Université Sidi Mohamed Ben Abdellah

Ecole Nationale des Sciences Appliquées Fès ENSAF

Département Génie Industriel

Optimisation

Filières : Génie Industriel /Génie Mécatronique S2

Pr : Mhamed SAYYOURI

mhamed.sayyouri@usmba.ac.ma

Année universitaire 2024-2025

Informations générales sur le module

❑ Processus stochastique et Optimisation

- 2 éléments du module :
  - Processus stochastique (24H de cours et TD)
  - Optimisation (24H de cours, TD et TP)

Optimisation

❑ Introduction à l'optimisation

- Définition et importance de l'optimisation
- Applications de l'optimisation dans divers domaines
- Types de problèmes d'optimisation

❑ Optimisation Combinatoire

- Complexité des problèmes Combinatoires
- Méthodes de résolution exactes et approchées

❑ Optimisation métaheuristique

- Classifications des Métaheuristique
- Métaheuristiques à solution unique
- Métaheuristiques à population de solution

Introduction à l'optimisation

❑ Définition

- Euler dit : « Il n'y a rien au monde qui ne se réalise sans la volonté de minimiser ou maximiser quelque chose »

L'optimisation est une branche des mathématiques et de l'informatique, en tant que disciplines, cherchant :

- ➢ à modéliser,
- ➢ à analyser et
- ➢ à résoudre analytiquement ou numériquement

les problèmes réels :

- ➢ qui consistent à déterminer quelles sont la ou **les solutions (inconnues)** satisfaisant un **objectif quantitatif** tout en respectant **d'éventuelles contraintes**.
- ➢ Cela revient à résoudre un problème d'optimisation.

Introduction à l'optimisation

❑ Domaine d'utilisation

L'optimisation est omniprésente dans la vie quotidienne et dans divers domaines, où elle vise à maximiser ou minimiser des objectifs sous certaines contraintes:

- **Optimisation des itinéraires de transport :** Les applications de navigation comme Google Maps utilisent des algorithmes d'optimisation pour trouver l'itinéraire le plus rapide ou le plus court entre deux points, en tenant compte de la circulation, des travaux et d'autres contraintes.
- **Gestion financière et investissements :** Les investisseurs cherchent à maximiser leurs rendements tout en minimisant les risques en choisissant des portefeuilles d'investissements optimaux en fonction de leurs objectifs et de leur tolérance au risque.
- **Planification de la production dans les usines :** Les fabricants optimisent les processus pour maximiser la production tout en minimisant les coûts liés aux matériaux, à la main-d'œuvre et aux délais.
- **Optimisation énergétique :** Les systèmes de gestion de l'énergie dans les bâtiments cherchent à minimiser la consommation d'énergie tout en maintenant un confort optimal pour les occupants.

Introduction à l'optimisation

❑ Domaine d'utilisation

- **Planification des emplois du temps** : Les écoles, universités et entreprises optimisent les emplois du temps pour minimiser les conflits d'horaires, maximiser l'efficacité du travail et satisfaire les préférences des utilisateurs.
- **E-commerce et tarification dynamique** :Les plateformes d'achat en ligne optimisent leurs prix en temps réel en fonction de la demande, de la concurrence et du comportement des consommateurs pour maximiser les ventes et les profits.
- **Conception des réseaux de télécommunications** : Les opérateurs de réseaux cherchent à maximiser la couverture réseau et à minimiser les coûts d'installation et d'exploitation tout en respectant les contraintes d'infrastructure.
- **Optimisation des ressources humaines** : Les entreprises utilisent des techniques pour allouer le personnel de manière optimale, réduire les coûts, maximiser la productivité et améliorer la satisfaction des employés.

Introduction à l'optimisation

❑ Formulation

- Un problème d'optimisation est un modèle mathématique (formel) d'un problème réel.
- L'objectif est de minimiser (ou maximiser) une fonction objectif sous des contraintes :
  - $x \in \mathbb{R}^n$: ensemble des variables
  - $f : \mathbb{R}^n \to \mathbb{R}$: fonction coût (objectif)
  - $D \subseteq \mathbb{R}^n$: ensemble des contraintes $g(x)$
- Formulation :

$$PO = \begin{cases} \min \text{ (ou max) } f(x) \\ g(x) \leq 0 \\ x \in \mathbb{R}^n \end{cases}$$

On cherche à minimiser (ou maximiser) f sur D, c'est-à-dire trouver $x^* \in D$ tel que :

$$f(x^*) = \min(\text{ou max}) f(x)$$

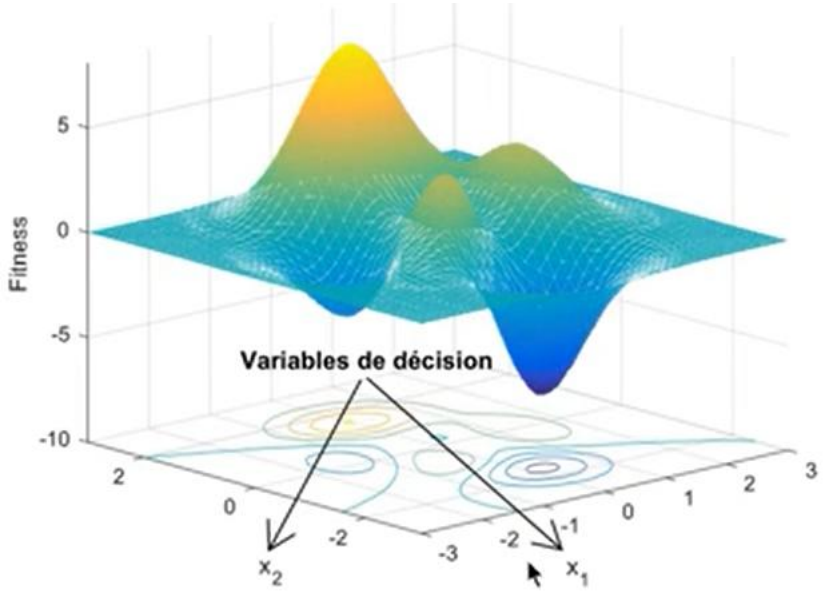
Avec :

$$f(x) \leq f(x^*), \quad \forall x \in D$$

Introduction à l'optimisation

❑ Quelques définitions de base en optimisation

1. **Variables de décision:** Les variables de décision représentent les informations inconnues dans un problème d'optimisation.



- **Vecteur X :** x1,x2,x3,...,xn
- **Domaines des variables**

$X = \mathbb{R}^n$ Pour les domaines **continus**

$X = \mathbb{Z}^n$ Pour les domaines **discrets**

2. Contraintes

Une contrainte est une condition d'un problème d'optimisation que les solutions doivent satisfaire.

$$h_j(x) = 0 \quad \text{pour } j = 1, 2, ..., u$$

$$g_k(x) \leq 0 \quad \text{pour } k = 1, 2, ..., v$$

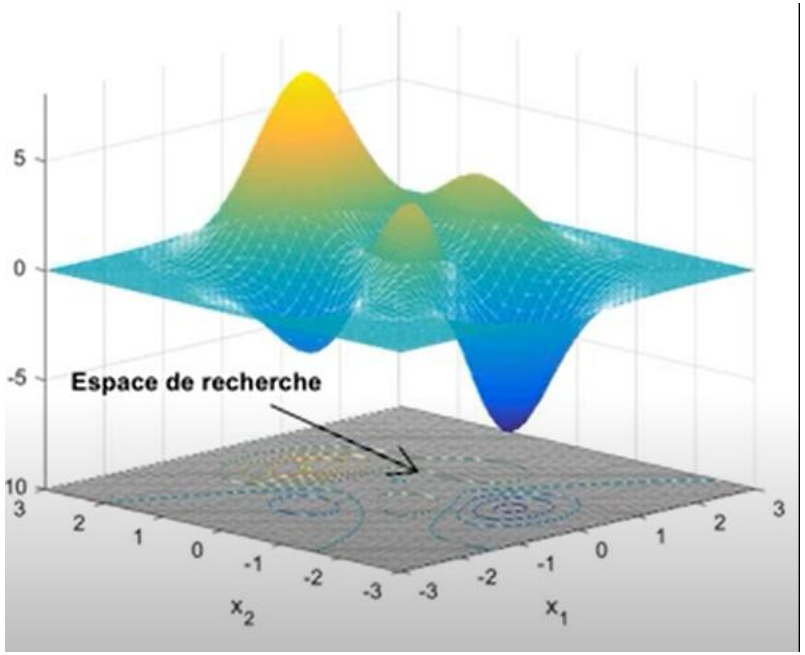
Introduction à l'optimisation

❑ Quelques définitions de base en optimisation

3. Espace de recherche

Un espace de recherche est l'ensemble de tous les points possibles d'un problème d'optimisation qui satisfait toutes les contraintes.

$$S := \{x \in \mathbb{R}^n | \text{Toutes les contraintes sont vérifiées}\}$$



4. Solution réalisable

- Une solution réalisable x′ est un membre d'un ensemble de solutions possibles.
- Elle se trouve dans l'ensemble qui satisfait toutes les contraintes

$$x' \in S$$

Introduction à l'optimisation

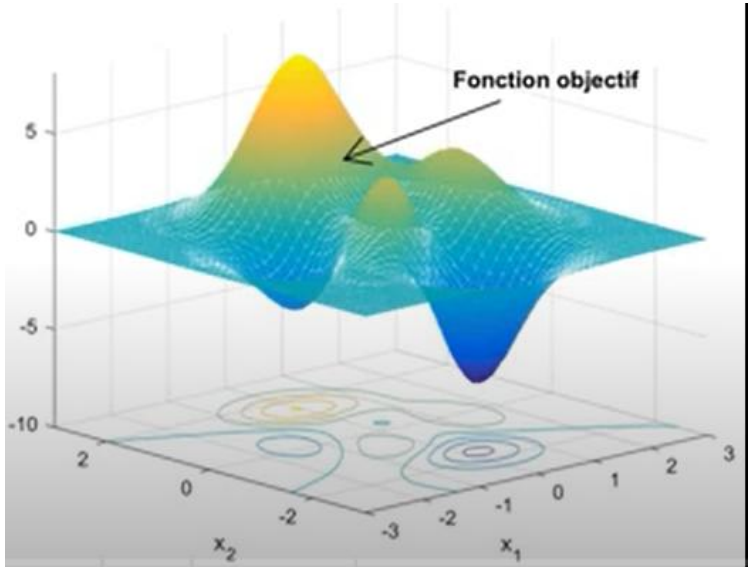
❑ Quelques définitions de base en optimisation

4. Fonction objectif

La fonction objectif est une équation mathématique écrite en fonction des variables de décision.

Permet de **mesurer la qualité d'une solution donnée** x′ pour un problème donné.

L'objectif est de minimiser (ou maximiser) cette fonction.



Soit $f : X = \mathbb{R}^n \to \mathbb{R}, x \to f(x)$

Un problème de minimisation peut être noté : $\min f(x)$

$$f(x_1, x_2) = 3(1 - x_1)^2 e^{-(x_1^2)-(x_2+1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) e^{-x_1^2-x_2^2} - \frac{1}{3} e^{-(x_1+1)^2-x_2^2}$$

Transformation d'un problème de maximisation en problème de minimisation :

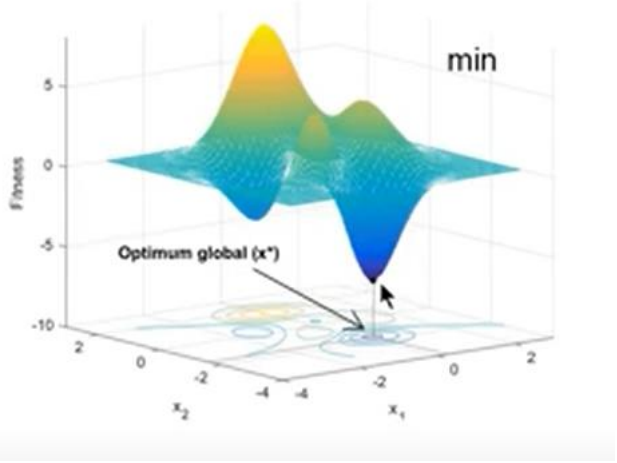
$$\max f(x) = -\min\{-f(x)\}$$

Introduction à l'optimisation

❑ Quelques définitions de base en optimisation

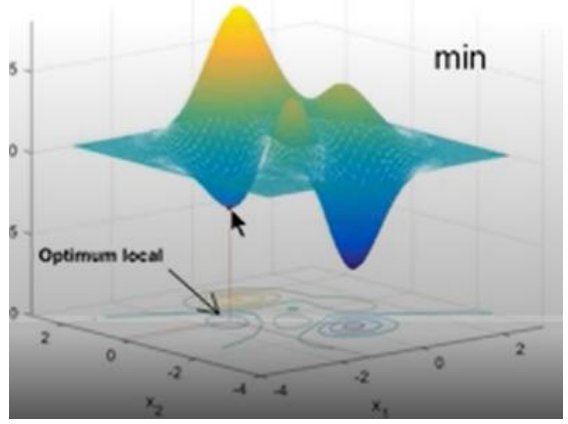
6. Optimum global (Solution optimale): Est une solution réalisable qui donne la valeur maximale (minimale) de la fonction objectif dans tout l'espace de recherche.

$$x^* = \arg\min\{f(x)\} := \{x | f(x) \leq f(x^*), \forall x' \in S\}$$



7. Optimum local: Est une solution réalisable qui donne la valeur maximale (minimale) de la fonction objectif dans le **voisinage** de cette solution.

$$f(x') \leq f(x), \forall x \in V(x')$$

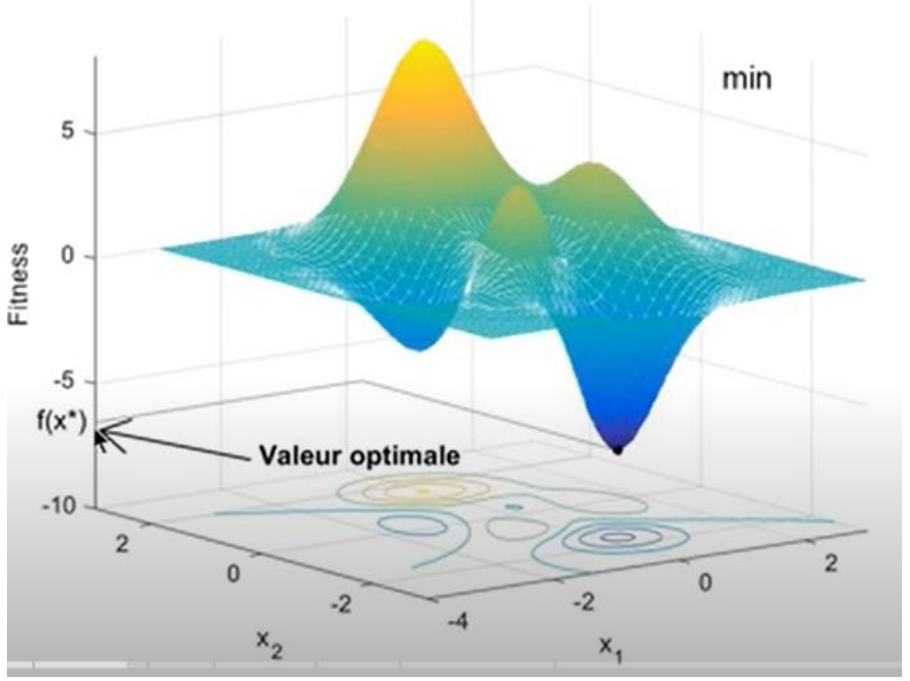


Introduction à l'optimisation

❑ Quelques définitions de base en optimisation

8. Valeur optimale : Est la valeur minimale (ou maximale) de la fonction objectif sur toute la région réalisable d'un problème d'optimisation.

$$f^* = f(x^*)$$



Introduction à l'optimisation

❑ Quelques définitions de base en optimisation

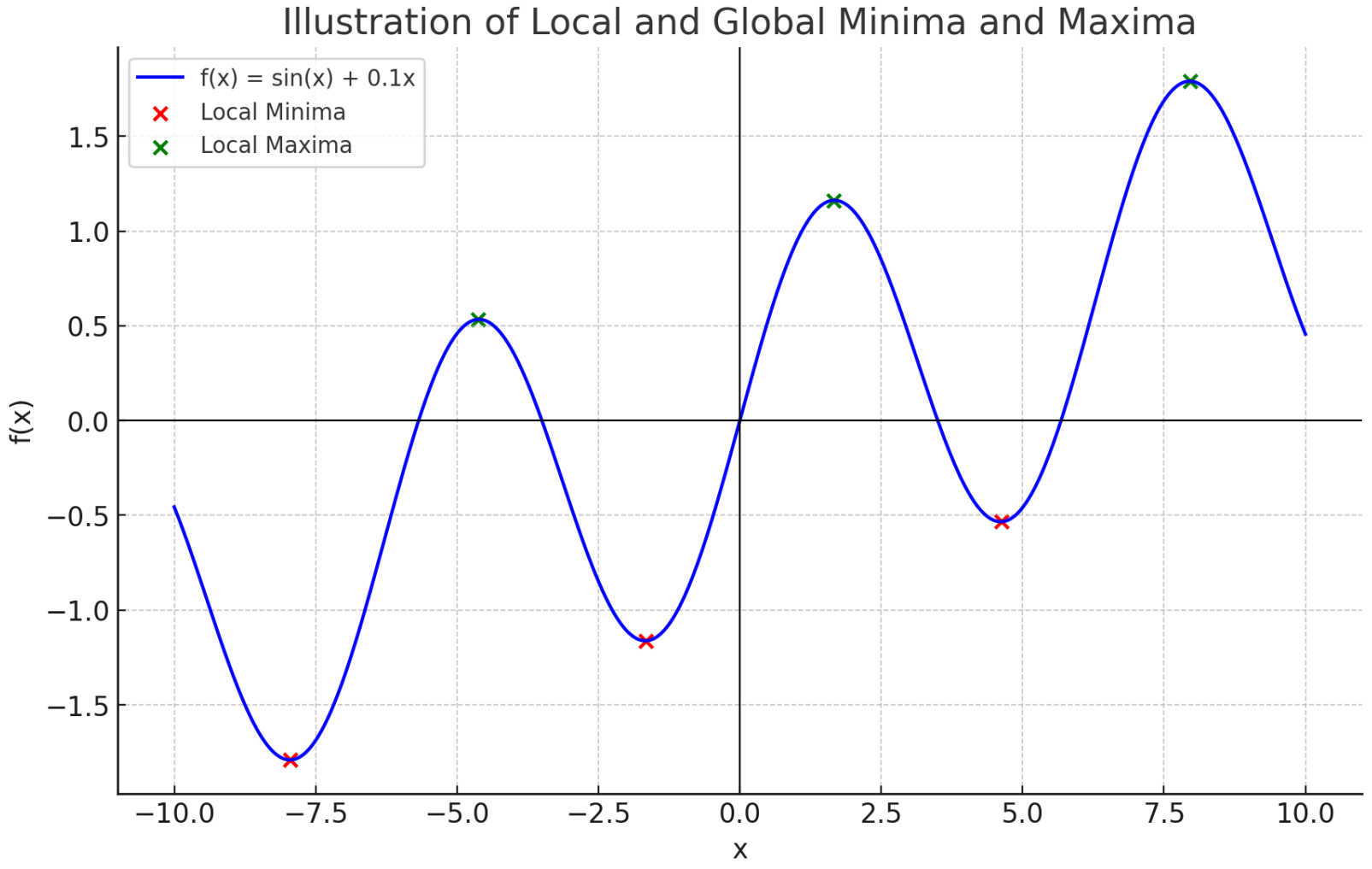
```
clc; clear;
% Définir les limites de l'espace de recherche
x1 = linspace(-2, 2, 100); % Valeurs pour x1
x2 = linspace(-2, 2, 100); % Valeurs pour x2
 % Créer une grille pour les coordonnées
[X1, X2] = meshgrid(x1, x2);
 % Calculer la fonction f(x1, x2)
F = 3 * (1 - X1).^2 .* exp(-(X1.^2) - (X2 + 1).^2) ...
    - 10 * (X1 / 5 - X1.^3 - X2.^5) .* exp(-X1.^2 - X2.^2) ...
    - (1/3) * exp(-(X1 + 1).^2 - X2.^2);
% Tracé 3D de la surface
figure;
mesh(X1, X2, F); % Surface 3D sans bordures
xlabel('x_1');
ylabel('x_2');
zlabel('f(x_1, x_2)');
title('Surface de la fonction f(x_1, x_2)');
colorbar;
grid on;
% Tracé des lignes de niveau (contour plot)
figure;
contour(X1, X2, F, 50); % Tracé des lignes de niveau avec 50 niveaux
xlabel('x_1');
ylabel('x_2');
title('Lignes de niveau de la fonction f(x_1, x_2)');
colorbar;
grid on;
```

Introduction à l'optimisation

❑ Illustration graphique

Voici une illustration de l'optimisation montrant des minima et maxima locaux pour une fonction représentative f(x)=sin(x)+0.1x.

Les points rouges indiquent les **minima locaux** et les points verts montrent les **maxima locaux**.



Introduction à l'optimisation

Exemple : Une entreprise fabrique des composants pour ordinateur.

Pour une quantité x, exprimée en milliers de composants, le coût total en milliers d'euros est :

$C(x) = 0{,}2x^2 + 24x + 20$ avec $x \in [0;30]$

La recette est alors égale à : $R(x)=30x$

Le bénéfice est la différence entre la recette et le coût total.

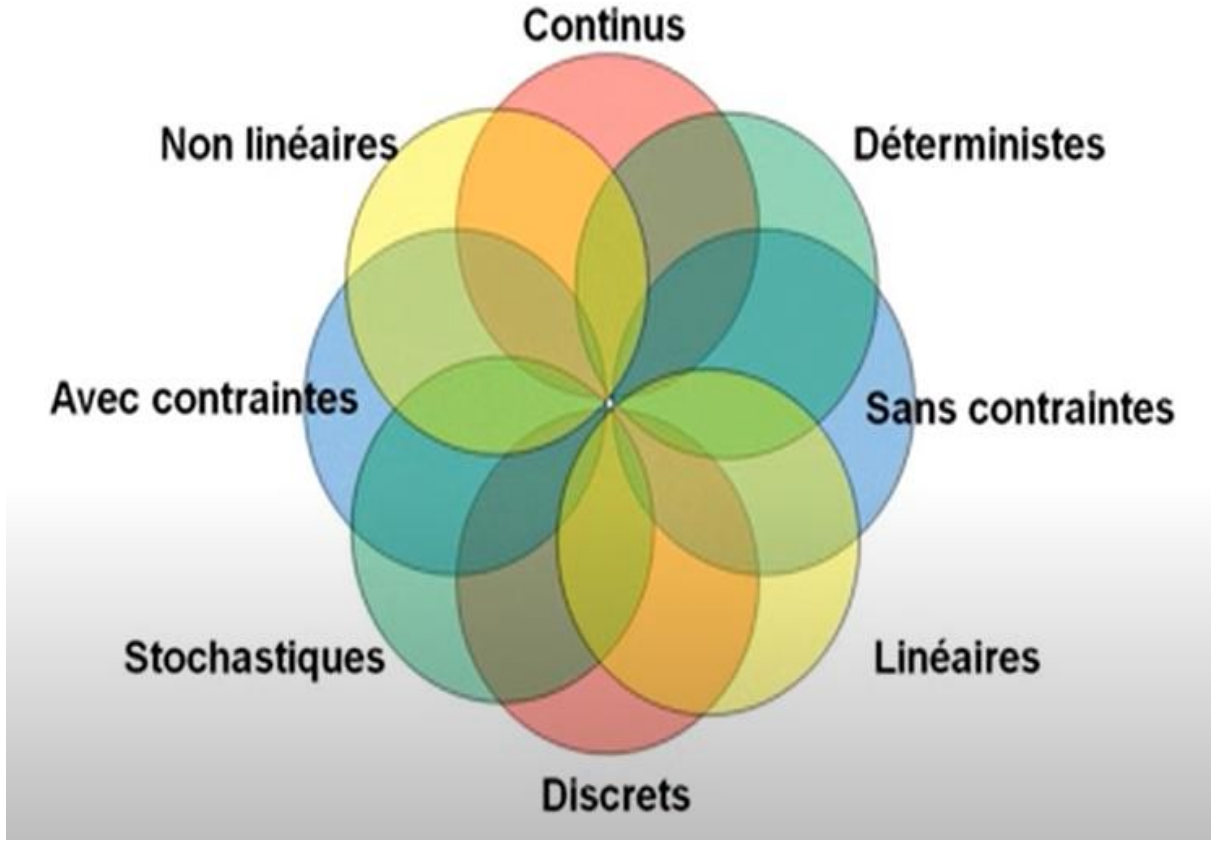
Objectif : Déterminer le bénéfice maximal et le nombre de composants correspondants à produire.

Introduction à l'optimisation

❑ Types des PO

Les problèmes d'optimisation peuvent être classés selon les critères suivants :

- Le type de contraintes
- La nature des variables de décision
- La structure physique du problème
- La nature des équations impliquées
- La nature déterministe du problème
- Le nombre d'objectifs
- ... etc



Introduction à l'optimisation

❑Types des PO

Caractéristiques	Propriétés	Classification
Nombre de variables	Une seule variable	Monovariable
	plus d'une variable	Multivariable
Type de variables	Réelles	Continue
	Entières	Discrète
	Réelles et entières	Mixte
	Entières avec permutation	Combinatoire
Type de fonction objectif	Linéaire en fonction des variables	Linéaire
	Quadratique en fonction des variables	Quadratique
	Non linéaire en fonction des variables	Non linéaire
Formulation du problème	Soumis à des limitations	Avec contraintes
	Pas de limitations	Sans contraintes

Introduction à l'optimisation

❑ Types des PO

Les problèmes d'optimisation peuvent être :

- Déterministes ou stochastiques
  - **Problèmes d'optimisation déterministes** : Les données du modèle sont connues avec précision et ne supposent aucune probabilité ou incertitude.
  - **Problèmes d'optimisation stochastiques** : Modèles qui ont un caractère aléatoire, pouvant inclure soit des **fonctions objectifs aléatoires** soit des **contraintes aléatoires**.

Exemple de PO déterministe : la **planification de la production** dans une usine. L'objectif est de minimiser les coûts de production tout en satisfaisant une demande fixe et connue à l'avance.

•**Fonction objectif** : Min $C(x)=cx+f$, où c est le coût unitaire, x est le nombre d'unités produites, et f est un coût fixe.

•**Contraintes** : La production doit satisfaire la demande D, telle que $x \geq D$, et respecter des limites de capacité.

Dans ce problème, toutes les données (coûts, demande, capacité) sont connues avec certitude.

Introduction à l'optimisation

❑ Types des PO

Exemple de PO stochastique : Prenons le même problème de planification de la production, mais avec une **demande incertaine** qui suit une distribution de probabilité (par exemple, une distribution normale avec une moyenne et un écart type donnés). L'usine doit minimiser ses coûts de production tout en satisfaisant cette demande incertaine.

- **Fonction objectif** : Minimiser l'espérance des coûts totaux, par exemple $E[C(x,D)]$, où D est une variable aléatoire représentant la demande.
- **Contraintes** : La production doit satisfaire la demande avec une certaine probabilité (par exemple, $P(x \geq D) \geq 0.95$.

Dans ce cas, l'optimisation tient compte de l'incertitude en modélisant les données comme des variables aléatoires et utilise des techniques spécifiques pour prendre des décisions robustes face à ces incertitudes.

Introduction à l'optimisation

❑Types des PO

- **Statiques** et **dynamiques** :
  - **Problèmes d'optimisation statiques** : La fonction objectif et les contraintes ne changent pas avec le temps.
  - **Problèmes d'optimisation dynamiques** : La fonction objectif ou les contraintes changent avec le temps.

Elle montre également une formulation générale des problèmes d'optimisation dynamique :

$f(x,t)$ est la fonction objectif.

$h_j(x,t) = 0$ pour $j = 1, 2, \ldots, u$ représente des contraintes d'égalité.

$g_k(x,t) \leq 0$ pour $k = 1, 2, \ldots, v$ représente des contraintes d'inégalité.

Introduction à l'optimisation

❑ Types des PO

Exemple : PO statique

•Fonction objectif : Minimiser C(x)=cx+s où ccc est le coût par unité produite, x est la quantité produite, et s est le coût de stockage.

•Contraintes : Capacité maximale de production, disponibilité des matières premières.

Exemple : PO dynamique

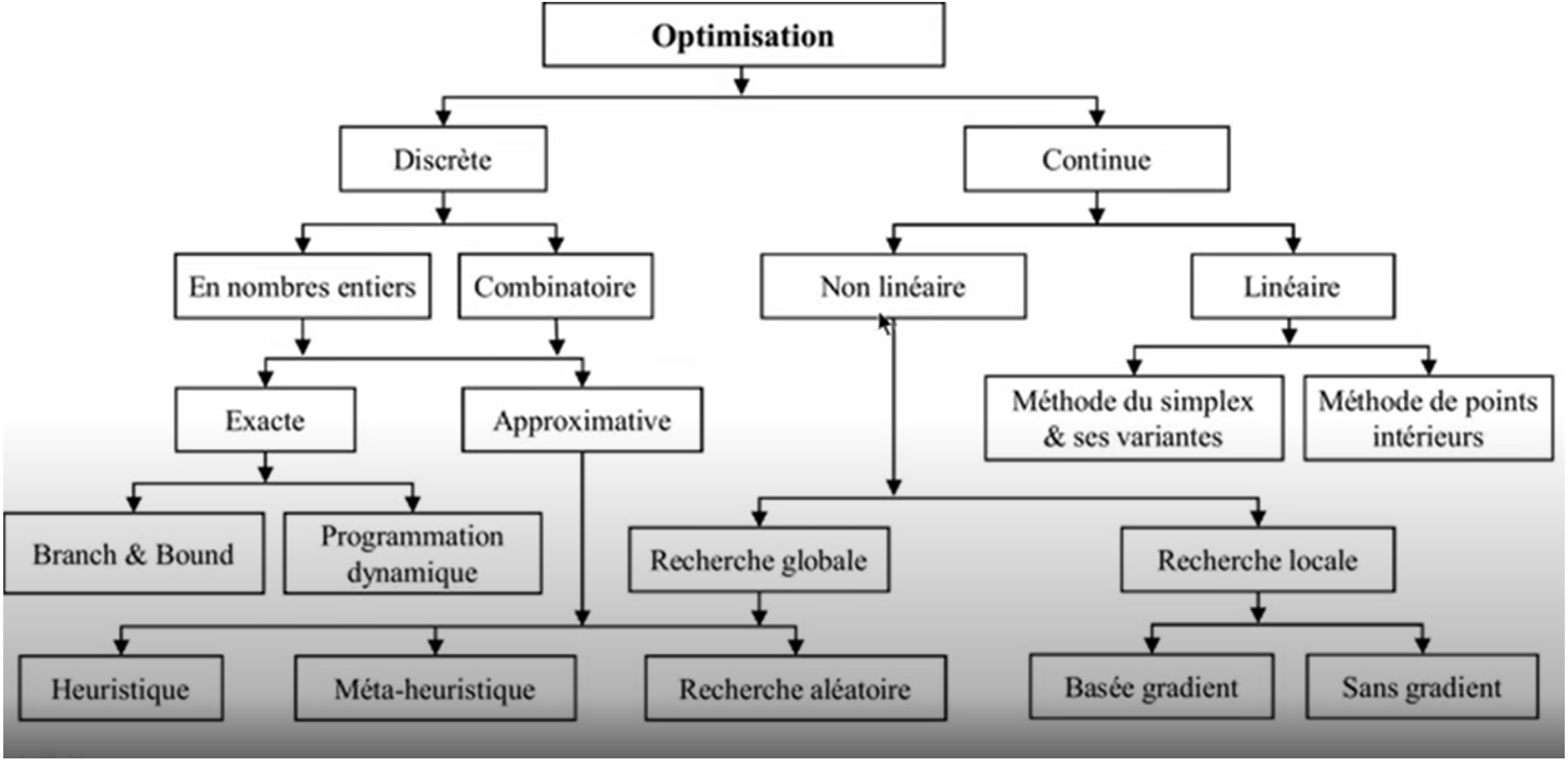
•Fonction objectif : Minimiser les coûts sur une période avec des fonctions dépendant du temps C(x,t)=c(t)x+s(t).

•Contraintes : Capacité qui peut changer selon les périodes de maintenance ou d'autres contraintes de production évolutives.

Cet exemple montre comment la prise en compte du facteur temps rend le problème dynamique, modifiant la façon de trouver une solution optimale.

Introduction à l'optimisation

❑ Classification des méthodes d'optimisation



Introduction à l'optimisation

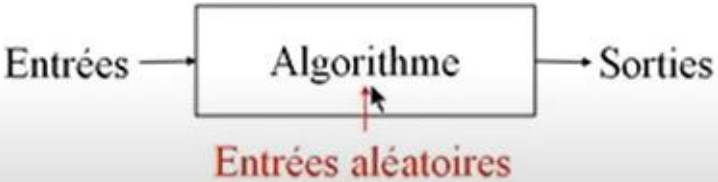
❑ Classification des méthodes d'optimisation

Parmi les méthodes d'optimisation on trouve aussi :

- **Méthodes déterministes** : Pour les mêmes entrées, ils produisent toujours la même sortie, en passant toujours par la même séquence d'états d'exécution.

Structure d'un algorithme déterministe

- **Méthodes stochastiques** : Ils utilisent le caractère aléatoire comme stratégie (par exemple, via des décisions probabilistes), ce qui peut entraîner des variations de sortie pour les mêmes entrées.



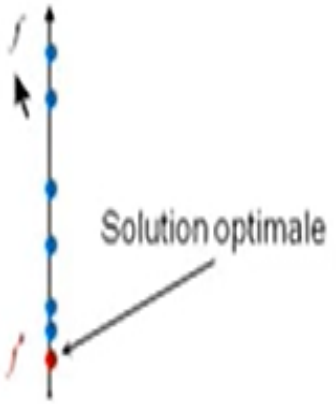
Structure d'un algorithme stochastique

Introduction à l'optimisation

❑ Classification des méthodes d'optimisation

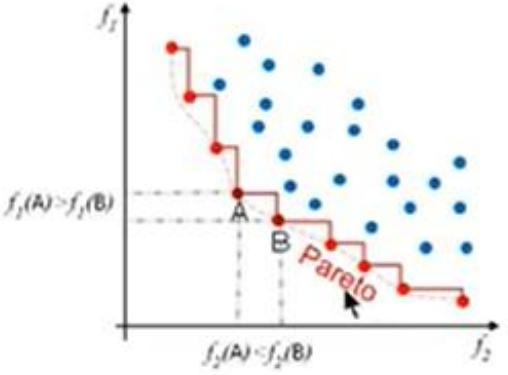
- **Méthodes à objectif unique** : Ces méthodes cherchent à trouver la meilleure solution correspondant à la valeur optimale d'une fonction objectif unique, notée minf(x). Il existe une solution optimale unique qui maximise ou minimise la fonction donnée.

$$\min f(x)$$



- **Objectif multiple** : Ces méthodes considèrent plusieurs objectifs, souvent conflictuels, notés minf1(x) et minf2(x), par exemple. Il n'existe généralement pas de solution optimale unique. Au lieu de cela, on recherche un ensemble de solutions dites **Pareto-optimales** où aucune amélioration de l'un des objectifs ne peut être obtenue sans dégrader un autre objectif.

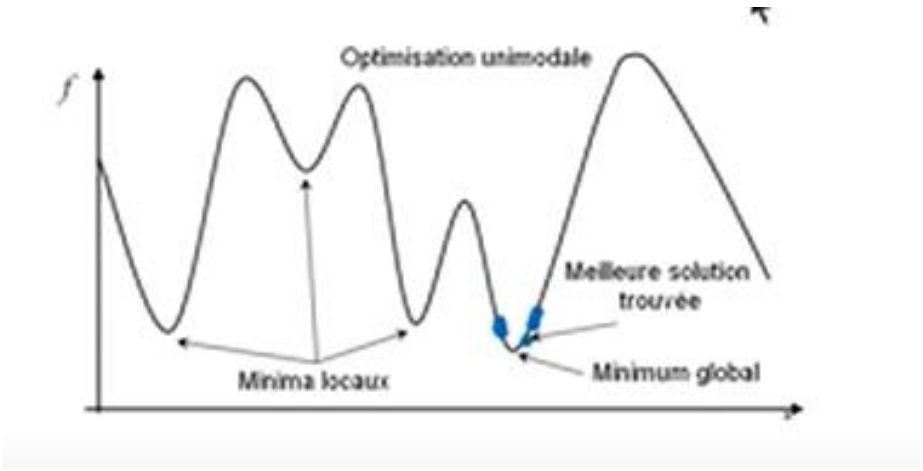
$$\begin{cases} \min f_1(x) \\ \min f_2(x) \end{cases}$$



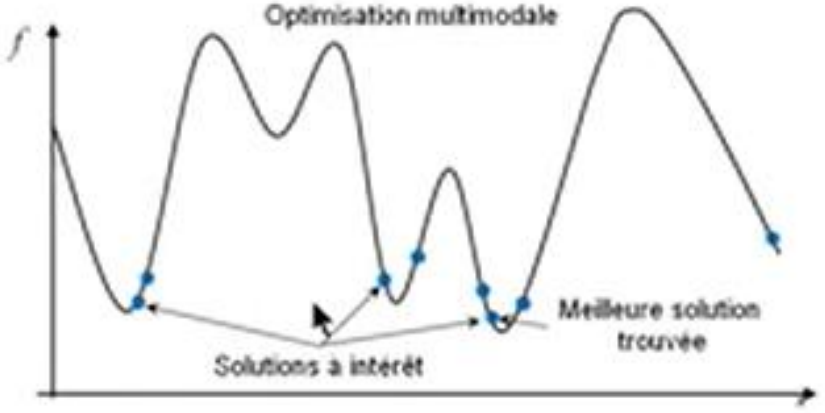
Introduction à l'optimisation

❑ Classification des méthodes d'optimisation

- **Optimisation unimodale** : Ces méthodes cherchent à trouver **une solution globalement optimale** pour un problème donné. Le graphique montre un cas où il existe un **minimum global**, et la méthode ignore les **minima locaux** pour se concentrer sur la meilleure solution possible.



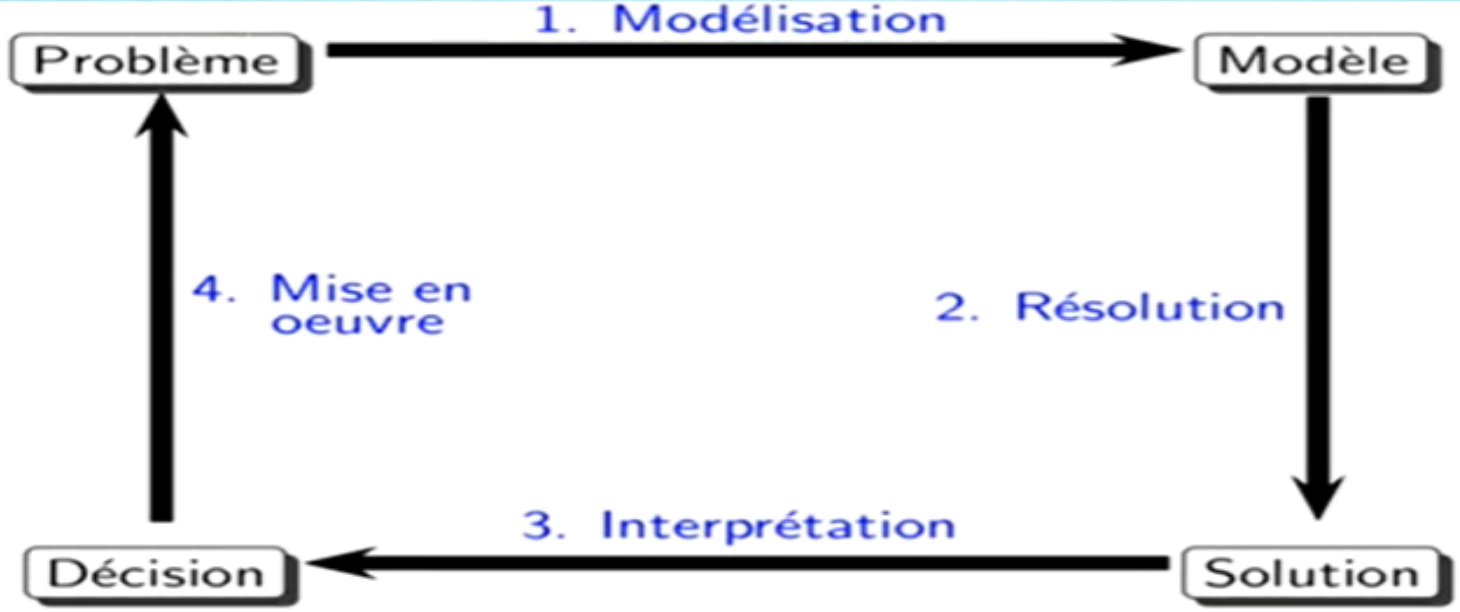
- **Optimisation multimodale** : Ces méthodes visent à identifier **un ensemble de bonnes solutions** plutôt qu'une seule solution optimale. Le graphique illustre plusieurs **solutions à intérêt**, ce qui permet d'explorer différents minima locaux tout en cherchant à maintenir un bon compromis parmi les solutions.



Introduction à l'optimisation

❑processus d'optimisation

Le **processus d'optimisation** passe par quatre étapes principales :



- **Modélisation** : Il s'agit de convertir un problème réel en un modèle mathématique en identifiant les variables, les contraintes et l'objectif.
- **Résolution** : Utilisation de méthodes analytiques ou numériques pour trouver la (ou les) solution(s) optimale(s) du modèle.
- **Interprétation** : Analyse des résultats obtenus pour évaluer leur pertinence et leur validité par rapport au problème initial.
- **Mise en œuvre** : Application de la solution optimale dans le contexte réel du problème.

Introduction à l'optimisation

❑ Résolution d'un problème d'optimisation

La **résolution d'un problème d'optimisation (PO)** se fait selon deux approches principales :

1.Résolution géométrique : Utilisation de la représentation graphique des contraintes pour visualiser et résoudre le problème d'optimisation.

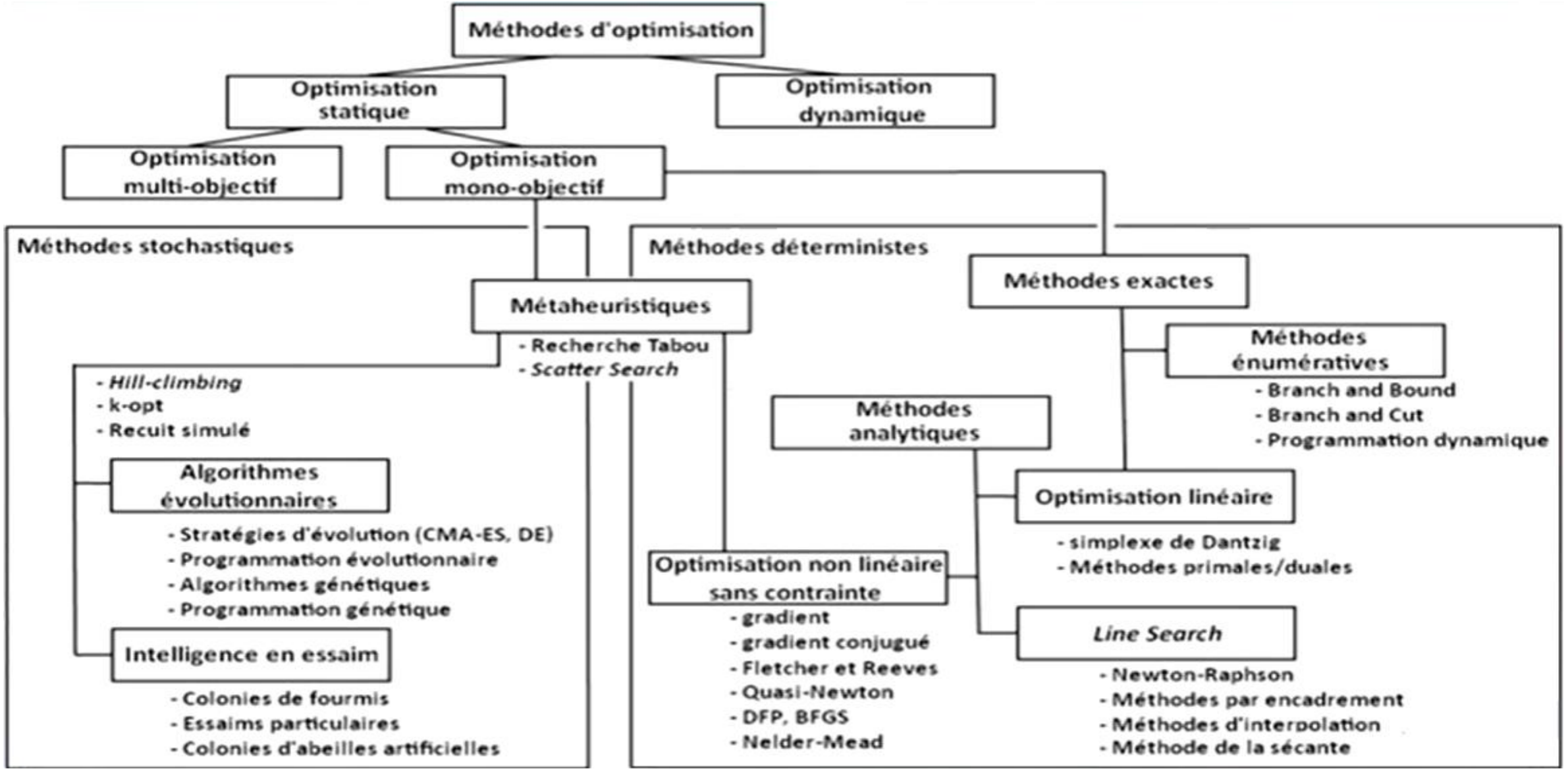
2.Résolution algorithmique : Utilisation d'une méthode d'optimisation algorithmique. Le choix de la méthode dépend de :

- La nature de la fonction objectif f, sa régularité (continuité, dérivabilité), ses propriétés spécifiques (parité, convexité) et la connaissance de voisinages de ses extrema.
- Les contraintes qui caractérisent l'ensemble D des points admissibles (réalisables).

Ces approches permettent de choisir la meilleure manière d'aborder et de résoudre un problème d'optimisation selon sa complexité et ses caractéristiques spécifiques

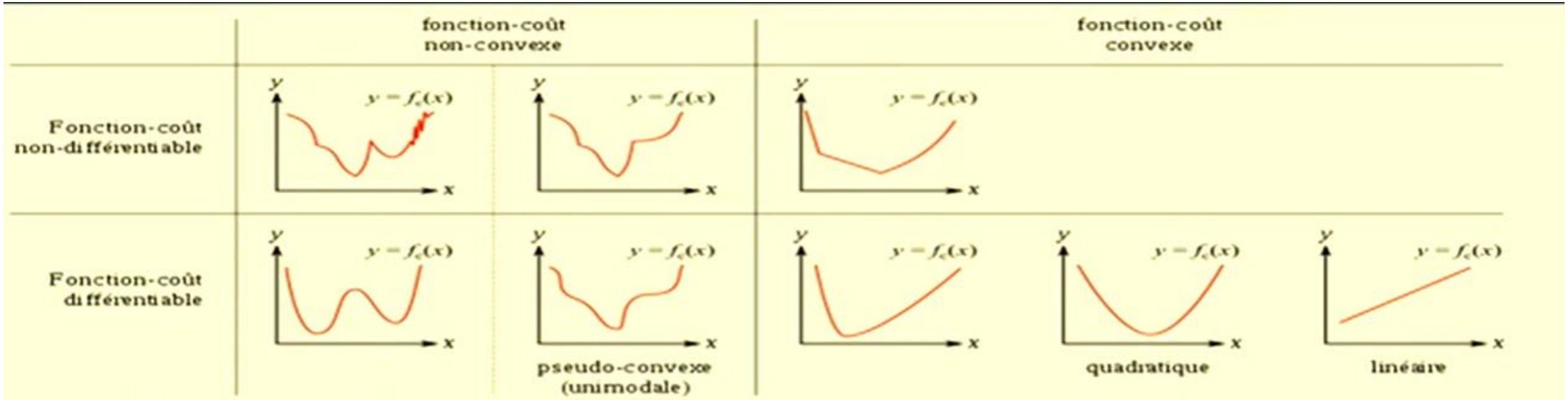
Introduction à l'optimisation

❑ Méthodes d'optimisation

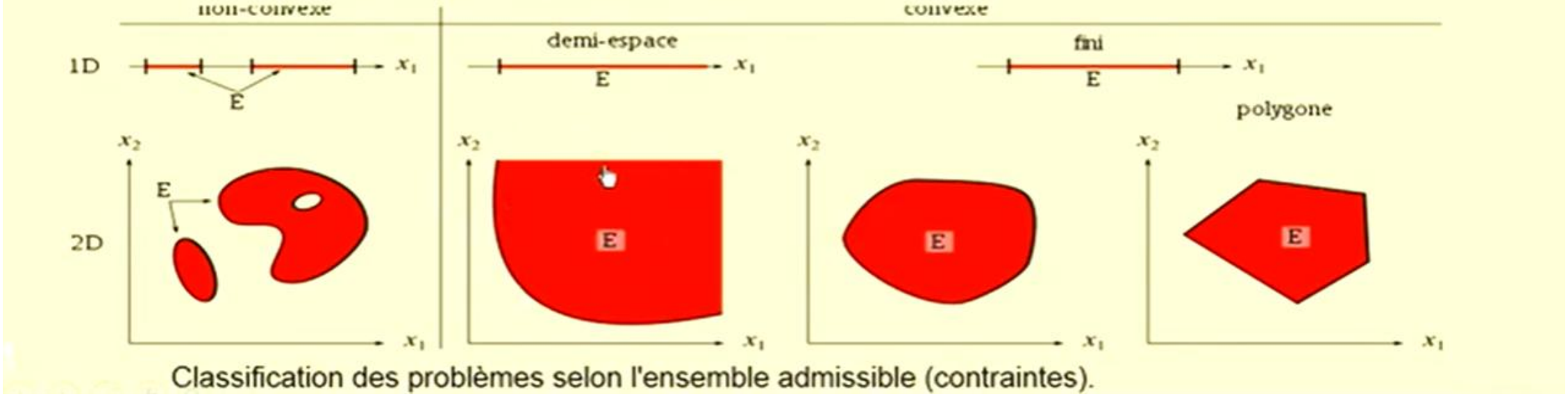


Introduction à l'optimisation

❑ Méthodes d'optimisation



Classification des problèmes selon la fonction-coût



Classification des problèmes selon l'ensemble admissible (contraintes).

Introduction à l'optimisation

❑Modélisation d'un PO

Exemple : Un fabricant souhaite vendre des unités d'un article à trois magasins (T1, T2 et T3). Il dispose de deux entrepôts (A et B) pour l'envoi. L'entrepôt A contient 5 unités et l'entrepôt B en contient 10. Les demandes des magasins T1, T2 et T3 sont respectivement de 8, 5 et 2 unités. Les coûts de transport d'une unité de chaque entrepôt vers chaque magasin sont indiqués dans le tableau.

	T1	T2	T3
A	1	2	4
B	3	2	1

Comment il faut transporter les unités pour être le plus économique passible ?

Les **étapes** proposées pour résoudre ce problème sont :

1.Lire et bien comprendre l'exercice.

2.Modéliser le problème sous forme d'un problème d'optimisation.

Introduction à l'optimisation

❑ Modélisation d'un PO

Pour la **modélisation** d'un problème d'optimisation, il faut bien lire l'exercice pour :

- comprendre la problématique.
- Identifier les données connues.
- Identifier les données inconnues.
- Déterminer le résultat demandé (la solution).
- Définir notre modèle comme un problème d'optimisation.
- Ressortir les concepts du modèle, notamment :
  - Les variables.
  - Leurs domaines.
  - Les contraintes.
  - L'objectif.

Introduction à l'optimisation

❑ Modélisation d'un PO

Problématique : Il s'agit d'un problème de transport.

Les données connues :

- La capacité de stockage des entrepôts.
- Les quantités demandées par les magasins.
- Le coût de transport entre les entrepôts et les magasins.

Les données inconnues :

- Quelles sont les quantités transportées des entrepôts vers les magasins ?

L'objectif : Trouver les quantités transportées des entrepôts vers les magasins qui minimisent le coût de transport tout en respectant la demande des magasins et la capacité des entrepôts (les contraintes).

Introduction à l'optimisation

❑Modélisation d'un PO

Étape 1 : Déterminer les variables de décision et les représenter de manière algébrique.

Dans ce cas :

Xi : numéro d'unités transportées de chaque entrepôt à chaque magasin

X1: nombre d'unités transportées de l'entrepôt A au magasin T1

X2: nombre d'unités transportées de l'entrepôt A au magasin T2

X3: nombre d'unités transportées de l'entrepôt A au magasin T3

X4: nombre d'unités transportées de l'entrepôt B au magasin T1

X5 : nombre d'unités transportées de l'entrepôt B au magasin T2

X6: nombre d'unités transportées de l'entrepôt B au magasin T3

Étape 2 : Déterminer la fonction objectif :

Minimiser $f(X) = X_1 + 2 \cdot X_2 + 4 \cdot X_3 + 3 \cdot X_4 + 2 \cdot X_5 + X_6$

Introduction à l'optimisation

❑Modélisation d'un PO

Étape 1 : Déterminer les contraintes et les formuler comme équation ou inéquations dépendants des variables de décision. Ces contraintes sont déduites de la disponibilité d'unités qu'il y a dans chaque entrepôt de même que la demande de chaque magasin :

- Disponibilité dans l'entrepôt A : X1+X2+X3=5
- Disponibilité dans l'entrepôt B : X4+X5+X6=10
- Demande du magasin T1 : X1+X4=8
- Demande du magasin T2 : X2+X5=5
- Demande du magasin T3 : X3+X6=2

❑Modélisation d'un PO

On obtient le problème d'optimisation suivant :

Minimiser $f(X) = X_1 + 2 \cdot X_2 + 4 \cdot X_3 + 3 \cdot X_4 + 2 \cdot X_5 + X_6$

sous les contraintes :

- $X_1 + X_2 + X_3 = 5$
- $X_4 + X_5 + X_6 = 10$
- $X_1 + X_4 = 8$
- $X_2 + X_5 = 5$
- $X_3 + X_6 = 2$
- $X_i \in \mathbb{N},\ i = 1, \ldots, 6$

La nature de ce PO : c'est un PO linéaire.

Sa représentation géométrique est un polygone.

❑Modélisation d'un PO

La solution optimale obtenue en utilisant la méthode du simplexe est la suivante :

Quantités transportées :

- X1=5 (5 unités de l'entrepôt A au magasin T1)
- X2=0 (0 unités de l'entrepôt A au magasin T2)
- X3=0 (0 unités de l'entrepôt A au magasin T3)
- X4=3 (3 unités de l'entrepôt B au magasin T1)
- X5=5 (5 unités de l'entrepôt B au magasin T2)
- X6=2 (2 unités de l'entrepôt B au magasin T3)

Coût total minimal : 26 unités.

Cette solution satisfait toutes les contraintes de capacité et de demande tout en minimisant le coût total.

Introduction à l'optimisation

❑ Modélisation d'un PO

```
% Résolution d'un problème d'optimisation linéaire avec des variables entières
clc; clear;
% Coefficients de la fonction objective (à minimiser)
f = [1, 2, 4, 3, 2, 1]; % Les coefficients sont négatifs pour maximiser la fonction
% Contraintes linéaires (Ax = b)
Aeq = [1, 1, 1, 0, 0, 0;  % x1 + x2 + x3 = 5
       0, 0, 0, 1, 1, 1;  % x4 + x5 + x6 = 10
       1, 0, 0, 1, 0, 0;  % x1 + x4 = 8
       0, 1, 0, 0, 1, 0;  % x2 + x5 = 5
       0, 0, 1, 0, 0, 1]; % x3 + x6 = 2
beq = [5; 10; 8; 5; 2]; % Valeurs des contraintes
% Contraintes de bornes (0 <= xi <= inf)
lb = zeros(6, 1); % Toutes les variables >= 0
ub = [];          % Pas de borne supérieure explicite
% Définir les indices des variables entières
intcon = 1:6;
% Résolution du problème avec intlinprog
[x, fval] = intlinprog(f, intcon, [], [], Aeq, beq, lb, ub);

% Affichage des résultats
disp('Solution optimale (xi) :');
disp(x);
disp('Valeur optimale de f(X) :');
disp(fval); % Valeur optimale (inverser le signe car on maximise)
```

Introduction à l'optimisation

❑ Exercice 1 :

Un agriculteur voudrait cultiver deux sortes de légumes : des brocolis et des courgettes. Il pourrait pour cela planter toute sa terre si le besoin nécessite. Il utilise deux sortes d'engrais (A et B).

- Les rendements des brocolis et des courgettes sont de 4 kg/m² et 5 kg/m² respectivement.
- Ses stocks sont de 8 litres d'engrais A et 7 litres d'engrais B.
- Les besoins en ces matières sont de 2 L/m² d'engrais A et 1 L/m² d'engrais B pour les brocolis contre 1 L/m² d'engrais A et 2 L/m² d'engrais B pour les courgettes.

L'agriculteur veut produire le maximum de poids de légumes. Il vous demande de l'aider.

Introduction à l'optimisation

❑ Étapes pour résoudre un problème d'optimisation

1.Analyse et formulation du problème

- Identifier les différentes variables, leurs natures et leurs domaines d'étude
- Définir l'objectif du problème
- Définir d'éventuelles contraintes

2.Modélisation mathématique du problème

- Formuler une fonction mathématique, appelée fonction objectif, qui décrit le problème
- Écrire les contraintes sous forme mathématique

3.Choix d'une méthode de résolution du problème modélisé

- Choisir une méthode d'optimisation adéquate au type du problème
- Appliquer la méthode d'optimisation pour résoudre le problème modélisé

Introduction à l'optimisation

❑ Étapes pour résoudre un problème d'optimisation

1.Analyse et formulation du problème

1. Les variables de décisions seront :

- x1: Surface pour les carottes (m2) de type réel.
- x2 : Surface pour les courgettes (m2) de type réel.

2. Objectif du problème :

- L'objectif est de déterminer les valeurs de x1 et x2 qui maximisent le poids total de la production.
- Formuler une fonction f qui calcule le poids total.

3. Contraintes :

- Les stocks d'engrais A sont limités à 8 litres.
- Les stocks d'engrais B sont limités à 7 litres.
- Les surfaces x1 et x2 ont des valeurs positives.

Introduction à l'optimisation

❑ Étapes pour résoudre un problème d'optimisation

Modélisation du problème

$$\text{Max } (f) : 4x_1 + 5x_2$$

sous les contraintes :

$$2x_1 + x_2 \leq 8$$

$$x_1 + 2x_2 \leq 7$$

$$x_1 \geq 0, \quad x_2 \geq 0$$

Introduction à l'optimisation

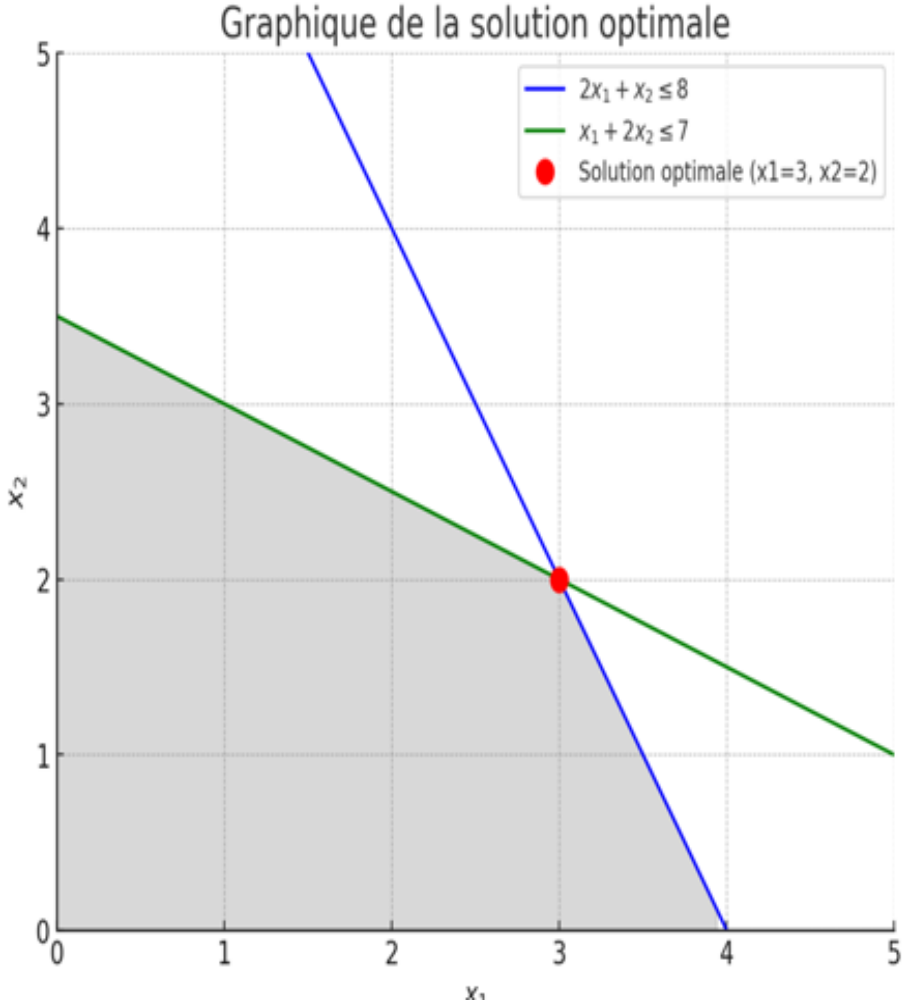
❑ Résolution géométrique :

Utiliser la méthode de résolution géométrique de la programmation linéaire.

- **Notion de point extrême**

Proposition: S'il en existe, il y a toujours une solution optimale sur un sommet (point extrême) de la région réalisable.

Corollaire: Pour trouver l'optimum, il "suffit" d'examiner les points extrêmes de la région réalisable



❑ Résolution géométrique :

Définition

Un polyèdre convexe est l'ensemble des solutions d'un système fini d'inégalités linéaires.

L'ensemble des solutions admissibles d'un PL est donc un polyèdre convexe.

Rappel : S est convexe si $\forall \mathbf{x}, \mathbf{y} \in S, \forall \lambda \in [0,1], \lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in S.$

Définition

Un point x_0 d'un ensemble convexe S est un point extrême de S s'il n'existe pas deux points

$$\mathbf{x_1}, \mathbf{x_2} \in S \text{ t.q. } \mathbf{x} = \lambda \mathbf{x_1} + (1 - \lambda \mathbf{x_2})$$

Théorème

Si le polyèdre formé par l'ensemble des solutions d'un PL est borné, alors il existe au moins une solution optimale et l'une d'elles est obtenue sur un point extrême.

Introduction à l'optimisation

❑ Résolution géométrique :

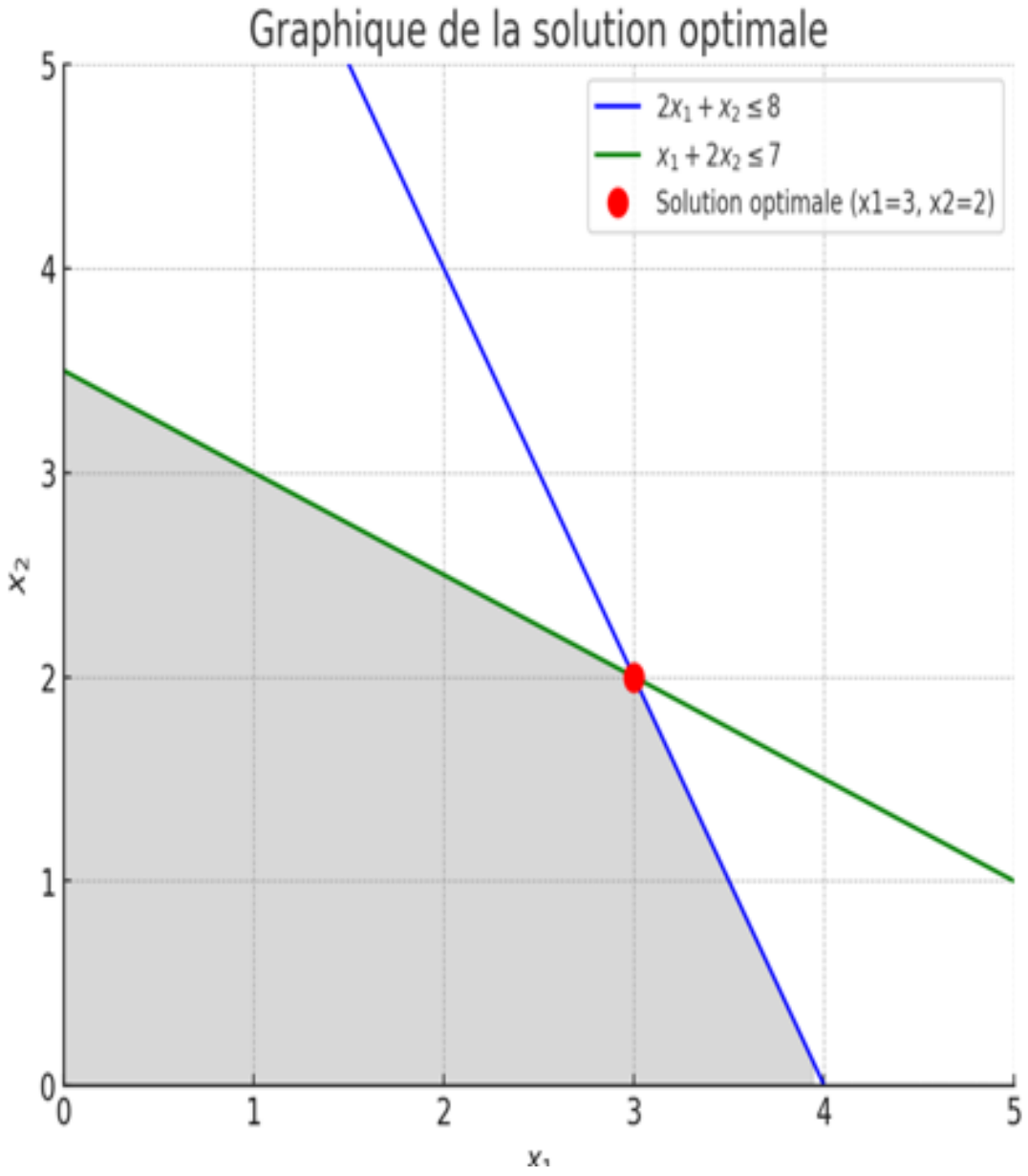
- Algorithme géométrique

1. Partir d'un point extrême x de la région réalisable

2. Déterminer une arête le long de laquelle l'objectif augmente.

S'il n'en existe pas, x est optimal, STOP

3. Se déplacer le long de l'arête jusqu'au point extrême y suivant. S'il n'existe pas, le problème est non borné, STOP Sinon, poser x ⟵ y et revenir en 2



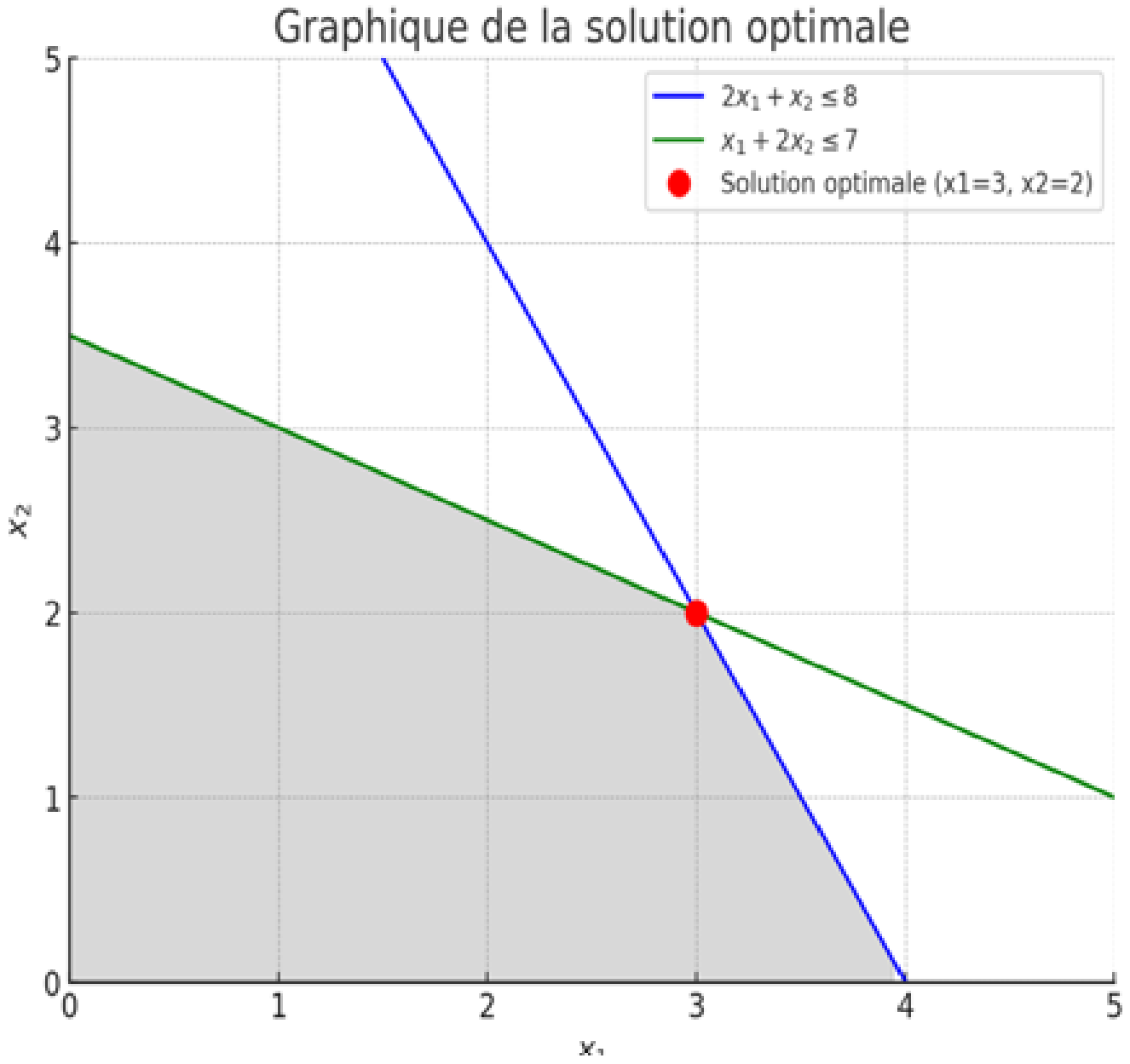
Introduction à l'optimisation

❑ Résolution géométrique :

La solution optimale est

•x1=3 m2, x2=2m2

•Le poids maximal qui peut être produit sera : *f(x1,x2)=22 kg*



Exercice 2:

Une entreprise met sur le marché un **shampoing** et un **revitalisant** pour les cheveux. Les produits sont vendus en bouteilles de 500 ml.

Une étude de marché a permis de recueillir les informations suivantes :

- À chaque mois, le nombre de bouteilles de shampoing vendues sera **supérieur ou égal** au nombre de bouteilles de revitalisant vendues.
- L'entreprise vendra au **maximum** 5000 bouteilles de ses nouveaux produits par mois.
- L'entreprise vendra au **moins** 1500 bouteilles de shampoing par mois.

L'équipe qui a mené l'étude de marché a proposé deux combinaisons de prix de vente :

- **3,00 $** par bouteille de shampoing et **3,00 $** par bouteille de revitalisant.
- **2,80 $** par bouteille de shampoing et **3,10 $** par bouteille de revitalisant.

Question : Quelle combinaison de prix l'entreprise doit-elle choisir pour maximiser ses revenus ?

Introduction à l'optimisation

Exercice 3:

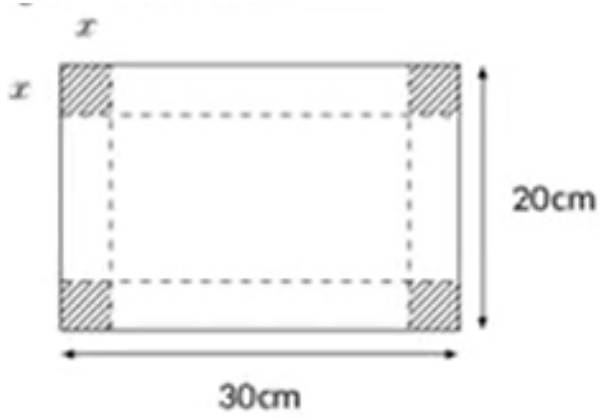
Pour réaliser des boîtes sans couvercle ayant la forme d'un parallélépipède rectangle, on dispose d'une feuille en carton de 20 cm par 30 cm. Pour réaliser cette boîte, on découpe aux quatre coins de la feuille quatre carrés identiques de côté x cm puis on plie le carton suivant les segments en pointillés sur la figure ci-dessous :

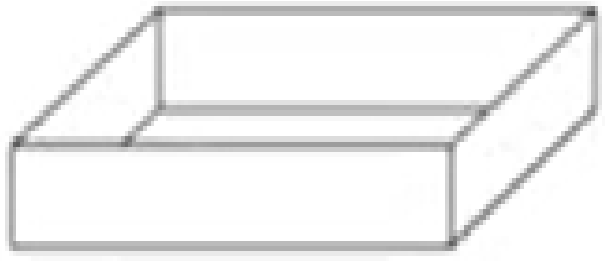
1.Dans quel intervalle I se situe la variable x ?

2.Montrer que le volume V(x) de la boîte obtenu après découpage et pliage est :

V(x)= 4x^3 - 100x^2 + 600x.

1.Étudier les variations de la fonction V sur l'intervalle I et en déduire les dimensions de la boîte qui possède un volume maximal.





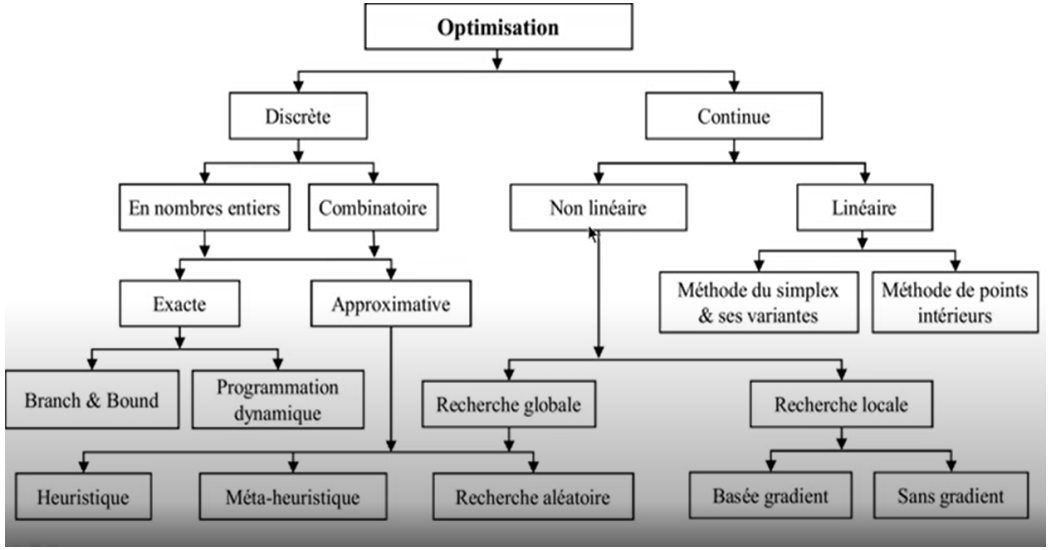
Chapitre 1: Optimisation Combinatoire

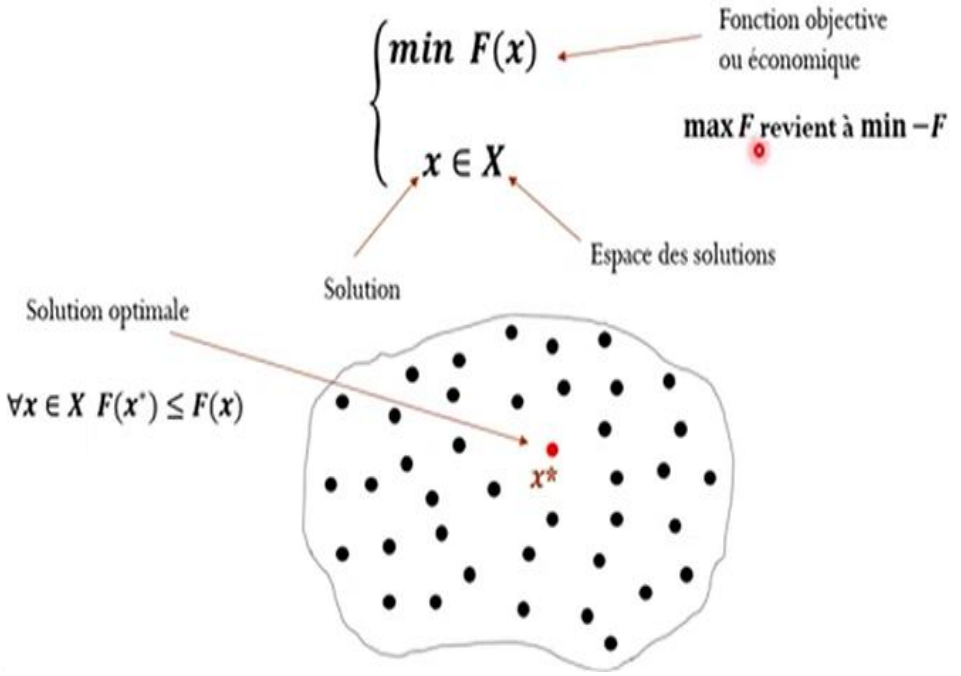
Optimisation Combinatoire

❑ Définition

Un problème d'optimisation combinatoire (POC) peut être défini par :

- ✓ Vecteur de variables x = (x1, x2, ..., xn),
- ✓ Domaine des variables D = (D1, D2, ..., Dn), où les (Di)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 = (x1, x2, , xn) ∈ D / x satisfait toutes les contraintes}, l'ensemble S est aussi appelé un espace de recherche.





❑ Modélisation mathématique

$$\begin{cases} \text{Min ou Max } Z = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \quad (\leq, =, \geq) \quad b_j \qquad \forall i \\ x_j \in \{0, 1\} \end{cases}$$

PLNE 0/1

$$\begin{cases} \text{Min ou Max } Z = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \quad (\leq, =, \geq) \quad b_j \qquad \forall i \\ l_j \leq x_j \leq u_j \\ x_j \quad \text{entier} \end{cases}$$

PLNE

Programme Linéaire en Nombres Entiers

Optimisation Combinatoire

❑ Problèmes d'Optimisation Combinatoires (POC) classiques

Problème	Applications
Problème du sac à dos	- Problèmes de chargement (avions, bateaux, …)
	- Problème d'investissement
Problème du Voyageur de Commerce	- Problèmes de transport et de logistique
	- Sous-problèmes dans les réseaux
Problème d'implantation d'équipements	- Problèmes de transport et de logistique
	- Installation des réseaux filaires
Problèmes de couverture, partitionnement et d'emballage	- Installation des relais GSM et réseaux sans fils
	- Organisation des votes, gestion des vols aériens
Problème de découpe	- Optimisation de la coupe dans les industries du papier, verre, métal, …

❑ Résolution de (POC)

- La résolution de (POC) consiste à trouver **la meilleure solution,** définie comme la solution globalement optimale ou un optimum global.
- La résolution des problèmes combinatoires **est assez délicate** puisque le nombre fini de solutions réalisables croît généralement avec la taille du problème, ainsi que sa complexité. Cela a poussé les chercheurs à développer de nombreuses méthodes de résolution en recherche opérationnelle (RO) et en intelligence artificielle (IA).

- **Les** méthodes de résolution peuvent être classées en deux catégories :
  - **Méthodes exactes** (*Branch and bound*, programmation dynamique, ...)
    - Résolution complète du problème
    - Obtention d'une solution optimale
    - Complexité exponentielle sur les problèmes difficiles

❑ Résolution de (POC)

➢ Méthodes approchées (*Heuristiques et métaheuristiques*)

- Résolution incomplète du problème
- Obtention d'une solution approchée
- Le critère d'arrêt est généralement un nombre d'itérations fixé au préalable

➢ Comparaison entre méthodes exactes et approchées :

Aspect	Méthodes exactes	Méthodes approchées
Qualité de la solution	Garantie optimale	Solution approchée, non garantie optimale
Temps de résolution	Long, voire exponentiel pour des problèmes complexes	Rapide, même pour des problèmes de grande taille
Applications	Problèmes de petite taille ou nécessitant une précision absolue	Problèmes NP-complets ou de grande taille
Complexité	Exponentielle (dans les pires cas)	Polynomial ou quasi-polynomial
Exemples	Simplexe, Branch and Bound, Programmation dynamique	Algorithmes génétiques, recuit simulé, heuristiques

❑ Notions sur la complexité

Généralement, le temps d'exécution est le facteur majeur qui détermine l'efficacité d'un algorithme, alors la complexité en temps d'un algorithme est le nombre d'instructions nécessaires (affectation, comparaison, opérations algébriques, lecture et écriture, etc.) que comprend cet algorithme pour une résolution d'un problème quelconque.

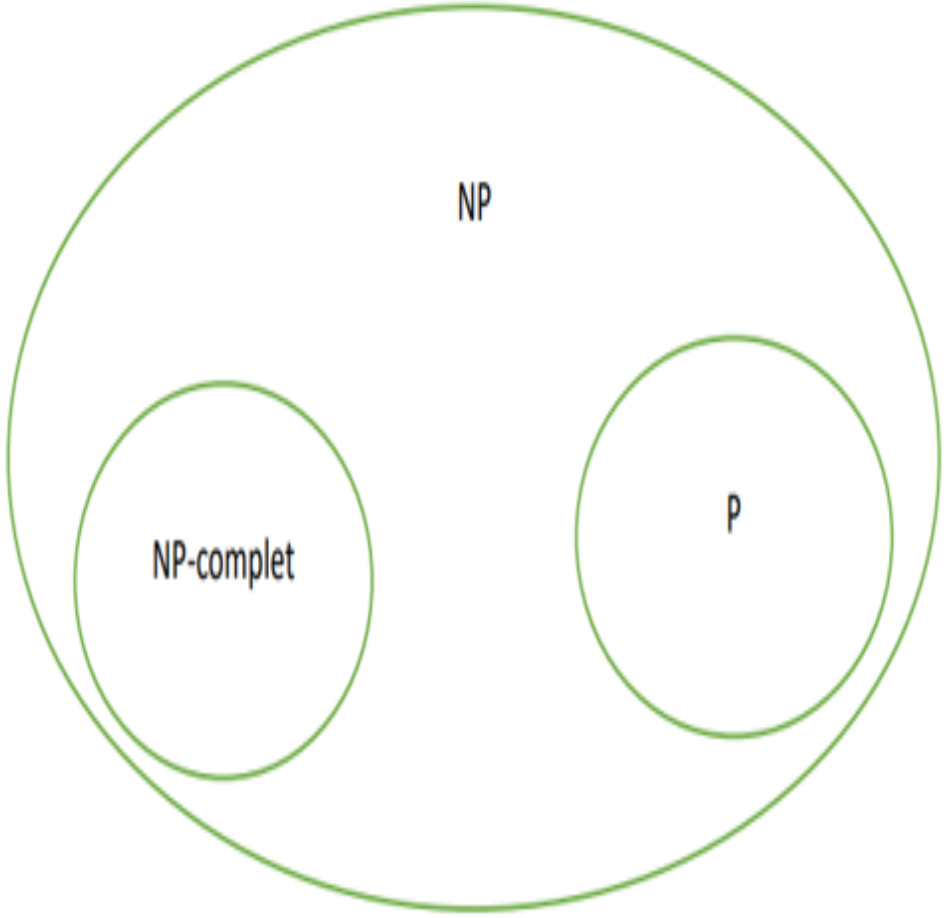
Définition : Une fonction f(n) est O(g(n)) (f(n) est de complexité g(n)), s'il existe un réel $c > 0$ et un entier positif n0 tel que pour tout $n \geq n0$ on a $| f(n) | \leq c.g(n)$.

- **Classe P** : Les problèmes faciles à résoudre, souvent pratiques et largement utilisés.
- **Classe NP** : Des problèmes complexes dont les solutions peuvent être vérifiées rapidement.
- **Classe NP-complet** : Les problèmes les plus complexes de NP, et leur résolution en temps polynomial reste un défi non résolu.

❑ Notions sur la complexité

- **Complexité algorithmique dans ces classes**

Classe	Temps de résolution	Exemple de problème
P	Polynomial (O(n^k), k constant)	Tri rapide (O(nlogn) , Dijkstra
NP	Vérification en temps polynomial	SAT, TSP, Sac à dos
NP-complet	Exponentiel (pire cas), aucun algo. polynomial connu	SAT, TSP, Coloration de graphe, Partitionnement

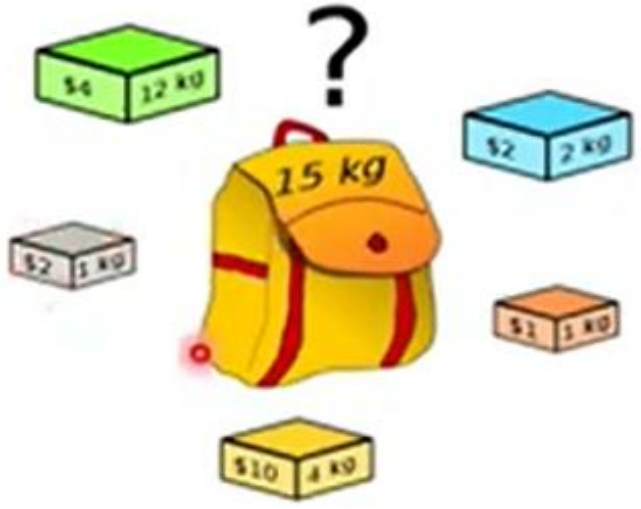


Optimisation Combinatoire

❑ Problème du Sac à Dos

Description : Le problème du sac à dos, aussi noté KP (en anglais, Knapsack Problem) est un problème d'optimisation combinatoire.

- Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble d'objets ayant chacun un poids et une valeur.
- Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.



❑ Problème du Sac à Dos

Formulation : Etant donné un ensemble de n objets chacun ayant un certain poids aj et une certaine valeur cj, et soit b un réel qui représente la charge (poids, volume, capacité) maximale que l'on peut emporter dans un sac à dos.

- La formulation du problème conduit à un PLNE0-1 à une seule contrainte :

$$\begin{cases} \text{Max } Z = \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_j x_j \le b \\ x_j \in \{0,1\} \end{cases}$$

xj = 1 si l'objet j est choisi

Variantes :

Knapsack entier : S'il existe plusieurs objets de chaque type ; soit Nj le nombre d'objets de type j, on obtient un PLNE

$$x_j \in \{0, 1, \ldots, N_j\} \text{ (ou } x_j \text{ entier si } N_j = \infty).$$

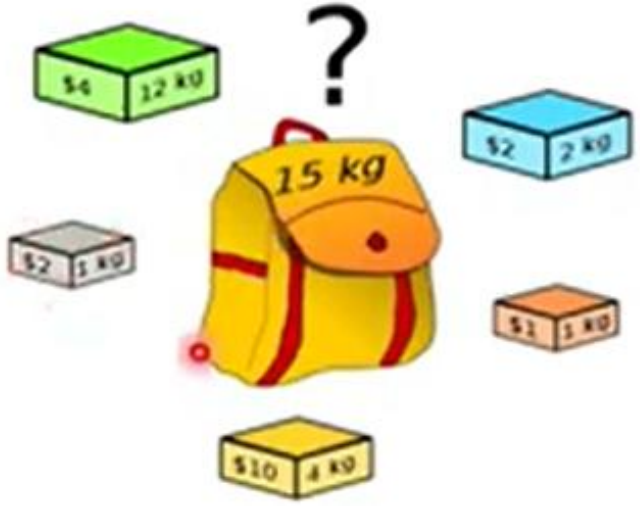
❑ Problème du Sac à Dos

Knapsack multi-dimensionnel : On considère ici que le sac à dos a d dimensions, avec d > 0 (d-KP).

Par exemple, on peut imaginer une boîte. Chaque objet a trois dimensions, et il ne faut pas déborder sur aucune des dimensions.

- La contrainte est alors remplacée par d contraintes : $\sum_{j=1}^{n} a_j^k x_j \leq b^k \quad \forall k = \{1, \ldots, d\}$

▪ **Exemple**



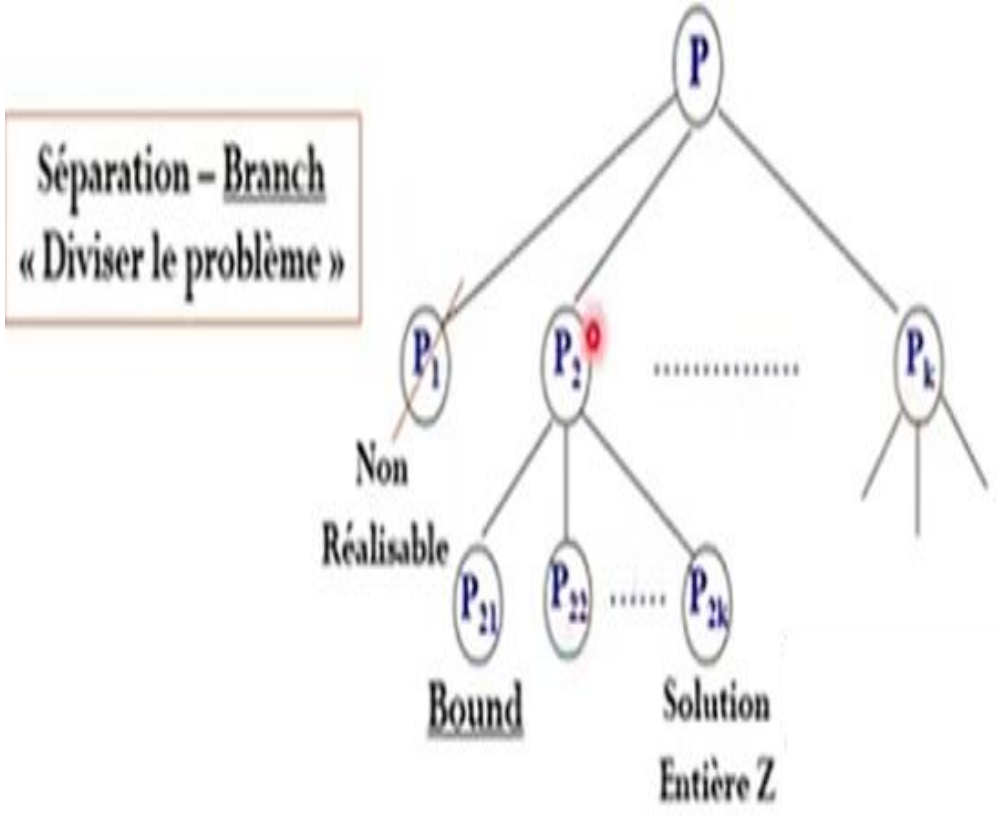
$$\begin{cases} \max Z = 2x_1 + 4x_2 + 2x_3 + x_4 + 10x_5 \\ x_1 + 12x_2 + 2x_3 + x_4 + 4x_5 \leq 15 \\ x_j \in \{0,1\}\ \forall j = 1, \ldots, 5 \end{cases}$$

Méthodes de résolution exactes

❑ Méthodes de résolution exactes : Branch and Bound

Les méthodes de résolution exactes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité.

- L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (B&B), 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 nœuds, et des feuilles.



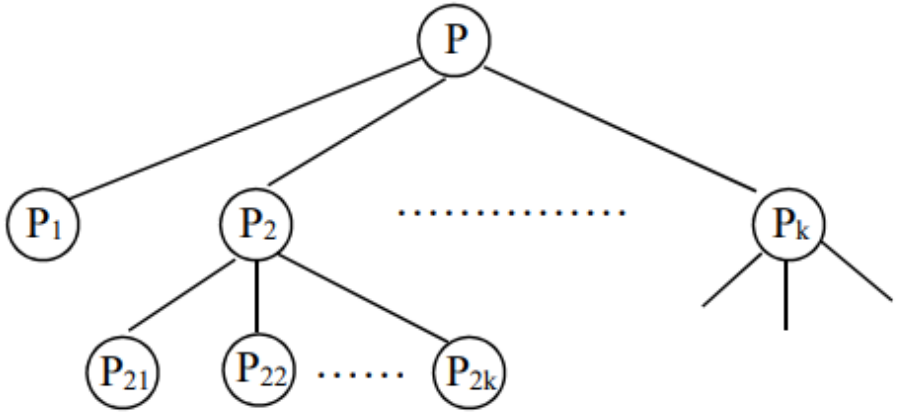
Méthodes de résolution exactes

❑ La méthode séparation et évaluation (Branch and Bound)

Le branch-and-bound est basé sur trois axes principaux :

- La séparation,
- L'évaluation,
- La stratégie de parcours.

- **La phase de séparation** est une étape où l'ensemble des solutions réalisables du problème initial est divisé en sous-ensembles plus petits et mutuellement exclusifs ce qui permet de structurer le problème en sous-problèmes plus faciles à gérer.

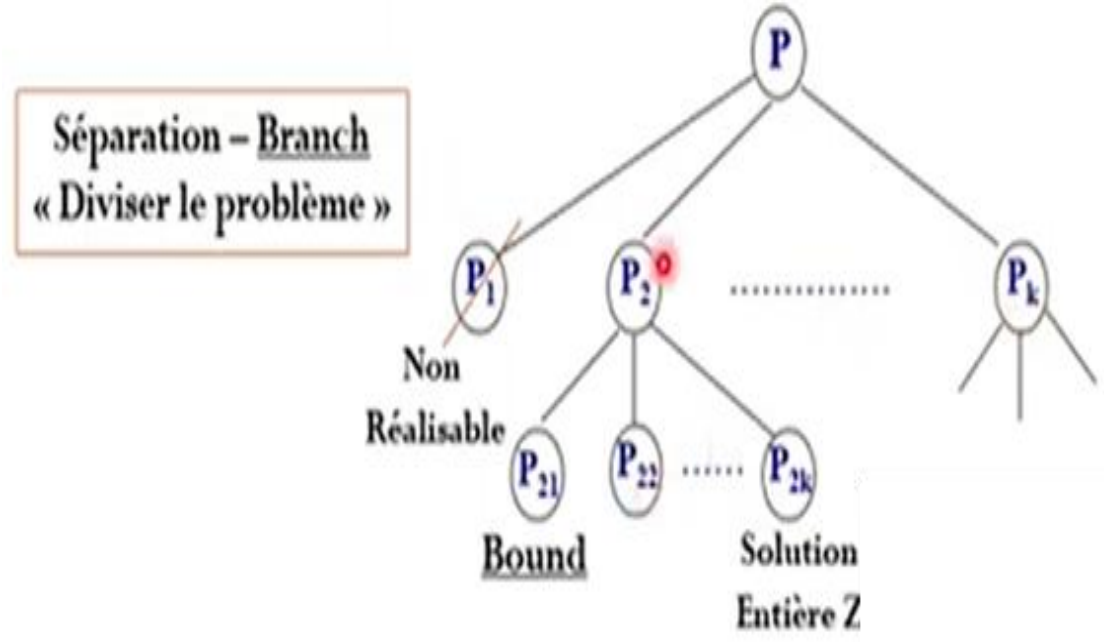


Méthodes de résolution exactes

❑ La méthode séparation et évaluation (Branch and Bound)

- **La phase d'évaluation** consiste à analyser ces sous-ensembles à l'aide de critères spécifiques. Elle vise à identifier les sous-ensembles susceptibles de contenir la solution optimale et à écarter ceux qui ne peuvent pas la contenir ce qui permet de réduire considérablement l'espace de recherche.

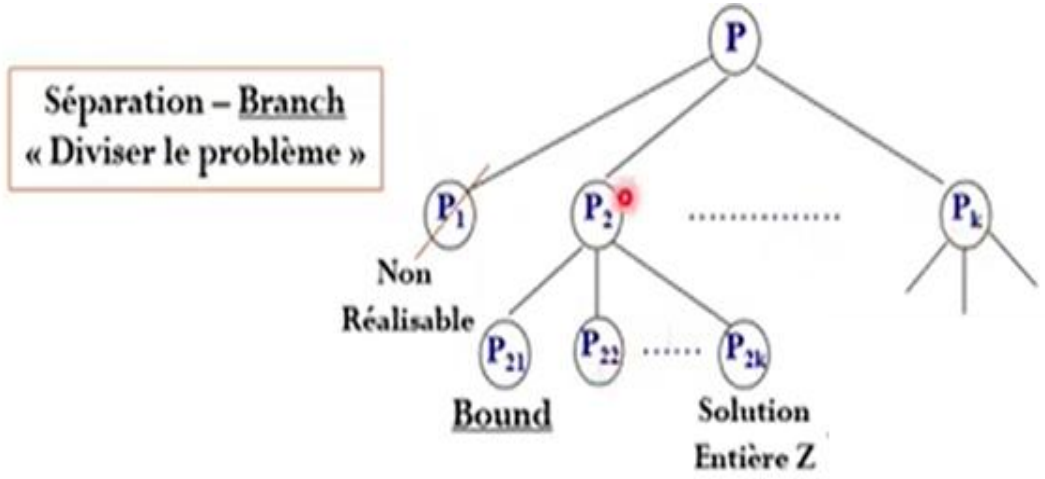
- Cette méthode se dote d'une fonction qui permet de mettre une borne inférieure (en cas de min) sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles.



Méthodes de résolution exactes

❑ La méthode séparation et évaluation (Branch and Bound)

- Le processus de séparation d'un sous-ensemble Pi s'arrête dans l'un des cas suivants (cas Min) :
  - Lorsque Pi admet une solution complète du problème initial.
  - Lorsque Pi n'admet pas de solution réalisable.
  - Lorsque la borne inférieure de Pi est à la meilleure solution trouvée jusqu'à maintenant pour le problème initial.



3 critères d'arrêt du Branch

- P_i admet une solution complète ou entière
- P_i n'admet pas de solution
- Bound : $Zinf(P_i) \geq Zbest$: min
  $Zsup(P_i) \leq Zbest$: max

Méthodes de résolution exactes

❑ La méthode séparation et évaluation (Branch and Bound)

- **La stratégie de parcours**
  - **La largeur d'abord** : Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées
  - **La 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,
  - **Le 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.

Méthodes de résolution exactes

❑ La méthode séparation et évaluation (Branch and Bound)

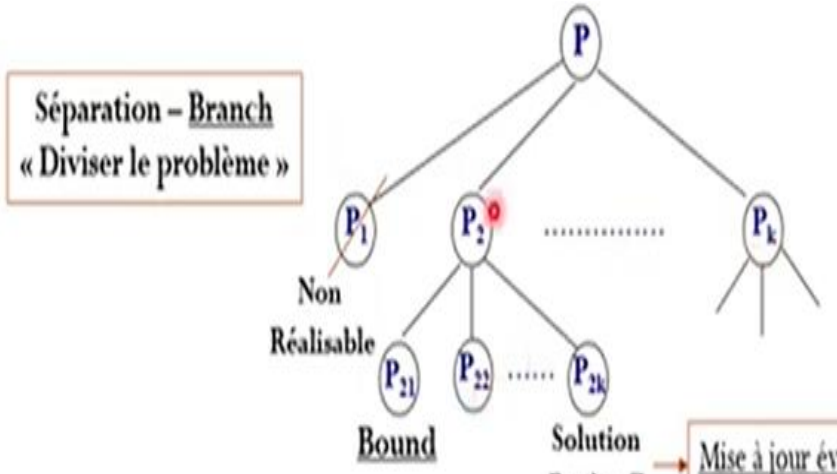
Algorithme général (cas Min)

A chaque instant, on maintient :

- une liste de sous problèmes actifs Pi
- Le coût Z de la meilleure solution obtenue jusqu'à maintenant (Initialisé à +inf ou à celui d'une solution initiale connue).

A une étape typique :

- Sélectionner un sous problème actif Pi
- Si Pi n'est pas réalisable, le supprimer (stop branch) sinon calculer sa borne inférieure Zinf(Pi)
- Si Zinf(Pi) ≥ Z, supprimer Pi (stop branch)
- Si Zinf(Pi) < Z, soit résoudre Pi directement, soit créer de nouveaux sous problèmes et les ajouter à la liste des sous problèmes actifs.



3 critères d'arrêt du Branch

- P_i admet une solution complète ou entière
- P_i n'admet pas de solution
- Bound : $Zinf(P_i) \geq Zbest$: min
  $Zsup(P_i) \leq Zbest$: max

Méthodes de résolution exactes

❑ Branch and Bound : Application au Problème du sac à dos

Exemple 1:

Une société dispose de 20 MDA à investir.

Les experts proposent 4 investissements possibles :

	Coût (MDA)	Bénéfice
Inv. 1	9	45
Inv. 2	7	21
Inv. 3	8	23
Inv. 4	7	11

Utilisation de la stratégie Best First

Méthodes de résolution exactes

❑ Branch and Bound : Application au Problème du sac à dos

Exemple 1:

	Coût (MDA)	Bénéfice	Rendement
Inv. 1	9	45	5.00
Inv. 2	7	21	3.00
Inv. 3	8	23	2.87
Inv. 4	7	11	1.57

Maximiser : $Z = 45x_1 + 21x_2 + 23x_3 + 11x_4$

$$9x_1 + 7x_2 + 8x_3 + 7x_4 \leq 20$$

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

1. Calcul des rendement ci/ai
2. Tri par ordre décroissant des rendements
3. Traiter le problème initial

$b = 20$

	a_i	c_i											
1	9	45											
2	7	21											
3	8	23											
4	7	11											

Méthodes de résolution exactes

❑ Branch and Bound : Application au Problème du sac à dos

Exemple 1:

	Coût (MDA)	Bénéfice	Rendement
Inv. 1	9	45	5.00
Inv. 2	7	21	3.00
Inv. 3	8	23	2.87
Inv. 4	7	11	1.57

Maximiser : $Z = 45x_1 + 21x_2 + 23x_3 + 11x_4$

$$9x_1 + 7x_2 + 8x_3 + 7x_4 \leq 20$$

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

	a_i	c_i	P										
1	9	45	1										
2	7	21	1										
3	8	23	1/2										
4	7	11	0										

$X^* = \begin{pmatrix} 1 \\ 1 \\ 1/2 \\ 0 \end{pmatrix} \rightarrow Zsup = 77{,}5$ $\bar{X} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \bar{Z} = 66$

P $Zsup = 77{,}5$

Zbest = 66

Méthodes de résolution exactes

❑ Branch and Bound : Application au Problème du sac à dos

Exemple 1:

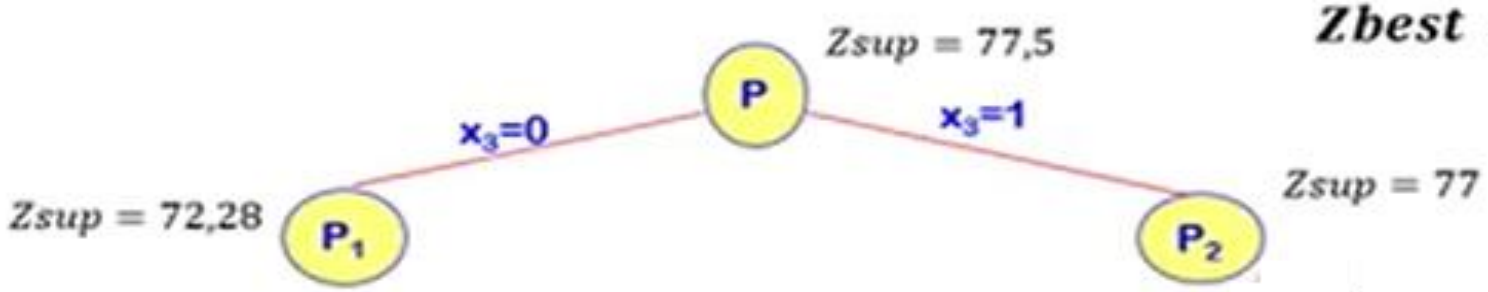
	Coût (MDA)	Bénéfice	Rendement
Inv. 1	9	45	5.00
Inv. 2	7	21	3.00
Inv. 3	8	23	2.87
Inv. 4	7	11	1.57

Maximiser : $Z = 45x_1 + 21x_2 + 23x_3 + 11x_4$

$$9x_1 + 7x_2 + 8x_3 + 7x_4 \leq 20$$

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

	a_i	c_i	P	P1	P2								
1	9	45	1	1	1								
2	7	21	1	1	3/7								
3	8	23	1/2	0	1								
4	7	11	0	4/7	0								

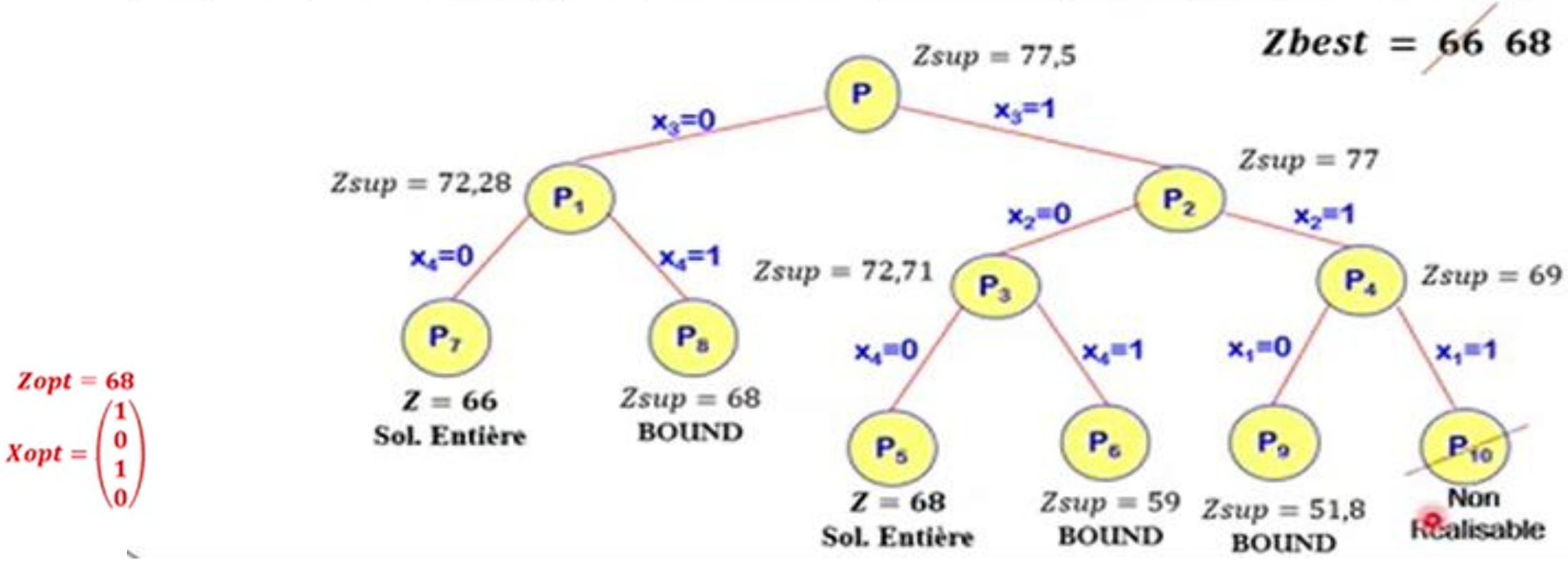


Méthodes de résolution exactes

❑ Branch and Bound : Application au Problème du sac à dos

Exemple 1:

	a_i	c_i	P	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	9	45	1	1	1	1	5/9	1	5/9	1	1	0	1
2	7	21	1	1	3/7	0	1	0	0	1	4/7	1	1
3	8	23	1/2	0	1	1	1	1	1	0	0	1	1
4	7	11	0	4/7	0	3/7	0	0	1	0	1	5/7	?



Méthodes de résolution exactes

❑ Branch and Bound : Application

Exemple 2:

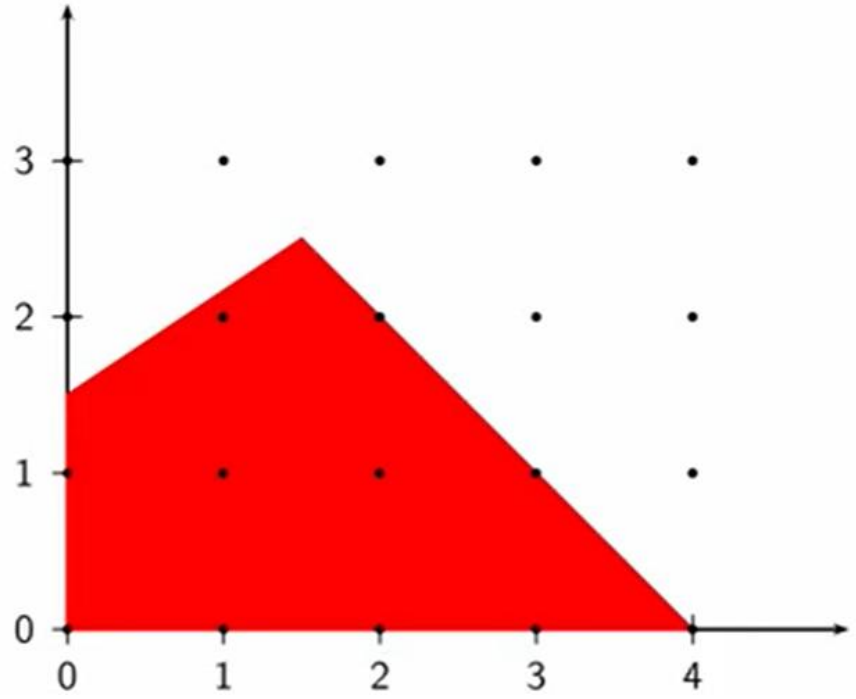
Soit le problème P_0

$$\min x_1 - 2x_2$$

sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{N}. \end{aligned}$$

Note : $(0, 0)$ est admissible. Donc $U = 0$.



Méthodes de résolution exactes

❑ Branch and Bound : Application

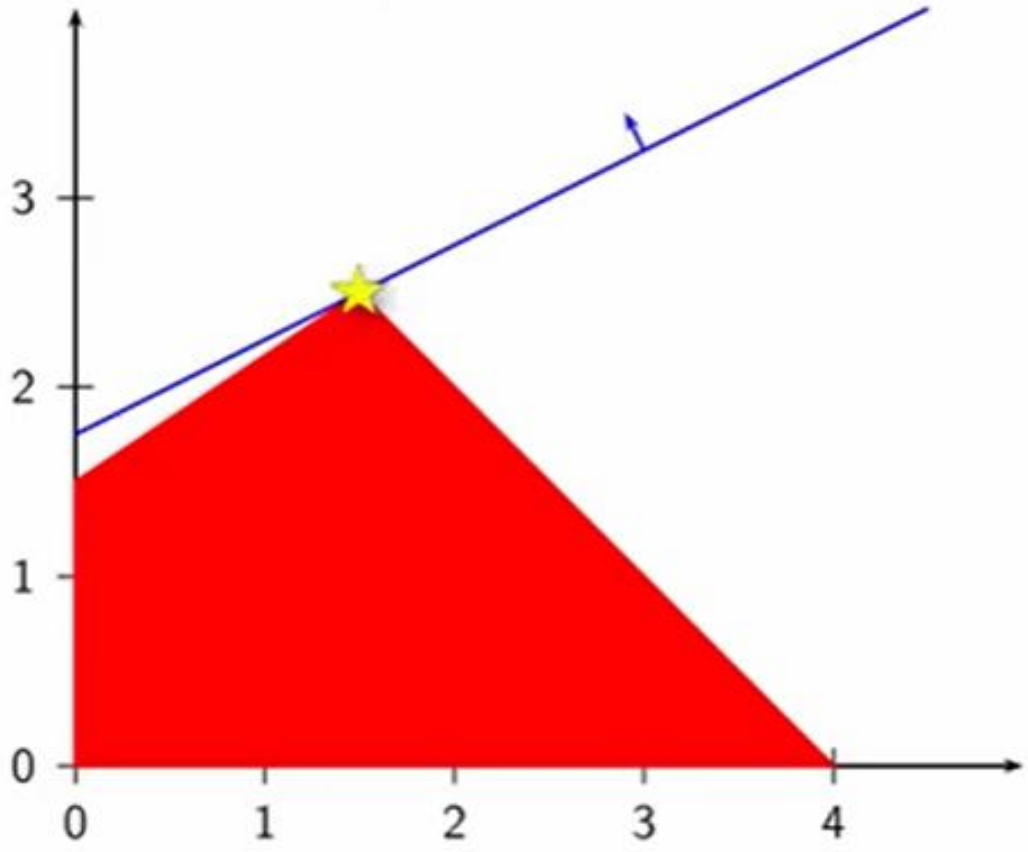
Exemple 2:

Problème relaxé $R(P_0)$.

$$\min x_1 - 2x_2$$

sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$



Solution optimale de $R(P_0)$: $(1.5, 2.5)$

Borne pour P_0 : $b_0 = -3.5$

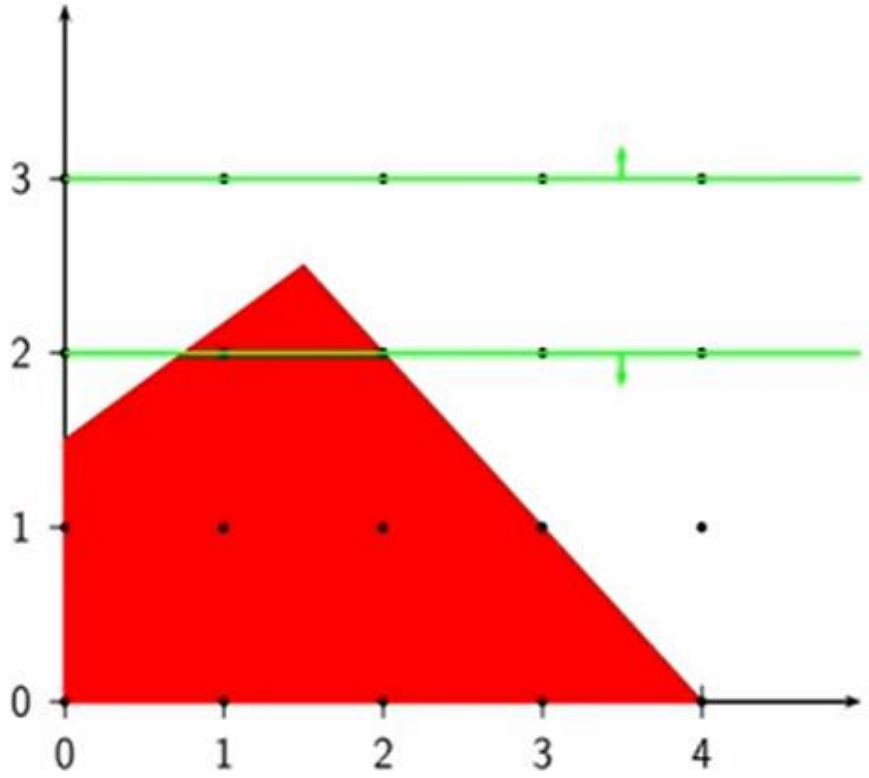
Méthodes de résolution exactes

❑ Branch and Bound : Application

Exemple 2:

- Solution optimale de $R(P_0)$: $(1.5, 2.5)$
- Borne pour P_0 : $b_0 = -3.5$
- $x_2 = 2.5$ est non entier. Partition :

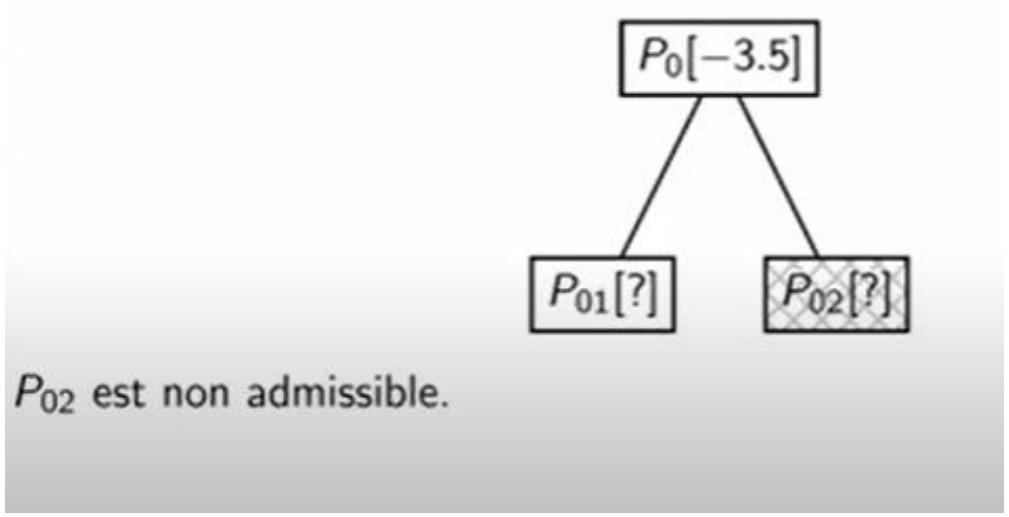
P_{01}			P_{02}		
$\min x_1 - 2x_2$			$\min x_1 - 2x_2$		
s.c.			s.c.		
$-4x_1 + 6x_2$	$\leq$	9	$-4x_1 + 6x_2$	$\leq$	9
$x_1 + x_2$	$\leq$	4	$x_1 + x_2$	$\leq$	4
x_1, x_2	$\geq$	0	x_1, x_2	$\geq$	0
x_1, x_2	$\in$	$\mathbb{N}$	x_1, x_2	$\in$	$\mathbb{N}$
x_2	$\leq$	2	x_2	$\geq$	3

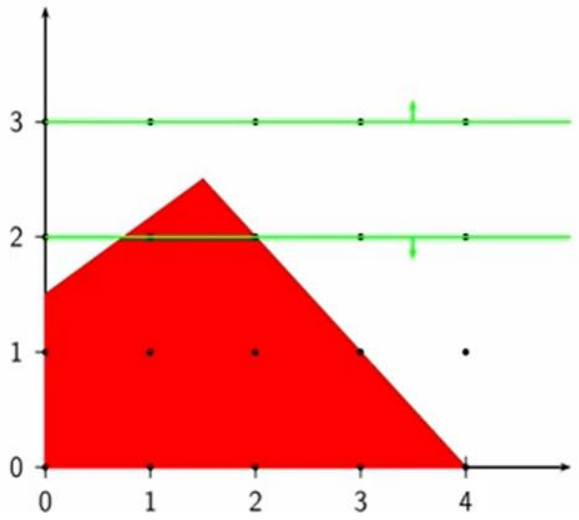


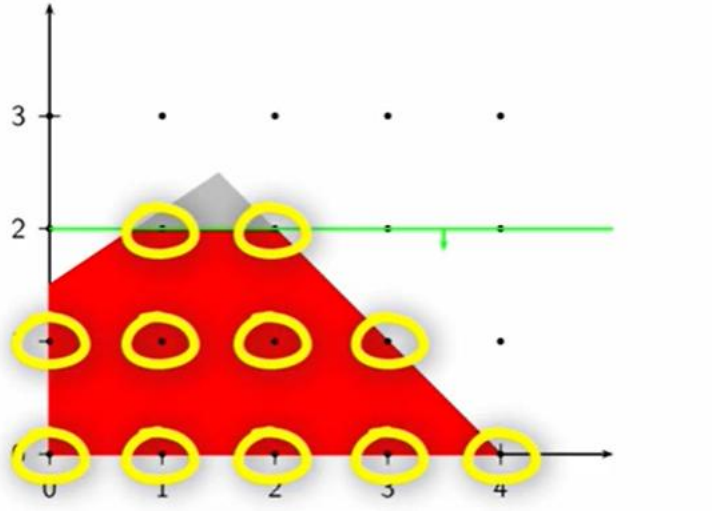
Méthodes de résolution exactes

❑ Branch and Bound : Application

Exemple 2:







Méthodes de résolution exactes

❑ Branch and Bound : Application

Exemple 2:

- Problème P_{01} : $\min x_1 - 2x_2$ sous contraintes

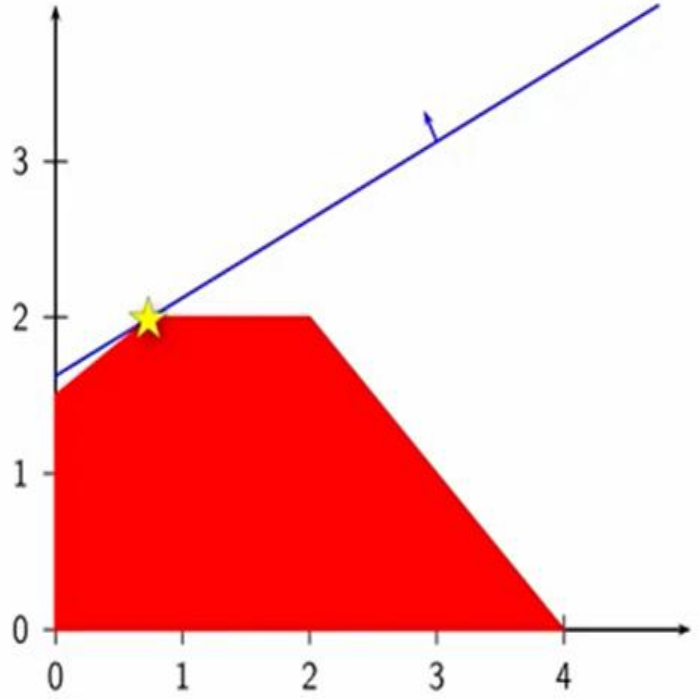
$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_2 &\leq 2 \\ x_1, x_2 &\in \mathbb{N}. \end{aligned}$$

- Problème relaxé $R(P_{01})$: $\min x_1 - 2x_2$ sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_2 &\leq 2 \end{aligned}$$

Solution optimale de $R(P_{01})$: $(0.75, 2)$

Borne pour P_{01} : $b_{01} = -3.25$



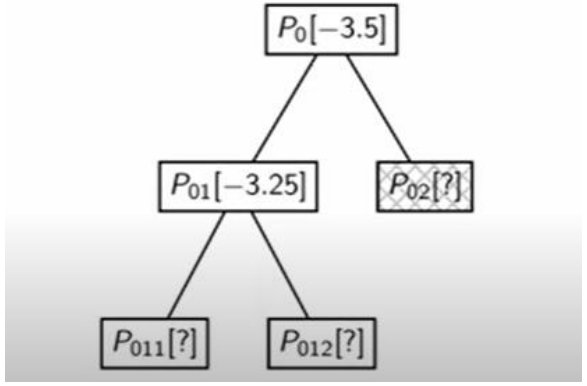
Méthodes de résolution exactes

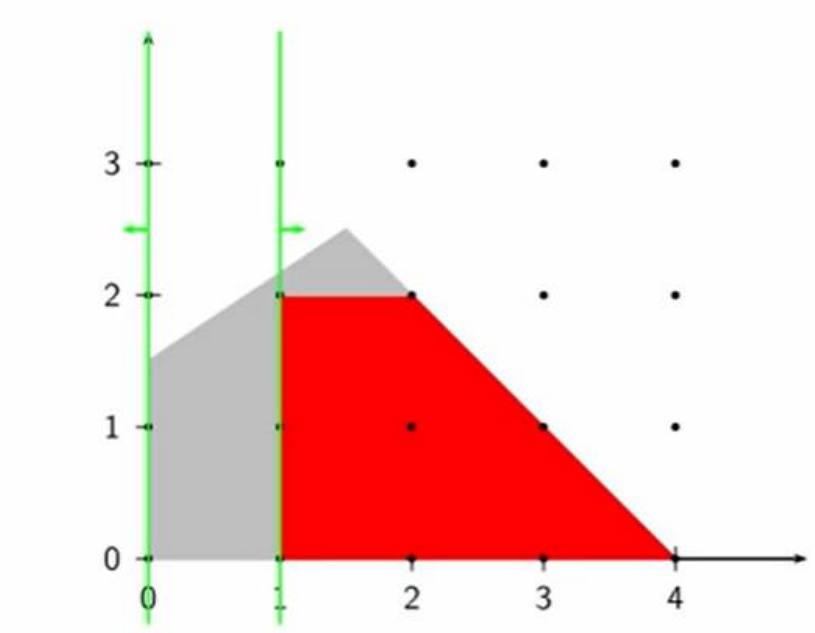
❑ Branch and Bound : Application

Exemple 2:

- Solution optimale de $R(P_{01})$: $(0.75, 2)$
- Borne pour P_{01} : $b_{01} = -3.25$
- $x_1 = 0.75$ est non entier. Partition :

P_{011}			P_{012}		
$\min x_1 - 2x_2$			$\min x_1 - 2x_2$		
s.c.			s.c.		
$-4x_1 + 6x_2$	$\leq$	9	$-4x_1 + 6x_2$	$\leq$	9
$x_1 + x_2$	$\leq$	4	$x_1 + x_2$	$\leq$	4
x_1, x_2	$\geq$	0	x_1, x_2	$\geq$	0
x_1, x_2	$\in$	$\mathbb{N}$	x_1, x_2	$\in$	$\mathbb{N}$
x_2	$\leq$	2	x_2	$\leq$	2
x_1	$\leq$	0	x_1	$\geq$	1





Méthodes de résolution exactes

❑ Branch and Bound : Application

Exemple 2:

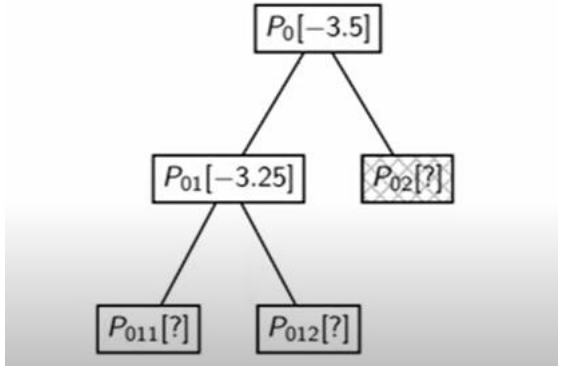
- Problème P_{011} : $\min x_1 - 2x_2$ sous contraintes

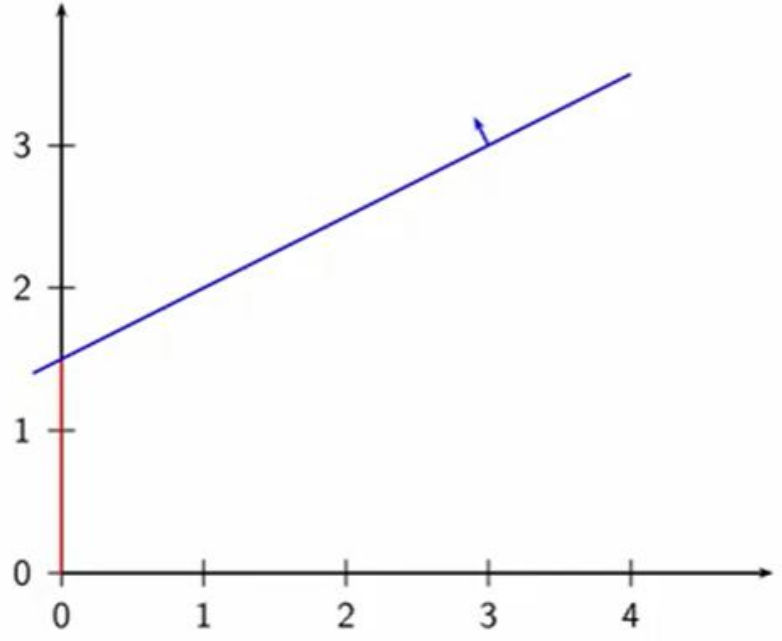
$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_2 &\leq 2 \\ x_1 &\leq 0 \\ x_1, x_2 &\in \mathbb{N}. \end{aligned}$$

- Problème relaxé $R(P_{01})$: $\min x_1 - 2x_2$ sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_2 &\leq 2 \\ x_1 &\leq 0 \end{aligned}$$

- Solution optimale de $R(P_{011})$: $(0, 1.5)$
- Borne pour P_{011} : $b_{011} = -3$





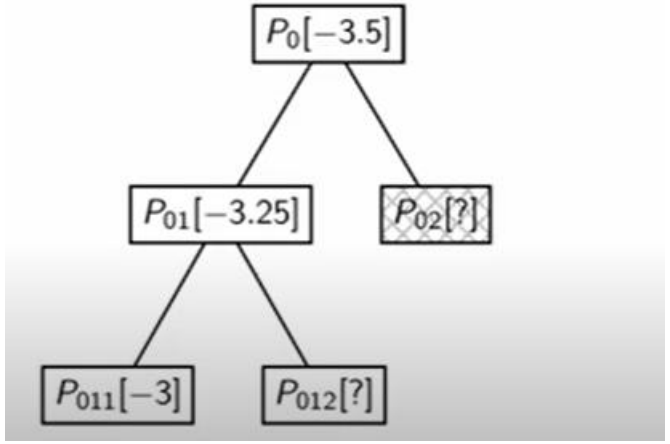
Méthodes de résolution exactes

❑ Branch and Bound : Application

Exemple 2:

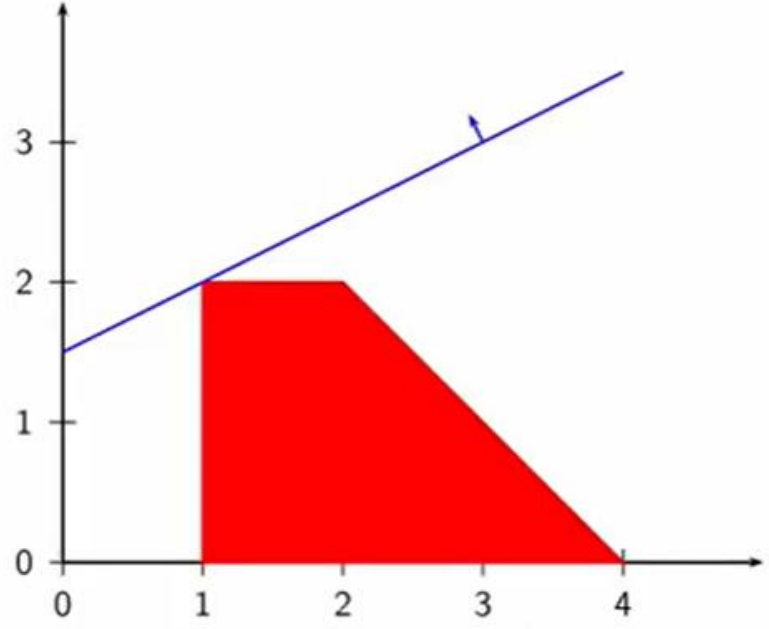
- Problème P_{012} : $\min x_1 - 2x_2$ sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_2 &\leq 2 \\ x_1 &\geq 1 \\ x_1, x_2 &\in \mathbb{N}. \end{aligned}$$



- Problème relaxé $R(P_{01})$: $\min x_1 - 2x_2$ sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_2 &\leq 2 \\ x_1 &\geq 1 \end{aligned}$$



- Solution optimale de $R(P_{012})$: $(1,2)$
- Solution entière.
- C'est donc la solution optimale pour P_{012}.
- $U = -3$.

Méthodes de résolution exactes

❑ Branch and Bound : Application

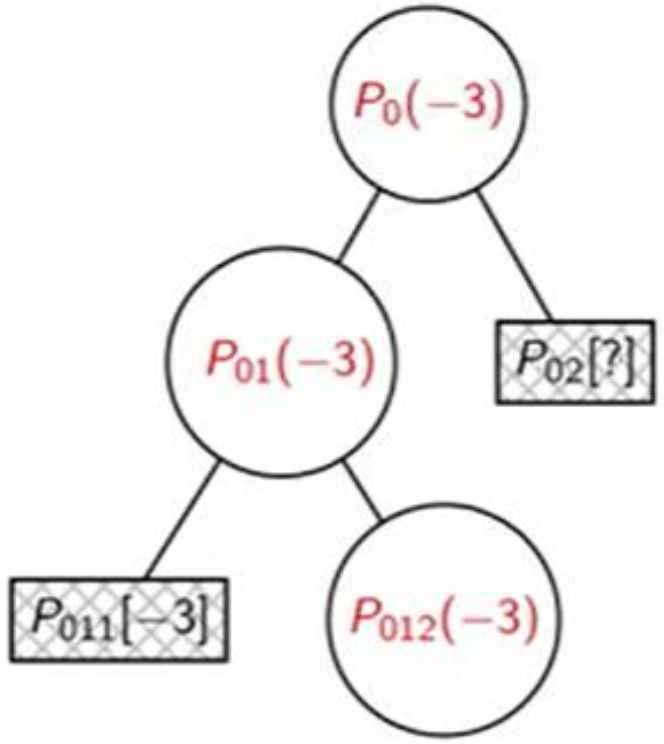
Exemple 2:

Soit le problème P_0

$$\min x_1 - 2x_2$$

sous contraintes

$$\begin{aligned} -4x_1 + 6x_2 &\leq 9 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{N}. \end{aligned}$$



- Solution optimale de $R(P_{012})$: $(1, 2)$
- Solution entière.
- C'est donc la solution optimale pour P_{012}.
- $U = -3$.

❑ Branch and Bound : Application

Exercice 01 :

Résoudre par la méthode de **Branch-Bound** le problème de sac-à-dos suivant :

$$(P_1) = \begin{cases} Z_{max} = 14x_1 + 15x_2 + 4x_3 + 5x_4 \\ 5x_1 + 6x_2 + 2x_3 + 3x_4 \leq 8, \\ x_i \in \{0,1\}, i = \overline{1,4}. \end{cases}$$

Exercice 02 :

Résoudre le problème suivant à l'aide de l'algorithme de **Branch-Bound :**

$$(P_2) = \begin{cases} Max(2x_1 + 3x_2), \\ x_1 + 2x_2 \leq 3, \\ 6x_1 + 8x_2 \leq 15, \\ x_1, x_2 \; entier. \end{cases}$$

Méthodes de résolution exactes

❑ Branch and Bound : Application

La fonction **intlinprog** dans MATLAB est utilisée pour résoudre des problèmes de **programmation linéaire en nombres entiers**. Elle permet d'optimiser une fonction objective linéaire sous des contraintes linéaires, en imposant que certaines ou toutes les variables soient des **entiers**.

```
[x, fval] = intlinprog(f, intcon, A, b, Aeq, beq, lb, ub)
```

Paramètres :

- **f** : Vecteur des coefficients de la fonction objective. Le but est de minimizer f(x).
- **intcon** : Indices des variables qui doivent être des entiers.
- **A et b** : Définit les contraintes de type inégalités linéaires ($A \cdot x \leq b$).
- **Aeq et beq** : Définit les contraintes de type égalités linéaires ($Aeq \cdot x = beq$).
- **lb et ub** : Définit les bornes inférieures et supérieures sur les variables ($lb \leq x \leq ub$).

Retourne :

- **x** : Solution optimale.
- **fval** : Valeur optimale de la fonction objective.

Méthodes de résolution exactes

❑ Branch and Bound : Application

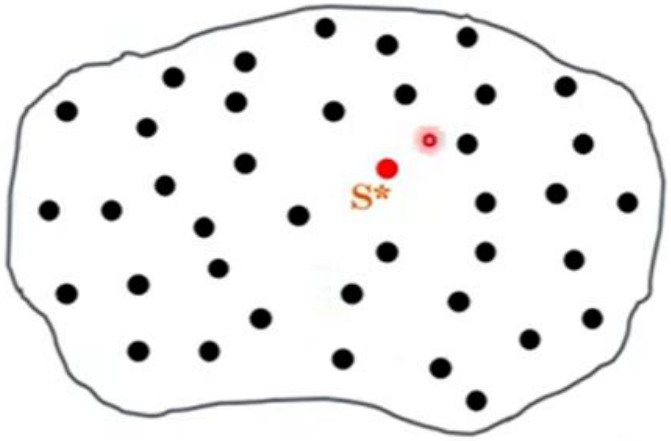
```
% Coefficients de la fonction objectif (maximisation)
    f = [-14, -15, -4, -5]; % On minimise l'opposé pour maximisation
% Contraintes
    A = [5, 6, 2, 3]; % Matrice des contraintes
    b = 8;              % Valeur maximale pour la contrainte
% Bornes des variables (x_i binaires, donc entre 0 et 1)
    lb = [0, 0, 0, 0]; % Borne inférieure
    ub = [1, 1, 1, 1]; % Borne supérieure
% Indices des variables binaires
    intcon = 1:4; % Les variables sont toutes entières

% Résolution avec intlinprog
    [x, z] = intlinprog(f, intcon, A, b, [], [], lb, ub);

% Affichage des résultats
disp('Solution optimale :');
disp(x');
disp(['Valeur optimale de Z :',num2str(-z)]); % -z pour revenir à la maximisation
```

Méthodes de résolution approchées

❑ Introduction

Les **méthodes de résolution approchées** sont des techniques utilisées pour résoudre des problèmes d'optimisation complexes ou de grande taille, où les méthodes exactes sont impraticables en raison de contraintes de temps ou de ressources.

Contrairement aux méthodes exactes, qui garantissent une solution optimale, les méthodes approchées fournissent une solution "suffisamment bonne" dans un délai raisonnable.



S* : Solution optimale, un REVE !

- **Instance complexe et de grande taille,**
- **Espace des solutions non convexe.**
- **L'utilisation d'une méthode exacte devient impossible**

Méthodes de résolution approchées

❑ Introduction

Les méthodes approchées se caractérisent par :

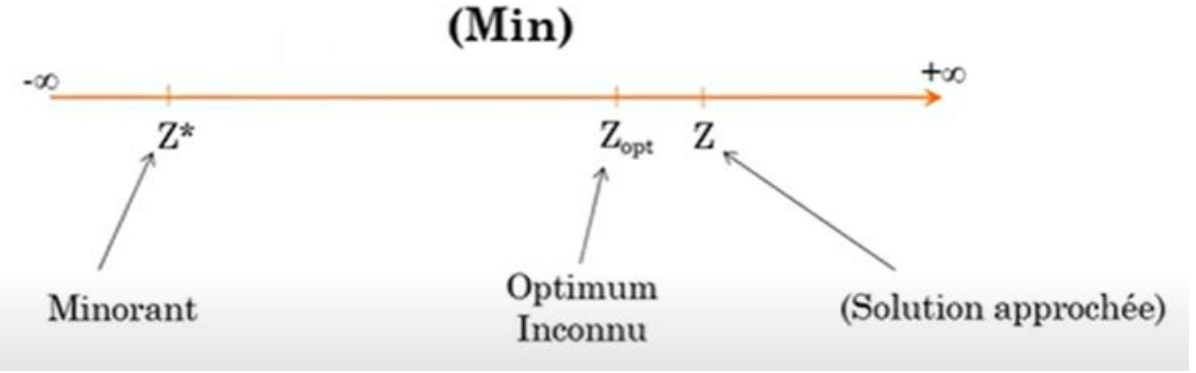
- **Efficacité** : Elles trouvent une solution en un temps raisonnable, même pour des problèmes complexes.
- **Compromis optimalité-temps** : Elles sacrifient l'optimalité garantie pour obtenir des résultats utilisables rapidement.
- **Flexibilité** : Elles peuvent s'appliquer à des problèmes variés et souvent non linéaires ou non convexes.
- **Non exhaustivité** : Elles ne nécessitent pas d'explorer tout l'espace des solutions
- **Robustesse** : Elles sont souvent capables de produire des résultats fiables même en présence d'incertitudes ou d'instances mal conditionnées.
- **Applications variées** : Elles sont utilisées dans des domaines tels que la logistique, l'intelligence artificielle, la gestion industrielle, et l'apprentissage automatique.

Méthodes de résolution approchées

❑ Problème d'évaluation de la qualité d'une solution

L'évaluation de la qualité d'une solution est une étape cruciale dans les algorithmes d'optimisation. Elle consiste à mesurer à quel point une solution trouvée est proche de la solution optimale, tout en tenant compte des contraintes spécifiques au problème.

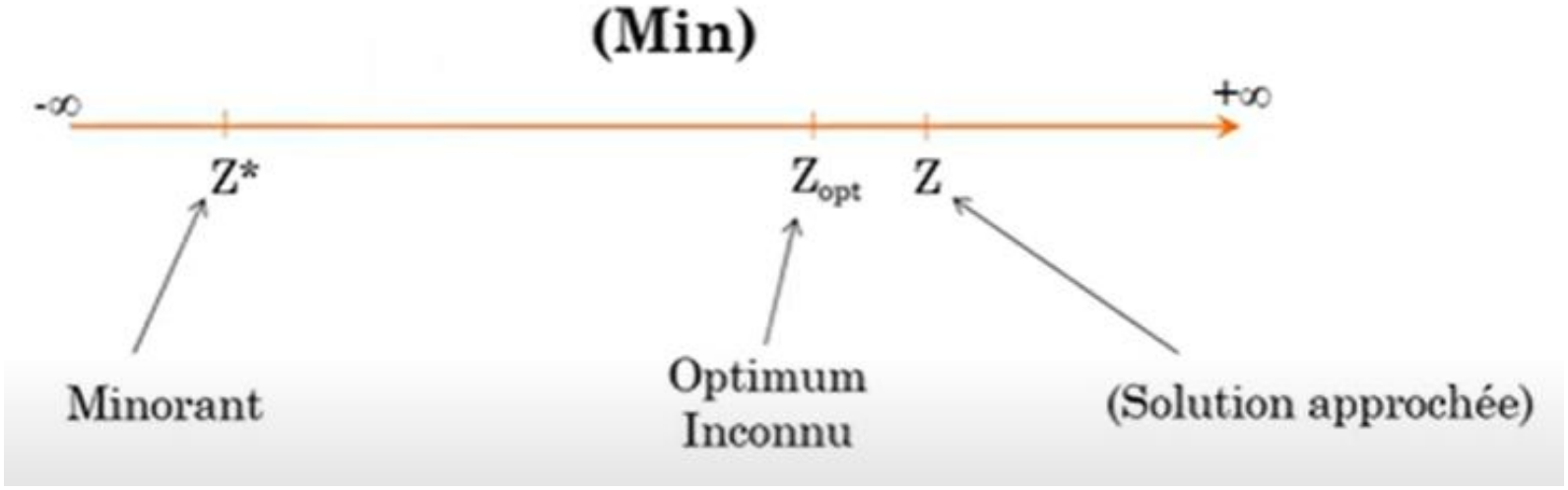
- **Définir les bornes inf (Zinf) ou sup (Zsup)** : Représente la meilleure performance théorique possible que l'on peut atteindre, calculée par des méthodes exactes (Branch-and-Bound, relaxation linéaire).
- **Calculer la solution obtenue (Zopt) :** La solution Zopt est généralement obtenue par une méthode approchées.
- **Mesurer l'écart** : L'écart entre Z et la borne sert à évaluer la qualité de la solution (écart relatif écart absolu). Ces métriques montrent à quel point la solution Z s'approche de l'optimalité théorique.



Méthodes de résolution approchées

❑ Problème d'évaluation de la qualité d'une solution

- **Considérer des métriques complémentaires :**
  - **Temps de calcul** : Si une solution est proche de l'optimalité mais obtenue rapidement, elle peut être jugée satisfaisante.
  - **Robustesse** : La solution obtenue est-elle stable face à des perturbations ou des variations dans les données ?

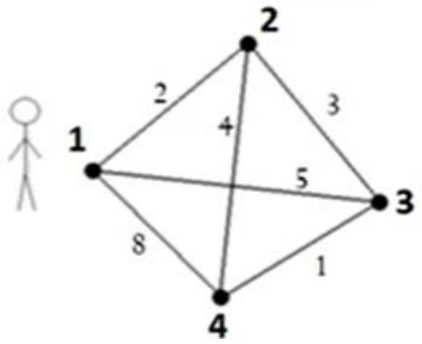


Méthodes de résolution approchées

❑ Types de méthodes approchées

- **Heuristiques :** elles utilisent des règles simples ou des stratégies guidées pour explorer l'espace des solutions d'un problème spécifique:

  - **Glouton (Greedy):** Une approche où l'on choisit à chaque étape la meilleure option locale en espérant atteindre l'optimum global.
  - **Construction progressive:** Une méthode qui construit progressivement une solution complète en ajoutant des composants de manière incrémentale.
  - **Partitionnement:** Une technique consistant à diviser un problème en sous-problèmes plus petits pour faciliter la résolution.



TSP : Plus proche voisin

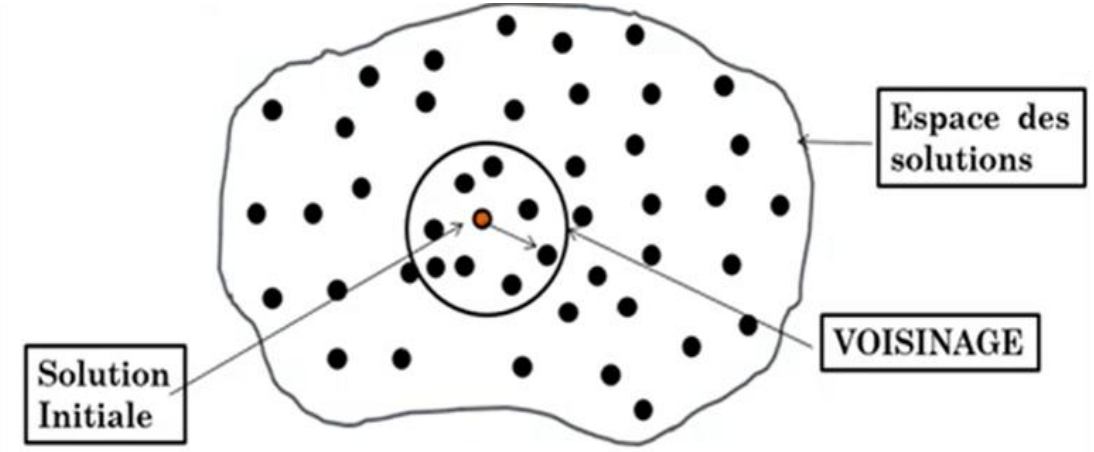
KP	ai	ci	ci/ai	Sol. glouton
1	9	45	5.00	1
2	7	21	3.00	1
3	8	23	2.87	0
4	7	11	1.57	0
b	20			Cout : 66

Méthodes de résolution approchées

❑ Méthode de descente & Optimalité locale

La **méthode de descente** est une approche courante pour résoudre des problèmes d'optimisation. Elle repose sur le principe suivant : **une solution est améliorée progressivement en se déplaçant dans la direction qui réduit (ou maximise) le plus rapidement la valeur de la fonction objectif.**

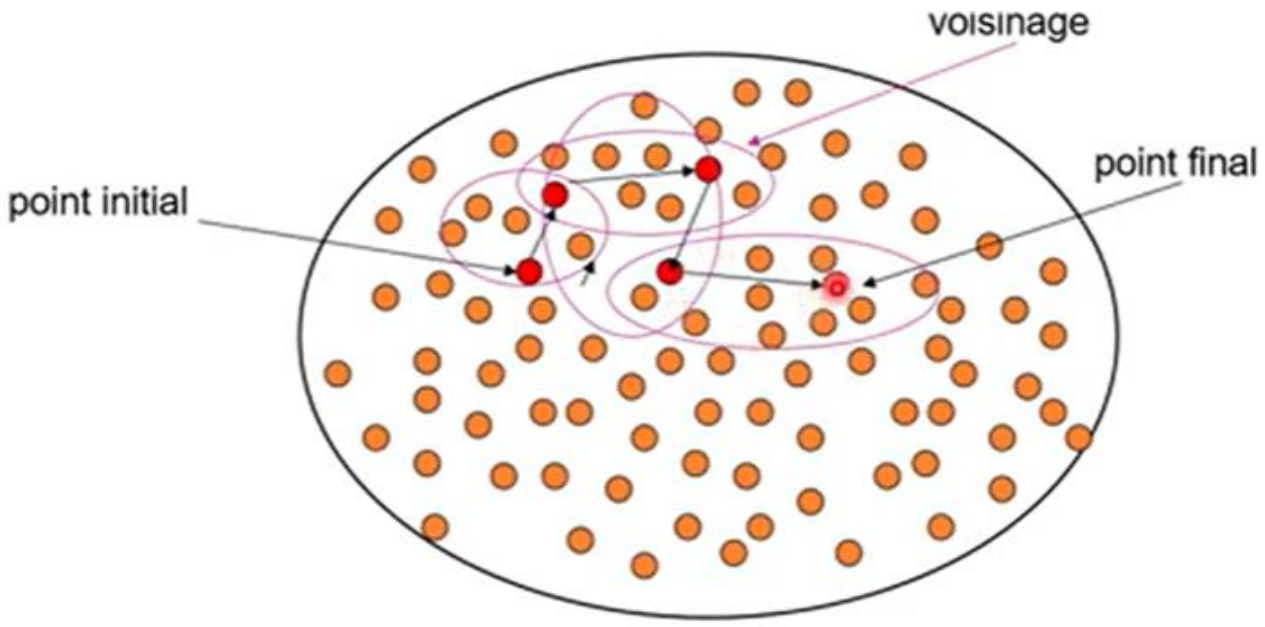
- **Solution initiale** : Le point de départ pour la recherche de l'optimum dans l'espace des solutions.
- **Passer à une solution voisine de meilleure qualité** :Le principe est de rechercher dans le voisinage (Neighborhood) de la solution actuelle une solution meilleure.
- **Arrêt si plus d'amélioration** : La méthode s'arrête lorsqu'il n'existe plus de solution voisine meilleure, ce qui indique qu'une **solution localement optimale** a été trouvée.



Ex Knapsack : $X = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ Z=45 ➔ Voisin : $X' = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ Z'=53

Méthodes de résolution approchées

❑ Méthode de descente & Optimalité locale



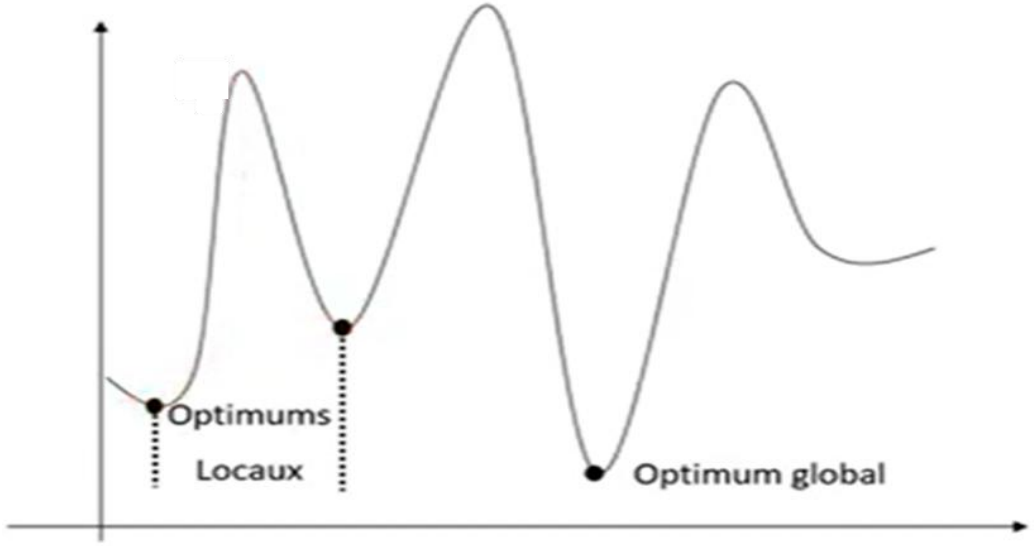
Si tous les voisins d'une solution sont de mauvaise qualité, alors cette solution ne serait-elle pas la solution optimale ?

La méthode peut atteindre un optimum local, sans garantir l'optimum global, en fonction de la solution initiale et de la configuration de l'espace des solutions.

Méthodes de résolution approchées

❑ Méthode de descente & Optimalité locale

La méthode de descente aboutit généralement à un optimum local, sans garantir l'atteinte de l'optimum global



Pour dépasser un optimum local, des techniques comme le recuit simulé ou la recherche taboue permettent d'explorer des solutions au-delà du voisinage immédiat.

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

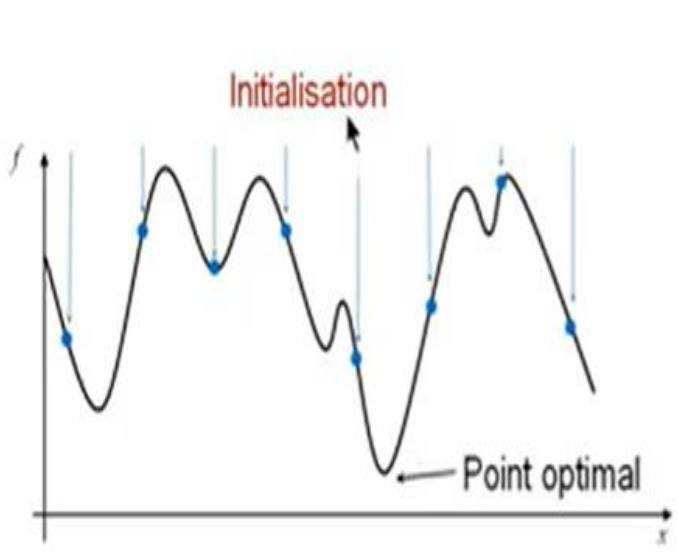
Les méta-heuristiques sont méthodes d'optimisation approximatives qui peuvent impliquer l'aspect aléatoire dans leur processus de recherche et qui permettent d'aborder des instances de problèmes difficiles en fournissant des solutions satisfaisantes (solution optimale ou proche de l'optimale) dans un délai raisonnable :

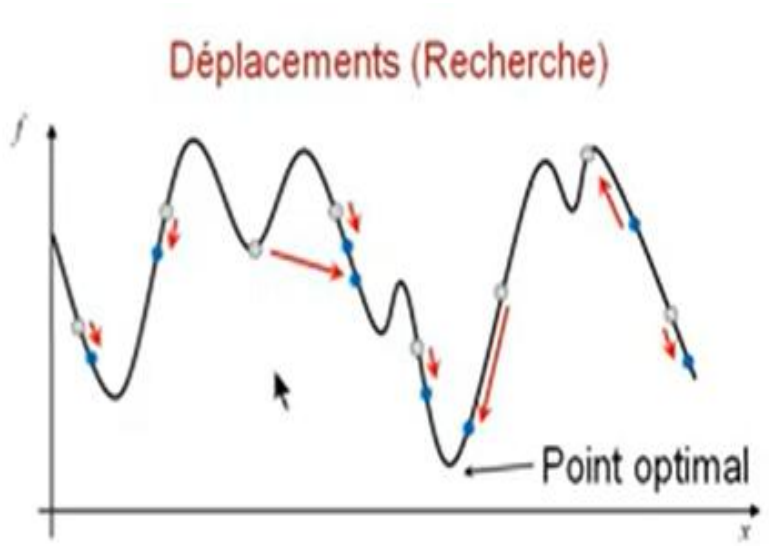
- Indépendant du problème ou cas à traiter (POC).
- Échapper à l'optimalité locale en Acceptant une dégradation temporaire de la qualité de la solution pour explorer de nouvelles zones prometteuses dans l'espace des solutions.
- Inspirer de phénomènes naturels :
  - ✓ **Algorithmes génétiques** : Basés sur la sélection naturelle (Holland, 1975).
  - ✓ **Recuit simulé** : Inspiré des processus de métallurgie (Kirkpatrick, 1983).
  - ✓ **Recherche tabou** : Basée sur la mémoire humaine (Glover, 1986).
  - ✓ **Algorithmes des fourmis** : Inspirés du comportement des colonies de fourmis (Dorigo, 1990).

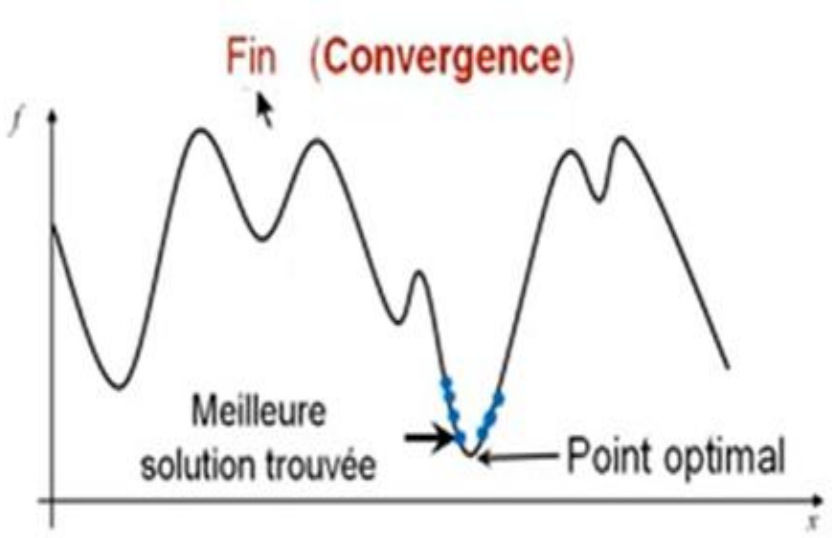
Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Exemple d'un comportement d'une métaheuristique**







Approches plus générales et adaptatives qui combinent exploration et exploitation de l'espace des solutions

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

▪ Exploration/Exploitation

- **L'exploration** consiste à diversifier la recherche en examinant différentes parties de l'espace des solutions afin d'éviter de rester bloqué dans un optimum local en découvrant de nouvelles zones prometteuses.
- **L'exploitation** intensifie la recherche autour des solutions déjà identifiées comme intéressantes afin d'approfondir et affiner les solutions dans une zone prometteuse pour se rapprocher de l'optimum.

▪ Équilibre entre exploration et exploitation :

Les métaheuristiques doivent trouver un compromis entre :

- **Exploration** (diversification) pour éviter de manquer de meilleures solutions dans d'autres régions.
- **Exploitation** (intensification) pour maximiser les gains autour des solutions prometteuses.

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Avantages/Inconvinients**

Avantages :

- Solutions pour les problèmes NP-difficiles
- Pas besoin de gradient
- Récupération des optima locaux
- Multi-objectifs
- Applicabilité universelle
- Combinabilité
- Facilité de mise en œuvre

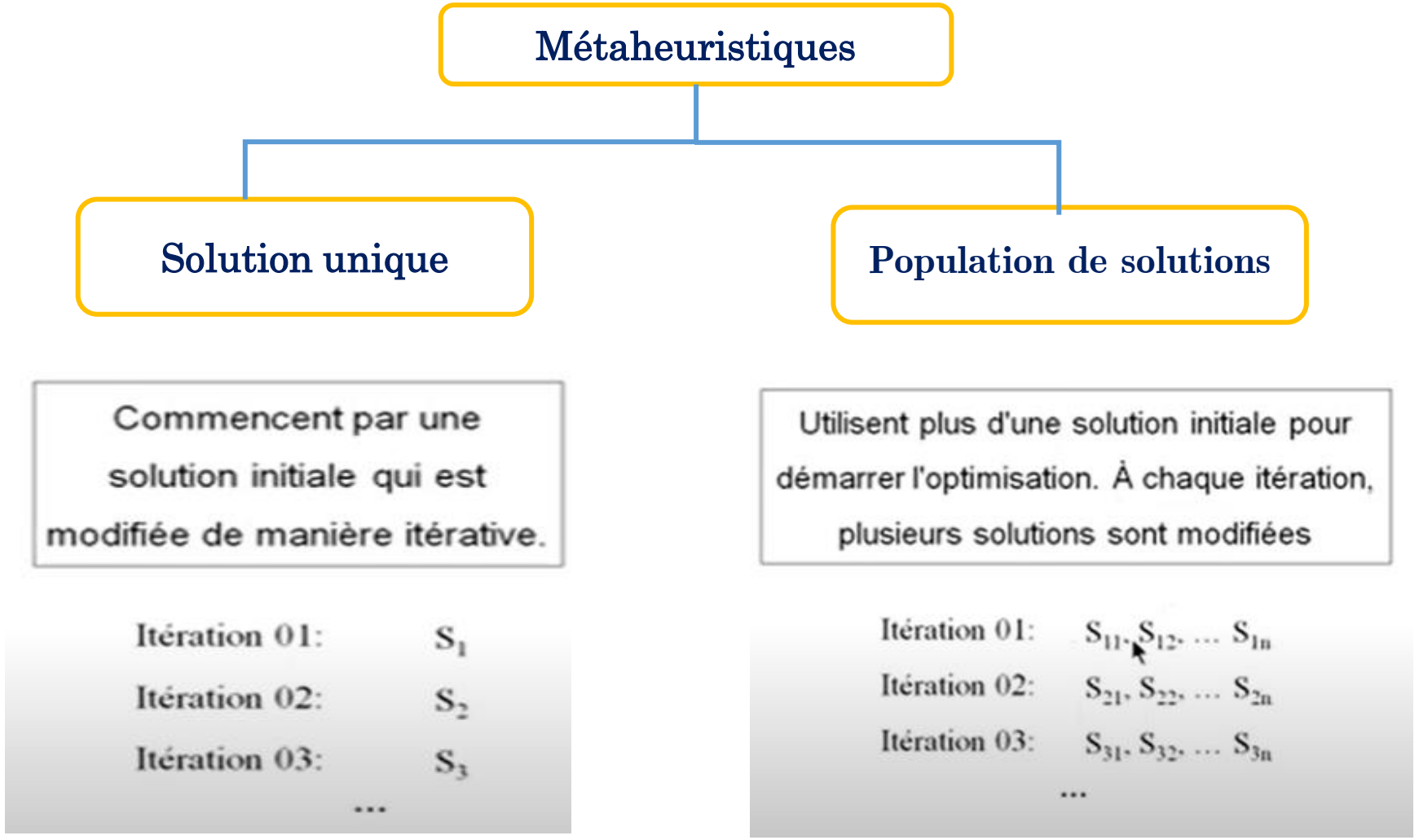
Inconvénients :

- Absence de fondement mathématique solide
- Pas de garantie d'optimalité
- Impossibilité de prouver l'optimalité
- Dépendance aux paramètres

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

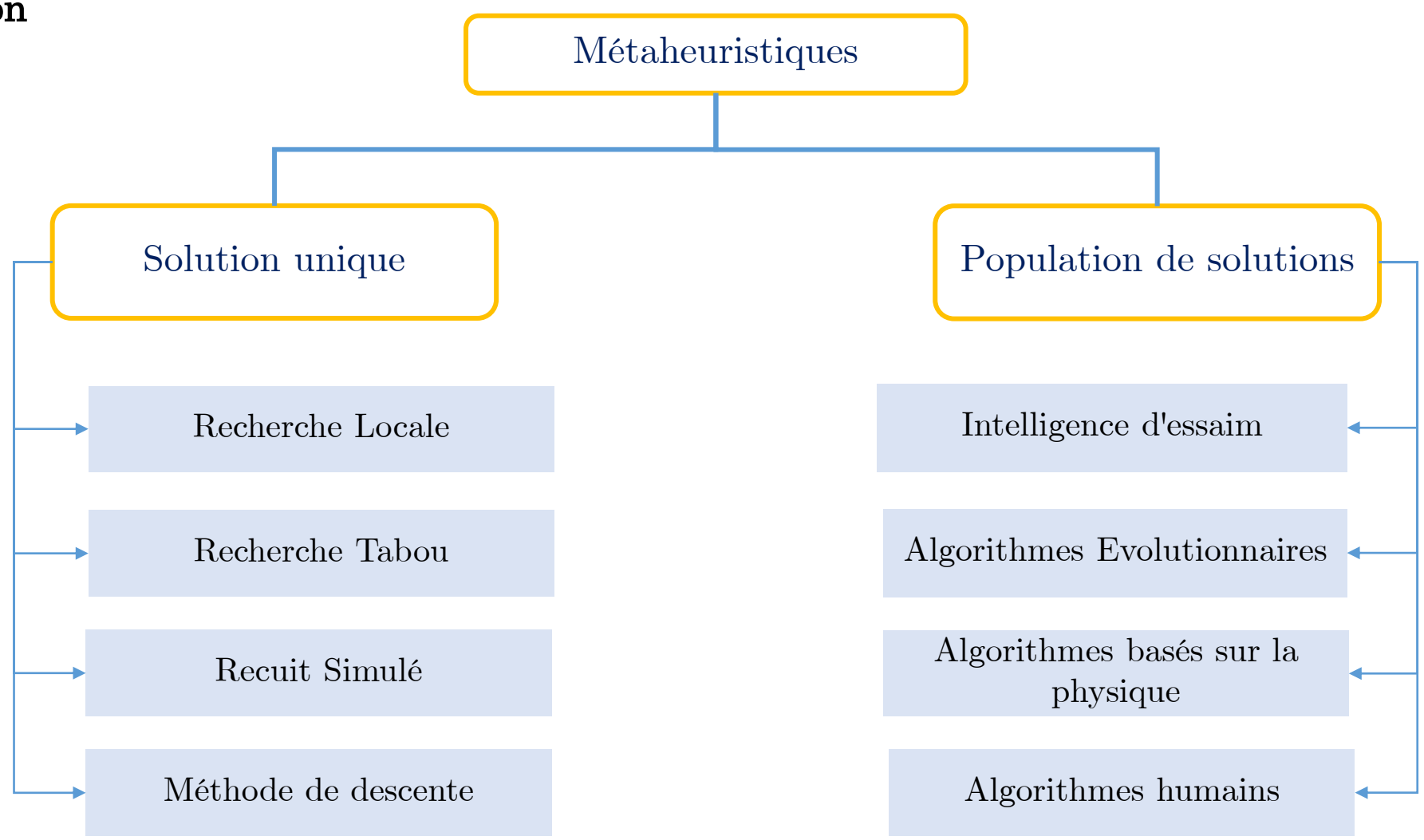
- Classification



Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Classification**



Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Intelligence d'essaim**

Algorithme	Inspiré par
Particle Swarm Optimization (PSO)	Les comportements naturels des particules d'essaim
Ant Colony Optimization (ACO)	Les fourmis déposent des phéromones sur le sol
Artificial Bee Colony (ABC)	Les comportements de la colonie d'abeilles mellifères
Firefly Algorithm (FA)	Lumière clignotante de lucioles dans les océans
Cuckoo Search (CS)	Le comportement du parasitisme d'élevage de coucou
Whale Optimization Algorithm (WOA)	Le comportement des baleines à bosse
Salp Swarm Algorithm (SSA)	Le comportement des salpes naviguant dans les océans
Moth Flame Optimization (MFO)	La méthode de navigation des papillons dans la nature
Lion Optimization Algorithm (LOA)	Mode de vie et coopération des lions
Marine Predators Algorithm (MPA)	Stratégie de recherche de nourriture des prédateurs dans les océans

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Algorithmes Evolutionnaires**

Algorithme	Inspiré par
Squirrel Search Algorithm (SSA)	Le comportement des écureuils volants du sud
Grasshopper Optimization Algorithm (GOA)	Le comportement des essaims de sauterelles
Golden Eagle Optimizer (GEO)	Le comportement des aigles royaux en vitesse de réglage
Genetic Algorithm (GA)	Théorie darwinienne de l'évolution
Differential evolution (DE)	Le phénomène naturel de l'évolution
Biogeography-Based Optimizer (BBO)	Biogéographie liée à la migration des espèces
Invasive Tumor Growth (ITG)	Processus rénal
Tree Growth Algorithm (TGA)	Concours d'arbres pour l'acquisition de nourriture et de lumière
Arithmetic Optimization Algorithm (AOA)	Le comportement de distribution de l'opérateur arithmétique principal

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Algorithmes basé sur la physique**

Algorithme	Inspiré par
Big Bang-Big Crunch (BBBC)	L'évolution de l'univers
Gravitational Search Algorithm (GSA)	La loi de la gravité et les interactions de masse
Multi-verse Optimizer (MVO)	Théorie des multivers
Central Force Optimization (CFO)	La métaphore de la cinématique gravitationnelle
Henry Gas Solubility Optimization (HGSO)	Le comportement de la loi de Henry
Thermal Exchange Optimization (TEO)	Loi de refroidissement de Newton
Electromagnetic Field Optimization (EFO)	Le comportement des électroaimants

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

- **Algorithmes humains**

Algorithme	Inspiré par
Teaching learning based Optimization (TLBO)	L'influence d'un enseignant sur la production des apprenants
Collective Decision Optimization (CSO)	Caractéristique de prise de décision humaine
Socio Evolution & Learning Optimization Algorithm (SELOA)	Comportement d'apprentissage social des humains

Méthodes de résolution approchées

❑ Méthodes Méta-heuristiques

Le choix de la métaheuristique à appliquer pour optimiser un problème donné dépendra de plusieurs facteurs :

- Type du problème d'optimisation
- Caractéristiques du problème
- Objectif voulu
- … etc.

❑ Particle Swarm Optimization (PSO)

- L'algorithme PSO est conçu pour résoudre des problèmes d'optimisation en simulant le comportement collectif d'un essaim (par exemple, une nuée d'oiseaux ou un banc de poissons). Il repose sur l'interaction entre les particules pour rechercher une solution optimale dans un espace multi-dimensionnel.
- L'information locale et la mémoire de chaque individu sont utilisées pour décider de son déplacement. Des règles simples, telles que « rester proche des autres individus », « aller dans une même direction » ou « aller à la même vitesse », suffisent pour maintenir la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs.

❑ Particle Swarm Optimization (PSO)

- **Principe général**

1.Particules et espace de recherche :

- Les particules représentent des solutions potentielles.
- Chaque particule se déplace dans l'espace de recherche en fonction de sa position et de sa vitesse.

2.Interaction entre particules :

- Chaque particule ajuste sa position en fonction :
  - ✓ De sa meilleure position connue (**pBest**, meilleure solution personnelle).
  - ✓ De la meilleure position trouvée par l'ensemble de l'essaim (**gBest**, meilleure solution globale).

3.Critère d'arrêt :

- L'algorithme s'arrête après un certain nombre d'itérations ou lorsqu'une condition de convergence est atteinte (par exemple, la solution ne s'améliore plus significativement).

❑ Particle Swarm Optimization (PSO)

- **Étapes principales de l'algorithme PSO :**

1. Initialisation :

- Générer aléatoirement les positions initiales (xi) et les vitesses (vi) des particules dans l'espace de recherche.
- Calculer les valeurs de la fonction objectif pour chaque particule.
- Définir :
  - ✓ pBest : Meilleure position connue de chaque particule (initialement sa position actuelle).
  - ✓ gBest : Meilleure position trouvée par l'ensemble des particules.

2. Mise à jour des vitesses et positions :

À chaque itération, pour chaque particule i :

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

w : Poids d'inertie, qui contrôle l'impact de la vitesse précédente.

❑ Particle Swarm Optimization (PSO)

- c1,c2 : Coefficients d'attraction vers pBest (meilleure solution personnelle) et gBest (meilleure solution globale).
- r1, r2 : Valeurs aléatoires uniformes entre 0 et 1 pour introduire de la stochasticité.

Ensuite, la position de la particule est mise à jour :

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

3. Évaluation et mise à jour :

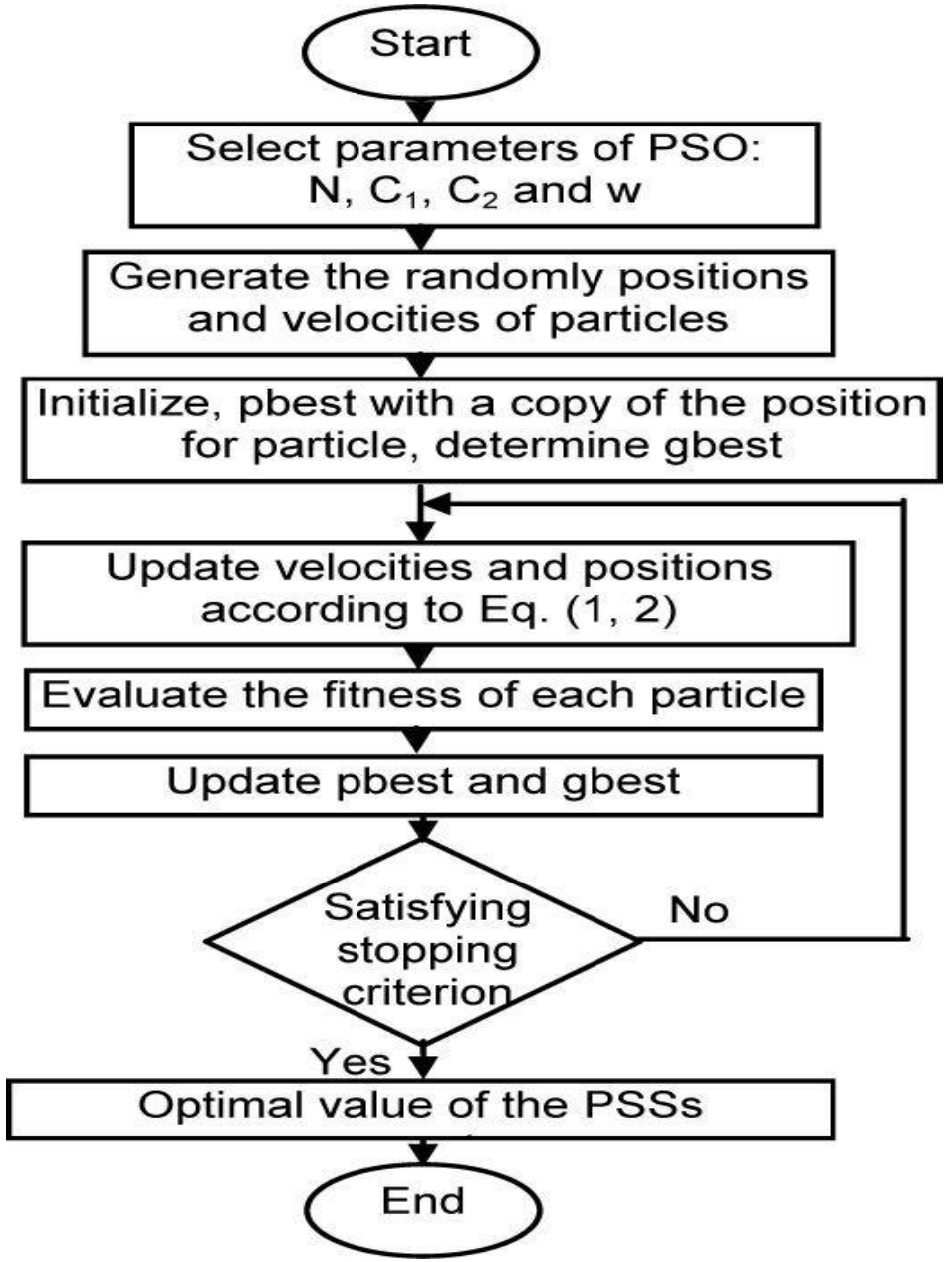
- Calculer la nouvelle valeur de la fonction objectif pour chaque particule.
- Si la position actuelle est meilleure que pBest, mettre à jour pBest.
- Si une particule trouve une position meilleure que gBest, mettre à jour gBest.

4. Critère d'arrêt :

- L'algorithme s'arrête si :
  - ✓ Le nombre maximal d'itérations est atteint.
  - ✓ L'amélioration de gBest est inférieure à un seuil donné.

Méthodes Méta-heuristiques

❑ Particle Swarm Optimization (PSO)



❑ Algorithme ABC

L'algorithme **Artificial Bee Colony** (ABC) est un algorithme d'optimisation inspiré du comportement des colonies d'abeilles dans la nature. Il est utilisé pour résoudre des problèmes d'optimisation continue et discrète.

L'algorithme ABC divise les abeilles en trois catégories:

1. **Abeilles ouvrières :**
   - Elles explorent des solutions initiales dans l'espace de recherche.
   - Chaque abeille ouvrière est assignée à une source de nourriture (solution candidate).
   - Elles génèrent de nouvelles solutions en ajustant la solution actuelle.
2. **Abeilles observatrices :**
   - Elles choisissent les meilleures solutions explorées par les abeilles ouvrières en fonction de leur qualité (fitness).
3. **Abeilles éclaireuses :**
   - Elles recherchent de nouvelles solutions de manière aléatoire si une source de nourriture devient non productive.

❑ Étapes principales de l'algorithme ABC

A. Initialisation

- La population initiale de solutions X_i (appelée sources de nourriture) est générée aléatoirement dans l'espace de recherche :

$$X_i^j = LB_j + r \cdot (UB_j - LB_j), \quad \forall i \in [1, N],\ j \in [1, D]$$

où :

- X_i^j est la j-ème variable de la i-ème solution.
- LB_j et UB_j sont respectivement les bornes inférieure et supérieure de la j-ème variable.
- r est un nombre aléatoire dans $[0, 1]$.
- N est la taille de la population (nombre de sources).
- D est le nombre de variables du problème.

❑ Étapes principales de l'algorithme ABC

B. Phase des abeilles ouvrières

Chaque abeille ouvrière génère une nouvelle solution V_i autour de la solution actuelle X_i en utilisant la formule suivante :

$$V_i^j = X_i^j + \phi \cdot (X_i^j - X_k^j)$$

où :

- X_k est une solution choisie aléatoirement telle que $k \neq i$.
- ϕ est un paramètre aléatoire dans $[-1, 1]$.

La **nouvelle solution** V_i est évaluée en utilisant la fonction objectif $f(X)$.

- Si $f(V_i)$ est meilleure que $f(X_i)$, alors X_i est remplacée par V_i.
- Sinon, la solution actuelle est conservée.

❑ Étapes principales de l'algorithme ABC

C. Calcul des fitness

La qualité (fitness) d'une solution X_i est définie par :

$$\text{fitness}_i = \begin{cases} \frac{1}{1+f(X_i)} & \text{si } f(X_i) \geq 0 \\ 1 + |f(X_i)| & \text{si } f(X_i) < 0 \end{cases}$$

D. Phase des abeilles observatrices

Si une solution X_i ne s'améliore pas après un certain nombre d'itérations (limite), elle est abandonnée. Une **nouvelle solution** aléatoire est générée :

$$X_i^j = LB_j + r \cdot (UB_j - LB_j)$$

E. Critère d'arrêt

L'algorithme s'arrête lorsqu'un des critères suivants est satisfait :

1. Le nombre maximal d'itérations est atteint.
2. Une solution optimale ou satisfaisante est trouvée.

❑ Algorithme de recuit simulé

L'algorithme de **recuit simulé** est une méthode d'optimisation stochastique inspirée du processus physique de refroidissement des métaux. Sa modélisation mathématique repose sur une exploration probabiliste de l'espace de recherche.

Voici une description détaillée et formalisée de sa modélisation mathématique :

- **Éléments du recuit simulé**
  - Température : Une variable T > 0, appelée température, régule la probabilité d'accepter des solutions sous-optimales. Elle décroît progressivement au cours de l'algorithme.
  - Solution courante : x_courant : La solution actuellement considérée.
  - Solution candidate : x_nouveau : Une solution générée aléatoirement à partir de x_courant.
  - Fonction d'acceptation : La probabilité d'accepter une solution candidate moins bonne est donnée par :

$$P(x_{\text{courant}}, x_{\text{nouveau}}, T) = \exp\left(-\frac{\Delta E}{T}\right),$$

où $\Delta E = f(x_{\text{nouveau}}) - f(x_{\text{courant}})$.

❑ Algorithme et formalisation mathématique

1. Initialisation :
   - Choisir une solution initiale x_courant.
   - Fixer une température initiale T_0 et une température finale T_finale.
   - Définir une fonction de refroidissement $T_{k+1} = \alpha T_k$, où $\alpha \in (0, 1)$ est le facteur de refroidissement.
2. Itérations principales :

   Pour chaque température T_k:
   - Générer une solution candidate : x_nouveau dans le voisinage de x_courant.
   - Calculer le changement d'énergie : $\Delta E = f(x\_nouveau) - f(x\_courant)$.
   - Critère d'acceptation :
     - Si $\Delta E \leq 0$, accepter x_nouveau : x_courant = x_nouveau.
     - Sinon, accepter x_nouveau avec une probabilité : $P = \exp(-\Delta E / T_k)$.
   - Mettre à jour la meilleure solution :
     - Si f(x_courant) < f(x_meilleur), mettre à jour : x_meilleur = x_courant.

❑ Algorithme de recuit simulé

3. Refroidissement : Réduire la température selon : $T_{K+1} = \alpha T_k$.
4. Critère d'arrêt :
   - L'algorithme s'arrête lorsque $T_k < T_{finale}$ ou après un nombre maximal d'itérations.

- **Fonction de refroidissement**

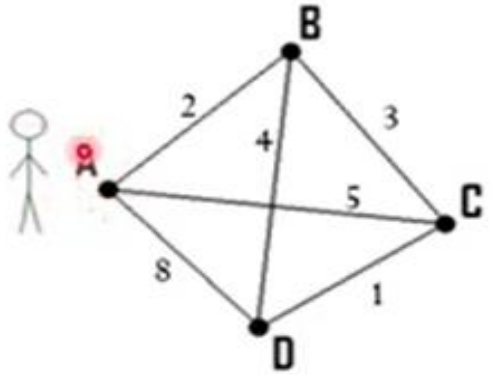
La température T_k joue un rôle crucial dans l'algorithme. Différentes fonctions de refroidissement peuvent être utilisées :

Refroidissement géométrique : $T_{k+1} = T_k - \delta, \quad \delta > 0.$

Refroidissement logarithmique : $T_{k+1} = \alpha T_k, \quad \text{avec } \alpha \in (0,1)$

Refroidissement linéaire : $T_k = \frac{T_0}{1 + \beta \log(1+k)}, \quad \beta > 0.$

❑ Problème du Voyageur de Commerce « Travel Salesman Problem (TSP) »



$$
\begin{cases}
\text{Min } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & \\
\sum_{j=1}^{n} x_{ij} = 1 & \forall i & (1) \\
\sum_{i=1}^{n} x_{ij} = 1 & \forall j & (2) \\
\sum_{i\in Q}\sum_{j\in Q} x_{ij} \geq 1 & \forall Q & (3) \\
x_{ij} \in \{0,1\} & \forall i, \forall j &
\end{cases}
$$

Z : Distance totale ou coût total que l'on souhaite minimiser.

c_{ij} : Coût ou distance associé au chemin allant de la ville i à la ville j.

x_{ij} : Variable binaire indiquant si le chemin entre la ville i et la ville j est emprunté :

- $x_{ij} = 1$ si le chemin est emprunté ;
- $x_{ij} = 0$ sinon.

La fonction objectif cherche à minimiser la distance ou le coût total pour parcourir toutes les villes une seule fois, tout en retournant à la ville de départ.

❑ Problème du Voyageur de Commerce « Travel Salesman Problem (TSP) »

Contraintes :

1. **Chaque ville est quittée exactement une fois :**

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in \{1, 2, \ldots, n\}.$$

2. **Chaque ville est visitée exactement une fois :**

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \in \{1, 2, \ldots, n\}.$$

3. **Pas de sous-tours (Subtour elimination constraints) :**

   - Les sous-tours (cycles partiels n'incluant pas toutes les villes) doivent être éliminés. Une contrainte courante est donnée par :

$$\sum_{i \in Q} \sum_{j \notin Q} x_{ij} \geq 1, \quad \forall Q \subset \{1, 2, \ldots, n\}.$$

4. **Binarité des variables :**

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in \{1, 2, \ldots, n\}.$$

❑ Problème du Voyageur de Commerce « Travel Salesman Problem (TSP) »

Applications réelles du TSP

- **Logistique :**
  - Optimisation des itinéraires de livraison pour réduire les coûts et le temps.
- **Planification touristique :**
  - Trouver le meilleur itinéraire pour visiter plusieurs destinations.
- **Industrie manufacturière :**
  - Planification des mouvements des bras robotiques.
- **Conception de circuits électroniques :**
  - Optimisation des connexions entre composants.