Problèmes de Satisfaction de Contraintes (suite et fin)

Cyril Terrioux

Laboratoire des Sciences de l'Information et des Systèmes

LSIS - UMR CNRS 6168

Plan

1. Méthodes de résolution énumératives (suite)
2. Méthodes structurelles
3. Méthodes incomplètes
4. Choix d'une méthode

Plan

1. Méthodes de résolution énumératives (suite)
2. Méthodes structurelles
3. Méthodes incomplètes
4. Choix d'une méthode

Améliorer Backtrack

- Analyser les échecs

Améliorer Backtrack

- Analyser les échecs
  - BJ
  - GBJ
  - CBJ

Améliorer Backtrack

- Analyser les échecs
  - BJ
  - GBJ
  - CBJ
- Mémoriser les échecs

Améliorer Backtrack

- Analyser les échecs
  - BJ
  - GBJ
  - CBJ
- Mémoriser les échecs
  - NR

Améliorer Backtrack

- Analyser les échecs
  - BJ
  - GBJ
  - CBJ
- Mémoriser les échecs
  - NR
- Simplifier le problème

Simplifier le problème

Objectif : éviter de développer certaines branches inutiles

Simplifier le problème

Objectif : éviter de développer certaines branches inutiles

branche inutile = branche ne conduisant pas à une solution

Simplifier le problème

Objectif : éviter de développer certaines branches inutiles

branche inutile = branche ne conduisant pas à une solution

On simplifie le problème en supprimant des valeurs inutiles

Simplifier le problème

Objectif : éviter de développer certaines branches inutiles

branche inutile = branche ne conduisant pas à une solution

On simplifie le problème en supprimant des valeurs inutiles

valeur inutile = valeur ne participant pas à une solution

Simplifier le problème

Objectif : éviter de développer certaines branches inutiles

branche inutile = branche ne conduisant pas à une solution

On simplifie le problème en supprimant des valeurs inutiles

valeur inutile = valeur ne participant pas à une solution

Ces méthodes anticipent les échecs immédiats de Backtrack

Simplifier le problème

Objectif : éviter de développer certaines branches inutiles

branche inutile = branche ne conduisant pas à une solution

On simplifie le problème en supprimant des valeurs inutiles

valeur inutile = valeur ne participant pas à une solution

Ces méthodes anticipent les échecs immédiats de Backtrack

L'opération de simplification est appelée **filtrage**.

Forward-Checking (FC)

Un filtrage purement local

Forward-Checking (FC)

Un filtrage purement local

Soit x la variable courante

Principe :

- on filtre les domaines des variables voisines de x en retirant les valeurs incompatibles avec l'affectation de x

Forward-Checking (FC)

Un filtrage purement local

Soit x la variable courante

Principe :

- on filtre les domaines des variables voisines de x en retirant les valeurs incompatibles avec l'affectation de x
- si un domaine devient vide, on essaie une nouvelle valeur pour x

Forward-Checking (FC)

Un filtrage purement local

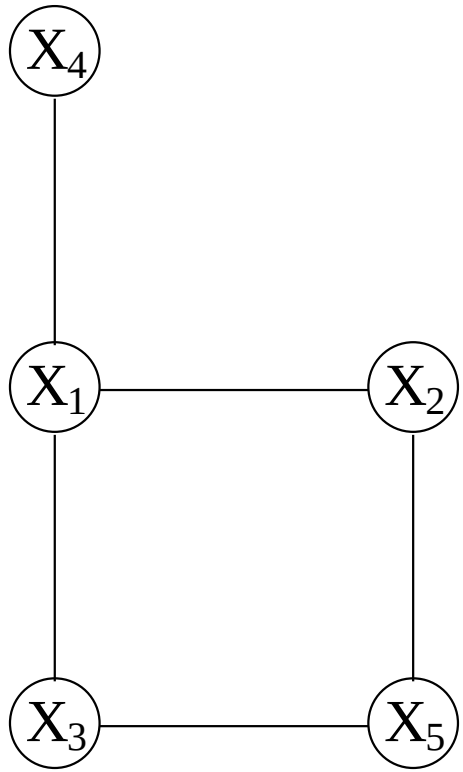
Soit x la variable courante

Principe :

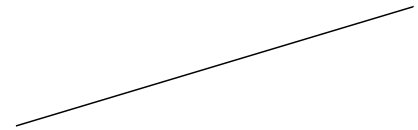
- on filtre les domaines des variables voisines de x en retirant les valeurs incompatibles avec l'affectation de x
- si un domaine devient vide, on essaie une nouvelle valeur pour x
- si toutes les valeurs ont été essayées, on revient en arrière sur la variable précédente

Exemple

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



Exemple

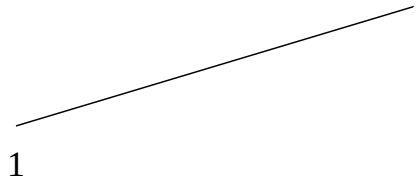


X_1 1

X_2

X_3

Exemple

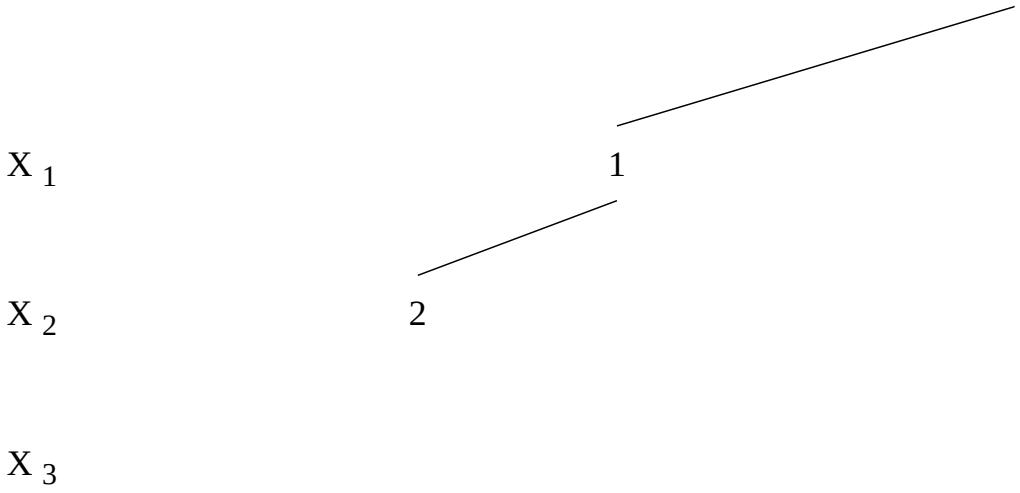


$d_{x_2} = \{2, 3\}$

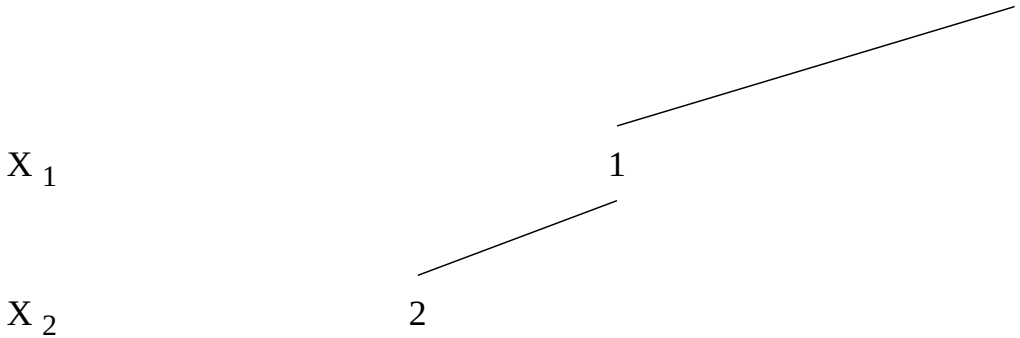
$d_{x_3} = \{2\}$

$d_{x_4} = \{2, 3\}$

Exemple

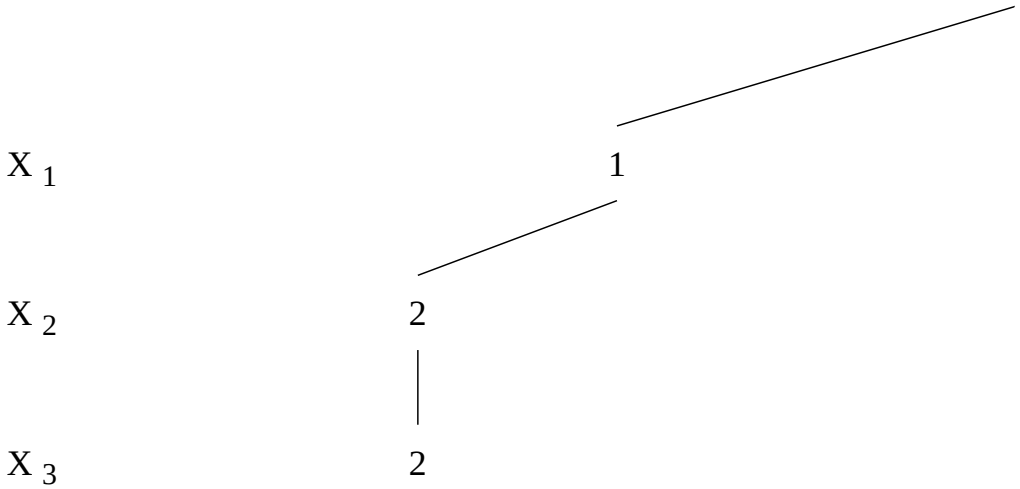


Exemple



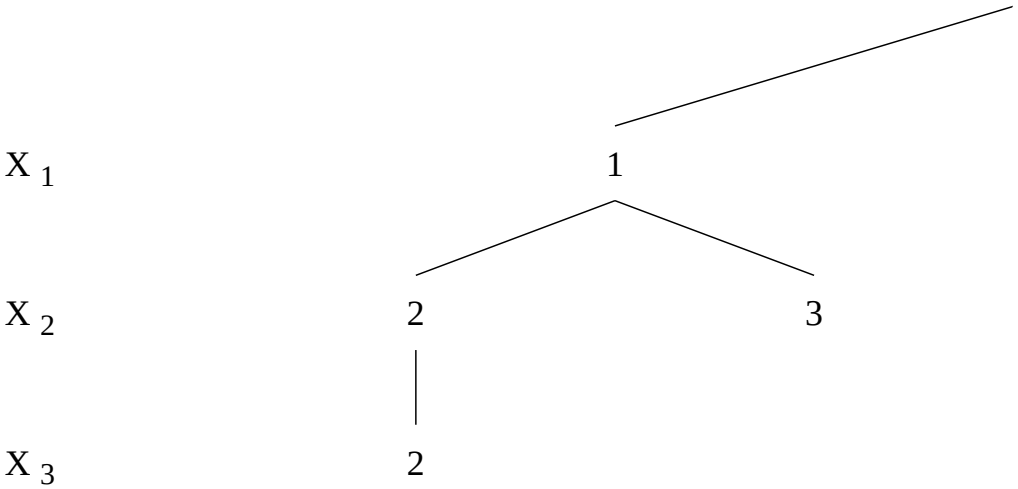
$d_{x_5} = \{3\}$

Exemple



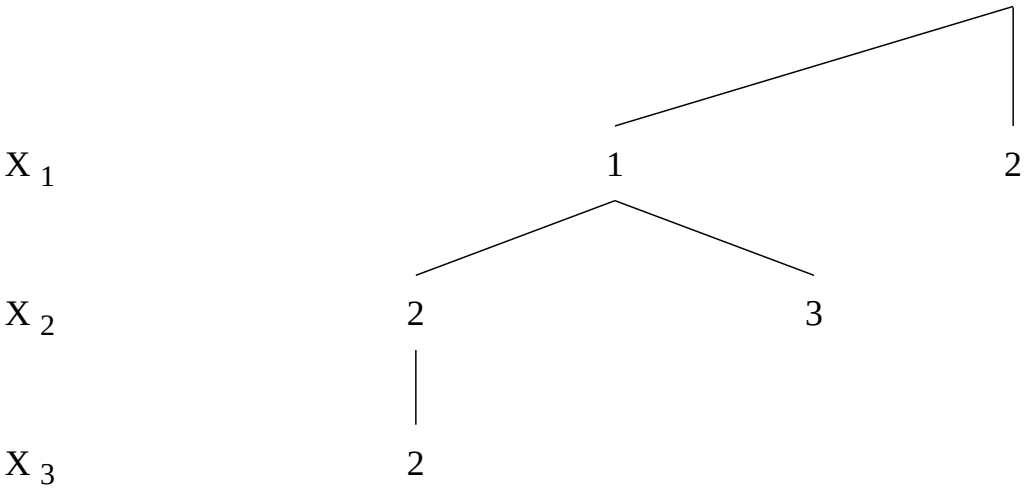
$d_{x_5} = \emptyset$

Exemple



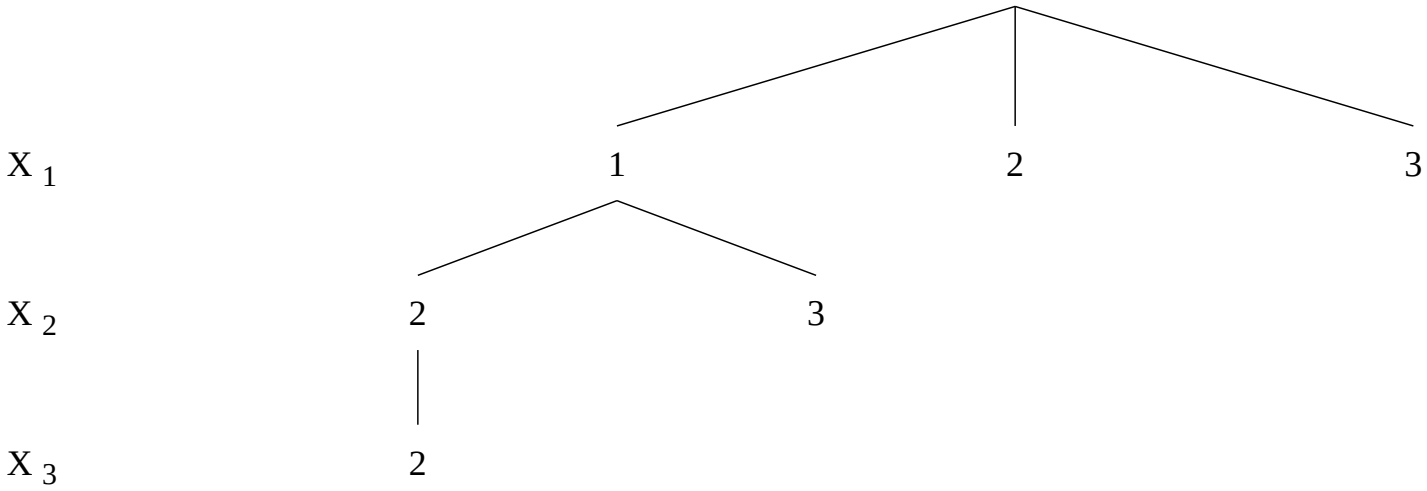
$d_{x_5} = \emptyset$

Exemple



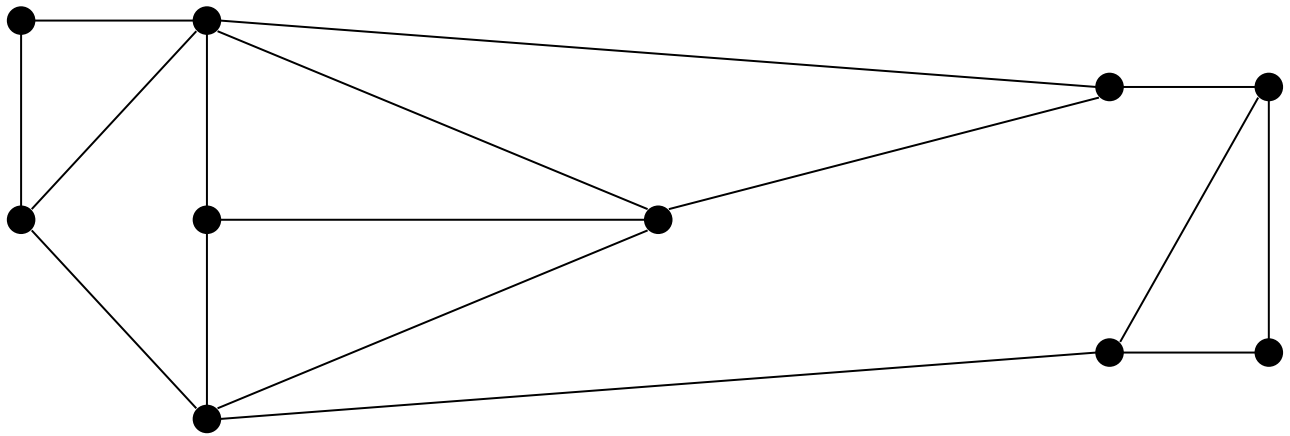
$d_{x_3} = \emptyset$

Exemple

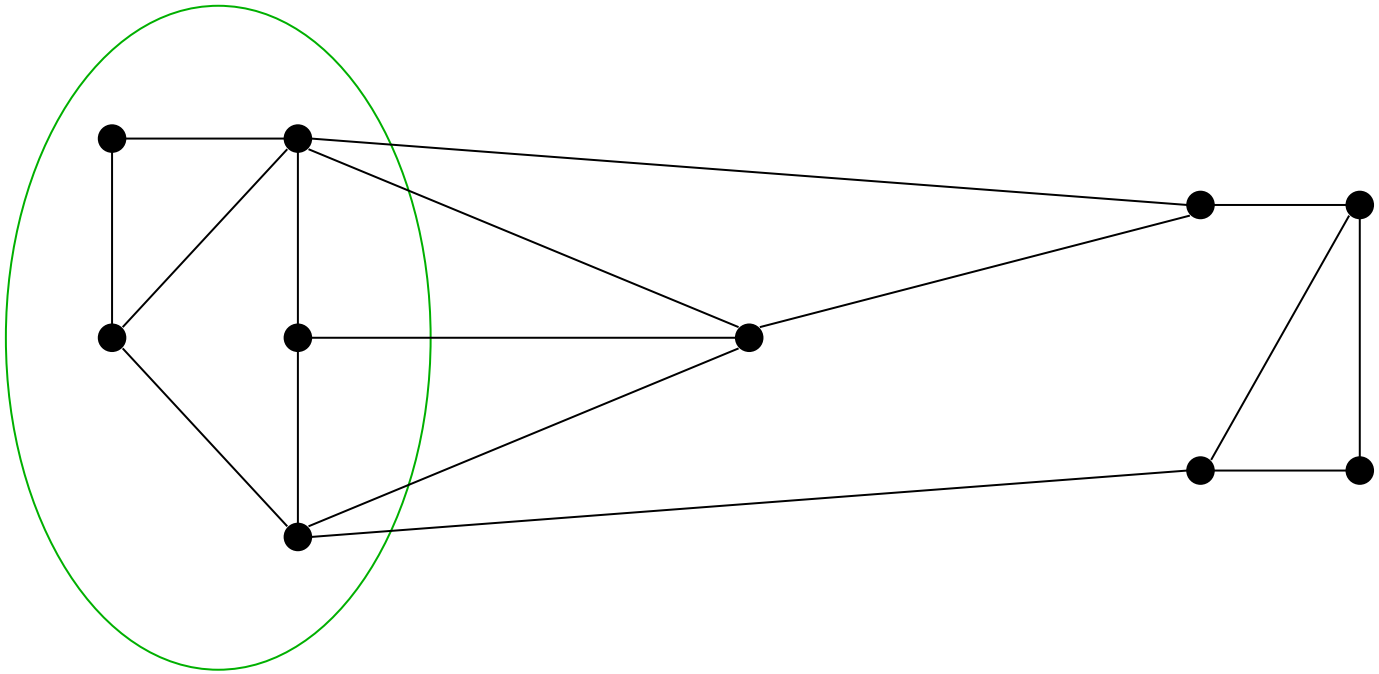


$d_{x_3} = \emptyset$

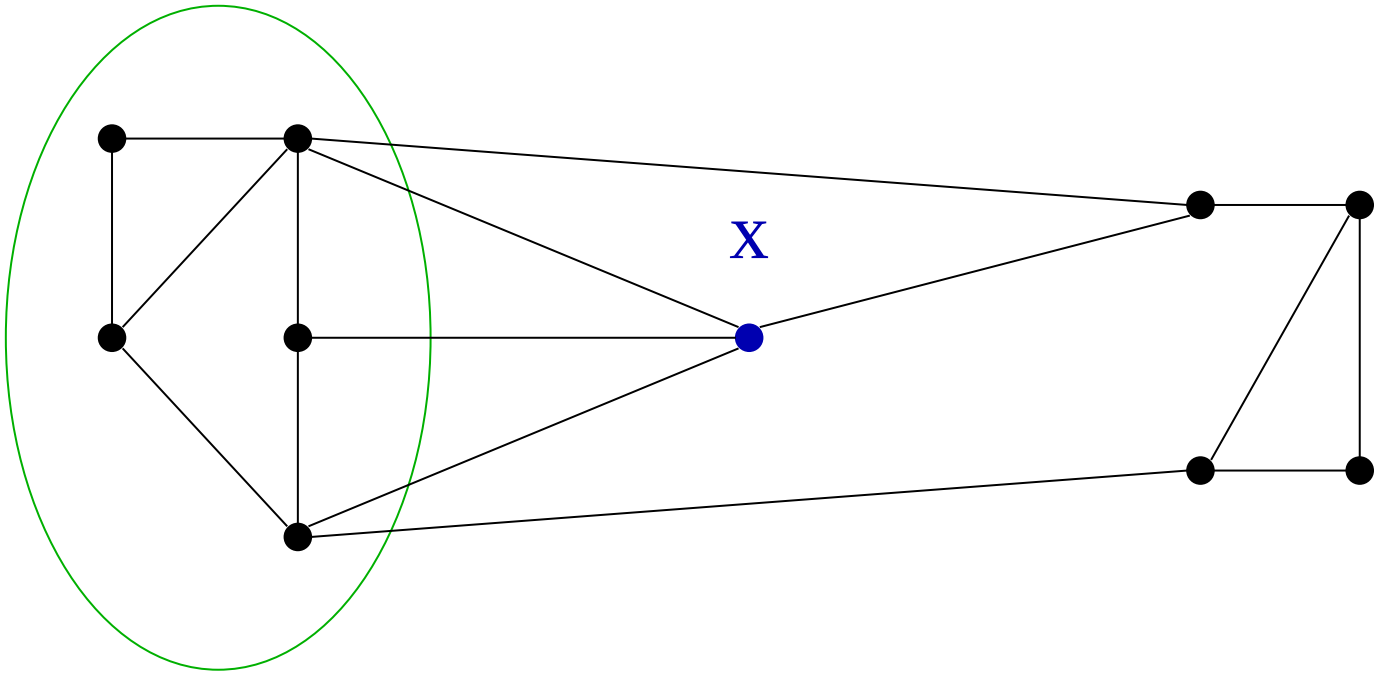
Comparaisons des tests de consistance



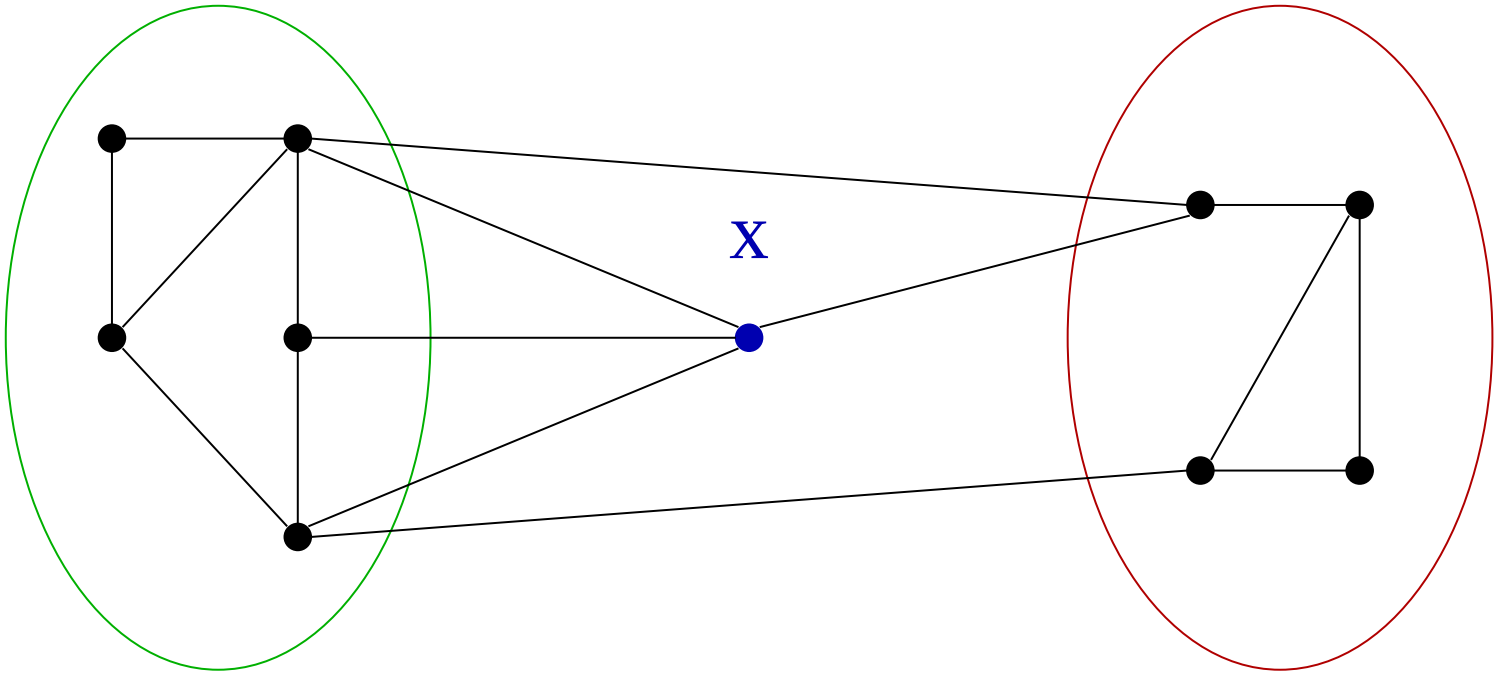
Comparaisons des tests de consistance



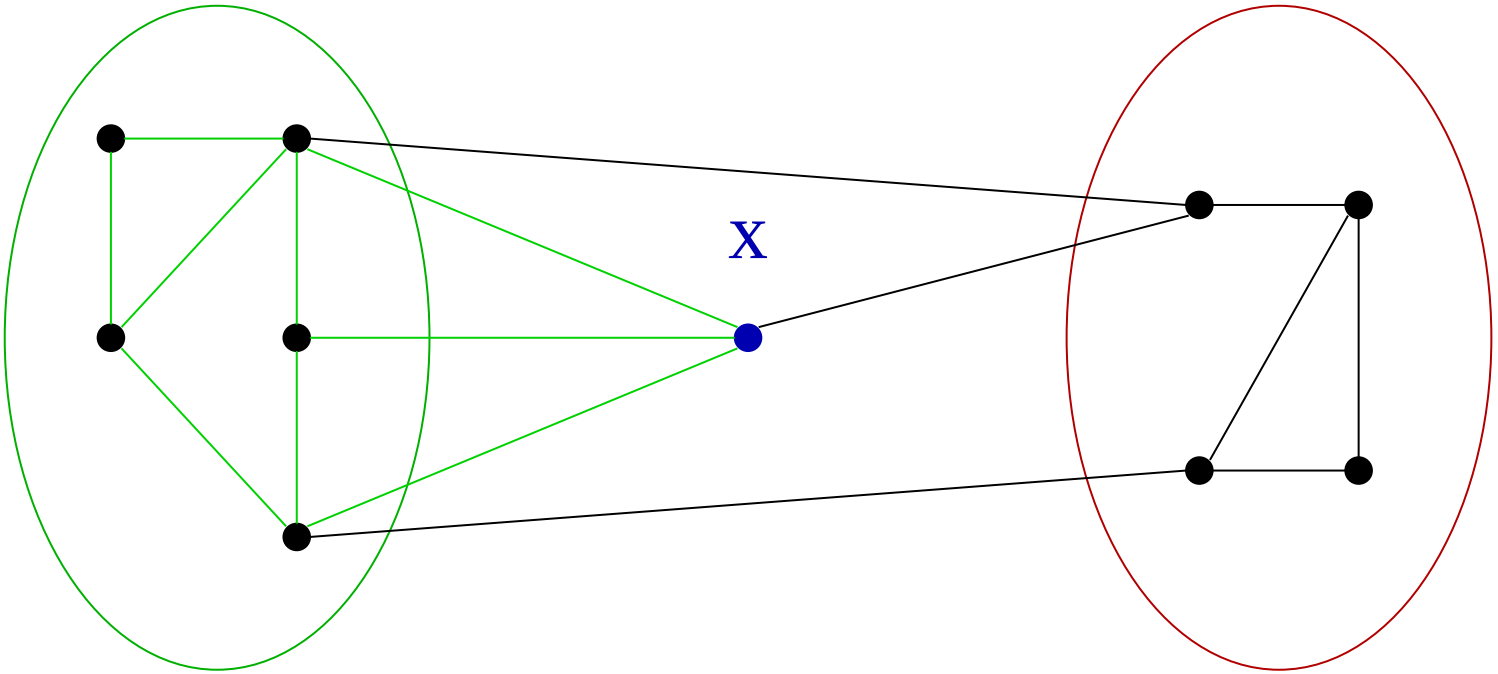
Comparaisons des tests de consistance



Comparaisons des tests de consistance

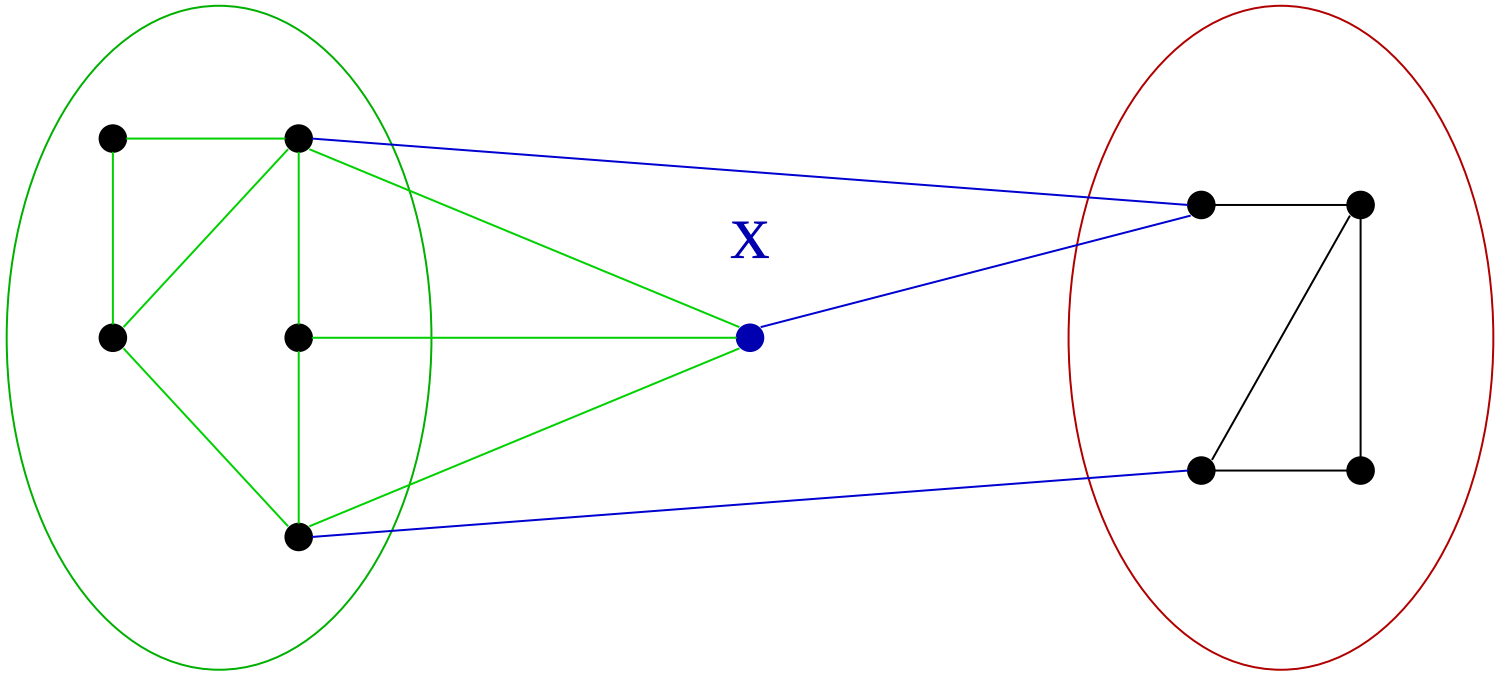


Comparaisons des tests de consistance



Backtrack

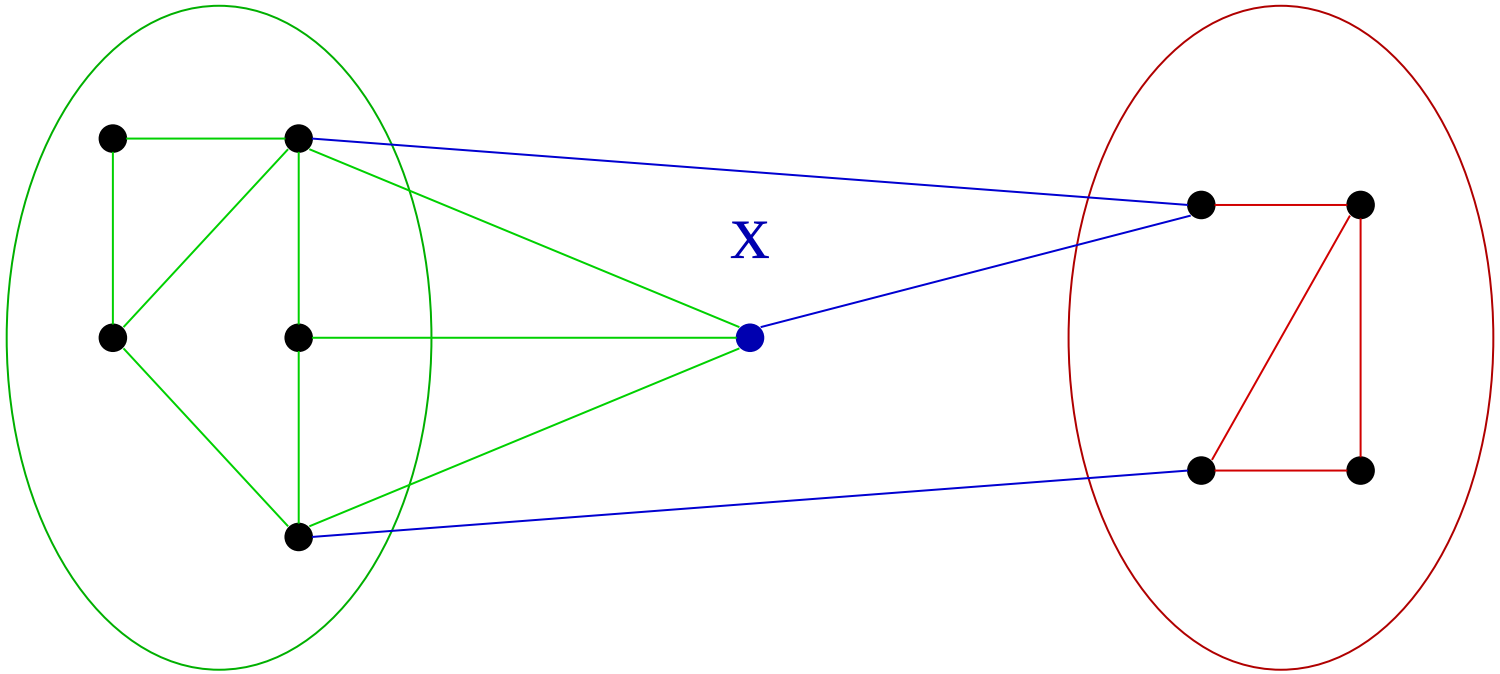
Comparaisons des tests de consistance



Backtrack

Forward-Checking

Comparaisons des tests de consistance



Backtrack

Forward-Checking

??

Consistance d'Arc

Un domaine d_x est **arc-consistant** si :

- $d_x \neq \emptyset$ et
- $\forall v \in d_x, \forall c = \{x, y\} \in C, \exists w \in d_y, (v, w) \in r_c$.

Consistance d'Arc

Un domaine d_x est **arc-consistant** si :

- $d_x \neq \emptyset$ et
- $\forall v \in d_x, \forall c = \{x, y\} \in C, \exists w \in d_y, (v, w) \in r_c$.

Un problème est dit **arc-consistant** si tous ses domaines sont arc-consistants.

Consistance d'Arc

Un domaine d_x est **arc-consistant** si :

- $d_x \neq \emptyset$ et
- $\forall v \in d_x, \forall c = \{x, y\} \in C, \exists w \in d_y, (v, w) \in r_c$.

Un problème est dit **arc-consistant** si tous ses domaines sont arc-consistants.

Filtrage par consistance d'arc = suppression des valeurs qui rendent un domaine arc-inconsistant

Consistance d'Arc

Un domaine d_x est **arc-consistant** si :

- $d_x \neq \emptyset$ et
- $\forall v \in d_x, \forall c = \{x, y\} \in C, \exists w \in d_y, (v, w) \in r_c$.

Un problème est dit **arc-consistant** si tous ses domaines sont arc-consistants.

Filtrage par consistance d'arc = suppression des valeurs qui rendent un domaine arc-inconsistant

Complexité en temps : $O(md^2)$

Exemple

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

Exemple

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

On supprime 3 de d_{x_1}

Exemple

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

On supprime 3 de d_{x_1}

On supprime 1 de d_{x_2}

Exemple

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

On supprime 3 de d_{x_1}

On supprime 1 de d_{x_2}

On supprime 1 de d_{x_3}

Exemple

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

On supprime 3 de d_{x_1}

On supprime 1 de d_{x_2}

On supprime 1 de d_{x_3}

On supprime 1 de d_{x_4}

Exemple

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

On supprime 3 de d_{x_1}

On supprime 1 de d_{x_2}

On supprime 1 de d_{x_3}

On supprime 1 de d_{x_4}

On supprime 1 de d_{x_5}

Exemple

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

On supprime 3 de d_{x_1}

On supprime 1 de d_{x_2}

On supprime 1 de d_{x_3}

On supprime 1 de d_{x_4}

On supprime 1 de d_{x_5}

On supprime 2 de d_{x_3}

Exemple

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

On supprime 3 de d_{x_1}

On supprime 1 de d_{x_2}

On supprime 1 de d_{x_3}

On supprime 1 de d_{x_4}

On supprime 1 de d_{x_5}

On supprime 2 de d_{x_3}

$\Rightarrow d_{x_3} = \emptyset$

Maintien de l'Arc-Consistance (MAC)

Filtrage par consistance d'arc

Maintien de l'Arc-Consistance (MAC)

Filtrage par consistance d'arc

Soit x la variable courante

Principe :

- on rend arc-consistants les domaines des variables non instanciées en retirant les valeurs incompatibles avec l'affectation de x

Maintien de l'Arc-Consistance (MAC)

Filtrage par consistance d'arc

Soit x la variable courante

Principe :

- on rend arc-consistants les domaines des variables non instanciées en retirant les valeurs incompatibles avec l'affectation de x
- si un domaine devient vide, on essaie une nouvelle valeur pour x

Maintien de l'Arc-Consistance (MAC)

Filtrage par consistance d'arc

Soit x la variable courante

Principe :

- on rend arc-consistants les domaines des variables non instanciées en retirant les valeurs incompatibles avec l'affectation de x
- si un domaine devient vide, on essaie une nouvelle valeur pour x
- si toutes les valeurs ont été essayées, on revient en arrière sur la variable précédente

Comparaison pratique entre FC et MAC

FC développe généralement plus de nœuds que MAC.

Comparaison pratique entre FC et MAC

FC développe généralement plus de nœuds que MAC.

Mais, le coût d'un nœud est généralement plus important avec MAC.

Comparaison pratique entre FC et MAC

FC développe généralement plus de nœuds que MAC.

Mais, le coût d'un nœud est généralement plus important avec MAC.

Car, MAC effectue plus de tests de contraintes en chaque nœud.

Comparaison pratique entre FC et MAC

FC développe généralement plus de nœuds que MAC.

Mais, le coût d'un nœud est généralement plus important avec MAC.

Car, MAC effectue plus de tests de contraintes en chaque nœud.

En temps, le meilleur est tantôt FC, tantôt MAC !!!

Passage aux CSP n-aires

BT et MAC n’ont qu’une seule généralisation.

Passage aux CSP n-aires

BT et MAC n’ont qu’une seule généralisation.

FC en possède plusieurs.

Une vision différente de FC

Une forme allégée de consistance d’arc

Une vision différente de FC

Une forme allégée de consistance d'arc

Exploitation des contraintes liant une variable instanciée à une non instanciée

FC n-aire

Plusieurs généralisations :

- selon les contraintes considérées
- selon le type de propagation

FC n-aire

Plusieurs généralisations :

- selon les contraintes considérées
- selon le type de propagation

nFC0 :

- exploitation des contraintes liant au moins la variable courante à une seule variable non instanciée
- application de la consistance d'arc en une seule passe

FC n-aire

nFC2 :

- exploitation des contraintes liant au moins la variable courante à au moins une variable non instanciée
- application de la consistance d’arc en une seule passe

FC n-aire

nFC2 :

- exploitation des contraintes liant au moins la variable courante à au moins une variable non instanciée
- application de la consistance d'arc en une seule passe

nFC5 :

- exploitation des contraintes liant au moins une variable instanciée à au moins une variable non instanciée
- application de la consistance d'arc jusqu'à l'obtention du point fixe

Hybridation

Méthodes hybrides mixant :

- analyse des échecs
- mémorisation des échecs
- simplification du problème

Hybridation

Méthodes hybrides mixant :

- analyse des échecs
- mémorisation des échecs
- simplification du problème

Attention, ces approches ne sont pas complémentaires !

Plan

1. Méthodes de résolution énumératives (suite)
2. Méthodes structurelles
3. Méthodes incomplètes
4. Choix d’une méthode

Méthodes structurelles

Structure d’un CSP = son graphe de contraintes

Méthodes structurelles

Structure d'un CSP = son graphe de contraintes

Propriété :

Un CSP dont le graphe de contraintes est un arbre peut être résolu en temps polynomial.

Méthodes structurelles

Structure d'un CSP = son graphe de contraintes

Propriété :

Un CSP dont le graphe de contraintes est un arbre peut être résolu en temps polynomial.

Idée : essayer de se ramener à un CSP arborescent

Regroupement en arbre (TC)

On construit un nouveau CSP.

Regroupement en arbre (TC)

On construit un nouveau CSP.

On regroupe les variables entre elles.

Regroupement en arbre (TC)

On construit un nouveau CSP.

On regroupe les variables entre elles.

Une variable du nouveau CSP = un regroupement de variables

Regroupement en arbre (TC)

On construit un nouveau CSP.

On regroupe les variables entre elles.

Une variable du nouveau CSP = un regroupement de variables

Domaine de chaque variable = l'ensemble des solutions du regroupement

Regroupement en arbre (TC)

On construit un nouveau CSP.

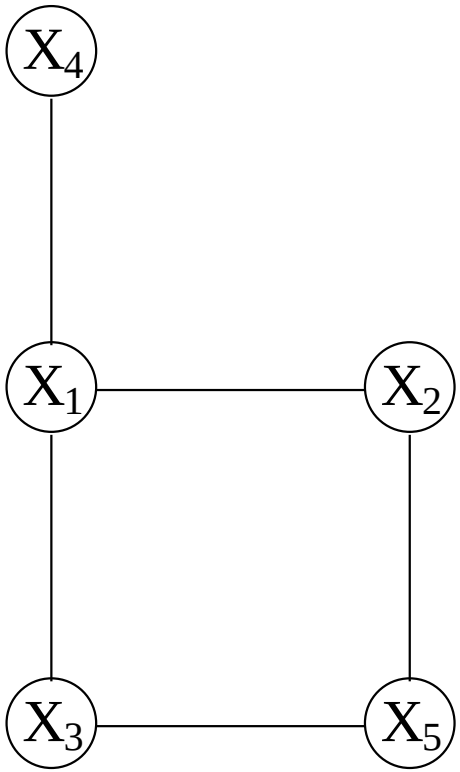
On regroupe les variables entre elles.

Une variable du nouveau CSP = un regroupement de variables

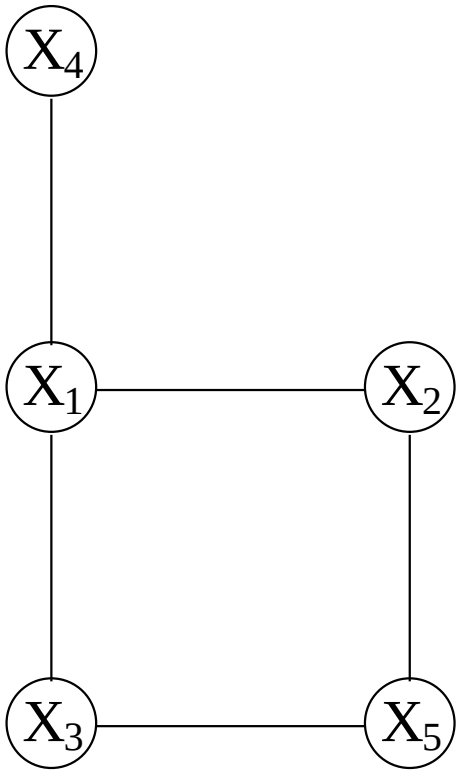
Domaine de chaque variable = l'ensemble des solutions du regroupement

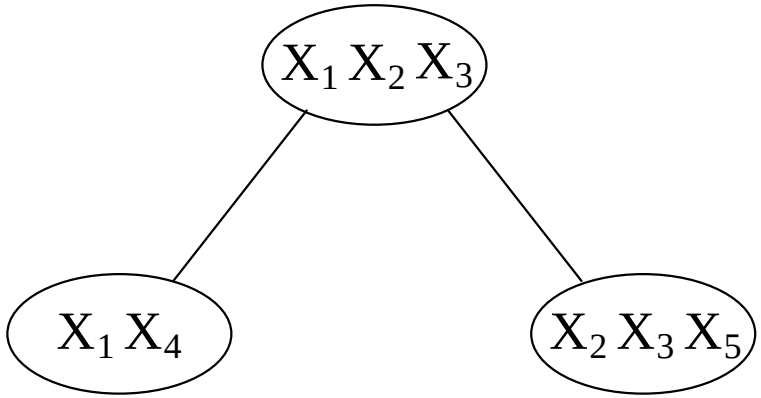
Complexité en temps en $O(nd^r)$

Exemple

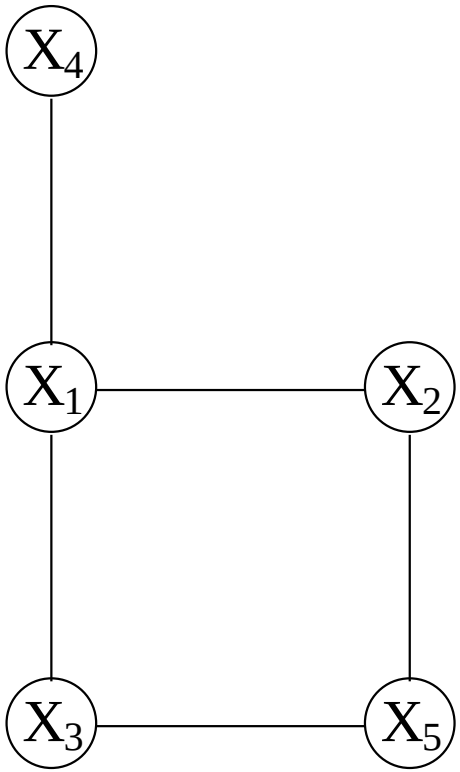


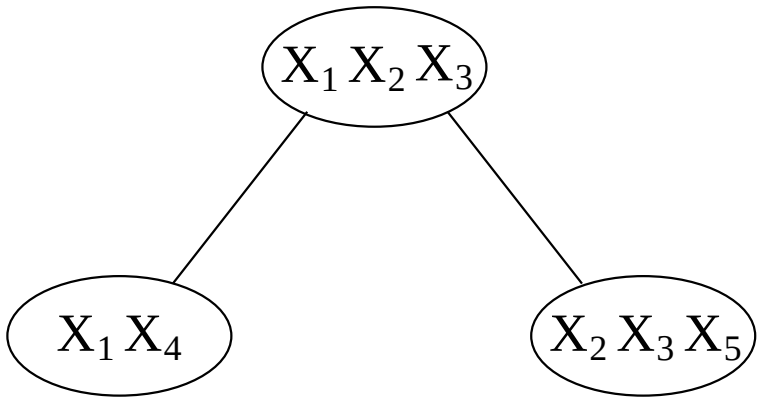
Exemple





Exemple





$r = 3$

Méthode Coupe-Cycle

Ensemble coupe-cycle = ensemble de sommets qui une fois retirés rendent le graphe arborescent

Méthode Coupe-Cycle

Ensemble coupe-cycle = ensemble de sommets qui une fois retirés rendent le graphe arborescent

Principe :

- on calcule un ensemble coupe-cycle

Méthode Coupe-Cycle

Ensemble coupe-cycle = ensemble de sommets qui une fois retirés rendent le graphe arborescent

Principe :

- on calcule un ensemble coupe-cycle
- on cherche les solutions du coupe-cycle

Méthode Coupe-Cycle

Ensemble coupe-cycle = ensemble de sommets qui une fois retirés rendent le graphe arborescent

Principe :

- on calcule un ensemble coupe-cycle
- on cherche les solutions du coupe-cycle
- on teste si on peut étendre une solution du coupe cycle sur le sous-problème arborescent

Méthode Coupe-Cycle

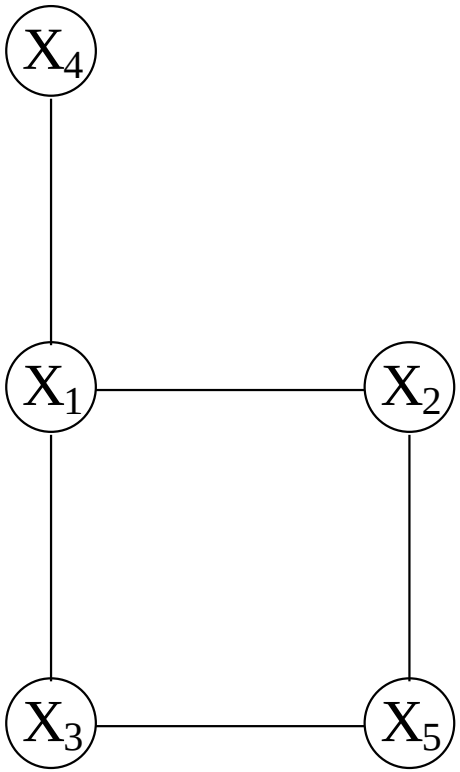
Ensemble coupe-cycle = ensemble de sommets qui une fois retirés rendent le graphe arborescent

Principe :

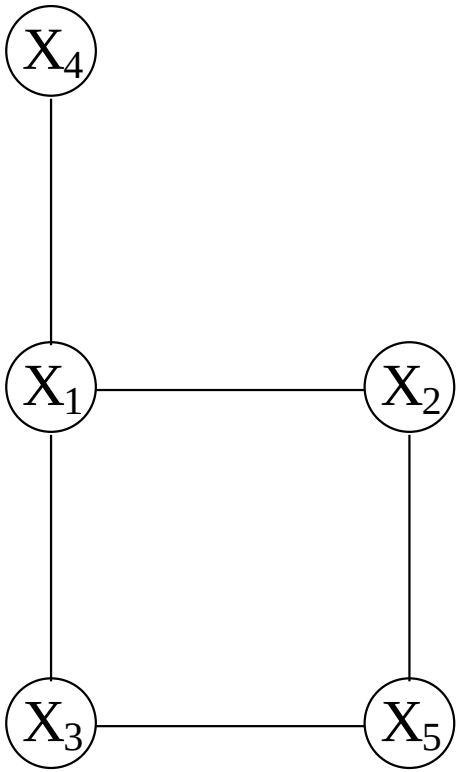
- on calcule un ensemble coupe-cycle
- on cherche les solutions du coupe-cycle
- on teste si on peut étendre une solution du coupe cycle sur le sous-problème arborescent

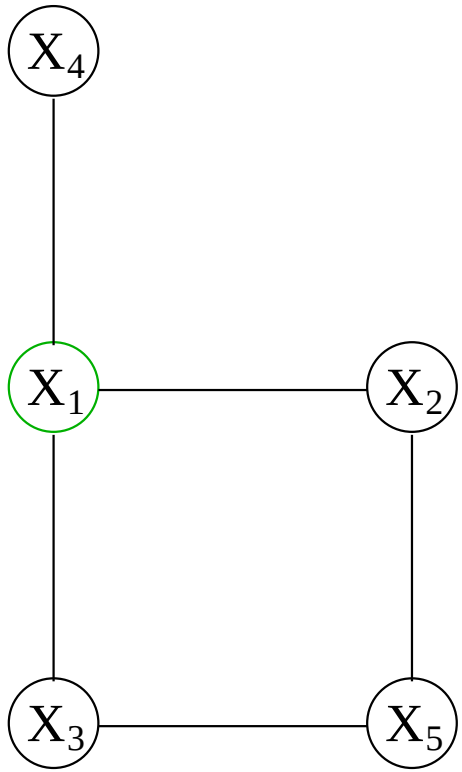
Complexité en temps en $O(nd^{c+2})$ avec c la taille du coupe-cycle

Exemple

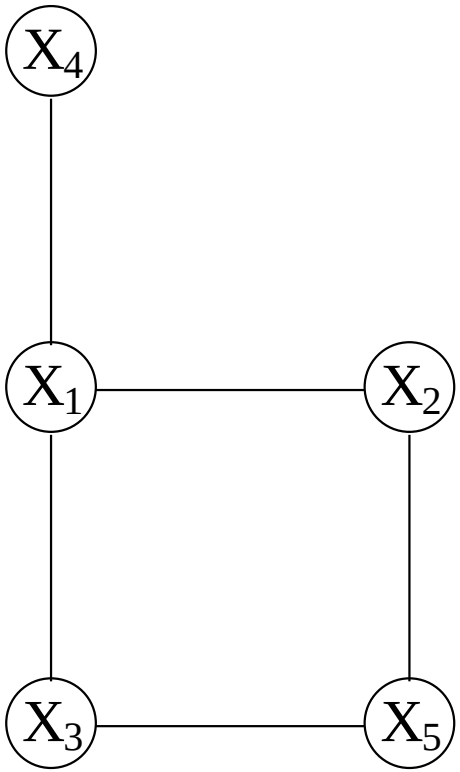


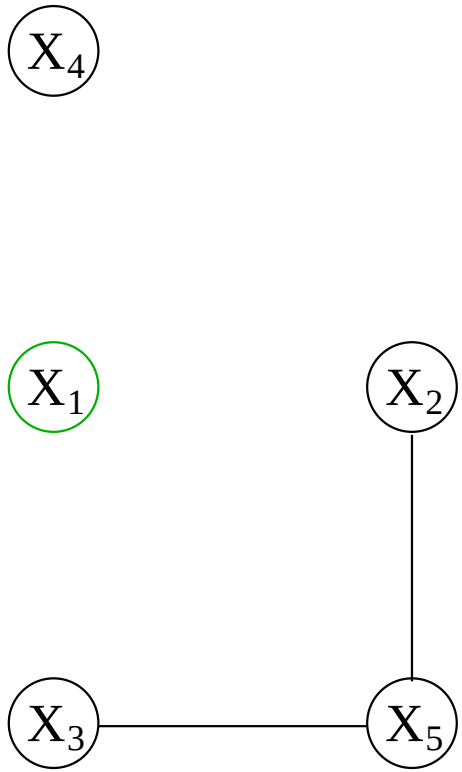
Exemple



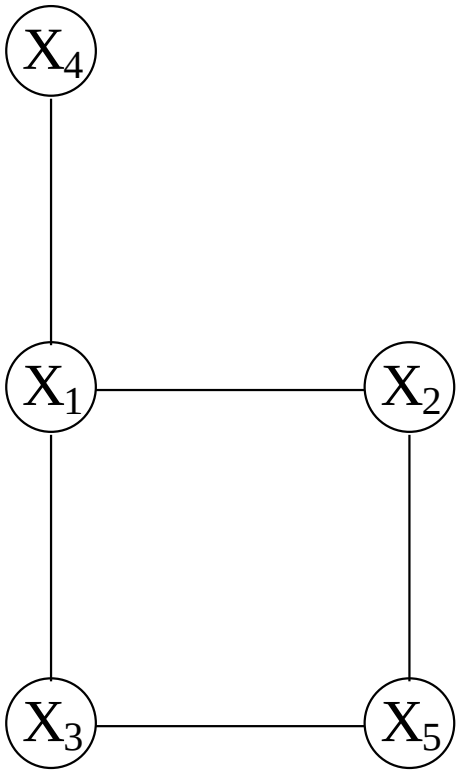


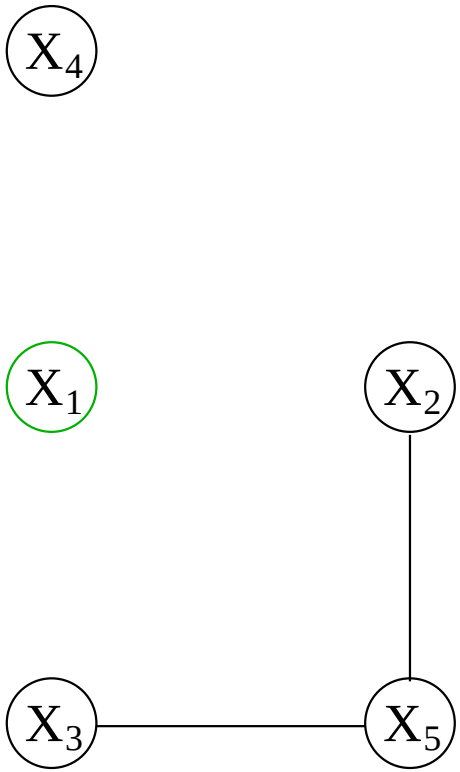
Exemple





Exemple





$c = 1$

Plan

1. Méthodes de résolution énumératives (suite)
2. Méthodes structurelles
3. Méthodes incomplètes
4. Choix d'une méthode

Les méthodes incomplètes

Effectuer une exploration non systématique

Les méthodes incomplètes

Effectuer une exploration non systématique

Principe :

- On part d'une première instanciation

Les méthodes incomplètes

Effectuer une exploration non systématique

Principe :

- On part d'une première instanciation
- A chaque étape, on choisit une nouvelle instanciation parmi toutes celles voisines de l'affectation courante

Les méthodes incomplètes

Effectuer une exploration non systématique

Principe :

- On part d'une première instanciation
- A chaque étape, on choisit une nouvelle instanciation parmi toutes celles voisines de l'affectation courante

Affectation voisine de $\mathcal{A}$: une affectation dont la valeur d'une variable diffère par rapport à $\mathcal{A}$

Les méthodes incomplètes

Effectuer une exploration non systématique

Principe :

- On part d'une première instanciation
- A chaque étape, on choisit une nouvelle instanciation parmi toutes celles voisines de l'affectation courante

Affectation voisine de $\mathcal{A}$: une affectation dont la valeur d'une variable diffère par rapport à $\mathcal{A}$

Le choix est souvent guidé par un critère de proximité

Les méthodes incomplètes

Effectuer une exploration non systématique

Principe :

- On part d'une première instanciation
- A chaque étape, on choisit une nouvelle instanciation parmi toutes celles voisines de l'affectation courante

Affectation voisine de $\mathcal{A}$: une affectation dont la valeur d'une variable diffère par rapport à $\mathcal{A}$

Le choix est souvent guidé par un critère de proximité

Recherche locale ou par tâtonnement

Les méthodes incomplètes

$\mathcal{A} \leftarrow \mathcal{A}_{init}$
TantQue $\mathcal{A}$ n'est pas une solution
    choisir $\mathcal{A}'$ parmi les voisines de $\mathcal{A}$
    $\mathcal{A} \leftarrow \mathcal{A}'$
FinTantQue

Les méthodes incomplètes

$\mathcal{A} \leftarrow \mathcal{A}_{init}$
TantQue $\mathcal{A}$ n'est pas une solution
    choisir $\mathcal{A}'$ parmi les voisines de $\mathcal{A}$
    $\mathcal{A} \leftarrow \mathcal{A}'$
FinTantQue

La façon de choisir la prochaine affectation caractérise la méthode.

Les méthodes incomplètes

Avantages :

- simples à implémenter

Les méthodes incomplètes

Avantages :

- simples à implémenter
- on peut rapidement trouver une solution

Les méthodes incomplètes

Avantages :

- simples à implémenter
- on peut rapidement trouver une solution

Inconvénients :

- on peut ne jamais atteindre une solution

Les méthodes incomplètes

Avantages :

- simples à implémenter
- on peut rapidement trouver une solution

Inconvénients :

- on peut ne jamais atteindre une solution
- on peut boucler à l'infini

Les méthodes incomplètes

Avantages :

- simples à implémenter
- on peut rapidement trouver une solution

Inconvénients :

- on peut ne jamais atteindre une solution
- on peut boucler à l'infini
- on ne peut pas conclure qu'il n'existe pas de solution

L’algorithme du British Museum

Choix aléatoire de la prochaine affectation parmi les affectations voisines

L’algorithme du British Museum

Choix aléatoire de la prochaine affectation parmi les affectations voisines

Si on rencontre une impasse, on repart d’une nouvelle affectation initiale

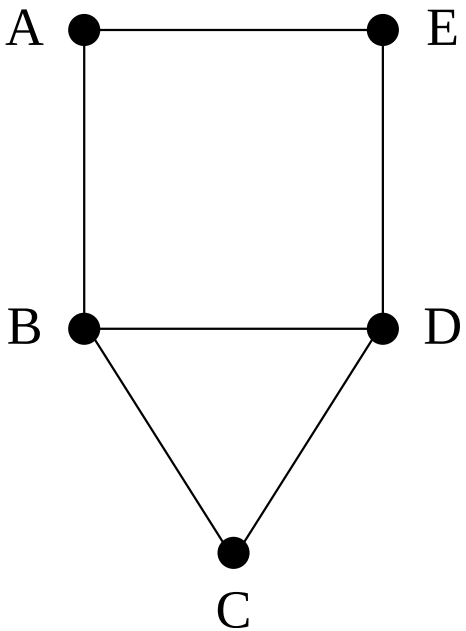
L’algorithme du British Museum

Choix aléatoire de la prochaine affectation parmi les affectations voisines

Si on rencontre une impasse, on repart d’une nouvelle affectation initiale

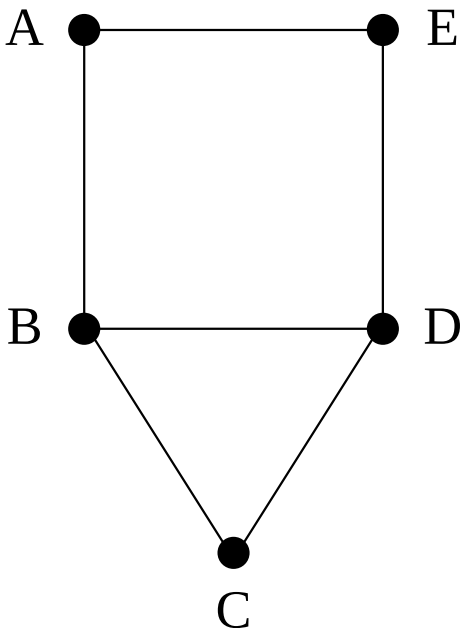
Très inefficace en pratique

Exemple



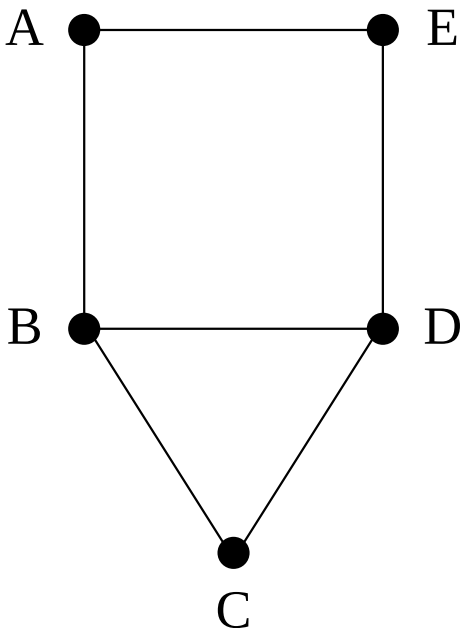
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$

Exemple



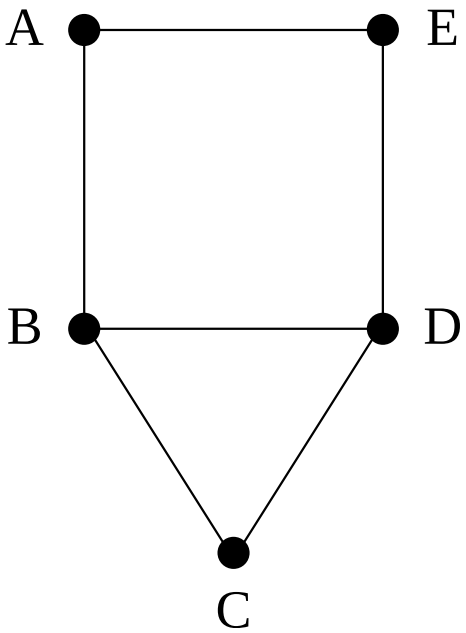
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$

Exemple



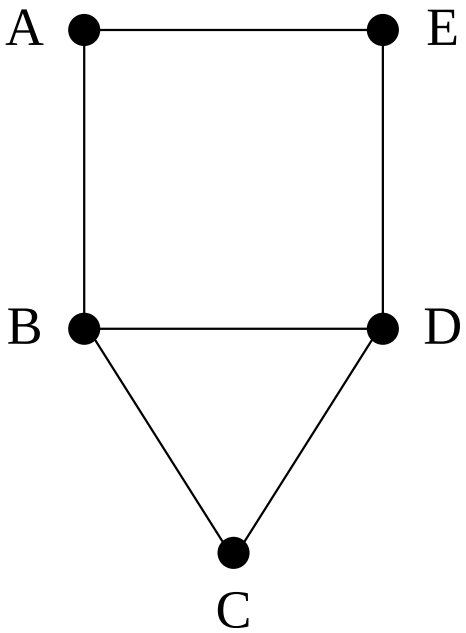
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$

Exemple



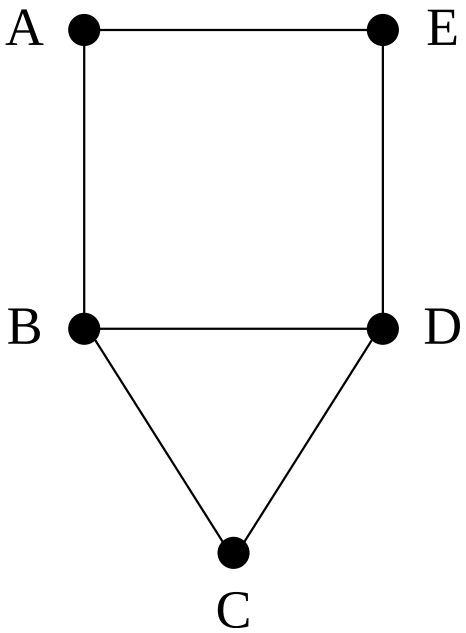
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow r, E \leftarrow r\}$$

Exemple



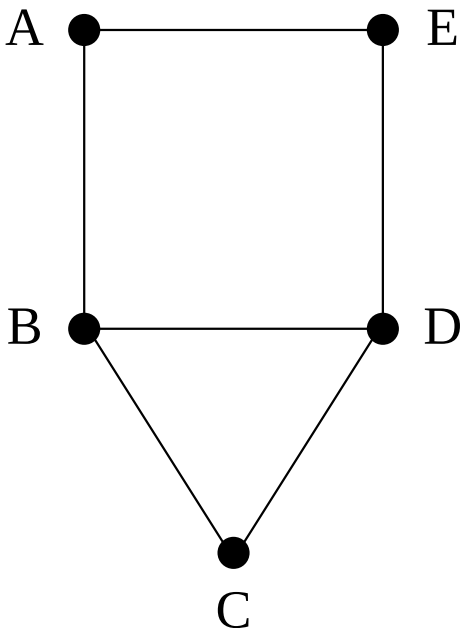
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$

Exemple



$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow v, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$

Exemple



$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow r, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\{A \leftarrow v, B \leftarrow v, C \leftarrow j, D \leftarrow v, E \leftarrow r\}$$
$$\ldots$$

Le Hill-Climbing (ou escalade)

Choix de la prochaine affectation :
une de celles qui améliorent le critère de proximité

Le Hill-Climbing (ou escalade)

Choix de la prochaine affectation :
une de celles qui améliorent le critère de proximité

Critère de proximité = une heuristique estimant
l'éloignement de l'affectation courante d'une solution

Le Hill-Climbing (ou escalade)

Choix de la prochaine affectation :
une de celles qui améliorent le critère de proximité

Critère de proximité = une heuristique estimant
l'éloignement de l'affectation courante d'une solution

Exemple : nombre de contraintes violées

Le Hill-Climbing (ou escalade)

Choix de la prochaine affectation :
une de celles qui améliorent le critère de proximité

Critère de proximité = une heuristique estimant
l'éloignement de l'affectation courante d'une solution

Exemple : nombre de contraintes violées

Inconvénient : la méthode peut atteindre un optimum local

Le Hill-Climbing (ou escalade)

Choix de la prochaine affectation :
une de celles qui améliorent le critère de proximité

Critère de proximité = une heuristique estimant l'éloignement de l'affectation courante d'une solution

Exemple : nombre de contraintes violées

Inconvénient : la méthode peut atteindre un optimum local

optimum local = affectation pour laquelle aucune affectation voisine n'améliore le critère

Le Hill-Climbing (ou escalade)

Choix de la prochaine affectation :
une de celles qui améliorent le critère de proximité

Critère de proximité = une heuristique estimant l'éloignement de l'affectation courante d'une solution

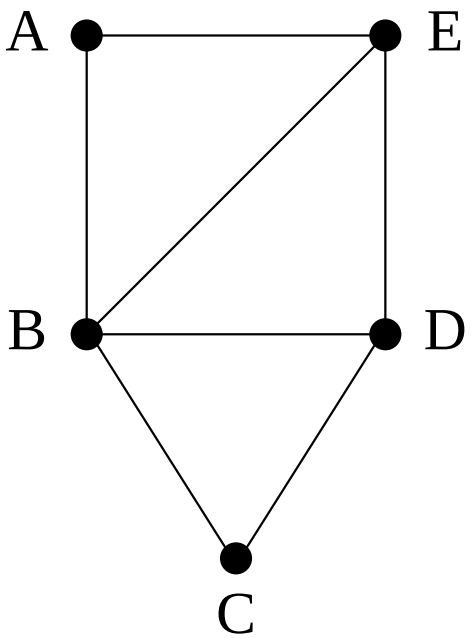
Exemple : nombre de contraintes violées

Inconvénient : la méthode peut atteindre un optimum local

optimum local = affectation pour laquelle aucune affectation voisine n'améliore le critère

Efficacité très variable suivant le nombre d'optima locaux

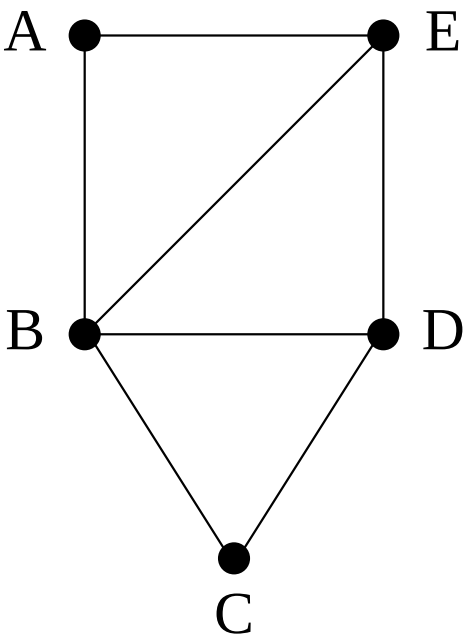
Exemple



$$\{A \leftarrow r, B \leftarrow r, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$

Critère : minimiser le nombre de contraintes violées

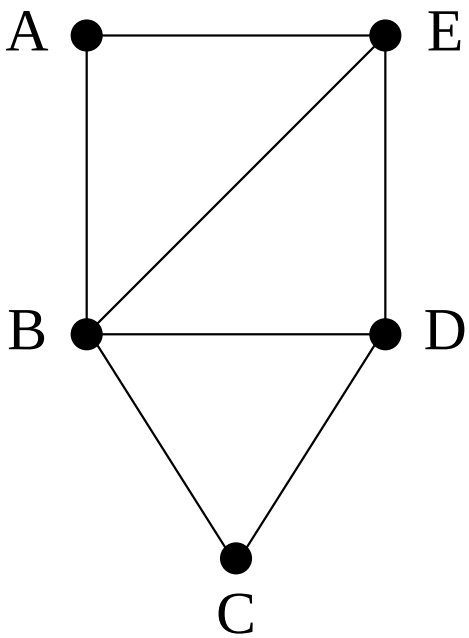
Exemple



$$\{A \leftarrow r, B \leftarrow r, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$

Critère : minimiser le nombre de contraintes violées

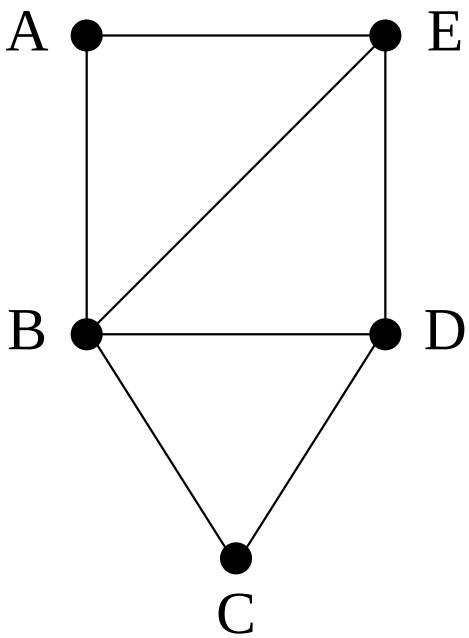
Exemple



$$\{A \leftarrow r, B \leftarrow r, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow r, D \leftarrow r, E \leftarrow j\}$$

Critère : minimiser le nombre de contraintes violées

Exemple



$$\{A \leftarrow r, B \leftarrow r, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow r, D \leftarrow r, E \leftarrow r\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow r, D \leftarrow r, E \leftarrow j\}$$
$$\{A \leftarrow r, B \leftarrow v, C \leftarrow j, D \leftarrow r, E \leftarrow j\}$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\begin{aligned} &\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \end{aligned}$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\begin{aligned} &\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \end{aligned}$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\begin{array}{l} \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 2, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \end{array}$$

Critère : minimiser le nombre de contraintes violées

La recherche taboue

Objectif : éviter de boucler

La recherche taboue

Objectif : éviter de boucler

Emploi d’une liste dite "taboue" contenant les affectations déjà étudiées

La recherche taboue

Objectif : éviter de boucler

Emploi d’une liste dite "taboue" contenant les affectations déjà étudiées

Choix de la prochaine affectation : la meilleure voisine n’appartenant pas à la liste taboue

La recherche taboue

Objectif : éviter de boucler

Emploi d’une liste dite "taboue" contenant les affectations déjà étudiées

Choix de la prochaine affectation : la meilleure voisine n’appartenant pas à la liste taboue

Inconvénient : le coût en mémoire peut devenir prohibitif

La recherche taboue

Objectif : éviter de boucler

Emploi d'une liste dite "taboue" contenant les affectations déjà étudiées

Choix de la prochaine affectation : la meilleure voisine n'appartenant pas à la liste taboue

Inconvénient : le coût en mémoire peut devenir prohibitif

$\Rightarrow$ en pratique, on emploie une liste de taille limitée

Exemple

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

$$\begin{aligned} &\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \end{aligned}$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\begin{aligned} &\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 2, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ &\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \end{aligned}$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\begin{array}{l} \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 2, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 3, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \\ \{x_1 \leftarrow 1, x_2 \leftarrow 3, x_3 \leftarrow 2, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2) \end{array}$$

Critère : minimiser le nombre de contraintes violées

Le recuit simulé

Un concept basé sur la technique du recuit utilisé en métallurgie

Le recuit simulé

Un concept basé sur la technique du recuit utilisé en métallurgie

Objectif : transformer un métal en cristal parfait

Le recuit simulé

Un concept basé sur la technique du recuit utilisé en métallurgie

Objectif : transformer un métal en cristal parfait

- on chauffe le métal pour le liquéfier

Le recuit simulé

Un concept basé sur la technique du recuit utilisé en métallurgie

Objectif : transformer un métal en cristal parfait

- on chauffe le métal pour le liquéfier
- on baisse lentement la température

Le recuit simulé

Un concept basé sur la technique du recuit utilisé en métallurgie

Objectif : transformer un métal en cristal parfait

- on chauffe le métal pour le liquéfier
- on baisse lentement la température
- on le réchauffe de temps en temps pour quitter un état métastable

Le recuit simulé

Objectif : sortir du piège des optima locaux

Le recuit simulé

Objectif : sortir du piège des optima locaux

Choix de la prochaine affectation $\mathcal{A}'$:

- soit une affectation voisine qui améliore le critère de proximité

Le recuit simulé

Objectif : sortir du piège des optima locaux

Choix de la prochaine affectation $\mathcal{A}'$:

- soit une affectation voisine qui améliore le critère de proximité
- soit une affectation voisine qui détériore le critère avec une probabilité de $e^{-\frac{\Delta f}{T}}$

Le recuit simulé

Objectif : sortir du piège des optima locaux

Choix de la prochaine affectation $\mathcal{A}'$:

- soit une affectation voisine qui améliore le critère de proximité
- soit une affectation voisine qui détériore le critère avec une probabilité de $e^{-\frac{\Delta f}{T}}$

Δf = différence de valeur entre $\mathcal{A}$ et $\mathcal{A}'$

Le recuit simulé

Objectif : sortir du piège des optima locaux

Choix de la prochaine affectation $\mathcal{A}'$:

- soit une affectation voisine qui améliore le critère de proximité
- soit une affectation voisine qui détériore le critère avec une probabilité de $e^{-\frac{\Delta f}{T}}$

Δf = différence de valeur entre $\mathcal{A}$ et $\mathcal{A}'$

T = température qui décroît au cours du temps

Le recuit simulé

Remarques :

- la probabilité décroît au cours du temps

Le recuit simulé

Remarques :

- la probabilité décroît au cours du temps
- la probabilité dépend de la qualité de l'affectation voisine

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$

Critère : minimiser le nombre de contraintes violées

Exemple

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

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(5)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 1\}(4)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$
$$\{x_1 \leftarrow 2, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(2)$$

ou

$$\{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 2, x_5 \leftarrow 2\}(3)$$

Critère : minimiser le nombre de contraintes violées

Algorithmes génétiques

Gène = la valeur d’une variable

Algorithmes génétiques

Gène = la valeur d'une variable

Individu = une affectation complète

Algorithmes génétiques

Gène = la valeur d'une variable

Individu = une affectation complète

Population = ensemble d'affectations complètes

Algorithmes génétiques

Principe :

- on choisit une population de départ

Algorithmes génétiques

Principe :

- on choisit une population de départ
- on produit de nouveaux individus par
  - reproduction sélective
  - croisement
  - mutation

Algorithmes génétiques

Principe :

- on choisit une population de départ
- on produit de nouveaux individus par
  - reproduction sélective
  - croisement
  - mutation
- on ne garde que les meilleurs individus

Algorithmes génétiques

Principe :

- on choisit une population de départ
- on produit de nouveaux individus par
  - reproduction sélective
  - croisement
  - mutation
- on ne garde que les meilleurs individus
- on réitère le procédé jusqu'à l'obtention d'une solution

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs
- croisement :
  - on sélectionne deux individus

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs
- croisement :
  - on sélectionne deux individus
  - on choisit une zone de croisement

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs
- croisement :
  - on sélectionne deux individus
  - on choisit une zone de croisement
  - on permute les valeurs des variables concernées

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs
- croisement :
  - on sélectionne deux individus
  - on choisit une zone de croisement
  - on permute les valeurs des variables concernées
- mutation :
  - on sélectionne un individu

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs
- croisement :
  - on sélectionne deux individus
  - on choisit une zone de croisement
  - on permute les valeurs des variables concernées
- mutation :
  - on sélectionne un individu
  - on change la valeur d'une variable de cet individu

Algorithmes génétiques

- reproduction sélective : on favorise la reproduction des individus les plus prometteurs
- croisement :
  - on sélectionne deux individus
  - on choisit une zone de croisement
  - on permute les valeurs des variables concernées
- mutation :
  - on sélectionne un individu
  - on change la valeur d'une variable de cet individu

Les croisements sont plus fréquents que les mutations

Algorithmes génétiques

Les principaux paramètres à régler :

- taille de la population

Algorithmes génétiques

Les principaux paramètres à régler :

- taille de la population
- critère de sélection

Algorithmes génétiques

Les principaux paramètres à régler :

- taille de la population
- critère de sélection
- pourcentage de croisements

Algorithmes génétiques

Les principaux paramètres à régler :

- taille de la population
- critère de sélection
- pourcentage de croisements
- pourcentage de mutations

Exemple

$\mathcal{A}_1 = \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 3\}$ c_{12}, c_{13}, c_{14}

$\mathcal{A}_2 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 2, x_4 \leftarrow 3, x_5 \leftarrow 2\}$ c_{25}, c_{35}

Exemple

$\mathcal{A}_1 = \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 3\}$ c_{12}, c_{13}, c_{14}

$\mathcal{A}_2 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 2, x_4 \leftarrow 3, x_5 \leftarrow 2\}$ c_{25}, c_{35}

- reproduction sélective : $\mathcal{A}_2$ a plus de chance de se reproduire

Exemple

$\mathcal{A}_1 = \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 3\}$ c_{12}, c_{13}, c_{14}

$\mathcal{A}_2 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 2, x_4 \leftarrow 3, x_5 \leftarrow 2\}$ c_{25}, c_{35}

- reproduction sélective : $\mathcal{A}_2$ a plus de chance de se reproduire
- croisement :

  $\mathcal{A}_3 = \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 2, x_4 \leftarrow 3, x_5 \leftarrow 2\}$ c_{12}, c_{13}, c_{35}

  $\mathcal{A}_4 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 3\}$ c_{13}

Exemple

$\mathcal{A}_1 = \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 3\}$ c_{12}, c_{13}, c_{14}

$\mathcal{A}_2 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 2, x_4 \leftarrow 3, x_5 \leftarrow 2\}$ c_{25}, c_{35}

- reproduction sélective : $\mathcal{A}_2$ a plus de chance de se reproduire
- croisement :
  $\mathcal{A}_3 = \{x_1 \leftarrow 3, x_2 \leftarrow 1, x_3 \leftarrow 2, x_4 \leftarrow 3, x_5 \leftarrow 2\}$ c_{12}, c_{13}, c_{35}

  $\mathcal{A}_4 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 1, x_4 \leftarrow 1, x_5 \leftarrow 3\}$ c_{13}
- mutation :
  $\mathcal{A}'_4 = \{x_1 \leftarrow 1, x_2 \leftarrow 2, x_3 \leftarrow 2, x_4 \leftarrow 1, x_5 \leftarrow 3\}$

Commentaires

Chacune de ces méthodes possède des paramètres :

Commentaires

Chacune de ces méthodes possède des paramètres :

- heuristique d’évaluation de l’éloignement

Commentaires

Chacune de ces méthodes possède des paramètres :

- heuristique d'évaluation de l'éloignement
- vitesse à laquelle décroît la température

Commentaires

Chacune de ces méthodes possède des paramètres :

- heuristique d'évaluation de l'éloignement
- vitesse à laquelle décroît la température
- longueur de la liste taboue

Commentaires

Chacune de ces méthodes possède des paramètres :

- heuristique d'évaluation de l'éloignement
- vitesse à laquelle décroît la température
- longueur de la liste taboue
- . . .

Commentaires

Chacune de ces méthodes possède des paramètres :

- heuristique d'évaluation de l'éloignement
- vitesse à laquelle décroît la température
- longueur de la liste taboue
- . . .

Leur efficacité dépend du choix judicieux des valeurs des paramètres.

Plan

1. Méthodes de résolution énumératives (suite)
2. Méthodes structurelles
3. Méthodes incomplètes
4. Choix d’une méthode

Choix d’une méthode

Méthodes complètes :

- méthodes énumératives : efficaces en pratique

Choix d’une méthode

Méthodes complètes :

- méthodes énumératives : efficaces en pratique
- méthodes structurelles : une meilleure complexité

Choix d’une méthode

Méthodes complètes :

- méthodes énumératives : efficaces en pratique
- méthodes structurelles : une meilleure complexité mais une efficacité pratique à prouver

Choix d’une méthode

Méthodes complètes :

- méthodes énumératives : efficaces en pratique
- méthodes structurelles : une meilleure complexité mais une efficacité pratique à prouver

Méthodes incomplètes : intéressantes pour résoudre les gros problèmes