,
  $r_{BC} = \{(v, j), (v, r), (j, v), (j, r), (r, v), (r, j)\}$,
  ...

Exemple 2 (Morpion)

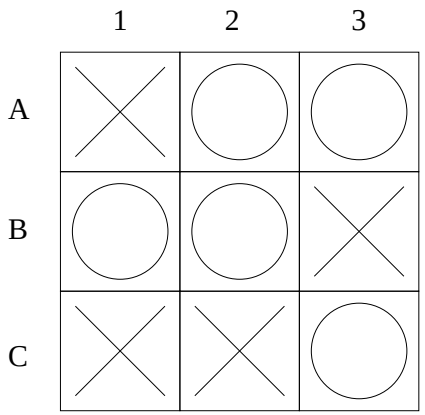


Exemple 2 (Morpion)

- $X = \{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$

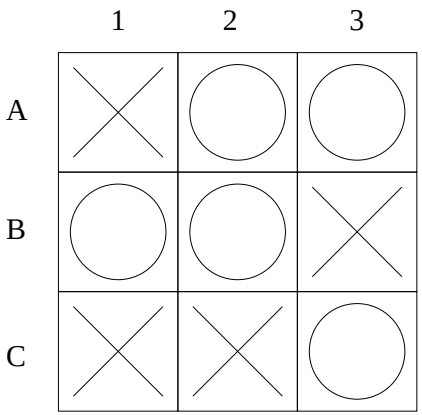


Exemple 2 (Morpion)



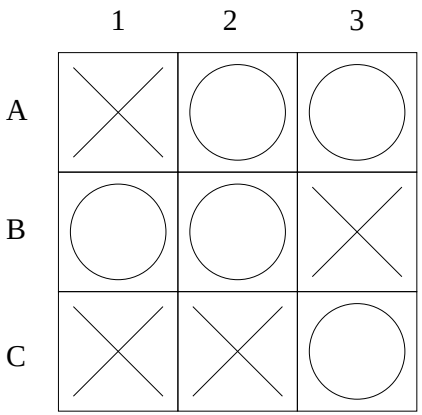
- $X = \{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$
- $D = \{d_{A1}, d_{A2}, \ldots, d_{C3}\}$ avec
  $d_{A1} = d_{A2} = \ldots = d_{C3} = \{O, X\}$

Exemple 2 (Morpion)



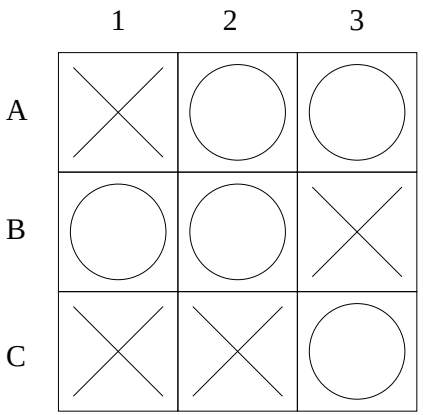
- $X = \{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$
- $D = \{d_{A1}, d_{A2}, \ldots, d_{C3}\}$ avec
  $d_{A1} = d_{A2} = \ldots = d_{C3} = \{O, X\}$
- $C = \{c_A, c_B, c_C, c_1, c_2, c_3, c_{d1}, c_{d2}, c_{all}\}$
  avec $c_A = \{A1, A2, A3\}$, $c_1 = \{A1, B1, C1\}$,
  $c_{d1} = \{A1, B2, C3\}$, $c_{all} = X$

Exemple 2 (Morpion)



- $X = \{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$
- $D = \{d_{A1}, d_{A2}, \ldots, d_{C3}\}$ avec
  $d_{A1} = d_{A2} = \ldots = d_{C3} = \{O, X\}$
- $C = \{c_A, c_B, c_C, c_1, c_2, c_3, c_{d1}, c_{d2}, c_{all}\}$
  avec $c_A = \{A1, A2, A3\}$, $c_1 = \{A1, B1, C1\}$,
  $c_{d1} = \{A1, B2, C3\}$, $c_{all} = X$
- $R = \{r_A, r_B, r_C, r_1, r_2, r_3, r_{d1}, r_{d2}, r_{all}\}$ avec
  $r_A = \{(O, O, X), (O, X, O), (X, O, O),$
  $(O, X, X), (X, O, X), (X, X, O)\}$,

Exemple 2 (Morpion)



- $X = \{A1, A2, A3, B1, B2, B3, C1, C2, C3\}$
- $D = \{d_{A1}, d_{A2}, \ldots, d_{C3}\}$ avec
  $d_{A1} = d_{A2} = \ldots = d_{C3} = \{O, X\}$
- $C = \{c_A, c_B, c_C, c_1, c_2, c_3, c_{d1}, c_{d2}, c_{all}\}$
  avec $c_A = \{A1, A2, A3\}$, $c_1 = \{A1, B1, C1\}$,
  $c_{d1} = \{A1, B2, C3\}$, $c_{all} = X$
- $R = \{r_A, r_B, r_C, r_1, r_2, r_3, r_{d1}, r_{d2}, r_{all}\}$ avec
  $r_A = \{(O, O, X), (O, X, O), (X, O, O),$
  $(O, X, X), (X, O, X), (X, X, O)\}$,
  $c_{all} = \{"la\ grille\ contient\ 5\ ronds\ et\ 4\ croix"\}$

Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

- $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

- $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$
- $D = \{d_{x_1}, d_{x_2}, d_{x_3}, d_{x_4}, d_{x_5}, d_{x_6}\}$
  avec $d_{x_i} = \{1, 2, 3\}$ $i \neq 5$, $d_{x_5} = \{1, 2\}$

Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

- $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$
- $D = \{d_{x_1}, d_{x_2}, d_{x_3}, d_{x_4}, d_{x_5}, d_{x_6}\}$
  avec $d_{x_i} = \{1, 2, 3\}$ $i \neq 5$, $d_{x_5} = \{1, 2\}$
- $C = \{c_{12}, c_{13}, c_{14}, c_{15}, c_{36}, c_{46}\}$
  avec $c_{ij} = \{x_i, x_j\}$

Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

- $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$
- $D = \{d_{x_1}, d_{x_2}, d_{x_3}, d_{x_4}, d_{x_5}, d_{x_6}\}$ avec $d_{x_i} = \{1, 2, 3\}$ $i \neq 5$, $d_{x_5} = \{1, 2\}$
- $C = \{c_{12}, c_{13}, c_{14}, c_{15}, c_{36}, c_{46}\}$ avec $c_{ij} = \{x_i, x_j\}$
- $R = \{r_{12}, r_{13}, r_{14}, r_{15}, r_{36}, r_{46}\}$ avec $r_{36} = \{x_3^2 < x_6\}$, $r_{46} = \{x_4 > x_6\}$

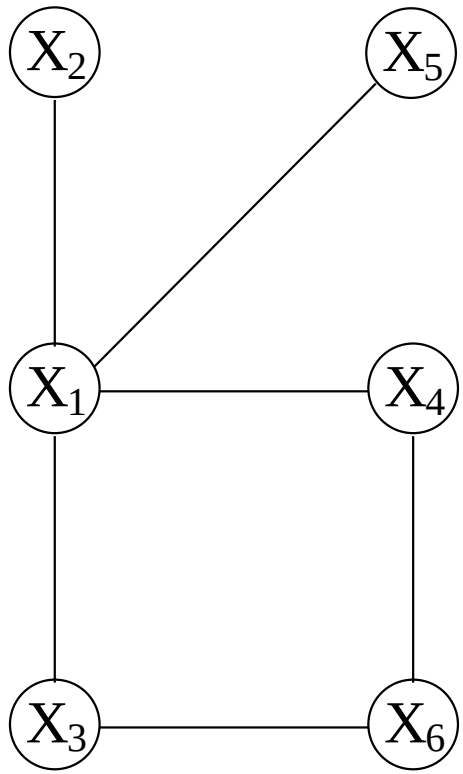
Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

- $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$
- $D = \{d_{x_1}, d_{x_2}, d_{x_3}, d_{x_4}, d_{x_5}, d_{x_6}\}$
  avec $d_{x_i} = \{1, 2, 3\}$ $i \neq 5$, $d_{x_5} = \{1, 2\}$
- $C = \{c_{12}, c_{13}, c_{14}, c_{15}, c_{36}, c_{46}\}$
  avec $c_{ij} = \{x_i, x_j\}$
- $R = \{r_{12}, r_{13}, r_{14}, r_{15}, r_{36}, r_{46}\}$ avec
  $r_{36} = \{x_3^2 < x_6\}$, $r_{46} = \{x_4 > x_6\}$
  $r_{ij} = \{x_i < x_j\}$

Exemple 3 (Système d'inéquations)

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$



Affectation / Instanciation

Instanciation des variables de $Y = \{y_1, \ldots, y_k\}$ = k-uplet $(v_1, \ldots, v_k)$ de $d_{y_1} \times \cdots \times d_{y_k}$.

Affectation / Instanciation

Instanciation des variables de $Y = \{y_1, \ldots, y_k\}$ = k-uplet $(v_1, \ldots, v_k)$ de $d_{y_1} \times \cdots \times d_{y_k}$.

Instanciation complète = instanciation de toutes les variables

Affectation / Instanciation

Instanciation des variables de $Y = \{y_1, \ldots, y_k\}$ = k-uplet $(v_1, \ldots, v_k)$ de $d_{y_1} \times \cdots \times d_{y_k}$.

Instanciation complète = instanciation de toutes les variables

Instanciation partielle = instanciation d'une partie des variables

Affectation / Instanciation

Instanciation des variables de $Y = \{y_1, \ldots, y_k\}$ = k-uplet $(v_1, \ldots, v_k)$ de $d_{y_1} \times \cdots \times d_{y_k}$.

Instanciation complète = instanciation de toutes les variables

Instanciation partielle = instanciation d'une partie des variables

Notation : $\{x_1 \leftarrow a, x_2 \leftarrow c\}$

Affectation / Instanciation

Une affectation $\mathcal{A}$ de $Y = \{y_1, \ldots, y_k\}$ **satisfait** une contrainte c si $\mathcal{A}[c] \in r_c$.

$\mathcal{A}$ **viole** c sinon.

Affectation / Instanciation

Une affectation $\mathcal{A}$ de $Y = \{y_1, \ldots, y_k\}$ **satisfait** une contrainte c si $\mathcal{A}[c] \in r_c$.

$\mathcal{A}$ **viole** c sinon.

Instanciation **consistante** = instanciation qui satisfait toutes les contraintes c telles que $c \subseteq Y$

Affectation / Instanciation

Une affectation $\mathcal{A}$ de $Y = \{y_1, \ldots, y_k\}$ **satisfait** une contrainte c si $\mathcal{A}[c] \in r_c$.

$\mathcal{A}$ **viole** c sinon.

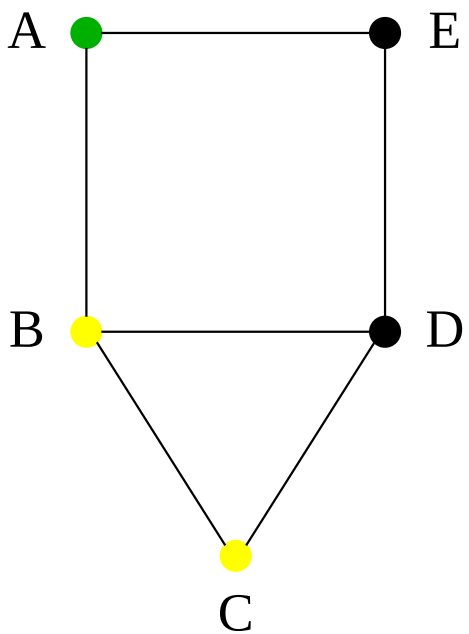
Instanciation **consistante** = instanciation qui satisfait toutes les contraintes c telles que $c \subseteq Y$

Instanciation **inconsistante** = instanciation qui viole au moins une contrainte c telle que $c \subseteq Y$

Exemple 1

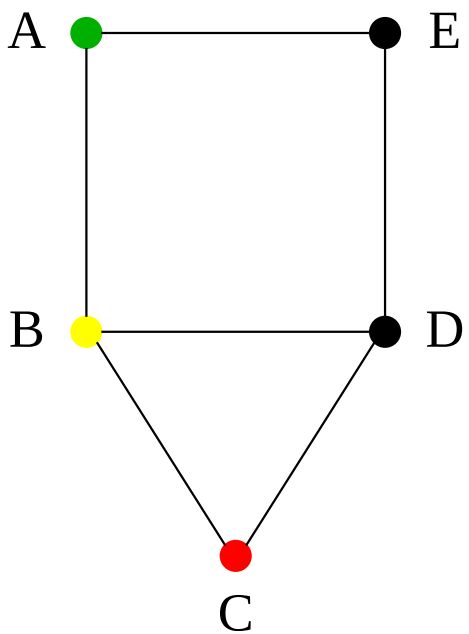
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)

- $\{A \leftarrow v, B \leftarrow j, C \leftarrow j\}$ viole c_{BC}



Exemple 1

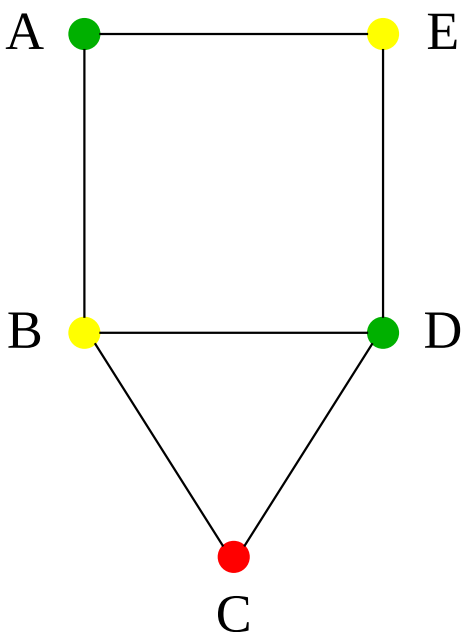
Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



- $\{A \leftarrow v, B \leftarrow j, C \leftarrow j\}$ viole c_{BC}
- $\{A \leftarrow v, B \leftarrow j, C \leftarrow r\}$ est consistante

Exemple 1

Coloration de graphe avec 3 couleurs (vert, jaune, rouge)



- $\{A \leftarrow v, B \leftarrow j, C \leftarrow j\}$ viole c_{BC}
- $\{A \leftarrow v, B \leftarrow j, C \leftarrow r\}$ est consistante
- $\{A \leftarrow v, B \leftarrow j, C \leftarrow r, D \leftarrow v, E \leftarrow j\}$ est une affectation complète consistante

Solution

Solution = affectation complète consistante

Solution

Solution = affectation complète consistante

Problème **consistant** = problème possédant au moins une solution

Solution

Solution = affectation complète consistante

Problème **consistant** = problème possédant au moins une solution

Problème **inconsistant** = problème ne possédant aucune solution

Exemple

- Les deux premiers problèmes sont consistants

  Par exemple, $\{A \leftarrow v, B \leftarrow j, C \leftarrow r, D \leftarrow v, E \leftarrow j\}$ est une solution du problème de coloration.

Exemple

- Les deux premiers problèmes sont consistants

  Par exemple, $\{A \leftarrow v, B \leftarrow j, C \leftarrow r, D \leftarrow v, E \leftarrow j\}$ est une solution du problème de coloration.

- Le troisième problème est inconsistant.

Problèmes posés

- Une instance possède-t-elle une solution ?

Problèmes posés

- Une instance possède-t-elle une solution ?
- Trouver une solution d’une instance.

Problèmes posés

- Une instance possède-t-elle une solution ?
- Trouver une solution d’une instance.
- Trouver toutes les solutions d’une instance.

Problèmes posés

- Une instance possède-t-elle une solution ?
- Trouver une solution d’une instance.
- Trouver toutes les solutions d’une instance.
- Trouver le nombre de solutions d’une instance.

Problèmes posés

- Une instance possède-t-elle une solution ?
- Trouver une solution d’une instance.
- Trouver toutes les solutions d’une instance.
- Trouver le nombre de solutions d’une instance.
- Trouver la meilleure solution d’une instance.

Problèmes posés

- Une instance possède-t-elle une solution ?
- Trouver une solution d’une instance.
- Trouver toutes les solutions d’une instance.
- Trouver le nombre de solutions d’une instance.
- Trouver la meilleure solution d’une instance.
- . . .

Problèmes posés

- Une instance possède-t-elle une solution ?
- Trouver une solution d'une instance.
- Trouver toutes les solutions d'une instance.
- Trouver le nombre de solutions d'une instance.
- Trouver la meilleure solution d'une instance.
- ...

Taille de l'espace de recherche : d^n

Méthodes de résolution

- Les méthodes complètes

Méthodes de résolution

- Les méthodes complètes
  - Les méthodes énumératives
  - Les méthodes structurelles

Méthodes de résolution

- Les méthodes complètes
  - Les méthodes énumératives
  - Les méthodes structurelles
- Les méthodes incomplètes

Plan

1. Introduction
2. Le formalisme CSP
3. Méthodes de résolution énumératives

Generate and Test (GT)

Principe :

- on génère une affectation complète,

Generate and Test (GT)

Principe :

- on génère une affectation complète,
- on teste s'il s'agit d'une solution,

Generate and Test (GT)

Principe :

- on génère une affectation complète,
- on teste s'il s'agit d'une solution,
- on réitère le procédé tant qu'on n'a pas trouvé une solution et qu'on n'a pas essayé toutes les affectations complètes

Generate and Test (GT)

Principe :

- on génère une affectation complète,
- on teste s'il s'agit d'une solution,
- on réitère le procédé tant qu'on n'a pas trouvé une solution et qu'on n'a pas essayé toutes les affectations complètes

Complexité : $O(md^n)$

Generate and Test (GT)

Principe :

- on génère une affectation complète,
- on teste s'il s'agit d'une solution,
- on réitère le procédé tant qu'on n'a pas trouvé une solution et qu'on n'a pas essayé toutes les affectations complètes

Complexité : $O(md^n)$

Inefficace en pratique !!!

Exemple

$$\begin{cases} x_1 < x_2 \\ x_1 < x_3 \\ x_1 < x_4 \\ x_1 < x_5 \\ x_3^2 < x_6 \\ x_4 > x_6 \\ x_5 \in \{1, 2\} \\ x_i \in \{1, 2, 3\}, i \neq 5 \end{cases}$$

$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 1\}$$

Exemple

$$
\begin{cases}
x_1 < x_2 \\
x_1 < x_3 \\
x_1 < x_4 \\
x_1 < x_5 \\
x_3^2 < x_6 \\
x_4 > x_6 \\
x_5 \in \{1, 2\} \\
x_i \in \{1, 2, 3\}, i \neq 5
\end{cases}
$$

$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 1\}$$

$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 2\}$$

Exemple

$$
\begin{cases}
x_1 < x_2 & \\
x_1 < x_3 & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 1\} \\
x_1 < x_4 & \\
x_1 < x_5 & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 2\} \\
x_3^2 < x_6 & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 3\} \\
x_4 > x_6 & \\
x_5 \in \{1, 2\} & \\
x_i \in \{1, 2, 3\}, i \neq 5 &
\end{cases}
$$

Exemple

$$
\begin{cases}
x_1 < x_2 & \\
x_1 < x_3 & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 1\} \\
x_1 < x_4 & \\
x_1 < x_5 & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 2\} \\
x_3^2 < x_6 & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1, x_6 \leftarrow 3\} \\
x_4 > x_6 & \\
x_5 \in \{1, 2\} & \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2, x_6 \leftarrow 1\} \\
x_i \in \{1, 2, 3\}, i \neq 5 & \ldots
\end{cases}
$$

Backtrack (BT)

Principe :

- on étend progressivement une affectation consistante

Backtrack (BT)

Principe :

- on étend progressivement une affectation consistante
- en cas d'échec, on change la valeur de la variable courante

Backtrack (BT)

Principe :

- on étend progressivement une affectation consistante
- en cas d'échec, on change la valeur de la variable courante
- s'il n'y a plus de valeur, on revient sur la variable précédente

Backtrack (BT)

Principe :

- on étend progressivement une affectation consistante
- en cas d'échec, on change la valeur de la variable courante
- s'il n'y a plus de valeur, on revient sur la variable précédente
- on réitère le procédé tant qu'on n'a pas trouvé une solution et qu'on n'a pas essayé toutes les possibilités

Backtrack (BT)

Principe :

- on étend progressivement une affectation consistante
- en cas d'échec, on change la valeur de la variable courante
- s'il n'y a plus de valeur, on revient sur la variable précédente
- on réitère le procédé tant qu'on n'a pas trouvé une solution et qu'on n'a pas essayé toutes les possibilités

Complexité : $O(md^n)$

Backtrack (BT)

BT$(\mathcal{A}, V)$

Si $V = \emptyset$ **Alors** $\mathcal{A}$ est une solution
Sinon
  Choisir $x \in V$
  $d \leftarrow d_x$
  TantQue $d \neq \emptyset$
    Choisir v dans d
    $d \leftarrow d \backslash \{v\}$
    Si $\mathcal{A} \cup \{x \leftarrow v\}$ est consistante
    Alors BT$(\mathcal{A} \cup \{x \leftarrow v\}, V \backslash \{x\})$
    FinSi
  FinTantQue
FinSi

Exemple

X_1 1

X_2

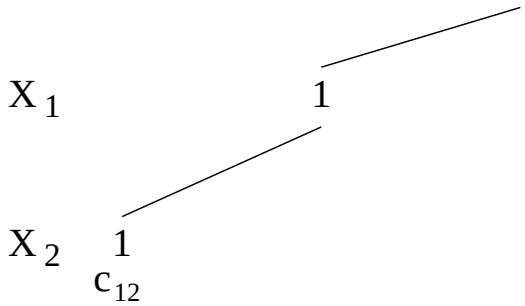
X_3

X_4

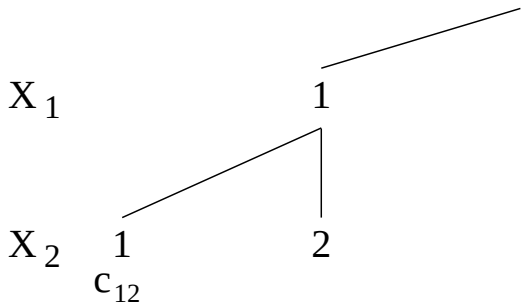
X_5

X_6

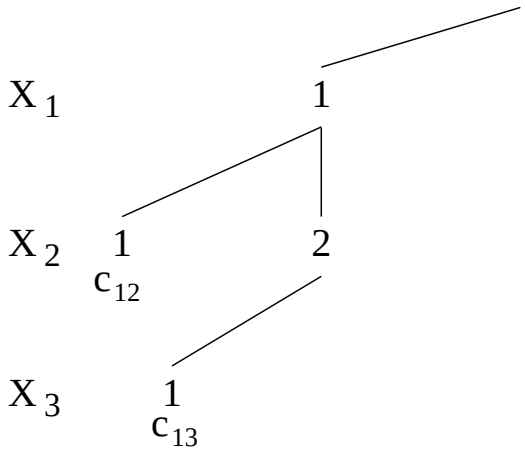
Exemple



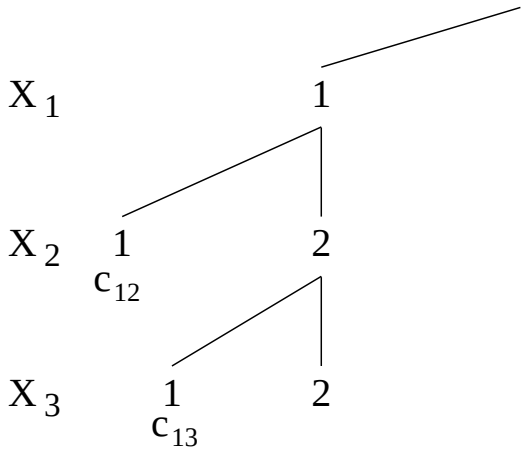
Exemple



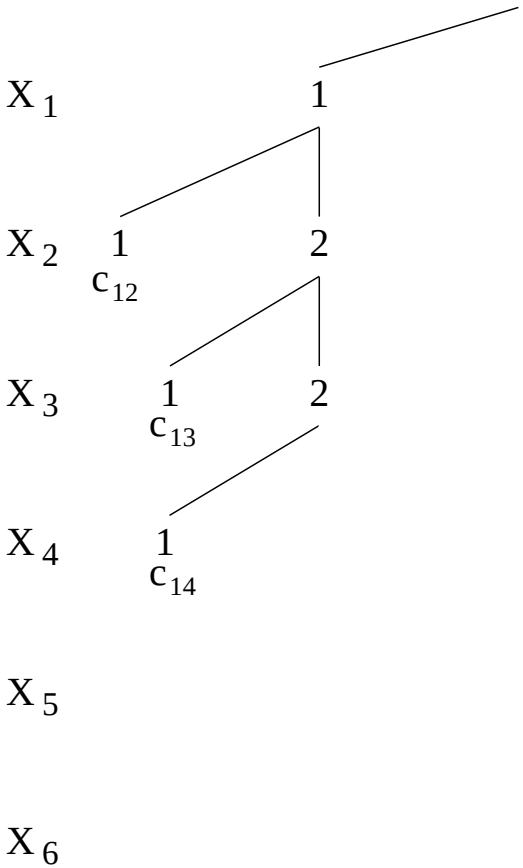
Exemple



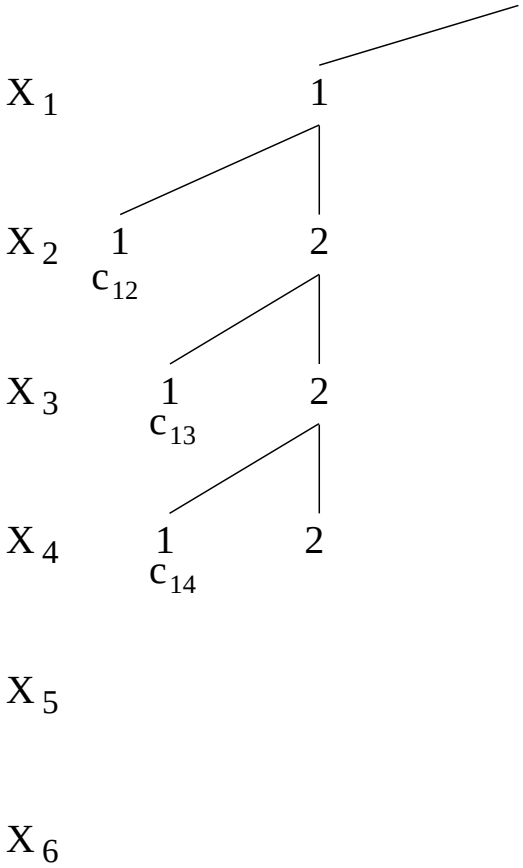
Exemple



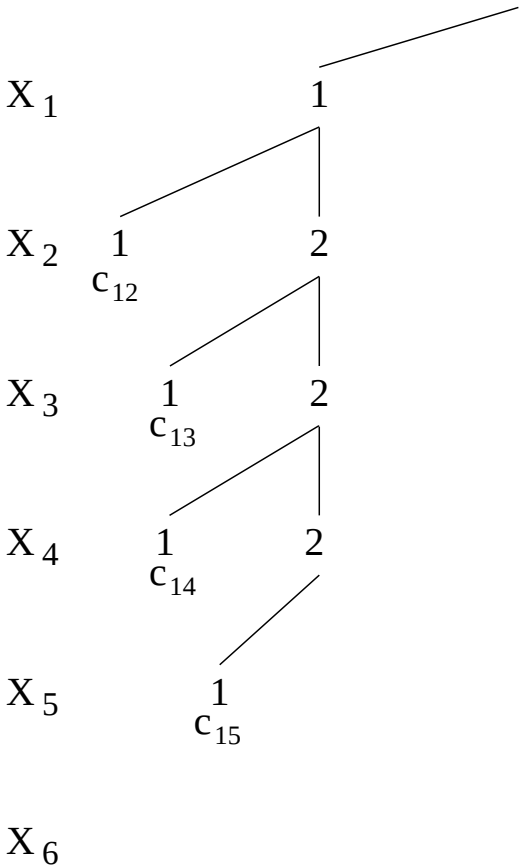
Exemple



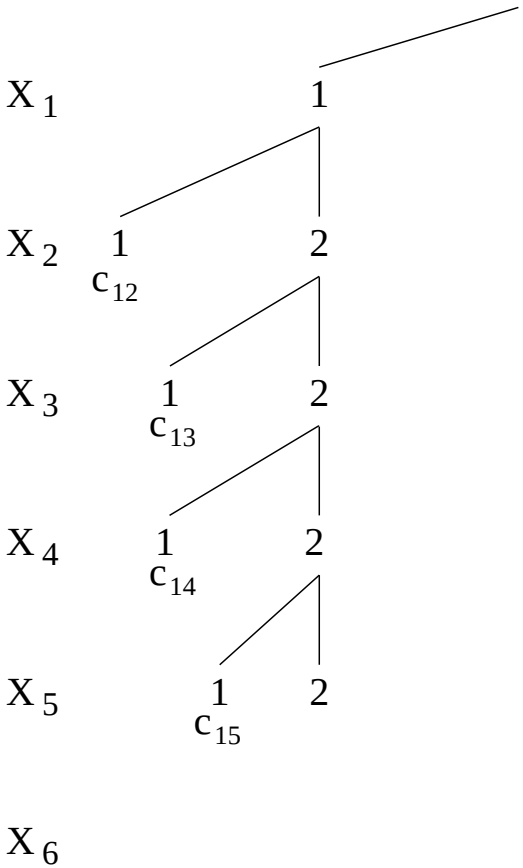
Exemple



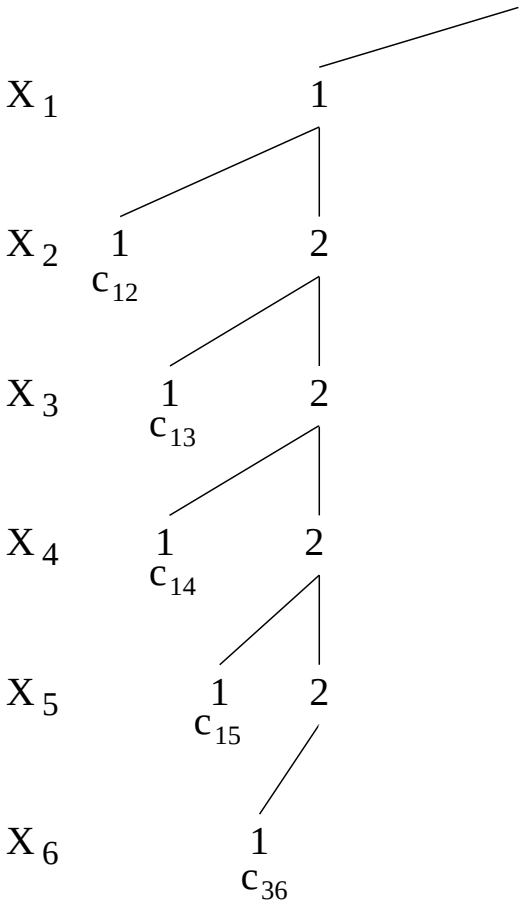
Exemple



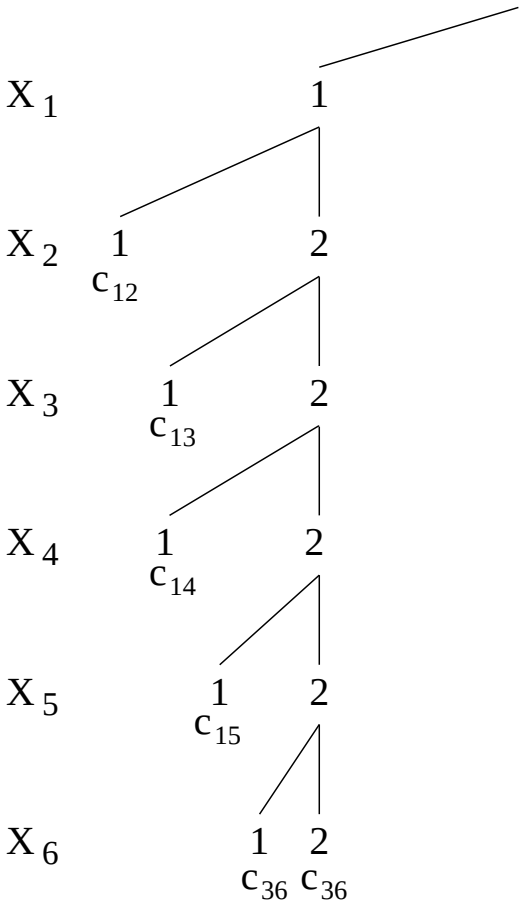
Exemple



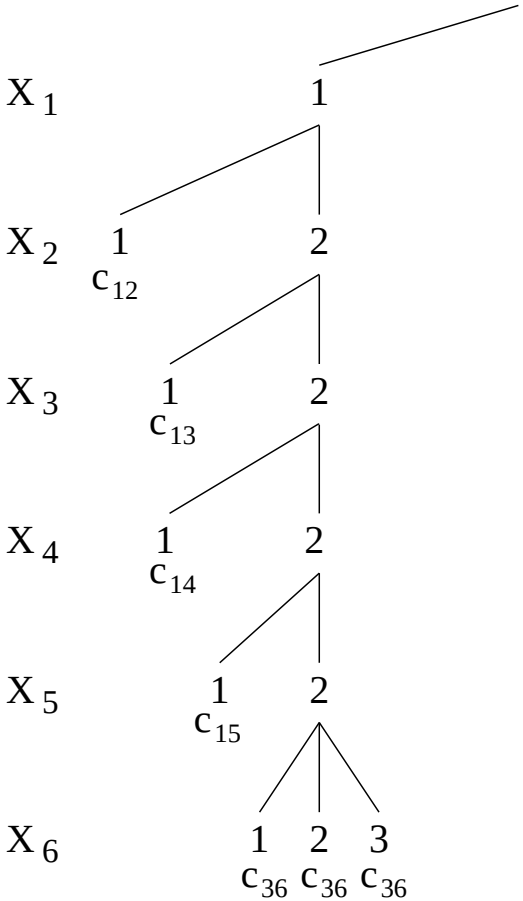
Exemple



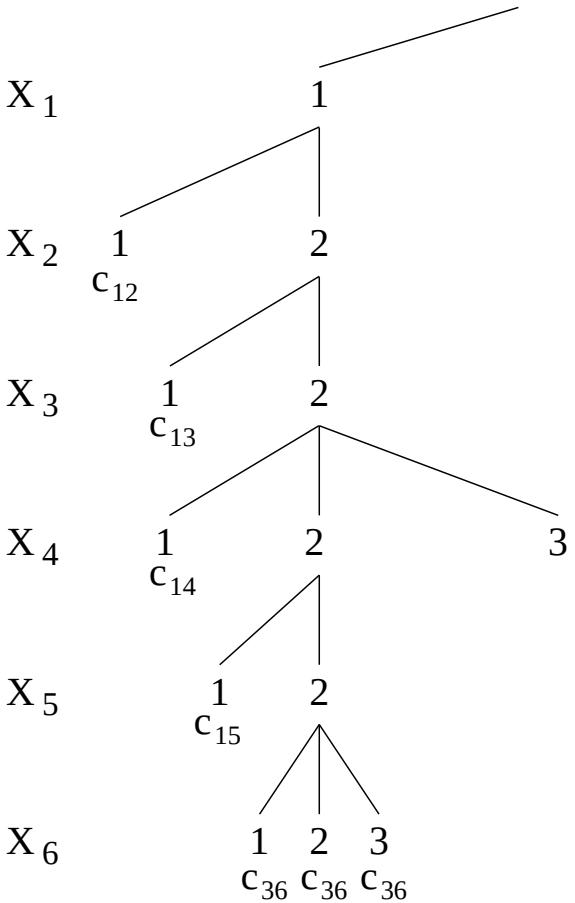
Exemple



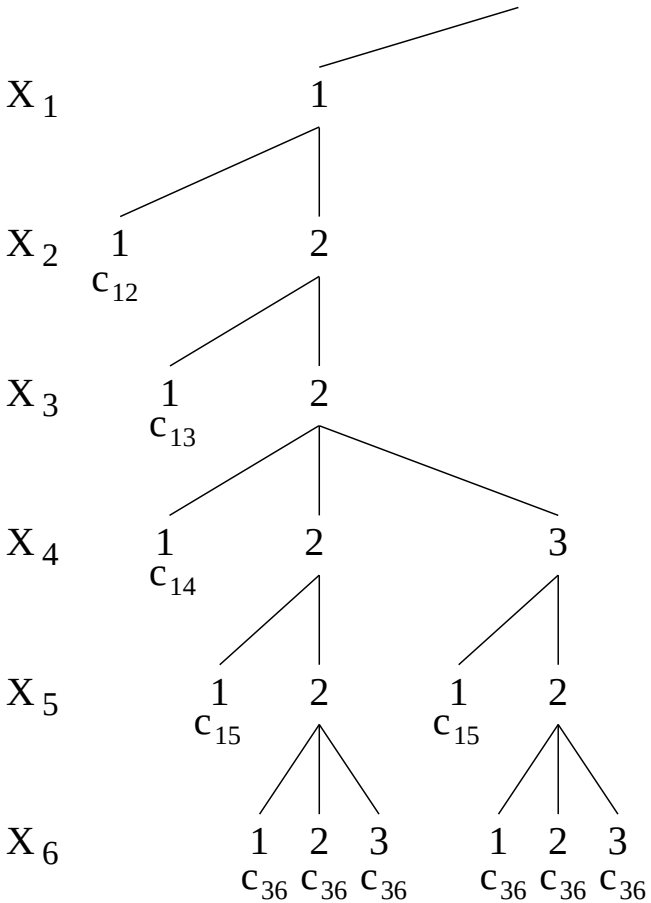
Exemple



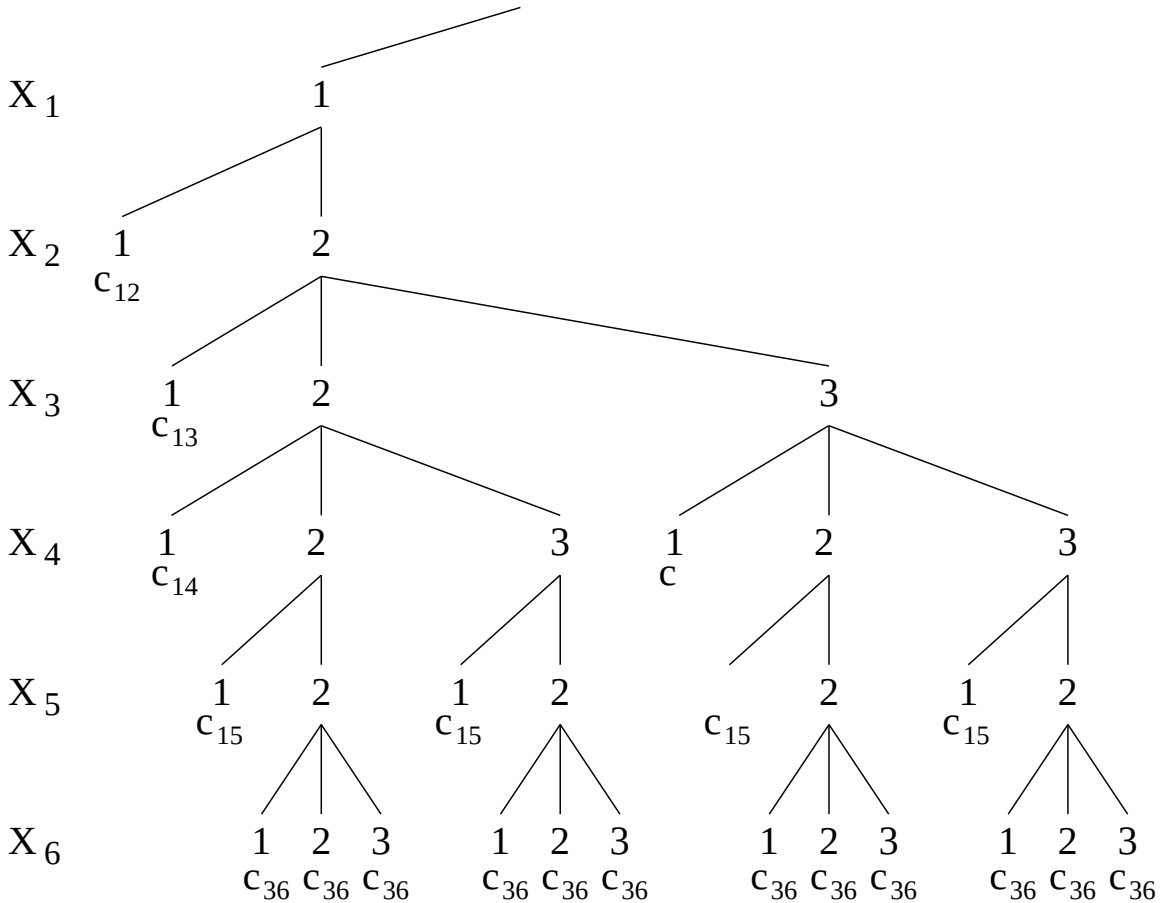
Exemple



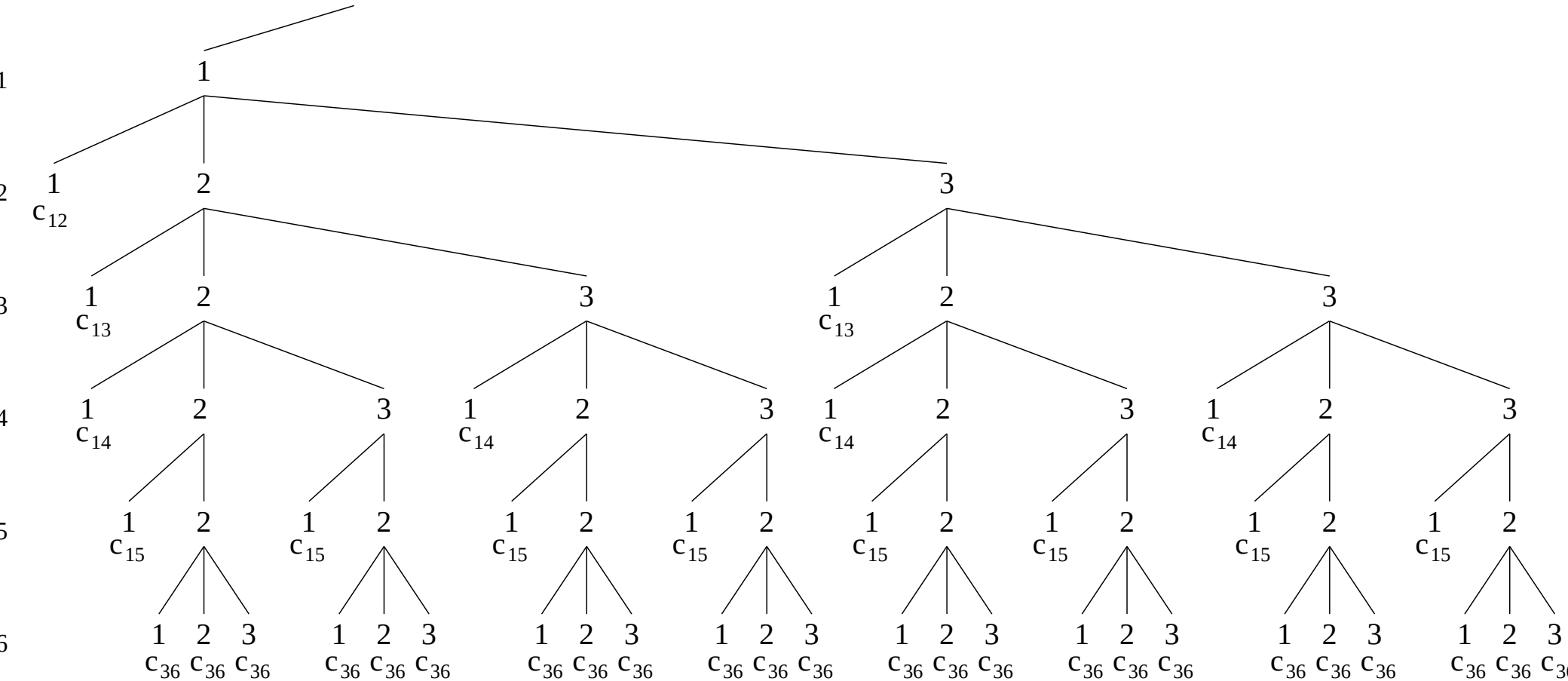
Exemple


X1
1
X2
1
c12
2
X3
1
c13
2
X4
1
c14
2
3
X5
1
c15
2
1
c15
2
X6
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36

Exemple


X1
1
X2
1
c12
2
X3
1
c13
2
3
X4
1
c14
2
3
1
c
2
3
X5
1
c15
2
1
c15
2
c15
2
1
c15
2
X6
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36

Exemple



Améliorer Backtrack

- Analyser les échecs

Améliorer Backtrack

- Analyser les échecs
- Mémoriser les échecs

Améliorer Backtrack

- Analyser les échecs
- Mémoriser les échecs
- Simplifier le problème

Analyser les échecs

Backtrack : retour en arrière chronologique sur la variable précédente

Analyser les échecs

Backtrack : retour en arrière chronologique sur la variable précédente

Principe :

- identifier les causes de l’échec

Analyser les échecs

Backtrack : retour en arrière chronologique sur la variable précédente

Principe :

- identifier les causes de l'échec
- modifier l'affectation d'une variable en cause dans l'échec

Analyser les échecs

Backtrack : retour en arrière chronologique sur la variable précédente

Principe :

- identifier les causes de l’échec
- modifier l’affectation d’une variable en cause dans l’échec

Ce principe est appelé "retour arrière intelligent".

Backjumping (BJ)

Soit x la variable courante

Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

Backjumping (BJ)

Soit x la variable courante

Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

On ne considère que les échecs "immédiats"

Backjumping (BJ)

Soit x la variable courante

Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

On ne considère que les échecs "immédiats"

On revient sur la variable la plus profonde dans l'arbre de recherche qui soit en cause dans l'échec

Backjumping (BJ)

Soit x la variable courante

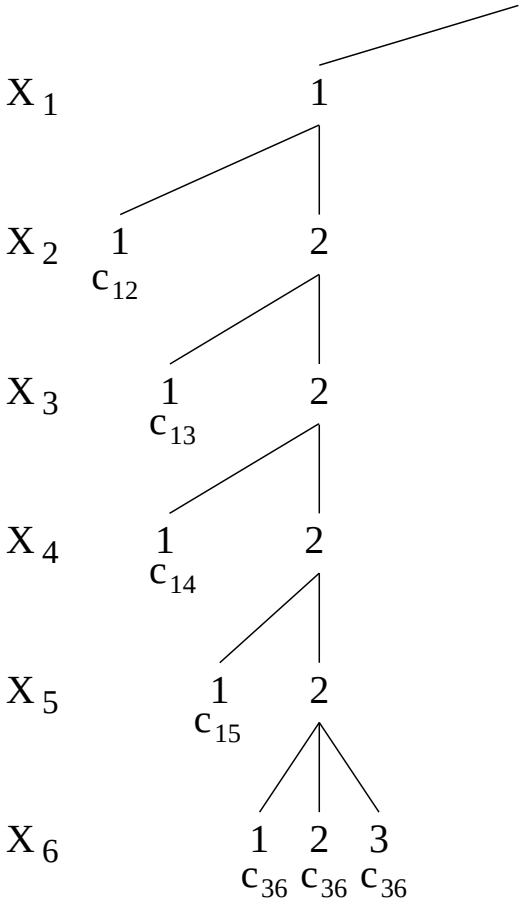
Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

On ne considère que les échecs "immédiats"

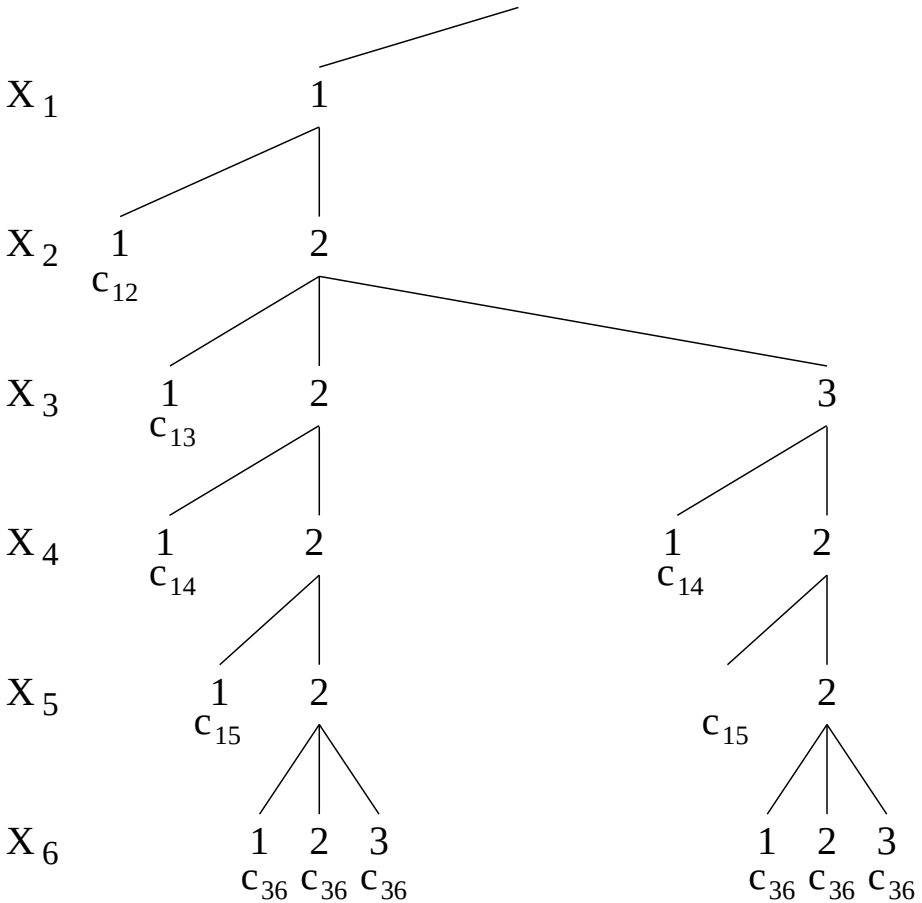
On revient sur la variable la plus profonde dans l'arbre de recherche qui soit en cause dans l'échec

Complexité : $O(md^n)$

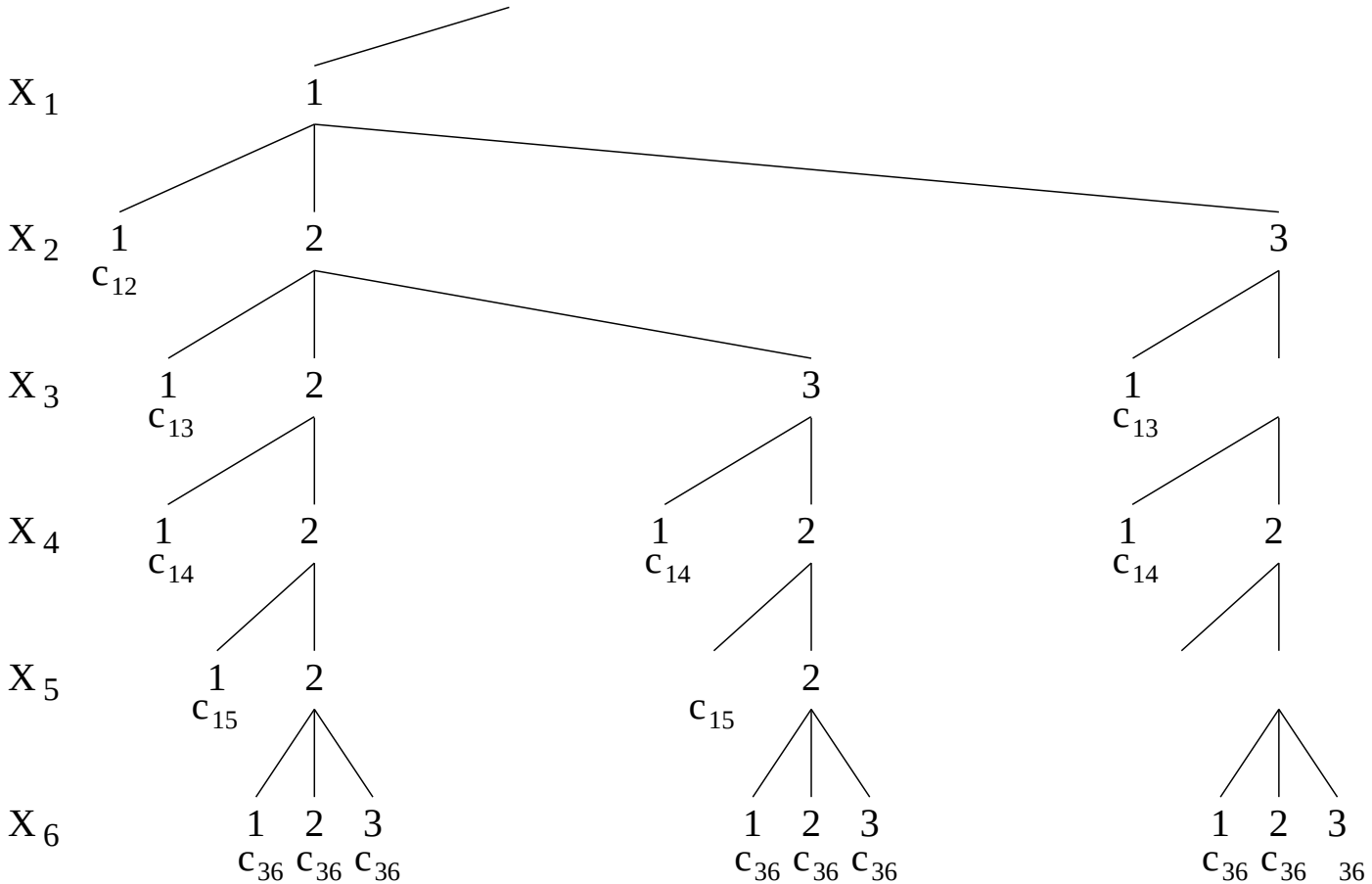
Exemple



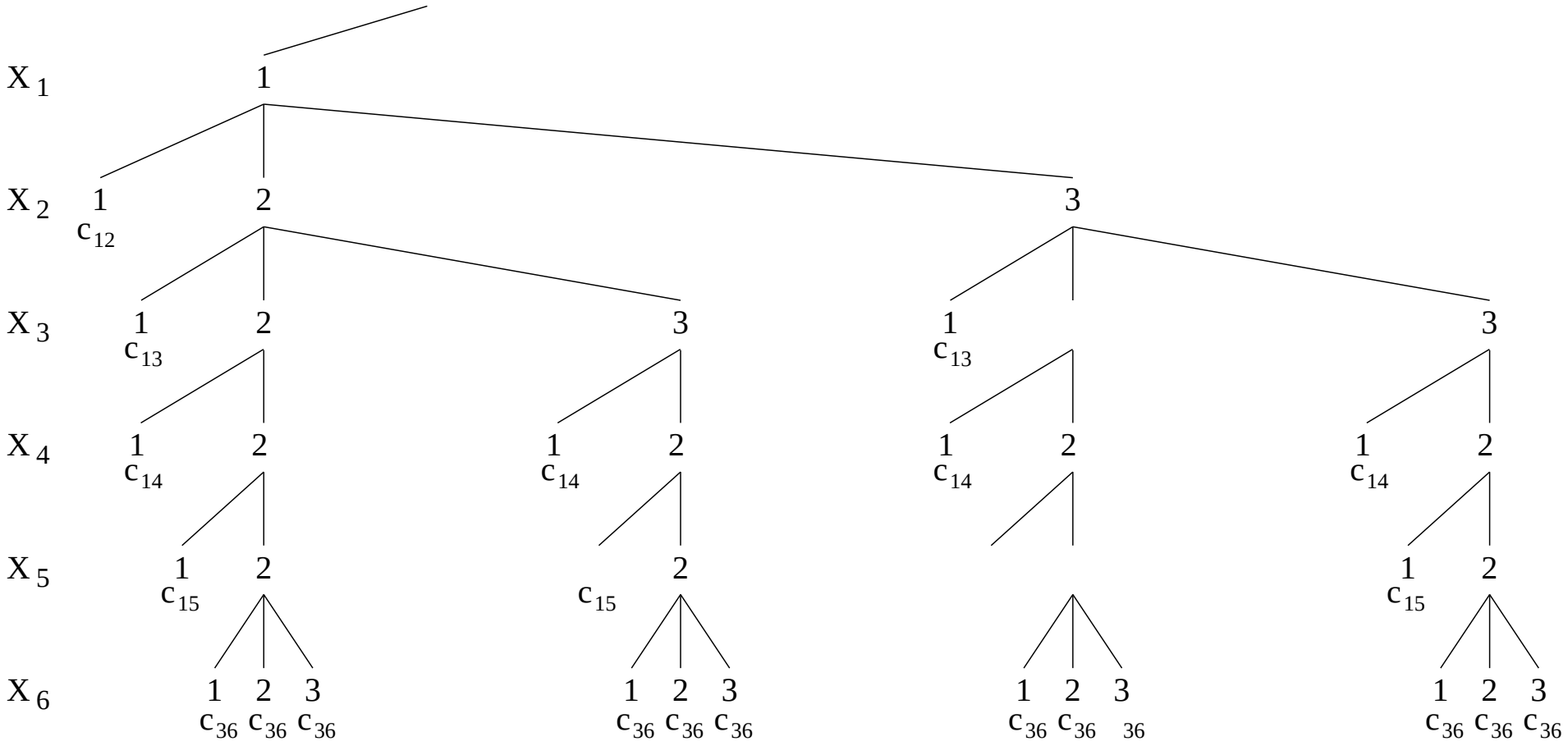
Exemple


X 1
1
X 2
1
c 12
2
X 3
1
c 13
2
3
X 4
1
c 14
2
1
c 14
2
X 5
1
c 15
2
c 15
2
X 6
1 2 3
c 36 c 36 c 36
1 2 3
c 36 c 36 c 36

Exemple


X 1
1
X 2
1
c12
2
3
X 3
1
c13
2
3
1
c13
X 4
1
c14
2
1
c14
2
1
c14
2
X 5
1
c15
2
c15
2
X 6
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36
1 2 3
c36 c36 36

Exemple



Graph-based Backjumping (GBJ)

Analyse d’après le graphe de contraintes

Graph-based Backjumping (GBJ)

Analyse d'après le graphe de contraintes

Soit x la variable courante

Cause d'un échec : toute variable instanciée y telle qu'il existe une contrainte entre x et y

Graph-based Backjumping (GBJ)

Analyse d'après le graphe de contraintes

Soit x la variable courante

Cause d'un échec : toute variable instanciée y telle qu'il existe une contrainte entre x et y

On revient sur la variable la plus profonde dans l'arbre de recherche qui soit en cause dans l'échec

Graph-based Backjumping (GBJ)

Analyse d'après le graphe de contraintes

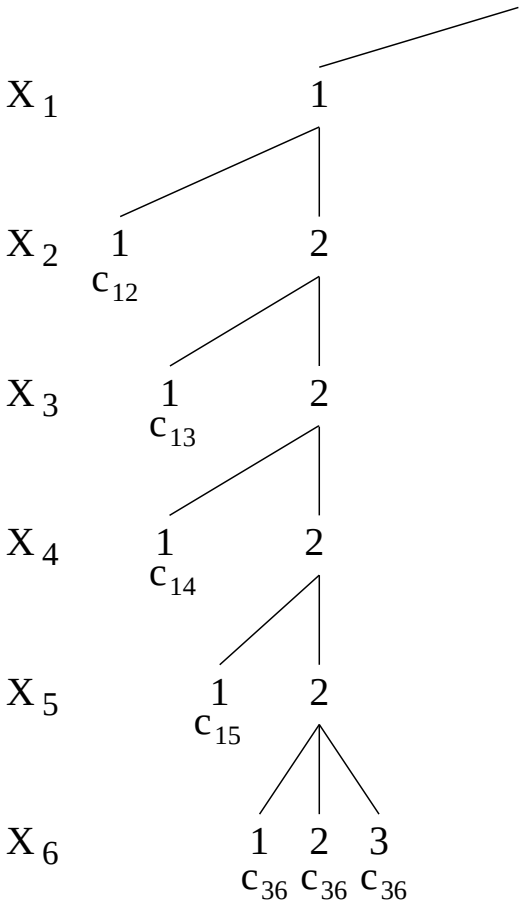
Soit x la variable courante

Cause d'un échec : toute variable instanciée y telle qu'il existe une contrainte entre x et y

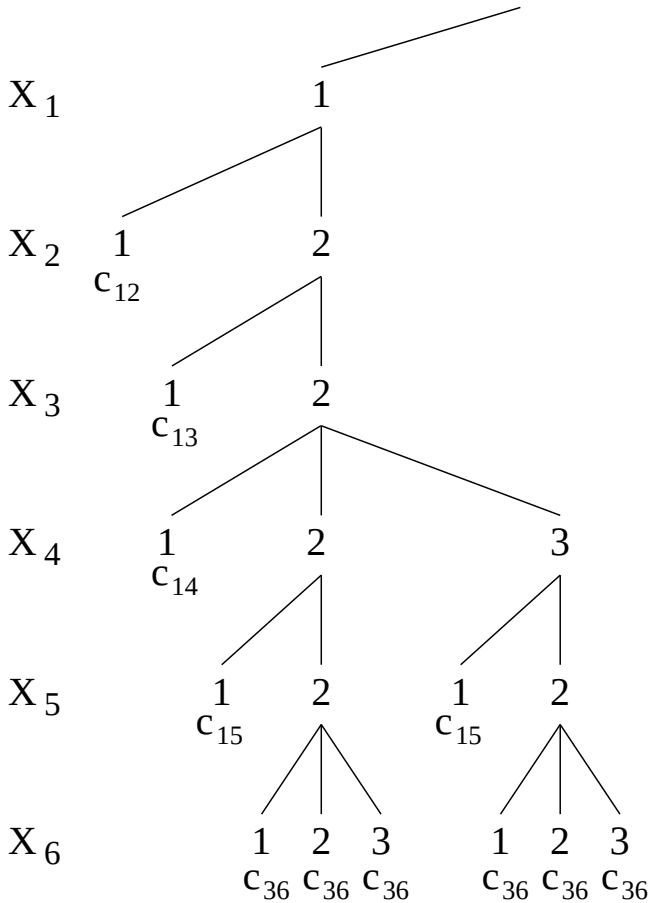
On revient sur la variable la plus profonde dans l'arbre de recherche qui soit en cause dans l'échec

Complexité : $O(md^n)$

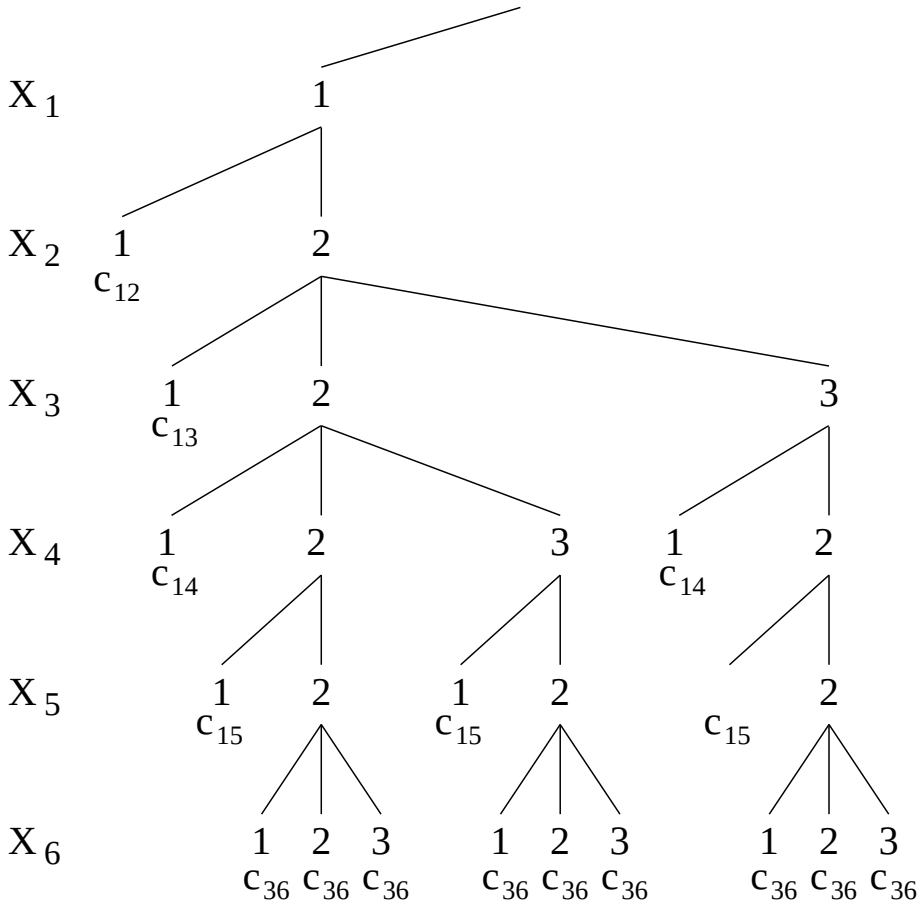
Exemple



Exemple


X1
1
X2
1
c12
2
X3
1
c13
2
X4
1
c14
2
3
X5
1
c15
2
1
c15
2
X6
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36

Exemple


X1
1
X2
1
c12
2
X3
1
c13
2
3
X4
1
c14
2
3
1
c14
2
X5
1
c15
2
1
c15
2
c15
2
X6
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36

Conflict-directed Backjumping (CBJ)

Analyse d’après les conflits effectivement rencontrés

Conflict-directed Backjumping (CBJ)

Analyse d'après les conflits effectivement rencontrés

Soit x la variable courante

Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

Conflict-directed Backjumping (CBJ)

Analyse d'après les conflits effectivement rencontrés

Soit x la variable courante

Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

On revient sur la variable la plus profonde dans l'arbre de recherche qui soit en cause dans l'échec

Conflict-directed Backjumping (CBJ)

Analyse d'après les conflits effectivement rencontrés

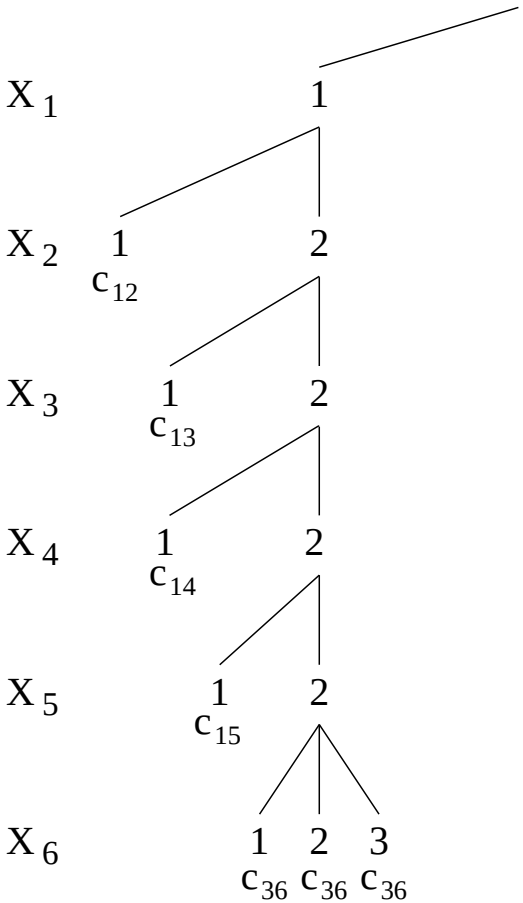
Soit x la variable courante

Cause d'un échec : une variable instanciée y telle que la contrainte $c = \{x, y\}$ soit violée

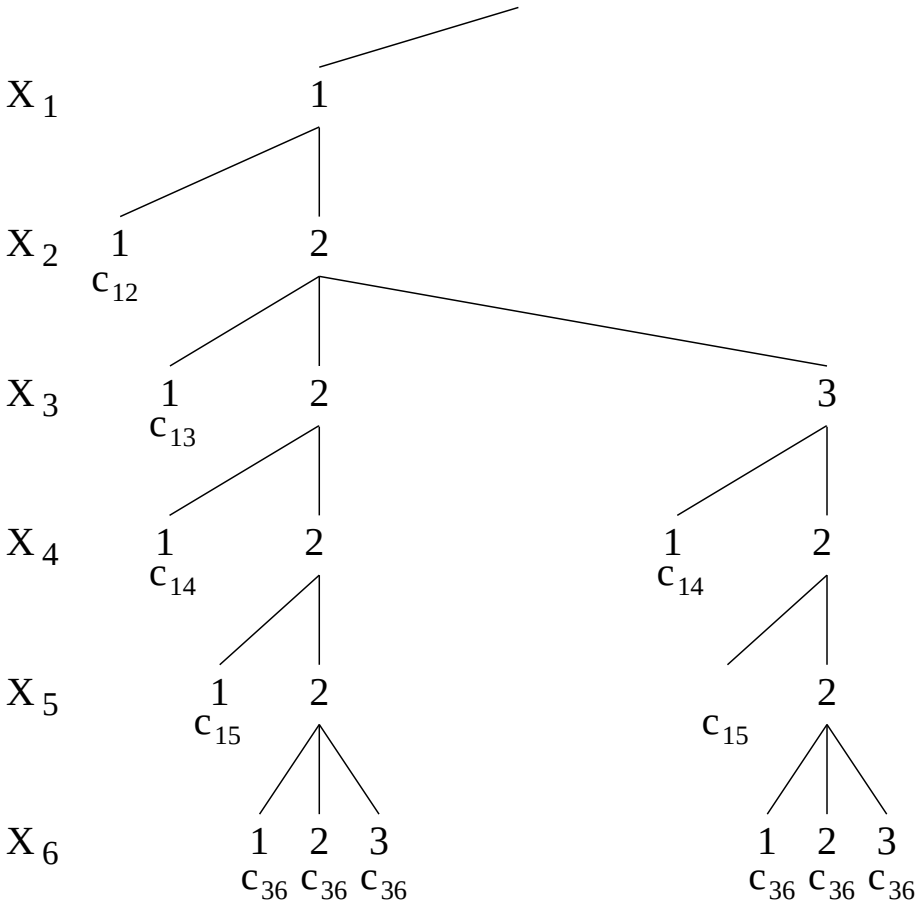
On revient sur la variable la plus profonde dans l'arbre de recherche qui soit en cause dans l'échec

Complexité : $O(md^n)$

Exemple



Exemple


X1
1
X2
1
c12
2
X3
1
c13
2
3
X4
1
c14
2
1
c14
2
X5
1
c15
2
c15
2
X6
1 2 3
c36 c36 c36
1 2 3
c36 c36 c36

Synthèse du backjumping

Avantages :

- éviter certaines redondances dans la recherche

Synthèse du backjumping

Avantages :

- éviter certaines redondances dans la recherche
- des méthodes plus efficaces que BT

Synthèse du backjumping

Avantages :

- éviter certaines redondances dans la recherche
- des méthodes plus efficaces que BT

Inconvénient :

- un surcoût en temps pas toujours compensé par les économies réalisées

Mémoriser les échecs

Echec = une information explicitée

Mémoriser les échecs

Echec = une information explicitée

Objectif : éviter certaines redondances dans la recherche

Mémoriser les échecs

Echec = une information explicitée

Objectif : éviter certaines redondances dans la recherche

Principe :

- identifier les causes de l'échec

Mémoriser les échecs

Echec = une information explicitée

Objectif : éviter certaines redondances dans la recherche

Principe :

- identifier les causes de l'échec
- mémoriser l'affectation des variables en cause dans l'échec comme une nouvelle contrainte

Mémoriser les échecs

Avantage :

- on évite certaines redondances

Mémoriser les échecs

Avantage :

- on évite certaines redondances

Inconvénients :

- un surcoût en temps pas toujours compensé par les économies réalisées

Mémoriser les échecs

Avantage :

- on évite certaines redondances

Inconvénients :

- un surcoût en temps pas toujours compensé par les économies réalisées
- le nombre d'échecs à mémoriser peut être exponentiel

Mémoriser les échecs

Avantage :

- on évite certaines redondances

Inconvénients :

- un surcoût en temps pas toujours compensé par les économies réalisées
- le nombre d'échecs à mémoriser peut être exponentiel

En pratique, on limite la quantité d'informations mémorisées.

Nogood Recording (NR)

Nogood = un tuple interdit par une contrainte

Nogood Recording (NR)

Nogood = un tuple interdit par une contrainte

Principe :

- identifier les causes de l'échec comme CBJ

Nogood Recording (NR)

Nogood = un tuple interdit par une contrainte

Principe :

- identifier les causes de l'échec comme CBJ
- mémoriser le nogood correspondant à l'affectation des variables en cause dans l'échec

Nogood Recording (NR)

Nogood = un tuple interdit par une contrainte

Principe :

- identifier les causes de l'échec comme CBJ
- mémoriser le nogood correspondant à l'affectation des variables en cause dans l'échec
- utiliser la technique de backjumping de CBJ

Nogood Recording (NR)

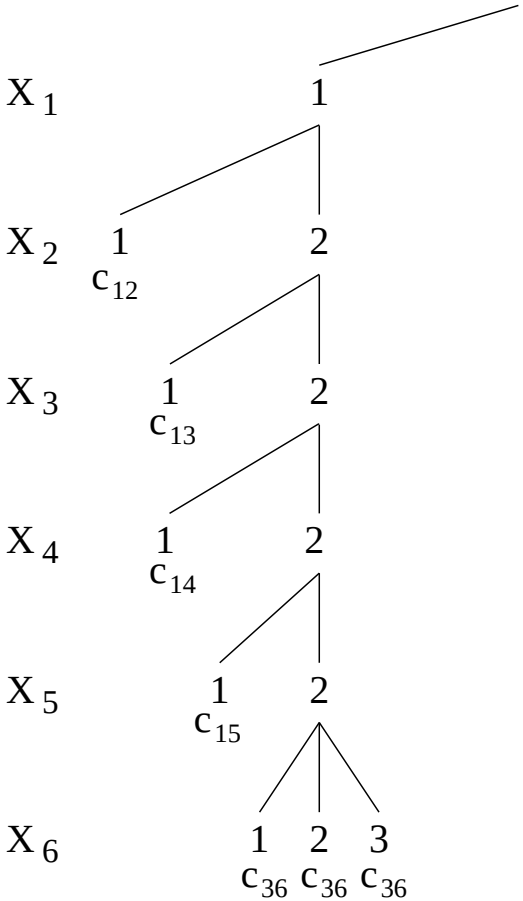
Nogood = un tuple interdit par une contrainte

Principe :

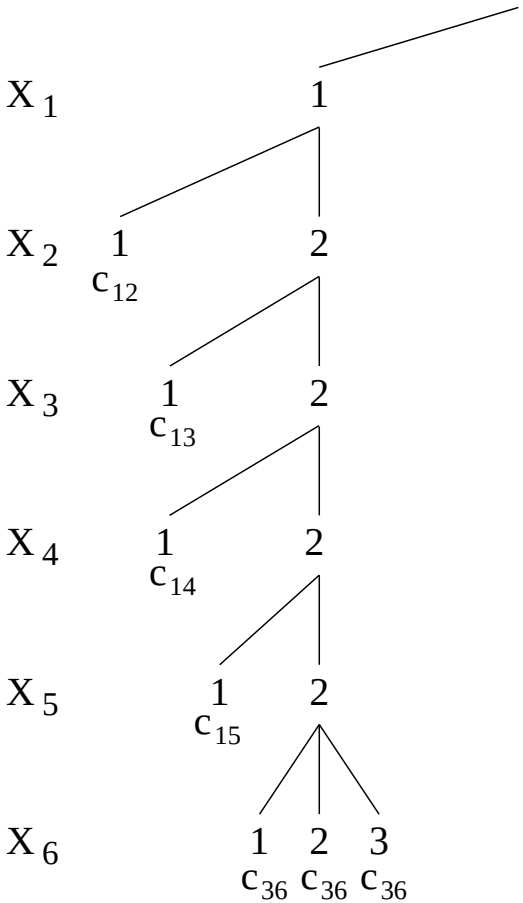
- identifier les causes de l'échec comme CBJ
- mémoriser le nogood correspondant à l'affectation des variables en cause dans l'échec
- utiliser la technique de backjumping de CBJ

En pratique, on limite la taille des nogoods à 2.

Exemple

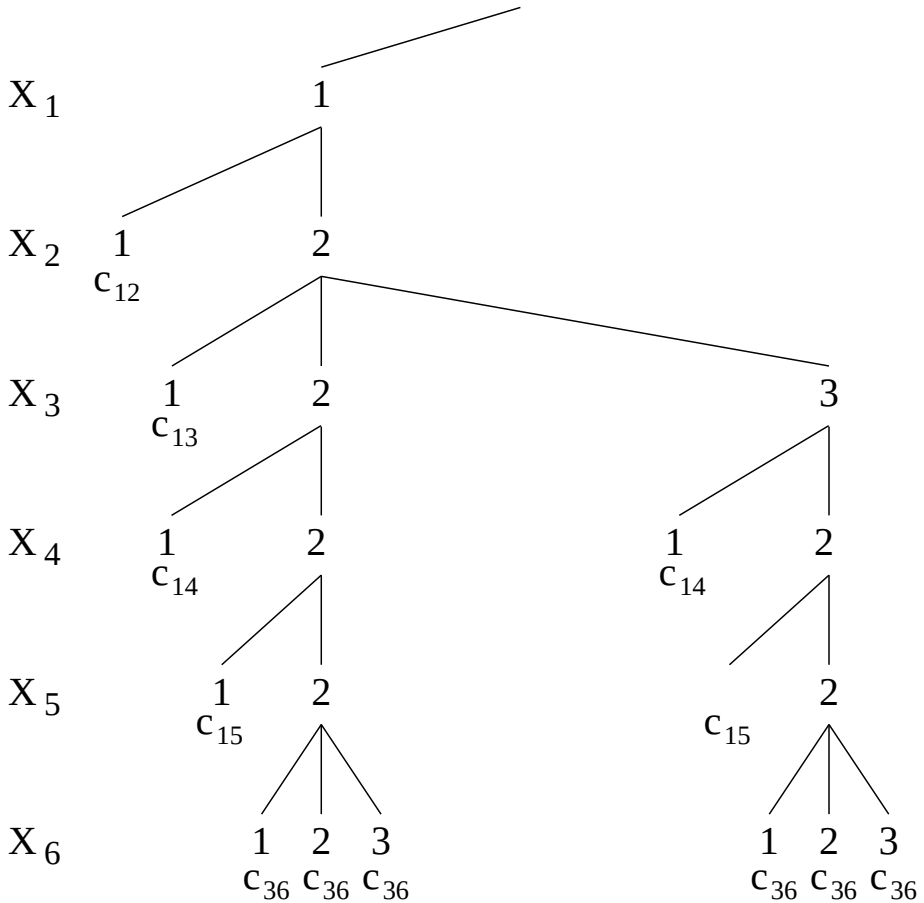


Exemple

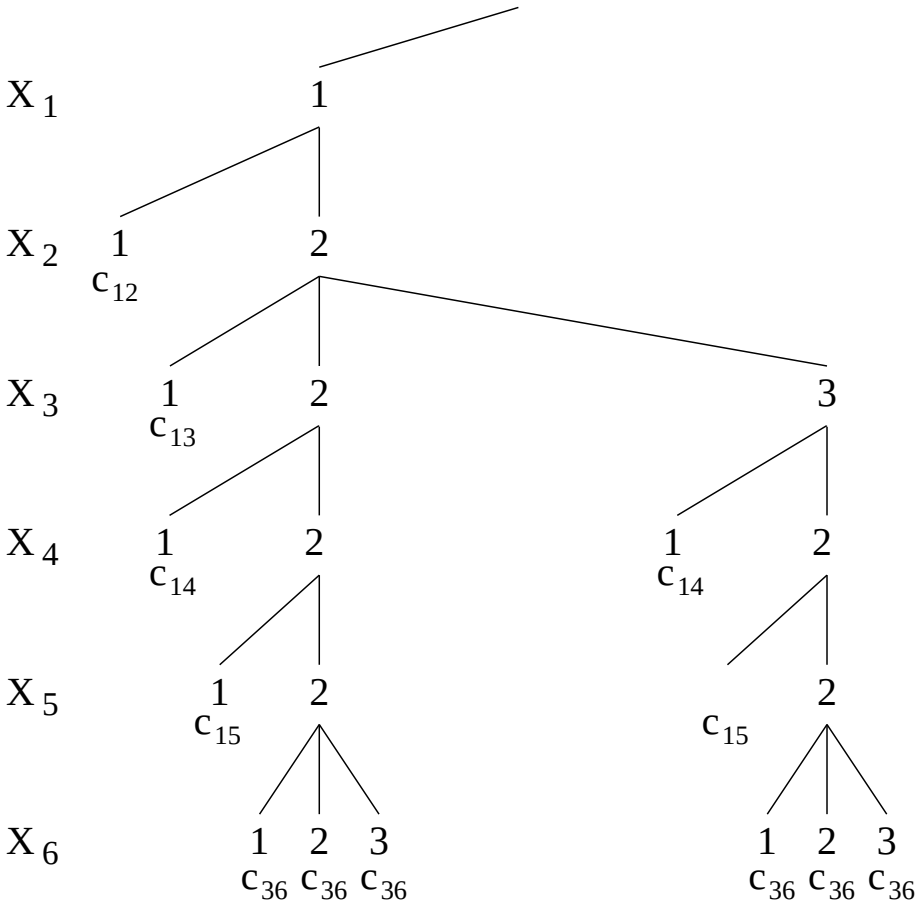


$\Rightarrow \{x_3 \leftarrow 2\}$

Exemple

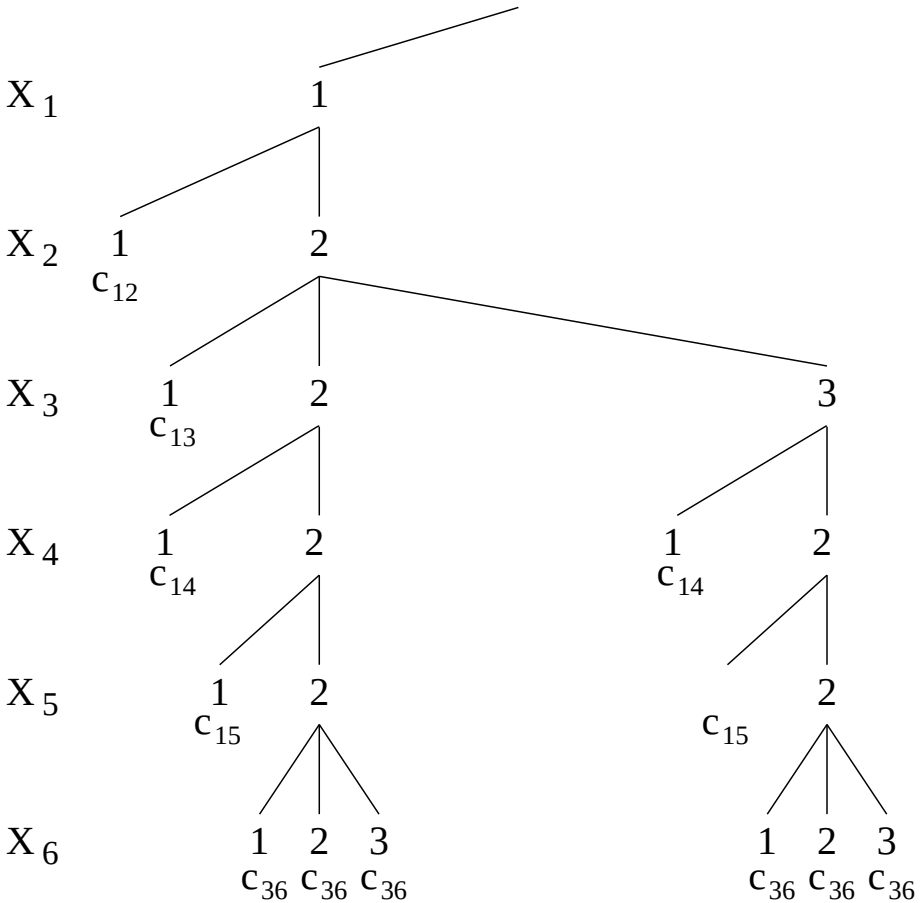

X 1
1
X 2
1
c 12
2
X 3
1
c 13
2
3
X 4
1
c 14
2
1
c 14
2
X 5
1
c 15
2
c 15
2
X 6
1 2 3
c 36 c 36 c 36
1 2 3
c 36 c 36 c 36

Exemple



$\Rightarrow \{x_3 \leftarrow 3\}$

Exemple



$\Rightarrow \{x_1 \leftarrow 1\}$