Analyse d’algorithmes et programmation

- Heuristiques et métaheuristiques -

Kahina BOUCHAMA

Université de Versailles, Saint-Quentin en Yvelines

Février 2023

Table des matières

L'optimisation combinatoire

- L'optimisation combinatoire (OC) occupe une place très importante en recherche opérationnelle et en informatique. De nombreuses applications peuvent être modélisées sous la forme d'un problème d'OC. Exemple: le problème du voyageur de commerce, l'ordonnancement de tâches, le problème de la coloration de graphes, etc.

L'optimisation combinatoire

- L'optimisation combinatoire (OC) occupe une place très importante en recherche opérationnelle et en informatique. De nombreuses applications peuvent être modélisées sous la forme d'un problème d'OC. Exemple: le problème du voyageur de commerce, l'ordonnancement de tâches, le problème de la coloration de graphes, etc.

- Un problème d'OC consiste à trouver la meilleure solution dans un ensemble discret, dit ensemble des solutions réalisables. En général, cet ensemble est fini mais compte un très grand nombre d'éléments, et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

L'optimisation combinatoire

- L'optimisation combinatoire (OC) occupe une place très importante en recherche opérationnelle et en informatique. De nombreuses applications peuvent être modélisées sous la forme d'un problème d'OC. Exemple: le problème du voyageur de commerce, l'ordonnancement de tâches, le problème de la coloration de graphes, etc.

- Un problème d'OC consiste à trouver la meilleure solution dans un ensemble discret, dit ensemble des solutions réalisables. En général, cet ensemble est fini mais compte un très grand nombre d'éléments, et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

- Pour définir la notion de meilleure solution, une fonction, dite fonction objectif, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution (ou solution optimale) est celle qui minimise ou maximise la fonction objectif.

L'optimisation combinatoire

- L'optimisation combinatoire (OC) occupe une place très importante en recherche opérationnelle et en informatique. De nombreuses applications peuvent être modélisées sous la forme d'un problème d'OC. Exemple: le problème du voyageur de commerce, l'ordonnancement de tâches, le problème de la coloration de graphes, etc.

- Un problème d'OC consiste à trouver la meilleure solution dans un ensemble discret, dit ensemble des solutions réalisables. En général, cet ensemble est fini mais compte un très grand nombre d'éléments, et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

- Pour définir la notion de meilleure solution, une fonction, dite fonction objectif, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution (ou solution optimale) est celle qui minimise ou maximise la fonction objectif.

Dans ce cours, nous allons nous intéresser aux méthodes de résolution des problèmes d'optimisation combinatoire.

Définition d'un POC

Un problème d'optimisation combinatoire peut être défini par :

- Vecteur de variables $x = (x_1, x_2, ..., x_n)$,
- Domaine des variables $D = (D_1, D_2, ..., D_n)$, où les $(D_i)_{i=1,...,n}$ sont des ensembles finis,
- Ensemble de contraintes,
- Une fonction objectif f à minimiser ou à maximiser,
- Ensemble de toutes les solutions réalisable possibles est $S = \{x = (x_1, x_2, , x_n) \in D \ / \ x$ satisfait toutes les contraintes$\}$, l'ensemble S est aussi appelé un espace de recherche.

Définition d'un POC

Un problème d'optimisation combinatoire peut être défini par :

- Vecteur de variables $x = (x_1, x_2, ..., x_n)$,
- Domaine des variables $D = (D_1, D_2, ..., D_n)$, où les $(D_i)_{i=1,...,n}$ sont des ensembles finis,
- Ensemble de contraintes,
- Une fonction objectif f à minimiser ou à maximiser,
- Ensemble de toutes les solutions réalisable possibles est $S = \{x = (x_1, x_2, , x_n) \in D \ / \ x$ satisfait toutes les contraintes$\}$, l'ensemble S est aussi appelé un espace de recherche.

Lorsque les contraintes d'un problème d'OC et la fonction objectif sont linéaires, on parle alors de **Programme Linéaire en Nombres Entiers (PLNE)**.
Dans ce qui suit, on va s'intéresser à cette classe de problèmes.

Résolution des problèmes d'OC

- Trouver une solution optimale dans un ensemble discret et fini est un problème facile en théorie: il suffit d'essayer toutes les solutions, et de comparer leurs qualités pour voir la meilleure.

Résolution des problèmes d'OC

- Trouver une solution optimale dans un ensemble discret et fini est un problème facile en théorie: il suffit d'essayer toutes les solutions, et de comparer leurs qualités pour voir la meilleure.
- Cependant, en pratique, l'énumération de toutes les solutions peut prendre trop de temps; or, le temps de recherche de la solution optimale est un facteur très important et c'est à cause de lui que les problèmes d'optimisation combinatoire sont réputés si difficiles.

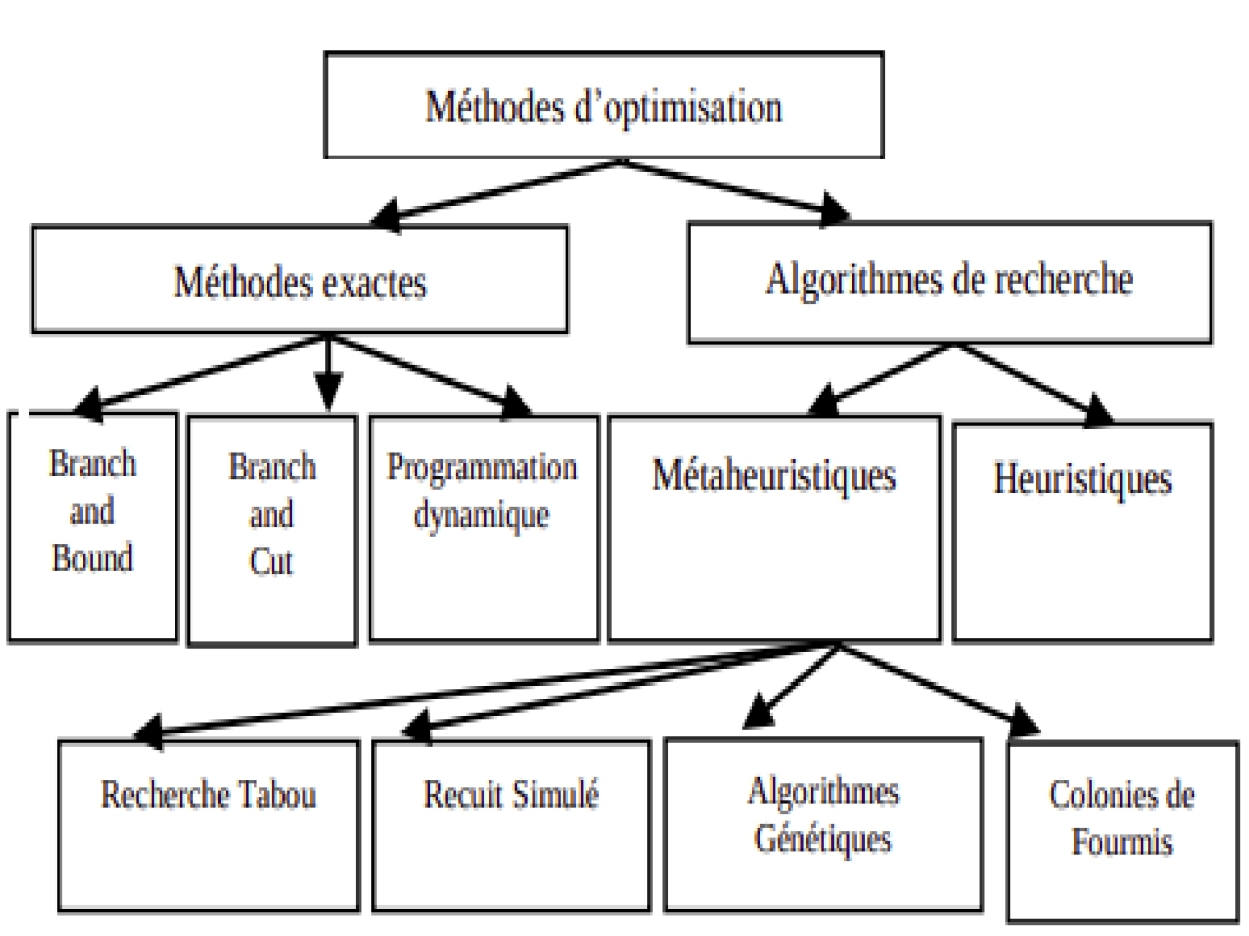
Résolution des problèmes d'OC

- Trouver une solution optimale dans un ensemble discret et fini est un problème facile en théorie: il suffit d'essayer toutes les solutions, et de comparer leurs qualités pour voir la meilleure.
- Cependant, en pratique, l'énumération de toutes les solutions peut prendre trop de temps; or, le temps de recherche de la solution optimale est un facteur très important et c'est à cause de lui que les problèmes d'optimisation combinatoire sont réputés si difficiles.
- La théorie de la complexité donne des outils pour mesurer ce temps de recherche. De plus, comme l'ensemble des solutions réalisables est défini de manière implicite, il est aussi parfois très difficile de trouver ne serait-ce qu'une solution réalisable.

Résolution des problèmes d'OC

- Trouver une solution optimale dans un ensemble discret et fini est un problème facile en théorie: il suffit d'essayer toutes les solutions, et de comparer leurs qualités pour voir la meilleure.
- Cependant, en pratique, l'énumération de toutes les solutions peut prendre trop de temps; or, le temps de recherche de la solution optimale est un facteur très important et c'est à cause de lui que les problèmes d'optimisation combinatoire sont réputés si difficiles.
- La théorie de la complexité donne des outils pour mesurer ce temps de recherche. De plus, comme l'ensemble des solutions réalisables est défini de manière implicite, il est aussi parfois très difficile de trouver ne serait-ce qu'une solution réalisable.
- Pour faire face à cette explosion combinatoire, de nombreuses méthodes de résolution en recherche opérationnelle (RO) et en intelligence artificielle (IA) ont été développées. Elles sont classées en deux catégories : les méthodes exactes et les méthodes approchées.

Méthodes de résolution des POC



Les méthodes exactes

Ce sont des méthodes qui permettent de trouver, à coup sûre, la solution optimale d'un problème, pour une instance de taille finie, en un temps limité et de prouver son optimalité.

Quelques méthodes exactes de résolution des POC:

- La programmation dynamique;
- La méthode séparation et évaluation (Branch and Bound);
- La méthode des coupes.
- Branch & cut (B&C);
- etc.

Les méthodes exactes

Ce sont des méthodes qui permettent de trouver, à coup sûre, la solution optimale d'un problème, pour une instance de taille finie, en un temps limité et de prouver son optimalité.

Quelques méthodes exactes de résolution des POC:

- La programmation dynamique;
- **La méthode séparation et évaluation (Branch and Bound);**
- La méthode des coupes.
- Branch & cut (B&C);
- etc.

La méthode de Branch & Bound

- L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (BB) (Land et Doig 1960), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des noeuds, et des feuilles.

La méthode de Branch & Bound

- L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (BB) (Land et Doig 1960), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des noeuds, et des feuilles.
- Le branch-and-bound est basé sur trois axes principaux:
  - L'évaluation,
  - La séparation,
  - La stratégie de parcours.

L'évaluation

- Elle permet de réduire l'espace de recherche en éliminant quelques sous ensembles qui ne contiennent pas la solution optimale. L'objectif est d'essayer d'évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence.

La méthode de Branch & Bound

- L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (BB) (Land et Doig 1960), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des noeuds, et des feuilles.
- Le branch-and-bound est basé sur trois axes principaux:
  - L'évaluation,
  - La séparation,
  - La stratégie de parcours.

L'évaluation

- Elle permet de réduire l'espace de recherche en éliminant quelques sous ensembles qui ne contiennent pas la solution optimale. L'objectif est d'essayer d'évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence.
- Le B&B élimine les branches comme suit: la recherche d'une solution de coût minimal, consiste à mémoriser la solution de plus bas coût rencontré pendant l'exploration, et à comparer le coût de chaque noeud parcouru à celui de la meilleure solution. Si le coût du noeud considéré est supérieur au meilleur coût, on arrête l'exploration de la branche et toutes les solutions de cette branche seront nécessairement de coût plus élevé que la meilleure solution déjà trouvée.

La séparation

- La séparation consiste à diviser le problème en sous-problèmes.
- En résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est sure d'avoir résolu le problème initial.
- Elle revient à construire un arbre permettant d'énumérer toutes les solutions.
- L'ensemble des noeuds de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des noeuds actifs.

La stratégie de parcours

Largeur d'abord: Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial.

Profondeur d'abord: Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

Meilleur d'abord: Cette stratégie consiste à explorer des sous-problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

Définitions

- La PLNE regroupe l'ensemble des techniques permettant de résoudre des programmes linéaires dont les solutions doivent être entières.
- Formellement, le PLNE (P) s'exprime comme suit:

$$(P)\begin{cases} Max\ Z = c\ x \\ A\ x \leq b \\ x \in Z \end{cases}$$

A d'ordre m x n, b d'ordre m, c d'ordre n et x est d'ordre n.

- Le programme linéaire (PR) associé est:

$$(Q)\begin{cases} Max\ Z = c\ x \\ A\ x \leq b \\ x \geq 0 \end{cases}$$

Quelques résultats

- ***Lemme***: Soient x^*, x^{opt} les solutions optimales respectivement des programmes (P) et (PR) et Z^*, Z^{opt} les valeurs de leurs fonctions objectifs respectivement, alors nous avons :

  $Z^* \leq Z^{opt}$,

- La valeur de la solution optimale du programme (PR) est une borne supérieure à la valeur de la solution optimale entière de P.
- La résolution de (P) ne peut pas se faire par la résolution de (PR) puis d'arrondir ensuite les solutions non entières.
  - En effet, le simple arrondi des variables non entières à des valeurs entières peut rendre la solution non réalisable ou simplement non optimale.

Algorithme de Branch & Bound

X^* : la solution du $(P) \rightarrow$ problème (PLNE)
X^{opt} :la solution du $(PR) \rightarrow$ problème (PLNR)
Z^* : la fonction économique du $(P) \rightarrow$ problème (PLNE)
Z^{opt} :la fonction économique du $(PR) \rightarrow$ problème (PLNR)

1)Résolution du problème relaxé(PR) par la méthode du simplexe :

-Si X^{opt} est entier :fin.
-Sinon, aller à 2).

2)Initialisation :

Soit Z^{opt} obtenu dans(PR) une borne supérieure pour un problème de maximisation respectivement (borne inférieure pour un problème de minimisation)pour (P_i).
Soit n_1 le sommet initial de l'arborescence et son ensemble$(S_1 = S)$.
$Z_1 = Z^{opt}$.

3)Séparation($k^{ième}$ itération) :

Choisir une variable non entière x_l,créer deux branches (deux sommets fils n_{i+1} et n_{i+2}),on obtient deux sous-problèmes sous la forme :

$$\begin{cases} P_{i+1,k} : P_i + \text{la contrainte } x_l \leq [x_l] \\ P_{i+2,k} : P_i + \text{la contrainte } x_l \geq [x_l] + 1 \end{cases}$$

Avec $[x_l]$: la partie entiére de x_l

4)Résolution des sous-problèmes :

Résoudre chaque sous-problème en utilisant le simplexe ou le dual simplexe.

5)Évaluation :

Examiner chaque sous-ensemble :
On peux tailler un sommet si :
* La solution est non réalisable.
* $Z_1 \leq Z$ pour un problème de maximisation ($Z_1 \geq Z$ pour un problème de minimisation), avec Z la solution du sous-problème.
* La solution est non entière et son Z inférieur ou égale à une solution entière pour un problème de maximisation (supérieure ou égale pour un problème de minimisation) .
Il est inutile de sépare si :
* La solution est entière.

6)test :

S'il y a plus de sous-ensembles à séparer ,alors :
On compare tous les Z des solutions entières et on prend la plus grande d'entre elles soit Z^* pour un problème de maximisation (la plus petite pour un problème de minimisation). Elle sera la valeur de la fonction économique de la solution optimale X^* de notre problème (P).
Sinon retour à 3).

Remarque :

$Z^* \leq Z^{opt}$ pour un problème de maximisation ($Z^* \geq Z^{opt}$ pour un problème de minimisation).
Dans l'itération 3)pour séparer les sous-problèmes on utilise l'une des stratégies cités précédemment.

Algorithme de Branch & Bound

Un exemple détaillé du fonctionnement de l'algorithme de B&B peut être trouvé dans le fichier en annexe.

3.4 Exemple d'application

Forme générale du problème :

$$(P)\begin{cases} (PR)\begin{cases} \max Z = x_1 + 2x_2 & \\ 4x_1 - 3x_2 & \leq 2 \\ -2x_1 + x_2 & \leq 1 \\ -6x_1 + 14x_2 & \leq 35 \\ x_1, x_2 \geq 0 & \end{cases} \\ x_1, x_2 \in \mathbf{N} \end{cases}$$

1)Résolution du problème (PR) :

- Sa forme standard :

$$(PR)\begin{cases} maxZ = x_1 + 2x_2 & \\ 4x_1 - 3x_2 + x_3 & = 2 \\ -2x_1 + x_2 + x_4 & = 1 \\ -6x_1 + 14x_2 + x_5 & = 35 \\ x_j \geq 0, j = 1, ..., 5. & \end{cases}$$

• Résoudre le(PR)par la méthode du simplexe :

		c_i	1	2	0	0	0	
c_b	base	b	x_1	x_2	x_3	x_4	x_5	θ
0	x_3	2	4	-3	1	0	0	/
0	x_4	1	-2	**1**	0	1	0	1
0	x_5	35	-6	14	0	0	1	35/14
		E_j	-1	**-2**	0	0	0	

•x_4 sort de la base.
•x_2 entre dans la base.

		c_i	1	2	0	0	0	
c_b	base	b	x_1	x_2	x_3	x_4	x_5	θ
0	x_3	5	-2	0	1	3	0	/
2	x_2	1	-2	1	0	1	0	/
0	x_5	21	**22**	0	0	-14	1	21/22
		E_j	**-5**	0	0	2	0	

•x_5 sort de la base.
•x_1 entre dans la base.

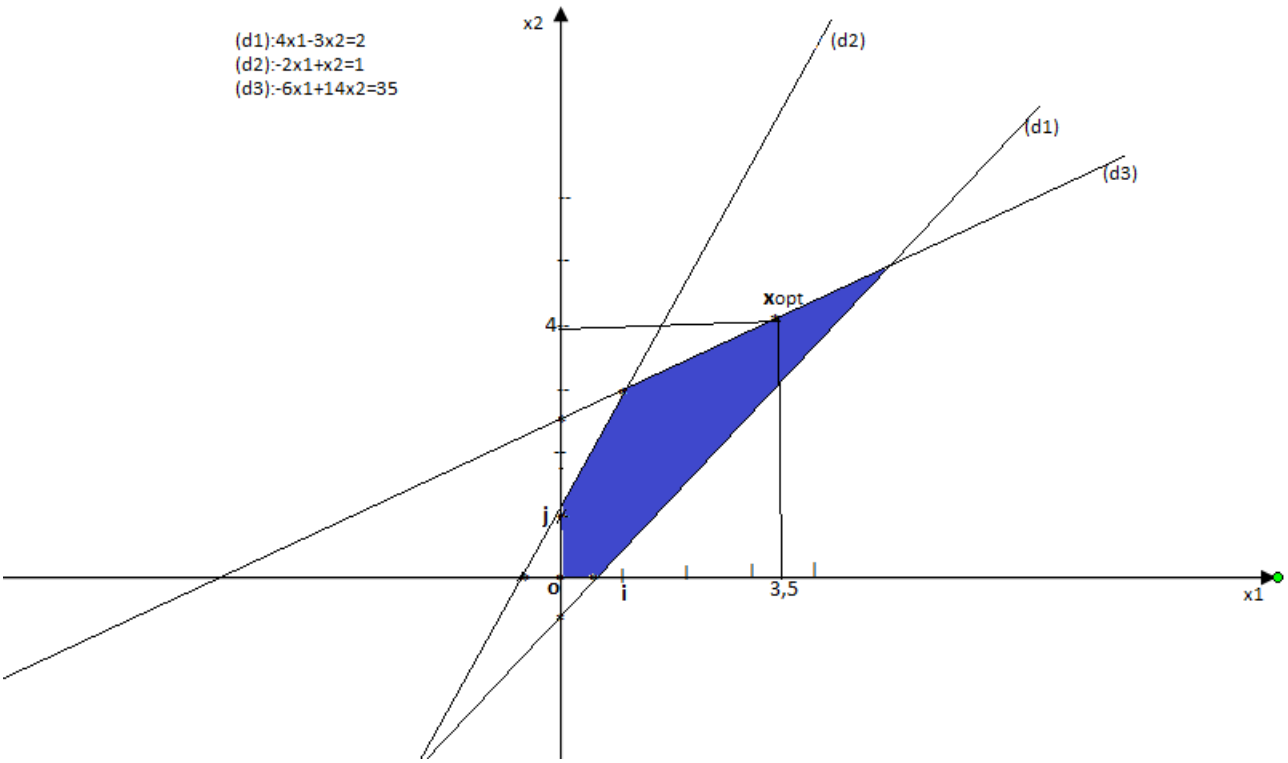
		c_i	1	2	0	0	0	
c_b	base	b	x_1	x_2	x_3	x_4	x_5	θ
0	x_3	76/11	0	0	1	**19/11**	1/11	76/19
2	x_2	32/11	0	1	0	-3/11	1/19	/
1	x_1	21/22	1	0	0	-7/11	1/22	/
		E_j	0	0	0	**-13/11**	5/22	

•x_3 sort de la base.
•$x_4$entre dans la base.

		c_i	1	2	0	0	0	
c_b	base	b	x_1	x_2	x_3	x_4	x_5	θ
0	x_4	4	0	0	11/19	1	1/19	/
2	x_2	4	0	1	3/19	0	2/19	/
1	x_1	7/2	1	0	7/19	0	3/38	/
		E_j	0	0	13/19	0	11/38	

• tous les $E_j \geq 0 \Rightarrow X^{opt} = (7/2, 4)$.
• Z^{opt}=(7/2)+2(4)=23/2.
$d'ou \begin{cases} x^{opt}\text{=(3.5,4).} \\ Z^{opt}\text{=11.5.} \end{cases}$

• **Résolution graphique de (PR) :**



2)Initialisation :

• Z^{opt}est la borne supérieure de (P).

• Soit (P_1) le problème initial (le sommet initial de l'arborescence) tel que :

$$\left[\begin{array}{l} Z_1{=}Z^{opt}{=}11.5. \\ X_1{=}3.5. \\ X_2{=}4. \end{array}\right]$$

(P_1) a des variables qui ne vérifient pas les contraintes d'intégrités d'où on passe à l'étape 3).

3)Séparation :

x_1 n'est pas entier on aura le modèle linéaire courant, ici (P_1) est divisé en deux sous-problèmes sous la forme :

$$\left\{\begin{array}{l} (P_2) = (P_1) + \textit{contrainte } x_1 \leq [x_1]. \\ (P_3) = (P_1) + \textit{contrainte } x_1 \geq [x_1] + 1. \end{array}\right.$$

avec $[x_1]$ la partie entière de x_1.

4)Résolution des sous-problèmes :

a)On résout (P_2) tq :
$(P_2) = (P_1)+$ la contrainte $x_1 \leq [x_1]$ c-à-d :
$(P_1)+$ la contrainte $x_1 \leq 3 \Rightarrow x_1 + x_6{=}3$.....★.
On rajoute au dernier tableau de (PR)la contrainte★et on applique le dual du simplexe :

		c_i	1	2	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6
0	x_4	4	0	0	11/19	1	1/19	0
2	x_2	4	0	1	3/19	0	2/19	0
1	x_1	7/2	1	0	7/19	0	3/38	0
0	x_6	3	**1**	0	0	0	0	1

•x_1 est une variable de base,on multiplie la $3^{éme}$ ligne par(-1) , on l'additionne à la $4^{éme}$ ligne . On obtient le tableau suivant :

		c_i	1	2	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6
0	x_4	4	0	0	11/19	1	1/19	0
2	x_2	4	0	1	3/19	0	2/19	0
1	x_1	7/2	1	0	7/19	0	3/38	0
0	x_6	**-1/2**	0	0	**-7/19**	0	-3/38	1
		E_j	0	0	13/19	0	11/38	0

•x_6 sort de la base.
•x_3 entre dans la base.

		c_i	1	2	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6
0	x_4	45/14	0	0	0	1	-1/14	11/7
2	x_2	53/14	0	1	0	0	1/14	3/7
1	x_1	3	1	0	0	0	0	1
0	x_3	19/14	0	0	1	0	3/14	-19/7
		E_j	0	0	0	0	1/7	13/7

• Tous les $b_j \geq 0$,la solution de (P_2) est :

$$\begin{bmatrix} Z_2=10.57. \\ X_1=3. \\ X_2=3.79. \end{bmatrix}$$

b)On résout (P_3) tq :
(P_3)=(P_1) +la contrainte $x_1 \geq [x_1]+1$ c-à-d :
(P_1) +la contrainte $x_1 \geq 4 \Rightarrow x_1$-$x_6$=4
$-x_1+x_6 = -4$....★.
On rajoute la contrainte ★ au dernier tableau de (PR)et on applique le dual simplexe :

		c_i	1	2	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6
0	x_4	4	0	0	11/19	1	1/19	0
2	x_2	4	0	1	3/19	0	2/19	0
1	x_1	7/2	1	0	7/19	0	3/38	0
0	x_6	-4	**-1**	0	0	0	0	1

•x_1 est une variable de base,on additionne la $3^{éme}$ ligne à la $4^{éme}$ ligne.On obtient le tableau suivant :

		c_i	1	2	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6
0	x_4	4	0	0	11/19	1	1/19	0
2	x_2	4	0	1	3/19	0	2/19	0
1	x_1	7/2	1	0	7/19	0	3/38	0
0	x_6	-1/2	0	0	7/19	0	3/38	1
		E_j	0	0	13/19	0	11/38	0

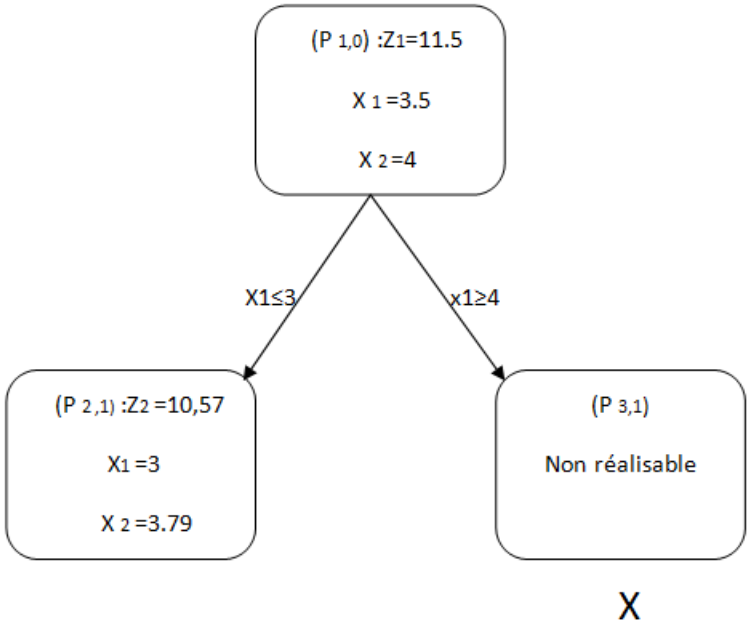
• Pas de pivot négatif ,(P_3) est non-réalisable .

5)Évaluation :

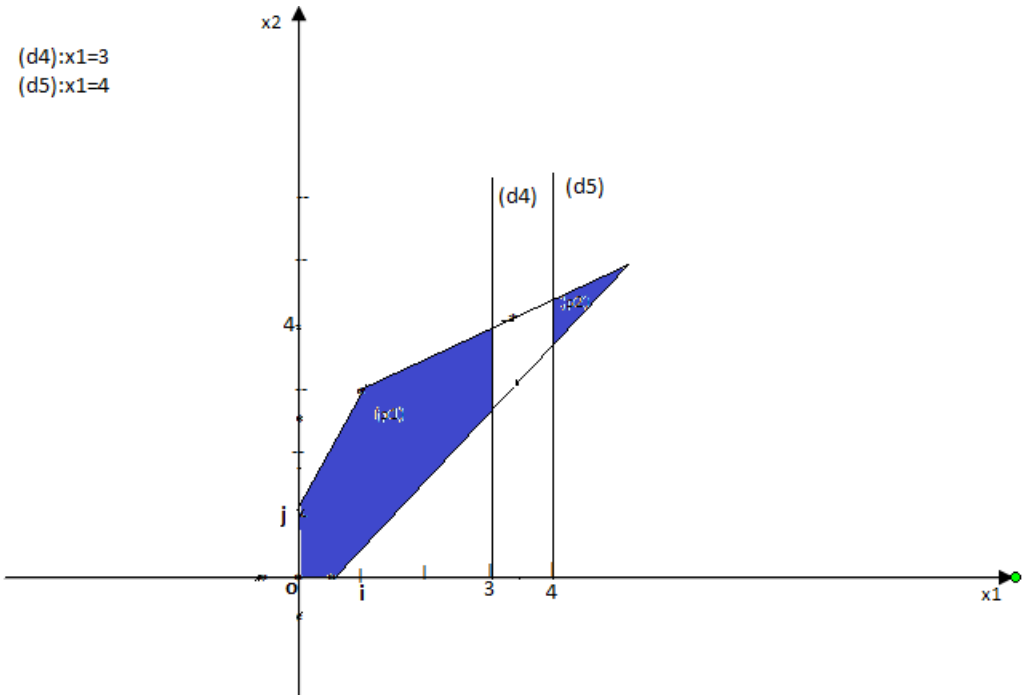
•(P_2)admet une solution non-entière et la valeur de $Z_2 = 10.75 \leq Z_1$ donc on peut séparer le sommet S_2 .
•(P_3) non réalisable donc on taille le sommet S_3 .

6) Test :

Il existe un sous-ensemble qui peut être séparé donc on retourne à 3).



• **Interprétation graphique de la séparation à partir de**(P_1) **:**



3)Séparation :

x_2 n'est pas entier d'où le modèle linéaire courant ici (P_2)est divisé en deux sous-problèmes sous la forme :

$$\begin{cases} (P_4) = (P_2) + contrainte\ x_2 \leq [x_2]. \\ (P_5) = (P_2) + contrainte\ x_2 \geq [x_2] + 1. \end{cases}$$

avec $[x_2]$ la partie entière de x_2.

4)Résolution des sous-problèmes :

a)On résout (P_4) tq :
$(P_4) = (P_2) + contrainte\ x_2 \leq [x_2]$ c-à-d :
$(P_2) + contrainte\ x_2 \leq 3 \Longrightarrow x_2 + x_7 = 3$.....★.
On rajoute au dernier tableau de(P_2) la contrainte★ et on applique le dual simplexe :

		c_i	1	2	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	45/14	0	0	0	1	-1/14	11/7	0
2	x_2	53/14	0	1	0	0	1/14	3/7	0
1	x_1	3	1	0	0	0	0	1	0
0	x_3	19/14	0	0	1	0	3/14	-19/7	0
0	x_7	3	0	**1**	0	0	0	0	1

•x_2 est une variable de base on multiplie la $2^{éme}$ligne par (-1),on l'additionne à la $5^{éme}$ ligne.On obtient le tableau suivant :

		c_i	1	2	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	45/14	0	0	0	1	-1/14	11/7	0
2	x_2	53/14	0	1	0	0	1/14	3/7	0
1	x_1	3	1	0	0	0	0	1	0
0	x_3	19/14	0	0	1	0	3/14	-19/7	0
0	x_7	-11/14	0	0	0	0	**-1/14**	-3/7	1
		E_j	0	0	0	0	1/7	13/7	0

•x_7 sort de la base.
•x_5 entre dans la base .

		c_i	1	2	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	4	0	0	0	1	0	2	-1
2	x_2	3	0	1	0	0	0	0	1
1	x_1	3	1	0	0	0	0	1	0
0	x_3	-1	0	0	1	0	0	**-4**	3
0	x_5	11	0	0	0	0	1	6	-14
		E_j	0	0	0	0	0	1	2

•x_3 sort de la base.
•x_6 entre dans la base .

		c_i	1	2	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	7/2	0	0	1/2	1	0	0	1/2
2	x_2	3	0	1	0	0	0	0	1
1	x_1	11/4	1	0	1/4	0	0	0	3/4
0	x_6	1/4	0	0	-1/4	0	0	1	-3/4
0	x_5	19/2	0	0	3/2	0	1	0	-19/2
		E_j	0	0	1/4	0	0	0	11/4

• tous les $b_j \geq 0$.La solution de (P_4) est :

$$\begin{bmatrix} Z_4\text{=}8.75. \\ X_1\text{=}2.75. \\ X_2\text{=}3. \end{bmatrix}$$

b)On résout (P_5) tq : $(P_5) = (P_2)$+la contrainte $x_2 \geq [x_2] + 1$ c-à-d :
(P_2) +la contrainte $x_2 \geq 4 \Longrightarrow x_2 - x_7 = 4 \Longrightarrow -x_2 + x_7 = -4$.....★.
on rajoute au dernier tableau de(P_2) la contrainte★ et on applique le dual simplexe :

		c_i	1	2	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	45/14	0	0	0	1	-1/14	11/7	0
2	x_2	53/14	0	1	0	0	1/14	3/7	0
1	x_1	3	1	0	0	0	0	1	0
0	x_3	19/14	0	0	1	0	3/14	-19/7	0
0	x_7	-4	0	**-1**	0	0	0	0	1

•x_2 est une variable de base on additionne la $2^{éme}$ ligne à la $5^{éme}$ ligne.On obtient le tableau suivant :

		c_i	1	2	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	x_4	45/14	0	0	0	1	-1/14	11/7	0
2	x_2	53/14	0	1	0	0	1/14	3/7	0
1	x_1	3	1	0	0	0	0	1	0
0	x_3	19/14	0	0	1	0	3/14	-19/7	0
0	x_7	-3/14	0	0	0	0	1/14	3/7	1
		E_j	0	0	0	0	1/7	13/7	0

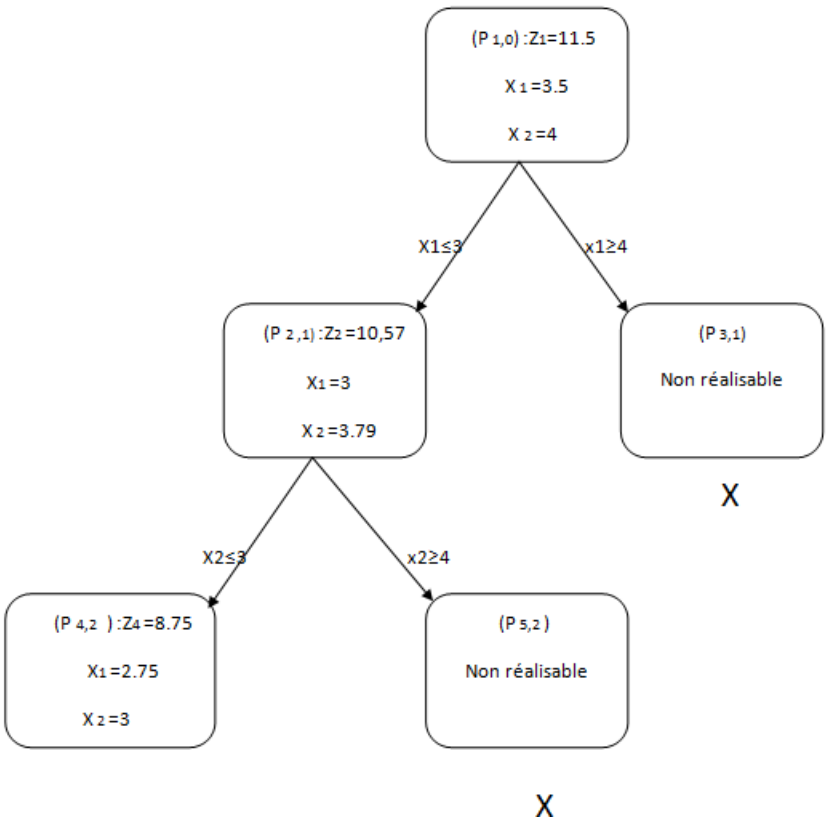
• Pas de pivot négatif.(P_5) n'admet pas de solution .

5)Évaluation :

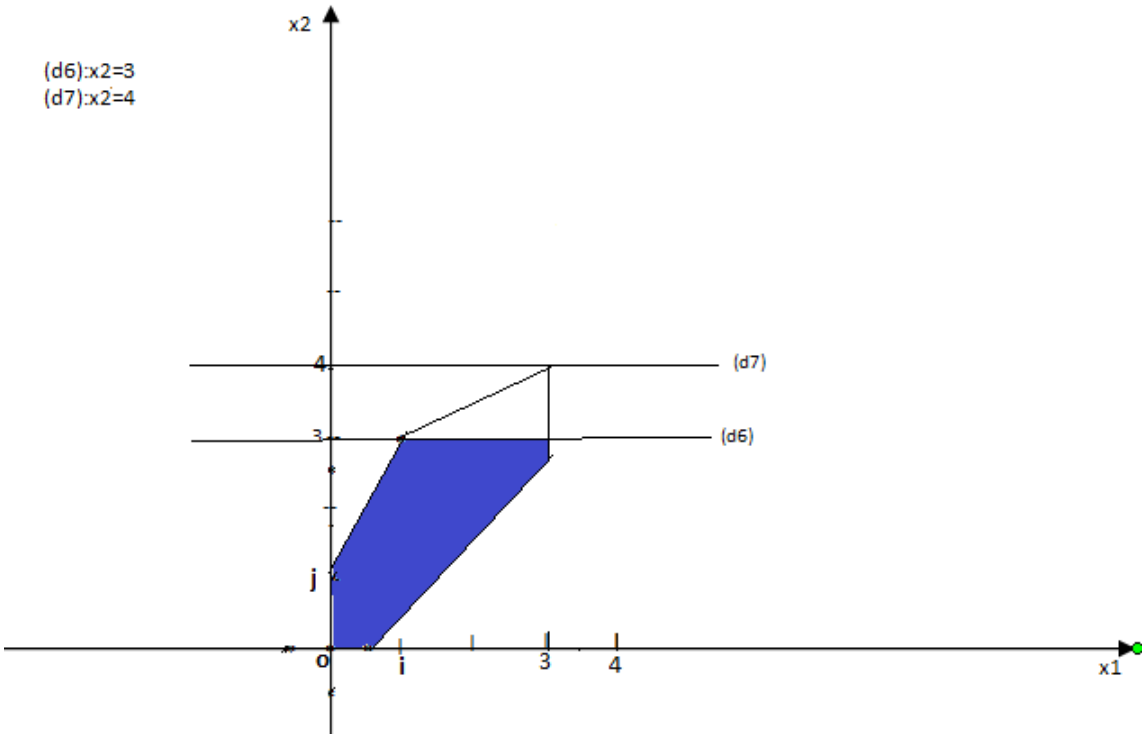
•(P_4) a une solution non entière et la valeur de $Z_4 \leq Z_1$ donc on peut le séparer .
•(P_5) non réalisable donc on peut le tailler .

6)Test :

Il existe un sous-ensemble qui peut être séparé donc retour à 3).



• **Interprétation graphique de la séparation à partir de (P_2) :**



3)Séparation :

x_1 n'est pas entier d'où le modèle linéaire courant ici (P_4) est divisé en deux sous-problèmes sous la forme :

$$\begin{cases} (P_6) = (P_4) + contrainte\ x_1 \leq [x_1]. \\ (P_7) = (P_4) + contrainte\ x_1 \geq [x_1] + 1. \end{cases}$$

avec $[x_1]$ la partie entière de x_1.

4)Résolution des sous-problèmes :

a)On résout (P_6) tq :

$(P_6) = (P_4) + contrainte\ x_1 \leq [x_1]$ c-à-d :

(P_4) +la contrainte $x_1 \leq 2 \Longrightarrow x_1 + x_8 = 2$.....★.

On rajoute au dernier tableau de(P_4) la contrainte★ et on applique le dual simplexe :

		c_i	1	2	0	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	x_4	7/2	0	0	1/2	1	0	0	1/2	0
2	x_2	3	0	1	0	0	0	0	1	0
1	x_1	11/4	1	0	1/4	0	0	0	3/4	0
0	x_6	1/4	0	0	-1/4	0	0	1	-3/4	0
0	x_5	19/2	0	0	3/2	0	1	0	-19/2	0
0	x_8	2	**1**	0	0	0	0	0	0	1

•x_1 est une variable de base on multiplie la $3^{éme}$ ligne par (-1) et on l'additionne à la $6^{éme}$ ligne.On obtient le tableau suivant :

		c_i	1	2	0	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	x_4	7/2	0	0	1/2	1	0	0	1/2	0
2	x_2	3	0	1	0	0	0	0	1	0
1	x_1	11/4	1	0	1/4	0	0	0	3/4	0
0	x_6	1/4	0	0	-1/4	0	0	1	-3/4	0
0	x_5	19/2	0	0	3/2	0	1	0	-19/2	0
0	x_8	-3/4	0	0	**-1/4**	0	0	0	-3/4	1
		E_j	0	0	1/4	0	0	0	11/4	0

•x_8 sort de la base.
•x_3 entre dans la base.

		c_i	1	2	0	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	x_4	2	0	0	0	1	0	0	-1	2
2	x_2	3	0	1	0	0	0	0	1	0
1	x_1	2	1	0	0	0	0	0	0	1
0	x_6	1	0	0	0	0	0	1	0	-1
0	x_5	5	0	0	0	0	1	0	14	6
0	x_3	3	0	0	1	0	0	0	3	-4
		E_j	0	0	0	0	0	0	2	1

• Tous les $b_j \geq 0$.La solution de (P_6) est :

$$\begin{bmatrix} Z_6=8. \\ X_1=2. \\ X_2=3. \end{bmatrix}$$

b)On résout (P_7) telle que :
$(P_7) = (P_4)$+ la contrainte $x_1 \geq [x_1] + 1$ c-à-d :
(P_4)+ la contrainte $x_1 \geq 3 \Longrightarrow x_1 - x_8 = 3 \Longrightarrow -x_1 + x_8 = -3$.....★.
On rajoute au dernier tableau de(P_4) la contrainte ★ et on applique le dual simplexe :

		c_i	1	2	0	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	x_4	7/2	0	0	1/2	1	0	0	1/2	0
2	x_2	3	0	1	0	0	0	0	1	0
1	x_1	11/4	1	0	1/4	0	0	0	3/4	0
0	x_6	1/4	0	0	-1/4	0	0	1	-3/4	0
0	x_5	19/2	0	0	3/2	0	1	0	-19/2	0
0	x_8	-3	**-1**	0	0	0	0	0	0	1

•x_1 est une variable de base d'où on additionne la $3^{éme}$ligne à la $6^{éme}$ ligne.On obtient le tableau suivant :

		c_i	1	2	0	0	0	0	0	0
c_b	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	x_4	7/2	0	0	1/2	1	0	0	1/2	0
2	x_2	3	0	1	0	0	0	0	1	0
1	x_1	11/4	1	0	1/4	0	0	0	3/4	0
0	x_6	1/4	0	0	-1/4	0	0	1	-3/4	0
0	x_5	19/2	0	0	3/2	0	1	0	-19/2	0
0	x_8	-1/4	0	0	1/4	0	0	0	3/4	1
		E_j	0	0	1/4	0	0	0	11/4	0

• Pas de pivot négatif .(P_7) non-réalisable.

5)évaluation :

• On a (P_6) a une solution entière et son $Z_6 \leq Z_1$ donc inutile de le séparer.

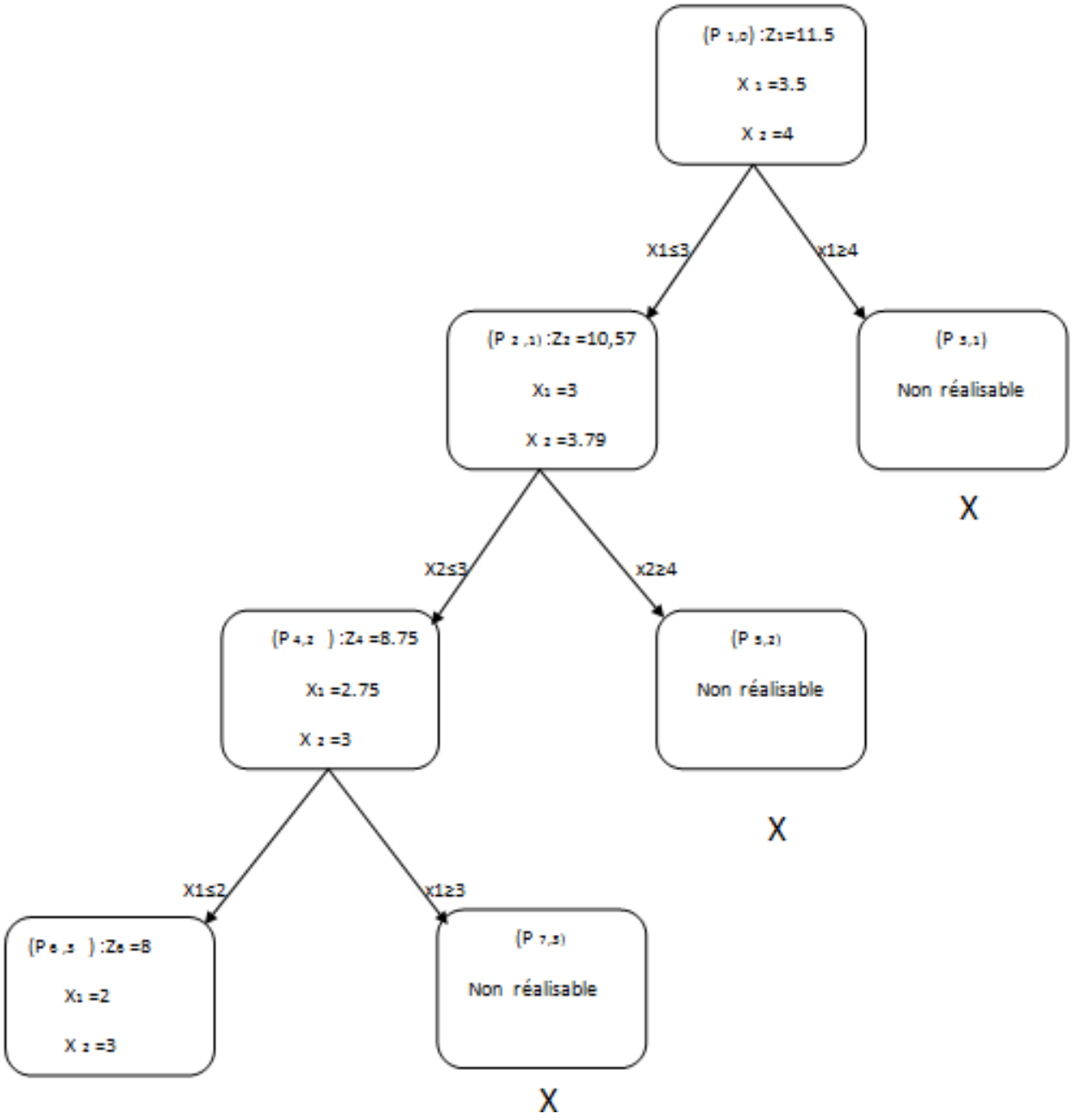
• On a (P_7) non réalisable donc on peut le tailler.

6)Test :

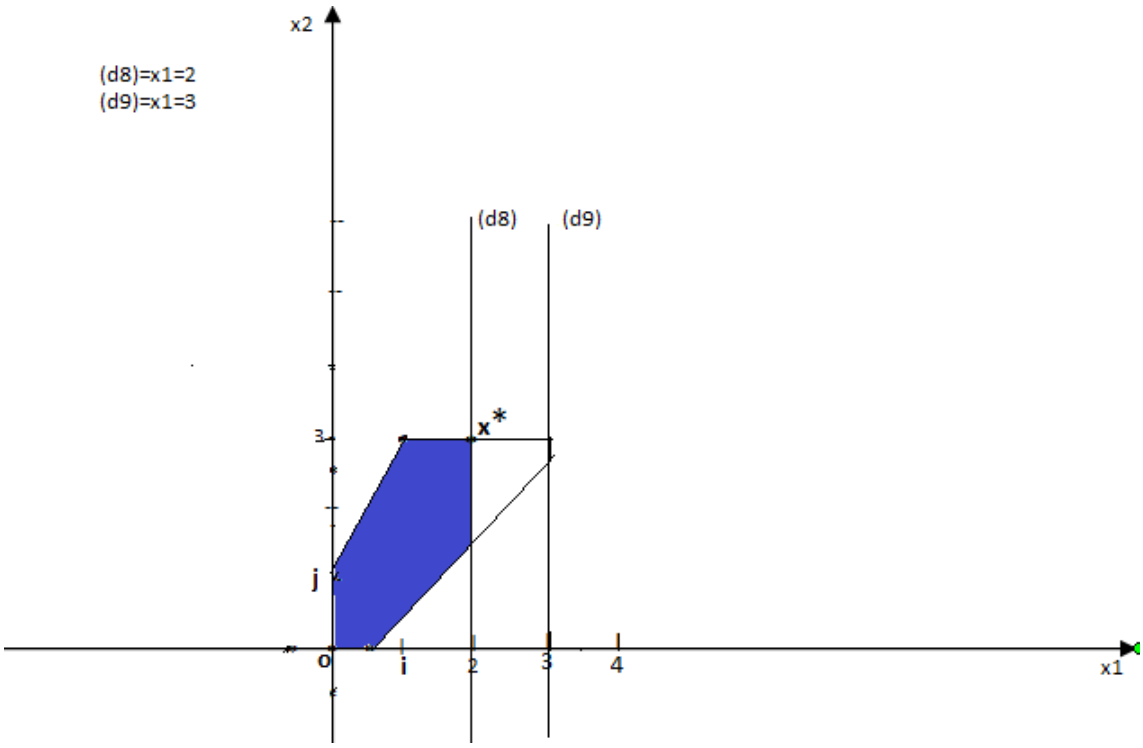
Il n'y a plus de sous-ensembles à séparer alors on compare tous les Z des solutions entières et on prend la plus grande car on a un problème de maximisation .Ici on a $Z_6 = 8$ est la seule qui a une solution entière .

D'où la solution de notre problème est :

X^*=(2,3)et son $Z^* = 8$.



• **Interprétation graphique de la séparation à partir de** (P_4) **:**



Remarque :
On a utilisé la stratégie :
Le meilleur d'abord.

Les heuristiques

Définition

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale.

- L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte.

Les heuristiques

Définition

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale.

- L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte.
- Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée.

Définition

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale.

- L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte.
- Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée.
- Les heuristiques peuvent être classées en deux catégories :
  1. Les méthodes constructives qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue;
  2. Les méthodes de fouilles locales qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage.

Définition:

Les métaheuristiques sont des heuristiques génériques qu'ils faut adapter à chaque problème. Elles permettent généralement d'obtenir une solution de très bonne qualité pour des pro- blèmes issus des domaines de la recherche opérationnelle ou de l'ingénie- rie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

Définition:

Les métaheuristiques sont des heuristiques génériques qu'ils faut adapter à chaque problème. Elles permettent généralement d'obtenir une solution de très bonne qualité pour des pro- blèmes issus des domaines de la recherche opérationnelle ou de l'ingénie- rie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

- La plupart des métaheuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à des problèmes comme l'explosion combinatoire.

Les métaheuristiques

Définition:

Les métaheuristiques sont des heuristiques génériques qu'ils faut adapter à chaque problème. Elles permettent généralement d'obtenir une solution de très bonne qualité pour des pro- blèmes issus des domaines de la recherche opérationnelle ou de l'ingénie- rie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

- La plupart des métaheuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à des problèmes comme l'explosion combinatoire.
- Une métaheuristique peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné.

Définition:

Les métaheuristiques sont des heuristiques génériques qu'ils faut adapter à chaque problème. Elles permettent généralement d'obtenir une solution de très bonne qualité pour des pro- blèmes issus des domaines de la recherche opérationnelle ou de l'ingénie- rie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

- La plupart des métaheuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à des problèmes comme l'explosion combinatoire.
- Une métaheuristique peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné.
- Plusieurs d'entre elles sont souvent inspirées par des systèmes naturels dans de nombreux domaines tels que : la biologie (algorithmes génétiques) la physique (recuit simulé), et aussi l'éthologie (algorithmes de colonies de fourmis).

Classification des métaheuristiques

Il existe plusieurs façons de classer les métaheuristiques. On va s'intéresser ici aux méthodes qui se basent sur les trajectoires et aux méthodes basées sur des populations.

- **Méthodes de trajectoire** : Manipulent un seul point à la fois et tentent itérativement d'améliorer ce point. Elles construisent une trajectoire dans l'espace des points en tentant de se diriger vers des solutions. Par exemple :
  - La recherche locale.
  - Le recuit simulé [Kirkpatrick et al., 1983].
  - La recherche tabou [Glover, 1986].
  - La recherche à voisinages variables (VNS) [Mladenović et Hansen, 1997].

- Méthodes qui travaillent avec une **population de points** : en tout temps on dispose d'une "base" de plusieurs points, appelée population. L'exemple le plus connu est l'algorithme génétique.

Les algorithmes génétiques (AG)

- Les algorithmes génétiques (AGs) sont des algorithmes fondés sur les mécanismes de la sélection naturelle et de la génétique, utilisant les principes de la survie des structures les mieux adaptées.

- Ces algorithmes fabriquent des chromosomes qui codent chacun une solution potentielle à un problème donné à chaque étape (appelée génération), ces chromosomes se combinent, se mutent et sont sélectionnés en fonction de leurs qualités à répondre au problème afin de les explorer dans la génération suivante avec l'espoir d'améliorer la performance qui en résulterait.

Fonctionnement des AGs

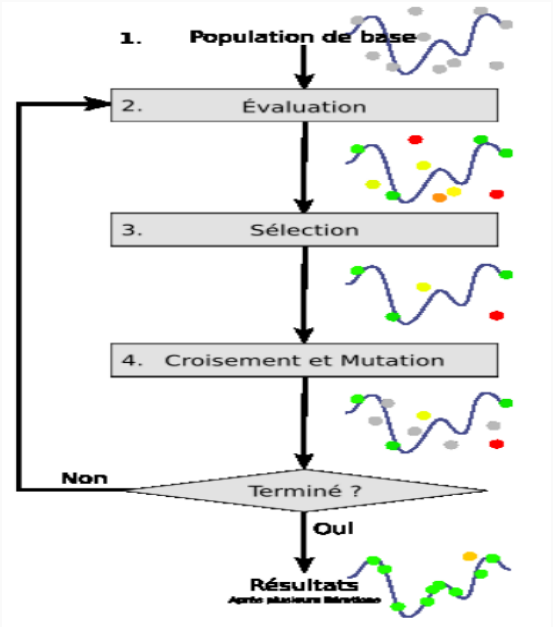
- Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative.

Fonctionnement des AGs

- Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative.
- Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation.

Fonctionnement des AGs

- Leur fonctionnement est extrêmement simple. On part avec une population de solutions potentielles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative.
- Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation.
- On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante.



Terminologie des AGs

Individu Les individus correspondent aux « solutions » du problème à optimiser. Ces solutions doivent être codées pour que le traitement puisse être effectué par l'algorithme génétique. Cette représentation codée d'une solution est appelée chromosome, et est composée de gènes. Chaque gène peut représenter une variable, un élément de la solution, ou encore une partie plus abstraite.

Terminologie des AGs

Individu Les individus correspondent aux « solutions » du problème à optimiser. Ces solutions doivent être codées pour que le traitement puisse être effectué par l'algorithme génétique. Cette représentation codée d'une solution est appelée chromosome, et est composée de gènes. Chaque gène peut représenter une variable, un élément de la solution, ou encore une partie plus abstraite.

Population initiale au départ d'un algorithme génétique, il faut créer une population d'individus. Ces individus sont générés par une fonction simple. Cette fonction affecte à chaque individu qu'elle génère une valeur aléatoire pour chacun de ses gènes. L'algorithme génétique peut également utiliser comme population de départ une population déjà créée a priori qui peut être le résultat d'une autre stratégie, la solution serait dans ce cas certes meilleure puisqu'on part d'une solution approchée qui substitue une solution aléatoire, on parle alors d'hybridation de méthodes.

Terminologie des AGs

Fitness d'un individu Le calcul de la qualité d'un individu est essentiel aux algorithmes génétiques. Cette fonction donne, en valeur numérique (habituellement réelle), la qualité d'un individu. C'est selon cette valeur numérique que sont calculées les chances de sélection de cet individu.

Terminologie des AGs

Fitness d'un individu Le calcul de la qualité d'un individu est essentiel aux algorithmes génétiques. Cette fonction donne, en valeur numérique (habituellement réelle), la qualité d'un individu. C'est selon cette valeur numérique que sont calculées les chances de sélection de cet individu.

Le codage Le codage est un processus très important des algorithmes génétiques, il s'agit d'une représentation formelle des individus pour faciliter l'implémentation d'un AG en adoptant un description adéquate aux opérateurs génétiques et à la fonction fitness. Il a été prouvé empiriquement que le codage jour un rôle primordial dans la vitesse et l'efficacité des AGs.

Terminologie des AGs

Fitness d'un individu Le calcul de la qualité d'un individu est essentiel aux algorithmes génétiques. Cette fonction donne, en valeur numérique (habituellement réelle), la qualité d'un individu. C'est selon cette valeur numérique que sont calculées les chances de sélection de cet individu.

Le codage Le codage est un processus très important des algorithmes génétiques, il s'agit d'une représentation formelle des individus pour faciliter l'implémentation d'un AG en adoptant un description adéquate aux opérateurs génétiques et à la fonction fitness. Il a été prouvé empiriquement que le codage jour un rôle primordial dans la vitesse et l'efficacité des AGs.

Fonction d'évaluation Pour calculer le coût d'un point de l'espace de recherche, on utilise une fonction d'évaluation. L'évaluation d'un individu ne dépendant pas de celle des autres individus, le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus ayant le meilleur coût en fonction de la population courante : c'est le rôle de la fonction fitness. Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

Les opérateurs génétiques

Les algorithmes génétiques sont basés sur un phénomène naturel : l'évolution. Plus précisément, ils supposent, qu'a priori, deux individus adaptés à leur milieu donnent, par recombinaison de leurs gènes, des individus mieux adaptés. Pour ce faire, trois opérateurs sont à disposition : la sélection, le croisement et la mutation.

La sélection sert à choisir dans l'ensemble de la population les individus qui participeront à la reproduction. Ainsi les meilleurs individus ont plus de chance de survivre et de se reproduire. Il existe plusieurs méthodes dans la littérature.

Les opérateurs génétiques

Les algorithmes génétiques sont basés sur un phénomène naturel : l'évolution. Plus précisément, ils supposent, qu'a priori, deux individus adaptés à leur milieu donnent, par recombinaison de leurs gènes, des individus mieux adaptés. Pour ce faire, trois opérateurs sont à disposition : la sélection, le croisement et la mutation.

La sélection sert à choisir dans l'ensemble de la population les individus qui participeront à la reproduction. Ainsi les meilleurs individus ont plus de chance de survivre et de se reproduire. Il existe plusieurs méthodes dans la littérature.

Le croisement combine les gènes des deux individus parents pour donner deux nouveaux chromosomes d'individus enfants. La zone de croisement est généralement choisie aléatoirement dans les chromosomes. Les méthodes de croisement sont liées au codage.

Les opérateurs génétiques

Les algorithmes génétiques sont basés sur un phénomène naturel : l'évolution. Plus précisément, ils supposent, qu'a priori, deux individus adaptés à leur milieu donnent, par recombinaison de leurs gènes, des individus mieux adaptés. Pour ce faire, trois opérateurs sont à disposition : la sélection, le croisement et la mutation.

La sélection sert à choisir dans l'ensemble de la population les individus qui participeront à la reproduction. Ainsi les meilleurs individus ont plus de chance de survivre et de se reproduire. Il existe plusieurs méthodes dans la littérature.

Le croisement combine les gènes des deux individus parents pour donner deux nouveaux chromosomes d'individus enfants. La zone de croisement est généralement choisie aléatoirement dans les chromosomes. Les méthodes de croisement sont liées au codage.

La mutation La mutation modifie aléatoirement un petit nombre de gènes, avec un faible taux de probabilité. Comme pour le croisement, la mutation dépend du problème.

Les opérateurs génétiques

La réinsertion Pour insérer de nouveaux individus dans la population, on évalue les individus obtenus suite aux croisements et aux mutation, on peut éliminer les moins bon parmi les éléments de la populations courante et les nouveaux, remplacer un des deux parents, ...

Les opérateurs génétiques

La réinsertion Pour insérer de nouveaux individus dans la population, on évalue les individus obtenus suite aux croisements et aux mutation, on peut éliminer les moins bon parmi les éléments de la populations courante et les nouveaux, remplacer un des deux parents, ...

Les paramètres des AGs:

- Taille de la population
- Probabilité de croisement
- Probabilité de mutation
- Critère d'arrêt

Pseudo-code des AGs

Algorithme 1: Algorithme génétique

Initialiser la population avec un ensemble de combinaisons de E
tant que *critères d'arrêt non atteints* **faire**
    Sélectionner des combinaisons de la population
    Créer de nouvelles combinaisons par recombinaison et mutation
    Mettre à jour la population
retourner *la meilleure combinaison ayant appartenu à la population*

4.3.6 Algorithme évolutionnaire

Exemple

Minimisation de $f(x) = x^2$

- On considère une population de 10 individus, représentés chacun par :
  - une valeur $x_{i,\ i=1 \text{ à } 10}$ comprise entre −16 et +16
  - son évaluation $f(x_i)$

- L'étape de reproduction comprend 4 étapes :
  - sélection des 8 meilleurs parents parmi les 10 → 8 parents
  - appariement des parents sélectionnés par paires → 4 paires de parents
  - croisement de chaque paire de parents pour engendrer 2 enfants → 8 enfants
  - mutation de 2 enfants parmi les 8

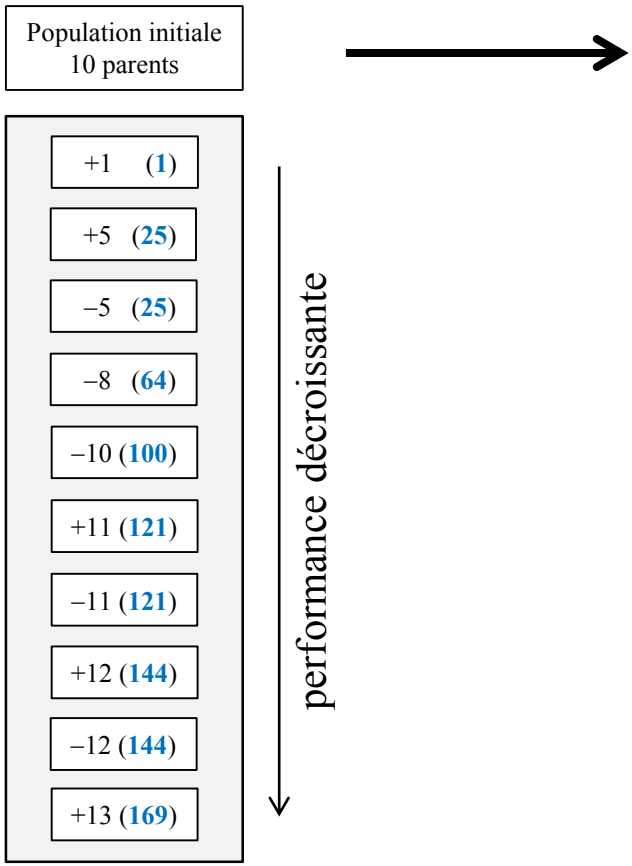
- Les 8 enfants $y_{j,\ j=1 \text{ à } 8}$ sont ensuite évalués → $f(y_j)$
  On dispose de 18 candidats pour la génération suivante :
  - 10 parents → x_i , $f(x_i)$ pour i = 1 à 10
  - 8 enfants → y_j , $f(x_j)$ pour j = 1 à 8

- L'étape de remplacement consiste à sélectionner les 10 meilleurs individus parmi les 18 pour constituer la génération suivante.

4.3.6 Algorithme évolutionnaire

Exemple

Minimisation de $f(x) = x^2$



Population initiale (génération k).

La population initiale comporte 10 individus (parents).

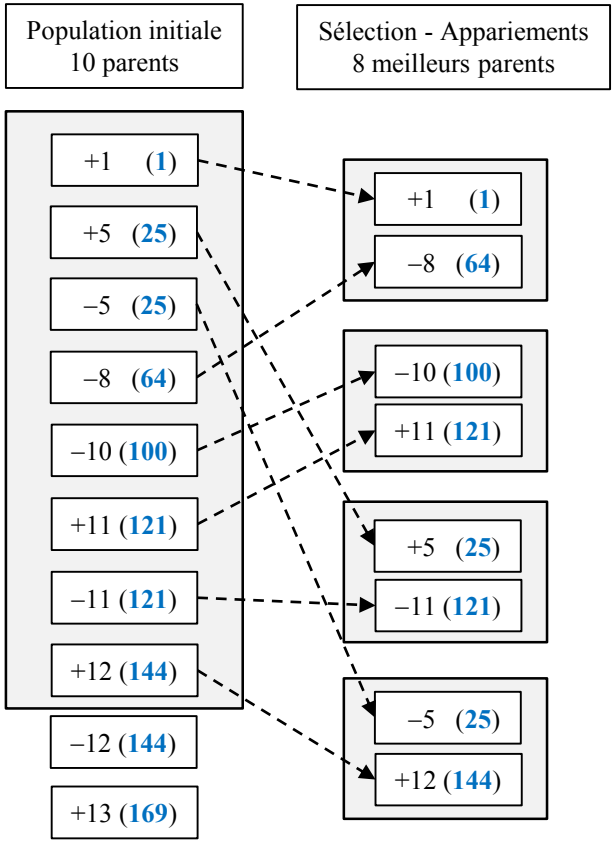
Chaque individu est représenté par
- une valeur $x_{i,\ i=1\ à\ 10}$ comprise entre −16 et +16
- son évaluation ou performance $\mathbf{f(x_i)}$

Les individus sont triés par performance décroissante (valeur de f croissante car f est à minimiser).

4.3.6 Algorithme évolutionnaire

Exemple

Minimisation de $f(x) = x^2$



Sélection pour la reproduction

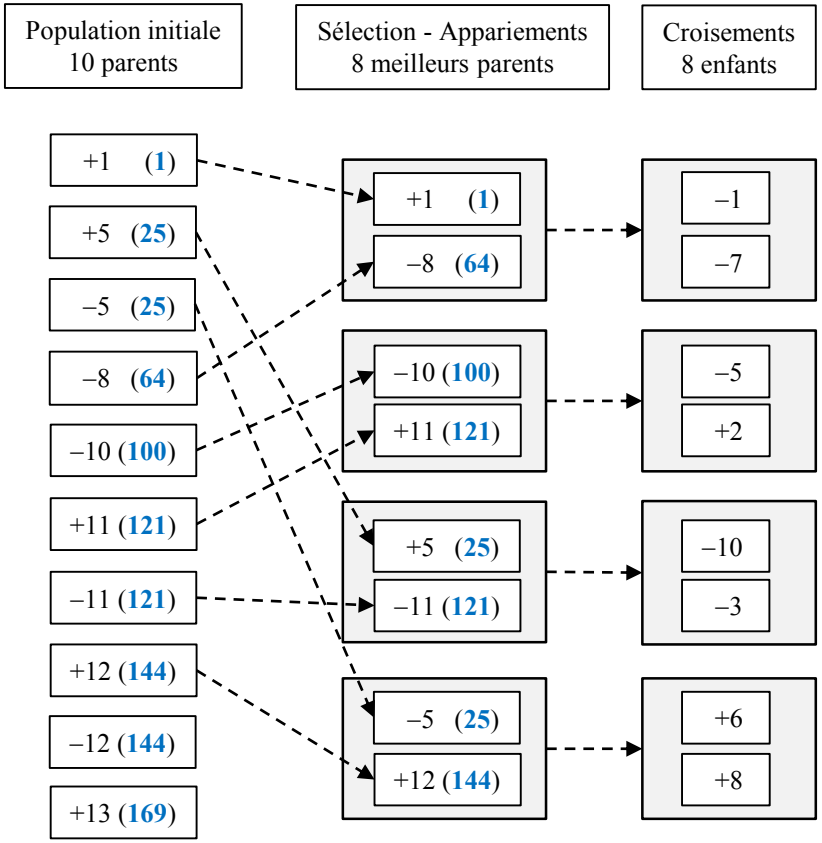
Les 8 meilleurs parents sont sélectionnés,
puis appariés aléatoirement 2 à 2.
→ sélection déterministe

Méthodes de sélection possibles
- Proportionnelle
  Un parent peut être sélectionné plusieurs fois proportionnellement à sa performance.
- Tournois
  Plusieurs parents sont tirés pour le tournoi.
  Le meilleur parent du tournoi est sélectionné.

4.3.6 Algorithme évolutionnaire

Exemple

Minimisation de $f(x) = x^2$



Croisement

2 parents engendrent 2 enfants par croisement.
Pour chaque enfant, la valeur de x est tirée aléatoirement entre celles des parents.

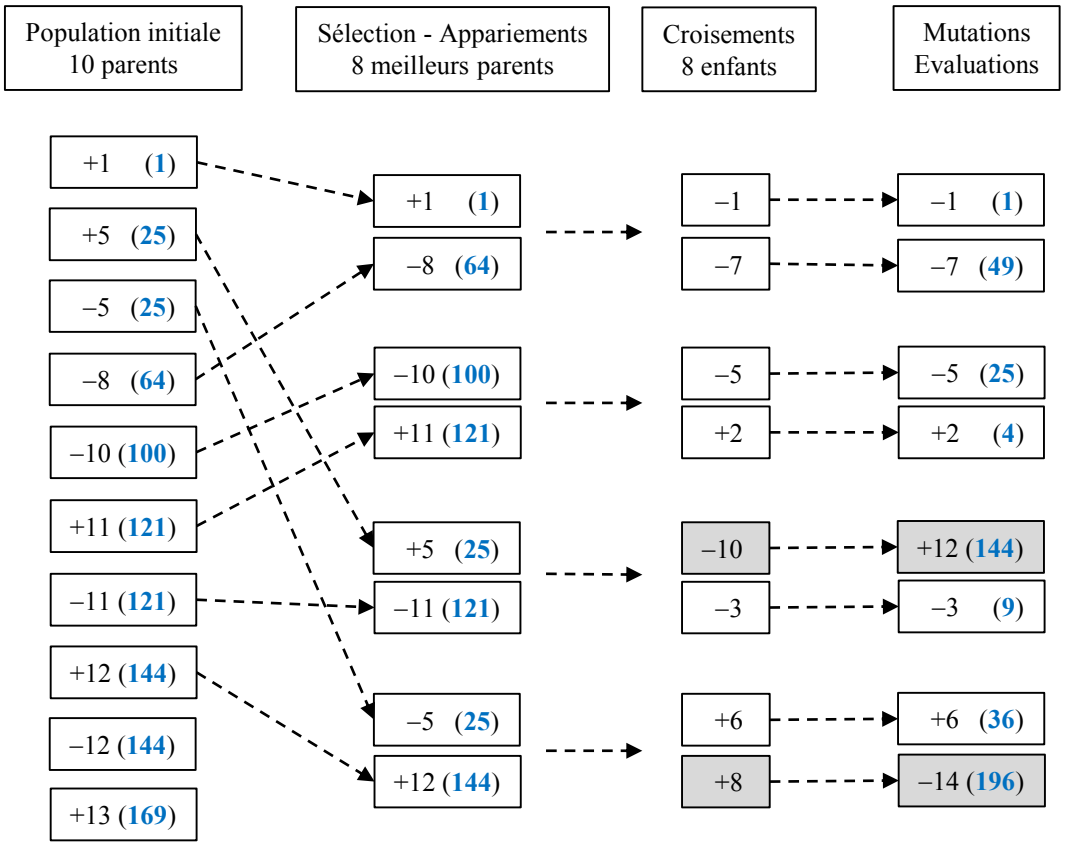
L'opérateur de croisement doit être adapté au problème :

- variables entières ou réelles
- fonction continue , minima locaux
- proximité parents / enfants

4.3.6 Algorithme évolutionnaire

Exemple

Minimisation de $f(x) = x^2$



Mutation

2 enfants sont modifiés aléatoirement.
Les 8 enfants sont ensuite évalués.

L'opérateur de mutation peut être uniforme ou gaussien.

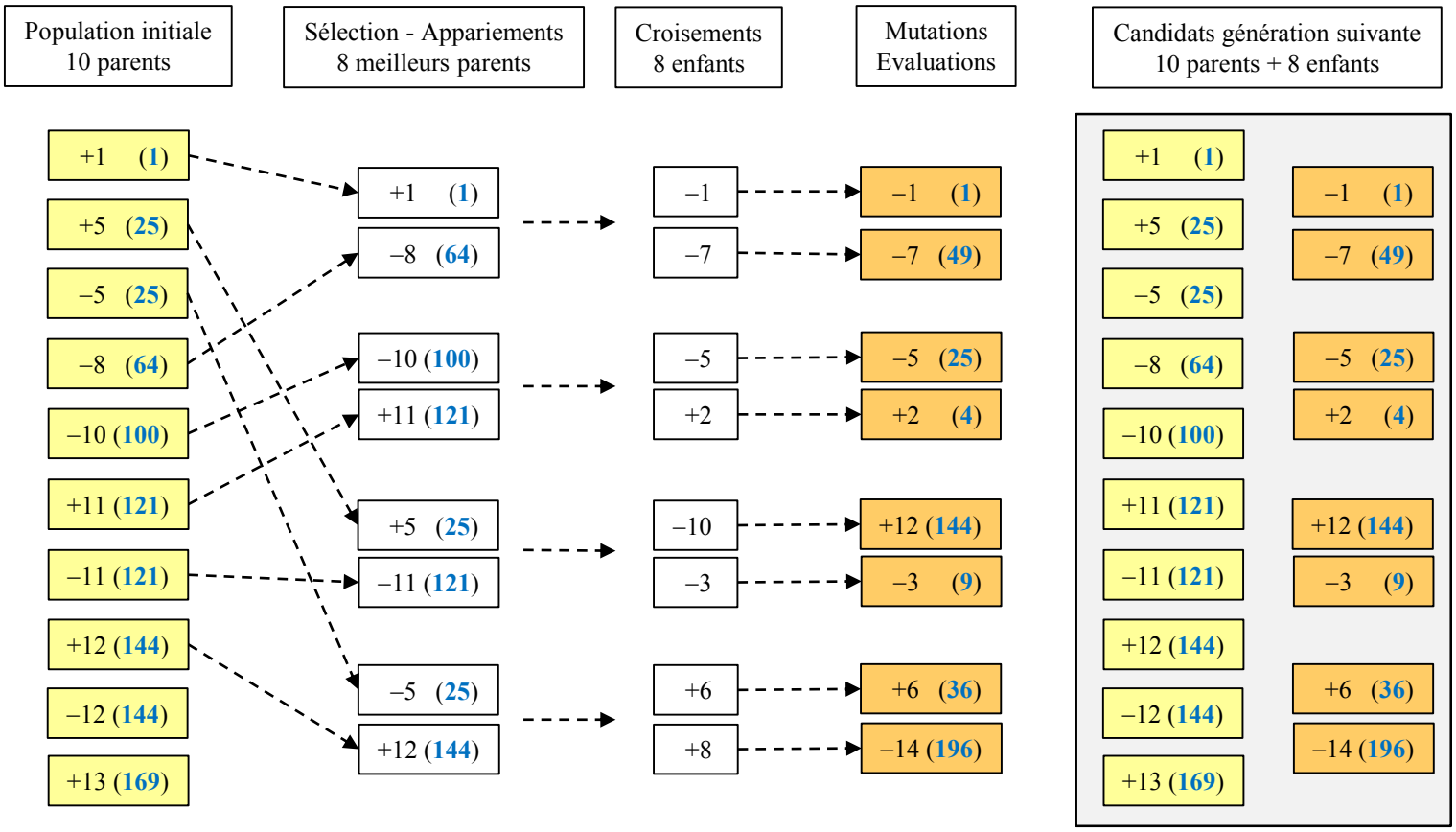
Pour une mutation gaussienne, l'écart-type σ évolue selon le taux τ de mutations favorables.

Règle des 1/5
- $\tau > 1/5$ → augmenter σ
- $\tau < 1/5$ → diminuer σ

4.3.6 Algorithme évolutionnaire

Exemple

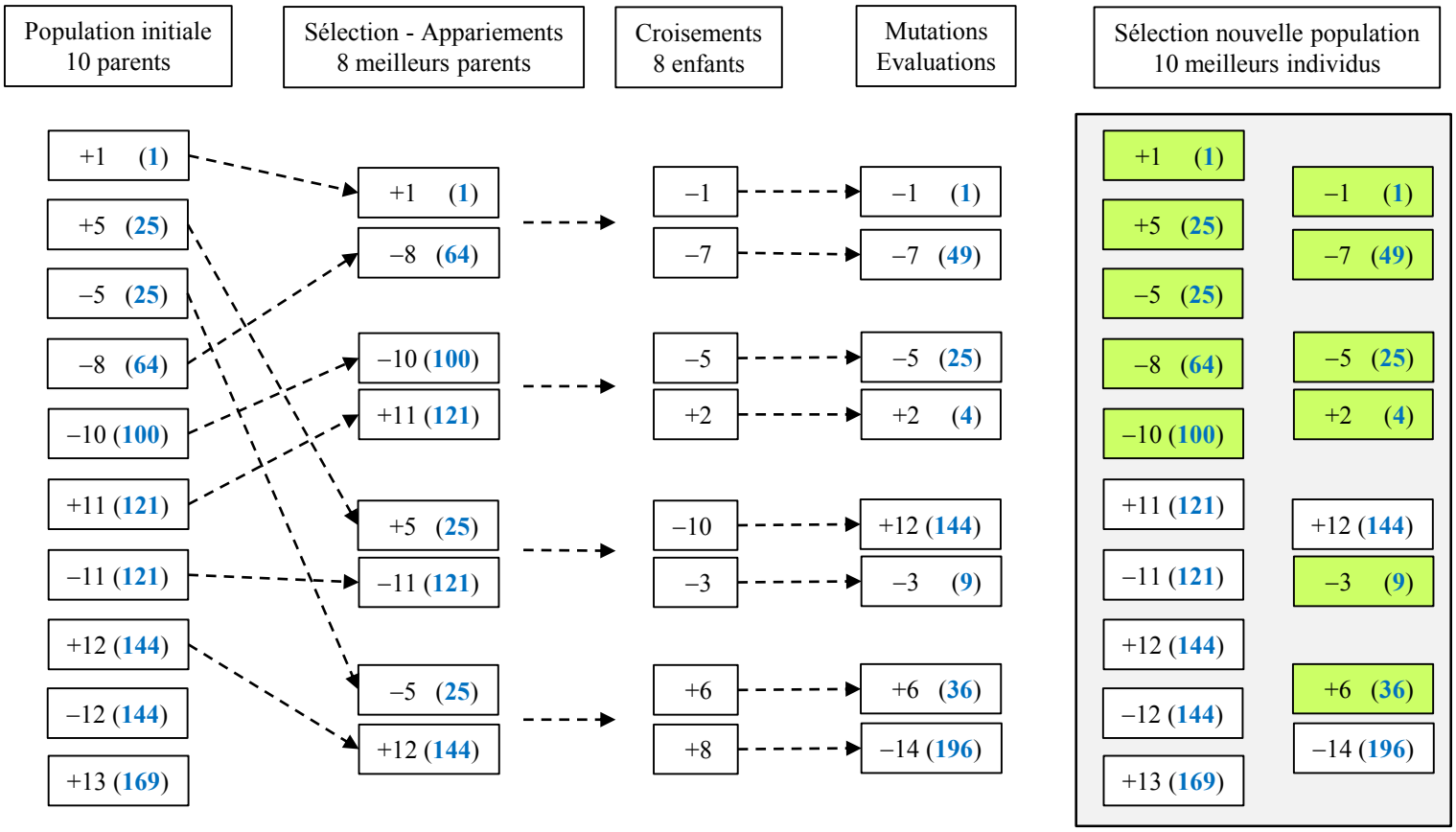
Minimisation de $f(x) = x^2$



4.3.6 Algorithme évolutionnaire

Exemple

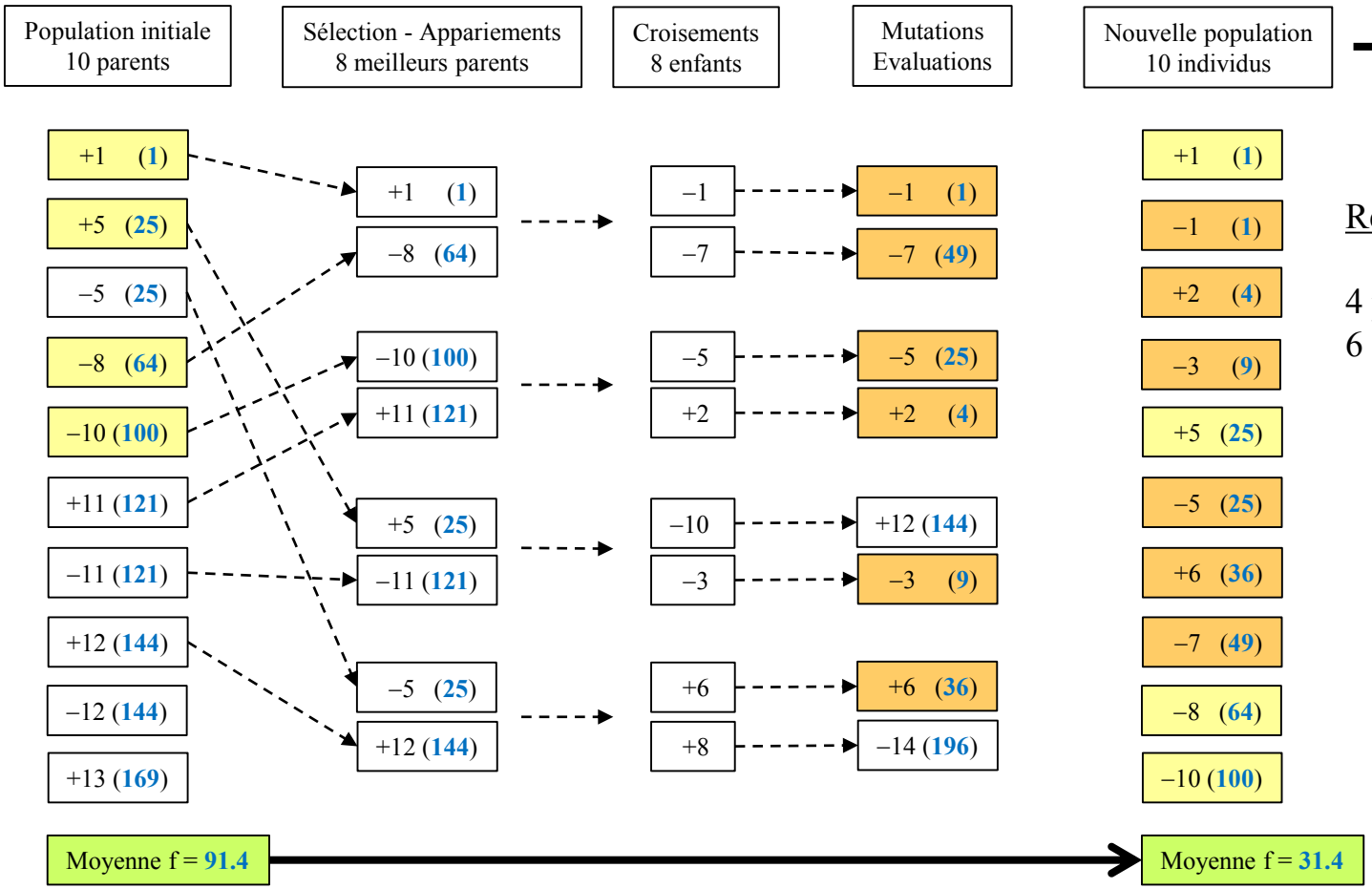
Minimisation de $f(x) = x^2$



4.3.6 Algorithme évolutionnaire

Exemple

Minimisation de $f(x) = x^2$



Remplacement

4 parents conservés
6 enfants sélectionnés