* 1. Introduction

L'optimisation combinatoire est un outil indispensable combinant diverses techniques des mathématiques discrètes et de l'informatique afin de résoudre des problèmes de la vie réelle. En effet, l'optimisation combinatoire, couvre aussi bien des domaines de l’ingénieur que les domaines de la recherche, et englobe un large éventail de techniques et fait toujours l'objet de recherches intensives. D’une manière simple, résoudre un problème d’optimisation combinatoire consiste à trouver l’optimum d’une fonction, parmi un nombre fini de choix, souvent très grand. Il s’agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objectif sous certaines contraintes. Le but est de trouver une solution optimale dans un temps d’exécution raisonnable. Néanmoins, ce but est loin d’être concrétisé pour plusieurs problèmes vu la leurs complexités grandissantes. La théorie de la NP-complétude a permis de classifier les problèmes d’optimisation selon leurs complexités et elle fournit des informations pertinentes sur le genre de méthodes à choisir en fonction de la difficulté intrinsèque des problèmes. Lorsqu’une solution est associée à une seule valeur, on parle de problèmes mono-objectifs, et lorsqu’elle est associée à plusieurs valeurs, on parle de problèmes multi-objectifs (ou multi-critères). Il faut noter que, l’optimisation d’un problème multi-obectif est souvent plus difficile que l’optimisation des problèmes mono-obectifs. En effet, L’optimisation multi-objectif permet de modéliser des problèmes réels faisant concourir de nombreux critères (souvent conflictuels) et contraintes. Dans ce contexte, la solution optimale recherchée n’est plus un simple point, mais un ensemble de bons compromis satisfaisant toutes les contraintes. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement pénibles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent pas encore de solutions algorithmiques efficaces et acceptables pour toutes les données. Les méthodes de l'optimisation combinatoire peuvent être classées en méthodes heuristiques et méthodes exactes (figure 1.1).

Les algorithmes exacts sont utilisés pour trouver au moins une solution optimale d’un problème. Les algorithmes exacts les plus réussis dans la littérature appartiennent aux paradigmes de la programmation dynamique, de la programmation linéaire en nombres entiers, ou des méthodes de recherche arborescente (Branch & Bound). Généralement, pour un problème donné, ces algorithmes procèdent par énumération de toutes les solutions possibles afin de trouver la meilleure solution. Lorsque les ampleurs du problème deviennent importantes, une énumération explicite s'avère irréalisable. Pour cela, On a recours aux procédures classiques par séparation et évaluation ("Branch and Bound"). Ces algorithmes réalisent une construction partielle d'un arbre de recherche des solutions en essayant de trouver une solution exacte pour un problème donné. Bien que les méthodes exactes trouvent des solutions optimales, leur grand inconvénient est l’explosion combinatoire ce qui rend leur utilisation pratique difficile.

D’un autre coté, Une méthode heuristique ou approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantie d'optimalité. L’avantage principale de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, bien ou mal formulés, avec ou sans contrainte. En plus, elles ne nécessitent pas une spécification mathématique du problème. D’un autre coté, Les algorithmes heuristiques tels que les algorithmes de recuit simulé, les algorithmes tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d’optimisation combinatoires.

Il faut noter qu’il n’existe pas une méthode sésame qui peut résoudre tous les problèmes, parfois c’est intéressant de faire hybrider plusieurs algorithmes de résolution au sein d’un seul algorithme afin de résoudre des problèmes très compliqués. Indépendamment de l’approche, une bonne méthode d’optimisation doit considérer les points suivants :

**• robustesse :** la méthode devrait fonctionner pour une large classe de problèmes, indépendamment des paramètres et des valeurs initiales,

**• efficacité:** elle ne devrait pas être prohibitivement chère en temps CPU et en mémoire vive,

**• précision:** le résultat numérique devrait bien approcher la vraie solution du problème de minimisation

**Algorithmes Heuristiques**

**Programmation linéaire**

**Programmation dynamique**

Recuit simulé

Hill Climbing

Recherche tabou

Colonies de fourmis

Algorithmes génétiques

Solution Unique

Population

**Branch and \***

Simplex

A\*

**Heuristiques Spécifiques**

**Métaheuristique**

**Algorithmes Exacts**

Figure 1.1 ‒ Taxonomie des méthodes d’optimisation combinatoire.

* 1. La théorie de la complexité

La théorie de la complexité consiste à estimer la difficulté ou la complexité d'une solution algorithmique d’un problème posé de façon mathématique. Elle se concentre sur les problèmes de décision qui posent la question de l’existence d’une solution comme le problème de satisfiabilité booléenne. La théorie de la complexité repose sur la notion de classes de complexité qui permettent de classer les problèmes en fonction de la complexité des algorithmes utilisés pour les résoudre. Parmi les classes les plus courantes, on distingue:

* La *classe P* (Polynomial time) qui englobe les problèmes pour lesquels il existe un algorithme déterministe de résolution en temps polynomial,
* La *classe NP* (Nondeterministic Polynomial time) qui contient des problèmes de décision pour lesquels la réponse *oui* peut être décidée par un algorithme non-déterministe en un temps polynomial par rapport à la taille de l’instance. Les problèmes NP-complets sont définis comme suit:

***Définition 1.1 (Problème NP-complet)*** *Un problème de décision п est NP-complet s’il satisfait les deux conditions suivantes : п ∈ NP, et tout problème NP se réduit à п en temps polynomial.*

L’une des questions ouvertes les plus fondamentales en informatique théorique est vraisemblablement la question si « P=NP ? » (Figure 1.2). Ceci revient à trouver un algorithme polynomial pouvant résoudre un problème NP-complet. Trouver un tel algorithme, pour un seul problème appartenant à la classe NP-complet, signifierait que tous les problèmes de cette classe pourraient être résolus en temps polynomial (voir Définition 1.1) et en conséquence, que P=NP. Cependant, il est commun de penser que P≠NP, mais aucune preuve n’a encore été trouvée jusqu’à aujourd’hui.

Il est important de préciser que tous les problèmes d’optimisation ne peuvent pas être classifiés comme problèmes NP-complets, puisqu’ils ne sont pas tous des problèmes de décision, même si pour chaque problème d’optimisation on peut définir un problème de décision qui a une complexité équivalente.

**Définition 1.2 (Problème NP-difficile)** *Un problème P* *quelconque (de décision ou non) est NP-difficile si et seulement s’il existe un problème NP-complet* P′ *qui est réductible à lui polynomialement.*

La définition d’un problème NP-difficile est donc moins étroite que celle de la NPcomplétude. De cette définition on peut observer que pour montrer qu’un problème d’optimisation est NP-difficile, il suffit de montrer que le problème de décision associé à lui est NP-complet. De cette façon un grand nombre de problèmes d’optimisation ont été prouvés NP-difficiles. C’est notamment le cas des problèmes du Voyageur de Commerce, de Partitionnement de Graphes et d’Affectation Quadratique.

P≠NP

P=NP

NP-difficile

NP-difficile

**Figure.1.2** **‒** classe P, NP, NP-complet, NP-difficile.

* 1. Formulation mathématique des problèmes d’optimisation

Les problèmes d’optimisation combinatoire peuvent être formulés comme suit :

Définition 1.3 (Problèmes mono-objectifs) : *Un problème d’optimisation est généralement formulé comme un problème de minimisation ou de maximisation, et écrit sous
la forme suivant:*

*minx f(x), Tel que,*

*gi(x) ≤0 , i = 1, . . . ,m ,* (1.1)

*hj(x) = 0 , j = 1, . . . , p ,*

*x ∈ S ⊂ Rn,*

Où *f* est la fonction (scalaire) à minimiser, appelée fonction coût ou fonction objectif, *x* représente le vecteur des variables d’optimisation, *gi* sont les contraintes d’inégalité et *hj* les contraintes d’égalité, et *S* est l’espace des variables (appelé aussi espace de recherche). S indique quel type de variables sont considérées : réelles, entières, mixtes (réelles et entières dans un même problème), discrètes, continues, bornées, etc.

Un point *xA* est appelé un point admissible si *xA ∈ S* et si les contraintes d’optimisation sont satisfaites : *gi(xA) ≤0 , i = 1, . . . ,m et hj(xA) = 0 , j = 1, . . . , p*. La solution de (1.1) est l’ensemble des optima *{x\*}.*

*x\**est un minimum global de *f* si et seulement si *f(x\*)≤f(x) ∀ x ∈ S*, et *x\**est un minimum local de f si et seulement *si f(x\*)≤ f(x) ∀ x ∈ S / ||x − x\*|| ≤ ɛ, ɛ > 0*. La Figure 1.3 présente un exemple d’une fonction à une variable, avec des minima locaux et un minimum global. Parmi les minima locaux, celui qui possède la plus petite valeur de f est le minimum global. Par ailleurs, une fonction multimodale présente plusieurs minima (locaux et globaux), et une fonction unimodale n’a qu’un minimum, le minimum global. La Figure 1.4 montre une fonction multimodale à deux variables.



**Figure 1.3 ‒**  Minima locaux et minimum global d’une fonction à une variable.



**Figure 1.4 ‒** Une fonction multimodale à deux variables.

Définition 1.4 (Problèmes multi-objectifs) : *D’un point de vue mathématique, un problème d’optimisation multi-objectif, se présente, dans le cas où le vecteur  regroupe k fonctions objectif, de la façon suivante:*

*minx , Tel que,*

*≤0 , i = 1, . . . ,m ,* (1.2)

* = 0 , j = 1, . . . , p , x ∈ S ⊂ Rn,*

De ce fait, résoudre un problème d’optimisation combinatoire nécessite l’étude de trois points suivants :

– la définition de l’ensemble des solutions réalisables,

– l’expression de l’objectif à optimiser,

– le choix de la méthode d’optimisation (exacte ou approchée) à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution.

**1.4 Les méthodes de résolutions des problèmes d’optimisation combinatoire**

Les techniques pour résoudre les problèmes mathématiques dépendent de la nature de la fonction objectif et l'ensemble des contraintes. Les sous-domaines majeurs suivants existent :

**La programmation linéaire** étudie les cas où la fonction objectif et les contraintes sont linéaires.

**La programmation linéaire en nombres entiers** étudie les programmes linéaires dans lesquels certaines ou toutes les variables sont contraintes à prendre des valeurs entières.

**La programmation quadratique** concerne les problèmes dont la fonction objectif contient des termes quadratiques ( ex :f(x)= x2+x), tout en conservant les contraintes linaires .

**La programmation non-linéaire** étudie le cas général dans lequel la fonction objectif ou les contraintes (ou les deux) contiennent des parties non-linéaires.

 **La programmation stochastique** concerne les problèmes avec des contraintes dépendant de variables aléatoires.

**La programmation dynamique** ce type de méthodes est utilisé dans le cas où les le problème est décomposables en petites entités facilement résolubles. Elle n’est utilisable que lorsque la fonction objectif est monotone croissante. De ce fait, la résolution d’un problème en programmation dynamique est basée sur une décomposition du problème en sous-problèmes plus simples. A chaque sous-problème correspond un ensemble d’options, représentant chacune un coût en terme de fonction objectif. Un ensemble de choix doit donc être effectué pour les différents sous-problèmes dans le but d’arriver à une solution optimale.

**1.5 Notions de base en optimisation combinatoire**

**Définition 1.5** L’ensemble S est **convexe**, si quels que soient les deux points P, Q ∈S, tout le segment PQ ∈ S.

**Définition 1.6** Une fonction à deux variables est dite **convexe** sur un ensemble convexe si et seulement si tout segment de droite reliant deux points sur la surface de  est toujours *au-dessus* de cette surface.

**Définition 1.7** Une fonctionà deux variables est dite **concave** sur un ensemble convexe si et seulement si tout segment de droite reliant deux points sur la surface de  est toujours *au-dessous* de cette surface.

**Définition 1.8** Un programme mathématique est dit **convexe** s’il s’agit de la **minimisation d’une fonction convexe** sur une région réalisable convexe, soit de la **maximisation d’une fonction concave** sur une région réalisable convexe.

**

**Figure 1.5‒** exemple de fonction convexe.

**

**Figure 1.6‒** exemple de fonction concave.

**Définition 1.9 (Voisinage et** **mouvement)** : *Soit une solution s, on dit que s\* est une solution voisine de s, si on peut obtenir s\* en modifiant légèrement s. On dit aussi qu’on peut passer de s à s\* en effectuant un mouvement. Le voisinage V(s) de s est l’ensemble des solutions voisines de s. La fonction V : S →S 2,∀s ∈ S, V(s)⊂ S*$V: S\rightarrow S²,∀s\in S,v\left(s\right)⊂S$ *est associe à chaque point de S un sous-ensemble de S* .

**Définition 1.10 (Optimum local)***:* soit S un ensemble convexe, f est une fonction définie sur S. *Une solution s ∈ S est un optimum local si et seulement si il n’existe pas de solution s0 ∈ V(s)* $s0\in v(s)$ *dont l’évaluation est de meilleure qualité que s, soit :*

$ ∀s0\in v(s)$ *∀ s0 ∈ v(s) f(s) ≤ f(s0 )* $f(s)\leq f(s0)$ *Dans le cas d’un problème de minimisation*

 *f(s) ≥ f(s0)* $f(s)\geq f(s0)$ *Dans le cas d’un problème de maximisation*

*Avec V (s) l’ensemble des solutions voisines de s.*

**Définition 1.11 (Optimum global)** : *Une solution est un optimum global à un problème d’optimisation s’il n’existe pas d’autres solutions de meilleure qualité. La solution s\** ∈ S *est un optimum global si et seulement si:*

 *∀ s∈ S* $ ∀s\in S$ *f(s\*) ≤ f(s)* $f(s)\leq f(s0)$$f(s\*)\leq f(s)$ *Dans le cas d’un problème de minimisation*

$f(s\*)\geq f(s)$ *f(s\*) ≥ f(s) Dans le cas d’un problème de maximisation*

Donc, l’optimum global est la solution s\* qui vérifie la propriété précédente pour toutes les structures de voisinage du problème. La figure 1.9 schématise la courbe d’une fonction d’évaluation en faisant apparaître l’optimum global dans le cas d’un problème de minimisation.



**Figure 1.7 ‒** L’optimum global et local.

Du point de vue pratique, les définitions des minima ne sont pas très utiles, pour s’assurer que s0 est bien un minimum local, il faudrait parcourir tous les s dans le voisinage de s0 ce qui est trop coûteux.

**1.5 Méthodes mathématiques pour résoudre les problèmes d’optimisations non linéaires et sans contraintes**

Pour les fonctions dérivables deux fois, des problèmes sans contraintes peuvent être résolus en trouvant les points où le gradient de la fonction est 0 et en utilisant la matrice hessienne pour classer la nature de point. Si le hessien est défini positif, le point est un minimum local ; s’il est un défini négatif, un maximum local.

Si la fonction est convexe sur l’ensemble des solutions admissibles alors tout minimum local est aussi un minimum global.

Rappel mathématiques 1: le gradient d’une fonction f à n variables est un vecteur
n-dimensionnel contenant toutes les dérivées partielles de f par rapport à chacune de ses variables



Si  pour le point (x1’, x2’,.., xn’) alors ce point est minimum (maximum) local. Ce point est appelé également point stationnaire.

**Rappel mathématiques 2:** Le **Hessien** est une façon d’écrire les dérivées d’un gradient à l'aide d’une matrice symétrique. pour une fonction f(x1,x2,…,xn), le Hessein est calculé en utilisant la formule suivante :

 pour i=1..n,j=1..n.



Pour le cas d’une fonction à deux variables : Soit une fonction deux fois dérivable, alors le Hessien est :

**.**

On a toujours.

Exemple où .

**Calculer le gradient de f** : ; .

**Calculer le Hessien de f** : ; ;

; .

Le Hessien est  et dans cet exemple, il est le même (constant) en tout point du domaine de, ce qui n’est pas toujours le cas.

* Soit, le **déterminant** de *A* est donné par .
* Une matrice A est **symétrique** si *b* = *c* ; le Hessien est toujours symétrique.

**Définitions**

Soit A une matrice n\*n symétrique et X un vecteur de Rn XT est le transposé de X.

* *A* est **définie positive** si et seulement si XT.A.X > 0 ∀ X ∈ S

 Pour le cas d’une fonction f à deux variables il faut vérifier seulement 

* *A* est **définie négative** si et seulement si XT.A.X < 0

Pour le cas d’une fonction f à deux variables il faut vérifier seulement 

* *A* est **semi-définie positive** si et seulement si XT.A.X ≥ 0

Pour le cas d’une fonction f à deux variables il faut vérifier seulement 

* *A* est **semi-définie négative** si et seulement si XT.A.X ≤ 0

Pour le cas d’une fonction f à deux variables il faut vérifier seulement 

* Note :signifie que le déterminant  peut s'annuler pour une ou plusieurs valeurs de (*x,y*).

**Exemples**

* ;  est non définie.
*   est définie négative.
*   est semi–définie positive.
*   est semi–définie positive*.*
* 
*  est à la fois semi–définie positive et semi-définie négative; c’est en fait le Hessien d’une fonction linéaire.

**Théorème : Soit  un point stationnaire de la fonction*.***

**Si le Hessien de *f* est défini positif en, alorsest un *minimum local*.**

**Si le Hessien de *f* est défini négatif en, alorsest un *maximum* *local*.**

**Si , alorsestun *point singulier* .**

**Si****, il faut utiliser d'autres techniques basées sur les dérivées d’ordre supérieur pour pouvoir se prononcer.**

**Exemple** .

1. Donner le **gradient**  et le **Hessien** .

Identifier la nature des points stationnaires de la fonction .

1. **; .**
2. **.**

Les points stationnaires sont : (0, 0) (1/2, 1/2) (2, -1).





 

**Exemple  Un problème de marketing**

La fonction de profit est 

et le seul point stationnaire est .

Quelle est la nature de ce point ?

Puisque , le Hessien est .

 avec *a* = –2 et *b* = –4.  est *défini négatif*. est donc un *maximum local*.

Pour une fonction convexe ou concave, la condition du premier ordre (gradient nul) est nécessaire et suffisante pour caractériser la nature d’un point stationnaire.

|  |
| --- |
| **Théorème : Soit  un point stationnaire de la fonction *.*****Si  est convexe, alors est un minimum absolu (global).****Si  est concave, alors est un maximum absolu (global).** |

Si la fonction  est deux fois dérivable, on peut la caractériser par le Hessien :

|  |
| --- |
| **Théorème : Soit  dérivable deux fois sur un ensemble** *E****.*** **est convexe si et seulement si** **est semi-défini positif .** **est concave si et seulement si**  **est semi-défini négatif .****Si   (H est défini), alors on dira que la fonction  est** *strictement* **convexe ou** *strictement* **concave.** |

**Exemple**.

.

  et 1/*x* > 0, ∀(*x*, *y*).

 est défini positif ∀(*x*, *y*), donc  est*strictement* **convexe.**

**Exemple**  .

Identifier la nature des points stationnaires de la fonction.

; 

 (1 , 1/2) est le seul point stationnaire de .

.  avec *a* = 8, ∀(*x*, *y*).

Le Hessien  est défini positif ∀(*x*, *y*), donc  est*strictement* **convexe.**

Par conséquent, = (1, 1/2) est un *minimum absolu*.

**Exemple  Problème de marketing**  (suite)

On a déjà vu que l’unique point stationnaire  est un maximum local. Mais il est possible d’avoir une information plus complète sur ce point stationnaire, puisque le Hessien  est le même partout.

 avec *a* = –2 et *b* = –4.  est *défini négatif* pour tout point  et donc  est *strictement* **concave**. Par conséquent,  est un *maximum absolu*.

**Propriétés des fonctions convexes et concaves**

Si  et  sont convexes (concaves), alors + est aussi convexe (concave).

Si *f* est convexe (concave), alors est convexe (concave), .

Si *f* est convexe (concave) alors *–f* est concave (convexe).

 est dite une **fonction affine** sur ; elle est à la fois convexe et concave parce que son Hessien est nul et qu’il est alors à la fois semi-défini positif et semi-défini négatif.

**Exercice**  Soit  Étudier la convexité de :

; ; .