Procédure Séparation et Evaluation (Branch-and-Bound/A*, RO/IA)

L'exemple du Taquin et du Voyageur de Commerce

8-puzzle = Taquin

Initiale

1	5	3
4		2
6	8	7

Finale

1	2	3
4	5	6
7	8	

Sol^n = chemin parcouru

P1

P11

1		3
4	5	2
6	8	7

P12

1	5	3
	4	2
6	8	7

P13

1	5	3
4	2	
6	8	7

P14

1	5	3
4	8	2
6		7

Algo. A*..

Distance de Mahattan.

0	1	0
0		2
3	0	2

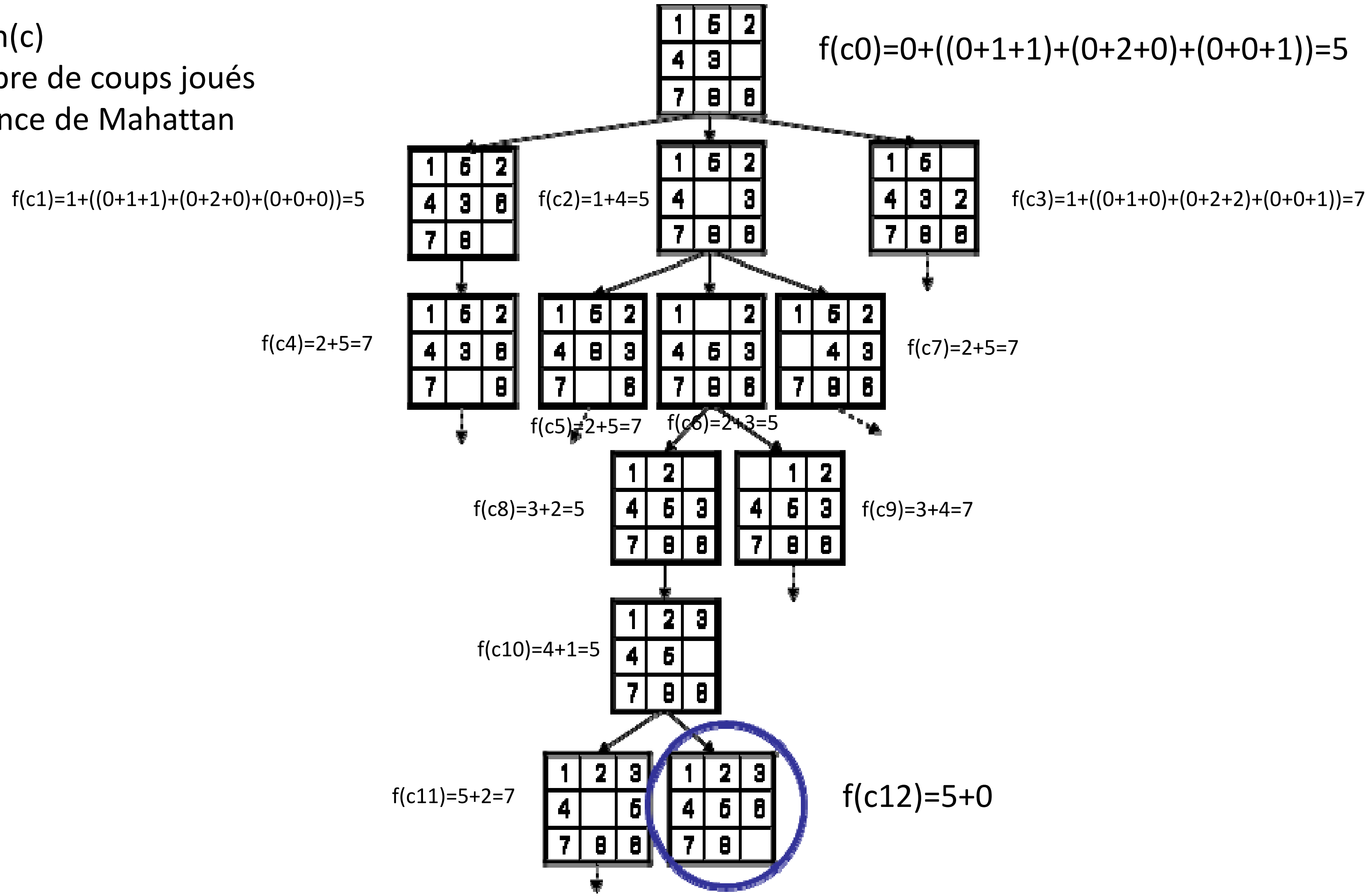
= 8 Borne inf.

Borne inf:

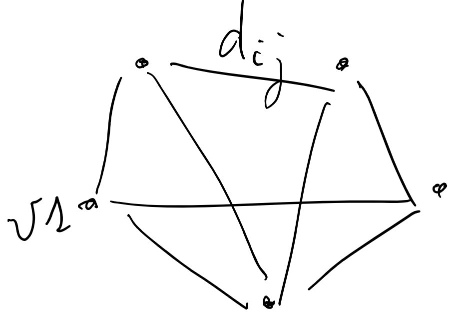
f(c)=g(c)+h(c)

g(c)=nombre de coups joués

h(c)=distance de Mahattan



Voyageur de Commerce (Traveling Salesman Problem).



$G = (V, E, d)$

NP-hard.

No Deteministic

Polynomial

$P \subset NP$

$\supset$?

$P \neq NP$

$=$?

cste

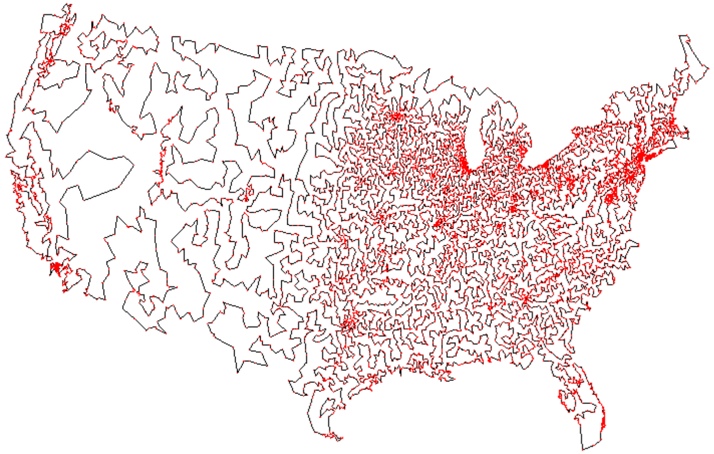
Polynomial : $O(n^k)$

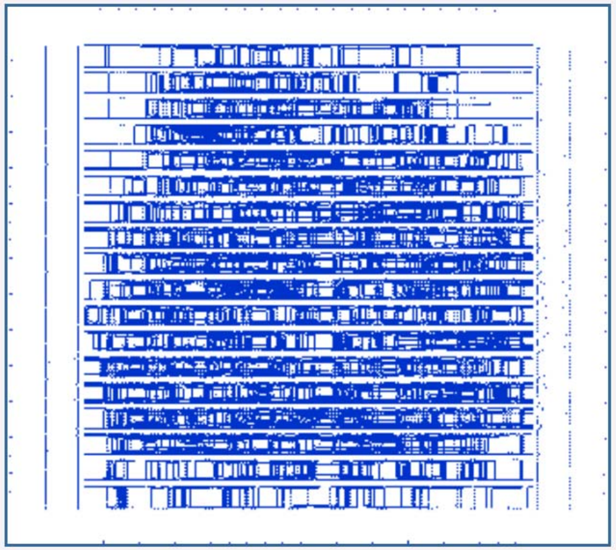
Exponential : $O(k^n)$

Traveling Salesman Problem (TSP) : a "simple" VRP

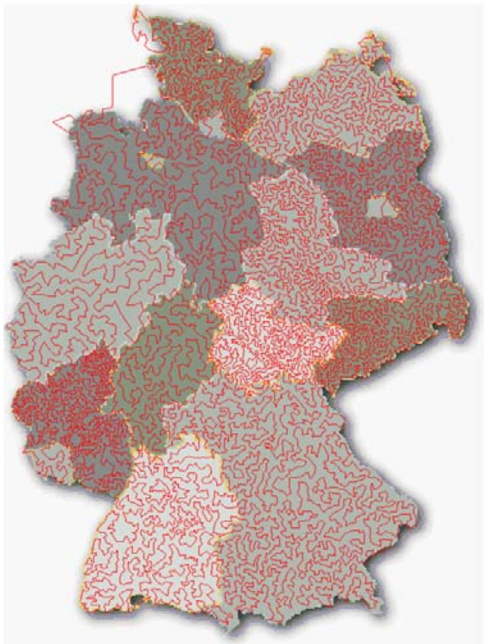
- The most well-known problem in Combinatorial Optimization
- http://www.math.uwaterloo.ca/tsp/
- The basic idea is to have a salesman traveling through N cities of a country by visiting each city once and only once.
- Number of possible solutions (N-1)!/2
  - From a given city, we have N-1 choice for the 2nd, etc.
  - Since the tour is symmetric, we need half of the solutions.

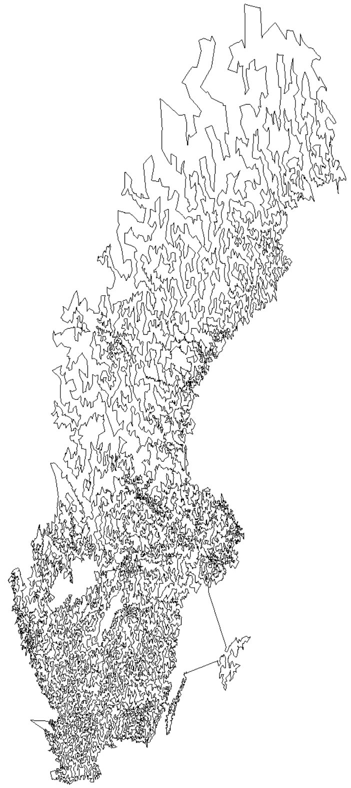
Somes examples
usa13509 (1998), d15112 (2001)
sw24978 (2004)



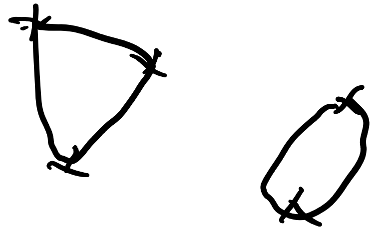


85,900 Locations in a VLSI Application
Solved in 2006





TSP formulation



$$
\begin{aligned}
z_{IP} = \min \quad & \sum_{e \in E} c_e \, x_e & & (1)\\
\text{sachant que} \quad & \sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V & & (2)\\
\sum_{e \in \Omega(S)} x_e \geq 2 \Leftrightarrow & \sum_{e \in E(S)} x_e \leq |S| - 1 \quad \forall S \subset V \ tq \ 2 \leq |S| \leq |V| - 1 & & (3)\\
& x_e \in \{0, 1\} \qquad \forall e \in E & & (4)
\end{aligned}
$$

└ cocycle

(1) Minimize the cost of the tour
(2) Constraints on the adjacency degree over each vertex of the graph
(3) Constraints on subtour elimination
(4) Constraints on the binary variables

$|V| = n \quad |\{S\}_{S \subset V}| = 2^{|V|}$

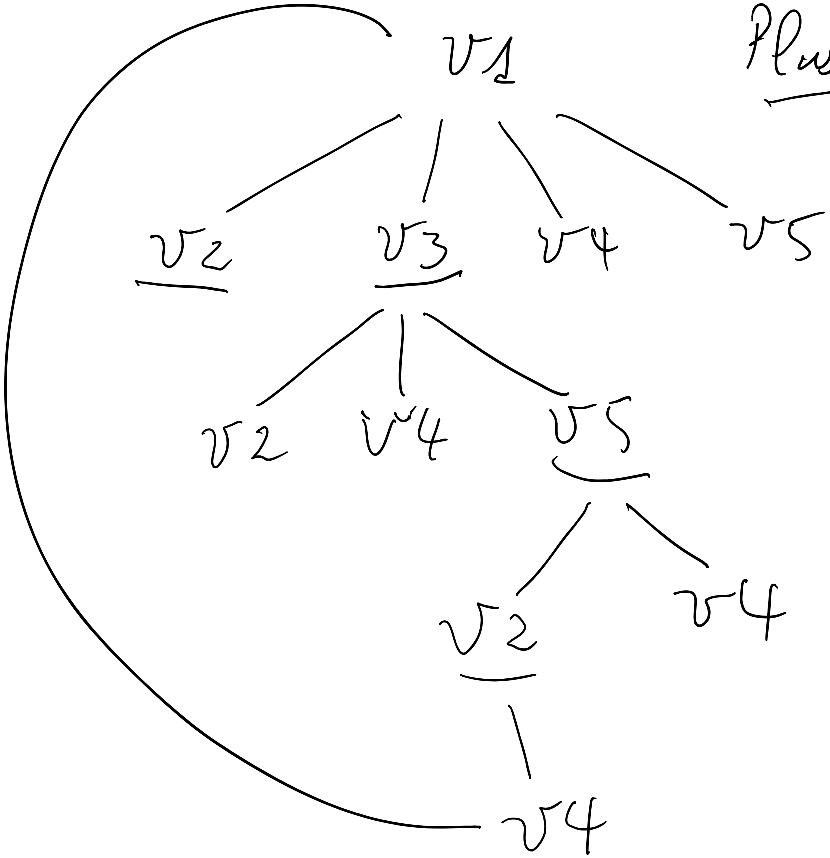
$\forall S \subset V$

TSP numerical example

$$(c_e) = \begin{pmatrix} - & 30 & 26 & 50 & 40 \\ & - & 24 & 40 & 50 \\ & & - & 24 & 26 \\ & & & - & 30 \\ & & & & - \end{pmatrix}$$

A greedy algorithm for an UB

- Choose a first city, 1 for instance.
- Take systematically the next city in the numerical order as the next city to visit (with out making a subtour)
- One solution : (1,2,3,4,5)
- Upper Bound = 30+24+24+30+40 = 148



Plus proche voisin d_{ij}

v_1 4

v_i 3

v_{i+1} 2

1

$\frac{n!}{2}$

$O(n!) > O(k^n)$.

Algorithme Glouton (Greedy)

||

DFS

Depth First Search

sans retour en arrière (back-tracking)

Combinatorial relaxation for a LB (1/3)

- For example, we can delete the subtour constraints which are very difficult to satisfied.
- We solve then a linear assignment problem (LAP): each city is visited once and only once.
- LAP could be viewed as a simple transpotation problem !

CR: LAP formulation (2/3)

$$\min \sum_i \sum_j c_{ij} x_{ij}$$

s.c.

$$\left.\begin{aligned} &\sum_i x_{ij} = 1 \qquad \forall j \\ &\sum_j x_{ij} = 1 \qquad \forall i \end{aligned}\right\} \quad (2)$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \quad (4)$$

CR : Solving using Excel (3/3)

Centres de vente						
Usines	1	2	3	4	5	Prod
1	999	30	26	50	40	1
2	30	999	24	40	50	1
3	26	24	999	24	26	1
4	50	40	24	999	30	1
5	40	50	26	30	999	1
Demandes	1	1	1	1	1	
Variables	0	1	0	0	0	1
	1	0	0	0	0	1
	0	0	0	1	0	1
	0	0	0	0	1	1
	0	0	1	0	0	1
	1	1	1	1	1	
Objectif	140					

Optimal solution methods

- Using better relaxations such as Lagrangean Relaxtion to prove that the LB=UB=148.
- If any LB $<$ UB then use a Branch-and-Bound algorithm combined with good bounding and cutting techniques to visit solutions within [LB, UB] to find the optimal solution.

	A	B	C	D	E
A	–	5	3	6	5
B	2	–	6	7	7
C	6	7	–	11	5
D	4	6	2	–	1
E	1	3	3	3	–

$$\begin{tikzcd} A \arrow[r, "3"] & C \arrow[r, "5"] & E \arrow[r, "3"] & B \arrow[r, "7"] & D \arrow[llll, bend left, "4"] \end{tikzcd}$$

Sol^{n} réalisable = 22

$\downarrow$

Borne Inf $\leq$ Optimale $\leq$ Borne Sup

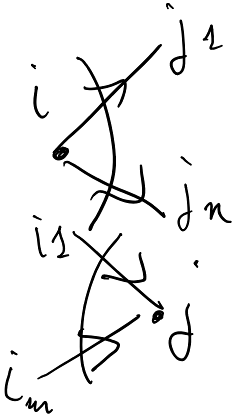
$G = (V, E, [d_{ij}])$ $\quad i \in V, j \in V, (i,j) \in E$

$$\text{Min} \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij}$$

$$x_{ij} = \begin{cases} 0 \\ 1 & \text{si } (i,j) \text{ est utilisé} \end{cases}$$

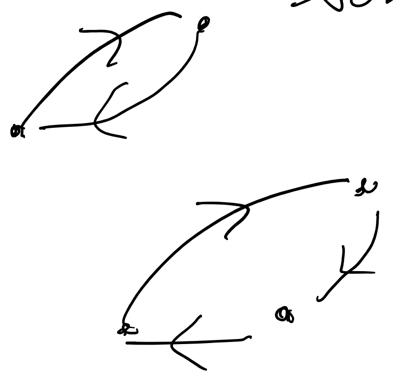
s.c. $\displaystyle\sum_{i \in V} x_{ij} = 1, \forall j \in V$

$\displaystyle\sum_{j \in V} x_{ij} = 1, \forall i \in V$



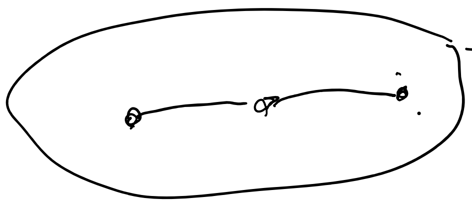
$d_{ij} \cdot x_{ij}$ est le coût associé

sous-tours



Contraintes sous-tours :

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset V \text{ tq } 2 \leq |S| \leq |V| - 1$$



$S \quad |S| = 3 - 1$

$2^{|V|}$

d_{ij}

	A	B	C	D	E
A	–	5	3	6	5
B	2	–	6	7	7
C	6	7	–	11	5
D	4	6	2	–	1
E	1	3	3	3	–
	–1	–3	–2	–3	–1

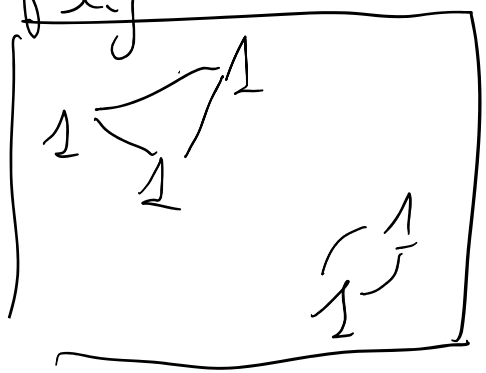
–	2	1	3	4	–1
1	–	4	4	6	–1
5	4	–	7	4	–4
3	3	0	–	0	
0	0	1	0	–	

d'_{ij}

–	1	$\boxed{0}$	2	3
$\boxed{0}$	–	3	3	5
1	$\boxed{0}$	–	3	0
3	3	0	–	$\boxed{0}$
0	0	1	$\boxed{0}$	–

$B_{Inf} = 16$

x_{ij}



$d_{ij} \cdot x_{ij}$

min ligne + min colonne

		B	C		
A	–	1	0	~~2~~ 1	3
~~B~~	0^{3}	–	3	3	5
	1	0^{0}	–	3	0^{0}
D	3	3	0^{0}	–	0^{0}
	0^{0}	0^{0}	1	0^{2}	–

~~$A \xrightarrow{0} C$~~

$A \xrightarrow{1} B$

$D \xrightarrow{0} C$

	B	C	D	E
~~A~~	–	0^{2}	2	3
C	0^{0}	–	3	0^{0}
D	3	0^{0}	–	0^{0}
E	0^{0}	1	0^{2}	–

	B	D	E
~~C~~	–	3	0^{3}
D	3	–	0^{3}
E	0^{3}	0^{3}	–

	B	D
D	3	–
E	–	0

$g = 16$ $BSup = 22 \to 19$

$\overline{BA}$: $g = 16+3 = 19$

BA: $g = 16+0 = 16$

$\overline{AC}$: $g = 16+2 = 18$

AC: $g = 16+0 = 16$

$\overline{CE}$: $g = 16+3 = 19$

CE: $g = 16+0 = 16$

ED: $g = 16+0 = 16$

DB: $g = 16+3 = 19$

	B	C	D	E	
~~A~~	~~–~~	~~–~~	~~$\cancel{2}$ 0|1~~	~~$\cancel{3}$ 1~~	–2
C	0|0	–	3	0|0	
D	3	0|0	–	0|0	
E	0|0	1	0|0	–	

	B	C	E
C	0|0	–	0|0
~~D~~	~~–~~	~~0|1~~	~~0|0~~
E	0|1	1	–

	B	E
C	–	0
E	0	–

$g = 16$

$\overline{AC}$ → $g = 16 + 2 = 18$

$\overline{AD}$ → $g = 18 + 1 = 19$

AD → $g = 18$

$\overline{DC}$ → $g = 18 + 1 = 19$

DC → $g = 18 + 0 = 18$

CE → $g = 18 + 0 = 18$

EB → $g = 18 + 0 = 18$

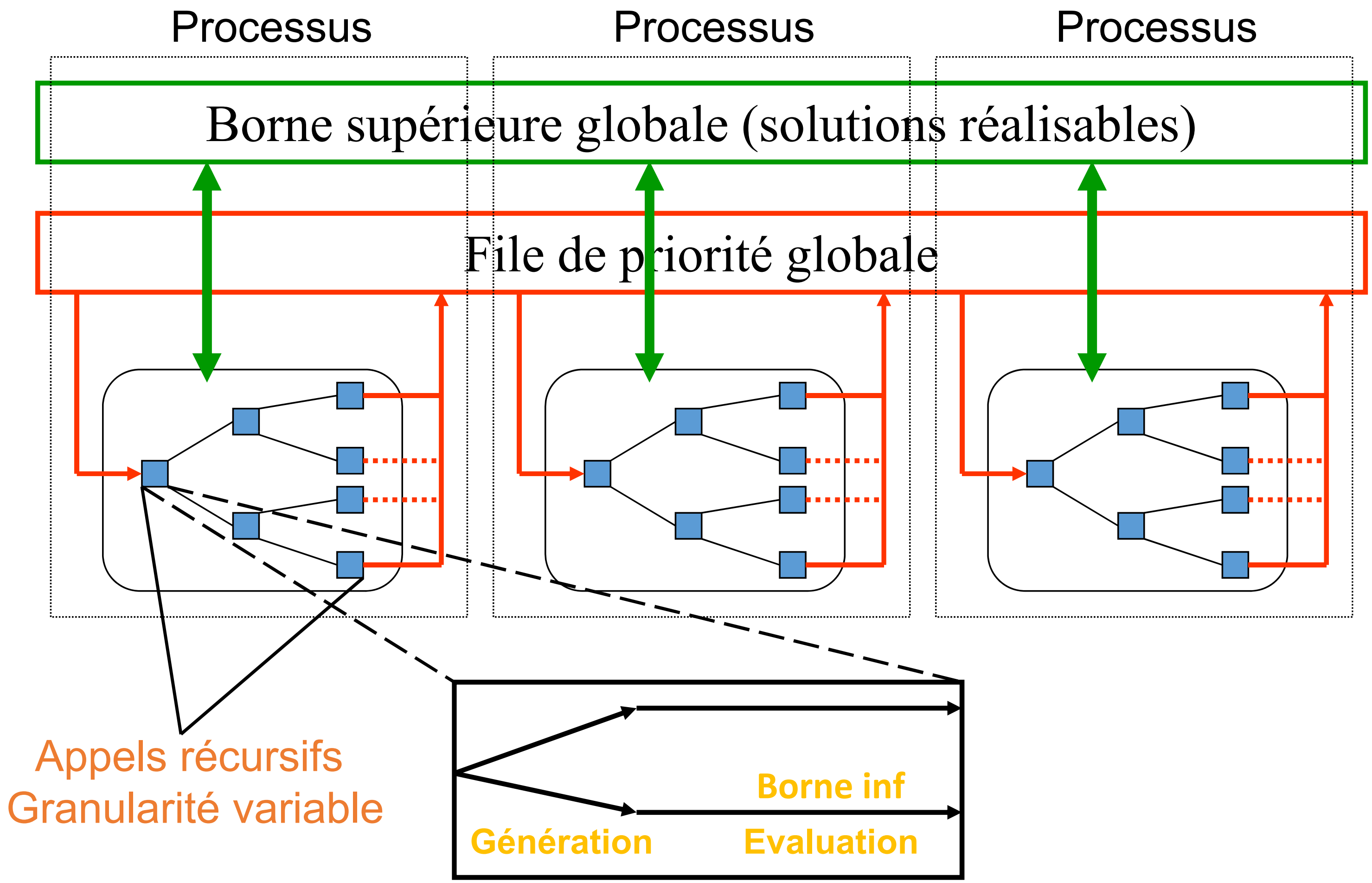
$BSup = 18$

$\Downarrow$

Sol^n optimale trouvée

```
void Procedure_SE(x0) {
 /** Minimisation **/
 ub = g(x0); // calculer une borne supérieure

 /** Créer une file de priorité h **/
 h = MakeQueue(x0);
 while (h != NULL) {  // Tant que la file de priorité n'est pas vide

   /** Choisir le meilleur sommet x à explorer selon un critère **/
   x = DeleteMin(&h);
   for (chaque fils y de x) {

     /** Màj de la borne supérieur et élaguer **/
     if (y est une solution réalisable)
        && (f(y) < ub) {  // et la valeur de y est meilleure que la borne sup.
        ub = f(y); // Maj de la borne sup.
        DeleteGreater(ub, &h); // élaguer les sommets restants à explorer
     }
     /** Ajouter les fils y dans la file de priorité **/
     if (y n'est pas un sommet terminal)
        && (f(y) < ub) // sa borne inf. est inférieure à la borne sup.
        Insert(y, &h);  // Insèrer y dans la file

   }  /* Fin for */
 } /* Fin while */
} /* Fin SE */
```

Modèle d'exécution de la procédure SE



Transposition au problèmede Découpe 2D ?

- Quel critère de séparation ?
- Calcul de borne inférieure (évaluation) ?

- Stratégies de parcours:
  - Depth First Search (Profondeur d'abord) ?
  - Breadth First Search (Largeur d'abord) ?
  - Best First Search (Meilleur d'abord) ?
  - Random Search (Recherche aléatoire) ?
  - Beam Search ?
  - Avec Look Ahead ?
  - Etc.
- File de priorité (AVL, Tas, PriorityQueue de Java, Listes, etc.) ?

Quelques mesures de performance

- Qualité des solutions obtenue par rapport à l'objectif fixé:
  - (BorneSup-Optimale)/Optimale (qd on connait une solution optimale)
  - (BorneSup-BorneInf)/BorneInf (à défaut d'une solution optimale)
- Temps CPU
- Sur l'arborescence de recherche (attention à l'explosion combinatoire):
  - Nombre de sommets générés (insertions)
  - Nombre de sommets max à un instant donné
  - Nombre de deleteMin (suppression du meilleur)