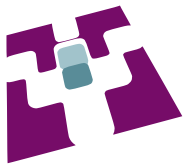
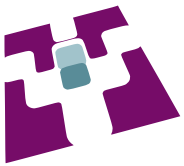
Méthodes de recherche locale et hybridations

GREYC, CNRS UMR 6072
Université de Caen Basse-Normandie, France

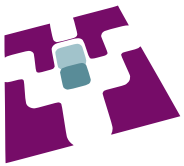


Plan de la présentation



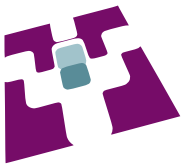
Motivation

- La programmation par contraintes :
  - propose un ensemble de mécanismes pour définir les objectifs, les contraintes et la recherche de solutions ;
  - n'est sujette à très peu de restriction en terme de modélisation ;
  - techniques de recherche essentiellement basées sur des méthodes complètes ;
  - mécanismes de filtrage très puissants.
- Devrait donc être un outil idéal pour résoudre des problèmes combinatoire sous contraintes

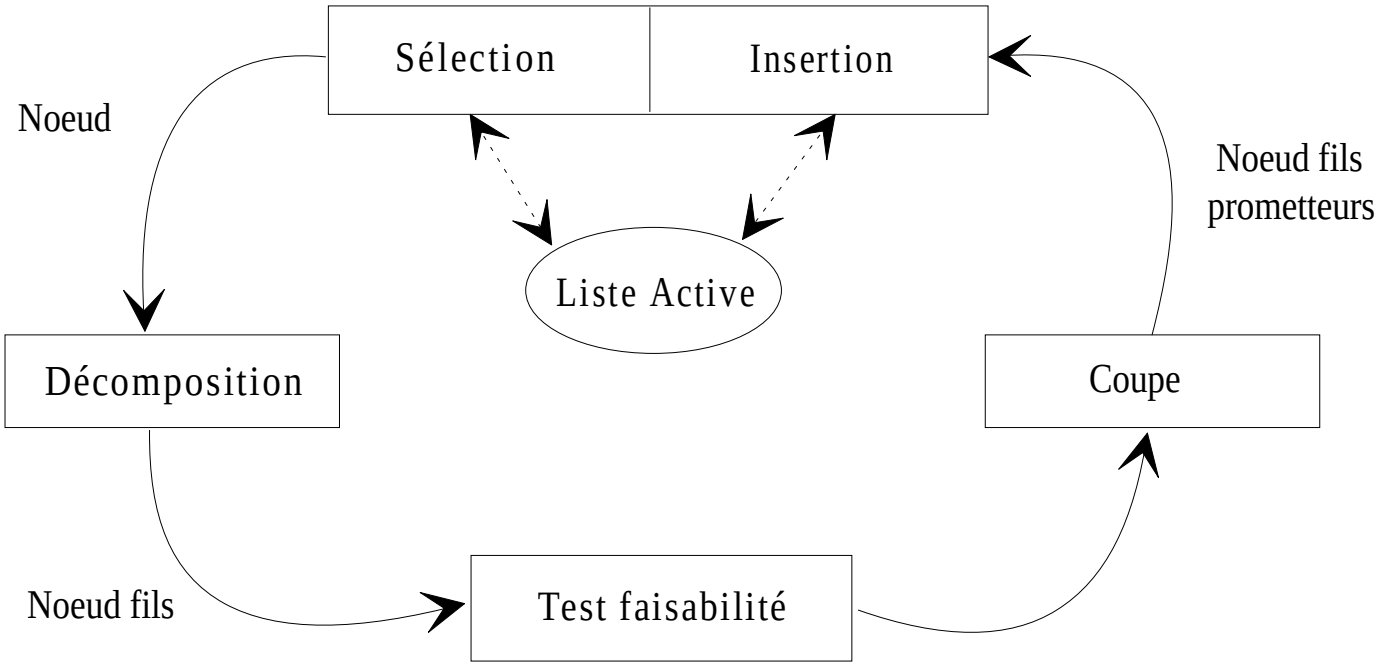


Motivation

- Par contre, avec la programmation par contraintes :
  - parcours systématique de l'arbre de recherche ;
  - temps de résolution peut être très important ;
  - toute interruption de la recherche en cours de route implique des solutions de qualité médiocres ;



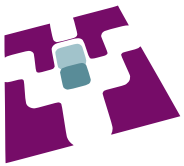
Principe d'une recherche arborescente



(a) stratégie de sélection/exploration (DFS/BFS,...) ;

(b) stratégie de décomposition/recherche ;

(b) stratégie de coupes ;



Exemple d'illustration

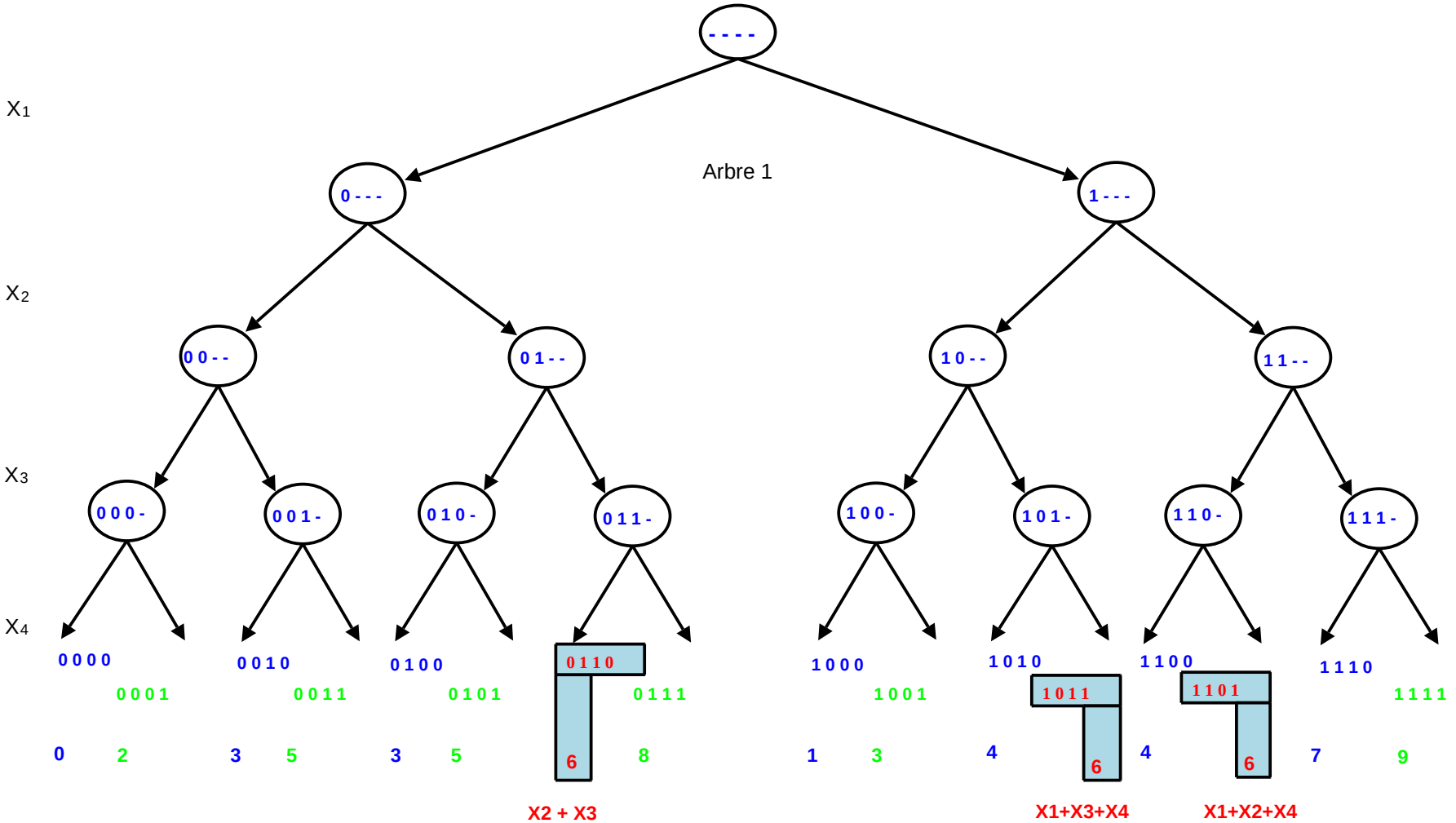
$x_1 : :\{0,1\}$
$x_2 : :\{0,1\}$
$x_3 : :\{0,1\}$
$x_4 : :\{0,1\}$

$x_1 + 3x_2 + 3x_3 + 2x_4 = 6$

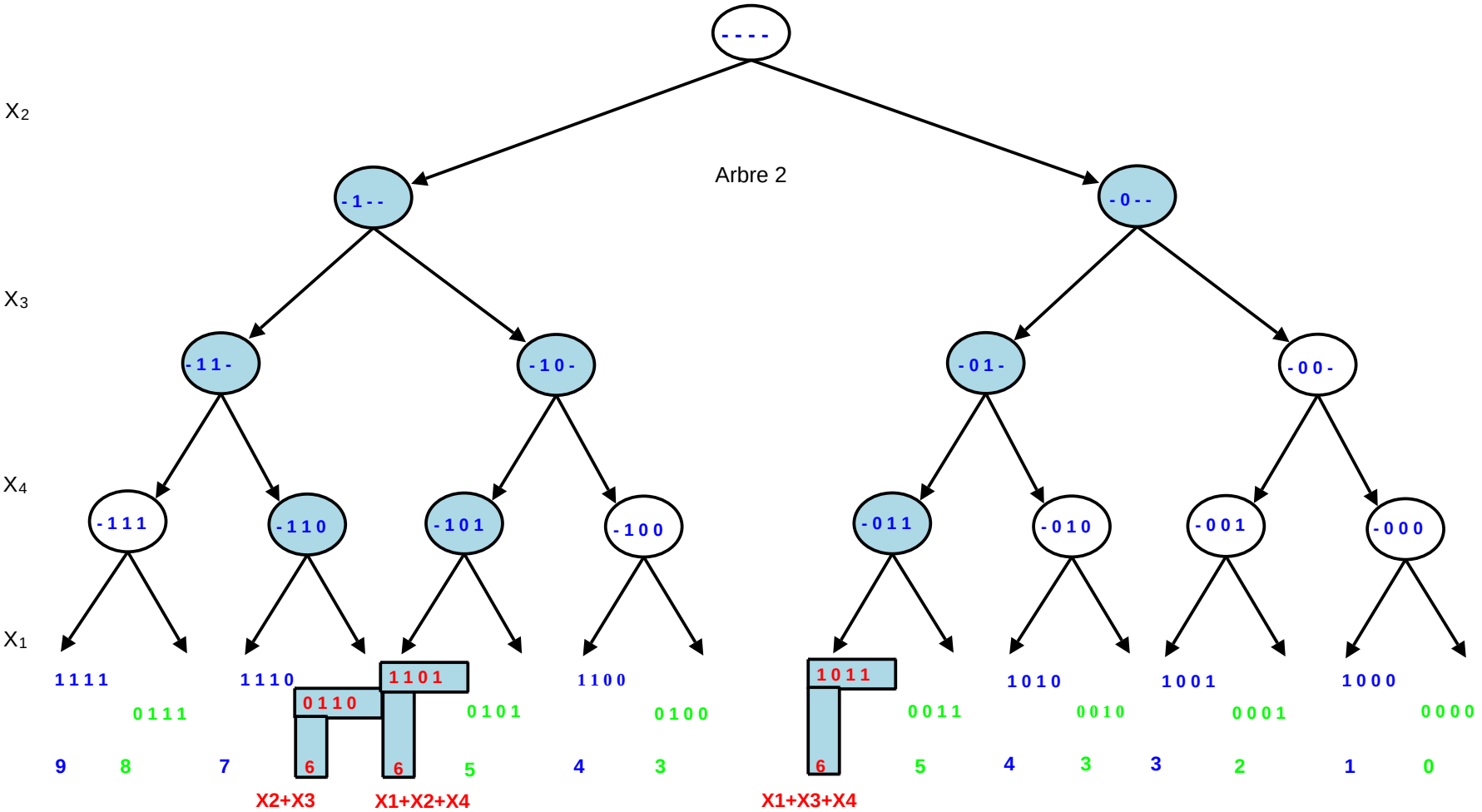
16 affectations
3 solutions

$$\begin{array}{cccccccccc} & & 3 & + & 3 & & & = & 6 & sol.1 \\ 1 & + & & & 3 & + & 2 & = & 6 & sol.2 \\ 1 & + & 3 & & & + & 2 & = & 6 & sol.3 \end{array}$$

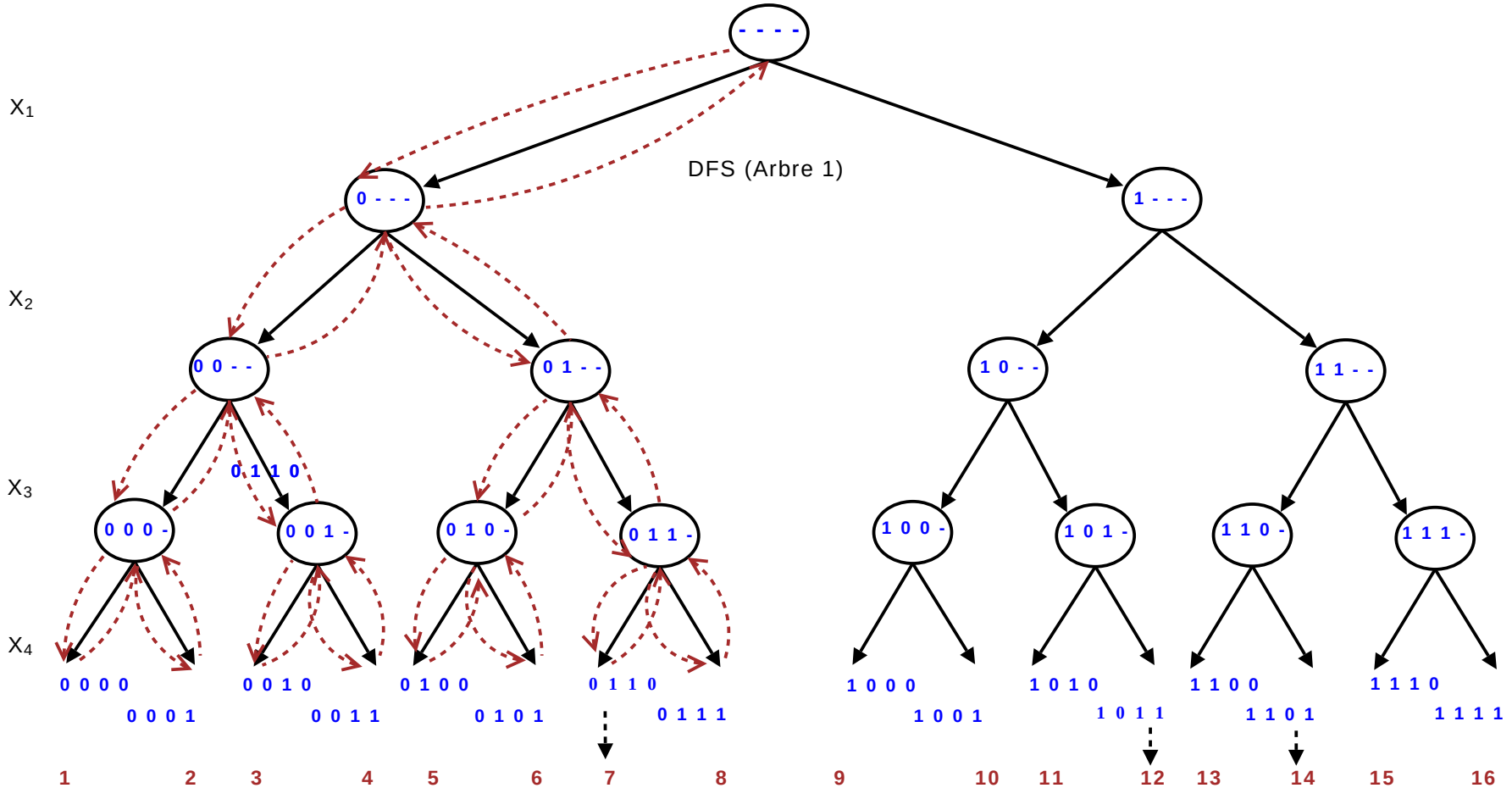
Stratégie de recherche : Input-order/Min



Stratégie de recherche : DecreasingCoef/Max

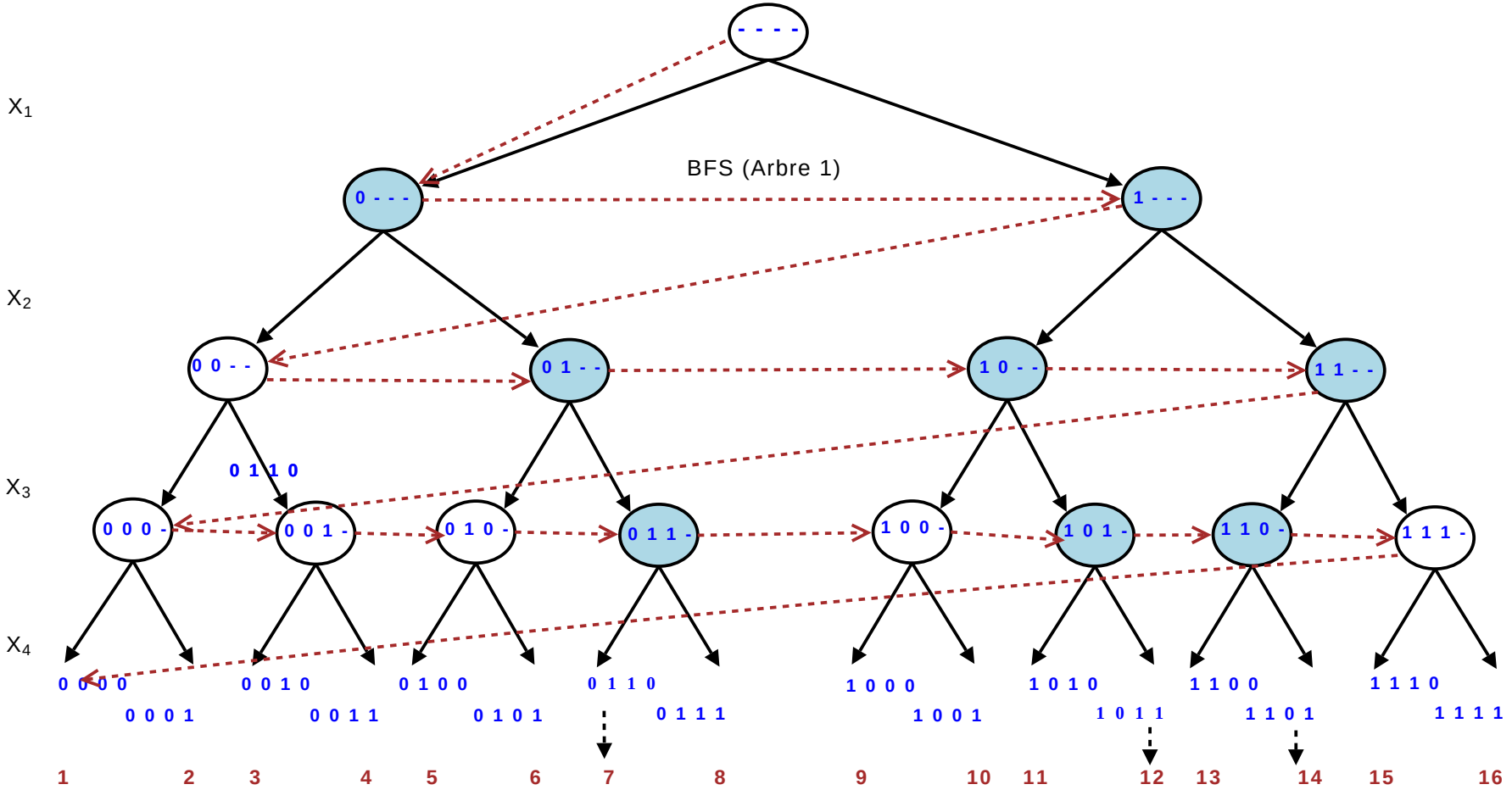


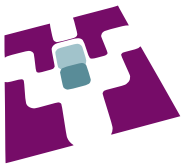
Stratégie d'exploration : Depth First Search



Ordre de parcours des noeuds

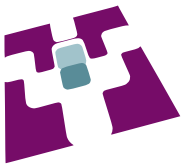
Stratégie d'exploration : Breadth First Search





Recherches arborescentes tronquées

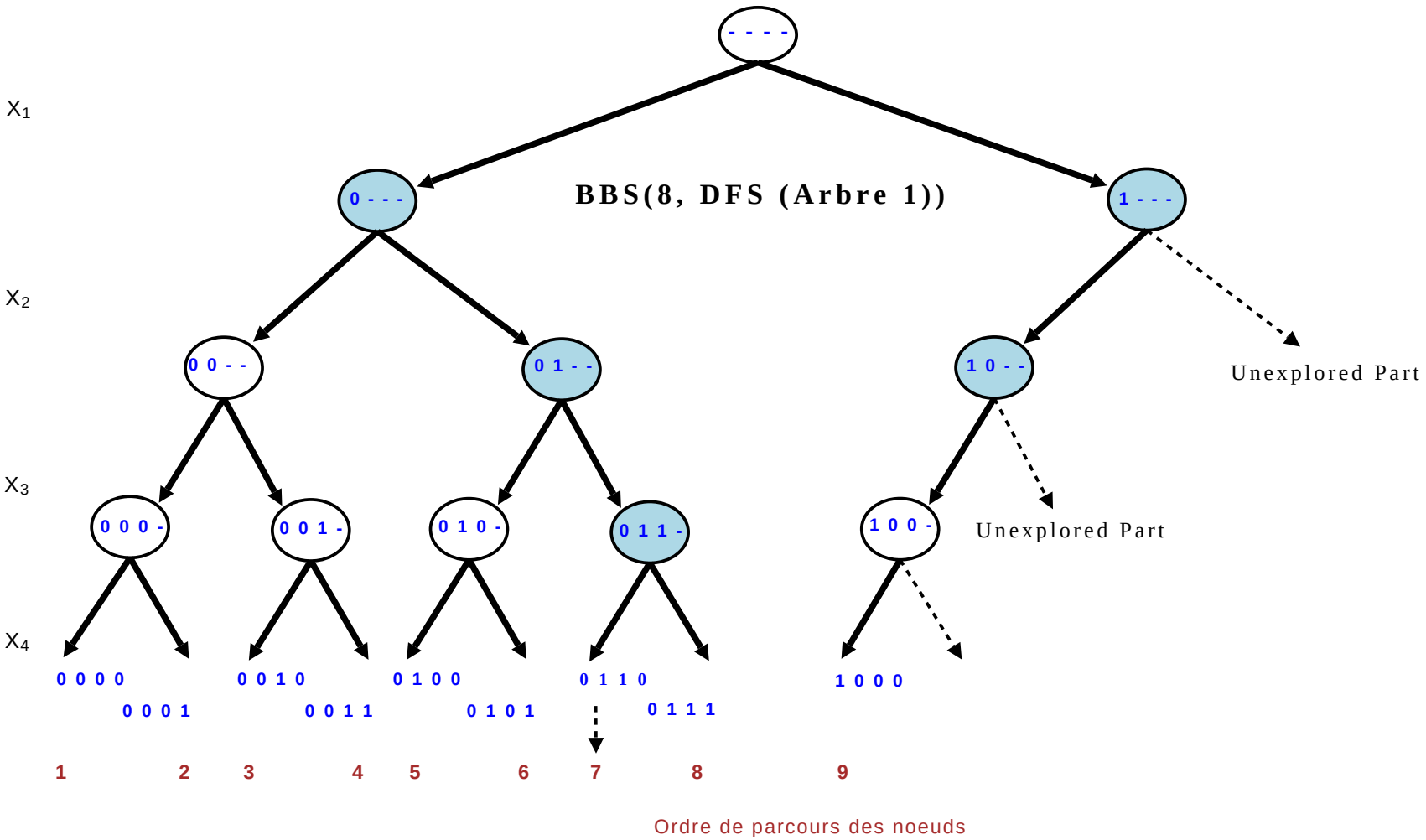
- A cutoff limit to stop exploring a (sub-)tree
  - some branches are skipped $\rightarrow$ incomplete search
- When no solution found, restart with enlarged cutoff limit.
- Provide a *good* solution very quickly.
- Diversify the exploration of the search space.



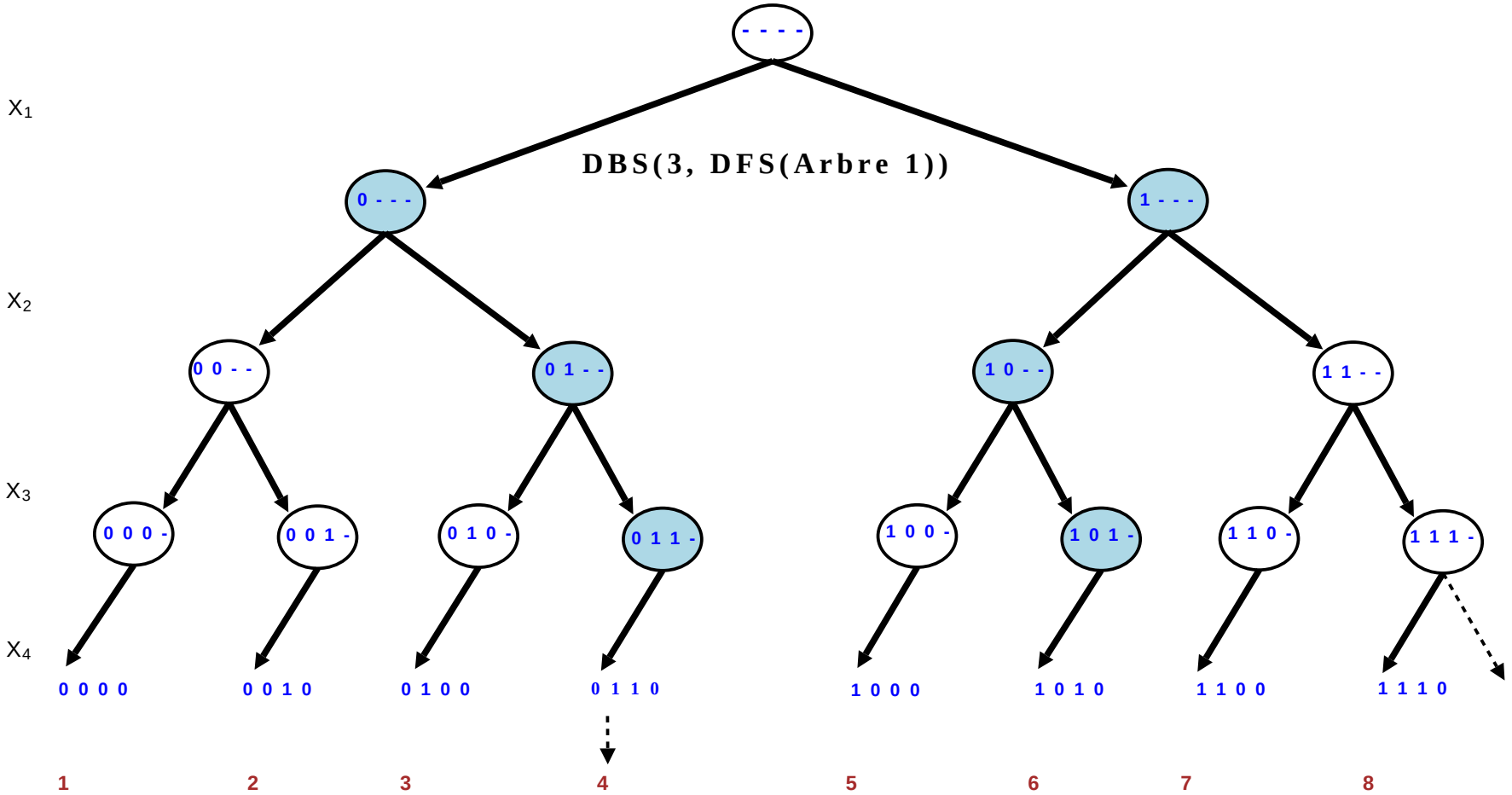
Recherches arborescentes tronquées

- Bounded Backtrack Search (Harvey, 1995)
  - restricts number of backtracks
- Depth-bounded Backtrack Search (Cheadle et al., 2003)
  - restricts depth where alternatives are explored
- Iterative Broadening (Ginsberg and Harvey, 1990)
  - restricts breadth in each node

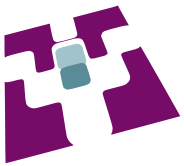
Bounded Backtrack Search - BBS



Depth-bounded Backtrack Search - DBS

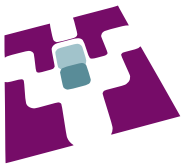


Ordre de parcours des noeuds



Heuristics search - some observations

- **Heuristics** - a guide of search
  - they recommend a value for assignment
  - quite often lead to a solution
- What to do upon a **failure of the heuristic** ?
  - BT rather repairs later assignments than the earliest ones
    thus BT assumes that the heuristic guides it well in the top part
- **Heuristics** are **less reliable in the earlier parts** of the search tree (as search proceeds, more information is available).
- The number of **heuristic violations** is usually **small**.

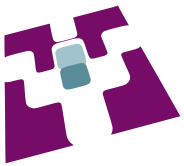


Discrepancy search

Principe

Soit h une heuristique dans laquelle on a une grande confiance. Le principe d'une recherche à déviation est de suivre l'heuristique h lors du parcours de l'arbre de recherche, mais en considérant que h peut se tromper un petit nombre (k) de fois. On s'autorise donc k écarts (*discrepancies*) à l'heuristique h lors du parcours de l'arbre.

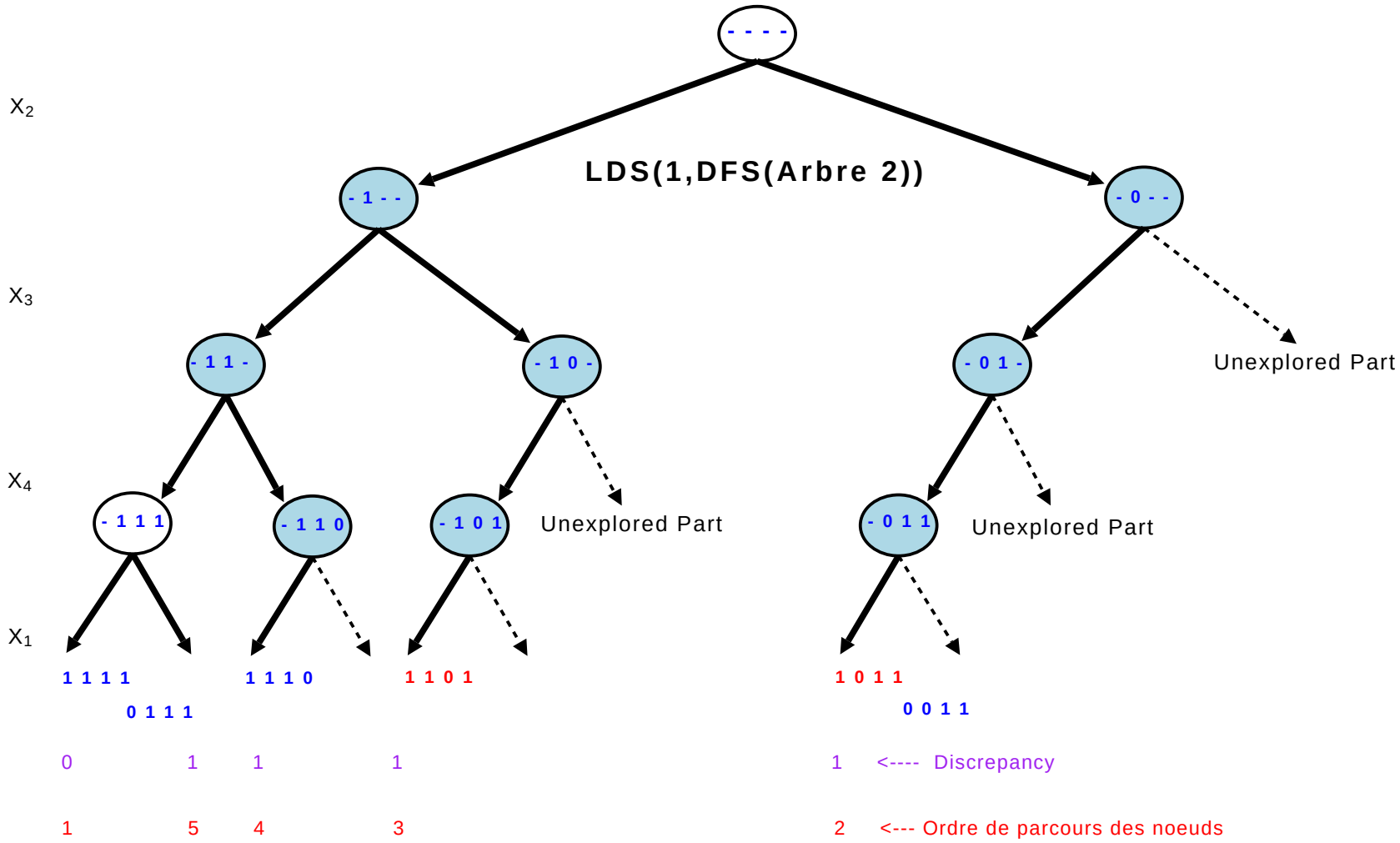
- **Discrepancy** = écart à l'heuristique h
- **L'idée de base** : changer l'ordre d'exploration des branches
  - préférer les branches ayant **moins discrepancies**
  - préférer les branches ayant **des discrepancies en haut** de l'arbre

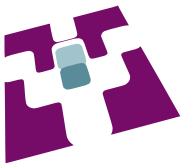


Discrepancy search

- Limited Discrepancy Search (Harvey & Ginsberg, 1995)
  - restricts a maximal number of discrepancy in the iterartion
- Improved LDS (Korf, 1996)
  - restricts a given number of discrepancies in the iteration
- Depth-bounded Discrepancy Search (Walsh, 1997)
  - restricts discrepancies till a given depth in the iteration

LDS(k) - Illustration sur l'exemple





Principe de base d'une recherche locale

- Une **recherche locale** fait appel à un voisinage défini sur l'ensemble des configurations.
- **Voisinage**
  - Fonction $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ qui associe à chaque configuration $s \in \mathcal{S}$ un sous-ensemble $\mathcal{N}(s)$ (**l'ensemble des voisins de** s) de $\mathcal{S}$.
- **Mouvement**
  - Opération qui consiste à modifier une configuration $s \in \mathcal{S}$ en une autre configuration voisine $s' \in \mathcal{N}(s)$.
- **Optimum local**
  - Configuration s de $\mathcal{S}$ telle que $f(s) \leq f(s')$, pour tout voisin $s' \in \mathcal{N}(s)$.

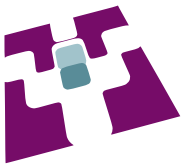
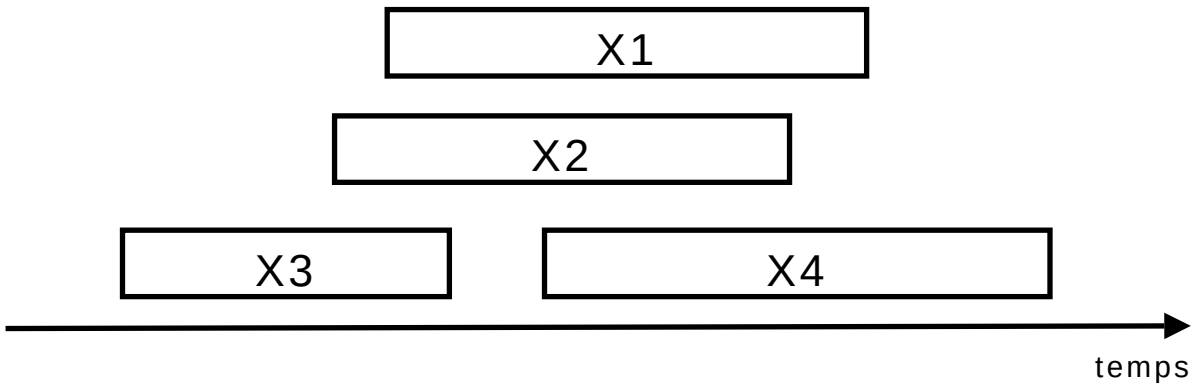


Schéma d'une recherche locale

Procédure générale

- **Étape 1 (initialisation)**
  a) choisir une solution initiale s dans $\mathcal{S}$
  b) $s^\star := s$ (i.e. mémoriser la meilleure solution trouvée)
- **Étape 2 (choix)**
  a) choisir s' dans $\mathcal{N}(s)$
  b) $s := s'$ (i.e. remplacer s par s')
  c) terminer et retourner la meilleure solution trouvée si la condition d'arrêt vérifiée
- **Étape 3 (mise à jour)**
  a) $s^\star := s$ si $f(s) < f(s^\star)$
  b) aller à l'étape 2

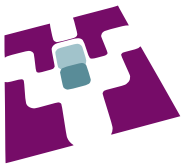
Exemple d'illustration



X1 = {B,C}
X2 = {A,C}
X3 = {B,C}
X4 = {A,B}

Les contraintes:

C(x1,x2) : {(B,A), (B,C), (C,A)}
C(x1,x3) : {(B,C), (C,B)}
C(x1,x4) : {(B,A), (C,B), (C,A)}
C(x2,x3) : {(A,B), (A,C), (C,B)}
C(x2,x4) : {(A,B), (C,A), (C,B)}

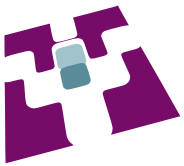


Recherche locale : descente stricte

- **Étape 2 (choix)**
  a) choisir s' dans $\mathcal{N}(s)$ tq $f(s') < f(s)$
  b) $s := s'$ (i.e. remplacer s par s')
  c) terminer si, pour tout s' $\mathcal{N}(s)$ $f(s') > f(s)$
  (i.e. s'arrêter sur un optimum local)

- **Remarques**
  - *Décision à prendre* : 1ère ou meilleure amélioration
  - La procédure s'arrête au 1er optimum local rencontré.
  - Il est possible de poursuivre la recherche en effectuant une relance aléatoire (**restart**).

$\rightarrow$ Comment **s'échapper des optima locaux** ?

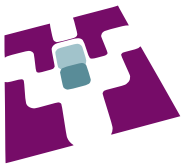


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits

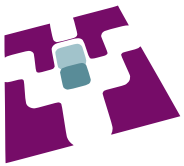


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits

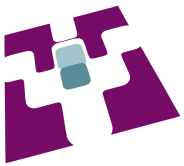


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$		

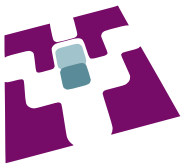


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1

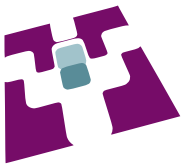


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$		

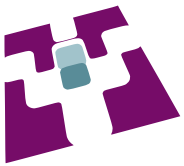


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1

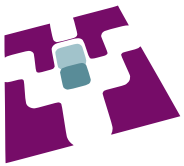


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$		

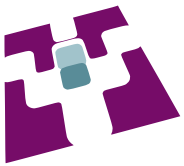


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$	$c(x_2, x_4)$	1

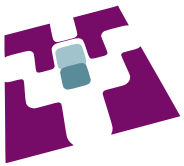


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_4 \rightarrow \text{B}$		

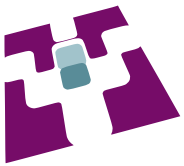


L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_4 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_1, x_4)$	2



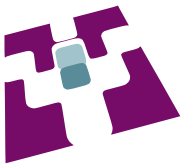
L'heuristique min-conflict

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_4 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_1, x_4)$	2

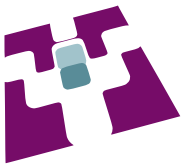
Accepter $(x_1 \rightarrow \text{C})$: $s' = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$



L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

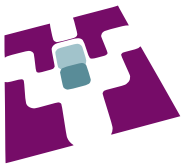
variable changée	conflits avec	nombre total de conflits



L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

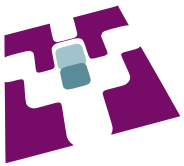
variable changée	conflits avec	nombre total de conflits



L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{B}$		



L'heuristique min-conflict

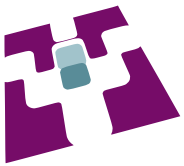
2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2

L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

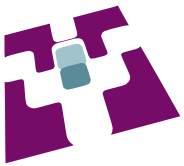
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \text{C}$		



L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

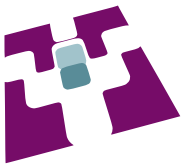
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \text{C}$	$c(x_1, x_2)$	1



L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

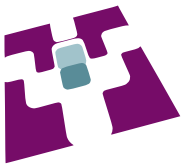
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_3 \rightarrow \text{C}$		



L'heuristique min-conflict

2ème itération : $s = (x_1 = \mathrm{C},\ x_2 = \mathrm{A},\ x_3 = \mathrm{B},\ x_4 = \mathrm{A})$

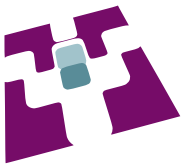
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \mathrm{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \mathrm{C}$	$c(x_1, x_2)$	1
$x_3 \rightarrow \mathrm{C}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2



L'heuristique min-conflict

2ème itération : $s = (x_1 = \mathrm{C},\ x_2 = \mathrm{A},\ x_3 = \mathrm{B},\ x_4 = \mathrm{A})$

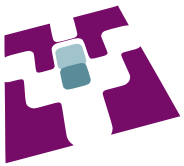
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \mathrm{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \mathrm{C}$	$c(x_1, x_2)$	1
$x_3 \rightarrow \mathrm{C}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_4 \rightarrow \mathrm{B}$		



L'heuristique min-conflict

2ème itération : $s = (x_1 = \mathrm{C},\ x_2 = \mathrm{A},\ x_3 = \mathrm{B},\ x_4 = \mathrm{A})$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \mathrm{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \mathrm{C}$	$c(x_1, x_2)$	1
$x_3 \rightarrow \mathrm{C}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_4 \rightarrow \mathrm{B}$	–	0



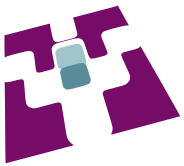
L'heuristique min-conflict

2ème itération : $s = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_2 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_3 \rightarrow \text{C}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_4 \rightarrow \text{B}$	–	0

Accepter $(x_4 \rightarrow \text{B})$: $s' = (x_1 = \text{C}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{B})$

$\rightarrow$ Solution ! !

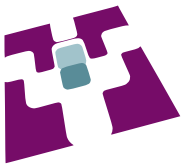


L'heuristique min-conflict - optimum local

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_4 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_1, x_4)$	2



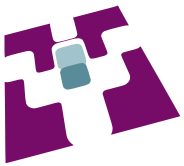
L'heuristique min-conflict - optimum local

1er itération : $s = (x_1 = \text{B}, x_2 = \text{A}, x_3 = \text{B}, x_4 = \text{A})$

$\rightarrow$ 2 **conflits** : $c(x_1, x_3)$ et $c(x_2, x_4)$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_2 \rightarrow \text{C}$	$c(x_1, x_3)$	1
$x_3 \rightarrow \text{C}$	$c(x_2, x_4)$	1
$x_4 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_1, x_4)$	2

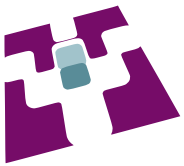
Accepter $(x_2 \rightarrow \text{C})$: $s' = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

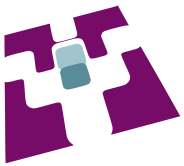
variable changée	conflits avec	nombre total de conflits



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

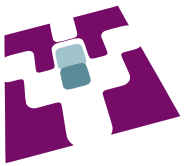
variable changée	conflits avec	nombre total de conflits



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

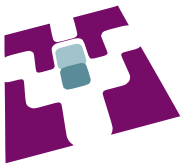
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$		



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

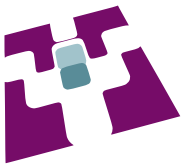
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

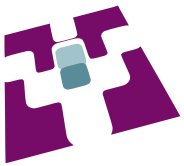
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$		



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

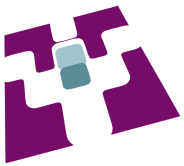
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

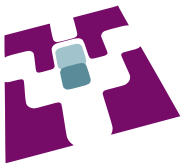
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_3 \rightarrow \text{C}$		



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

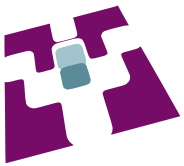
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_3 \rightarrow \text{C}$	$c(x_2, x_3)$	1



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

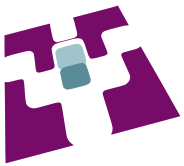
variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_3 \rightarrow \text{C}$	$c(x_2, x_3)$	1
$x_4 \rightarrow \text{B}$		



L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_3 \rightarrow \text{C}$	$c(x_2, x_3)$	1
$x_4 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2



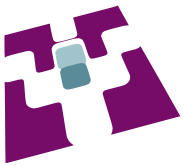
L'heuristique min-conflict - optimum local

2ème itération : $s = (x_1 = \text{B}, x_2 = \text{C}, x_3 = \text{B}, x_4 = \text{A})$

variable changée	conflits avec	nombre total de conflits
$x_1 \rightarrow \text{C}$	$c(x_1, x_2)$	1
$x_2 \rightarrow \text{A}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2
$x_3 \rightarrow \text{C}$	$c(x_2, x_3)$	1
$x_4 \rightarrow \text{B}$	$c(x_1, x_3)$ et $c(x_2, x_4)$	2

Aucun changement qui réduit le nombre de conflits

$\rightarrow$ FIN



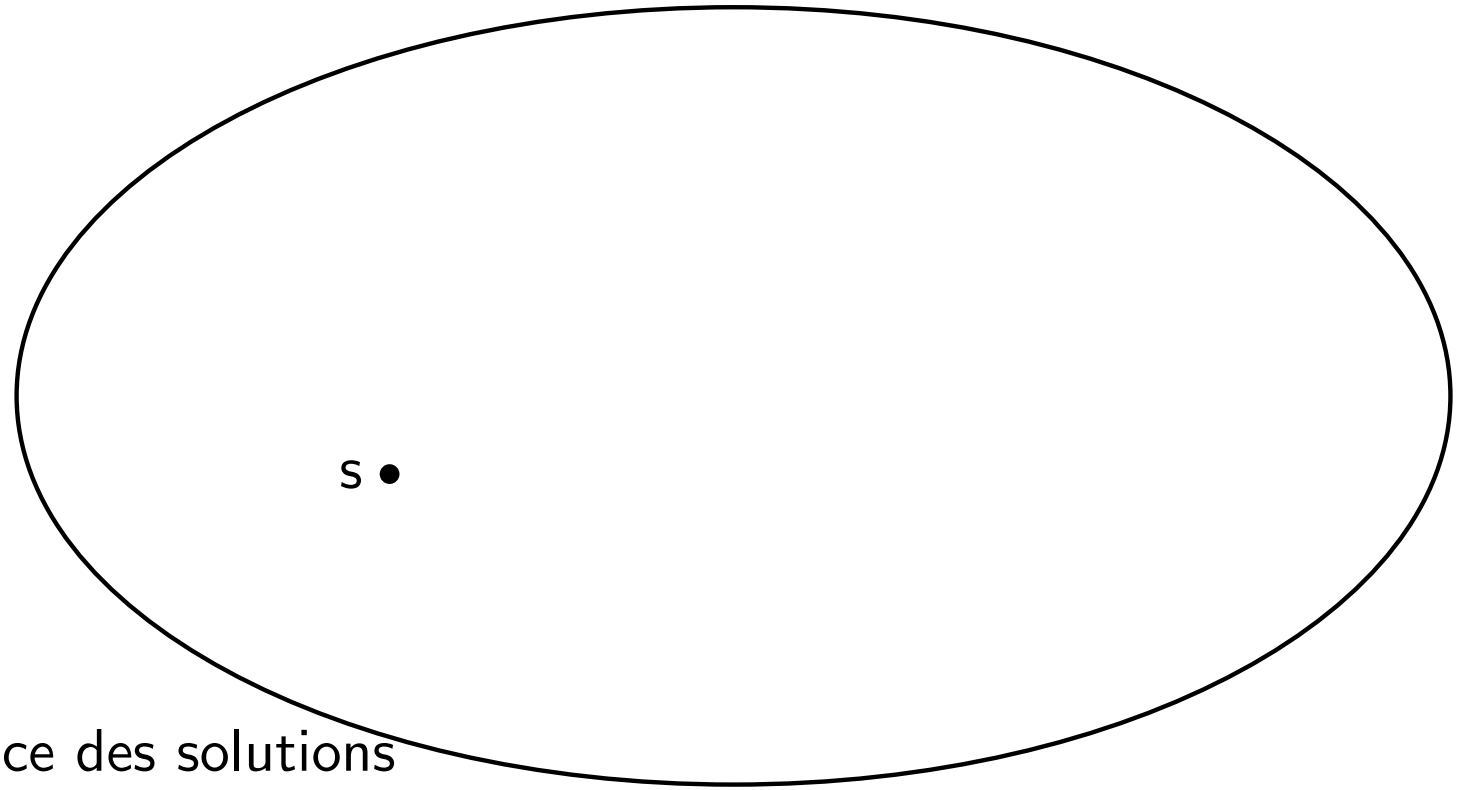
Recherche locale : recuit simulé

- **Étape 2 (choix)**
  a) choisir au hasard s' dans $\mathcal{N}(s)$, calculer $\Delta = f(s') - f(s)$
  b) si $\Delta \leq 0$ alors accepter s
  sinon accepter s avec une probabilité $p(\Delta, T)$
  C) si terminer si la condition d'arrêt est réalisée
  (un nb. max d'itér est effectué...)

- **Remarques**
  - La probabilité d'accepter dépend du niveau de la dégradation de la solution (Δ) et d'un paramètre appelé température.
  - La température décroît au cours de la recherche, ce qui rend les dégradations de la solution de moins en moins probables.

Recherche à voisinage variable (VNS, Hansen et al., 1997)

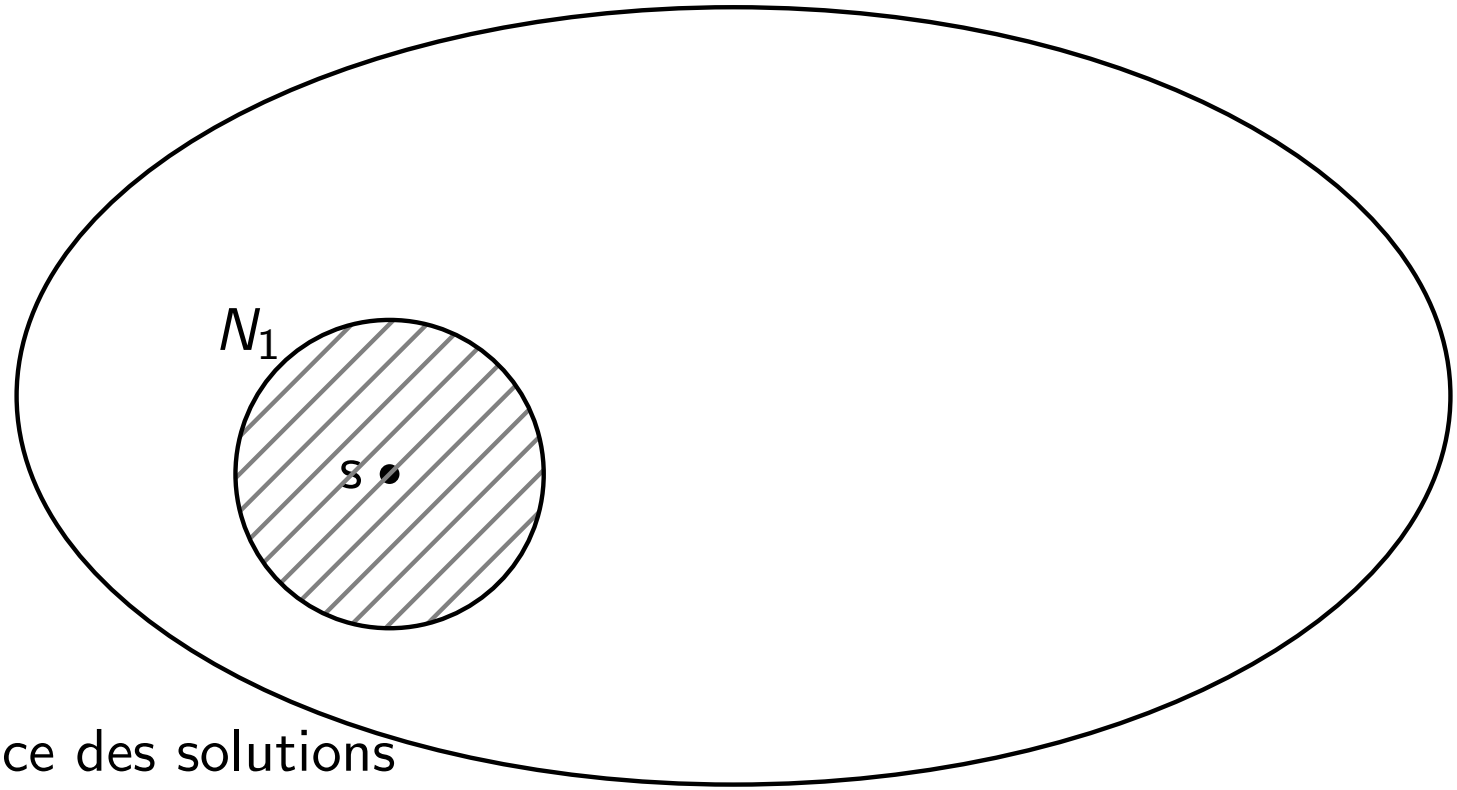
- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



Espace des solutions

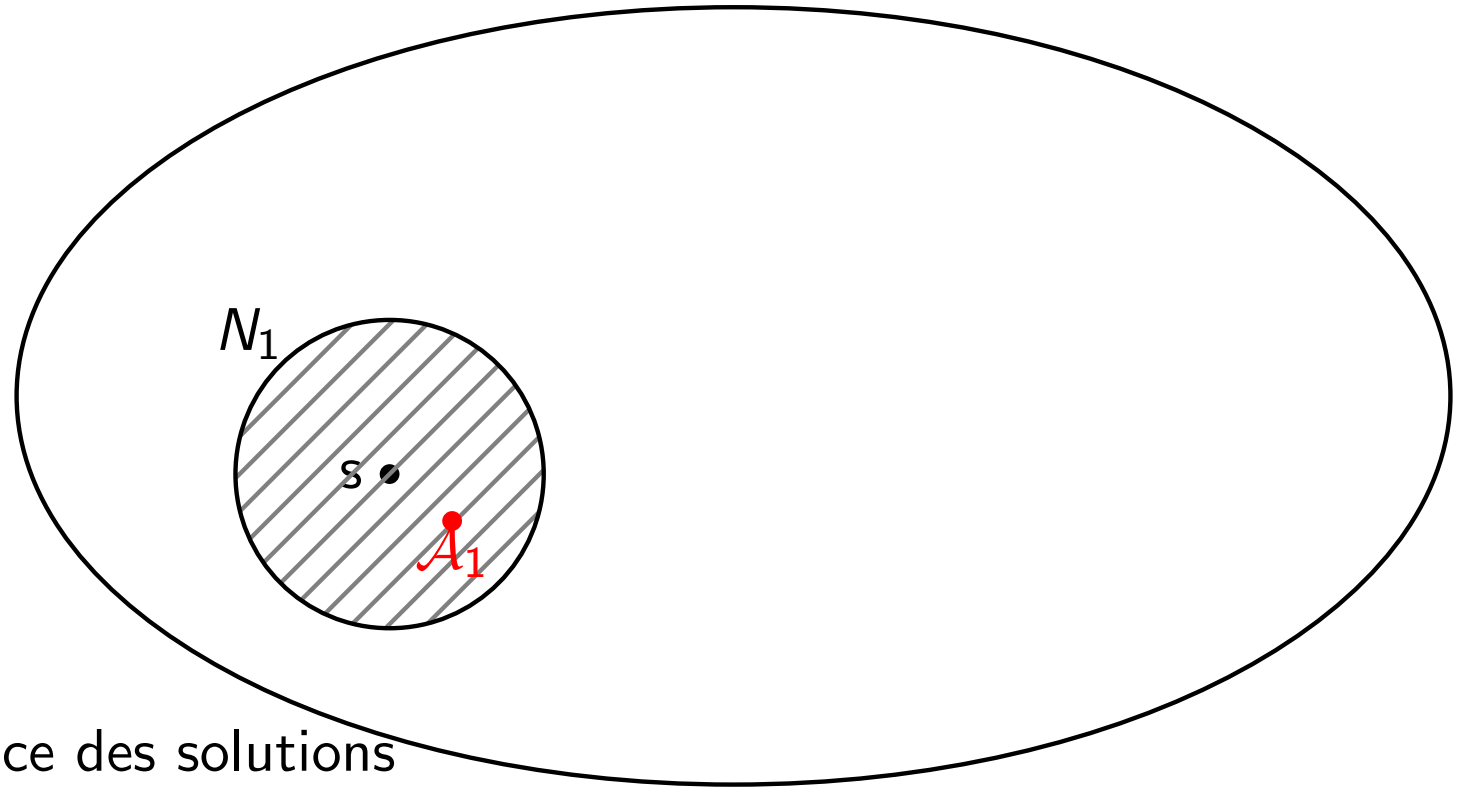
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



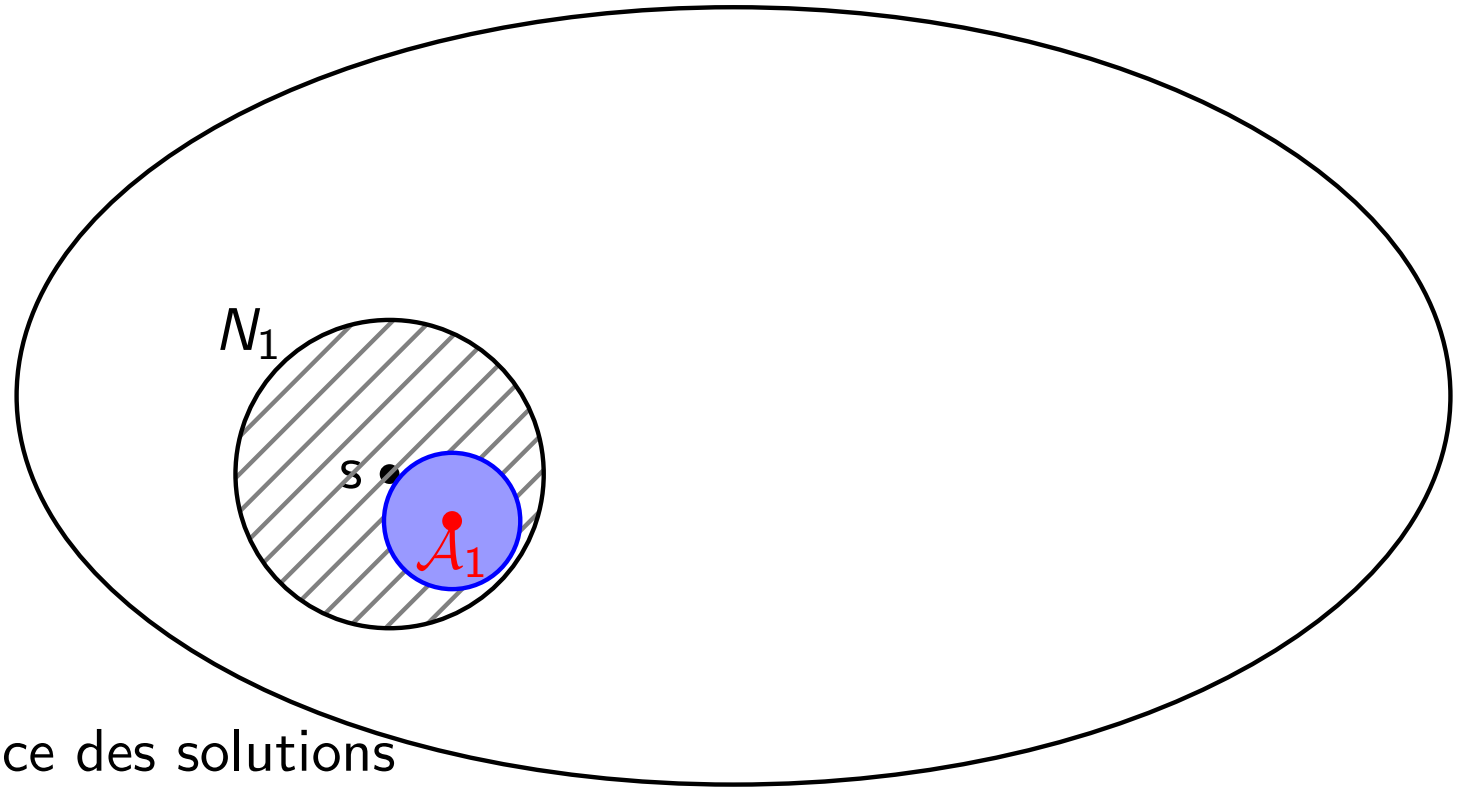
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



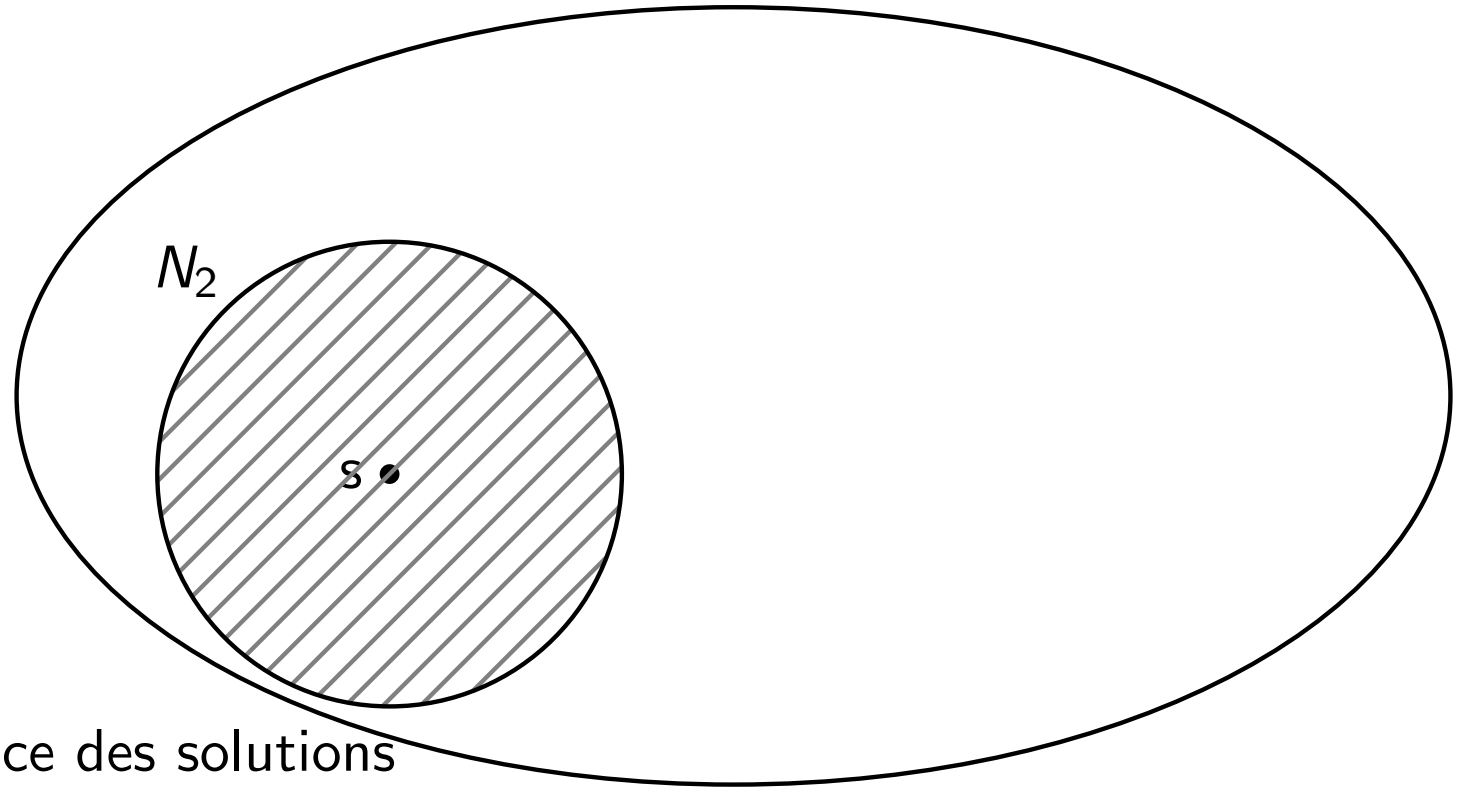
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



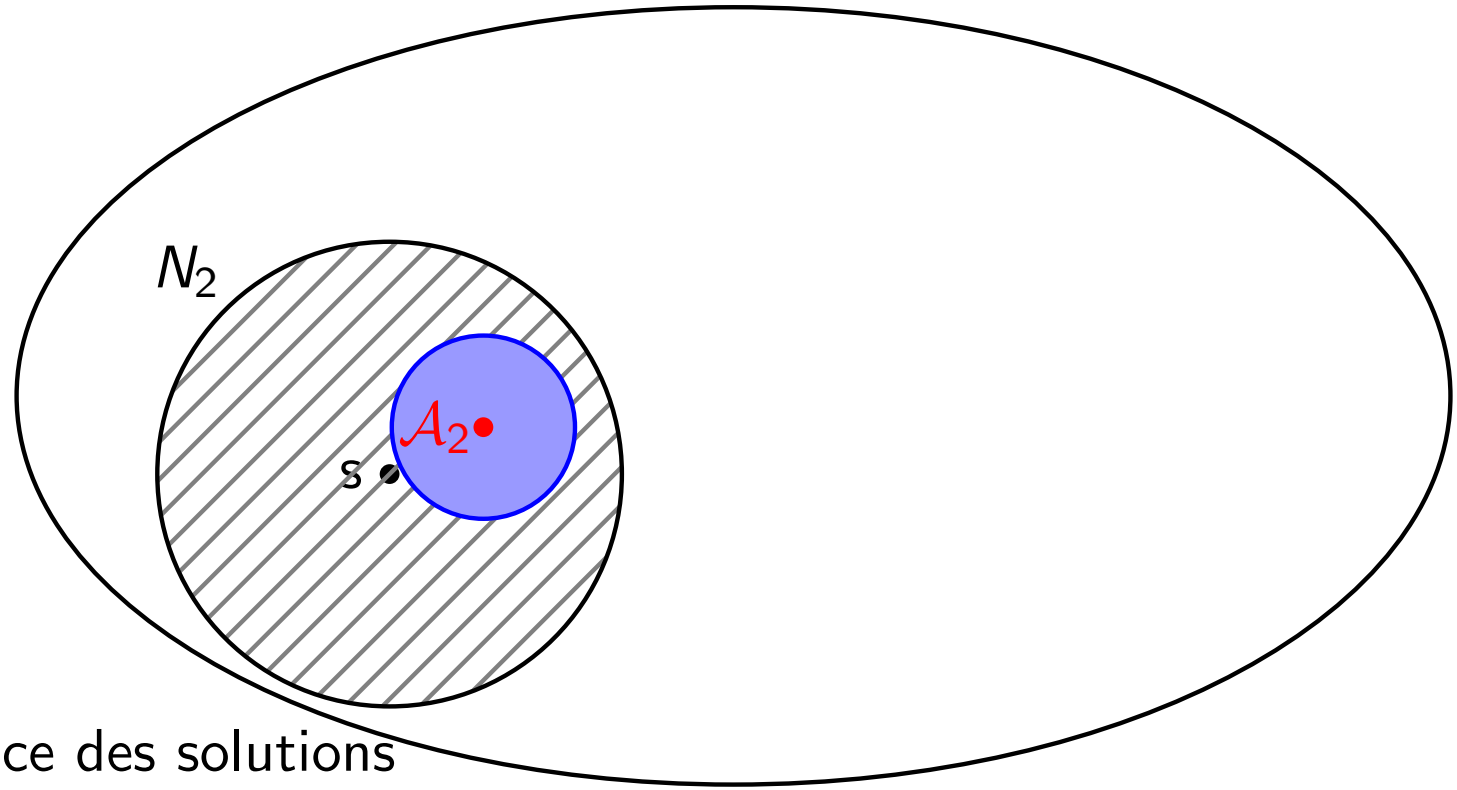
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



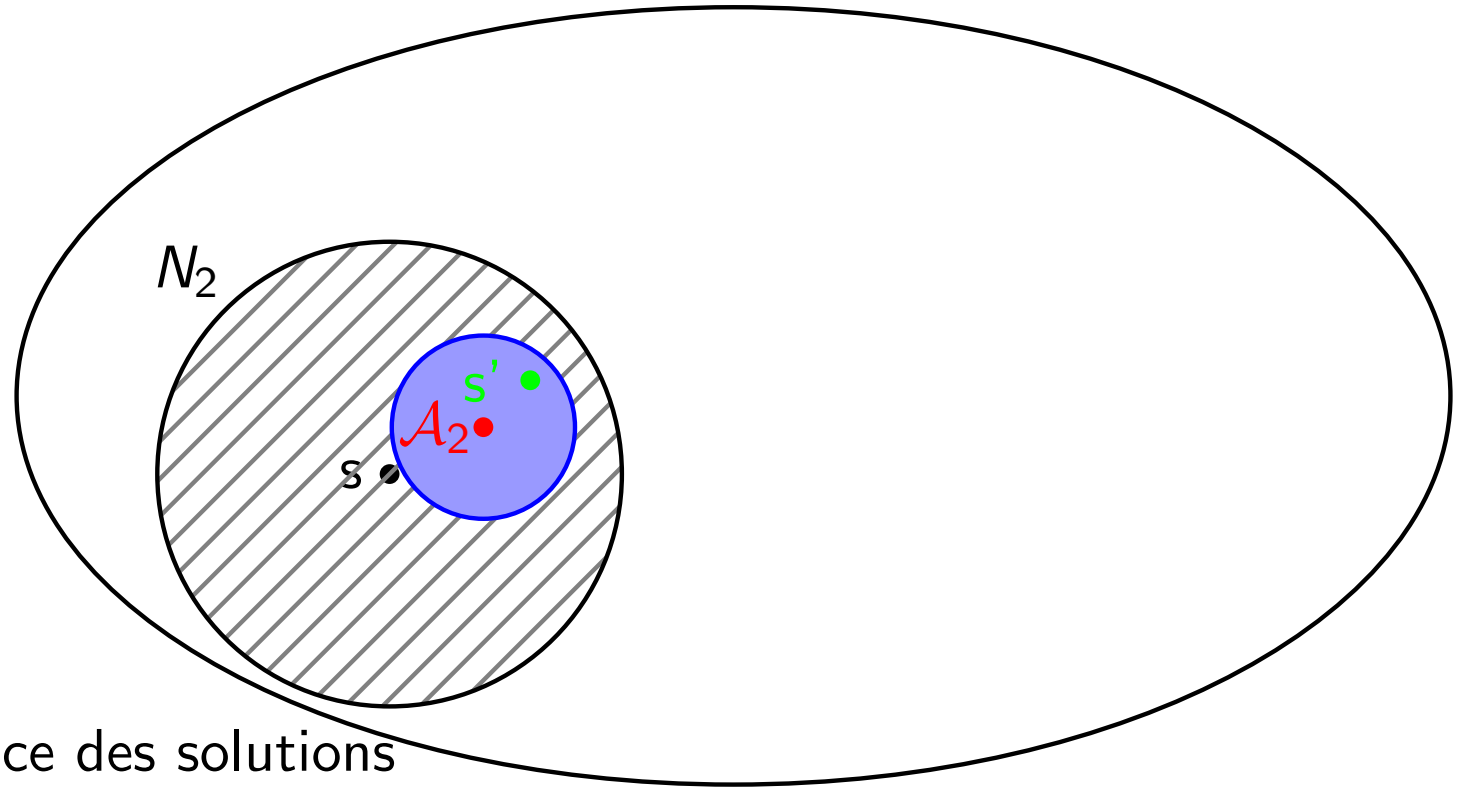
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



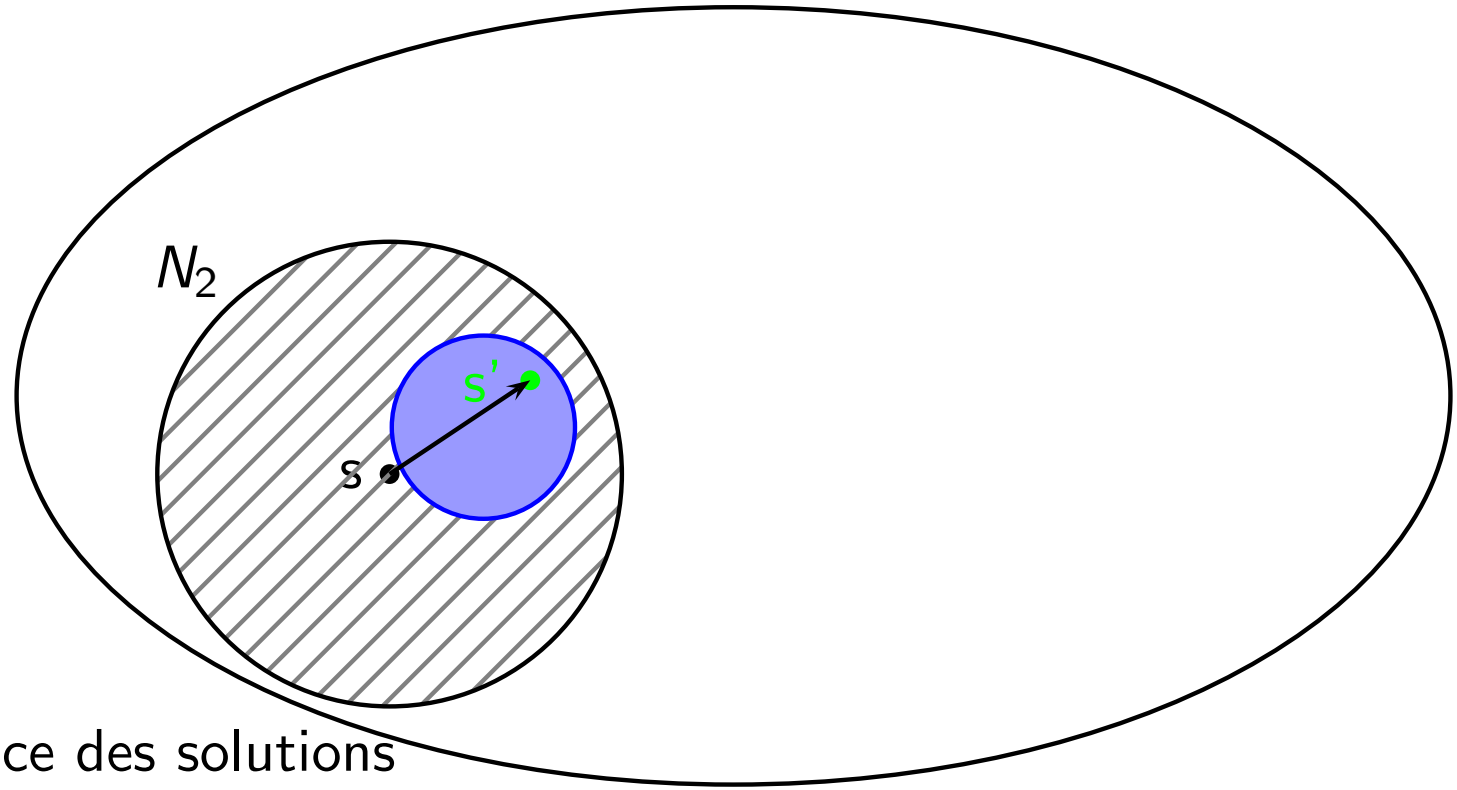
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



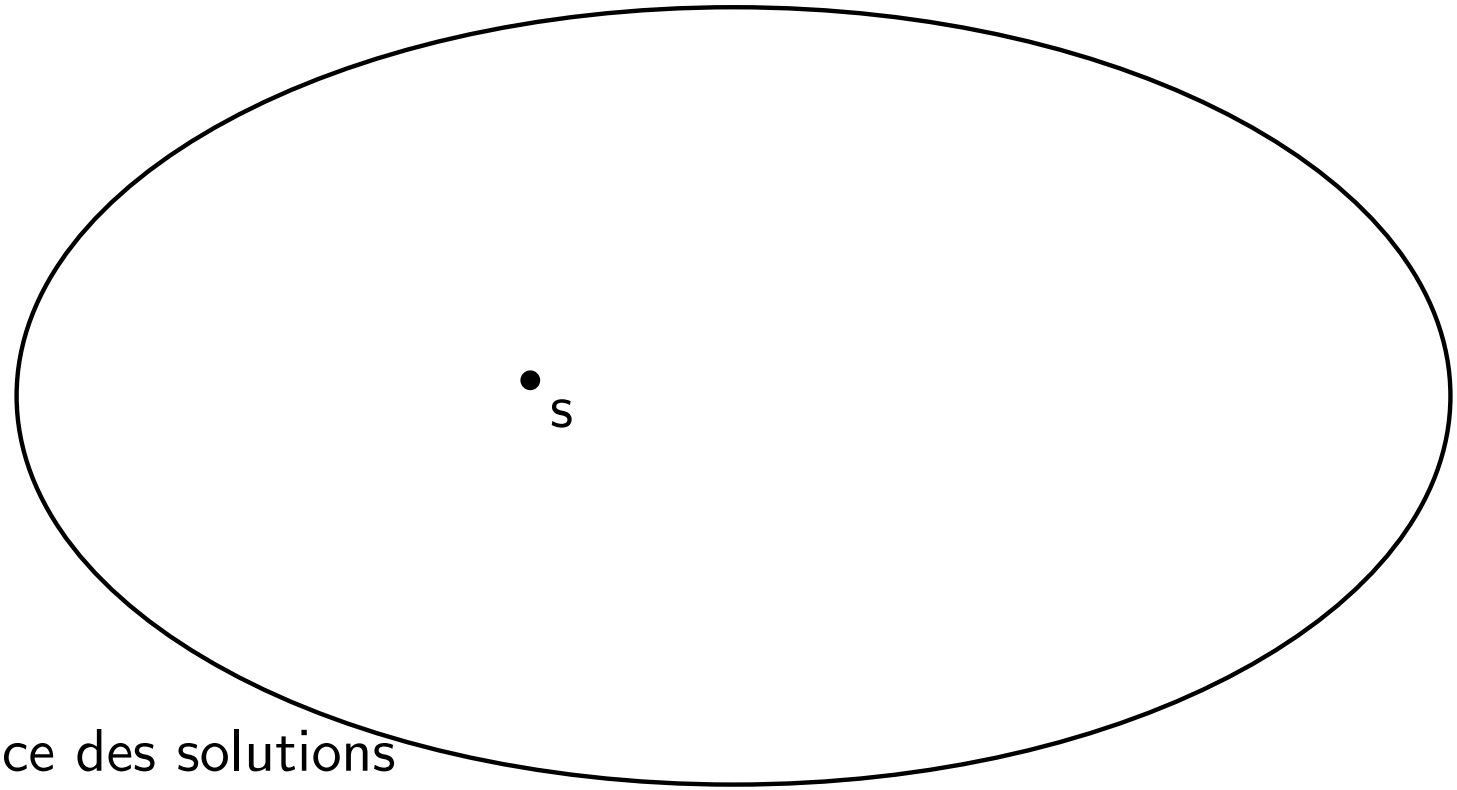
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



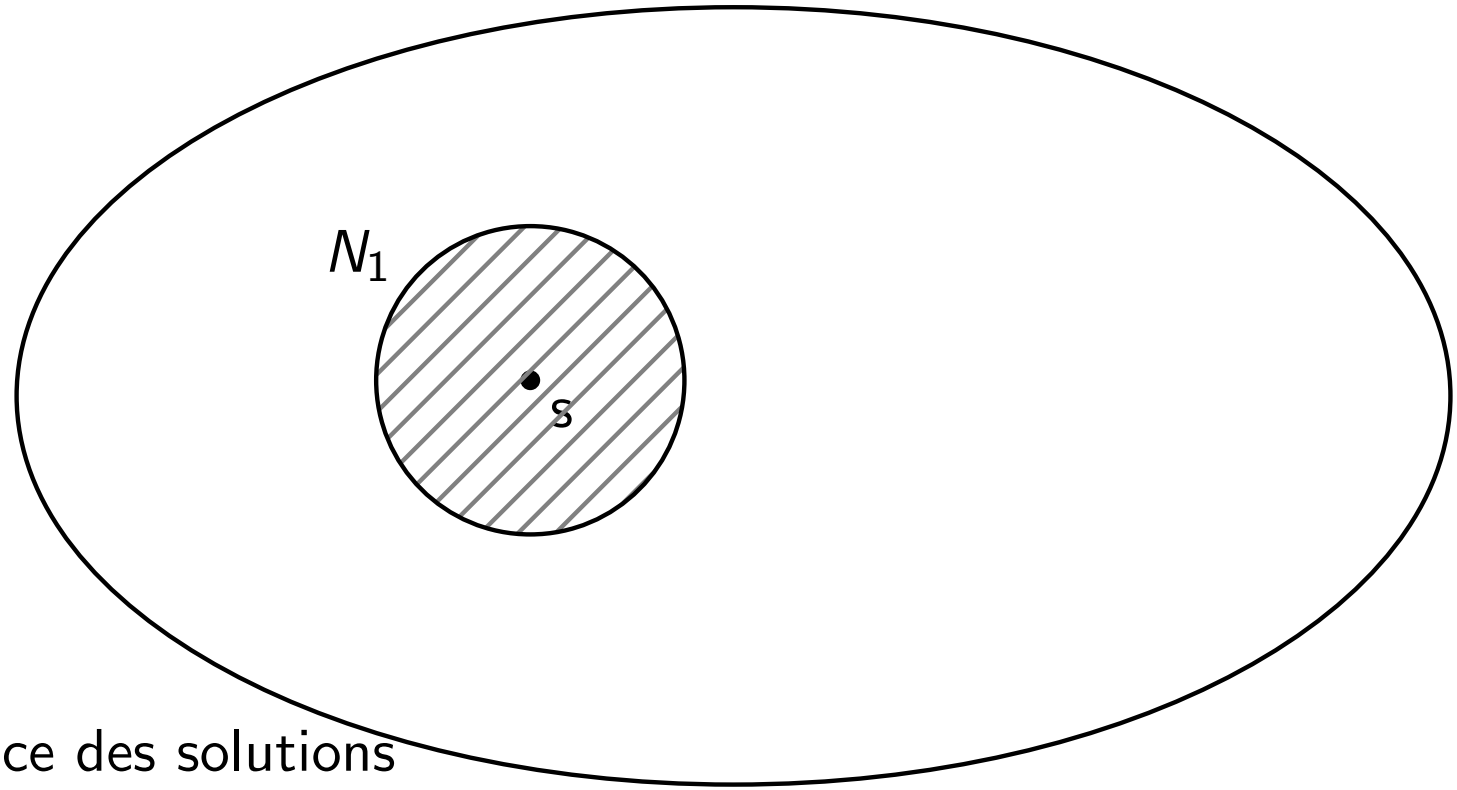
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



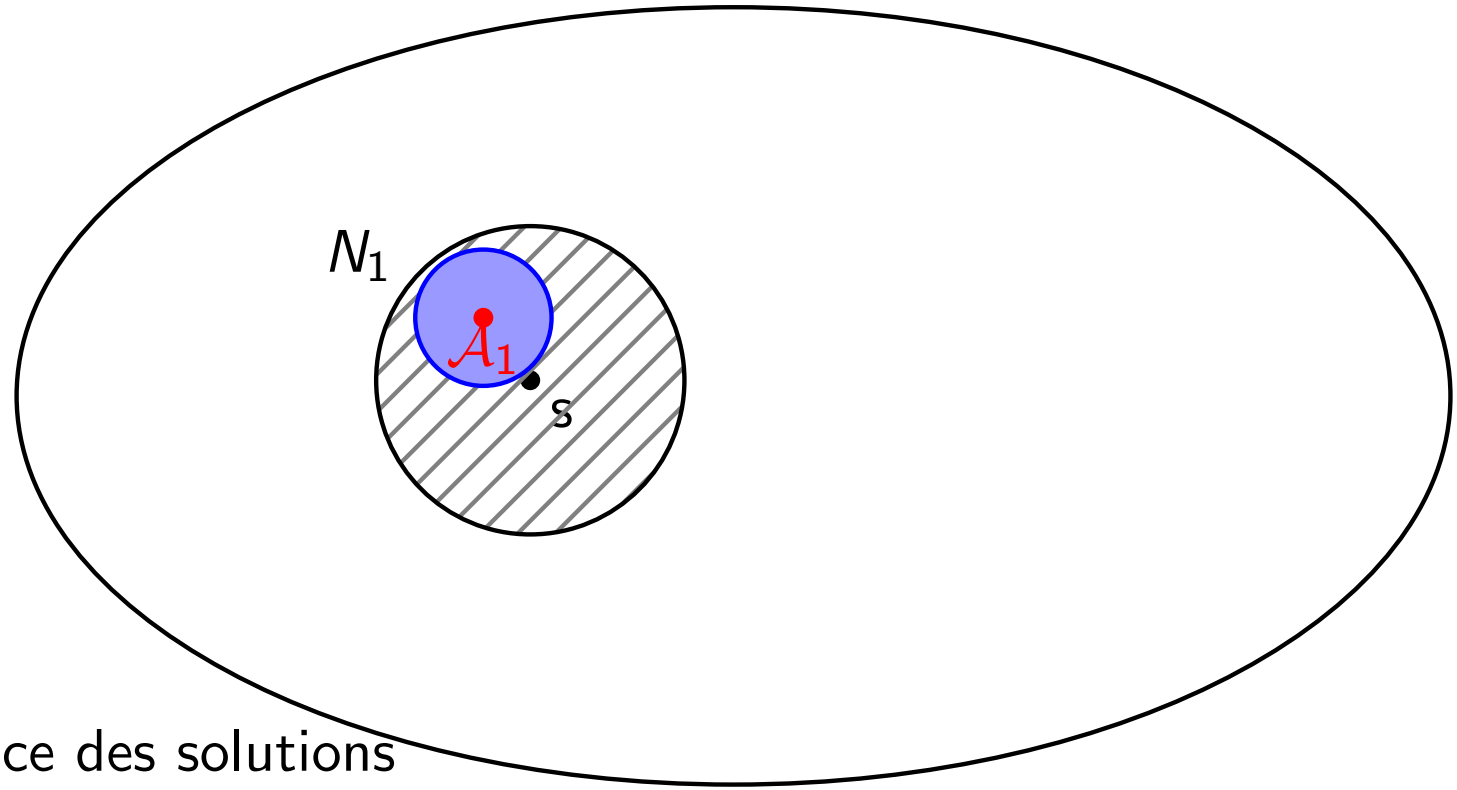
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



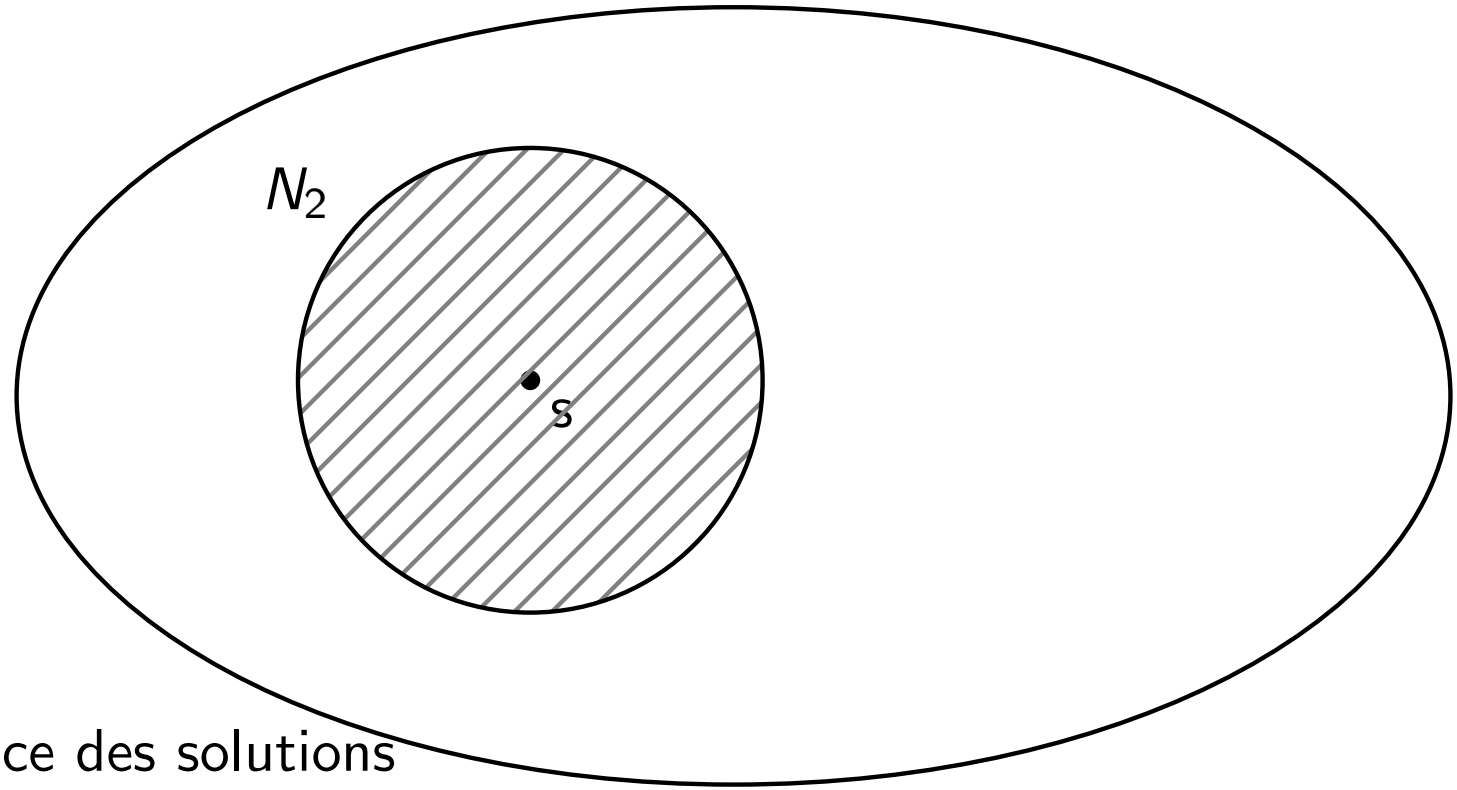
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



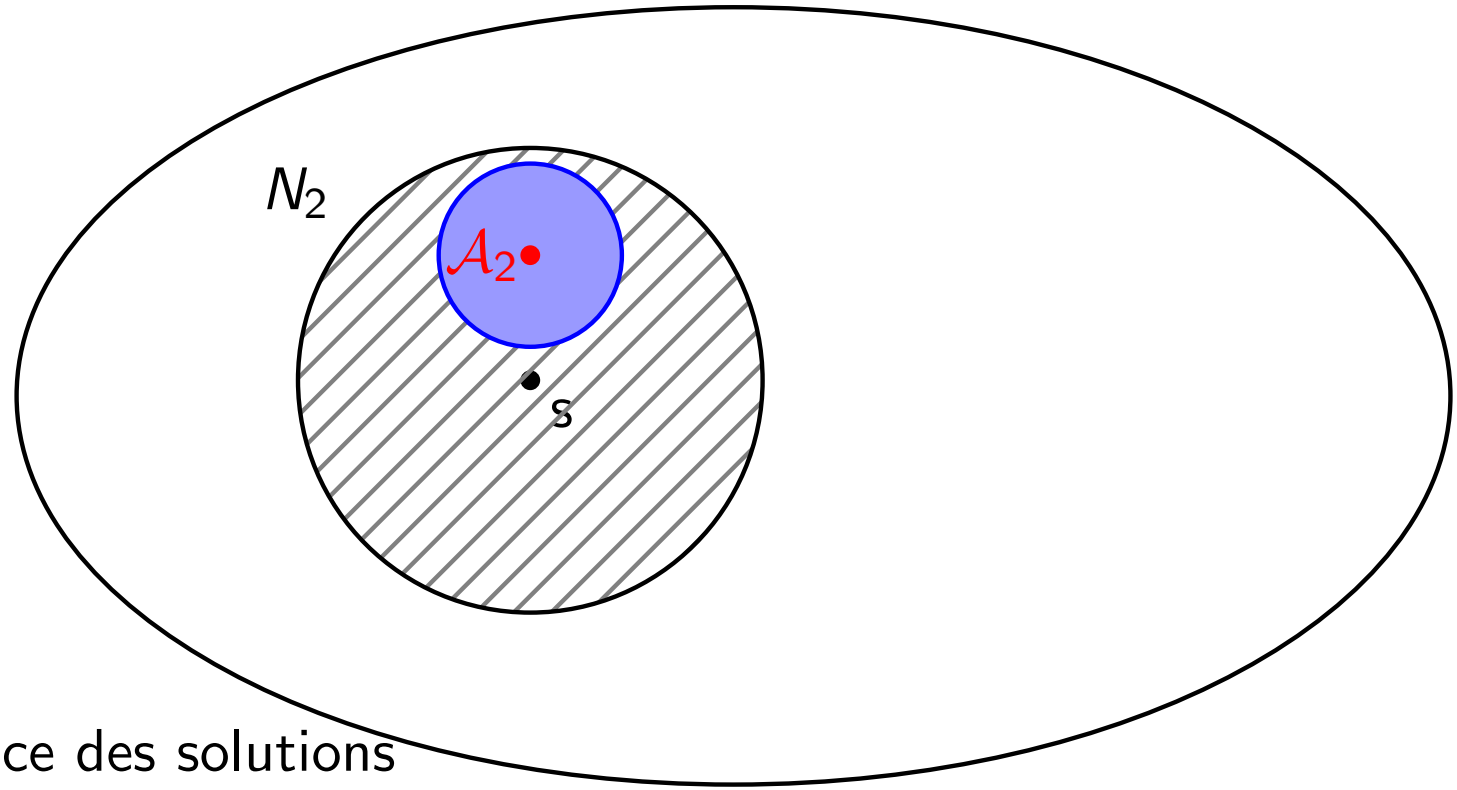
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



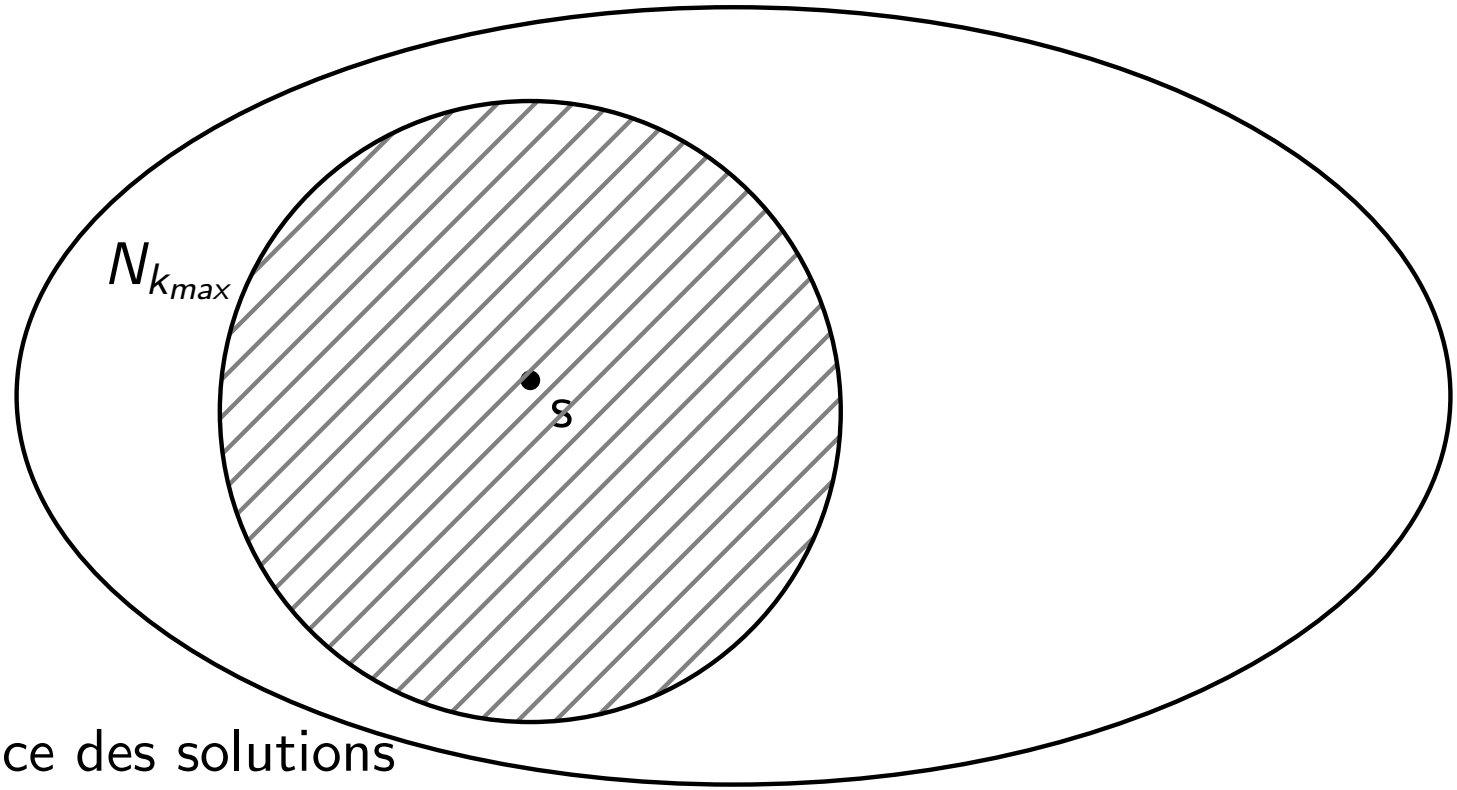
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



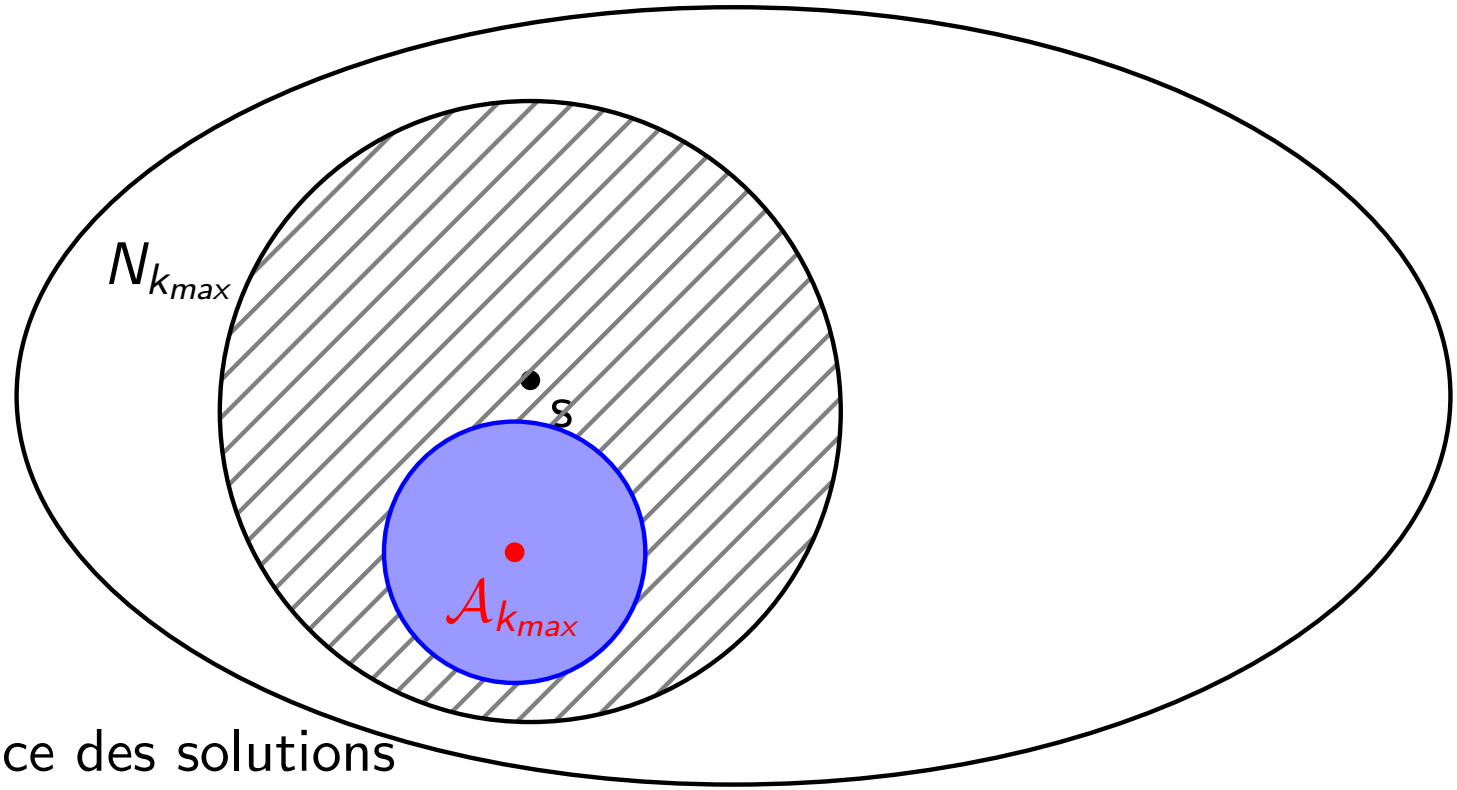
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



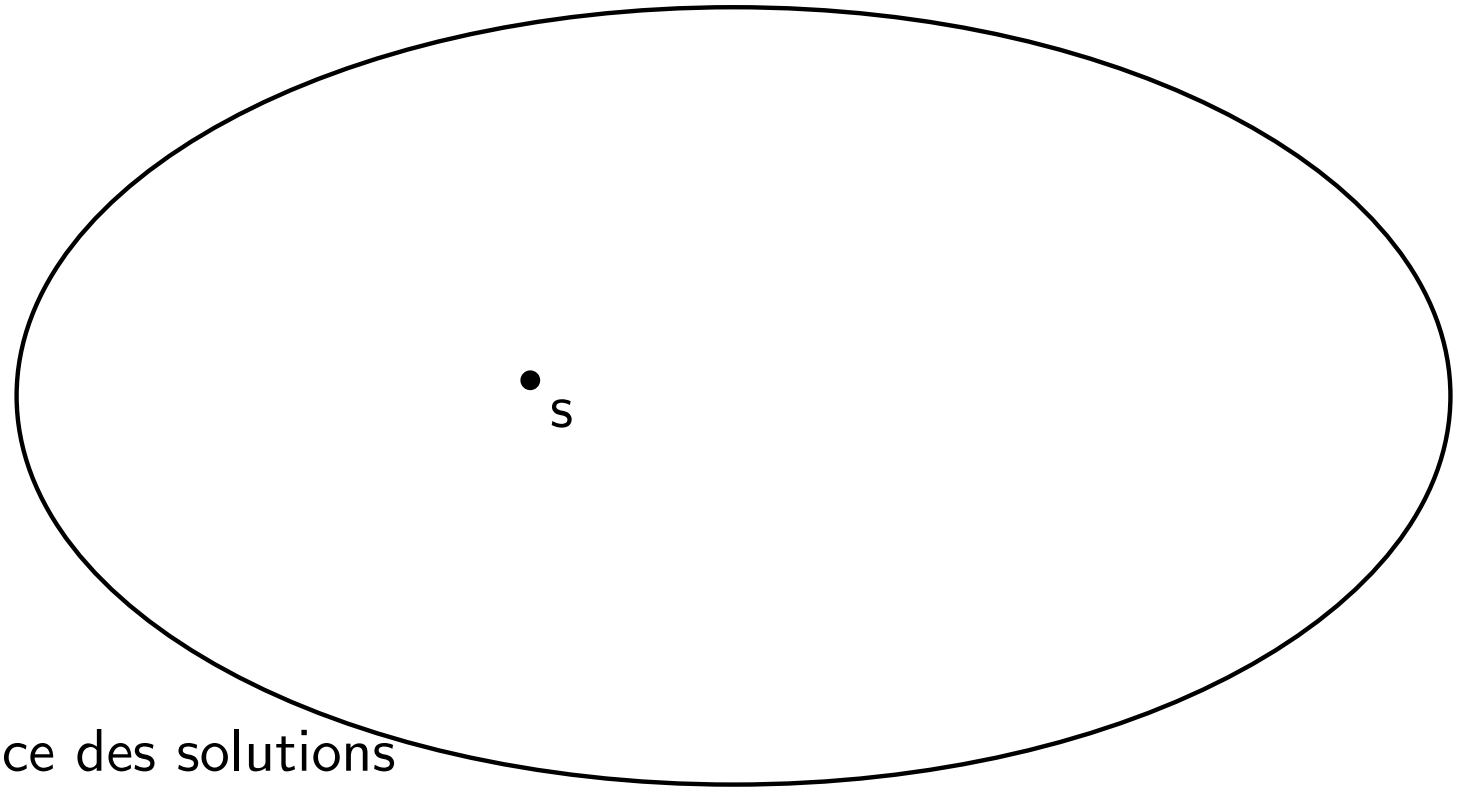
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



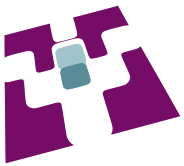
Recherche à voisinage variable (VNS, Hansen et al., 1997)

- un ensemble de voisinages $\{N_{k_{min}}, N_{k_{min+1}}, \ldots, N_{k_{max}}\}$,
- changer de voisinage durant la recherche,
- explorer chaque voisinage par une recherche locale.



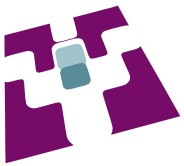
Exploration des voisinages étendus par de la PPC

Pourquoi les voisinages étendus ?



Voisinages étendus : Avantages

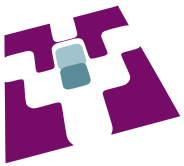
- A **larger neighborhood** means :
  - More solutions are considered
  - Better chance of avoiding local minima



Voisinages étendus : Inconvénients

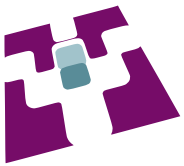
- A **larger neighborhood** also means :
  - More solutions need to be evaluated
  - The complexity of evaluating all solutions makes having neighborhoods too large unattractive

- Unless we do not evaluate all the solutions !
  - This is were Constraint Programming is useful



Constraint Programming vs. Local Search

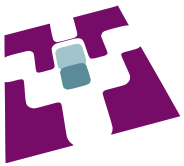
- Local search methods
  - Very fast and efficient
  - Model and search strategies closely linked
  - Complex constraints hard to model
- Constraint Programming
  - The traditional Depth-First Search strategy is too slow
  - Model and Search are completely separated
  - Complex constraints fairly easy to model
- **The ultimate goal would be to get all the advantages without the inconveniences**



Approches hybrides

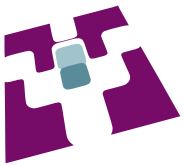
Deux classes principales d'hybridations :

1. les hybridations dites **imbriquées**, dans lesquelles RL et PPC sont étroitement liés durant la recherche,
   - hybridations imbriquées sur la base d'un algorithme complet,
   - hybridations imbriquées sur la base d'un algorithme de RL.
2. les hybridations par **compositions** (ou non imbriquées), dans lesquelles les deux mécanismes complets et incomplets cohabitent et coopèrent de façon indirecte.



Objectif : Améliorer l'efficacité d'une méthode de recherche arborescente en y greffant des mécanismes de RL.

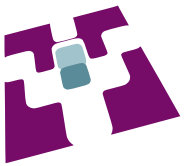
- **RL aux points de choix d'une recherche arborescente**
  appliquer une RL à certains points de choix, pour essayer de trouver rapidement de bonnes solutions, et améliorer rapidement le majorant courant de l'optimum (Prestwich, 2000).
- **recherche locale à partir des instanciations partielles**
  construire une bonne instanciation partielle à l'aide de LDS, puis compléter cette instanciation par un algorithme de recherche locale (caseau, 99).



Objectif : Utiliser des mécanismes complets pour mieux exploiter la notion de voisinage et de choix dans le voisinage.

- **Exploration de voisinages par une recherche complète** :
  - Relaxer (désinstancier) une grande partie d'une instanciation complète courante, puis essayer de la reconstruire si possible de meilleure qualité.
  - Large Neighborhood Search (LNS, Show98) utilise un LDS pour explorer le voisinage étendu.
  - LNS/CP/GR (Lobjois, 2000), dans le cadre des VCSP, utilise un glouton avec renforcement de cohérence.

$\rightarrow$ C'est de loin les hybridations les plus fructueuses.



VNS/LDS+CP
(Loudni & Boizumault, EJOR 2008)

Un *schéma d'hybridation imbriquée sur la base d'une RL* :

- une recherche locale dans un voisinage de taille variable (VNS), à base de relaxation/reconstruction d'un sous-ensemble des variables du problème ;
- une reconstruction à base de LDS combinée avec une propagation de contraintes à base de calcul de minorant, pour évaluer le coût et la légalité des *mouvements* effectués.

VNS/LDS+CP

(Loudni & Boizumault, EJOR 2008)

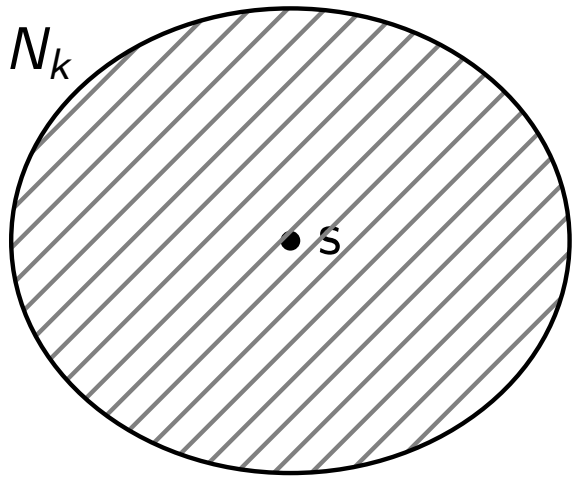
Pour k une dimension de voisinage, et s une solution courante :

VNS/LDS+CP

(Loudni & Boizumault, EJOR 2008)

Pour k une dimension de voisinage, et s une solution courante :

- N_k correspond à l'ensemble des combinaisons de k variables parmi X,

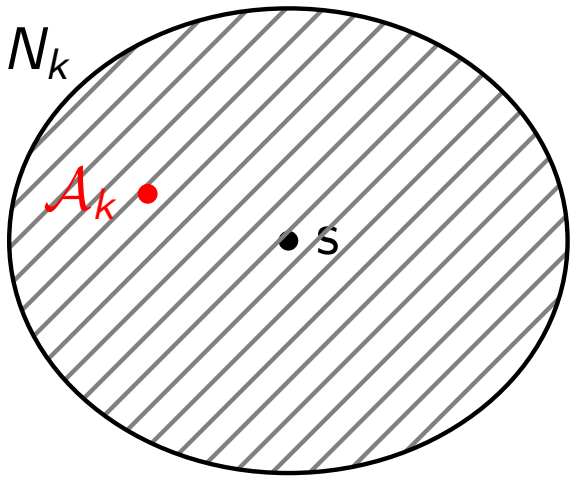


VNS/LDS+CP

(Loudni & Boizumault, EJOR 2008)

Pour k une dimension de voisinage, et s une solution courante :

- N_k correspond à l'ensemble des combinaisons de k variables parmi X,
- l'heuristique de choix de voisinage sélectionne dans N_k un sous-ensemble de k variables, noté X_r,
- une affectation partielle $\mathcal{A}_k$ est définie en désaffectant les k variables : $\mathcal{A}_k = s \setminus \{(x_i = v_i)\ t.q.\ x_i \in X_r\}$,

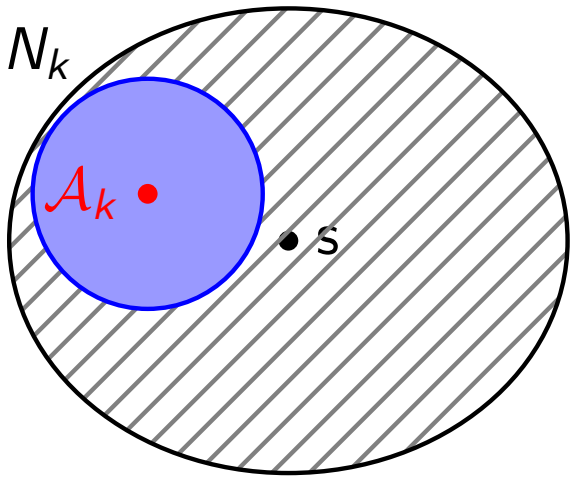


VNS/LDS+CP

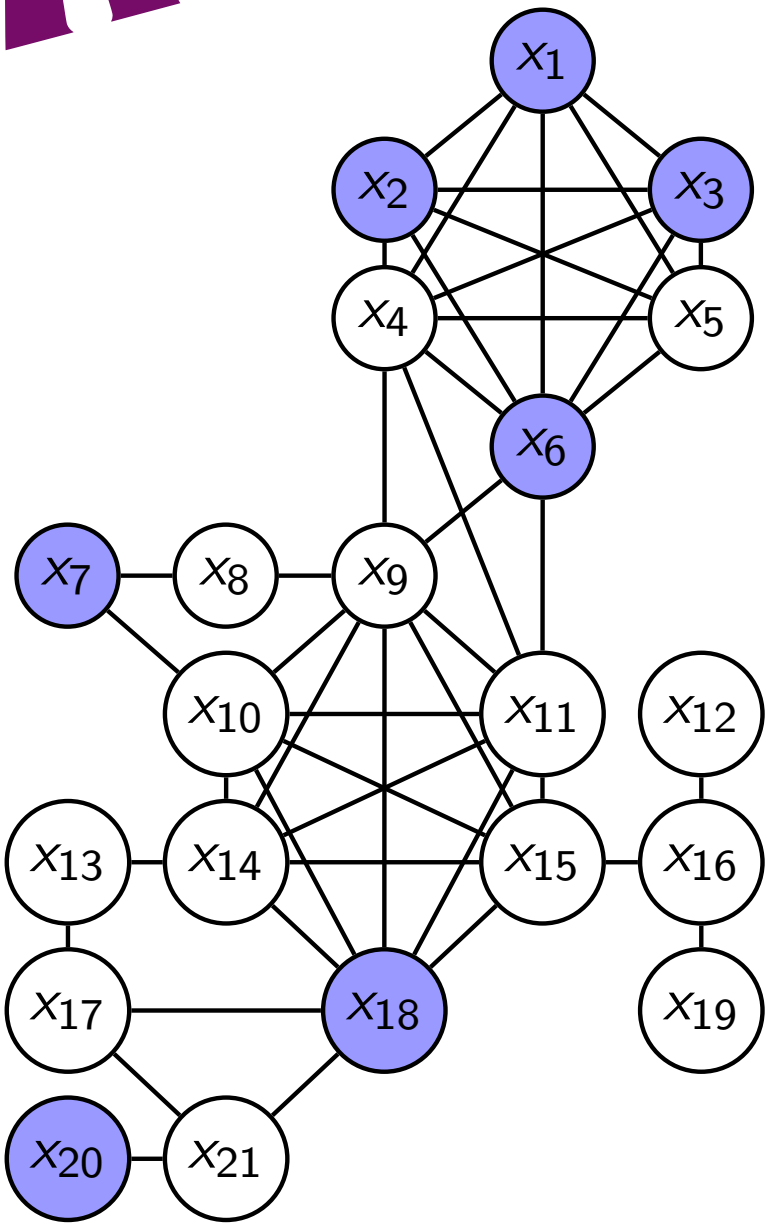
(Loudni & Boizumault, EJOR 2008)

Pour k une dimension de voisinage, et s une solution courante :

- N_k correspond à l'ensemble des combinaisons de k variables parmi X,
- l'heuristique de choix de voisinage sélectionne dans N_k un sous-ensemble de k variables, noté X_r,
- une affectation partielle $\mathcal{A}_k$ est définie en désaffectant les k variables : $\mathcal{A}_k = s \setminus \{(x_i = v_i)\ t.q.\ x_i \in X_r\}$,
- les variables désaffectées sont ensuite reconstruites par une recherche arborescente partielle (LDS+CP).



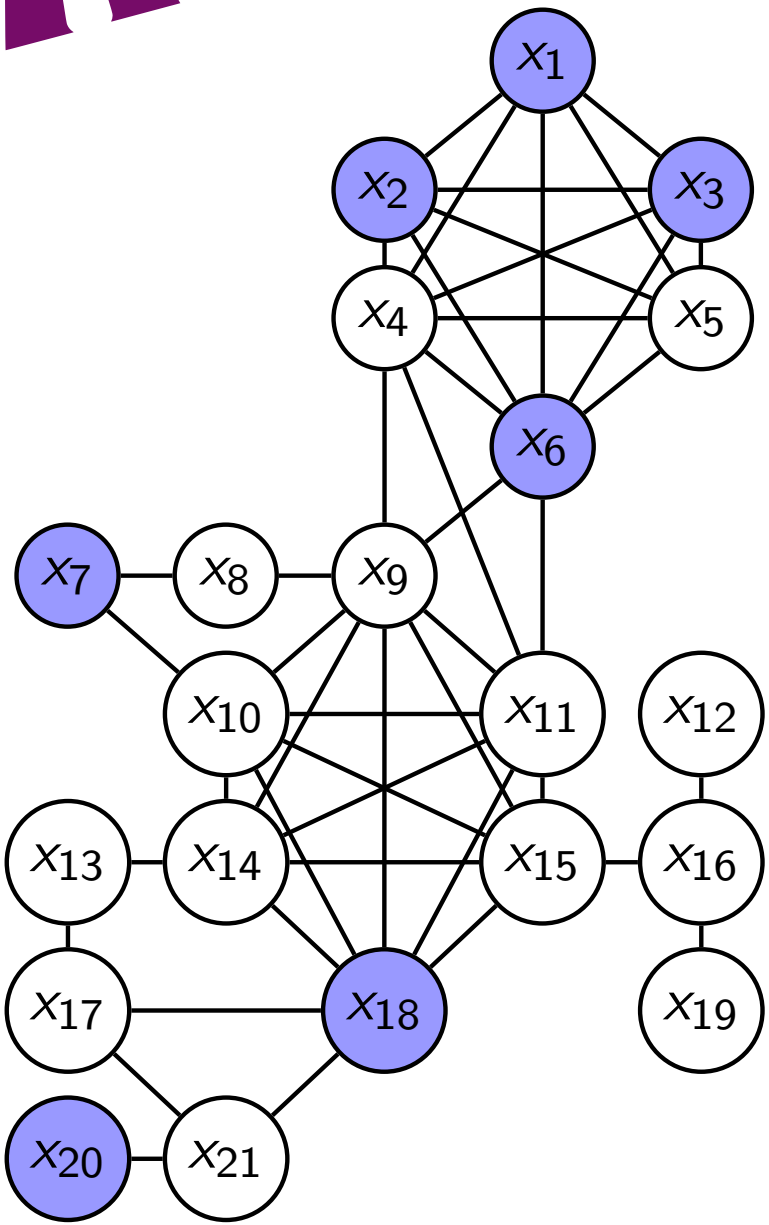
VNS/LDS+CP



1. $s = \{(x_1 = v_1), \ldots, (x_{21} = v_{21})\}$

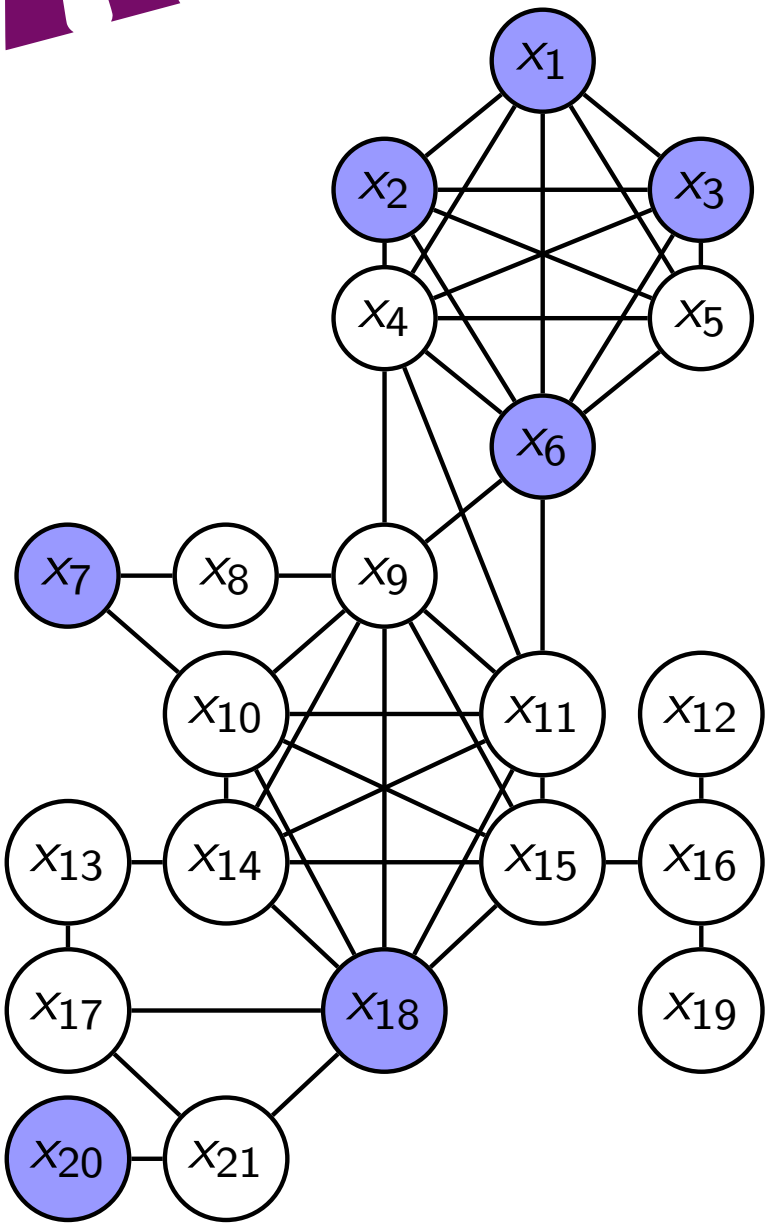
2. $X_r = \{x_1, x_2, x_3, x_6, x_7, x_{18}, x_{20}\}$

VNS/LDS+CP



1. $s = \{(x_1 = v_1), \ldots, (x_{21} = v_{21})\}$
2. $X_r = \{x_1, x_2, x_3, x_6, x_7, x_{18}, x_{20}\}$
3. $\mathcal{A} \leftarrow s \setminus \{(x_i = v_i)\ t.q.\ x_i \in X_r$

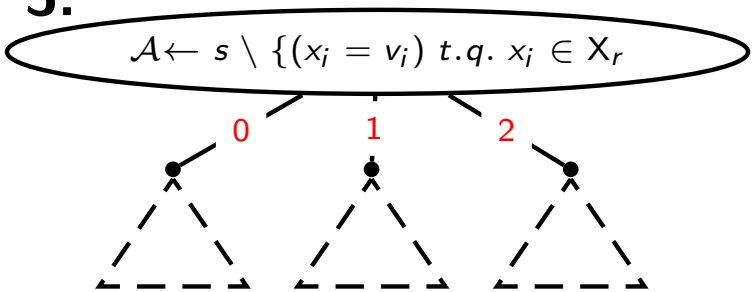
VNS/LDS+CP



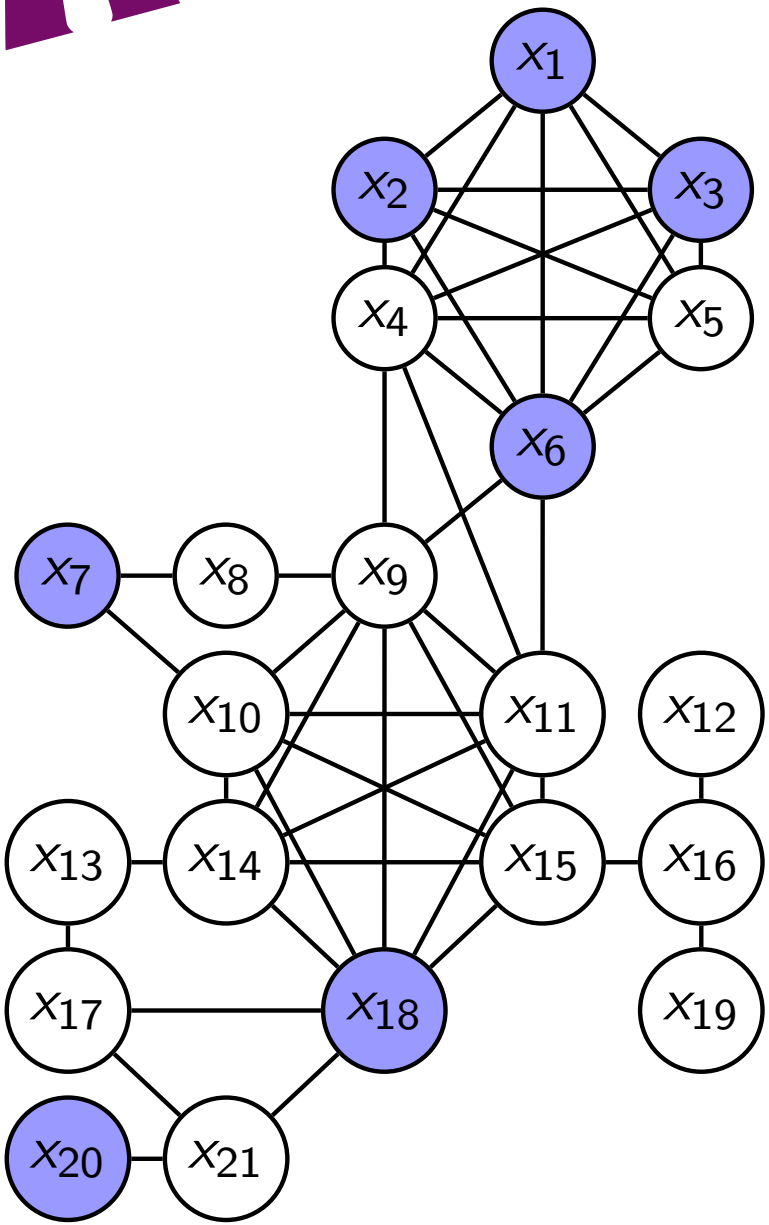
1. $s = \{(x_1 = v_1), \ldots, (x_{21} = v_{21})\}$

2. $X_r = \{x_1, x_2, x_3, x_6, x_7, x_{18}, x_{20}\}$

3.



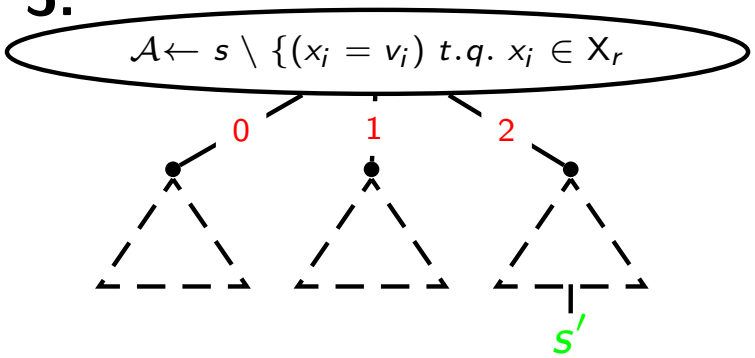
VNS/LDS+CP



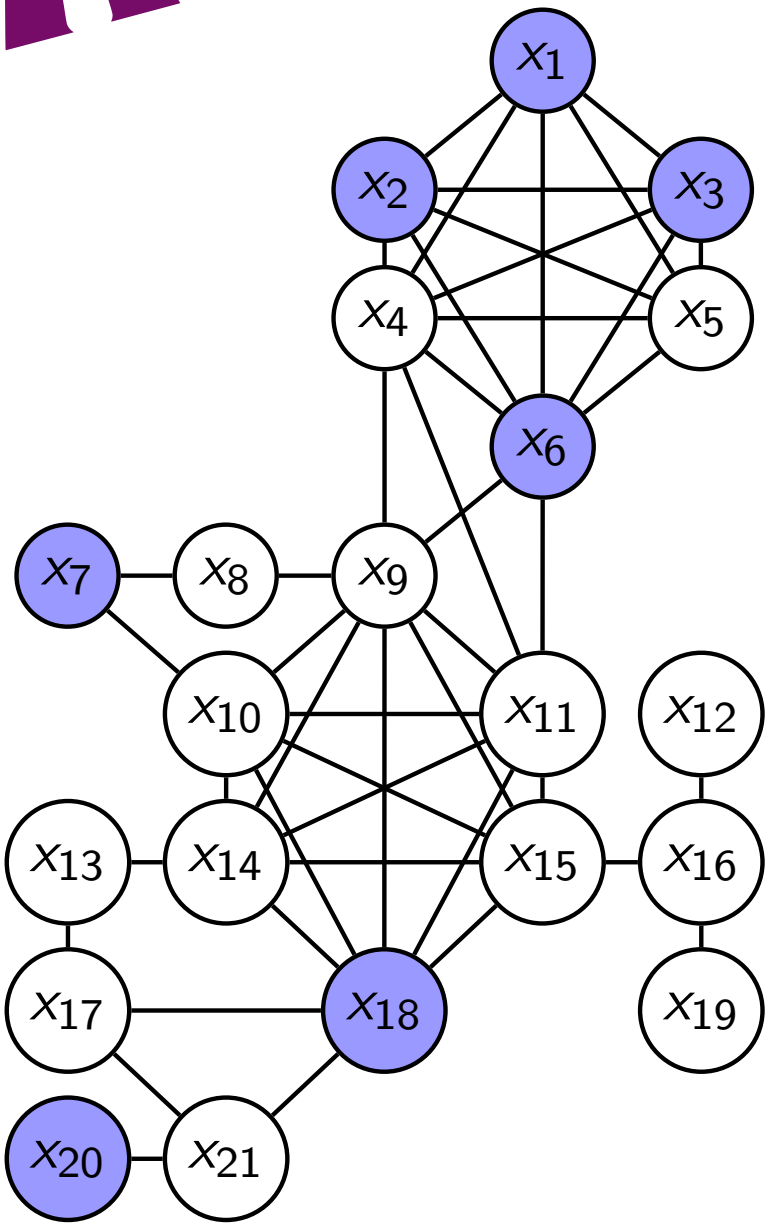
1. $s = \{(x_1 = v_1), \ldots, (x_{21} = v_{21})\}$

2. $X_r = \{x_1, x_2, x_3, x_6, x_7, x_{18}, x_{20}\}$

3.



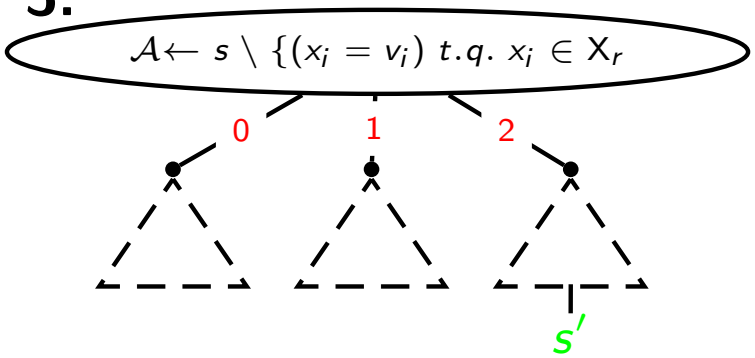
VNS/LDS+CP



1. $s = \{(x_1 = v_1), \ldots, (x_{21} = v_{21})\}$

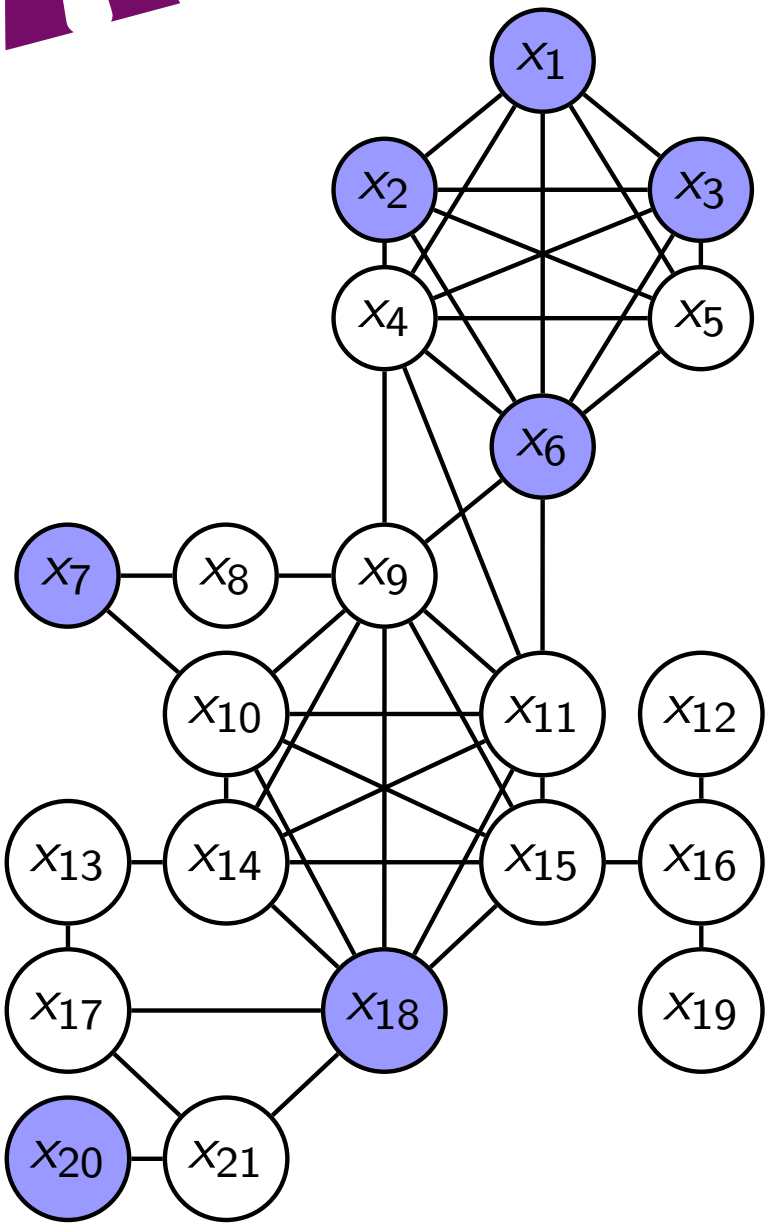
2. $X_r = \{x_1, x_2, x_3, x_6, x_7, x_{18}, x_{20}\}$

3.



4. Si $f(s') < f(s)$ **alors**
$s \leftarrow s'$ **et** $k \leftarrow k_{min}$
Sinon $k \leftarrow k + 1$

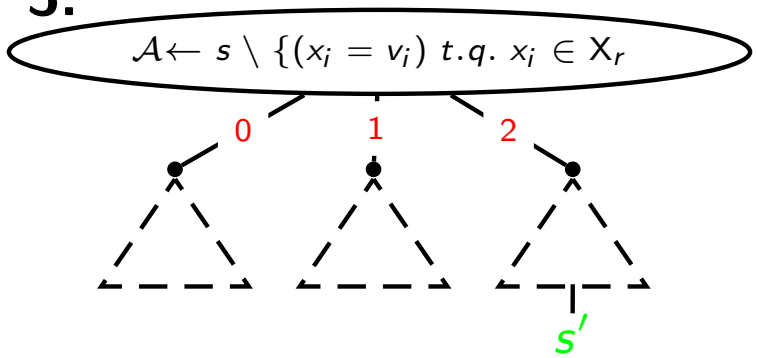
VNS/LDS+CP



1. $s = \{(x_1 = v_1), \ldots, (x_{21} = v_{21})\}$

2. $X_r = \{x_1, x_2, x_3, x_6, x_7, x_{18}, x_{20}\}$

3.



4. Si $f(s') < f(s)$ **alors**

$s \leftarrow s'$ **et** $k \leftarrow k_{min}$

Sinon $k \leftarrow k + 1$

5.

Tant que $(k \leq k_{max})$

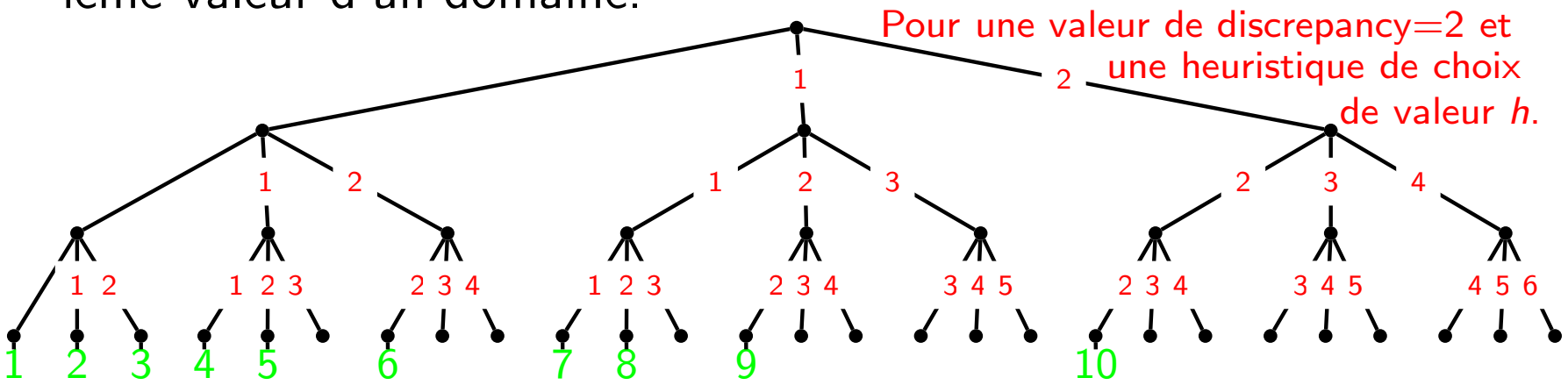
Algorithme 1: VNS/LDS+CP

```
function VNS/LDS+CP(𝒳, 𝒞, k_init, k_max, δ_max) ;
begin
1     S ← genInitSol() ;
2     k ← k_init;
3     while (k < k_max) ∧ (notTimeOut) do
4         𝒳_unassigned ← Hneighborhood(N_k, S) ;
5         A_k ← S\{(x_i = a) | x_i ∈ 𝒳_unassigned} ;
6         S' ← LDS+CP(A_k, 𝒳_unassigned, δ_max, f(S), S) ;
7         if f(S') < f(S) then
8             S ← S' ;
9             k ← k_init ; // intensification
          end
10        else k ← k + 1 ; // diversification
      end
11    return S ;
end
```

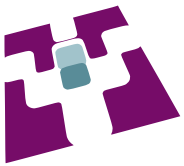
Limited Discrepancy Search

La version n-aire (LDS+CP) étendu à l'optimisation de LDS :

- effectue un parcours en profondeur d'abord,
- effectue seulement l'itération avec la valeur maximale de discrepancy,
- comptabilise une discrepancy de $(k-1)$ lors de la sélection de la k ième valeur d'un domaine.



- valeur de discrepancy
- ordre de visite des nœuds



L'heuristique ConflictVar

Peu d'heuristiques indépendantes du problème existent. Parmi celles-ci, nous pouvons citer `ConflictVar` basée sur la notion de conflit.

Définition d'une variable en conflit

Soit $\mathcal{A}$ une affectation complète, une variable est en dite en conflit lorsqu'elle figure dans au moins une contrainte violée dans $\mathcal{A}$.

Principe de `ConflictVar` : pour k la dimension du voisinage, `ConflictVar` sélectionne aléatoirement k variables à reconstruire parmi celles en conflit.

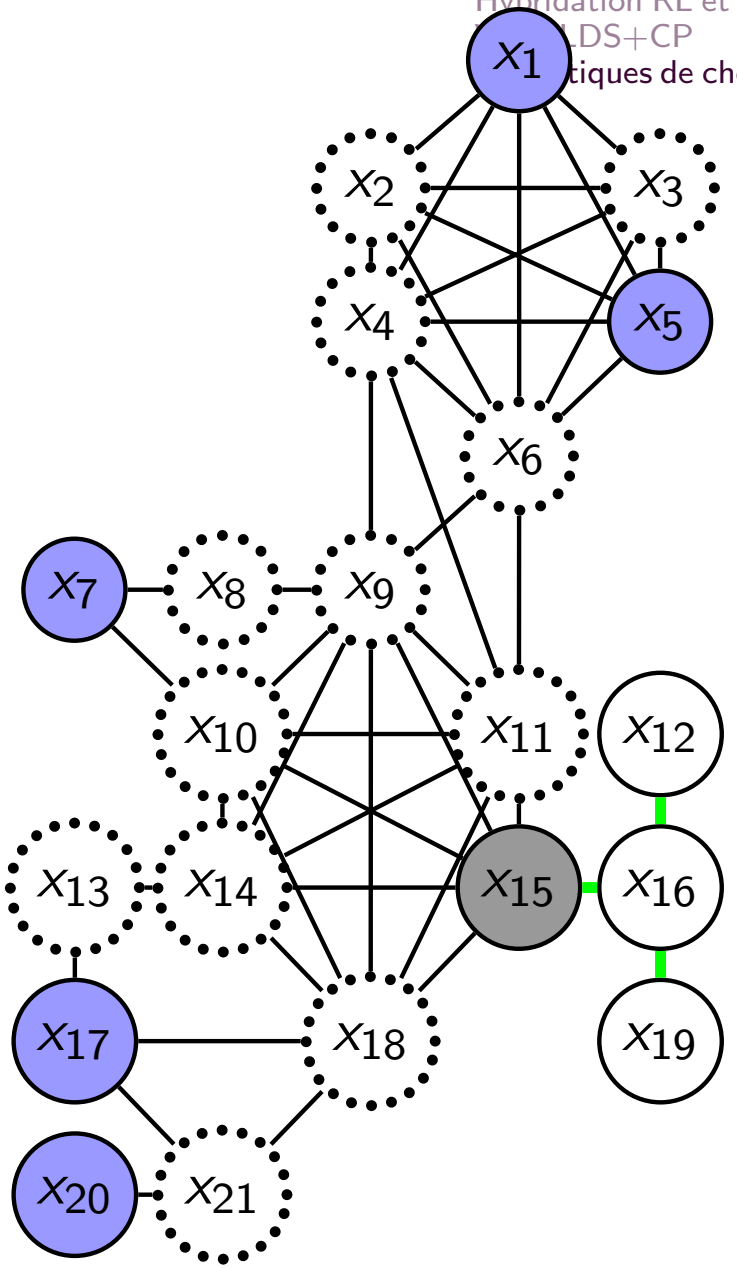
Défauts de ConflictVar

Pour $k = 6$, `ConflictVar` pourrait sélectionner les variables en bleu. Or,

- aucune contrainte n'est complètement désaffectée,
- la phase de reconstruction a peu de chances de les satisfaire.

Légende

- contraintes satisfaites
- 1ière variable sélectionnée
- variables sélectionnées
- prochaines variables pouvant être sélectionnées

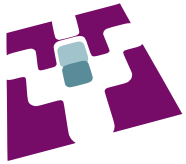


Contributions (Levasseur, Loudni, Boizumault, EAAI 2010)

Contributions : Plusieurs heuristiques exploitant outre la notion de conflit :

- la topologie du graphe de contraintes,
- le coût des contraintes.

L'objectif de ces heuristiques est de permettre au mécanisme de reconstruction de trouver de meilleures solutions.



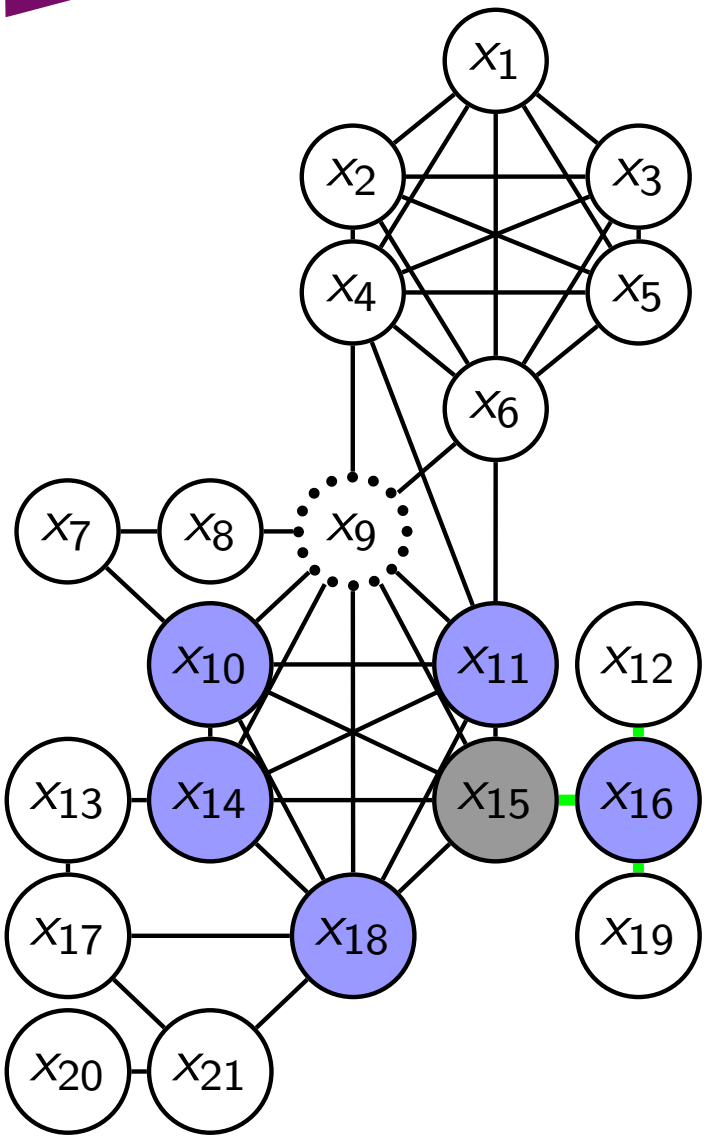
Contributions
(Levasseur, Loudni, Boizumault, EAAI 2010)

Idée de base : Maximiser le **degré de liberté** des variables à reaffecter.

Degré de liberté d'une variable

Soient X_r un ensemble de variables à reconstruire et x une variable incluse dans X_r, le degré de liberté de x est égal au nombre de variables voisines de x et incluses dans X_r.

ConflictVar-MaxDeg



Principe :

$\rightarrow$ choisit une 1ière variable en conflit,

$\rightarrow$ sélectionne ensuite les variables (en conflit ou non) ayant le plus de voisins déjà choisis.

Légende

- contraintes satisfaites
- 1ière variable sélectionnée
- variables sélectionnées
- prochaines variables pouvant être sélectionnées