Optimisation multi-objectifs : un tutoriel

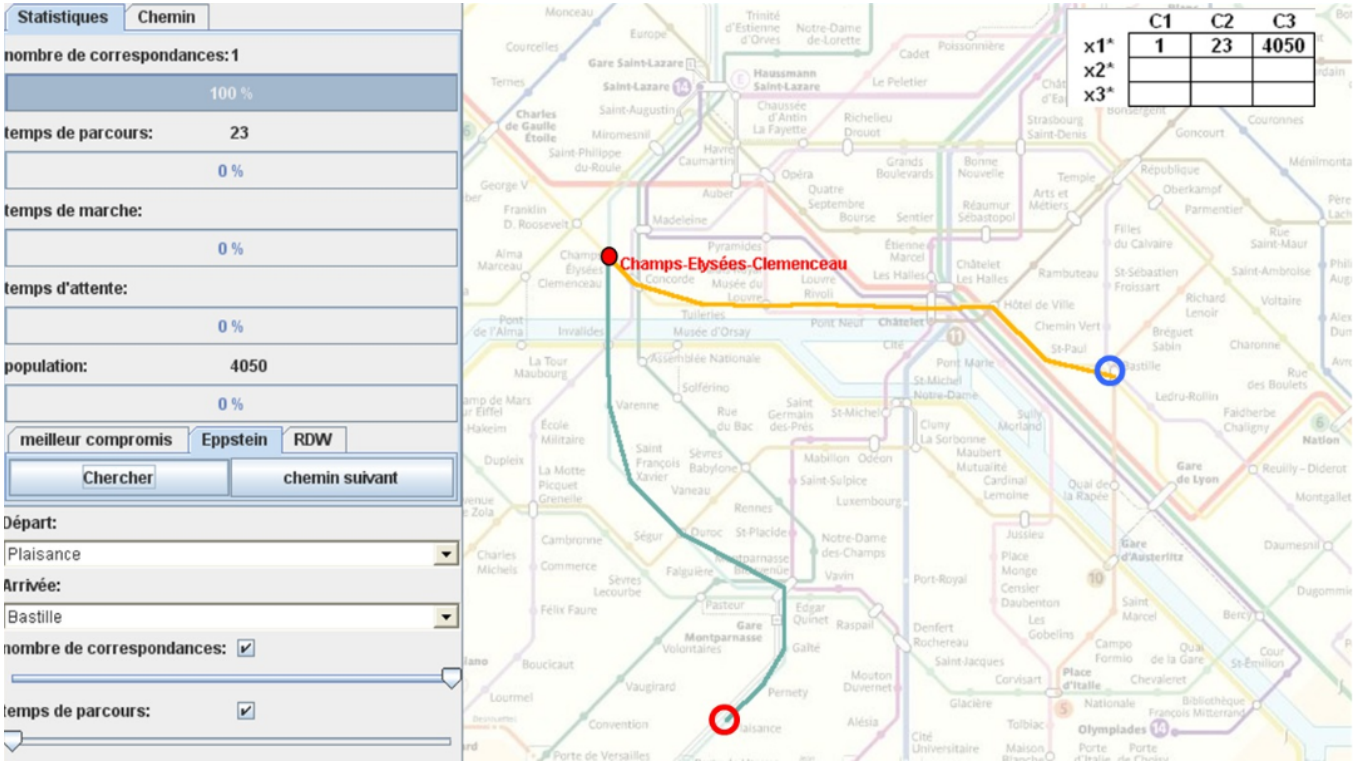
olivier.spanjaard@lip6.fr

33ème journée JFRO - 23 juin 2015

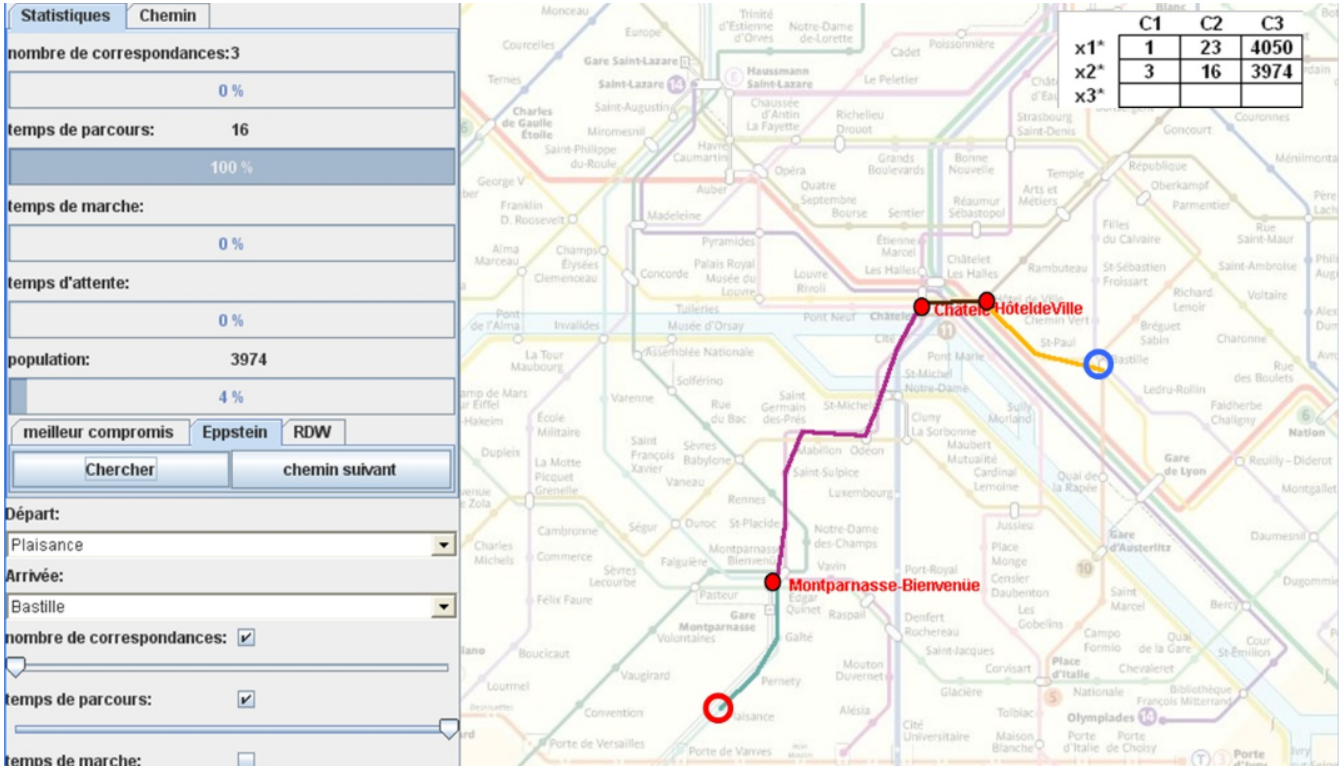
Exemples introductifs

Exemple 1 : optimisation multicritère

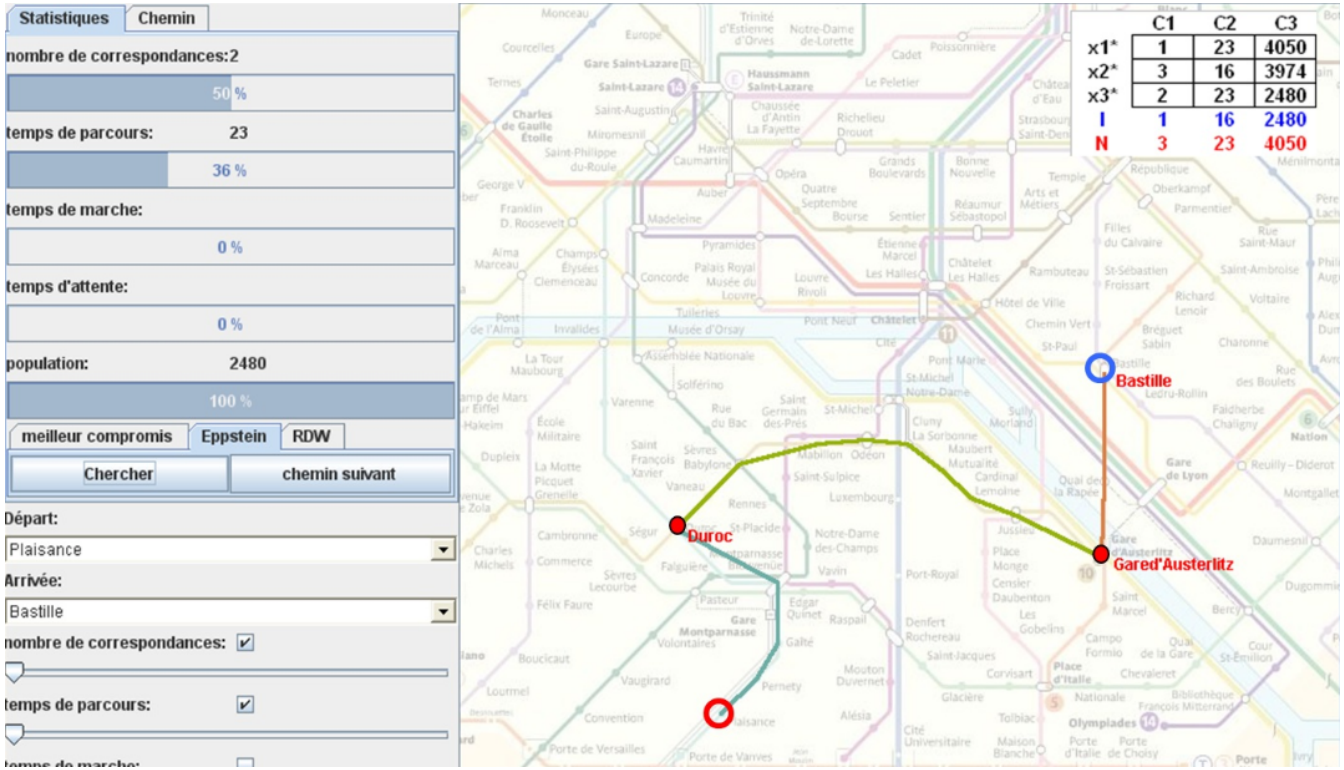
Exemple 1 : optimisation multicritère



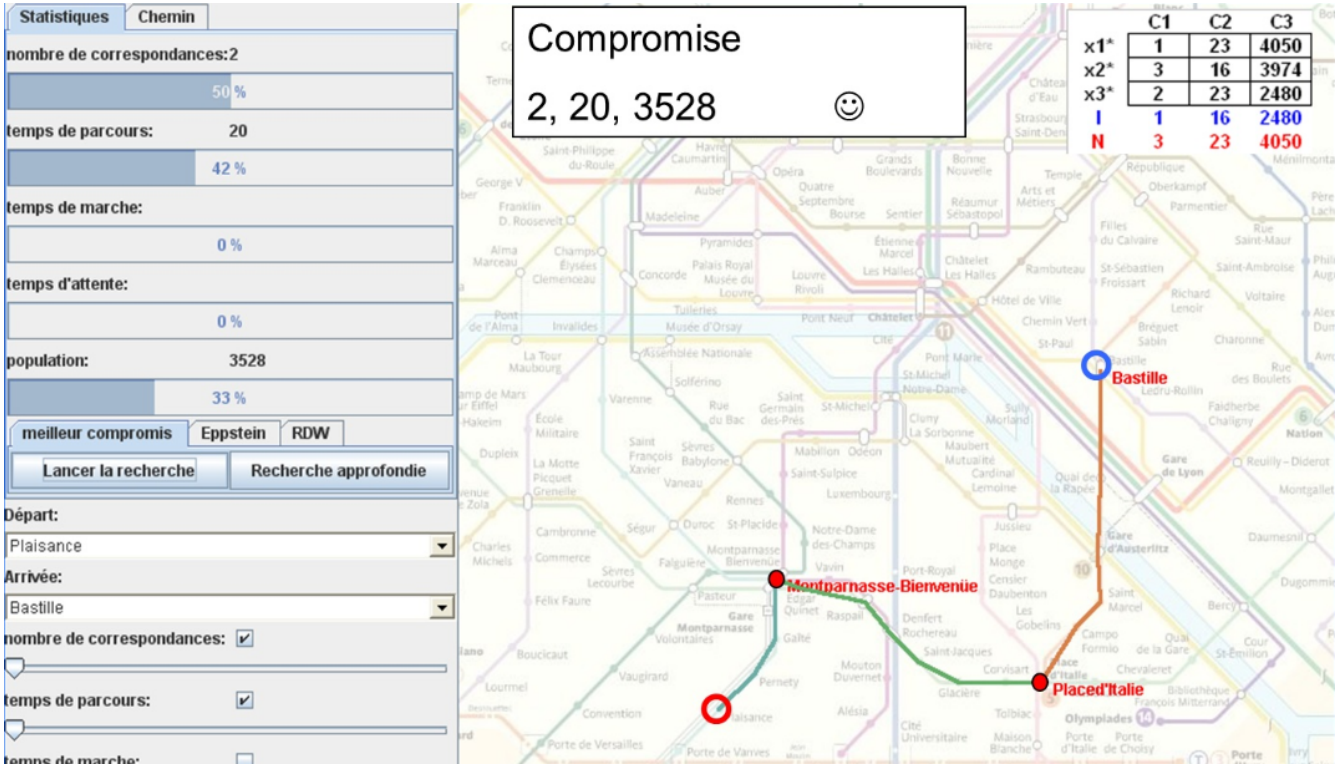
Exemple 1 : optimisation multicritère



Exemple 1 : optimisation multicritère



Exemple 1 : optimisation multicritère



Exemple 2 : optimisation ordinale

Transfert de footballeurs :

- Recruter k = 2 joueurs, avec un budget B = 6
- Joueurs sur la *short-list* :

top joueur

j_1

$w_1 = 5$

international

j_2

$w_2 = 2$

international

j_3

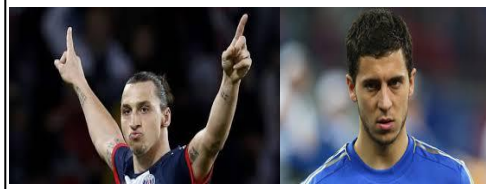
$w_3 = 4$

joueur de club

j_4

$w_4 = 1$

Exemple 2 : optimisation ordinale

	j_1	j_2	j_3	j_4	recrutements possibles (B = 6)				choix	
coût	5	2	4	1	$\{j_1, j_4\}$	$\{j_2, j_3\}$	$\{j_2, j_4\}$	$\{j_3, j_4\}$		
T: 8, I: 4, C: 1	8	4	4	1	9	8	5	5	$\{j_1, j_4\}$	
T: 8, I: 5, C: 1	8	5	5	1	9	10	6	6	$\{j_2, j_3\}$	

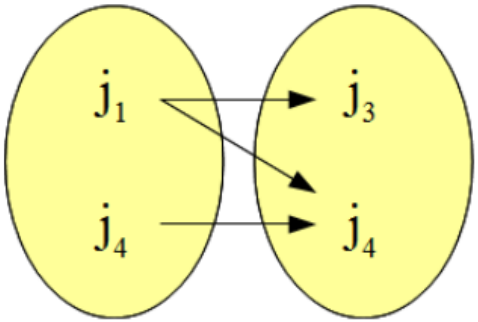
Besoin de définir des préférences ordinales

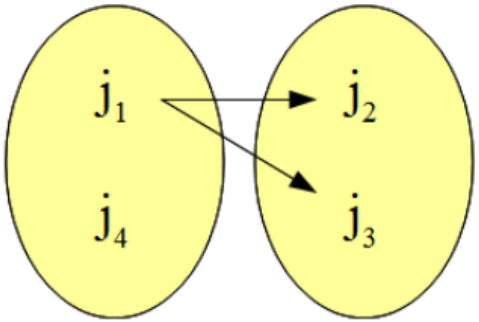
Exemple 2 : optimisation ordinale

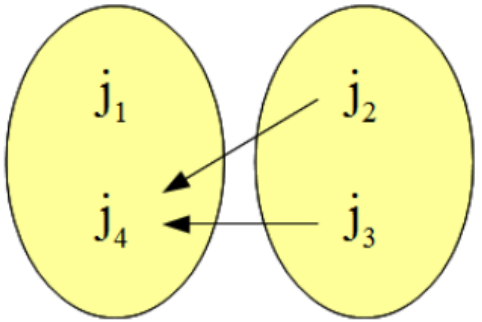
Bossong et Schweigert (1996) :

E $\succcurlyeq$ F si il existe une injection π : F$\rightarrow$E tq :

$$\forall\ f \in F : \pi(f) \succcurlyeq f$$

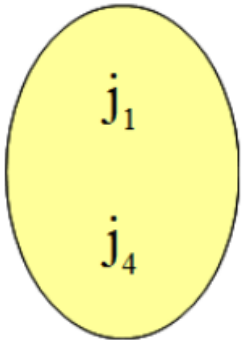






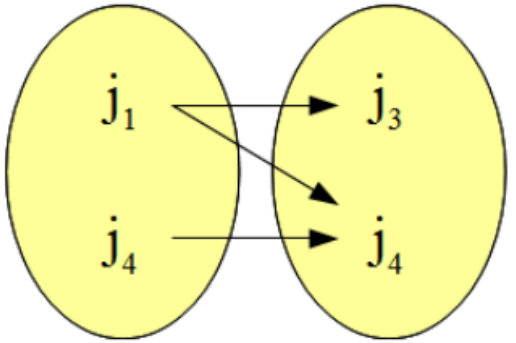
$\{j_1, j_4\} \succcurlyeq \{j_3, j_4\}$ $\quad$ $\{j_1, j_4\} \not\succcurlyeq \{j_2, j_3\}$ $\quad$ $\{j_1, j_4\} \not\preccurlyeq \{j_2, j_3\}$

Exemple 2 : optimisation ordinale



T I C
(1, 1, 2)

$$V(\{j_1, j_4\}) = (1, 1, 2)$$
$$V(\{j_3, j_4\}) = (0, 1, 2)$$
$$V(\{j_1\}) = (1, 1, 1)$$
$$V(\{j_3\}) = (0, 1, 1)$$
$$V(\{j_4\}) = (0, 0, 1)$$



(1,1,2) $\succcurlyeq$ (0,1,2)

Dominance ordinale $\Longleftrightarrow$ Dominance de Pareto

Exemple 3 : optimisation multi-agent

2 reviewers per paper

u_{ij}	*Paper 1*	*Paper 2*	*Paper 3*	*Paper 4*	*Paper 5*
Reviewer 1	3	3	4	3	4
Reviewer 2	3	4	4	2	3
Reviewer 3	1	2	3	2	3

Solution 1 $1 \leftarrow \{1,3,4,5\}$ $2 \leftarrow \{1,2,3\}$ $3 \leftarrow \{2,4,5\}$

$x = (14, 11, 7)$ Σ=32 min = 7

Solution 2 $1 \leftarrow \{1,4,5\}$ $2 \leftarrow \{1,2,3\}$ $3 \leftarrow \{2,3,4,5\}$

$x = (10, 11, 10)$ Σ=31 min = 10

Cadre et notations

Calcul de l'ensemble de Pareto

Approche classique : générer l'ensemble de Pareto

Justification

Naturel si on ne dispose pas d'information préférentielle : un décideur rationnel choisira parmi les solutions Pareto optimales.

Procédures de résolution

- algorithmes gloutons multi-objectifs
- programmation dynamique multi-objectifs
- énumération ordonnée
- méthodes en deux phases
- branch and bound multi-objectifs

Notations

Points and solutions are basically compared using the concept of Pareto dominance.

For any $z, z' \in \mathcal{Z}$, we define the following relations:

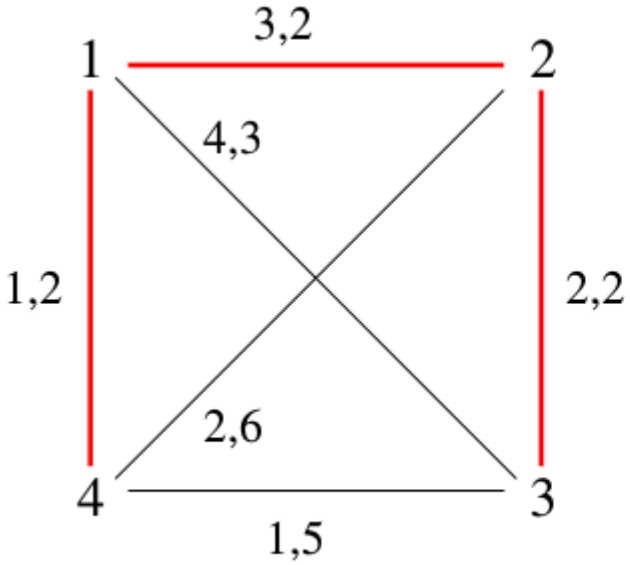
$$
\begin{array}{lll}
z = z' & & \text{iff} \quad z_j = z'_j, \text{ for all } j \in \{1, \ldots, p\} \\
z \leqq z' & (z \textit{ weakly dominates } z') & \text{iff} \quad z_j \leq z'_j, \text{ for all } j \in \{1, \ldots, p\} \\
z < z' & (z \textit{ strictly dominates } z') & \text{iff} \quad z_j < z'_j, \text{ for all } j \in \{1, \ldots, p\} \\
z \leq z' & (z \textit{ dominates } z') & \text{iff} \quad z \leqq z' \text{ and } z' \neq z
\end{array}
$$

We define in the same way the relations $\geqq$, $>$ and $\geq$.

These definitions are propagated in the decision space as follows, for any $x, x' \in \mathcal{X}$:

$$
\begin{array}{lll}
x \approx x' & & \text{iff} \quad c(x) = c(x') \\
x \precapprox x' & (x \textit{ weakly dominates } x') & \text{iff} \quad c(x) \leqq c(x') \\
x \prec x' & (x \textit{ strictly dominates } x') & \text{iff} \quad c(x) < c(x') \\
x \precsim x' & (x \textit{ dominates } x') & \text{iff} \quad c(x) \leq c(x')
\end{array}
$$

Espace de décision, espace des objectifs



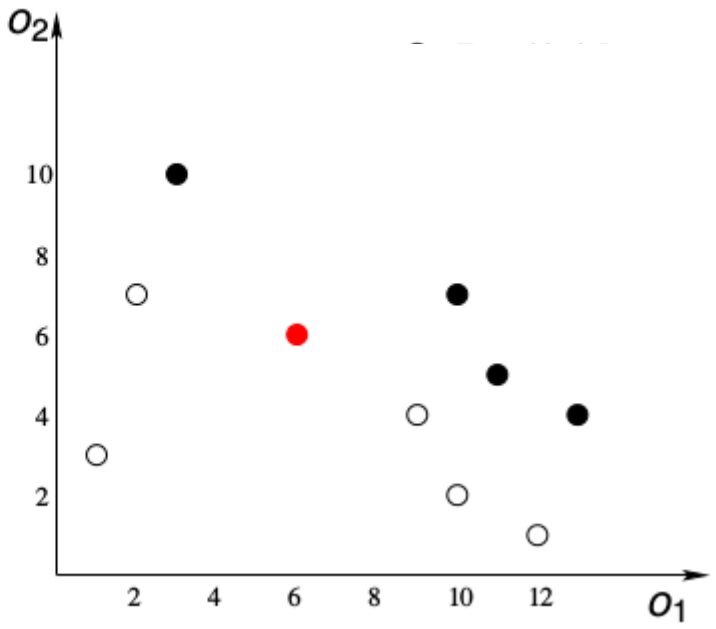
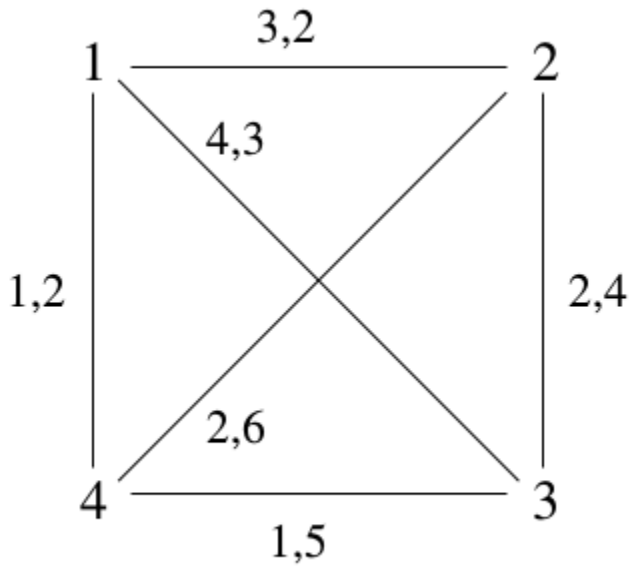


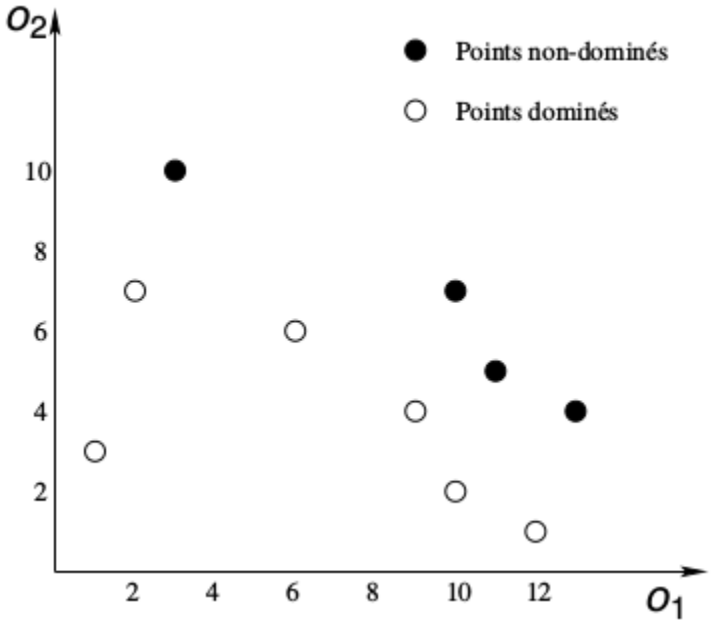
image = **(6,6)**

Espace de décision

Espace des objectifs

Espace de décision, espace des objectifs





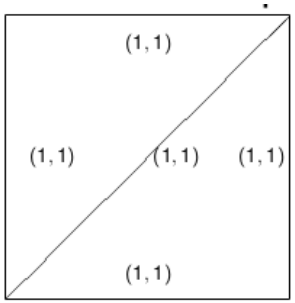
Espace de décision

Espace des objectifs

Espace de décision, espace des objectifs

On recherche **une** solution / point non-dominé

- Si toutes les arêtes ont la même valeur, tous les arbres couvrants sont Pareto optimaux !



T_1	$(3,3)$
T_2	$(3,3)$
T_3	$(3,3)$
T_4	$(3,3)$

T_5	$(3,3)$
T_6	$(3,3)$
T_7	$(3,3)$

- En pratique, les cardinalités dans l'espace de décision et dans l'espace des objectifs diffèrent fortement (les valeurs indiquées correspondent au problème d'affectation multi-objectifs) :

	Espace de décision			Espace des objectifs		
	min	moy	max	min	moy	max
A-20-100	233	374	1487	67	89	129
A-20-200	732	47271	888863	107	136	159

Sélection multi-objectifs

Sélection multi-objectifs (MOSS)

Données : *n* objets, chaque objet *i* est valué par $(c_1^i,\ldots,c_p^i)$

Solution réalisable : sous-ensemble de *k* objets

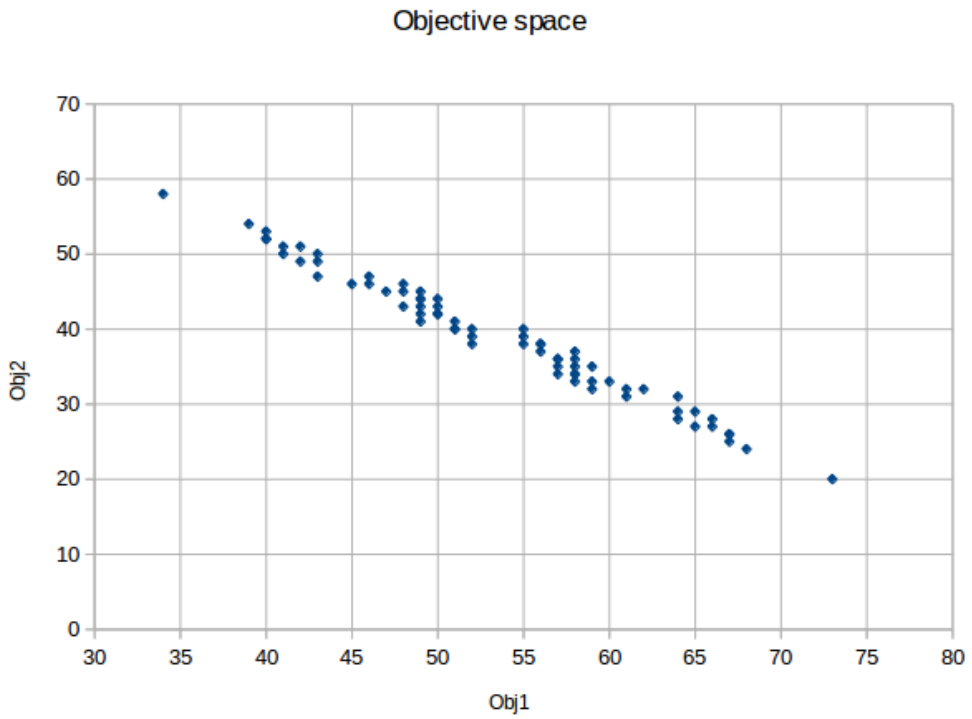
But : déterminer un ensemble de Pareto (min)

Toutes les procédures de résolution présentées dans ce tutoriel seront illustrées sur ce problème.

Illustration

Sélectionner 4 objets parmi 8 (70 solutions réalisables) :

item	Obj1	Obj 2
1	18	5
2	17	7
3	9	14
4	12	11
5	2	20
6	18	6
7	20	2
8	11	13



Complexité

Complexité

Evaluée en étudiant (Serafini 1986, Ehrgott 2000) :

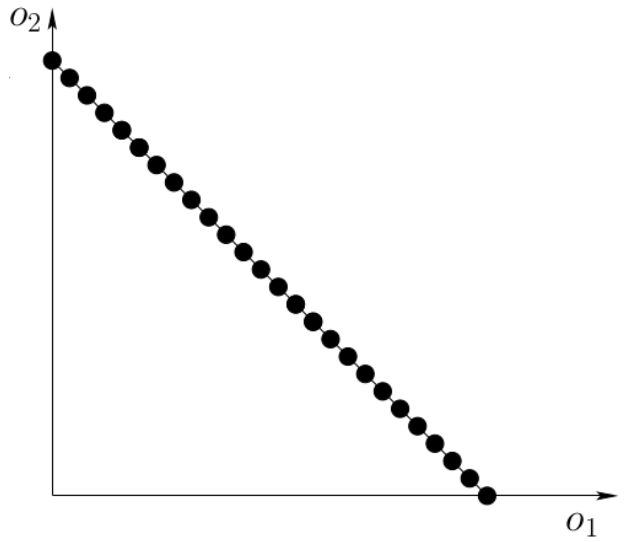
- le nombre de points non-dominés au pire cas. Plusieurs exemples d' instances pathologiques où toutes les solutions réalisables ont des images distinctes dans l'espace des objectifs et sont Pareto-optimales.

- la complexité du problème de décision associé :

"Etant données p valeurs $v_1,...,v_p$ et un problème d'optimisation combinatoire multi-objectifs, existe-t-il une solution réalisable qui domine au sens faible $(v_1,...,v_p)$?"

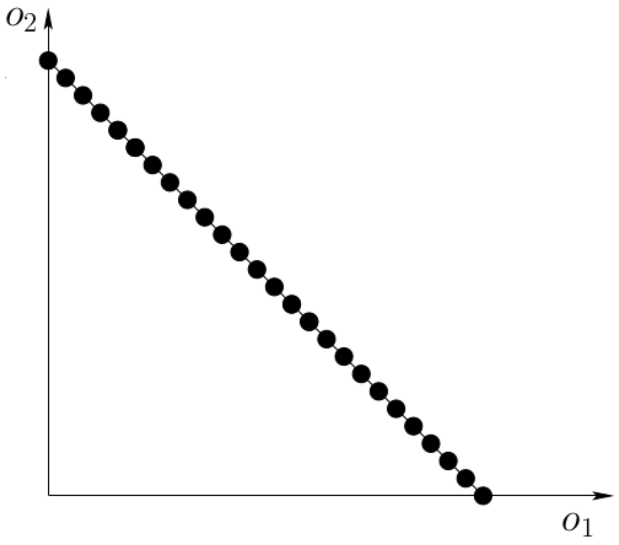
Nombre de points non-dominés

Comment construire une instance avec l'image ci-dessous dans l'espace des objectifs ?



Nombre de points non-dominés

Comment construire une instance avec l'image ci-dessous dans l'espace des objectifs ?



- valuer chaque objet i par $(2^{i-1}, 2^n-2^{i-1})$ pour i=1,...,n
- pour tout ensemble S de cardinalité k on a :
  $f_1(S) + f_2(S) = k2^n$
- la **valeur sur le premier objectif** est :

S	$f_1(S)$	Binaire
{4,3,2,1}	$2^3+2^2+2^1+2^0$	00001111
{8,6,4,2}	$2^7+2^5+2^3+2^1$	10101010
{8,7,6,5}	$2^7+2^6+2^5+2^4$	11110000

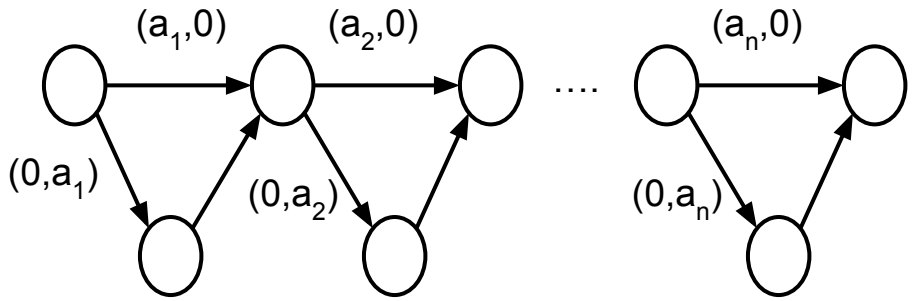
Problème de décision associé

Pour simplifier, considérons le **problème de plus court chemin multi-objectifs**.

(*Partition problem*)
Given $n \in \mathbb{N}$ and a set of numbers $\{a_1, \ldots, a_n\}$, $a_i \in \mathbb{N}$.
Question: **Does there exist a subset $S \subset N := \{1, \ldots, n\}$ such that**

$$\sum_{i \in S} a_i = \sum_{j \in N \setminus S} a_j$$

Answer: **yes or no**



$$\left(\frac{\sum_i a_i}{2}, \frac{\sum_i a_i}{2} \right)$$

Algorithme glouton multi-objectifs

Algorithme glouton multi-objectifs

La version à un seul objectif :

Algorithm 1: Greedy algorithm

Data: $E = \{e^1, \ldots, e^n\}$
Compute a topological numbering τ of the elements of E
$x \longleftarrow \emptyset$
for $i = 1, \ldots, n$ **do**
  if $x \cup \{e^{\tau(i)}\} \in \hat{\mathcal{X}}$ **then** $x \longleftarrow x \cup \{e^{\tau(i)}\}$
return x

où $\tau : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ such that $c(e^i) < c(e^j) \Rightarrow \tau(i) < \tau(j)$

$\hat{\mathcal{X}}$ est l'ensemble des solutions réalisables partielles

Comment étendre au cas multi-objectifs ?

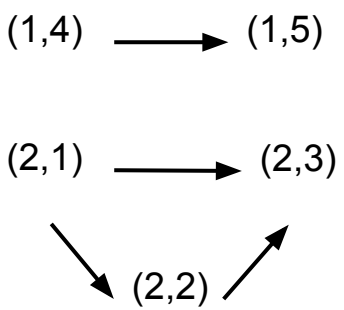
Algorithme glouton multi-objectifs

Algorithme glouton multi-objectifs

1. **Calculer** l'ensemble T de tous les tris topologiques compatibles avec Pareto
2. X ← ∅
3. **Pour** chaque tri topologique τ dans T :
   x ← *Greedy algorithm* selon τ
   X ← X U {x}
4. **Retourner** les solutions Pareto optimales de X

	Obj. 1	Obj. 2
Item 1	1	4
Item 2	1	5
Item 3	2	1
Item 4	2	2
Item 5	2	3

10 solutions réalisables pour k=2



tris	valeur
(1,4), (1,5)	(2,9)
(1,4), (2,1)	(3,5)
(2,1), (2,2)	(4,3)

Algorithme glouton multi-objectifs

Theorem 3. *Let x be a Pareto optimal solution, then there exists a topological numbering on E such that the greedy algorithm returns x.*

Proof. It is sufficient to observe that for all elements e' in $E \setminus x$ and e in x, we cannot have $c(e') \leq c(e)$, otherwise $x \cup \{e'\} \setminus \{e\}$ would Pareto dominate x. Consequently there exists at least one topological numbering on E in which the elements of x are ranked in the k first positions. The greedy algorithm obviously returns x for this topological numbering. □

MAIS :

	Obj. 1	Obj. 2
Item 1	1	4
Item 2	2	2
Item 3	2	2
Item 4	4	1

6 solutions réalisables pour k=2

(1,4) (2,2)

(4,1) (2,2)

(pas d'arcs)

dominée

tris	valeurs
(1,4), (2,2)	(3,6)
(1,4), (4,1)	**(5,5)**
(2,2), (2,2)	(4,4)
(2,2), (4,1)	(6,3)

Algorithme glouton multi-objectifs

L'algorithme glouton multi-objectifs a été principalement appliqué au problème de l'arbre couvrant multi-objectifs :

- Corley (1985) avec une généralisation de l'algorithme de Prim,
- Serafini (1986) avec une généralisation de l'algorithme de Kruskal.

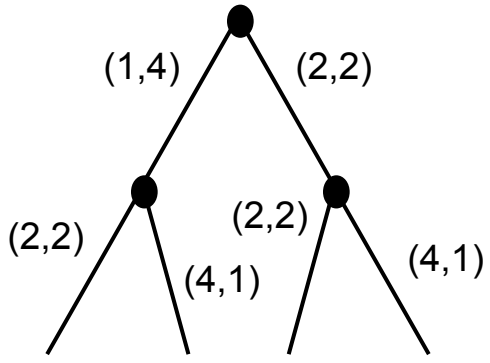
Mais les deux algorithmes ne sont pas capables de résoudre des instances de taille réelle du fait du nombre combinatoire de tris topologiques.

Algorithme glouton multi-objectifs

- L'algorithme glouton multi-objectifs retourne un sur-ensemble de l'ensemble de Pareto
- Il y a eu des tentatives pour échapper à cet écueil, en appliquant un filtre de Pareto à chaque niveau de l'arbre d'énumération des tris topologiques :

Filtre de Pareto

Filtre de Pareto



tris	valeur
(1,4), (2,2)	(3,6)
(1,4), (4,1)	**(5,5)**
(2,2), (2,2)	(4,4)
(2,2), (4,1)	(6,3)

Algorithme glouton multi-objectifs

ELSEVIER

European Journal of Operational Research 143 (2002) 543–547

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

Enumeration of Pareto optimal multi-criteria spanning trees – a proof of the incorrectness of Zhou and Gen's proposed algorithm

Joshua D. Knowles *, David W. Corne

Department of Computer Science, University of Reading, Reading, UK

Received 16 January 2001; accepted 25 September 2001

Abstract

The minimum spanning tree (MST) problem is a well-known optimization problem of major significance in operational research. In the *multi-criteria* MST (mc-MST) problem, the scalar edge weights of the MST problem are replaced by vectors, and the aim is to find the complete set of Pareto optimal minimum-weight spanning trees. This

Et ce n'est même pas Zhou and Gen qui l'ont proposé initialement !

Algorithme glouton multi-objectifs

- Hamacher et Ruhe (1994) sont les premiers à avoir proposé cette modification de l'algorithme :

  "In our experiments, we used a modified version [...] which excludes in each iteration subtrees which are non-efficient".

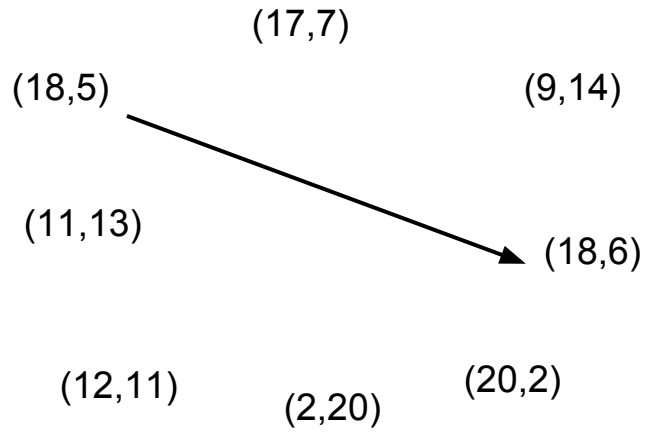
- Cette version a ensuite été reprise par Zhou and Gen (1999) et par Ehrgott (2000).

- Le remède est ici pire que le mal : la version modifiée peut manquer des solutions Pareto-optimales et par conséquent également retourner des solutions dominées qui ne peuvent pas être filtrées.

Algorithme glouton multi-objectifs

Retour sur l'instance initiale:

item	Obj1	Obj 2
1	18	5
2	17	7
3	9	14
4	12	11
5	2	20
6	18	6
7	20	2
8	11	13



De très nombreux tris topologiques

⇨ L'algorithme glouton multi-objectifs va être inefficace.

Programmation dynamique multi-objectifs

Programmation dynamique multi-objectifs

- Programmation dynamique : un problème est résolu en identifiant une collection de sous-problèmes et en les traitant un par un, les plus petits en premier, les solutions des petits problème étant utilisées pour aider à la résolution des plus grands, jusqu'à ce que les problèmes soient tous résolus.

- La validité de l'approche repose sur le principe d'optimalité de Bellman.

- Ce principe tient toujours si on recherche l'ensemble de Pareto.

Programmation dynamique multi-objectifs

Taille	{1}	…	{1,...,j-1}	{1,...,j}	…	{1,...,n}
0 ↓	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
1						
...						
i-1			F			
i			G	E		
...						
k						

PARETO

E : image des sous-ens Pareto optimaux de taille i dans {1,...,j}

F : image des sous-ens Pareto optimaux de taille i-1 dans {1,...,j-1}

G : image des sous-ens Pareto optimaux de taille i dans {1,...,j-1}

Relation de récurrence :
E = OPT(F+f(j) U G)

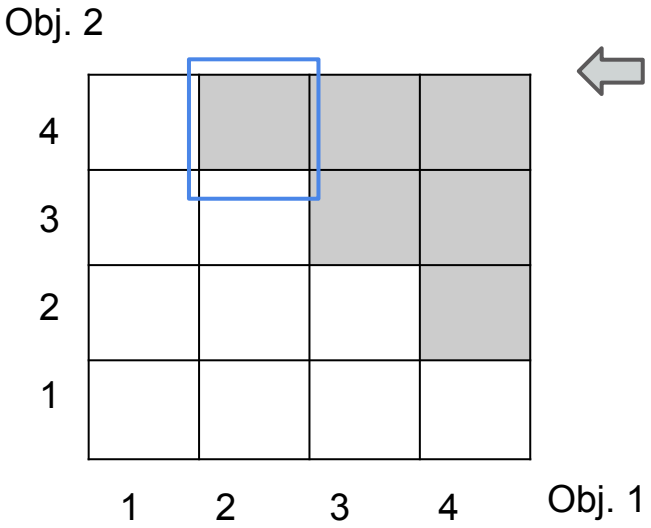
Programmation dynamique multi-objectifs

Size ↓	{1} (1,4)	{1,2} (2,3)	{1,2,3} (5,2)	{1,...,4} (2,2)	{1,...,5} (3,1)	{1,...,6} (2,5)	{1,...,7} (3,4)
0	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
1	(1,4)	(1,4) (2,3)	(1,4) (2,3) (5,2)	(1,4) (5,2) (2,2)	(1,4) (2,2) (3,1)	(1,4) (2,2) (3,1)	(1,4) (2,2) (3,1)
2	0	(3,7)	(3,7) (6,6) (7,5)	(3,6) (4,5) (7,4)	(3,6) (4,5) (5,3)	(3,6) (4,5) (5,3)	(3,6) (4,5) (5,3)
3	0	0	(8,9)	(5,9) (8,9) (9,7)	(5,9) (6,7) (7,6) (10,5)	(5,9) (6,7) (7,6) (10,5)	(5,9) (6,7) (7,6) (10,5)

PARETO

Programmation dynamique multi-objectifs

- Complexité dans le cas bi-objectifs ?
- k lignes et n colonnes → kn cellules dans le tableau
- O(M) vecteurs / cellule où M est une borne supérieure sur la valeur d'un objectif



⇦ au plus **un** point non-dominé par ligne !

O(knM)
(pseudo-polynomial)

- Appliquer la relation de récurrence prend un temps O(M) en utilisant un arbre rouge-noir

Applications

De multiples applications de la programmation dynamique (DP) multi-objectifs :

- sac-à-dos multi-objectifs,
- plus court chemin multi-objectifs,
- etc.

(en fait tous les problèmes qui peuvent être résolus par DP dans le cas à un seul objectif)

Enumération ordonnée (cas bi-objectifs)

Le cas bi-objectifs

Définition : problème combinatoire bi-objectifs

- E : ens fini d'éléments,
- $c_1(.)$ et $c_2(.)$: deux fonctions objectifs $E \rightarrow \mathbb{Z}^+$,
- $\mathcal{F} \subseteq 2^E$: ens de solutions réalisables,
- $\forall S \in \mathcal{F}$, $c_i(S) = \sum_{e \in S} c_i(e)$.

But : ens $\mathcal{F}^*$ des solutions Pareto-optimales

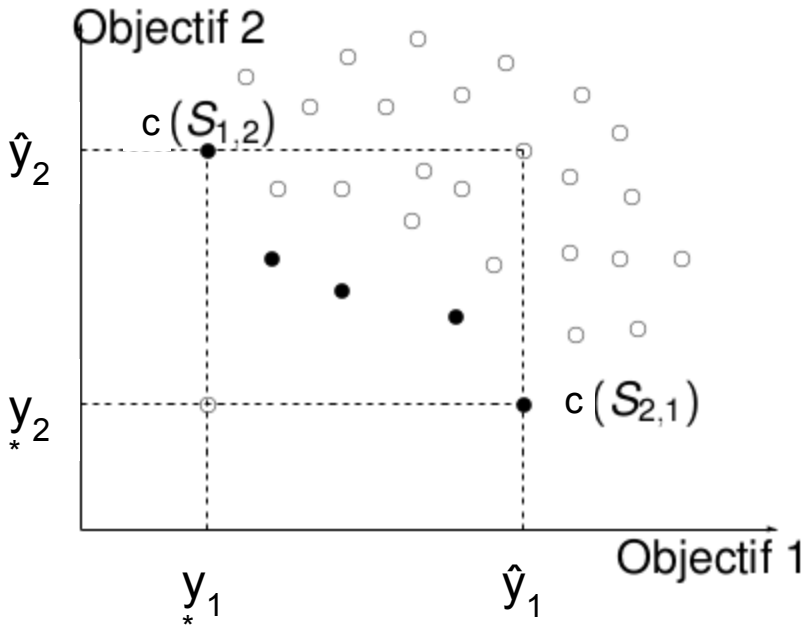
Représentation graphique commode de l'espace des objectifs :

Point idéal et point nadir

Définition : point idéal et point nadir

Le point idéal est le vecteur y^* défini par : $y_i^* = \min_{S \in \mathcal{F}} c_i(S)$

Le point nadir est le vecteur $\hat{y}$ défini par : $\hat{y}_i = \max_{S \in \mathcal{F}^*} c_i(S)$



Algorithme Climaco et Martins (82)

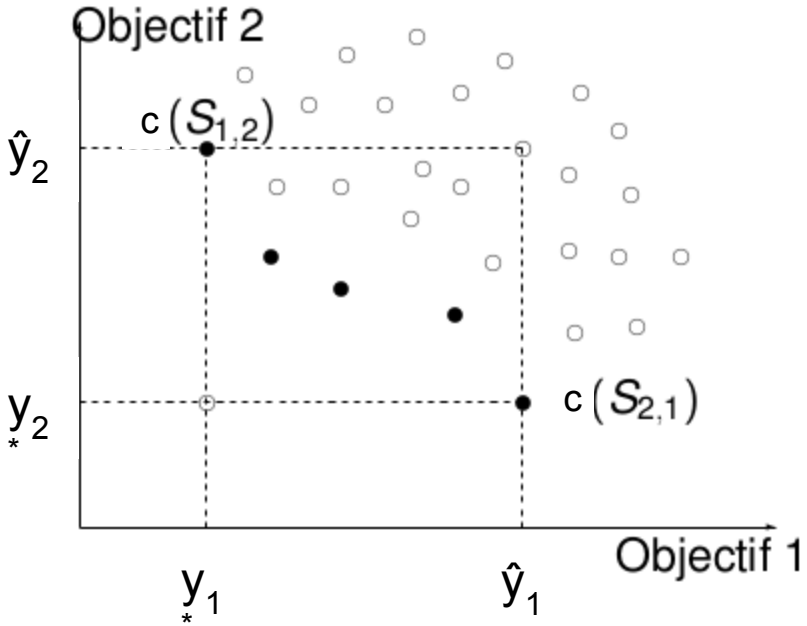
Proposition

Une solution non-dominée S vérifie :

$$y_1^* \leq c_1(S) \leq \hat{y}_1 \text{ et } y_2^* \leq c_2(S) \leq \hat{y}_2$$

Idée de la preuve : Une solution S telle que $c_1(S) > \hat{y}_1$ est dominée par $S_{2,1}$.

Principe de l'algorithme : Commencer avec la solution leximin $S_{1,2}$ puis construire la seconde selon c_1, c_2… jusqu'à atteindre $S_{2,1}$.



Un exemple : MOSS

INSTANCE

objet	objectif 1	objectif 2
1	1	3
2	2	2
3	2	3
4	3	1
5	3	2
6	4	4

Choisir 3 objets parmi 6

20 solutions réalisables

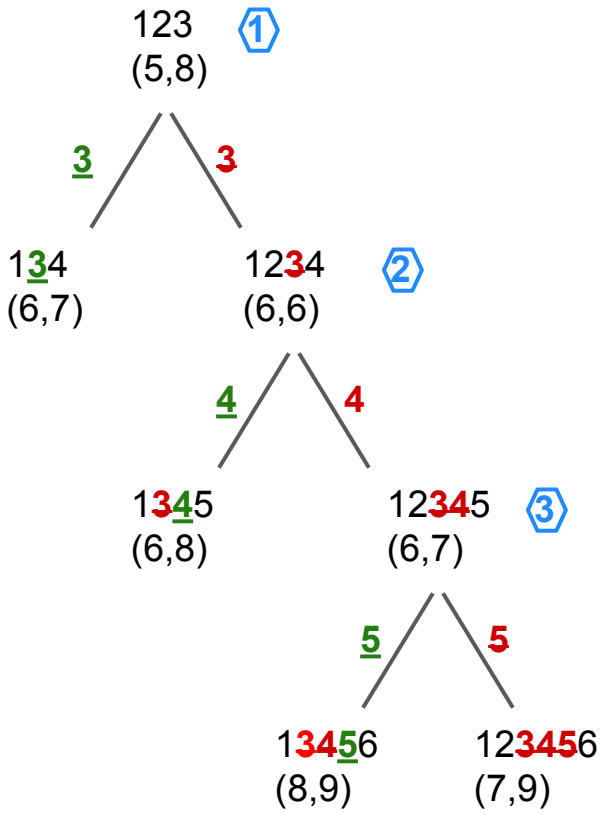
$\hat{y} = (8,8)$

Enumération ordonnée

123 → (5,8) ①
124 → (6,6) ②
125 → (6,7) ③
134 → (6,7)
135 → (6,8)
145 → (7,6)
234 → (7,6)
235 → (7,7)
126 → (7,9)
136 → (7,10)
245 → (8,5)
345 → (8,6)
146 → (8,8)

---

156 → (8,9)
236 → (8,9)
246 → (9,7)
256 → (9,8)
346 → (9,8)
356 → (9,9)
456 → (10,7)



Arbre d'énumération ordonnée

Enumération ordonnée des K meilleures solutions (K meilleurs)

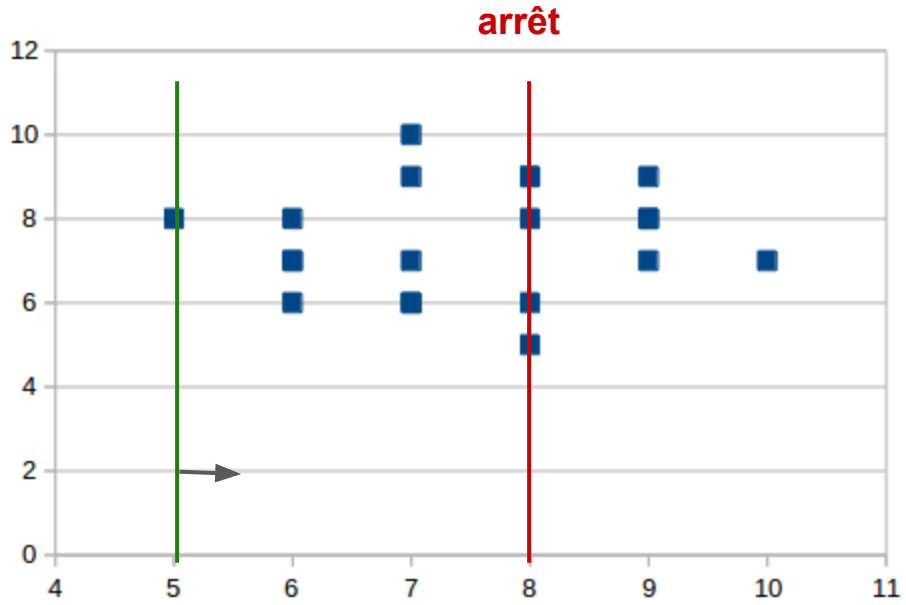
Algorithme inspiré de Gabow (77), Hamacher & Queyranne (85)

1. S←solution de coût minimal
2. Sol←{S}
3. IN←∅ ; OUT←∅
4. empiler(S,IN,OUT)
5. k←1
6. **tant que** k<K **faire**
   - 6.1. (S,IN,OUT)←dépiler()
   - 6.2. (i,j)←echange(S,IN,OUT) // i∈S, j∉S
   - 6.3. S_k←S∪{j}\{i}
   - 6.4. Sol←Sol∪{S_k}
   - 6.5. empiler(S,IN∪{i},OUT)
   - 6.6. empiler(S_k,IN,OUT∪{i})
   - 6.7. k←k+1

Sortie : Sol, l'ensemble des K meilleures solutions

- IN: élément obligatoires
- OUT: éléments interdits
- echange(S,IN,OUT): détermine le meilleur échange à réaliser

Limite de l'approche de Climaco et Martins



Inconvénient de l'approche de Climaco et Martins : la condition d'arrêt se déclenche très tardivement.

Une approche plus efficace

Idée : parcourir les solutions dans la direction orthogonale au segment liant les deux points leximin (5,8) et (8,5).

INSTANCE (RETRIEE)

objet	objectif 1	objectif 2
1	1	3
2	2	2
4	3	1
3	2	3
5	3	2
6	4	4

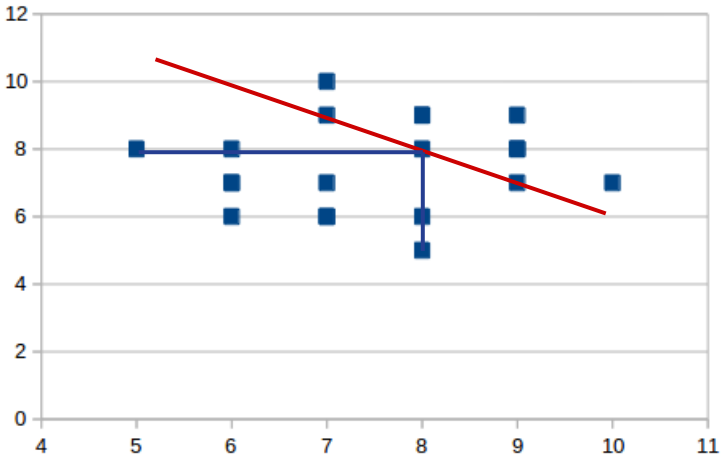
Choisir 3 objets parmi 6

20 solutions réalisables

Enumération ordonnée

124 → (6,6), 12
123 → (5,8), 13
125 → (6,7), 13
143 → (6,7), 13
145 → (7,6), 13
243 → (7,6), 13
245 → (8,5), 13
135 → (6,8), 14
235 → (7,7), 14
435 → (8,6), 14
126 → (7,9), 16
146 → (8,8), 16
246 → (9,7), 16

136 → (7,10), 17
156 → (8,9), 17
etc.



Une approche plus efficace

Idée : parcourir les solutions dans la direction orthogonale au segment liant les deux points leximin (5,8) et (8,5).

INSTANCE (RETRIEE)

objet	objectif 1	objectif 2
1	1	3
2	2	2
4	3	1
3	2	3
5	3	2
6	4	4

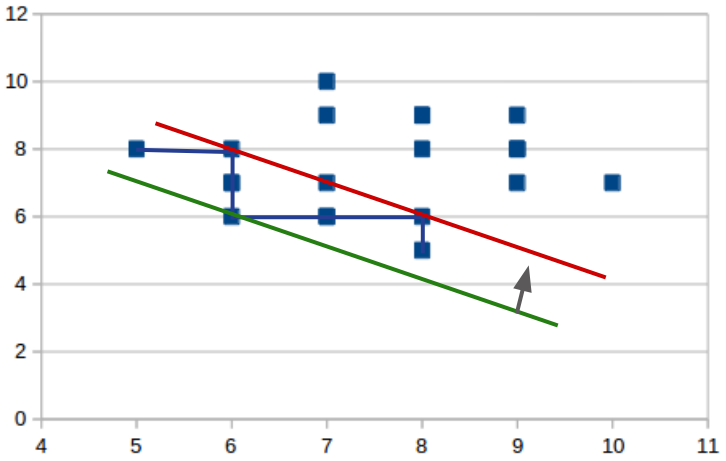
Choisir 3 objets parmi 6

20 solutions réalisables

Enumération ordonnée

124 → (6,6), 12
123 → (5,8), 13
125 → (6,7), 13
134 → (6,7), 13
145 → (7,6), 13
234 → (7,6), 13
245 → (8,5), 13
135 → (6,8), 14
235 → (7,7), 14
345 → (8,6), 14

126 → (7,9), 16
146 → (8,8), 16
246 → (9,7), 16
136 → (7,10), 17
156 → (8,9), 17
etc.



Une approche plus efficace

Idée : parcourir les solutions dans la direction orthogonale au segment liant les deux points leximin (5,8) et (8,5).

INSTANCE (RETRIEE)

objet	objectif 1	objectif 2
1	1	3
2	2	2
4	3	1
3	2	3
5	3	2
6	4	4

Choisir 3 objets parmi 6

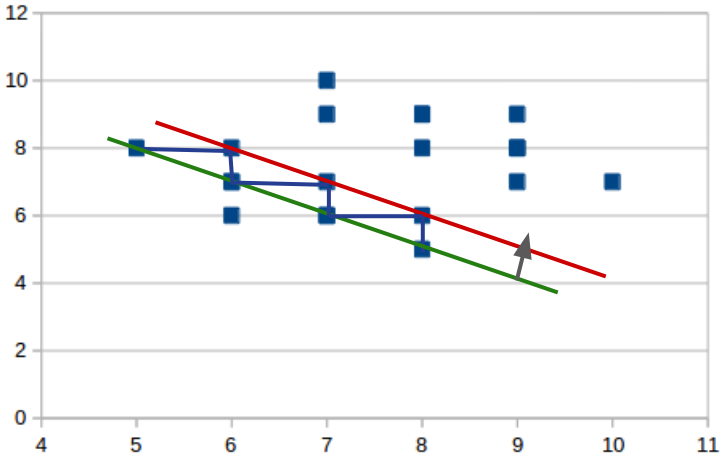
20 solutions réalisables

Enumération ordonnée

124 → (6,6), 12
123 → (5,8), 13
125 → (6,7), 13
134 → (6,7), 13
145 → (7,6), 13
234 → (7,6), 13
245 → (8,5), 13
135 → (6,8), 14
235 → (7,7), 14
345 → (8,6), 14

---

126 → (7,9), 16
146 → (8,8), 16
246 → (9,7), 16
136 → (7,10), 17
156 → (8,9), 17
etc.



Une approche plus efficace

Idée : parcourir les solutions dans la direction orthogonale au segment liant les deux points leximin (5,8) et (8,5).

INSTANCE (RETRIEE)

objet	objectif 1	objectif 2
1	1	3
2	2	2
4	3	1
3	2	3
5	3	2
6	4	4

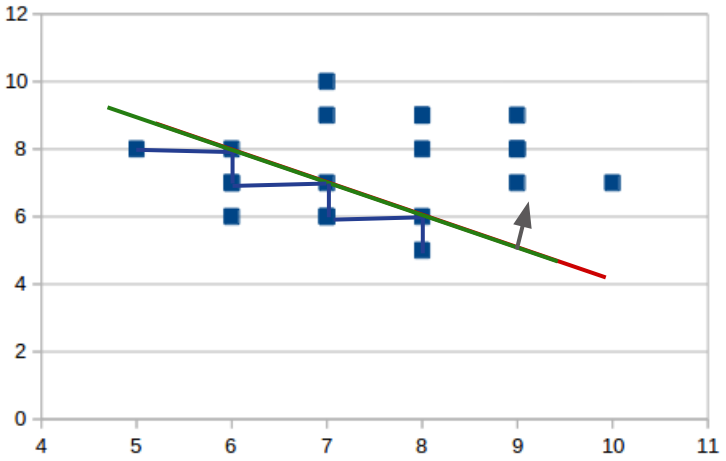
Choisir 3 objets parmi 6

20 solutions réalisables

Enumération ordonnée

124 → (6,6), 12
123 → (5,8), 13
125 → (6,7), 13
134 → (6,7), 13
145 → (7,6), 13
234 → (7,6), 13
245 → (8,5), 13
135 → (6,8), 14
235 → (7,7), 14
345 → (8,6), 14

126 → (7,9), 16
146 → (8,8), 16
246 → (9,7), 16
136 → (7,10), 17
156 → (8,9), 17
etc.



Applications

Ces méthodes peuvent s'appliquer à tout problème d'optimisation combinatoire pour lequel il existe un algorithme efficace d'énumération ordonnée des solutions :

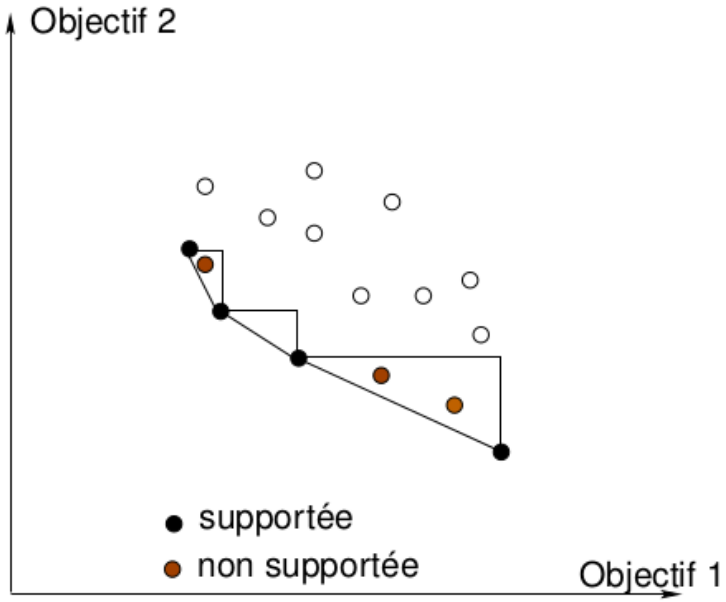
- **Plus court chemin** [Yen 1972, Eppstein 1998, Jimenez & Marzal 2003]
- **Arbre couvrant minimal** [Gabow 1977, Katoh et al. 1981, Eppstein 1992]
- **Affectation** [Katoh et al. 1981, Pedersen et al. 2008]
- **Couplage** [Hamacher & Queyranne 1985, Chegireddy & Hamacher 1987]

L'intérêt est bien sûr qu'une fois qu'on dispose de l'algorithme d'énumération ordonnée des solutions, l'adapter pour trouver les solutions Pareto-optimales d'un problème bi-objectifs ne pose pas de difficulté.

Méthodes en deux phases (cas bi-objectifs)

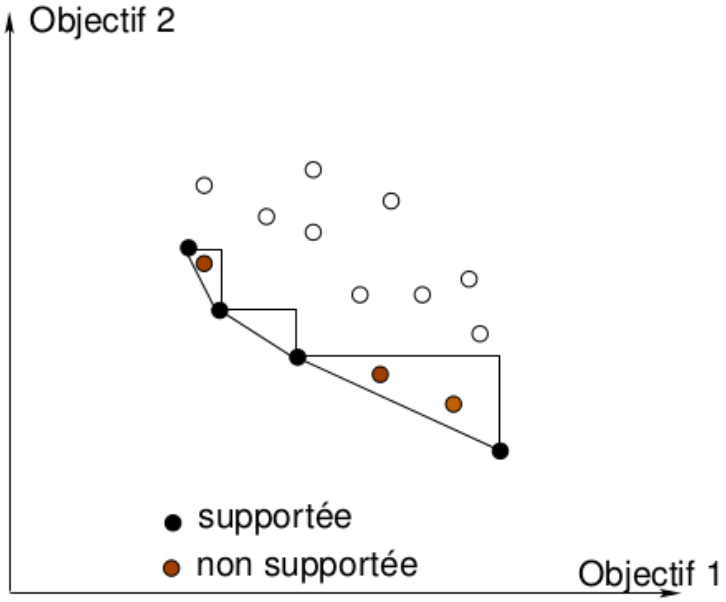
Méthode en deux phases

- Certaines solutions Pareto-optimales sont plus faciles à calculer que d'autres : ce sont les solutions supportées, qui optimisent une combinaison linéaire des objectifs.
- Les autres solutions Pareto-optimales sont dites non-supportées.
- Les solutions supportées situées aux sommets de l'enveloppe convexe sont dites extrêmes, les autres sont dites non-extrêmes.



Méthode en deux phases

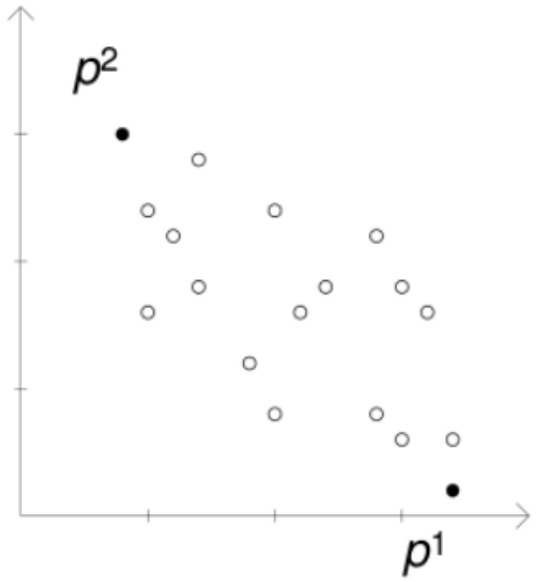
- méthode générale pour résoudre des problèmes bi-objectifs introduite par Ulungu et Teghem (1993).
- **première phase :** trouver toutes les solutions **extrêmes.**
- **deuxième phase :** trouver toutes les solutions **non-extrêmes**.
- assez efficace lorsque le problème mono-objectif sous-jacent est “facile” à résoudre.



Méthode en deux phases : phase 1

Méthode d'Aneja et Nair :

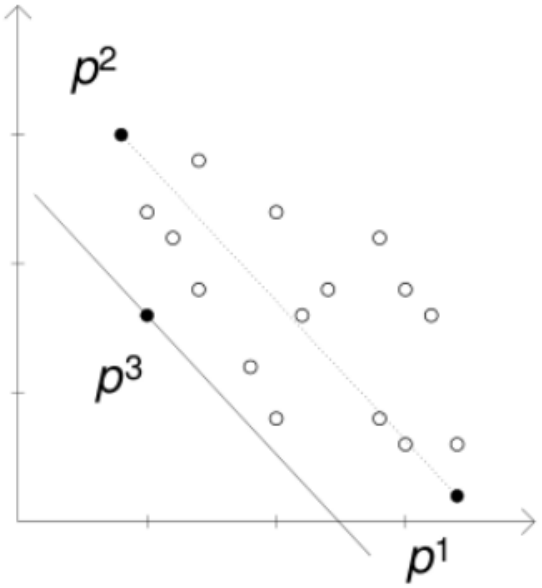
- on identifie les deux points leximin
- le calcul de telles solutions revient à de l'optimisation à un seul objectif
- les deux points leximin sont p^1 et p^2



Méthode en deux phases : phase 1

Méthode d'Aneja et Nair :

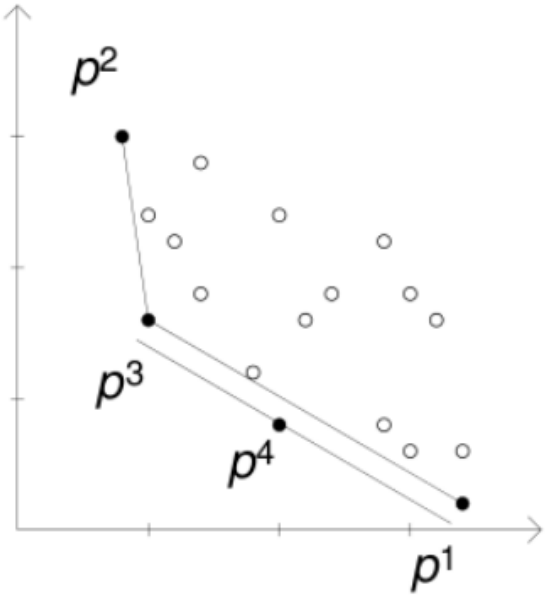
- on calcule récursivement les points extrêmes
- combinaison linéaire des objectifs selon la droite qui passe par les points p^1 et p^2
- on trouve le point p^3



Méthode en deux phases : phase 1

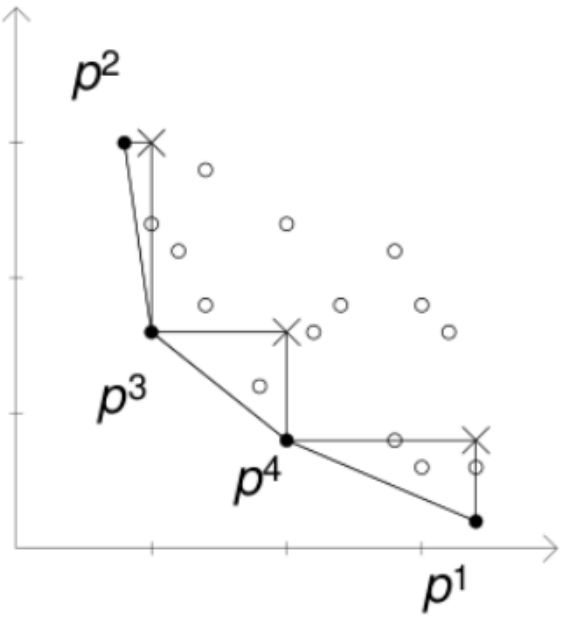
Méthode d'Aneja et Nair :

- on calcule récursivement les points extrêmes
- on teste si il existe un point supporté entre p^1 et p^3, et entre p^3 et p^2
- on trouve le point p^4



Méthode en deux phases : phase 1

A l'issue de la phase 1, les points non-dominés restants se situent nécessairement **dans les triangles** entre deux solutions consécutives.



Pseudo-code de la phase 1

Algorithme d'Aneja et Nair

pour i=1,2 **faire**

ranger lexicographiquement / objectif i

résoudre le problème pour cet ordre - sol : $s_i = (x_i, y_i)$

si $s_1 = s_2$ **alors retourner** $\{s_1\}$

sinon

$L := RechFront(s_1, s_2)$

retourner concaténation de $\{s_1\}$, L, $\{s_2\}$

Procédure RechFront(s', s'')

calculer les nouveaux coûts $\varphi_1(e)(y' - y'') + \varphi_2(e)(x'' - x')$

résoudre le problème pour ces coûts - sol : $s = (x, y)$

si $s = s'$ ou $s = s''$ **alors retourner** la liste vide

sinon

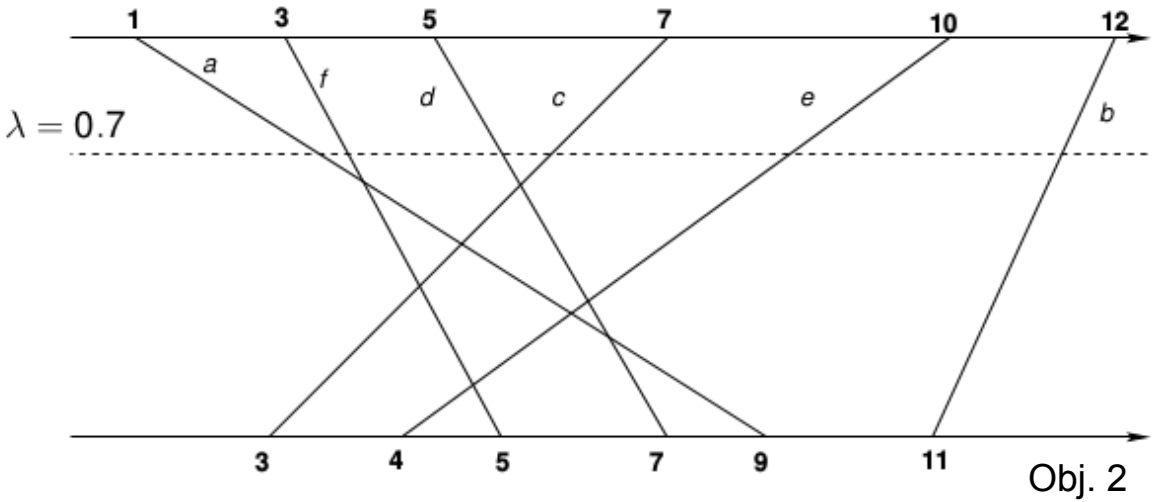
$L' := RechFront(s', s)$; $L'' := RechFront(s, s'')$

retourner concaténation de L', $\{s\}$, L''

Complexité de la phase 1 pour le problème MOSS

A chaque objet est associé un segment reliant la valeur sur Obj. 1 et la valeur sur Obj. 2.
Le rangement pour λ = 0.7 is (a,f,d,c,e,h,g,b).
Le rangement pour λ = 0.1 is (c,e,f,d,a,h,b,g).

	Obj. 1	Obj. 2
Item a	1	9
Item b	12	11
Item c	7	3
Item d	5	7
Item e	10	4
Item f	3	5

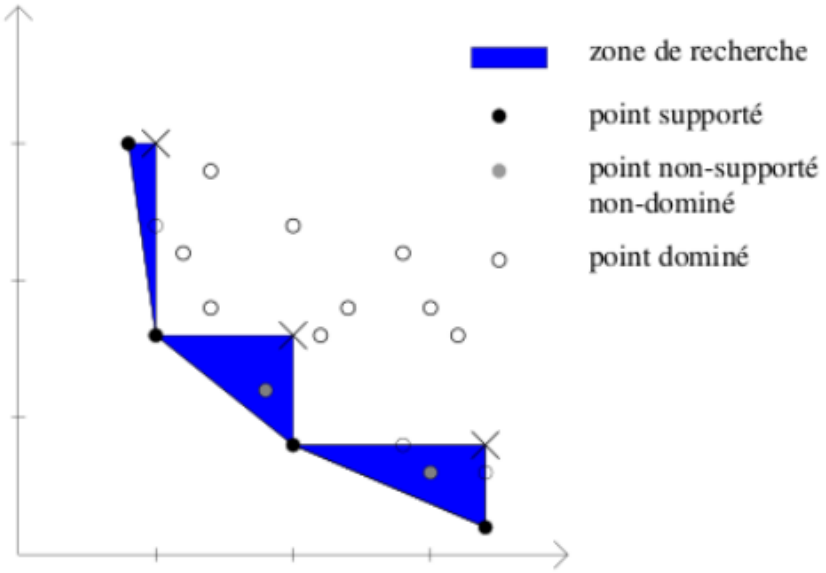


Géométrie algorithmique. Le nombre de points d'intersection est polynomial du nombre de segments : $O(n^2)$ où n est le nombre d'objets.

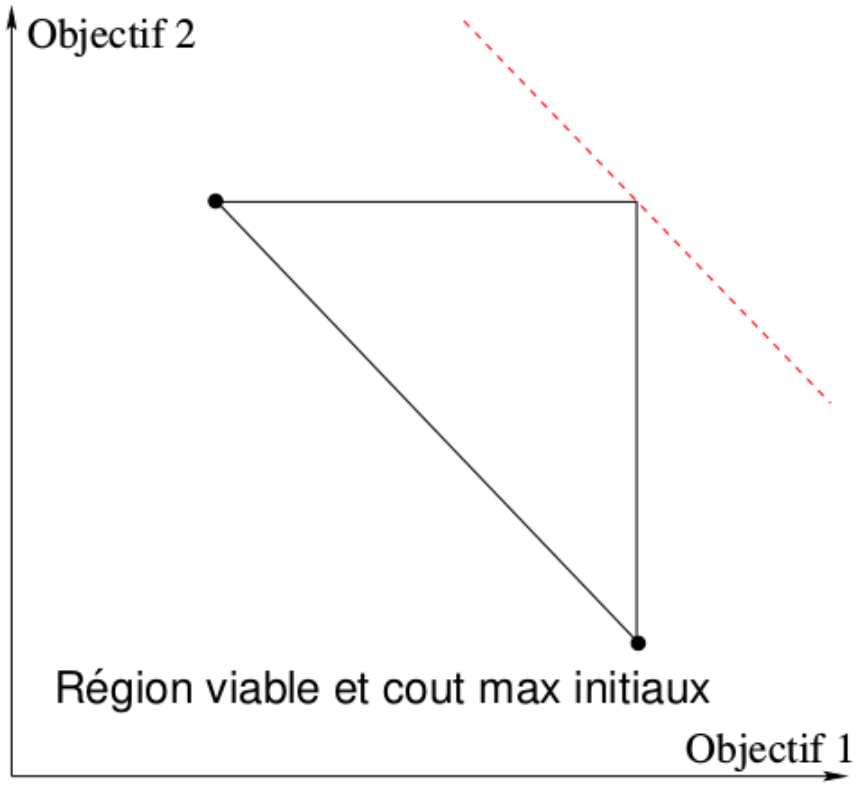
⇒ **nombre polynomial d'appel récursif dans la méthode d'Aneja et Nair.**

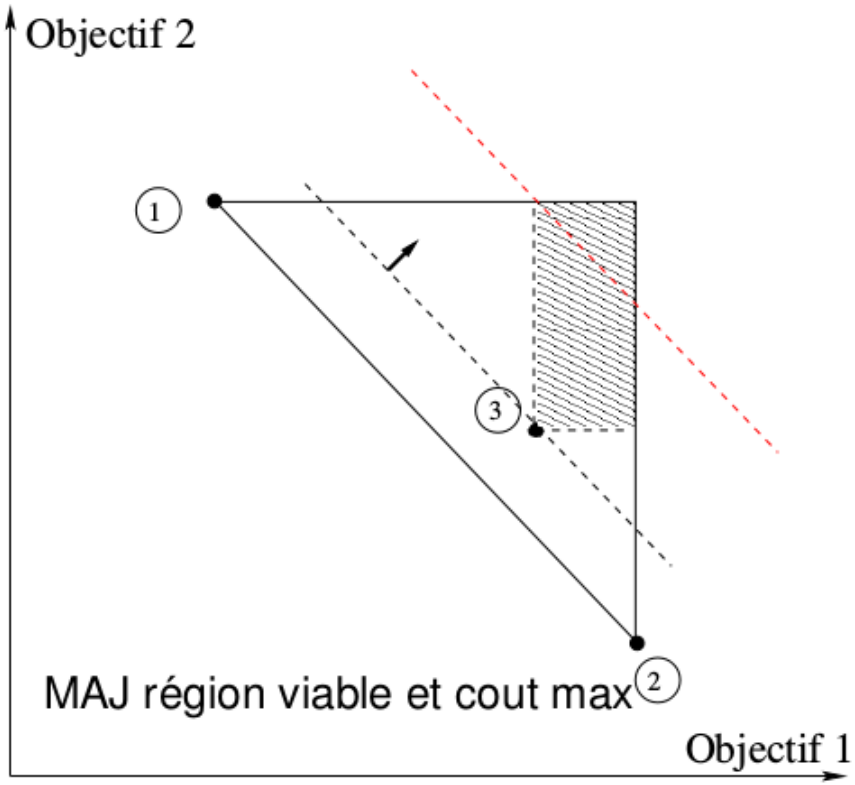
Méthode en deux phases : phase 2

- Explorer les triangles un à un ;
- Version exacte : branch and bound, énumération ordonnée…
- Version approchée : recherche locale

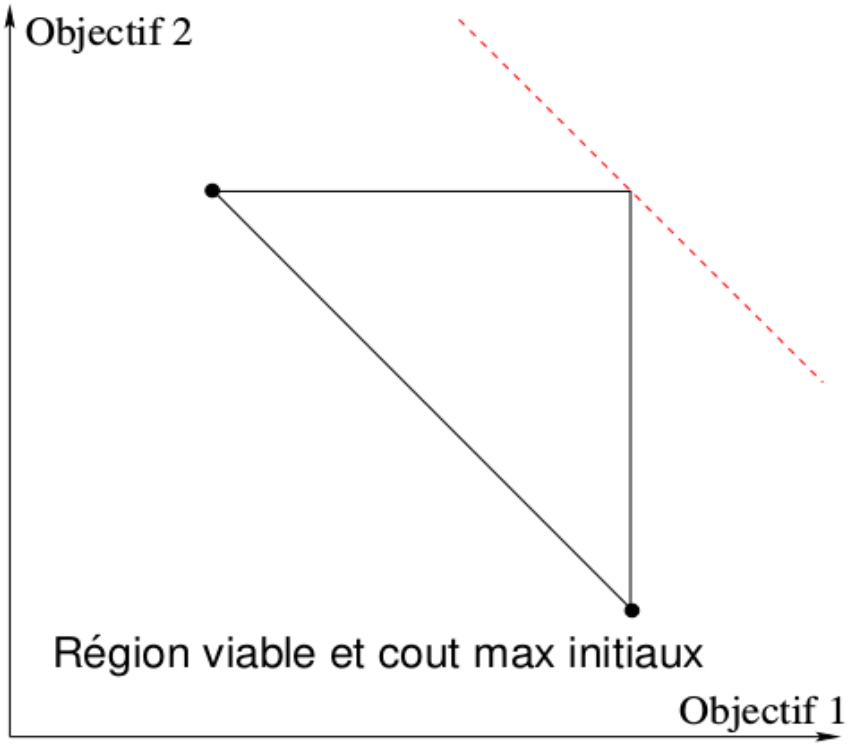


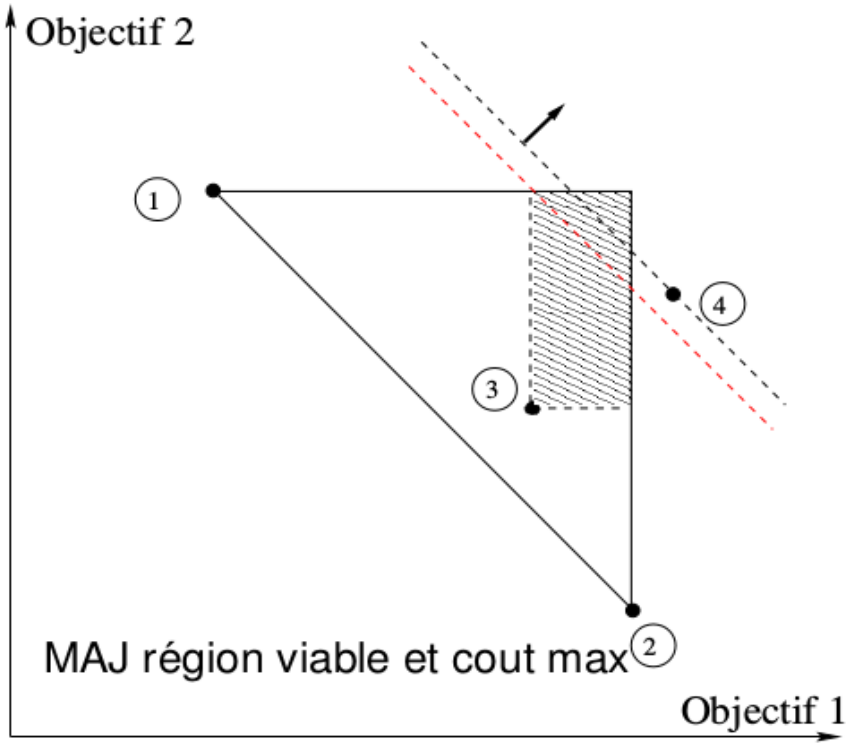
Phase 2 - énumération ordonnée





Phase 2 - énumération ordonnée





Pseudo-code phase 2 - K meilleurs

Algorithme Phase 2

pour tout couple de solutions extrêmes consécutives $s' = (x', y')$, $s'' = (x'', y'')$ **faire**

définir la région viable et le coût maximum

pour $k = 1, \ldots, \infty$ **faire**

déterminer la k^e meilleure solution $s_k = (x_k, y_k)$

si il n'y a plus de solution **alors** sortir

sinon

si s_k est dans la région viable **alors**

ajouter s_k à liste des sol. efficaces non-extrêmes

mettre à jour la région viable et le coût maximum

sinon si s_k atteint ou dépasse le coût max **alors** sortir

Applications

La méthode en deux phases a été appliquée à la version bi-objectifs de divers problèmes d'optimisation classiques, parmi lesquels :

- **Affectation** [Przybylski et al. 2008, Delort et Spanjaard 2011]
- **Sac à dos** [Visée et al. 1998, Delort et Spanjaard 2010]
- **Flot à coût minimum** [Raith et Ehrgott, 2009]
- **Arbre couvrant minimal** [Ramos et al. 1998, Steiner et Radzik 2008]

La méthode a été adaptée pour trois objectifs par Przybylski et al. (2010) et appliquée au problèmes tri-objectifs d'affectation [Przybylski et al. 2010] et de sac à dos [Jorge 2010].

Exemple : arbre couvrant bi-objectifs

Trois variantes selon la procédure d'exploration des triangles :

- *une variante approchée* [Hamacher et Ruhe 1994] :
  recherche par voisinage pour explorer les triangles
- *deux variantes exactes*
  - Ramos et al. 1998 : branch and bound pour explorer les triangles
  - Steiner et Radzik 2006 : énumération ordonnée pour explorer les triangles

Pour les variantes exactes, les meilleurs résultats (en rapidité de calculs) sont obtenus avec une énumération ordonnée en phase 2 (résolution sur des cliques comportant jusqu'à 38 sommets en moins d'une minute).

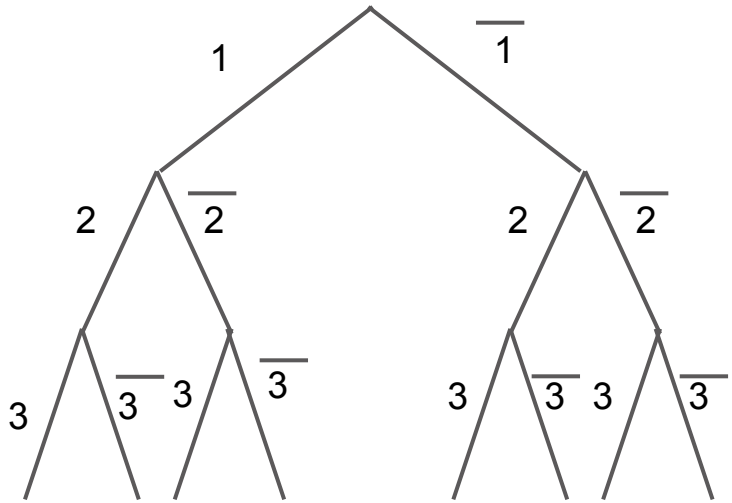
Néanmoins, la **recherche locale** en phase 2 permet d'obtenir une bonne approximation sur des instances de **tailles beaucoup plus importantes** (jusqu' à 400 sommets).

Inconvénient : cette variante est approchée. **Solution :** Prouver l'optimalité de la frontière (ou la compléter) avec un branch and bound multi-objectifs

Branch and bound multi-objectifs

Branch and bound multi-objectifs

- Fondé sur l'exploration d'un arbre d'énumération :



- Chaque noeud de l'arbre correspond à un sous-ensemble de solutions réalisables

Branch and bound multi-objectifs

- L'idée principale est d'utiliser une **procédure de calcul de borne** pour explorer une portion la plus petite possible de l' arbre d'énumération :

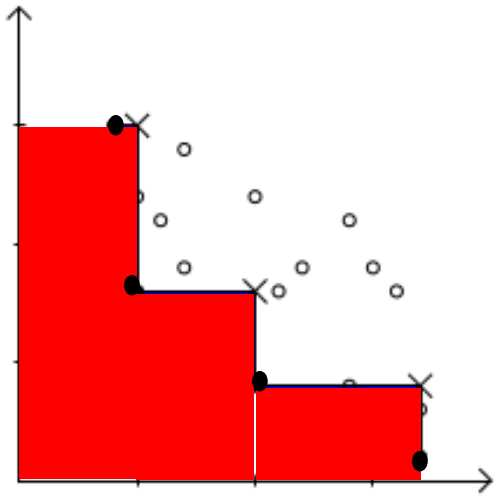
  si on peut prouver qu'aucune solution dans le sous-ensemble considéré ne pourra faire mieux que la meilleure solution trouvée jusqu'alors, alors il est inutile d'explorer le sous-arbre correspondant

- Comment étendre ce principe au cas multi-objectifs ?

- D'abord on doit calculer un ensemble de Pareto approché de bonne qualité, par exemple avec une recherche locale

Branch and bound multi-objectifs

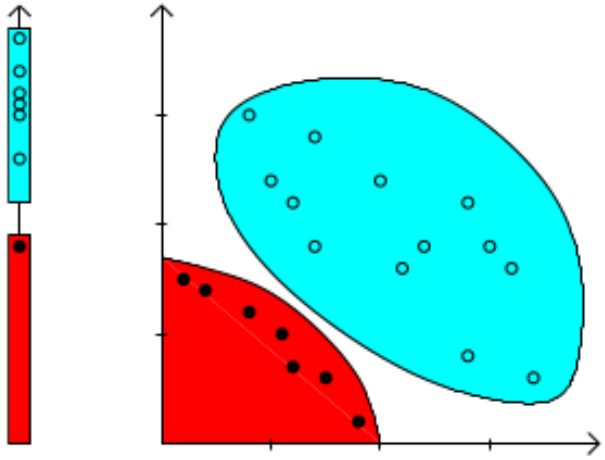
A l'issue de la procédure, les points non-dominés appartiennent nécessairement à la zone rouge.



Branch and bound multi-objectifs

Comment tester si un sous-ensemble ne peut inclure de nouveaux points non-dominés ? → utiliser des ensembles bornant

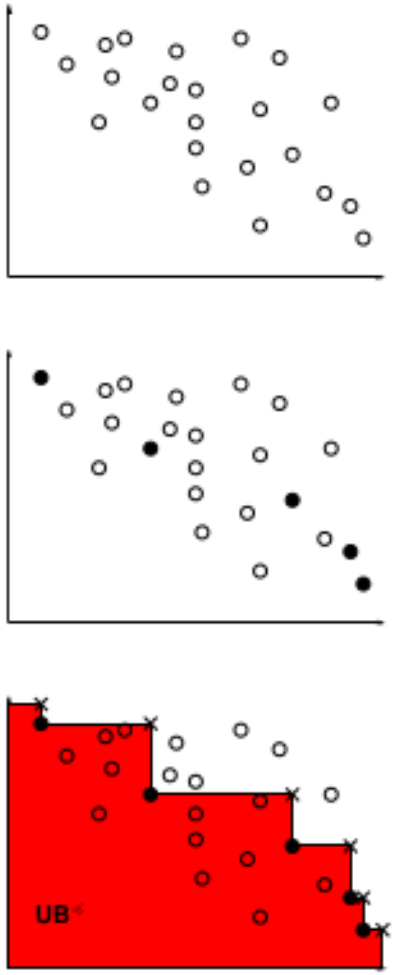
- proposés par Villareal et Karwan (1981).
- Mis en oeuvre indépendamment par Ehrgott et Gandibleux (2007) et Sourd et Spanjaard (2008)



Branch and bound multi-objectifs

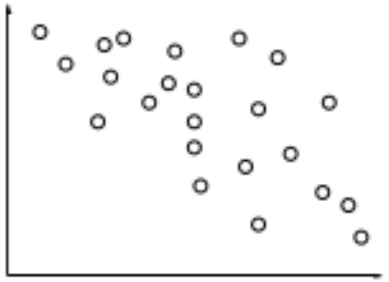
- Un **ensemble bornant supérieur** consiste en un ensemble d'images de solutions réalisables (les points noirs sur la figure)

- La **zone de recherche** est la portion de l'espace des objectifs qui n'est dominée par aucun des points de l' ensemble bornant supérieur (la zone rouge sur la figure)

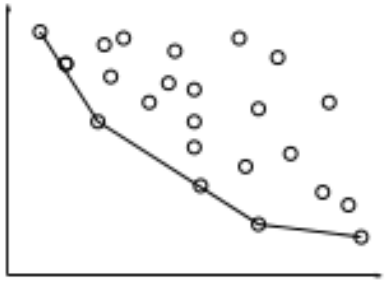


Branch and bound multi-objectifs

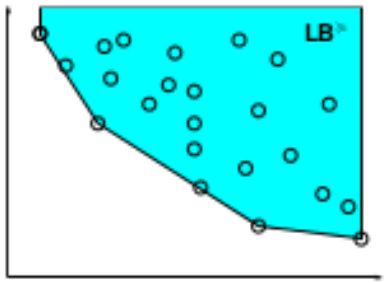
- Un **ensemble bornant inférieur** consiste en un ensemble LB de points tels que tout point réalisable est dominé au sens large par au moins un point de LB



- La **frontière non-dominée** de l'enveloppe convexe des points réalisables est un ensemble bornant inférieur valide

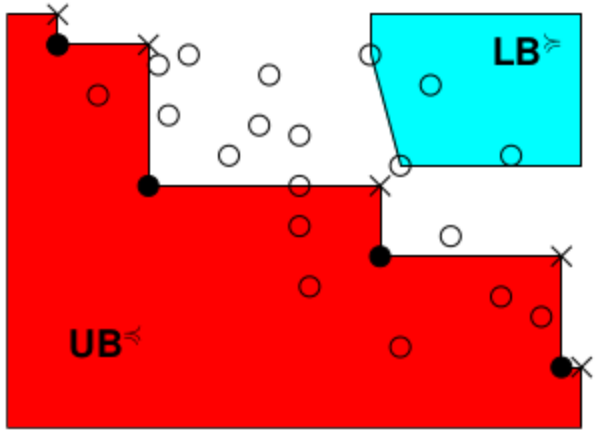


- Tous les points réalisables sont nécessairement au-dessus de l'ensemble bornant inférieur (la zone bleue sur la figure)



Branch and bound multi-objectifs

- En chaque noeud de l'arbre d' énumération, on teste la vacuité de l'intersection entre la zone rouge et la zone bleue
- Cela revient à tester si il n'y a pas de "croix" à l'intérieur de la zone bleue
- Dans ce cas, le sous-ensemble correspondant de solutions ne peut pas inclure de nouveau point non-dominé !



Branch and bound multi-objectifs

Une procédure de branch and bound multi-objectifs a été mise en oeuvre pour le problème de l'arbre couvrant bi-objectifs (Sourd and Spanjaard, 2008) et a permis de multiplier par 10 la taille des instances qui peuvent être résolue dans un temps raisonnable (de 40 à 400 sommets).

Une thèse a été récemment soutenue (Lacour, juillet 2014) qui présente une tentative d'étendre la méthode au cas à tri-objectifs, ce qui rend l'approche plus délicate.

Synthèse

Synthèse

Plusieurs procédures de résolution pour les problèmes combinatoires multi-objectifs, dont :

- algorithme glouton multi-objectifs,
- programmation dynamique multi-objectifs,
- énumération ordonnée,
- méthode en deux phases,
- branch and bound multi-objectifs.

Il y en a d'autres ! (fondées sur la programmation mathématique par exemple)

Un des principaux verrous du domaine :

Les procédures de résolution présentées fonctionnent bien pour deux objectifs. Pour trois objectifs ou plus, les problèmes deviennent plus difficiles, et il reste beaucoup à faire.